ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ
ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS

School of Information Sciences and Technology
Department of Informatics
Athens, Greece

Bachelor Thesis
in
Computer Science

Leveraging Retrieval-Augmented Generation for Student Support: A Document-Centric QA System for the AUEB Informatics Studies Guide

Nikos Mitsakis

Supervisors: Prof. Ion Androutsopoulos
Department of Informatics
Athens University of Economics and Business

Assoc. Prof. Themos Stafylakis
Department of Informatics
Athens University of Economics and Business

July 2025

Nikos Mitsakis

Leveraging Retrieval-Augmented Generation for Student Support: A Document-Centric QA System for the AUEB Informatics Studies Guide

July 2025

Supervisors: Prof. Ion Androutsopoulos, Associate Prof. Themos Stafylakis

Athens University of Economics and Business
School of Information Sciences and Technology
Department of Informatics
Information Processing Laboratory,
Natural Language Processing Group
Athens, Greece

Abstract

This thesis examines the design, development, and evaluation of a Retrieval-Augmented Generation (RAG) system specifically designed to support undergraduate students in the Department of Informatics at the Athens University of Economics and Business (AUEB). The central objective is to create a cost-effective yet high-quality AI assistant capable of answering studies guide-related questions, ensuring that all responses are explicitly grounded in the latest edition of the department's official Studies Guide. To achieve this, the system ingests the newest version of the Studies Guide. It represents its contents at three levels of granularity: chunks (bodies of text corresponding to paragraphs or groups of paragraphs on a specific topic, based on the document's structure), sentences (extracted by sentence tokenizing each chunk), and propositions (decontextualized factual statements synthetically generated from the chunks). The retrieval architecture explores traditional lexical search (BM25), dense vector search, and a hybrid ensemble retriever to maximize retrieval coverage and relevance. Question-answering capabilities are assessed using both real-world and synthetic QA pairs, with the generation module leveraging self-hosted state-of-the-art large language models (LLMs). The thesis conducts a comprehensive evaluation across all document granularities and retrieval configurations, employing both classical information retrieval metrics and more modern LLM-based evaluation. Results demonstrate the feasibility of delivering a factual, responsive, and modular assistant using modest computational resources. The thesis further discusses the limitations and potential extensions of the approach, aiming to provide a blueprint for deploying similar RAG-based assistants in other academic settings.

Keywords
Retrieval-Augmented Generation; Information Retrieval; Question Answering; Large Language Models; Natural Language Processing; Document Indexing; Prompt Engineering.

Περίληψη

Η παρούσα πτυχιακή εργασία εξετάζει το σχεδιασμό, την ανάπτυξη και την αξιολόγηση ενός συστήματος τύπου Retrieval-Augmented Generation (RAG), ειδικά διαμορφωμένου ώστε να υποστηρίζει τους προπτυχιακούς φοιτητές του Τμήματος Πληροφορικής του Οικονομικού Πανεπιστημίου Αθηνών (ΟΠΑ). Ο κεντρικός στόχος είναι η δημιουργία ενός οικονομικά προσιτού αλλά υψηλής ποιότητας βοηθού τεχνητής νοημοσύνης, ικανού να απαντά σε ερωτήσεις που σχετίζονται με τις σπουδές τους, διασφαλίζοντας ότι όλες οι απαντήσεις είναι τεκμηριωμένες αποκλειστικά βάσει της τελευταίας έκδοσης του επίσημου Οδηγό Σπουδών του Τμήματος. Για το σκοπό αυτό, το σύστημα ενσωματώνει την πιο πρόσφατη έκδοση του Οδηγού Σπουδών, αναπαριστώντας το περιεχόμενό του σε τρία διαφορετικά επίπεδα: τα chunks (τμήματα κειμένου που αντιστοιχούν σε παραγράφους ή ομάδες παραγράφων με βάση τη δομή του εγγράφου), τις προτάσεις (που προκύπτουν από τον τεμαχισμό των chunks σε επιμέρους προτάσεις) και τις αποπλαισιωμένες προτάσεις (δηλώσεις γεγονότων που παράγονται συνθετικά από τα chunks, ως αυτόνομες πληροφορίες). Η αρχιτεκτονική ανάκτησης εξετάζει παραδοσιακές τεχνικές λεξικής αναζήτησης (BM25), πυκνή διανυσματική αναζήτηση και έναν υβριδικό μηχανισμό ανάκτησης, ώστε να επιτυγχάνεται η μέγιστη δυνατή κάλυψη και συνάφεια αποτελεσμάτων. Οι δυνατότητες απάντησης ερωτημάτων αξιολογούνται με βάση ζεύγη ερωτο- απαντήσεων που προέρχονται τόσο από πραγματικά όσο και από συνθετικά δεδομένα, αξιοποιώντας σύγχρονα μεγάλα γλωσσικά μοντέλα (LLMs). Η εργασία πραγματοποιεί εκτενή αξιολόγηση σε όλα τα επίπεδα αναπαράστασης του εγγράφου και για όλες τις διαφορετικές διαμορφώσεις ανάκτησης, χρησιμοποιώντας τόσο κλασικές μετρικές ανάκτησης πληροφοριών όσο και μεθόδους αυτόματης αξιολόγησης βασισμένες σε LLMs. Τα αποτελέσματα αποδεικνύουν ότι είναι εφικτή η δημιουργία ενός αξιόπιστου, γρήγορου και ευέλικτου βοηθού, αξιοποιώντας σχετικά περιορισμένους υπολογιστικούς πόρους. Τέλος, η εργασία αναφέρει τους περιορισμούς και τις δυνητικές επεκτάσεις της προτεινόμενης προσέγγισης, με στόχο να παρέχει ένα πρότυπο για την ανάπτυξη παρόμοιων βοηθών τύπου RAG και σε άλλα ακαδημαϊκά πλαίσια.

Acknowledgements

I would like to sincerely thank my supervisor, Prof. Ion Androutsopoulos, for his invaluable guidance, detailed feedback, and meaningful insights throughout the development of this thesis. His mentorship, extensive knowledge, and continuous support have profoundly shaped both the direction and depth of this work.

I am also deeply grateful to my co-supervisor, Prof. Themos Stafylakis, for his constructive feedback and insightful suggestions, which significantly contributed to the quality and refinement of this thesis.

A special and heartfelt thank you goes to Chris Vlachos, PhD student at AUEB, for his unwavering patience, availability, and openness to all my questions. His clear explanations and constant support provided clarity and confidence at every step of the research and implementation process.

I am also very grateful to Elisavet Palogiannidi for her presence in my thesis meetings, her genuine interest in my project, and her willingness to offer help in any way possible. Her support and encouragement are deeply appreciated.

Last but certainly not least, I would like to sincerely thank my family and friends for their continuous support and patience throughout my year-long research journey. Without their presence, encouragement, and understanding, none of this would have been possible. Their unwavering belief in me provided the strength and motivation to persevere during the most challenging moments, and for that I am deeply grateful.

Contents

1 Introduction

Recent advancements in Artificial Intelligence (AI) and Natural Language Processing (NLP) have fundamentally transformed the way information is accessed and utilized. A significant development in this domain is the emergence of intelligent conversational agents—often referred to as chatbots or virtual assistants—that leverage the capabilities of Large Language Models (LLMs). These systems are increasingly employed across diverse application domains, ranging from customer service automation to specialized educational support. Despite their broad applicability, building highly capable assistants often requires substantial computational resources and extensive training data, posing challenges particularly in contexts with limited resources.

In the academic environment, particularly at the undergraduate level, students frequently encounter scenarios where timely access to accurate and context-specific information is vital for their academic progression and overall educational experience. Across universities, undergraduates routinely rely on department Studies Guides, which serve as authoritative resources for curriculum structures, course syllabi, academic policies, and administrative procedures. Yet, the sheer breadth and organizational complexity of these documents can present considerable obstacles to effective information retrieval, often resulting in frustration and inefficiency. Addressing this widespread issue, the present thesis proposes a generalizable pipeline for the development of a conversational assistant designed to support undergraduate students in navigating their academic requirements. Although the implementation and evaluation focus on the Studies Guide of the Department of Informatics at the Athens University of Economics and Business (AUEB) as a representative case study, the methodology, system architecture, and insights are intended to be broadly applicable across diverse institutional contexts. By grounding all responses in up-to-date, institutionally sanctioned documentation, the proposed assistant aims to enhance student autonomy and facilitate more informed decision-making throughout the course of their studies. Furthermore, depending on the scope and nature of the ingested documents, the system could be extended to support not only enrolled students but also university personnel seeking administrative information, as well as prospective students exploring academic programs and requirements to inform their application decisions.

Motivated by these considerations, this thesis introduces **AUEBbot**, an assistant specifically designed to serve the information needs of undergraduate students at the Department

of Informatics at AUEB[1]. The primary objective of the assistant is to facilitate intuitive and efficient access to relevant, accurate, and up-to-date information derived exclusively from the official Studies Guide. By employing a Retrieval-Augmented Generation (RAG) architecture, the assistant integrates advanced NLP techniques for preprocessing textual content, extracting meaningful document representations at various granularities, and efficiently retrieving contextually relevant information. The retrieved information is then synthesized into precise natural-language responses through state-of-the-art generative AI technologies.

This thesis thoroughly explores the various methodological stages involved in developing such an assistant. These stages encompass the preprocessing and representation of the Studies Guide content at the chunk, sentence, and proposition (concise, self-contained factual units) levels, the generation and curation of both synthetic and real-world question-answer datasets to benchmark system performance, detailed architectural design decisions underpinning the retrieval and generation processes, and rigorous evaluation employing both quantitative and qualitative metrics. Special emphasis is placed on examining the practical trade-offs between system performance and computational efficiency, particularly within the resource constraints typical of an undergraduate research environment.

1.1 Motivation and Problem Statement

Undergraduate students at the Department of Informatics at AUEB, frequently need reliable access to information about curriculum planning, faculty contacts, university facilities, regulations, and administrative procedures. However, the existing search process relies heavily on fragmented sources such as PDF files, spreadsheets, or direct emails to various department secretariats, an approach that is often outdated, inefficient, and frustrating.

The motivation stems from the realization that students deserve a modern, centralized platform where they can easily ask questions and receive actionable answers in an engaging, conversational format. The proposed system would allow students to ask about campus services, curriculum structure, and administrative processes.

Aiming to fill this need in student services, this thesis aims to develop a lightweight, domain-specific RAG-powered assistant that bridges the information gap in AUEB's current ecosystem.

[1] All the code is publicly available on GitHub here: `https://github.com/NIKOMAHOS/rag_bot_AUEB`

The proposed system aims to:

- **Centralize access** to official AUEB content, with the present version limited to the Studies Guide and future expansions envisaged to cover faculty information, campus facilities, and administrative procedures.
- Provide **verifiable answers**, grounded in up-to-date univeristy documentation.
- Deliver results in an **accessible and engaging conversational interface**.
- Maintain **low resource consumption**, enabling deployment on modest infrastructure such as university servers or inexpensive cloud instances.

The overarching problem statement is therefore: "How can one design and implement a lightweight, transparent, and resource-efficient RAG-based virtual assistant that meets the real-world information-seeking needs of AUEB students while ensuring answers are accurate, verifiable, and engaging?"

1.2 Thesis Structure

This thesis is structured into 7 main chapters (including this one), outlined as follows:

Chapter 2 introduces the necessary background material and reviews related work in RAG, institutional chatbot pipelines, document chunking techniques, LLMs, synthetic data generation, and information retrieval.

Chapter 3 focuses on the ingestion and preprocessing of the AUEB Studies Guide, the sole knowledge base used in this work. It details the methodology used to extract semantically coherent paragraph-level chunks from the document, followed by their transformation into sentence-level and fully decontextualized proposition-level units. The benefits and tradeoffs of each granularity are discussed in the context of downstream retrieval and generation.

Chapter 4 explains the creation of Question Answer (QA) data for evaluation. It describes the synthetic generation of QA pairs at all three granularities using structured prompting techniques, as well as the small-scale collection of real-world student queries.

Chapter 5 describes the core assistant system design. It outlines the three retrieval setups that we experimented with: BM25-based sparse retrieval, dense vector search via FAISS,

and Hybrid retrieval through reciprocal rank fusion (RRF). It also discusses the prompt engineering and LLM configuration used for answer generation.

Chapter 6 outlines the methodology and experimental setup used to evaluate the system. It compares performance across all three document granularities, using both real and synthetically generated QA pairs and different retrievers. Results for both retrieval and response generation are presented and analyzed.

Chapter 7 concludes the thesis by summarizing its contributions, outlining limitations, and proposing future work directions. These include expanding the document collection, experimenting with other advanced retrieval tactics, deploying the assistant for practical use by AUEB students, and getting real-user feedback.

Appendix A contains auxiliary resources, including prompt templates used in different stages of the pipeline.

2 Background and Related Work

Retrieval-Augmented Generation (RAG) has emerged as a significant approach in addressing the limitations of traditional Large Language Models (LLMs), which are trained on vast corpora and rely entirely on their internal representations of knowledge, making them susceptible to issues like hallucinations, which is when the models generate plausible but incorrect information. These models cannot efficiently update their knowledge without retraining, making them less practical for dynamic, knowledge-intensive tasks where accurate and contextually grounded outputs are essential [GRS24]. This chapter, among other things, provides a background and explores related work in the domain of RAG systems, with a specific focus on their applicability as institutional assistants, like smart assistants for universities.

2.1 Background

2.1.1 Retrieval Augmented Generation

RAG has been cemented as a powerful, reliable, and relatively low-cost technique to enhance LLMs by conditioning their generation on external evidence retrieved at inference time [Sha25]. This approach enables LLMs to incorporate data from external knowledge bases, significantly enhancing the accuracy and credibility of generated content. RAG is particularly useful for knowledge-intensive tasks, allowing for continuous knowledge updates and the integration of domain-specific information in the LLM response [Lew+20; Gao+24a]. RAG synergistically merges the intrinsic knowledge of LLMs with the vast, dynamic repositories of external knowledge bases. Since its introduction, RAG has been widely adopted as a key technology for advancing chatbots and improving the suitability of LLMs for real-world applications [Gao+24a].

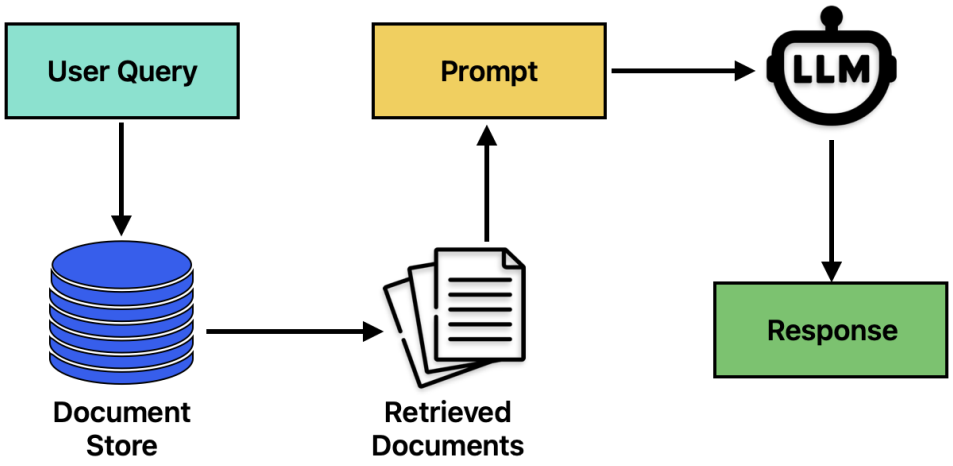

Fig. 2.1.: The main idea behind a RAG pipeline. The system retrieves relevant documents from a document store to supplement the user prompt, enabling the generator (LLM) to produce grounded responses.

A typical RAG system generally consists of three core stages: **indexing**, **retrieval**, and **generation** [Gao+24a].

Indexing involves several interconnected steps, beginning with the extraction and preprocessing of raw data from various input formats, including plain text, structured documents, code, and tables, into a uniform textual representation. Given that each LLM has a specific context window, which defines the maximum number of tokens it can process, this textual data is typically segmented into smaller, semantically meaningful units through a process known as *chunking*.

Chunking is critical to mitigating the well-documented "*Lost in the Middle*" phenomenon, where LLMs tend to disproportionately attend to the beginning and end of lengthy texts, neglecting the middle segments [Liu+23]. Effective chunking strategies, therefore, strive to produce segments that comfortably fit within the model's context window while preserving semantic coherence and completeness. The choice between fixed-length chunking, structured chunking based on document headers and subheaders, or alternative segmentation methods heavily depends on the specific characteristics of the input data and the intended application of the RAG system.

After chunking, these textual segments are encoded into suitable representations, commonly as sparse vectors using traditional lexical methods (e.g., BM25), dense embeddings leveraging neural encoders (e.g., SBERT [RG19]), or other variants such as learned sparse embeddings (e.g., SPLADE [FPC21]) and multi-vector models (e.g., ColBERT [KZ20]), which support token-level matching via late interaction—that is, they encode each token separately and compute relevance by matching each query token to the highest-scoring document token. Unlike dual-encoder models such as SBERT, which encode each segment

as a single fixed vector—typically via pooling across tokens—and thus compute similarity holistically, these late-interaction models preserve fine-grained semantic interactions across tokens. These encoded representations are then stored in databases optimized for efficient retrieval during inference. Depending on factors such as data complexity, scale, and the specific relationships within data chunks, the indexing phase might utilize traditional vector stores or graph databases. Graph databases, in particular, allow the encoding and querying of complex interrelations among indexed chunks, thus facilitating richer semantic retrieval and enhancing the RAG system's ability to provide contextually accurate and interconnected responses.

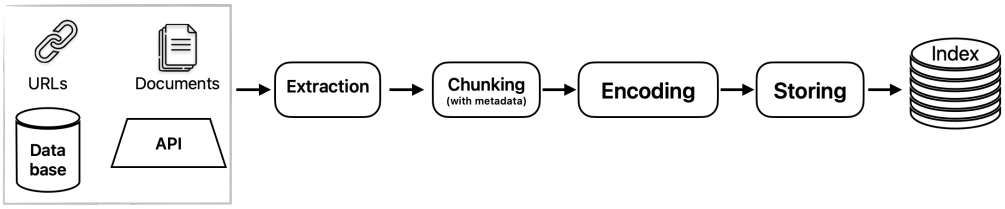

Fig. 2.2.: Overview of a typical indexing process.

Retrieval: Upon receiving a user query, the RAG system employs the same text encoding method used during indexing to transform the query into an appropriate text representation for the retriever. Then, all the saved chunk representations are compared to the user query representation and ranked from the top to less similar. The K most similar chunks are then added to the prompt as relevant context to help the model in the answer generation phase [Gao+24a].

Generation: During the generation phase, the query posed by the user and the documents retrieved from previous steps are synthesized into a coherent prompt provided to an LLM. The LLM subsequently generates a response, potentially utilizing its intrinsic parametric knowledge or strictly adhering to the information from the retrieved documents, depending on the instructions included in the prompt. In conversational systems, dialogue history may also be incorporated into the prompt to support coherent multi-turn interactions [Gao+24a].

The above three phases establish the **Naive RAG** paradigm, which represents the earliest methodology and a standard starting point of the development process of a RAG system [Gao+24a]. It involves a singular retrieval step followed by generation, which is typically insufficient for complex problems requiring multi-step reasoning, as it provides a limited scope of information, and it may even reduce answer generation accuracy if the retrieved context is irrelevant.[Yor+24; Sha25; Gao+24a].

The RAG paradigm has continued to evolve, advancing with each new framework introduced. **Advanced RAG** introduces specific improvements to overcome the limitations of

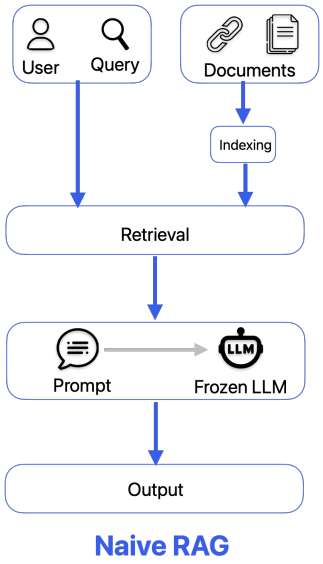

Fig. 2.3.: Overview of the Naive RAG paradigm (figure reconstructed by the author from Gao et al. [Gao+24a]).

Naive RAG, primarily focusing on enhancing retrieval quality through pre-retrieval and post-retrieval strategies.

Pre-Retrieval techniques include the following:
Query Rewriting seeks to reformulate the user's original query into a more effective search input by resolving ambiguity, coreference, ellipsis and aligning the query to the retrieval system's vocabulary. In conversational settings, this can involve rewriting incomplete questions into self-contained forms [Wu+22]. More recent RAG-oriented approaches frame query rewriting as a learnable module inserted before retrieval. For example, the *Rewrite–Retrieve–Read* paradigm uses a lightweight rewriter (e.g., a small LLM) to generate a reformulated query that aligns better with both the retriever and the LLM reader (a frozen LLM that comprehends the question together with the retrieved info to produce an answer) and can be fine-tuned via reinforcement learning to maximize downstream QA performance [Ma+23].

Query Expansion augments the rewritten or original query by injecting additional terms that are semantically related, such as synonyms, hypernyms, or statistically co-occurring words, to mitigate vocabulary mismatches [CR12]. Traditional methods include pseudo-relevance feedback and thesaurus-based augmentations, whereas modern techniques exploit embeddings or LLMs to propose context-aware expansion terms [Jag+23; Nas+21]. This process often yields gains in recall and occasionally precision [AD19].

Query Routing (also termed **dynamic retrieval selection**) determines which retrieval subsystem(s) should handle a given query. In hybrid retrieval architectures, one may use a lightweight classifier or metadata matcher to route queries to a sparse retriever, dense

retriever, or even domain-specific index segments. This way, each query is handled by the most suitable retriever [Gao+24a].

These three pre-retrieval techniques are complementary: rewriting improves query clarity, expansion increases recall, and routing selects the most appropriate retriever(s). Together, they significantly improve the quality of the candidate set entering the main retrieval phase.

Post-Retrieval techniques include the following:
Reranking is a refinement step performed after the initial document retrieval, where another model, often a cross-encoder [PMM24] of some kind, is tasked to reassess and reorder the retrieved candidates by their actual relevance to the query [NC19; Nog+20].

Fusion methods aim to further improve retrieval quality by combining or aggregating the results of multiple retrieval strategies or queries. In advanced RAG systems, fusion refers to techniques where retrieved document lists from different retrievers or query formulations are merged and re-ranked to produce a single, more comprehensive set of candidates. Rank fusion approaches, and specifically the Reciprocal Rank Fusion (RRF) method, are discussed in greater detail in Section 2.1.4

Context Compression (or **Contextual Compression**) condenses the retrieved documents, ensuring that only the most relevant information is passed to the language model. By summarizing or filtering content based on the user query, it reduces token usage, cuts inference time, and limits noise, without sacrificing response quality [Ver24].

Building upon Advanced RAG, the **Modular RAG** paradigm restructures the pipeline by decomposing it into discrete, interchangeable modules and operators. Instead of a rigid retrieve–generate chain, systems can dynamically orchestrate components such as routing, rewriting, retrieval, fusion, and generation into flexible workflows that support conditional branching, iterative loops, or parallel execution [Gao+24a; Gao+24b]. As illustrated in Figure 2.5, these architectures operate more like "LEGO-style" frameworks: modules such as Routing, Search, Rewrite, Retrieve, Rerank, Fusion, and Predict can be recombined to implement diverse patterns. These include linear flows as in Naive RAG, enhanced flows with reranking (Advanced RAG), or demonstration-augmented search flows, such as DSP, which stands for Demonstrate–Search–Predict and is a pattern in which the system first demonstrates by prompting the LLM with examples or few-shot context, then performs a search using the LLM's output or query context, and finally predicts the answer based on the retrieved information. Another notable pattern is the iterative retrieval–generation loops pattern, such as ITER-RETGEN, in which retrieval and generation are interleaved in a loop: each generation step produces intermediate output, which is used to perform more focused retrieval, progressively refining the context. This loop continues for a predefined number of iterations or until convergence. One more

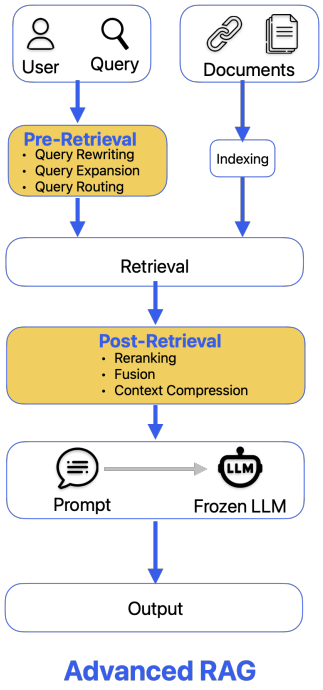

Fig. 2.4.: Overview of the Advanced RAG paradigm (figure reconstructed by the author from Gao et al. [Gao+24a]).

example of adapted behaviour that is enabled by this modularity is that the orchestration layer may route queries to different retrieval subsystems depending on metadata or query type [Gao+24a; Gao+24b]. By framing RAG as a configurable graph of operators, Modular RAG offers increased flexibility, enabling task-specific pipelines, easy experimentation, and the evolution of new retrieval and generation patterns beyond the limitations of prior paradigms [Gao+24b].

Retrieval Strategies

In RAG, retrieval is achieved by calculating the similarity between the representation of the question and document chunks. This representation is usually created from a sparse retriever or a dense retriever.

2.1.2 Sparse Lexical Retrieval with BM25

BM25, which stands for Best Match 25, is a widely used retrieval model that, like any retriever, ranks documents based on their relevance to a given query, and it forms the backbone of many modern Information Retrieval (IR) systems [RZ09]. Unlike simple term frequency-based approaches, BM25 accounts for term saturation (diminishing returns for repeated term occurrences), document length normalization, and inverse document frequency, making it robust and effective across various text corpora.

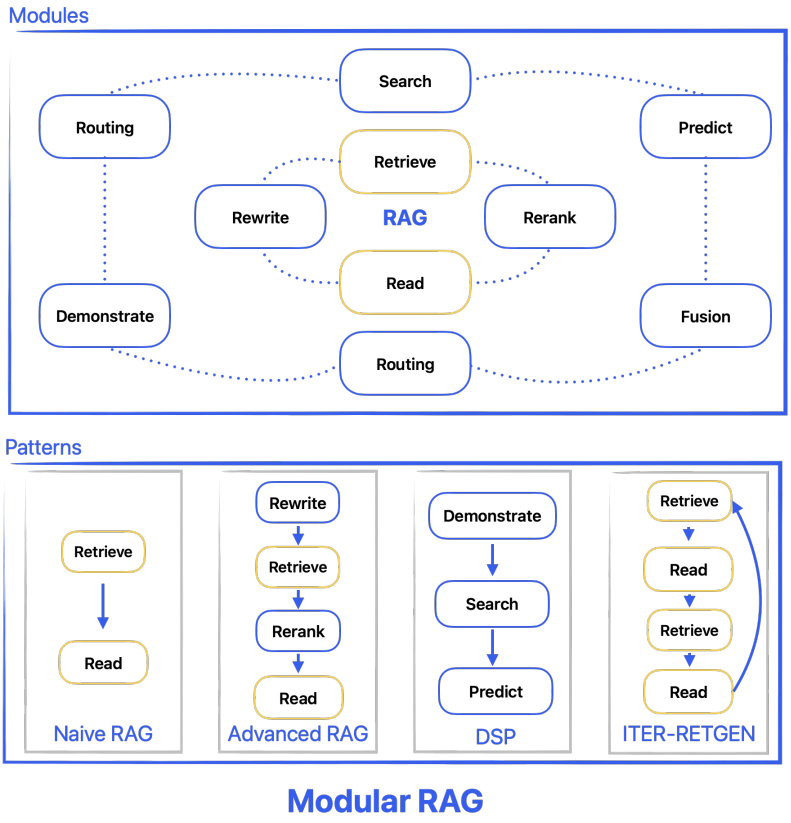

Fig. 2.5.: Overview of the Modular RAG paradigm (figure reconstructed by the author from Gao et al. [Gao+24a]).

Formally, the BM25 score for a document D given a query Q is defined as:

$$\text{BM25}(D,Q) = \sum_{q_i \in Q} IDF(q_i) \cdot \frac{f(q_i,D) \cdot (k_1+1)}{f(q_i,D) + k_1 \cdot (1 - b + b \cdot \frac{|D|}{\text{avgdl}})} \tag{2.1}$$

where $f(q_i, D)$ is the term frequency of query term q_i in document D, $|D|$ is the document length, $avgdl$ is the average document length in the corpus, and k_1, b are tunable hyperparameters controlling term frequency scaling and length normalization, respectively. $IDF(q_i)$ denotes the *inverse document frequency*, typically calculated as

$$\text{IDF}(q_i) = \log \frac{N - n(q_i) + 0.5}{n(q_i) + 0.5} + 1, \tag{2.2}$$

where N is the total number of documents in the corpus and $n(q_i)$ is the number of documents containing term q_i, thereby emphasizing terms that are relatively rare in the collection.

BM25 has been extensively adopted due to its lexical matching capability, which efficiently narrows down a large document set to a manageable subset of potentially relevant passages or chunks.

Overall, BM25 remains a foundational component in modern RAG systems, serving as both a reliable baseline and a tactic to boost the performance of multiple retriever (commonly referred to as mixed or hybrid retrieval) setups, by providing precise lexical keyword-based matching that captures exact term overlap, thus complementing dense retrieval that focuses on semantic similarity with fine-grained relevance [Kuz+20; WZZ21].

2.1.3 Dense Semantic Retrieval with Sentence-BERT

Dense retrieval models represent a significant advance in information retrieval for RAG. Among the most widely adopted models in this space is **Sentence-BERT (SBERT)**, introduced by [RG19]. SBERT is a modification of the classic BERT architecture that Devlin et al. [Dev+19], specifically designed to generate semantically meaningful sentence embeddings, enabling efficient and accurate similarity search, clustering, and large-scale information retrieval.

Standard BERT-based models for sentence-pair tasks use a cross-encoder architecture, where both sentences are fed together into the transformer, and their similarity is predicted. While highly effective for many supervised tasks, this approach is computationally infeasible for large-scale retrieval, as it requires running the transformer for every possible pair. In contrast, SBERT employs a siamese (two identical networks with shared weights) or triplet network structure to derive semantically meaningful sentence embeddings. During training, SBERT uses pairs or triplets of sentences, depending on the chosen architecture, to learn sentence embeddings such that sentences with similar meanings are closer in the embedding space [RG19].

In more detail, the siamese structure works by taking two sentences, encoding them individually into embeddings using the shared SBERT model, and then applying a pooling operation to obtain fixed-sized embeddings. These embeddings are compared using a similarity measure (typically cosine similarity) to quantify the degree of conceptual alignment between the input sentences. The SBERT model is then fine-tuned using objective functions (classification, regression, and triplet objective functions) designed to directly optimize semantic similarity [RG19].

Because each document or sentence is encoded into an embedding vector independently of the query, this process needs to be done only once for a given corpus of documents. After this encoding, the embeddings can be reused across multiple queries, significantly reducing computational overhead during retrieval compared to traditional cross-encoder architectures that must process each sentence pair individually every time a query is made.

Formally, given a query q and a candidate document or passage d, both are independently encoded as vectors using a shared SBERT model:

$$\mathbf{u} = \text{SBERT}(q), \qquad \mathbf{v} = \text{SBERT}(d) \tag{2.3}$$

The relevance score between q and d is then computed by various similarity metrics, with cosine similarity being a common choice:

$$\text{Score}(q, d) = \cos(\mathbf{u}, \mathbf{v}) = \frac{\mathbf{u} \cdot \mathbf{v}}{\|\mathbf{u}\|\|\mathbf{v}\|} \tag{2.4}$$

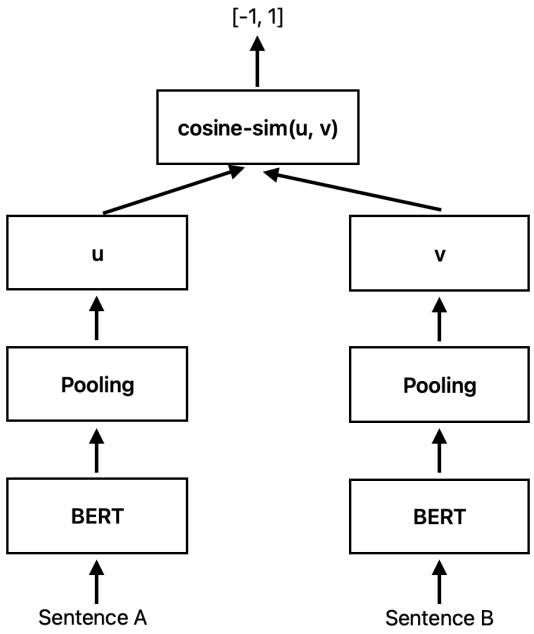

Fig. 2.6.: Overview of the SBERT architecture (figure reconstructed by the author from Reimers and Gurevych [RG19]).

This, coupled with vector databases such as FAISS[2] [JDJ21], allows for fast and accurate similarity search and retrieval. A vector database is a type of database that stores data as high-dimensional vectors, typically produced by embedding functions (such as SBERT), making them particularly suitable for managing large-scale semantic search operations.

Compared to approaches that rely on simpler embedding strategies, such as using the CLS (which stands for Classify) token, which encodes a general representation of the entire input sequence, SBERT significantly improves semantic coherence by explicitly training embeddings to reflect sentence-level meaning. The CLS token in standard BERT models is a special token used to aggregate the entire input sequence representation, yet it may not always capture nuanced sentence-level semantics effectively. In contrast, SBERT directly optimizes embeddings for sentence-level semantic tasks, thus dramatically improving the semantic coherence of the embedding space.

[2]FAISS stands for Facebook AI Similarity Search.

Empirical results have consistently demonstrated that SBERT outperforms previous unsupervised and supervised sentence embedding methods across a range of semantic textual similarity and transfer tasks [RG19]. Notably, the time required for semantic search over a corpus of 10,000 sentences decreases from hours (with traditional BERT cross-encoders) to mere seconds using SBERT [RG19].

2.1.4 Hybrid Retrieval and Rank Fusion

Hybrid Retrieval refers to the integration of different retrieval techniques into a single search pipeline. A major challenge in IR and RAG pipelines is that no single retrieval model—whether sparse (BM25), dense (embedding-based), or otherwise—consistently dominates across all query types and domains. Sparse and dense retrievers capture different notions of relevance: sparse models excel at matching exact keywords and rare entities, while dense models leverage semantic representations to bridge lexical gaps and handle paraphrasing or concept matching [MGG25; Sha+24]. However, each approach exhibits limitations when applied in isolation, especially in complex, knowledge-intensive settings where both lexical fidelity and semantic coverage are important. The goal behind hybrid retrieval setups is to capitalize on the strengths of both. Practically, a hybrid system will issue both dense and sparse vector searches in parallel and then combine the results in a single unified ranking.

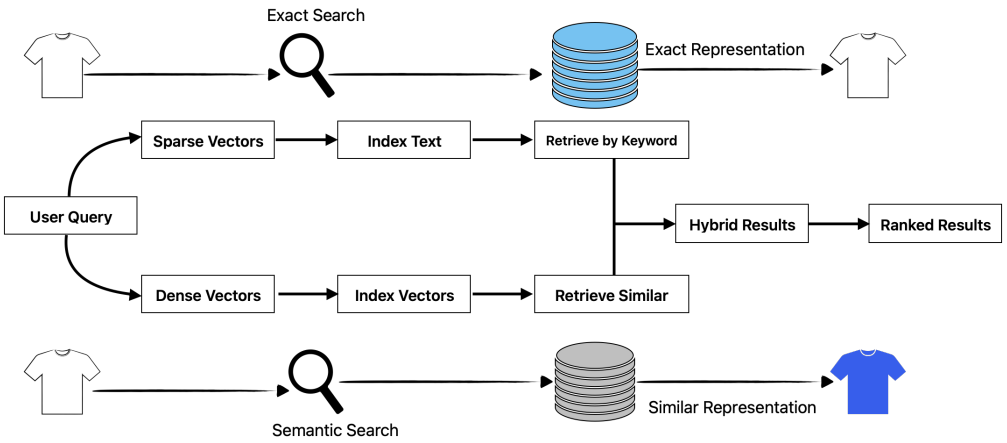

Fig. 2.7.: Intuitive Overview of a hybrid retrieval setup, leveraging both lexical and semantic search (figure reconstructed by the author from this website: `https://www.couchbase.com/blog/hybrid-search/`.

Rank fusion methods are a family of techniques, used within hybrid retrieval setups, that merge the outputs (i.e., ranked document lists) of multiple, frequently diverse retrievers into one coherent ranking. The central motivation is to leverage the complementary strengths of different retrievers, thereby boosting recall, increasing robustness, and reducing the risk of missing relevant documents due to the blind spots of any individual retriever [Kuz+20;

Sha25; Gao+24a]. Notably, empirical studies have demonstrated that hybrid retrieval, such as combining BM25 with neural retrievers, consistently achieves higher recall and mean average precision (see Section 6.1.1 for more information) than either approach alone, with improvements remaining robust across a range of queries and retrieval depths. In the context of RAG, these improvements in retrieval recall and ranking translate directly to better factual grounding, higher answer consistency, and reduced hallucination rates in generated outputs [Kuz+20; Sha25].

Among various fusion strategies, **Reciprocal Rank Fusion (RRF)** has emerged as a simple yet highly effective method for merging ranked lists from different retrieval systems. RRF requires no training and operates purely on the rank positions of retrieved documents, making it robust and model-agnostic [CCB09; MGG25]. The RRF score for a document d across a set of retrieval runs R is calculated as:

$$\text{RRFscore}(d) = \sum_{r \in R} \frac{1}{c + r(d)} \tag{2.5}$$

where $r(d)$ denotes the rank position of d in retriever r's list of ranked documents (with unranked documents assigned an infinite rank), and c is a smoothing constant, typically set to 60. This formulation assigns higher scores to documents ranked highly by any individual system, and particularly favors documents that are ranked highly by more than one retriever, so that items appearing near the top of several lists receive substantially higher aggregate scores than those ranked highly by only a single retriever [CCB09; MGG25]. This way, RRF encourages both consensus (i.e., retrieval depth), by up-weighting documents highly ranked by multiple retrievers, and cross-retriever coverage (i.e., retrieval diversity), by ensuring that unique high-quality documents from any individual list are still considered for the final ranking.

A common extension of RRF incorporates trainable weights to prioritize one retrieval signal over another. In this variant, each retriever's contribution is scaled by a predefined weight, allowing the system designer to emphasize, for example, dense semantic rankings over BM25 or vice versa. The weighted RRF score for document d is calculated as:

$$\text{WeightedRRF}(d) = \sum_{r \in R} w_r \cdot \frac{1}{c + r(d)} \tag{2.6}$$

where w_r is the weight assigned to retriever r, $r(d)$ is the rank of $\mathbf{d}$ in $\mathbf{r}$'s list of ranked documents, and c is again a smoothing constant (typically set to 60). This weighted extension allows for fine-tuning the influence of each retrieval signal according to task-specific needs, and has been shown to improve performance in hybrid and even multi-modal retrieval scenarios [Sam+25].

Unlike alternative fusion methods (e.g., CombMNZ [FS93] or Condorcet voting [MA02]), RRF does not depend on calibrated scores or require global rank normalization. Experimental results across a wide range of TREC[3] and benchmark datasets show that RRF consistently outperforms both individual retrieval systems and other unsupervised fusion approaches [CCB09; MGG25]. In Advanced RAG pipelines, applying RRF to combine sparse and dense retriever outputs not only improves top k recall but also reduces LLM hallucination rates by ensuring that the most contextually relevant documents, whether retrieved by sparse or dense means, are surfaced for answer generation [MGG25].

In summary, reciprocal rank fusion provides an efficient, interpretable, and empirically validated mechanism to ensemble multiple retrieval signals in modern RAG architectures, supporting more accurate, robust, and trustworthy question-answering systems.

Generation Step

In RAG, the generation step is the final stage where the LLM synthesizes the user query together with the retrieved context to produce the answer. This phase is critical, as it directly determines the factual accuracy, fluency, and overall usefulness of the system's output. Depending on the prompting strategy, the model may strictly adhere to the retrieved evidence, combine it with its own parametric knowledge for reasoning, or balance between the two.

2.1.5 Generative Model Characteristics

In the RAG paradigm, the generative component plays a pivotal role by synthesizing final answers conditioned on retrieved context. The LLaMA[4] family of models, and especially the LLaMA 3 family of models, exhibit several architectural and methodological choices that render them particularly effective for tasks such as Question-Answering [Lla24].

First, LLaMA 3 models are trained on a rigorously curated, filtered, and deduplicated corpus of approximately 15 trillion multilingual tokens, including substantial quantities of code, mathematical, and reasoning data. Such scale and diversity result in strong generalization capabilities, robustness to diverse query formulations, and reliable performance in knowledge-intensive and multi-lingual settings. Additionally, targeted and domain-specific "annealing" phases during pre-training, where the learning rate is linearly reduced to 0 over the final 40 million tokens, high-quality reasoning and code data are upsampled, and

[3] TREC (Text REtrieval Conference) is an ongoing, annual evaluation workshop series organized by the U.S. National Institute of Standards and Technology (NIST). TREC serves as a foundational platform for collaborative, pre-competitive benchmarking in IR, offering reusable test collections and an open forum for rigorous system comparison. You can find out more at https://trec.nist.gov.

[4] LLaMA is short for Large Language Model Architecture

checkpoints are averaged. According to the Llama Team [Lla24], this step further enhances the model's mathematical and reasoning performance, which is crucial for academic and technical question answering.

A second cornerstone is that they employ the Transformer architecture with **Rotary Positional Embeddings (RoPE)** [Su+21]. In contrast to traditional absolute positional encodings, which are simply added to the input embeddings, RoPE introduces position through a sequence of rotations applied to the query and key vectors within each attention head. Specifically, RoPE represents each token's position by rotating pairs of embedding dimensions in the complex plane, with the angle of rotation proportional to the token's position index. By integrating positional information directly into the self-attention mechanism, RoPE ensures that the similarity between tokens depends not only on their content but also on their relative positions within the sequence. This design enables the model to capture both absolute and relative positional relationships, thus allowing it to generalize more effectively to longer or previously unseen sequence lengths, a capability that is critical for applications where retrieved context windows can be extensive. The latest LLaMA models further increase the RoPE base frequency, enabling context windows of up to 128,000 tokens, far surpassing most open-source counterparts.

Third, LLaMA 3 models are trained using a straightforward yet robust two-phase training procedure. This process begins with large-scale pre-training on diverse, high-quality data to build general language understanding. The subsequent post-training phase relies on a streamlined combination of **supervised fine-tuning (SFT)** [XZ25], **rejection sampling (RS)** [XZ25], and **direct preference optimization (DPO)** [Raf+23], rather than more complex reinforcement learning algorithms. Here, rejection sampling refers to generating multiple output candidates per prompt and selecting the highest-scoring one according to a reward model, essentially sampling several outputs, evaluating them via the reward function, and retaining only the best for training. By leveraging this multitude of relatively simple but effective techniques, LLaMA models improve alignment with human instructions and factual correctness while minimizing the engineering overhead and instability often associated with reinforcement learning from human feedback (RLHF) methods [Zie+20; Ouy+22]. This results in models that are both highly capable of following detailed instructions and well-aligned with human values, critical for reliable answer generation in RAG systems.

Finally, LLaMA 3 models intentionally employ a standard dense Transformer architecture [Vas+17] with minor modifications from the LLaMA 2 series [Lla24], deliberately avoiding the additional complexity of **mixture-of-experts (MoE)** approaches [Sha+17]. MoE architectures represent a class of neural network designs in which a gating network dynamically routes each input token to a small, specialized subset of "expert" feed-forward sub-layers, rather than sending all tokens through the full feed-forward network at each layer. This design principle allows MoE-based models to scale to trillions of parameters, as

only a fraction of the network is utilized at any given time. However, the routing decisions made by the gating network, as well as the auxiliary losses required to balance the load among experts, introduce significant engineering complexity and may lead to variability in inference behavior. By contrast, the LLaMA 3.1 family deliberately avoids this architectural paradigm, opting for a dense, decoder-only Transformer backbone that processes all tokens through the same layers. This design choice is motivated by a desire to maximize stability, scalability, and ease of deployment, as explicitly noted by the developers in the "LLaMA 3 Herd of Models" paper [Lla24]. This choice, combined with other optimizations, including grouped query attention (GQA)-a technique that partitions the query heads into multiple groups with each group sharing one key-value head (with fewer groups than query heads), thereby balancing efficiency and performance [Ain+23]-allows LLaMA models to achieve efficient inference, high-quality multilingual performance, and state-of-the-art generation capabilities [Lla24].

Collectively, these properties position the LLaMA 3 family of models as state-of-the-art open-source choices for the generation step in RAG pipelines, supporting robust, contextually grounded, and instruction-following outputs across a wide array of knowledge-intensive domains [Lla24].

The interaction between LLMs and RAG forms a powerful paradigm, as RAG can effectively leverage the superior reasoning capabilities of LLMs, combined with the broad knowledge scope of external data, to explore the potential applications of LLMs more extensively [HH24]. On the other hand, LLMs can serve as crucial components in RAG, functioning as the decision maker, answer generator, or even evaluator of certain aspects of a RAG pipeline [Yu+25]

2.2 Related Work

RAG has gained significant traction within the field of NLP as it addresses critical limitations associated with LLMs, particularly their tendency toward hallucinations, input token constraints, outdated knowledge, and limited access to real-time or domain-specific information. The emergence of RAG methodologies has allowed LLMs to condition their generation on external knowledge sources retrieved at inference time, significantly enhancing the accuracy, relevance, and factual consistency of their outputs [Lew+20]

2.2.1 Significance of Retrieval Granularity

The granularity at which documents are segmented for retrieval has a profound impact on the effectiveness of both the retrieval process and downstream LLM generation in RAG systems. While segmenting text into smaller chunks reduces irrelevant information

and noise, it can also result in the loss of critical context, as key relationships may be split across multiple segments. To mitigate this, recursive splitting and sliding window techniques have been proposed. Recursive splitting is a chunking strategy that attempts to divide long texts into manageable chunks following a hierarchy of separators—such as paragraphs, lines, words, and finally character boundaries. If a segment exceeds the defined chunk size, it recursively tries the next finer-grained separator until all chunks fall within the specified limit. In the sliding window chunking strategy, the text is segmented into fixed-size, overlapping chunks by moving a sliding window across the text. Each chunk overlaps with its neighbor, thereby maintaining important context across chunk boundaries. While this preserves semantic continuity, it does introduce redundancy and additional computation overhead due to repeated content processing. However, these methods still struggle to fully balance the need for semantic coherence and semantic completeness against the strict limitations imposed by the context length of large language models [Gao+24a].

Recent studies highlight that the retrieval unit significantly impacts a RAG system's performance. Recent research by Chen et al. [Che+24] demonstrate that decomposing documents into propositions, defined as atomic, self-contained statements expressing distinct facts, can significantly improve the relevance and factuality of retrieval outputs. Propositions offer a level of granularity finer than just splitting a text passage into sentences, as they isolate semantically complete, context-independent pieces of information. This proposition-level indexing not only reduces contextual noise but also mitigates issues of ambiguity and co-reference that can hinder retrieval and downstream answer generation. Chen et al. [Che+24]'s findings indicate that using propositions as the fundamental retrieval unit enhances precision, improves grounding, and supports more efficient utilization of the limited context window available in LLMs, thus representing a promising direction for robust RAG pipelines. Similarly, Vlachos et al. [Vla+25] find the use of propositions more beneficial (as opposed to other granularities, like chunks or sentences), particularly for synthetic dialog generation for conversation question-answering.

2.2.2 Synthetic Data Generation

A persistent bottleneck in the development and evaluation of RAG systems is the creation of high-quality, large-scale annotated QA datasets. Manual annotation remains costly, time-consuming, and often limited in both scale and adaptability to new domains or languages, introducing potential annotation biases and limiting coverage of emerging or specialized topics. To address these challenges, the use of synthetic data generation methods, particularly those leveraging LLMs, has rapidly gained traction as a means to automate the creation of synthetic text data (ranging from question answer pairs and whole dialogues to even code), thus dramatically reducing reliance on human annotation and accelerating RAG research and deployment [Sou+24].

Recent work provides systematic taxonomies of conversational data generation pipelines, categorizing them into key phases such as seed data creation, utterance generation, and quality filtering. These synthetic data pipelines are capable of transforming existing textual resources—including documents, tables, and knowledge graphs—into multi-turn, contextually rich dialogues or factual QA pairs. This process enables the efficient augmentation of datasets for three broad classes of dialogue systems: task-oriented, open-domain, and information-seeking systems, which are each central to modern RAG applications [Sou+24].

Notably, techniques such as *dialog inpainting* [Dai+22] have been introduced to address the scarcity of high-quality multi-turn conversational data. Dialog inpainting leverages LLMs to convert documents into simulated conversations between a writer and an imagined reader, interleaving real document sentences with automatically generated user queries. The resulting datasets, such as WikiDialog and WebDialog, achieve not only orders of magnitude larger than prior manually constructed datasets, but also human-competitive conversational adequacy and answer quality. According to Dai et al. [Dai+22], using these datasets to develop ConvQA retrieval systems showed substantial gains in key retrieval metrics, indicating increased retrieval performance.

2.2.3 LLM-as-Judge Evaluation

The evaluation of Retrieval-Augmented Generation (RAG) systems presents unique methodological challenges due to their complex hybrid architecture, which couples retrieval and generative modules. Traditional evaluation metrics, such as BLEU [Pap+02], ROUGE [Lin04], METEOR [LA07], and BERTscore [Zha+20] (see Section 6.2.1 for more information), while useful for certain subtasks, often fail to capture the nuanced interplay between retrieval accuracy, factual consistency, and answer quality within end-to-end RAG pipelines. This limitation is particularly pronounced in knowledge-intensive or open-ended domains, where the gold standard remains expert human annotation, a process that is both time-consuming and expensive to scale [Gan+25; Li+24].

To address these bottlenecks, the paradigm of LLM-as-a-Judge has emerged as a scalable and cost-effective surrogate for human evaluation in both RAG and general LLM-driven applications. In this approach, a strong, large language model (typically GPT-4 or comparable) is prompted to assess the factuality, relevance, and completeness of candidate answers. The model is tasked with mimicking human-like evaluative reasoning, allowing for rapid, context-sensitive, and explainable assessments at a fraction of the cost of human annotation [Zhe+23; Li+24].

Recent empirical studies demonstrate that LLM-based judges can match, and occasionally exceed, the consistency of human evaluators, especially when equipped with robust prompt

engineering and bias-mitigation strategies. For example, Zheng et al. [Zhe+23] show that GPT-4, when used as a judge in the MT-Bench[5] and Chatbot Arena[6] settings, achieves over 80% agreement with expert human raters—comparable to human-human agreement rates—across thousands of dialogue and QA interactions. Moreover, LLM judges can provide natural language justifications for their scores, enhancing the transparency and interpretability of the evaluation process [Li+24]. Other surveys corroborate these findings, documenting the adaptability and reliability of LLMs-as-judges across diverse domains, including education, legal, and technical QA [Li+24; Gan+25].

Nevertheless, the LLM-as-Judge evaluation scheme is sensitive to prompt design and susceptible to certain biases, such as LLM evaluators favoring more verbose outputs [Ye+24], choosing the first or last option rather than assessing all available context equally [Ye+24], or exhibiting self-enhancement bias, which refers to the tendency of the evaluating model to favor responses that it generated itself—i.e., assigning higher preference or scores to its own outputs over those from other models [PBF24]. These biases can skew evaluation outcomes and reduce reliability. Another factor that may decrease the reliability of this evaluation scheme is the task difficulty and the domain in which it is employed. Specifically, Zheng et al. [Zhe+23] found that LLMs face significant difficulties with grading mathematical or other highly specialized reasoning tasks. To maximize reliability, best practices include prompt calibration (i.e., iteratively refining the evaluation prompt to ensure that the LLM applies the intended criteria), randomized answer order, and benchmarking LLM judgments against a subset of human gold labels [Zhe+23; Li+24]. Despite these caveats, the LLM-as-Judge paradigm has rapidly become an indispensable tool in RAG evaluation pipelines, balancing scalability, cost, and human-like discernment with unprecedented efficiency.

2.2.4 Institutional RAG Assistants

Recent work has explored the specific application of RAG systems within educational and institutional contexts. Kuratomi et al. [Kur+25a] developed a RAG-based virtual assistant for the University of São Paulo (USP), designed to enhance information retrieval, taking advantage of the plethora of available USP documents, significantly improving LLM performance on QA regarding the institution by providing contextually accurate responses. Similarly, Neupane et al. [Neu+24] introduced BARKPLUG V.2, a RAG-powered chatbot aimed to unlock Mississippi State University's (MSU) resources by effectively handling both academic and non-academic user inquiries, achieving impressive quantitative performance and user satisfaction by effectively leveraging university-specific datasets by curating data of 42 different departments within the university, including academic departments, financial aid, admissions, housing, dining services, library, health center, etc., using web crawlers. Antico et al. [Ant+24] designed "Unimib Assistant," a student-friendly RAG-based

[5] https://github.com/lm-sys/FastChat/tree/main/fastchat/llm_judge
[6] https://lmsys.org/blog/2023-05-03-arena/

chatbot specifically tailored to address the specific needs of students at the University of Milano-Bicocca (UNIMIB), which include easier access to hard-to-find academic and administrative information, simplified navigation across multiple university platforms, and reliable answers with transparent source links. Despite its positive reception for user-friendliness and conversational quality, technical challenges such as hallucinations and inaccurate link generation highlighted ongoing limitations in RAG systems. These studies underscore the importance of tailoring RAG applications to specific institutional contexts to maximize their utility and effectiveness [Ant+24]. Additionally, a recent comprehensive survey by Swacha and Gracel [SG25] investigated 47 distinct educational RAG chatbot applications, highlighting diverse educational functionalities ranging from facilitating direct learning and generating personalized dialogues to supporting administrative and organizational processes. Their findings emphasized RAG's strengths in mitigating hallucinations, a primary barrier to the widespread adoption of LLM-based chatbots in educational contexts. Moreover, the survey underscored the critical role of carefully selecting retrieval methods, integrating relevant and authoritative educational datasets, and fine-tuning LLM prompts to maximize factual grounding, conversational coherence, and overall user satisfaction [SG25].

3 Document Ingestion

3.1 Studies Guide Preprocessing and Chunk-Based Indexing

The foundation of the knowledge base for the assistant is the English-language PDF of the AUEB Department of Informatics Studies Guide, specifically the most recent (2024-2025) edition [7]. The assistant was developed using the English version of the Studies Guide, primarily due to the wider availability and maturity of open-source tools and language models for English. Nevertheless, the design is language-agnostic and can be easily extended to other languages, including Greek, either by translating the source documents or by employing appropriate language-specific retrieval and generation models. Furthermore, although the present thesis centers its attention on this specific Studies Guide, it is important to emphasize that the overall document ingestion pipeline is not inherently tied to any particular document structure. Rather, the process described herein is designed to be broadly applicable to a wide range of institutional documents, provided that minimal adaptations are made to accommodate the inconsistencies present in the source material. Special consideration is given to the treatment of document components such as tables and images, which may require bespoke or manual interventions to ensure their correct integration and representation within the knowledge base, as was indeed the case in the present work.

In order for the Studies Guide document to pass smoothly through the indexing phase of the RAG system, it was necessary to design a highly structured parsing, chunking, and metadata extraction pipeline, closely tailored to the guide's specific structure. The overarching design goal was to maximize semantic coherence within each chunk while strictly preserving logical document boundaries such as headers, sub-headers, and paragraphs. The chunk structure in the Studies Guide is visualized in Figure 3.1. The visualization shows how a header ("VII. Undergraduate Studies") contains a subheader ("VII.1 Introduction and General Information"), which in turn encapsulates a paragraph ("Basic Principles of the Study Programme"), all together forming a coherent chunk of content with a full set of metadata extracted using the custom chunking pipeline.

[7] https://www.dept.aueb.gr/sites/default/files/cs/CS_Manuals/CSStudiesGuide2024-2025.pdf

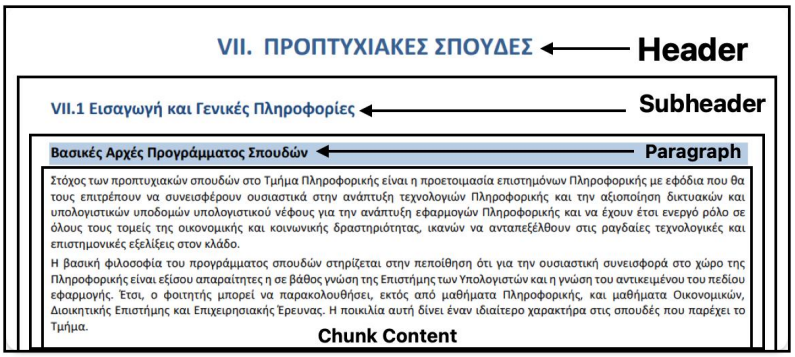

Fig. 3.1.: Visualization of the chunk structure in AUEB's latest Informatics Studies Guide, illustrating how headers, subheaders, and paragraphs are hierarchically organized into a chunk.

This chunking pipeline was explicitly engineered to account for the complex and at times inconsistent structure of the original PDF document. The workflow began with the use of the `pdfplumber`[8] library to extract the raw text of each page, storing page contents in a dictionary structure indexed by page number. All tables within the PDF were detected using `pdfplumber`'s table extraction API. Where tables were deemed irrelevant or contained redundant information, they were programmatically removed from the page text to prevent downstream noise; conversely, critical tables, such as those listing course modules or free elective courses, were manually reviewed, with the relevant information re-integrated into the text in narrative form and assigned to the appropriate chunk.

By leveraging both native PDF font attributes (such as boldness and size), regular expressions that capture Roman numerals and hierarchical section markers, and text alignment heuristics, the detection of headers and sub-headers achieved to further enhance structural accuracy. The Table of Contents was parsed to map each major and minor section to its corresponding page range, providing a high-level navigational map for the subsequent chunking process. Using these structural markers, the Studies Guide was then sliced into distinct, logically bounded text chunks, with each chunk typically corresponding to a unique section, subsection, or paragraph within a given header or sub-header. Chunk boundaries were deliberately aligned with the document's logical structure, ensuring that each chunk encapsulates a coherent, self-contained unit of information. Each chunk may span one or more paragraphs, depending on the structure of the text under each section of the document.

Each chunk is equipped with an array of detailed metadata. These `metadata` include the name of the source file (in this case, the name of the PDF file), the page of the start of the chunk's content in the original document, and the header, sub-header, and paragraph names that the chunk belongs to, extracted from the document's structure. Note that the metadata fields `chunk_id`, `page`, and `file_name` are always present for every

[8] `https://github.com/jsvine/pdfplumber`

chunk. For each chunk, the metadata fields `header`, `subheader`, and `paragraph` may be `None` depending on the document's structure at that point. However, every chunk is always associated with at least one of these fields, so no chunk is created without a contextual reference. As discussed in Section 3.1, these metadata can be leveraged to improve retrieval accuracy by enabling a pre-retrieval filtering step, thereby reducing the pool of candidate documents considered during retrieval [Gao+24a]. Additionally, the systematic extraction of these metadata lay the groundwork for a citation functionality, which would allow users to manually trace each answer back to the precise location in the original document from which supporting information was retrieved. Beyond citation support, the availability of rich metadata annotations also opens up avenues for enhancing retrieval accuracy. In particular, metadata-aware filtering, such as restricting retrieval to specific course modules, academic years, or document sections, could be integrated to constrain the search space and reduce semantic noise during retrieval, especially in cases where user intent is known or can be inferred. It should be emphasized, however, that these functionalities are not actively utilized in the present version of the system. Rather, the document ingestion and chunk extraction pipeline was deliberately designed to capture and expose such metadata so that functionalities like these could be explored in future iterations of the system.

```
CHUNK CONTENT:
Prerequisite Courses
All students, regardless of their year of admission, must have
↪  successfully completed one of the prerequisite courses listed
↪  for each course in the course table on the following pages in a
↪  previous semester to enroll in a course. First-year courses,
↪  courses offered by other departments, and the courses "Logic"
↪  and "Investment Evaluation with Applications in Informatics"
↪  have no prerequisite courses.

CHUNK METADATA:
{
  "chunk_id": 88,
  "header": "VII. UNDERGRADUATE STUDIES",
  "subheader": "VII.1. Introduction and General Information",
  "paragraph": "Prerequisite Courses",
  "page": 24,
  "file_name": "Study Guide 2024-2025 ENG.pdf"
}
```

Fig. 3.2.: Example of an extracted chunk with a complete set of metadata.

Some manual corrections and interventions were necessary in certain sections. Errors, such as spurious page numbers, typographical errors, and encoding issues, were removed or corrected during this step. Specifically, the term "Elective Courses" was systematically replaced by "Free Elective Courses" to match the terminology of the original Greek version. Additionally, information from tables critical to the curriculum, such as course module

assignments and lists of free elective courses, was converted into narrative sentences and inserted into the relevant chunks' text.

Several parsing decisions were necessitated by the presence of tables within the Studies Guide, as four of these contained critical information not found elsewhere in the document. The first was a table listing all the course modules available to students of AUEB's Informatics department. A course module is a title referring to a specific research area of Informatics, which can be printed on a student's degree provided that he/she has passed 5 or more courses that belong to that course module. Some examples of course modules include Data Science, Cybersecurity, and Theoretical Computer Science. This table required a degree of cleaning and normalization; nevertheless, its content was already largely textual and could be integrated into the chunked corpus with only minor adjustments. An example of a course description's extracted text chunk enhanced with the corresponding information from the Course Modules Table is presented in Figure3.3.

```
CHUNK CONTENT (not fully shown for brevity):
Course Modules
Elective core courses and elective courses are organized in course
↪  modules. Provided a student completes a sufficient number of
↪  courses of a module, that module is completed and noted in the
↪  student's transcripts and diploma supplement awarded at
↪  graduation. Completing a module is not required for graduation.
↪  The modules are as follows:
1. Data Science 5. Systems and Networks
2. Operations Research 6. Software Systems
3. Applied Mathematics 7. Data and Knowledge Management
4. Theoretical Computer Science 8. Cybersecurity
The following rules apply to modules:
...

CHUNK METADATA:
{
  "chunk_id": 89,
  "header": "VII. UNDERGRADUATE STUDIES",
  "subheader": "VII.1. Introduction and General Information",
  "paragraph": "Course Modules",
  "page": 24,
  "file_name": "Study Guide 2024-2025 ENG.pdf"
}
```

Fig. 3.3.: Part of the extracted chunk, with its metadata, that contains the Course Module Table in text format.

Another important table was the one detailing the maximum number of ECTS units that students are allowed to enroll in per semester, depending on their year of study. This table was parsed and its information reformatted into a standalone chunk in plain text, ensuring that the structural and regulatory information it conveyed remained directly accessible and uniquely referenced. The extracted text chunk with this table's information is presented in Figure3.4.

```
CHUNK CONTENT (not fully shown for brevity):
VII.3. Course Enrollment and Examination
In order to attend and be graded in courses, in the beginning of
 ↪  each semester students must complete an online course
 ↪  enrollment, which they must submit to the Department's
 ↪  Electronic Secretariat. Course enrollment is mandatory and must
 ↪  be completed at the dates and times announced by the University
 ↪  at the beginning of each semester. Following their enrollment in
 ↪  courses, students are required to submit an electronic textbook
 ↪  selection form through the EYDOXOS platform. It is stressed that
 ↪  course enrollment and textbook selection are distinct, and one
 ↪  does not substitute the other.
The maximum number of ECTS units that students may select and be
 ↪  examined about in each semester is as follows:
The maximum number of ECTS units for 1st year students are 38 ECTS
 ↪  per semester.
The maximum number of ECTS units for 2nd year students are 46 ECTS
 ↪  per semester.
The maximum number of ECTS units for 3rd year students are 54 ECTS
 ↪  per semester.
The maximum number of ECTS units for 4th and subsequent years
 ↪  students are 60 ECTS per semester.

Students are strongly advised, however, not to enroll in more than 6
 ↪  courses per semester, as the requirements of the Department are
 ↪  high for all courses. An effort is made by the Department so
 ↪  that the weekly schedules of courses in the same year are not in
 ↪  conflict. Part-time students ... during the September
 ↪  examination period.

CHUNK METADATA:
{
  "chunk_id": 96,
  "header": "VII. UNDERGRADUATE STUDIES",
  "subheader": "VII.3. Course Enrollment and Examination",
  "paragraph": "VII.3. Course Enrollment and Examination",
  "page": 31,
  "file_name": "Study Guide 2024-2025 ENG.pdf"
}
```

Fig. 3.4.: Part of the extracted chunk, with its metadata, that contains the Table containing the maximum number of ECTS units available to collect each semester in text format.

A further notable table was a comprehensive list of free elective courses accessible to Informatics students from the other departments of AUEB. The entirety of this table was transformed into a sequence of sentences, each describing a course and its corresponding course code, explicitly stating that the course is recognized as a free elective. This approach ensured that all information previously confined to table format was preserved in natural language, aligning it for text retrieval while avoiding information loss.

The most complex table encountered during parsing was the Course Overview Table[9], which compiled information regarding course names, course codes, and prerequisite

[9]The parsing of this table should, in the author's opinion, be avoided, and in later editions of the Studies Guide, this information should be added in each course's description page.

requirements. Except for one particular column, the contents of this table were redundant, as all other details could be found within the detailed course descriptions later in the Studies Guide. The unique information offered by this table was the mapping of each course to its corresponding course modules, a detail absent from the individual course description pages. To preserve this essential linkage, the extracted course module information was converted into sentences indicating the specific modules to which each course belonged. These sentences were then inserted at the beginning of each course's description page, thereby enriching each course chunk with information that would otherwise have been lost in the conversion process. As a result, each course description chunk (typically averaging one document page in length) now contains an explicit statement of its course module assignments, providing both comprehensive coverage and fine-grained semantic annotation for downstream retrieval and question answering.

```
CHUNK CONTENT (not fully shown for brevity):
3515 Logic
Elective Core Course, 5th semester, 7 ECTS units.
<< The course Logic belongs to the following course modules:
 ↪  Theoretical Computer Science, Data and Knowledge Management. >>
Instructor: Assistant Professor Evgenia Foustoukou
URL: https://eclass.aueb.gr/courses/INF441/
Course Description
Formal analysis of the concepts of provability and semantical
 ↪  implication. Propositional Logic: propositional formulas,
 ↪  assignments and satisfiability, logical implication, complete
 ↪  set of connectives , axiomatic system using the Modus Ponens
 ↪  rule, axiomatic system using the resolution rule, formal proofs,
 ↪  soundness and completeness theorems, compactness theorem.
 ↪  Predicate Logic: propositional formulas, structures, valuations,
 ↪  truth within a structure, logical implication, formal proofs,
 ↪  axiomatic system with Modus Ponens rule of proof, axiomatic
 ↪  system with resolution rule of proof, the soundness and
 ↪  completeness theorems, the compactness theorem. Introduction to
 ↪  the principles of Logic Programming. Other topics of logic with
 ↪  applications in Computer Science may include: (Monadic) Second
 ↪  Order Logic, modal logics and temporal logics.
<...>
Assessment Criteria
The final grade is set to the final written examination grade.
 ↪  Active participation in the classroom (with well-posed
 ↪  observations, answers and questions) as well as in the
 ↪  intermediate written examinations and the submission of homework
 ↪  will raise the final grade.

CHUNK METADATA:
{
  "chunk_id": 130,
  "header": "IX. COURSE DESCRIPTIONS",
  "subheader": "None",
  "paragraph": "Logic",
  "page": 55,
  "file_name": "Study Guide 2024-2025 ENG.pdf"
}
```

Fig. 3.5.: Part of the extracted chunk, with its metadata, that contains the description of the "Logic" course. The manual addition of the course modules is enclosed in "«" for display purposes.

Finally, the latest version of the Academic Calendar for the academic year 2024-2025 was also ingested, which was a PDF document with a single section containing three visible subsections, one for each academic period. For alignment with the chunks extracted from the Studies Guide document, the calendar was split into three distinct chunks, corresponding to the “Fall Semester,” “Spring Semester,” and “September Period".

3.1.1 Indexing in Numbers

This process resulted in a knowledge base containing a total of **212** richly annotated, semantically coherent chunks, an otherwise paragraph-level representation of the document (as mentioned before, a chunk may contain more than one paragraph, but for simplicity chunks will be referred to as paragraph-level representations of the document throughout this thesis), ready for use in both semantic and lexical search and downstream RAG-based question answering. By carefully combining programmatic extraction, document structure-driven chunking, metadata extraction, and targeted manual curation, the resulting dataset provides both high coverage of the source material and robust support for systematic, contextually grounded retrieval. This methodology, including the code and custom routines developed, is fully documented and reproducible for future iterations or application to other versions of the Studies Guide.

To gain a deeper insight into the structure of the resulting knowledge base, a statistical analysis of the chunks’ length distribution was conducted, both as word sequences and token sequences, as tokenized by the LLaMA 3.1 tokenizer. Notably, in terms of chunk length measured in words, the average chunk contained approximately 195.76 words, with a standard deviation of 181.44, and a median value of 136.50, indicating considerable variability across sections. When considering the same units in terms of token sequences, the average chunk comprised 293.47 tokens with a standard deviation of 265.57 and a median value of 191.50. These findings underscore the heterogeneous nature of the Studies Guide’s content, as well as the importance of tailored preprocessing strategies that can effectively manage this variability before indexing.

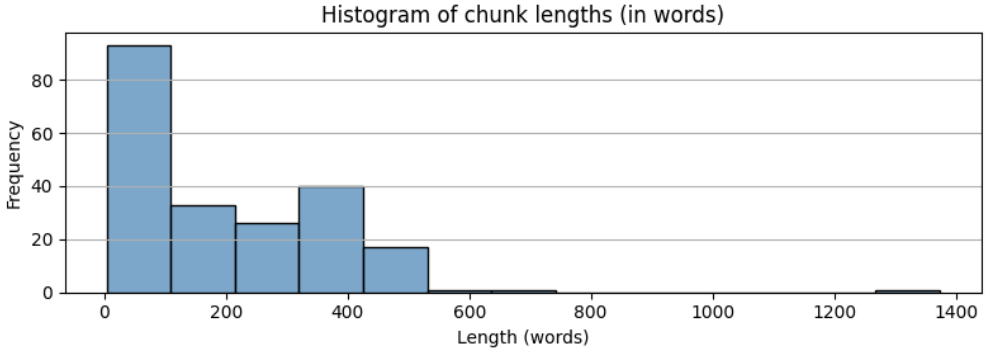

Fig. 3.6.: Distribution of chunk lengths (in words).

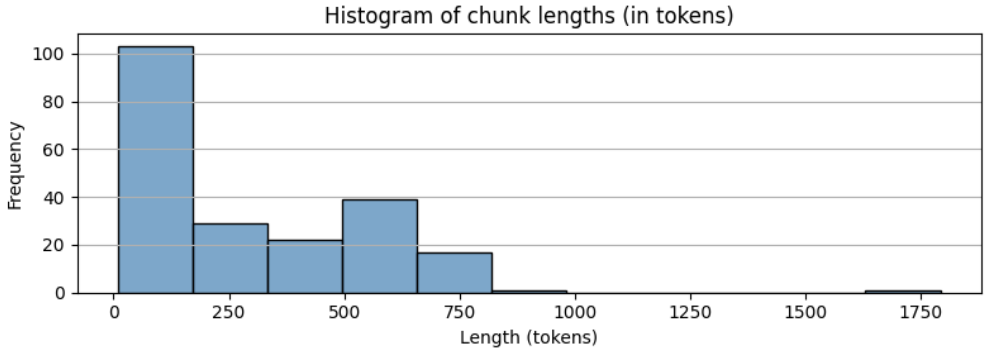

Fig. 3.7.: Distribution of chunk lengths (in tokens, as tokenized by the LLaMA 3.1 tokenizer).

Figure 3.8 illustrates the distribution of the number of chunks that can be accommodated within a single context window of the LLaMA 3-8B-Instruct model, based on 1,000 randomized simulations. Within this setting, each simulation represents a random ordering of the chunk set, followed by sequential filling of the model's context window until the token limit is reached. The x-axis reports the number of chunks that fit, while the y-axis shows the frequency of simulations achieving each value. Most simulations fit between 20 and 35 chunks, with a peak near 28. This result demonstrates the practical effect of context window size on the granularity of retrievable content, highlighting the constraints of working with fixed-length LLM input contexts in RAG systems and the significance of utilizing them effectively by retrieving context as accurately as possible.

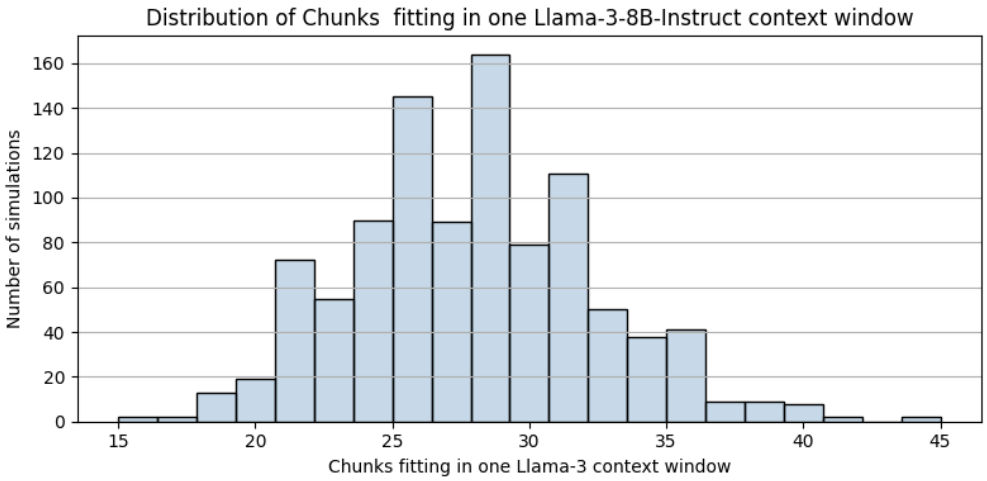

Fig. 3.8.: The number of Chunks that fit in `Llama-3.1-8B-Instruct`'s context window, on average. Note that the default context window with an 8,192k token limit is used here.

3.1.2 Benefits and Limitations of Chunk-Level Document Representation

The decision to adopt a document-specific, chunking strategy for the Studies Guide dataset brings significant strengths but also introduces certain limitations.

Advantages:

- **Semantic Coherence and Contextual Enrichment:** Each chunk typically encapsulates a complete, self-contained unit of meaning, preserving the logical structure imposed by the original document. When additional context, such as the previously detailed metadata, accompanies these chunks, the resulting enhancement—termed contextual enrichment—can help both dense and sparse retrievers make more accurate retrievals.

- **Preservation of Hierarchical Structure:** By embedding metadata such as headers, subheaders, and page numbers, each chunk maintains strong links to its document context. This enables precise traceability and facilitates context-aware question answering.

- **Minimization of Fragmentation:** Grouping information into larger, coherent segments reduces the risk of excessive fragmentation, which could otherwise hinder user comprehension or degrade retrieval performance due to a lack of context.

Limitations:

- **Variable Length:** Chunks exhibit substantial variability in size, ranging from short statements to multi-paragraph bodies of text. This irregularity complicates both storage and retrieval operations.

- **Information Density:** The information density within each chunk is also highly heterogeneous. Some chunks are concise and focused, while others embed numerous, often unrelated facts, making targeted retrieval of specific information more difficult.

- **Implications for Retrieval and Answer Generation:** This heterogeneity in both length and density can adversely affect retrieval granularity and answer quality, especially when user queries require pinpoint precision. Furthermore, excessively long or dense chunks increase the computational load and inference latency for the answer-generating model, a particular challenge when relying on relatively small, resource-constrained open-source language models.

3.1.3 Motivation for Alternative Retrieval Granularities

Despite these trade-offs, chunks provide a robust starting point for information retrieval and question answering, particularly in complex, structured academic documents like the Studies Guide. However, to address the limitations of retrieval precision and answer specificity, subsequent processing stages were introduced: (1) splitting each chunk into its constituent sentences, and (2) generating synthetic, decontextualized propositions using an LLM through a rigorously crafted instruction prompt. These finer granularities aimed to mitigate the aforementioned issues when using chunks as the retrieval unit.

3.2 Sentence-Level Document Representation

While the chunk granularity provides a robust foundation for document retrieval, certain information needs, particularly those requiring concise, factoid-style answers, are better served by finer document granularities. To this end, each chunk was further decomposed into its constituent sentences, yielding a sentence-level dataset optimized for high-precision retrieval and more targeted answer generation.

3.2.1 Sentence Extraction Methodology

The sentence extraction process was implemented using **Natural Language Toolkit (NLTK)**[10]. Each chunk's text was segmented into sentences leveraging pre-trained models for English sentence boundary detection. For each sentence extracted, a unique `sent_id` was assigned, and all metadata from the parent chunk, such as `chunk_id`, `header`, `subheader`, `paragraph`, `page number`, and `file_name` were preserved. This design ensures that every sentence can be precisely traced back to its original context. The output of this process was a collection of **2,554 sentences**. Each sentence contained the raw extracted text along with the complete metadata set.

3.2.2 Rationale and Implementation Details

The choice of NLTK's sentence tokenizer was motivated by its ease of use and open-source availability. The pipeline is fully reproducible and requires minimal computational resources, making it suitable for large-scale or frequently updated corpora.

[10] `https://www.nltk.org`

```
SENTENCE CONTENT:
At AUEB, a Committee for Equal Access of Persons with Disabilities
 ↪  and Persons with Special Educational Needs has been established.

SENTENCE METADATA:
{
  "sent_id": 2465
  "chunk_id": 188,
  "header": "XI. GENERAL INFORMATION FOR STUDENTS",
  "subheader": None,
  "paragraph": "Services for students with disabilities",
  "page": 105,
  "file_name": "Study Guide 2024-2025 ENG.pdf"
}

SENTENCE CONTENT:
Quality Assurance Unit
AUEB implements a quality assurance policy aimed at continuously
 ↪  improving the quality of its study programs, research
 ↪  activities, and administrative services, enhancing academic and
 ↪  administrative work, and improving overall university operation.

SENTENCE METADATA:
{
  "sent_id": 2533
  "chunk_id": 200,
  "header": "XI. GENERAL INFORMATION FOR STUDENTS",
  "subheader": None,
  "paragraph": "Quality Assurance Unit",
  "page": 108,
  "file_name": "Study Guide 2024-2025 ENG.pdf"
}

SENTENCE CONTENT:
Teaching and Learning Activities
Lectures (2 lectures of 2 hours weekly), group assignment.

SENTENCE METADATA:
{
  "sent_id": 2363
  "chunk_id": 175,
  "header": "IX. COURSE DESCRIPTIONS",
  "subheader": None,
  "paragraph": "Digital Learning Materials.",
  "page": 100,
  "file_name": "Study Guide 2024-2025 ENG.pdf"
}
```

Fig. 3.9.: Examples of extracted sentences with their associated metadata. Each sentence inherits the structural context of its parent chunk (header, subheader, paragraph, page, and file name) while being assigned a unique `sent_id`, enabling precise traceability within the Studies Guide.

A critical technical consideration was the preservation of metadata throughout the transformation: each sentence "inherits" all the contextual information of its parent chunk, augmented with its own globally unique `sent_id`.

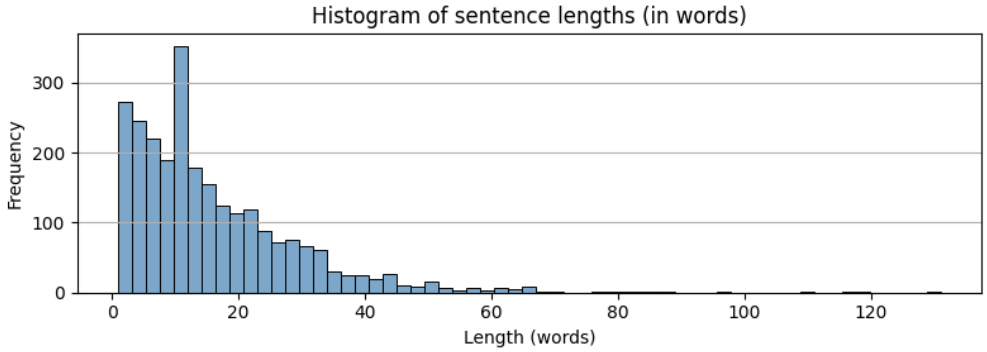

Fig. 3.10.: Distribution of sentence lengths (in words).

A detailed analysis of the extracted sentences reveals significant variability in their length, measured both in words and tokens. Across the 2,554 sentences in the dataset, the average sentence length is approximately 16.25 words, with a standard deviation of 13.82 words. The shortest sentence consists of a single word, while the longest sentence contains 131 words, highlighting the presence of substantial outliers. The median sentence length is 12 words, indicating that half of the sentences are relatively concise, while a smaller subset consists of much longer sentences. This distribution reflects the diversity in the Studies Guide's narrative style and content, encompassing everything from succinct informational statements to extended explanatory passages. Also, it hints at the fact that the sentence segmentation using NLTK may have produced some outliers.

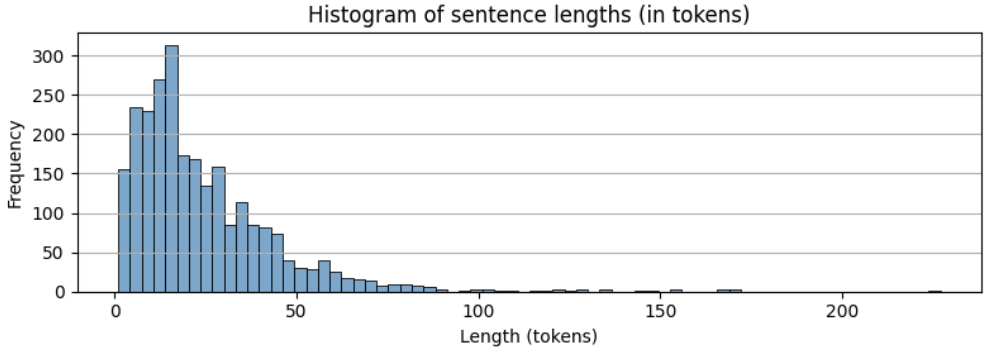

Fig. 3.11.: Distribution of sentence lengths (in tokens, as tokenized by the LLaMA 3.1 tokenizer).

Examining the sentences at the token level, the average length is 24.39 tokens, with a standard deviation of 20.49 tokens. The shortest sentence contains just 1 token, and the longest extends to 227 tokens, with a median of 19 tokens per sentence. This further demonstrates the presence of considerable variation in sentence complexity and informativeness, which is shaped by differences in linguistic structure and subject matter throughout the document.

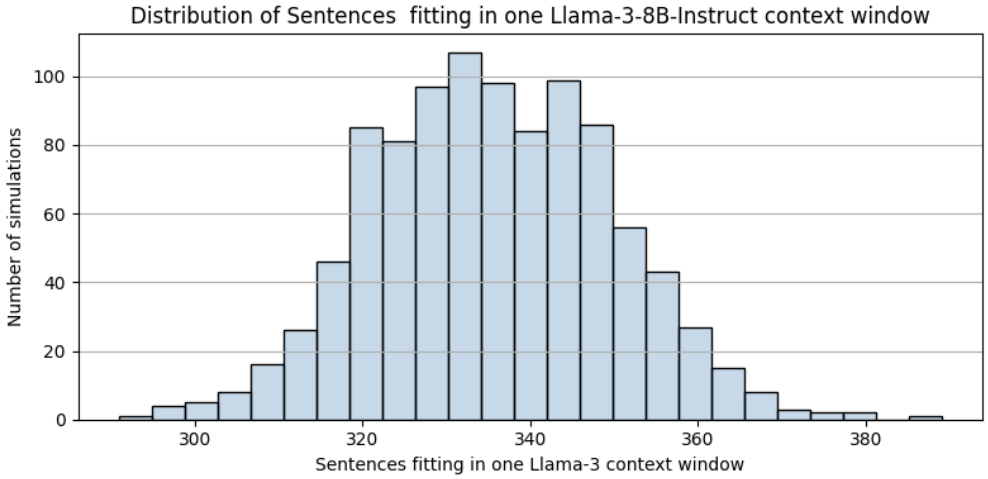

Fig. 3.12.: The number of Sentences that fit in `Llama-3.1-8B-Instruct`'s context window, on average. Note that the default context window with an 8,192k token limit is used here.

Figure 3.12 presents the distribution of the number of sentences that can be accommodated within a single context window of the LLaMA 3-8B-Instruct model, based on 1,000 randomized simulations. In each simulation, sentences from the corpus are sequentially added until the 8,192-token context limit is reached. The distribution is relatively symmetric and tightly clustered, with the majority of runs yielding approximately 310 to 355 sentences per window, and a mode centered around 335. This finding demonstrates the significant increase in granularity and context coverage afforded by sentence-level segmentation, enabling the inclusion of a much larger number of information units in a single retrieval operation. At the same time, it underscores the efficiency gains in packing the context window when working with more fine-grained textual units, while still being subject to the fixed-length constraints of LLMs.

3.2.3 Benefits and Limitations of Sentence-Level Document Representation

Sentence-level document representations offer several advantages in the context of retrieval-augmented question answering systems, particularly when precision and minimality are essential. By segmenting documents into individual sentences, the retrieval component can operate at a finer granularity, enabling highly targeted retrieval of relevant content. This increased specificity supports more focused and accurate answers, especially for questions that seek discrete facts or isolated pieces of information. Additionally, this granularity helps mitigate redundancy, as it reduces the inclusion of tangential or irrelevant context that might otherwise accompany broader textual units such as full paragraphs or sections. Importantly, the process of segmenting a document into sentences is relatively straight-

forward, computationally inexpensive, and reproducible with standard natural language processing tools, making it a practical choice for large-scale indexing and retrieval.

Despite their advantages, sentence-level document representations also present several important limitations that must be considered. A primary concern is the potential loss of context when sentences are treated as isolated units. Individual sentences may lack the necessary semantic context to support accurate answers for queries that require reasoning across sentence boundaries. Additionally, sentence segmentation does not eliminate all noise: not all sentences contribute equally to the informational value of a document. Some sentences may be overly long, vague, or heavily dependent on preceding content, thereby diminishing retrieval precision for specific information needs. Moreover, even after segmentation, many sentences lack semantic independence. For instance, elliptical sentences are a common occurrence in natural language, making certain sentences unintelligible or ambiguous when detached from their original context. These factors collectively constrain the standalone interpretability of sentence-level units and can impact both the retrieval and generation stages of the RAG system.

A statistical analysis of the sentence-level corpus further underscores the variability inherent in the document structure. On average, each chunk contains approximately 12 sentences, with the shortest chunk comprising just a single sentence and the longest extending to 112 sentences. The median number of sentences per chunk is 6, indicating that while a small number of chunks are quite large, the typical chunk is much more succinct. This pronounced spread is also reflected in the standard deviation, which is calculated at 13.41 sentences per chunk. Such heterogeneity is a direct consequence of the Studies Guide's diverse content, ranging from brief policy statements to extensive, multi-paragraph course descriptions and regulatory sections. These findings highlight both the strengths and potential retrieval challenges associated with sentence-level segmentation: while the approach facilitates high-precision retrieval, it must also contend with variable-length input units and a non-uniform distribution of information density across the corpus.

3.2.4 Motivation for an Alternative Granularity

Nevertheless, the persistent challenge of context-dependent sentences and implicit references motivated the development of a third, even finer representation: the creation of fully decontextualized propositions inspired by the work of [Che+24; Vla+25]. This next stage, described in the following section, seeks to maximize both granularity and semantic independence, ensuring every unit of information is self-contained and directly usable for question answering.

3.3 Proposition-Level Document Representation

In pursuit of maximally fine-grained, context-independent retrieval, the third document representation constructed for the Studies Guide corpus consists of **synthetically decontextualized propositions**. Each proposition represents a minimal, standalone statement that is fully interpretable on its own, eliminating the context loss issues associated with sentences and mitigating the problem of excessive information density associated with chunks.

3.3.1 Decontextualization Prompt and Methodology

The process begins by passing each chunk through an LLM, specifically the 'GPT-o1-mini' model, using a highly structured prompt (see Appendix A). This prompt instructs the model to first decompose complex, compound sentences into atomic, syntactically simpler statements. Additionally, any descriptive or qualifying content about named entities is separated into distinct propositions to ensure clarity. Crucially, all propositions are rendered fully decontextualized: the model is required to resolve pronouns, anaphoric expressions, and any implicit references so that each proposition stands independently, without reliance on external context. The output is returned as a JSON[11] structure, where each proposition is accompanied by its metadata, including a unique *prop_id*, the *chunk_id* from which it was derived, and all other previously mentioned chunk metadata.

This approach leverages the generative and reasoning capabilities of the LLM to extrapolate facts and synthesize fully context-independent statements that can serve as atomic knowledge units for a RAG system. The prompt was crafted manually by enhancing and tweaking the prompt of [Che+24] to fit custom needs.

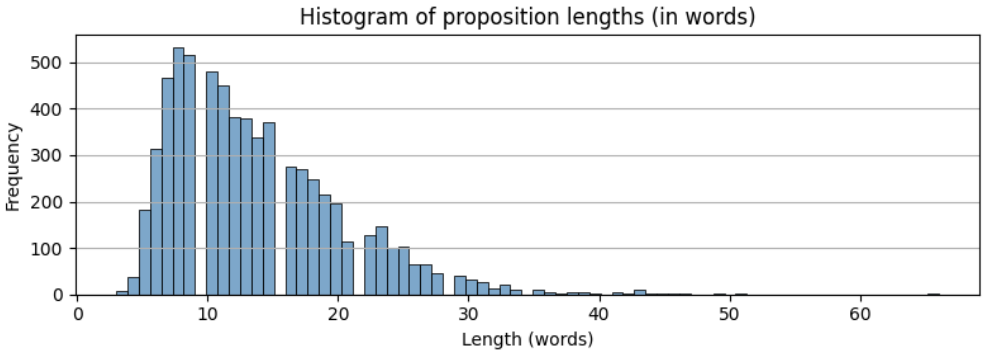

Fig. 3.14.: Distribution of proposition lengths (in words).

[11] JSON stands for *JavaScript Object Notation*, a lightweight, text-based format for representing structured data using name–value pairs and arrays.

```
ORIGINAL CHUNK CONTENT AND METADATA:
Inclusion of completed modules in student transcripts
If a student completes one or more modules, the modules are
 ↪  specified in that student's transcripts. Students are not
 ↪  obliged to declare their chosen modules beforehand; it is enough
 ↪  for them to declare the module at the time of their graduation
 ↪  application.

{
  "chunk_id": 104,
  "header": "VIII. DEGREE REQUIREMENTS",
  "subheader": VIII.2. Other Provisions (Applicable to All
   ↪  Students),
  "paragraph": "Inclusion of completed modules in student
   ↪  transcripts",
  "page": 33,
  "file_name": "Study Guide 2024-2025 ENG.pdf"
}

DECONTEXTUALIZED PROPOSITIONS CONTENT (with only unique prop_id
 ↪  shown for brevity):
["prop_id": 1571]

If a student of the Informatics Department of Athens University of
 ↪  Economics and Business completes one or more modules, then the
 ↪  modules are specified in the student's transcripts.
---
["prop_id": 1572]

Students of the Informatics Department of Athens University of
 ↪  Economics and Business are not obliged to declare their chosen
 ↪  modules beforehand.
---
["prop_id": 1573]

It is enough for students of the Informatics Department of Athens
 ↪  University of Economics and Business to declare their modules at
 ↪  the time of their graduation application.
```

Fig. 3.13.: Example of a chunk decomposed into decontextualized propositions.

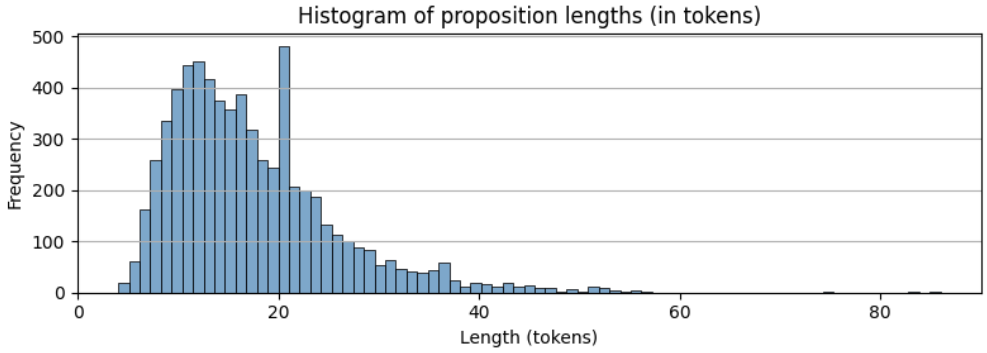

Fig. 3.15.: Distribution of proposition lengths (in tokens, as tokenized by the LLaMA 3.1 tokenizer).

A detailed statistical analysis of the extracted propositions underscores their compact and relatively uniform character compared to full sentences. For the **6,625 propositions** in the corpus, the average length is approximately 13.74 words, with a standard deviation of 6.57 words. The shortest proposition comprises just 3 words, while the longest extends to 66 words, reflecting a notable but bounded range. The median proposition length is 12 words, suggesting that most propositions are concise, self-contained factual statements, even though a subset is substantially longer. This statistical profile highlights the effectiveness of the proposition extraction process in distilling the Studies Guide's content into granular, decontextualized units of information.

When analyzed in terms of token count, the 6,625 propositions display a similar pattern of moderate variability. The average proposition contains 17.42 tokens, with a standard deviation of 8.28 tokens. The shortest proposition is 4 tokens in length, and the longest comprises 86 tokens. The median is 16 tokens per proposition, reinforcing the observation that most propositions are succinct and well-suited for fine-grained retrieval. The consistency in token length further facilitates the design of efficient retrieval and subsequent generation within the system.

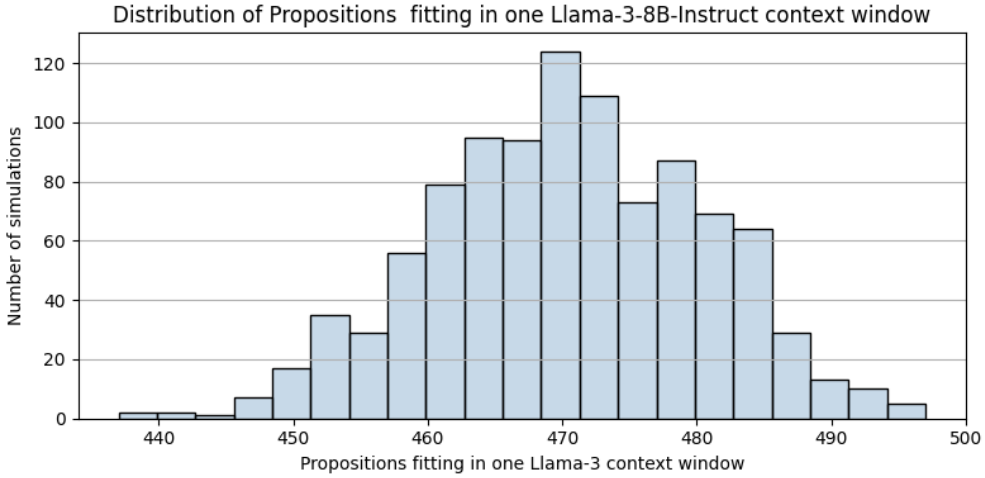

Fig. 3.16.: The number of Propositions that fit in `Llama-3.1-8B-Instruct's` context window, on average. Note that the default context window with an 8,192k token limit is used here.

Figure 3.16 displays the distribution of the number of propositions that can be accommodated within a single context window of the LLaMA 3-8B-Instruct model, based on 1,000 randomized simulations. In each simulation, propositions are sequentially added from the corpus until the 8,192-token context limit is reached. The distribution is fairly symmetric and tightly clustered, with the vast majority of runs fitting between approximately 450 and 485 propositions per window, and a clear mode near 470. This result illustrates the substantial increase in retrieval granularity enabled by proposition-level segmentation, which allows the system to include a much larger number of atomic, context-independent informational units within each query. It also highlights the advantage of finer granularity

for maximizing context utilization, while still respecting the inherent constraints of the model's fixed context window.

3.3.2 Strengths and Limitations of Proposition-Level Document Representation

The proposition-level representation's main advantage is that it offers **high precision** as decontextualized statements isolate chunks of information, enabling pinpoint retrieval and eliminating context leakage from neighboring sentences or paragraphs, thus allowing the system to respond to highly specific user queries with minimal noise or irrelevant information.

However, several limitations are associated with this granularity. Notably, the decontextualization process is contingent on the accuracy and consistency of the LLM. Imperfect decontextualization, dropped facts, or over-decontextualization can introduce noise or loss of nuance. While atomicity aids retrieval, it may also produce a larger number of small, sometimes redundant or semantically similar propositions. Processing each chunk through an LLM at scale incurs non-negligible computational and monetary costs, particularly for large documents or frequent collection updates.

```
{
"88": {
    "chunk_id": 89,
    "sentence_ids": [381, 382, 383, 384, 385, 386, 387, 388, 389,
    ↪  390, 391, 392, 393, 394, 395, 396, 397, 398],
    "proposition_ids": [1149, 1150, 1151, 1152, 1153, 1154, 1155,
    ↪  1156, 1157, 1158, 1159, 1160, 1161, 1162, 1163, 1164, 1165,
    ↪  1166, 1167, 1168, 1169, 1170, 1171]
  }
}
```

Fig. 3.17.: Example of a chunk and its corresponding sentence and proposition ids.

A statistical summary of the proposition-level corpus highlights the significant size increase compared to both chunk and sentence-level segmentations. On average, each chunk yields approximately 31 decontextualized propositions, with the number of propositions per chunk ranging from as few as 2 to as many as 224. The standard deviation, calculated at 32.62 propositions per chunk, underscores the pronounced variability in information density and structural complexity across the Studies Guide. This heterogeneity is a direct result of the diverse content and varying lengths of source chunks, as some sections of the document contain densely packed factual material, while others remain succinct and narrowly focused. The shift to proposition-level representation thus produces a substantially larger and more fine-grained retrieval corpus, amplifying both the opportunities for precise,

context-independent retrieval and the attendant challenges of increased fragmentation and redundancy. The full prompt used for decontextualization is provided in Appendix A.

3.4 Retrieval Granularities: Pros and Cons

Overall, three distinct collections were created by preprocessing the Studies Guide's PDF, one for each granularity:

- A collection of one or multiple-paragraph bodies of text, named chunks, derived through document-specific chunking.
- A collection of individual sentences, derived through sentence tokenization.
- A collection of decontextualized propositions, produced via processing each chunk through an LLM.

The resulting corpus forms the basis for retrieval and generation experiments of the RAG system across multiple document granularities.

Tab. 3.1.: Comparison of Retrieval Granularities

Granularity	Advantages	Limitations
Chunk	Semantic coherence, preservation of hierarchical structure	Variable length, uneven information density
Sentence	High precision, minimal redundancy, computationally inexpensive	Context loss, ambiguous standalone sentences
Proposition	Maximal precision, context independence	High computational cost, risk of redundancy

Each retrieval granularity possesses distinct strengths and limitations, summarized in Table 3.1, while Figure 3.18 provides a schematic overview of how the three document granularities are produced and how metadata is preserved across transformations.

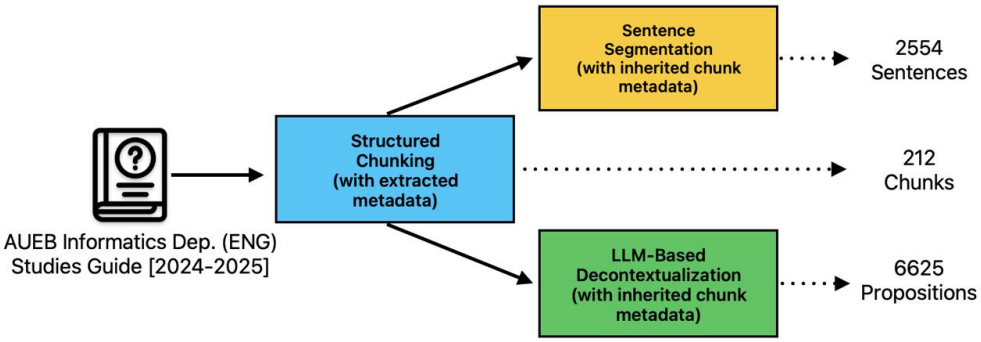

Fig. 3.18.: Overview of the document representation pipeline. The Studies Guide is first parsed into semantically coherent chunks with extracted metadata. Each chunk is then further split into sentences via sentence segmentation and transformed into decontextualized propositions via LLM-based decomposition. This yields three granularities (chunks, sentences, propositions) for retrieval in the proposed RAG system.

4 QA Pairs Dataset Creation

A core requirement of evaluating any information retrieval system is the existence of a gold-standard dataset of question-answer pairs annotated with their relevant source text passages. However, no such dataset currently exists for the AUEB Department of Informatics curriculum. To address this, synthetic QA data were generated based on the Studies Guide, resulting in aligned datasets at all three document granularities: chunk-level, sentence-level, and proposition-level.

The main rationale behind this approach lies in the need to objectively evaluate retrieval effectiveness at varying granularities, ensure sufficient coverage across different sections of the document, and simulate real-world student questions in a controlled and reproducible manner.

4.1 Synthetic Question Answer Generation

The generation of the synthetic QA pairs was facilitated through the use of a structured prompting pipeline that fed selected document subsets to an LLM and requested JSON-formatted QA pairs annotated with the relevant `chunk_id`'s used for the answer generation. This process was repeated for all three document granularities: chunks, sentences, and propositions. The prompt aimed to enforce clear formatting rules, minimum coverage guarantees, and realistic phrasing constraints tailored to the AUEB undergraduate student context (see Appendix A for the full prompt template).

4.1.1 Splitting Chunks to Random Subsets

To enable robust evaluation and cross-comparison between document granularities (chunks, sentences, and propositions), random subsets were generated in a systematic manner. The complete set of chunk documents was first randomly partitioned into 27 distinct subsets, each containing between five and ten chunks. Any remaining chunks after the initial division were redistributed as needed to maintain a balanced size across subsets. Sampling was conducted without replacement, so that each chunk appears in exactly one subset, resulting in disjoint, non-overlapping groups of chunks. This approach prevents redundancy in question generation and evaluation because using disjoint subsets ensures that no duplicate QA pairs are produced and reduces the risk of biased evaluation, as

repeated coverage of the same material could potentially leave certain sections of the Studies Guide underrepresented. Additionally, an explicit mapping file in JSON format was created to record the chunk, sentence, and proposition identifiers present in each subset. This mapping ensures full traceability and enables precisely aligned annotations, facilitating fair and direct comparisons across retrieval runs at different levels of document granularity.

4.1.2 Why Random Subsets?

The random subset construction strategy was deliberately chosen to encourage generalization, as this way the LLM is prompted with diverse, sometimes non-contiguous content, better simulating the unpredictability of real student queries. Furthermore, splitting the input corpus into relatively small, randomized subsets served to limit prompt lengths to manageable windows. This ensures that the LLM can (i) better attend to the input without truncation and (ii) likely produce focused, high-quality responses without suffering from prompt overload or degraded generation quality. Also, this strategy was preferred for its simplicity and reproducibility. With a fixed random seed, the same subset splits can be regenerated deterministically for validation, reruns, and ablation studies.

While alternative strategies for chunk subset construction, such as topic-based clustering or header-aligned grouping, might yield QA pairs with greater semantic coherence, they also risk grouping together overly similar content, which could reduce the diversity of questions and answers generated. In contrast, randomized splitting provides a pragmatic balance between coverage of the entire Studies Guide content and fairness in the distribution of that content across subsets. It should be noted, however, that this approach is not the only valid solution and that future work may benefit from a comparative analysis of sampling and splitting strategies to further optimize dataset construction and system evaluation.

Consequently, the number of chunks per subset ranges from 5 to 10, with a mode of 9 and an average of approximately 7.8 chunks per subset. This balance ensures uniform coverage while keeping the LLM prompt input length manageable. This balanced distribution not only supports fair evaluation across curriculum sections but also ensures that no single subset dominates the generation process. Such properties are essential for constructing a representative and generalizable QA dataset.

4.1.3 Instruction Prompt

The generation of synthetic QA pairs was guided by a highly structured instruction prompt that was fed to an LLM along with a specific document subset. The prompt was designed to elicit coherent, relevant, and faithfully grounded QA pairs while ensuring adherence to a strict output JSON schema. The main objective was to simulate realistic student interactions with a virtual assistant who was knowledgeable only about the content of the AUEB's Department of Informatics Studies Guide.

Each prompt consists of two components: (i) a subset of text chunks and (ii) a detailed set of generation constraints. The following constraints were intended to direct the model's output towards the desired structure and content, though the model may not always fully comply with them.

- **Coverage:** At least one question-answer pair must be generated for each input chunk. No chunk should be left unreferenced in the output.
- **Faithfulness:** All answers must be strictly grounded in the information provided by the input chunks.
- **Formatting:** The response must follow a strict JSON structure, with numerically indexed QA pairs. Each entry includes three fields: the user's question, the list of `chunk_ids` the answer draws from, and the assistant's response.
- **Language Style:** The questions must be phrased in natural language as if asked by an AUEB undergraduate student. Similarly, answers must be clear, polite, concise, and complete.
- **Minimal Output Quality Guarantees:** Each prompt explicitly requested at least 15 question-answer pairs to encourage depth, reuse of chunks where possible, and high response density.
- **Multi-chunk Reasoning:** The model was permitted (but not required) to combine information from multiple chunks into a single QA pair. However, it was explicitly instructed to ensure that each chunk appears in at least one unique QA pair.

By enforcing these constraints, the resulting QA pairs dataset, containing 438 QA pairs, maintains consistency, document traceability, and transparency in evaluation, making it straightforward to interpret retrieval results and their relation to the underlying source material. The full instruction prompt used can be found at Appendix A.2.

Examples of synthetic QA pairs are shown in Figure 4.1 below.

```
{
    "423": {
        "user": "What are the course modules available in the
        ↪  Informatics Department at AUEB?",
        "chunks_used_ids": [89],
        "system": "The course modules available in the Informatics
        ↪  Department at AUEB include Data Science, Operations
        ↪  Research, Applied Mathematics, Theoretical Computer
        ↪  Science, Systems and Networks, Software Systems, Data
        ↪  and Knowledge Management, and Cybersecurity."
    },
    "424": {
        "user": "How many courses must I complete to finish a module
        ↪  in the Informatics program?",
        "chunks_used_ids": [89],
        "system": "To complete a module in the Informatics program,
        ↪  you must take at least 5 out of the 7 to 9 courses
        ↪  offered in that module."
    },
    "425": {
        "user": "Is completing a module required for graduation in
        ↪  the Informatics Department?",
        "chunks_used_ids": [89],
        "system": "No, completing a module is not required for
        ↪  graduation. However, if you complete a module, it will
        ↪  be noted in your transcripts and diploma supplement
        ↪  awarded at graduation."
    }
}
```

Fig. 4.1.: Snippet from the JSON file containing the synthetic QA Pairs annotated with chunk ids.

4.2 Sentence and Proposition QA Alignment

To enable fine-grained evaluation of document retrieval performance beyond chunk-level reasoning, each synthetic QA pair was subsequently annotated with sentence-level and proposition-level justifications. This step aimed to establish a precise alignment between the assistant's answer and the specific sentence or proposition from which it was derived. This annotation process ensures direct comparability and consistency of evaluations across granularities. Consequently, three synthetically generated datasets are produced, each rigorously structured and systematically annotated with unique identifiers for streamlined evaluation. This annotation process was entirely LLM-driven, leveraging structured instruction prompts tailored to each granularity. Examples of synthetic QA pairs annotated with sentence and proposition ids are shown in Figures 4.2 and 4.3, respectively.

It should be noted that this two-step setup, first generating the QA pairs given the random chunk subsets and then creating alternate annotations of these with sentence and proposition identifiers, is not the only possible approach. In principle, the same prompt

that generated QA pairs at the chunk level could have been used to request sentence- or proposition-level justifications, since sentence and proposition subsets containing essentially the same information could be constructed by directly mapping each sentence or proposition to its corresponding chunk. Indeed, this strategy was attempted in the early stages of dataset creation, where each subset was passed through the same prompt separately for each granularity. While this method was both logical and feasible, manual inspection revealed that the quality and phrasing of the generated QA pairs varied noticeably in phrasing, informational scope, and overall quality depending on whether the LLM was conditioned on chunks, sentences, or propositions as input. Such variability risked introducing confounding factors into subsequent evaluations. To mitigate this risk and to ensure consistency across granularities, the final pipeline adopted the two-step process described above, in which QA generation and fine-grained justification annotation are handled as distinct stages. This separation provided clearer control over output quality and reduced the likelihood of evaluation results being compromised.

Given the limited human resources and time available, a large-scale manual validation of the sentence- and proposition-level annotations was not feasible. However, a small-scale manual inspection was conducted on a subset of the annotated QA pairs. This inspection suggested that the LLM annotations were generally adequate and aligned with the intended justifications, although this observation cannot be stated with a high degree of confidence. Consequently, it is acknowledged that some degree of inaccuracy may remain in the automatically produced annotations.

4.2.1 Sentence-Level Annotation

To establish a direct mapping between assistant answers and the specific textual segments they draw from, each chunk-level QA pair was extended with supporting sentence-level annotations. For each pair, a large language model was instructed to identify the **minimal set of sentence IDs** that most directly justified the given response.

The input to this process included:

- A single QA pair, consisting of the user's question, the assistant's answer, and the corresponding `chunk_ids`,
- A pool of candidate sentences filtered to only those belonging to the QA pair's `chunk_ids`.

The model was instructed to:

- Leave the `user` and `system` fields unmodified,

- Replace the `chunks_used_ids` with a new `sentences_used_ids` field listing the most relevant sentence identifiers,

- Avoid over-selection by including only those sentences that convey information directly reflected in the assistant's answer, excluding sentences that are merely contained in a chunk labeled as relevant but unused in the response.

Each prompt was limited to a *single QA pair*. That is, for each question-answer pair, a separate LLM call was made. The prompt contained only that QA pair, along with its corresponding filtered sentences. This design choice ensured tight focus and full attention within the model's context window, while also eliminating inter-pair interference or leakage across questions, preventing prompt overload or truncation, and supporting precise LLM cost tracking (measured as the sum of $ per input and output tokens) at the per-pair level.

This form of annotation aims to enable high-resolution retrieval evaluation, allowing us to compute metrics such as Recall@k and Mean Reciprocal Rank using sentence-level gold targets (as discussed later in Chapter 6). The full prompt template can be found in Appendix A.3.

4.2.2 Proposition-Level Annotation

In parallel with sentence-level annotation, a second fine-grained dataset was produced in which each QA pair was annotated with supporting proposition IDs. These propositions were derived from earlier decontextualization of the Studies Guide into atomic, self-contained statements (see Chapter 3).

The goal remained the same: identify the smallest subset of text units that directly justify the assistant's answer. However, instead of selecting from complete sentences, the model now operated over (decontextualized) `prop_ids` within the relevant `chunk_ids` (see Appendix A.4 for more information).

```
{
    "423": {
        "user": "What are the course modules available in the
        ↪  Informatics Department at AUEB?",
        "sentences_used_ids": [385, 386, 387, 388, 389, 390, 391,
        ↪  392],
        "system": "The course modules available in the Informatics
        ↪  Department at AUEB include Data Science, Operations
        ↪  Research, Applied Mathematics, Theoretical Computer
        ↪  Science, Systems and Networks, Software Systems, Data
        ↪  and Knowledge Management, and Cybersecurity."
    },
    "424": {
        "user": "How many courses must I complete to finish a module
        ↪  in the Informatics program?",
        "sentences_used_ids": [392, 393],
        "system": "To complete a module in the Informatics program,
        ↪  you must take at least 5 out of the 7 to 9 courses
        ↪  offered in that module."
    },
    "425": {
        "user": "Is completing a module required for graduation in
        ↪  the Informatics Department?",
        "sentences_used_ids": [382, 383],
        "system": "No, completing a module is not required for
        ↪  graduation. However, if you complete a module, it will
        ↪  be noted in your transcripts and diploma supplement
        ↪  awarded at graduation."
    }
}
```


Fig. 4.2.: Snippet from the JSON file containing the synthetic QA Pairs annotated with sentence ids.

For each QA pair:

- Only the propositions corresponding to the original `chunks_used_ids` were included as candidates.
- The model was instructed to insert a new `propositions_used_ids` field into the JSON output, leaving all other fields unchanged.
- It was required to include only propositions whose content was reflected in the assistant's answer, not all propositions derived from the chunk.

Each prompt once again annotated a *single* QA pair. This consistent design mirrored the sentence-level pipeline, allowing for easy comparison between granularities. Prompt formatting, cost tracking, and output validation were handled in a similar manner. This final annotated dataset enables proposition-level retrieval evaluation and offers a high-resolution lens on how precisely the assistant's answer maps back to factual atomic units. The corresponding prompt is detailed in Appendix A.4.

```
{
    "423": {
        "user": "What are the course modules available in the
        ↪  Informatics Department at AUEB?",
        "propositions_used_ids": [1155, 1156, 1157, 1158, 1159,
        ↪  1160, 1161, 1162],
        "system": "The course modules available in the Informatics
        ↪  Department at AUEB include Data Science, Operations
        ↪  Research, Applied Mathematics, Theoretical Computer
        ↪  Science, Systems and Networks, Software Systems, Data
        ↪  and Knowledge Management, and Cybersecurity."
    },
    "424": {
        "user": "How many courses must I complete to finish a module
        ↪  in the Informatics program?",
        "propositions_used_ids": [1163, 1164],
        "system": "To complete a module in the Informatics program,
        ↪  you must take at least 5 out of the 7 to 9 courses
        ↪  offered in that module."
    },
    "425": {
        "user": "Is completing a module required for graduation in
        ↪  the Informatics Department?",
        "propositions_used_ids": [1154, 1169, 1153],
        "system": "No, completing a module is not required for
        ↪  graduation. However, if you complete a module, it will
        ↪  be noted in your transcripts and diploma supplement
        ↪  awarded at graduation."
    }
}
```


Fig. 4.3.: Snippet from the JSON file containing the synthetic QA Pairs annotated with proposition ids.

As a result of this multi-stage annotation process, a synthetic QA dataset in three parallel variations—each reflecting a different level of document granularity was constructed. Specifically:

- The original version associates each QA pair with the `chunk_ids` that were provided as input context,
- The second version extends this with fine-grained `sentences_used_ids`, denoting the specific sentences that justify each answer,
- The third version further refines grounding precision by associating answers with the minimal set of `prop_ids` (fully decontextualized propositions) that support them.

4.3 Real-World QA Collection

While synthetic QA pairs enabled wide coverage and controlled alignment across document granularities, it was equally important to validate system behavior on questions that students naturally formulate. To that end, a small-scale **real-world QA dataset** of 60 QA pairs total was created by directly collecting authentic questions from undergraduate students of the Department of Informatics at AUEB.


```
{
"3": {
        "user": "Is it necessary to pass English to graduate?",
        "chunks_used_ids": [102, 105],
        "system": "It is not strictly necessary to pass English
        ↪  alone. You can satisfy the language requirement by
        ↪  obtaining a C2 certificate in English, German, or
        ↪  French, or by completing one of the 4th semester courses
        ↪  English IV, German IV, or French IV."
    },
"14": {
        "user": "What are the holidays during the academic year
        ↪  2024-2025?",
        "chunks_used_ids": [6],
        "system": "The official holidays are October 28th, 2024;
        ↪  January 6th, 2024; January 30th, 2024; March 3rd, 2024;
        ↪  March 25th, 2024; May 1st, 2024; and June 9th, 2024."
    },
"29": {
        "user": "What is the maximum number of ECTS units I can
        ↪  register for, given that I am in my 2nd year of
        ↪  studies?",
        "chunks_used_ids": [96],
        "system": "As a 2nd year student, you can register for a
        ↪  maximum of 46 ECTS units in each semester."
    },
"30": {
        "user": "I am in my 4th year and want to cancel a course
        ↪  that I passed in my 1st year of studies. Is that
        ↪  allowed?",
        "chunks_used_ids": [97],
        "system": "If you wish to be re-examined in a course you
        ↪  have passed last year, you can submit an application to
        ↪  the Department's Secretariat following the guidelines in
        ↪  a relevant announcement by the Rectorate. Keep in mind
        ↪  that the maximum number of times a student can request a
        ↪  re-examination is 4 and that the application must be
        ↪  submitted in the interval between the examination period
        ↪  you obtained the passing grade and the immediately
        ↪  following period where the course will be examined. You
        ↪  can participate in the course's examination anytime in
        ↪  the future. So, because you didn't pass the course the
        ↪  previous year, you cannot be re-examined."
    }
}
```


Fig. 4.4.: Snippet from the JSON file containing the real QA Pairs annotated with chunk ids.

4.3.1 Collection Process

A short-form questionnaire was circulated to students through multiple informal and formal channels:

- **Academic email service** managed by department administration,
- **Announcements** through internal online platforms and student group chats,
- **Peer-to-peer messaging**, where participants were encouraged to share the form with fellow classmates.

The questionnaire prompted students to write any study-related questions they might have about their curriculum, course prerequisites, specializations, or general academic policies, and optionally requested them to reference pages of AUEB's Studies Guide where relevant information to their query was mentioned.[12] Participants were explicitly informed that the questions would be used to improve an AI-powered assistant based on the Studies Guide.

4.3.2 Manual Annotation

The author manually answered each collected question based solely on the information in the official English Studies Guide. Responses were annotated with the corresponding `chunk_ids` from which the answers were derived. This step ensured that all answers remained strictly grounded in the knowledge base used for system training and evaluation. All annotations were performed by hand, without model assistance, to ensure reliability. A small number of questions, close to ten, were deemed by the author as either ambiguous, excessively complex for the current iteration of the system, or redundant due to significant overlap with other submissions. Some examples of omitted submissions are displayed in 4.5. These instances were excluded from the current dataset to maintain the clarity and quality of the evaluation set. However, it is important to note that such questions represent authentic challenges faced by students and will be addressed in future work.

4.3.3 Aligning Annotations Across Granularities

While manual chunk-level annotation sufficed for this limited dataset, repeating the same process at the sentence and proposition level would have been prohibitively time-consuming. Given the limited human resources (i.e., only the thesis author), the same automated LLM-based annotation pipeline employed for the synthetic data was applied.

[12]The questionnaire can be found here: `https://forms.gle/N7oWSF8Rz6Lo8dsx9`

- Each QA pair was annotated with `sentences_used_ids` using a sentence-level justification prompt (again see Appendix A.3),

- Similarly, `propositions_used_ids` were generated using the same proposition-level annotation prompt described earlier (see Appendix A.4).

This procedure resulted in a triplet of real-world QA datasets (chunk-level, sentence-level, proposition-level), enabling uniform evaluation across all retrieval granularities. Despite its smaller size compared to the synthetic dataset, this collection plays a crucial role in assessing how well the system handles natural, student-initiated queries.

```
{
- If I pass 42 courses, of which 20 complete 4 cycles (each course
↪  is assigned to only 1 cycle), will my average be the average of
↪  my passed courses divided by 42?

- Can elective courses from other departments be used to complete
↪  course modules?

- How many textbooks am I entitled to for free through the "Eudoxus"
↪  platform?

- I have completed all the mandatory courses, along with 3 core
↪  elective courses (orange) and 4 general elective courses
↪  (green). How many more ECTS do I need to graduate according to
↪  the 2020-2021 study guide?

- Recommend two courses from the 6th course module.

- Can I pass a course that has no written examination, but only
↪  mandatory assignments, in the September exam period?
}
```

Fig. 4.5.: Examples of queries collected from AUEB students, that were **not featured** in the real QA pairs dataset.

4.4 Comparative Analysis of the QA Datasets

This section analytically explores the rationale behind experimenting with various retrieval granularities (chunks, sentences, and propositions) by examining statistics such as average support and coverage of each granularity's synthetic and real-world QA datasets. The analysis highlights how finer granularities help in effectively pinpointing relevant information and discusses the significant impact of LLM output quality on both annotation and the crucial decontextualization process.

4.4.1 Synthetic QA Sets

For each QA pair in the evaluation process, the average number of retrieved supporting documents required at each document granularity was computed. In the synthetic QA sets, each QA pair was supported, on average, by two chunks, four sentences, or six propositions, respectively. This increase in the number of supporting units as granularity becomes more fine-grained reflects the fact that more atomic passages (such as sentences or propositions) are often needed to cover the full scope of an answer. While chunk-level retrieval provides broader context in fewer units, more fine-grained units require the retriever to piece together multiple, precise fragments in order to reconstruct the same information.

To capture this more clearly, the coverage of each QA dataset was measured, separately for synthetic and real QA pairs at each granularity. Coverage is defined as the percentage of all available units of a given granularity that are used at least once as justifications across the corresponding QA pairs dataset (see Equation 4.1 below).

$$\text{Coverage} = \frac{\text{Used Units}}{\text{Total Available Units}} \times 100\% \tag{4.1}$$

In synthetic datasets, chunk-level annotations unsurprisingly achieve almost complete coverage (99.06%), reflecting the broad but imprecise nature of chunk-based retrieval. On average, each answer is supported by approximately two chunks; however, these chunks typically contain many more sentences or propositions than are strictly required to answer the question. This excess highlights that chunk-level retrieval, while convenient, often brings in substantial irrelevant information. In contrast, coverage drops significantly at the sentence level (48.36%) and proposition level (36.20%), demonstrating that not all content within a chunk is necessary for effective answer support. More fine-grained granularities, such as sentence or proposition-level retrieval, provide more targeted and efficient retrieval by selecting only the specific information pertinent to each user query. This observation underscores the trade-off between coverage and precision inherent to different document granularities.

Tab. 4.1.: Coverage Statistics for Synthetic QA Sets

Granularity	Total Available IDs	Unique IDs Used	Coverage (%)
Chunks	212	210	99.1%
Sentences	2554	1235	48.4%
Propositions	6625	2398	36.2%

4.4.2 Real-world QA Sets

In the real-world QA set, each QA pair was supported, on average, by two chunks, four sentences, or eleven propositions. Notably, real-world QA pairs required substantially more fine-grained evidence at the proposition level—nearly double the number observed in the synthetic set, indicating that authentic student questions often span multiple discrete facts dispersed throughout the Studies Guide and are generally more complex than the synthetic ones. This was manually verified by the author on a subset of QA Pairs (this is evident in Figure 4.5.

Real-world datasets further reinforce the importance of granularity selection. Chunk-level coverage is lower (27.36%), indicating a narrower but still relatively broad capture of information. Sentence-level annotations show a significant decrease (7.48%), underscoring the need for precision when addressing detailed user-generated questions. Interestingly, proposition-level annotations slightly improve relative coverage (8.63%), demonstrating their flexibility in capturing detailed nuances within user queries, provided the initial decontextualization is accurate.

Tab. 4.2.: Coverage Statistics for Real-world QA Sets

Granularity	Total Available IDs	Unique IDs Used	Coverage (%)
Chunks	212	58	27.4%
Sentences	2554	191	7.5%
Propositions	6625	572	8.6%

4.4.3 Impact of LLM Quality

It is paramount to stress that the trustworthiness of the coverage results and the overall quality of the datasets heavily depend on the quality of the LLM's output. This dependency is particularly pronounced during the annotation processes for sentence- and proposition-level granularities, as well as in the decontextualization step necessary for proposition extraction. Hence, ensuring high-quality LLM outputs is crucial for producing reliable datasets and obtaining accurate coverage statistics at finer granularities.

5 System Design and Implementation

As discussed in detail in Section 2.1, RAG integrates a document retriever with an LLM to produce answers that are both contextually fluent and factually grounded. In this chapter, the components of modern RAG implementations, the retriever and the generator, are configured to implement a lightweight RAG system that will accurately and effectively answer questions according to the AUEB Studies Guide, tuning retrieval strategies at multiple granularities and generation hyperparameters to balance accuracy, coverage, and resource constraints.

Before delving into the live operation and query-time components of the system, it is useful to briefly recall the offline indexing procedures described in Chapter 3. The official Studies Guide of the Department of Informatics at AUEB was subjected to a preprocessing pipeline that segmented the content into three increasingly fine-grained levels of representation: chunk, sentence, and proposition. Each granularity serves a different purpose in the information retrieval pipeline and was indexed separately.

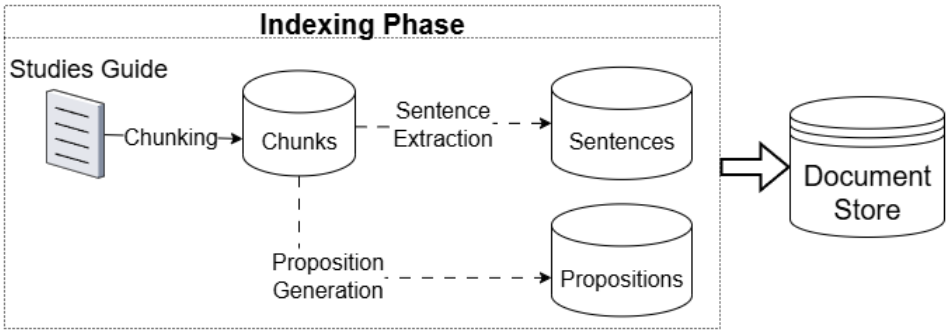

Fig. 5.1.: Overview of the Indexing process for the Studies Guide, and the formation of multiple granularities.

5.1 Retriever Setup

In the retrieval phase, experimentation involved sparse, dense, and hybrid retrieval architectures across chunk-, sentence-, and proposition-level document representations as retrieval units, enabling a systematic comparison of their performance.

5.1.1 BM25 Retriever

The system utilizes Langchain's[13] BM25Retriever[14] component to serve as its sparse retriever, which employs the widely used **Okapi BM25** ranking function, explained in Section 2.1.2.

The retriever is initialized directly by indexing the Studies Guide document, with the default (untuned) parameter values, $k1 = 1.5$, $b = 0.75$ (see Equation 2.1). These values are determined from extensive experimentation in the information retrieval community (such as TREC evaluations), which has shown that $k_1 \approx 1.2$–1.5 and $b \approx 0.75$ yield strong performance across diverse datasets [Ke22].

5.1.2 VectorStore Retriever

To complement precise keyword matching, the system incorporates dense semantic retrieval using the *all-MiniLM-L6-v2*[15] model in conjunction with a FAISS *IndexFlatIP*[16], which are encapsulated by Langchain's VectorStoreRetriever [17] component. The MiniLM encoder maps each query and document chunk into a 384-dimensional dense vector space. Thanks to an L_2 normalization step, these embeddings lie on the unit hypersphere, which simplifies cosine similarity computation to a direct inner product, hence the suitability of IndexFlatIP for the present task [JDJ21; RG19; Wan+20].

all-MiniLM-L6-v2: *MiniLM* is a highly efficient Transformer-based encoder distilled from BERT-base [Dev+19; Wan+20]. The *all-MiniLM-L6-v2* variant consists of only six Transformer layers and is further fine-tuned in the Sentence-BERT (SBERT) framework via supervised contrastive learning on over a billion sentence pairs [RG19; Wan+20]. The result is a compact model (22.7 million parameters, approximately 43 MB in float16 precision) that delivers high-quality, 384-dimensional semantic embeddings at high throughput (up to 14,000 sentences per second), making it particularly suitable for real-time, resource-constrained settings like ours.

IndexFlatIP from FAISS performs exhaustive, exact inner-product search, comparing the query embedding against every stored vector in the index. While computationally expensive at a large scale, an exhaustive scan is optimal for small to medium-sized corpora, such as the university-level Studies Guide dataset. This approach guarantees high recall without

[13] `https://www.langchain.com`
[14] `https://python.langchain.com/api_reference/community/retrievers/langchain_community.retrievers.bm25.BM25Retriever.html`
[15] `https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2`
[16] `https://faiss.ai/cpp_api/struct/structfaiss_1_1IndexFlatIP.html`
[17] `https://python.langchain.com/api_reference/core/vectorstores/langchain_core.vectorstores.base.VectorStoreRetriever.html`

approximation error and remains efficient given the manageable size of the knowledge base.

In summary, the dense retrieval module, through combining MiniLM embeddings and FAISS's IndexFlatIP, enables efficient and semantically meaningful retrieval over the Studies Guide corpus. As explained in Section 2.1.3, this component enhances the system's ability to identify and return conceptually relevant passages, even when users' queries differ lexically or structurally from the source text.

5.1.3 Ensemble Retriever

To combine both sparse and dense retrieval signals, the system implements a hybrid retrieval strategy using **Weighted Reciprocal Rank Fusion**, integrating result lists from both the BM25Retriever and VectorStoreRetriever mentioned previously, as explained in Section 2.1.4. This setup utilizes Langchain's EnsembleRetriever [18] component.

This retrieval component is configured with equal weights for each sub-retriever and employs the default and widely used fusion constant **c** value of 60. By using equal weighting, BM25Retriever and VectorstoreRetriever are each given the same influence in the fusion process, enabling balanced consideration of exact keyword matches and semantic similarity. A fusion constant value of 60 helps maintain fairness across retrieved lists and favors consensus among retrievers, a strategy shown to improve result robustness in hybrid RAG systems, as discussed previously in Section 2.1.4).

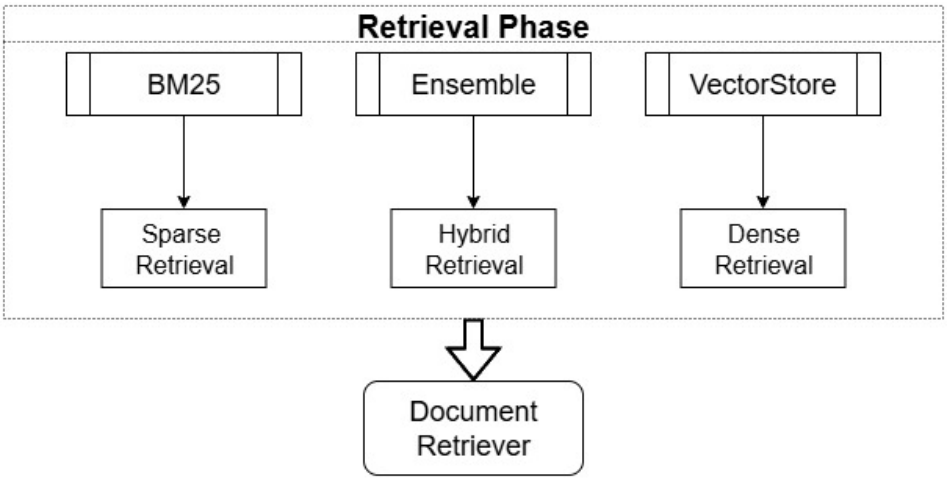

Fig. 5.2.: Overview of the retrieval architecture with support for sparse, dense, and hybrid search across document granularities.

[18] https://python.langchain.com/api_reference/langchain/retrievers/langchain.retrievers.ensemble.EnsembleRetriever.html

5.2 Generator Setup

In the proposed system, the answer generator is implemented using Meta's instruction-tuned model Llama-3.1-8B-Instruct[19], selected for its balance between performance and resource requirements. The model supports a context window of up to 8192 tokens (which can be extended to 128,000 after certain configurations on supported runtimes) and exhibits effective instruction-following behavior, aligning well with the demands of the RAG system. The model (and the entire RAG system) was developed to run on the two NVIDIA T4 GPUs (each with 15 GB of usable memory) offered via Kaggle's free 30-hour/week GPU time quota.

5.2.1 GPU Memory Management through Quantization

Quantization is a widely adopted model compression technique in the domain of large language models (LLMs), which operates by converting the model's weights and activations from high-precision floating-point representations, such as 32-bit or 16-bit floats, into lower-precision formats, typically 8-bit or even 4-bit integers. The principal goal of quantization is to reduce the overall memory footprint of the model and accelerate inference, thereby enabling the deployment of state-of-the-art models on hardware with limited computational resources and memory capacity. Among its main advantages, quantization significantly decreases the storage and memory bandwidth requirements, allows for faster inference due to more efficient integer arithmetic, and reduces energy consumption. However, these benefits come at the cost of certain disadvantages. Notably, quantization can introduce numerical errors and may degrade model accuracy, especially when aggressively reducing the precision to very low bit-widths without careful calibration [Has24].

To manage the available GPU memory efficiently, the model was quantized using the 16-bit bfloat16 precision. According to internal studies and third-party benchmarks, 8-bit quantization of Llama 3.1 8B Instruct retains nearly identical performance on reasoning and QA benchmarks compared to full precision (<1% accuracy loss), while halving memory usage [Kur+25b]. In the current setup, model weights are loaded in bfloat16 and can be further reduced to INT8 if GPU memory constraints require it. This allows deployment even on single GPU setups, although with a risk of noticeable performance degradation [Mek+25].

[19] `https://huggingface.co/meta-llama/Llama-3.1-8B-Instruct`

5.2.2 Generation Configuration.

Generation is controlled with the following, untuned hyperparameter values, empirically chosen to balance coherence and creativity:

$$\text{temperature} = 0.4, \quad \text{top\_p} = 0.9, \quad \text{max\_new\_tokens} = 1024$$

These settings ensure stable and accurate responses with minimal randomness, while allowing a sufficient expressive range. Sampling is enabled (`do_sample=True`), with `eos_token` as both stopping and padding token, and attention masks are correctly handled to ignore padding tokens. In this configuration, sampling is performed using nucleus (top-p) sampling with $p = 0.9$, which restricts the candidate token pool to the smallest set whose cumulative probability exceeds 0.9. This balances diversity and coherence by avoiding low-probability outliers while still allowing variability. Furthermore, the relatively low temperature value of 0.4 further sharpens the probability distribution, biasing generation toward higher-probability tokens and ensuring stable, deterministic-like outputs with only limited randomness. Such behavior is desirable for the proposed AUEB assistant, as it prioritizes factual accuracy and consistency over excessive creativity, ensuring that responses remain reliable and grounded in the Study Guide.

5.2.3 System Prompt

The system prompt serves as the foundational instruction set that integrates retrieval and generation in RAG systems. It describes in great detail the assistant's role, knowledge domain, and desired output style. This aims to enhance factual grounding by establishing robust guardrails that prevent adversarial jailbreak attacks, ensuring the model rejects malicious instructions, and maintains consistent performance across diverse queries.

The aim behind the design of the system prompt was to clearly state the model's name, task, and inform it about the specific knowledge that it has access to. It also enforces the behavior that the generated responses must be strictly grounded in the retrieved context. The assistant is instructed to handle out-of-domain queries gracefully by returning a safe fallback response when the question lies outside the AUEB Studies Guide scope. Finally, the prompt prescribes a clear and consistent style for the assistant's responses, polite, concise, and complete, by setting out explicit instructions for how answers should be formulated. This ensures that students always receive responses that are easy to understand and appropriate for an academic setting while avoiding ambiguity or unnecessary verbosity. These constraints are essential in RAG systems to minimize the risk of irrelevant or hallucinated responses, maintain user trust, and uphold the educational mission of the

assistant. Appendix A.5 includes the complete system prompt used to define the assistant's behavior.

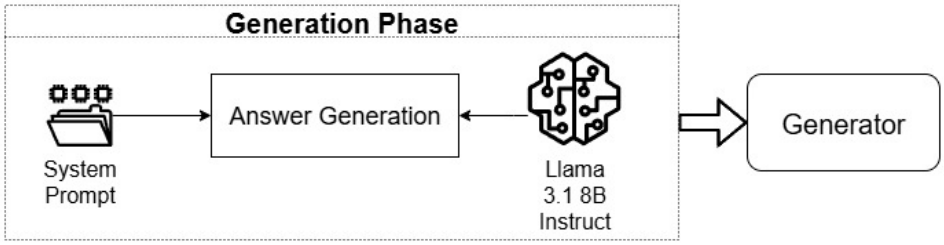

Fig. 5.3.: Overview of the Response Generation phase using an Instruct LLM, conditioned on top-k retrieved Studies Guide passages.

5.3 Query Flow

The overall operation of the RAG-based assistant can be separated into two main phases: the offline (pre-deployment) phase and the online (per-query) phase.

5.3.1 Offline Phase: Corpus Indexing, and System Configuration

Prior to deployment and actual user interaction, a substantial set of preparatory processes were conducted. First, the Studies Guide corpus is preprocessed and segmented at various granularities, specifically into chunks, sentences, and propositions, each representing a different unit of information for potential retrieval, as explained thoroughly in Chapter 3. For each granularity, the corresponding segments are encoded and indexed using the relevant retrieval methods: a sparse (BM25) index for lexical matching, a dense vector store (using ALL-MINILM-L6-V2 and a FAISS `FlatIP` Index), and an ensemble setup combining both through RRF. This approach allows for systematic experimentation and direct comparison between retrieval strategies, as each index can be used independently. During this phase, the active retrieval granularity and retriever method are chosen as fixed system settings. It must be noted that at present, only one combination is active at a time, ensuring experimental clarity and consistent evaluation.

5.3.2 Online Phase: Evidence Retrieval and Response Generation

Once deployed, the system enters the online phase, during which it processes individual user queries in real time. Upon receiving a query, the system routes it through the selected retrieval component. This component transforms the input into an appropriate representation, either a sparse vector or a dense embedding, depending on the retrieval architecture in use. The retriever then searches the indexed corpus at the selected granularity level (chunk, sentence, or proposition) and ranks candidate segments according to their relevance to the query, as determined by the retriever's internal scoring mechanism.

Once the *topk* most relevant segments have been identified, they are concatenated in a fixed, consistent order and formatted to serve as contextual input. This context is appended to the user's original query, and together with the predefined system prompt, the complete input is passed to the generator module. The current generation component, LLAMA-3.1-8B-INSTRUCT, is responsible for producing the final response. The model is explicitly instructed to maintain a helpful, polite, and concise tone, to ground its answers strictly on the retrieved context, and to refrain from conjecture or unsupported claims. In cases where the provided context is insufficient to confidently answer the query, the assistant is configured to return an appropriate fallback message.

The system supports operation at multiple document granularities and retrieval methods to allow for comparative analysis of how retrieval unit size and retrieval strategy influence answer accuracy, completeness, and overall system reliability. However, the system, as currently implemented, does not support dynamic selection or combination of retrieval granularities at query time, but this is identified as a promising avenue for future research (see Chapter 7.3 for more details).

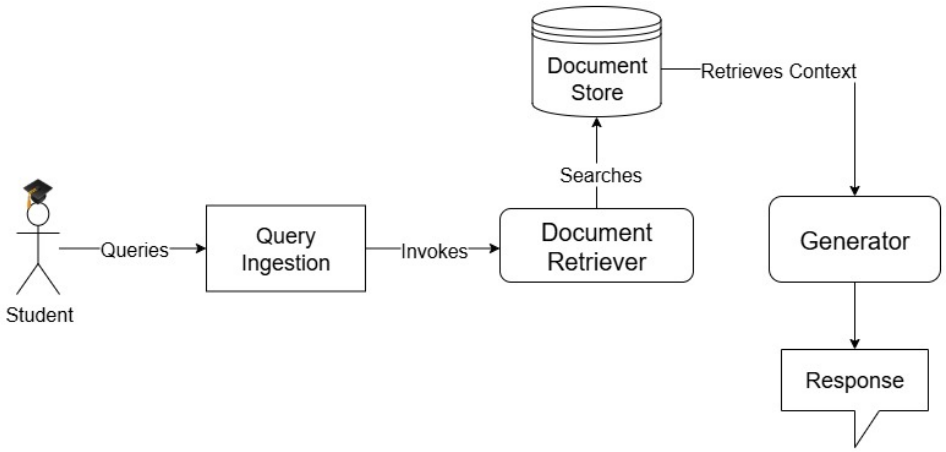

Fig. 5.4.: Complete end-to-end online query flow of the proposed RAG-based system, illustrating retrieval and generation phases.

6 Evaluation

6.1 Retrieval Evaluation

Evaluating the retrieval component is a critical step in the development and optimization of Retrieval-Augmented Generation (RAG) systems, as it directly influences the quality and factual grounding of the generated responses. The retrieval step is typically assessed using well-established Information Retrieval (IR) metrics, each designed to quantify different aspects of retrieval effectiveness and ranking quality. These metrics not only quantify how accurately and efficiently the system retrieves relevant documents, but also provide essential diagnostic signals for optimizing hybrid or fused retriever architectures and guiding ablation studies [Gan+25].
The retrieval evaluation employs binary relevance judgments, where each document is labeled as either relevant (labeled as 1) or non-relevant (labeled as 0).

6.1.1 Retrieval Metrics

Recall

Recall@k (R@k) measures the fraction of relevant documents successfully retrieved within the top-k results. A higher Recall@k ensures the system captures essential evidence across queries, which is crucial for a QA task. It is defined as:

$$\text{Recall@}k = \frac{|\{\text{relevant docs}\} \cap \{\text{top-}k \text{ retrieved}\}|}{|\{\text{relevant docs}\}|} \tag{6.1}$$

Its range is from 0 to 1, where 1 indicates that all relevant documents appear within the top-k, and 0 means none do. Higher values reflect better coverage of relevant evidence. This metric is widely used due to its straightforward interpretation in offline evaluation.

Mean Average Precision

Precision is the fraction of retrieved documents (up to a given rank) that are relevant to the query. In Equation 6.2, $P_q(i)$ denotes the precision at rank i, and $\mathrm{rel}_q(i)$ is a binary indicator that equals 1 if the document at rank i is relevant (q denotes a given query).

$$P_q(i) = \frac{\sum_{j=1}^{i} \mathrm{rel}_q(j)}{i} \tag{6.2}$$

Mean Average Precision (MAP@k) is the mean across queries of the **Average Precision (AP)**, which itself is the average of precision values computed at ranks where relevant documents appear, considering only the top-k retrieved results.

$$\mathrm{AP}_q@k = \frac{\sum_{i=1}^{k} P_q(i) \cdot \mathrm{rel}_q(i)}{\sum_{i=1}^{k} \mathrm{rel}_q(i)} \quad \mathrm{MAP@k} = \frac{1}{Q} \sum_{q=1}^{Q} \mathrm{AP}_q@k \tag{6.3}$$

MAP@K takes values in the range $[0, 1]$, with 1 indicating perfect ranking (i.e., all relevant documents ranked before any non-relevant ones within the cutoff k). This metric is particularly informative in scenarios where both the presence and high ranking of relevant documents are important. It is sensitive to both recall and ranking quality: ranking relevant documents higher improves MAP.

Precision thus measures the fraction of retrieved documents up to rank i that are relevant, and Average Precision (AP) extends this by averaging precision values only at the ranks where relevant documents occur, before Mean Average Precision (MAP) generalizes further across all queries.

Normalized Discounted Cumulative Gain

Normalized Discounted Cumulative Gain (NDCG@k) evaluates ranking quality with graded relevance and logarithmic rank discounting, ideal for scenarios with varying relevance levels. Nevertheless, it can still be used under binary relevance by assigning $rel_i \in \{0, 1\}$. Under binary relevance, it behaves similarly to MAP. NDCG@K normalizes DCG@K by the IDEAL DCG@K (IDCG@K). Formally, it is defined as:

$$\mathrm{DCG@}k = \sum_{i=1}^{k} \frac{2^{\mathrm{rel}_i} - 1}{\log_2(i+1)} \quad \mathrm{NDCG@}k = \frac{\mathrm{DCG}_k}{\mathrm{IDCG}_k} \tag{6.4}$$

Here $\mathrm{IDCG@}k$ is the ideal DCG@K when all relevant documents are optimally ordered. NDCG@K is normalized between 0 and 1: 1 represents a perfectly ranked list, and 0 indicates none of the retrieved documents are relevant. This metric balances the relevance of top-ranked items against their order in the list [Wan+13].

Mean Reciprocal Rank

Mean Reciprocal Rank (MRR) quantifies the average position of the first relevant result per query:

$$\mathrm{MRR} = \frac{1}{Q} \sum_{q=1}^{Q} \frac{1}{\mathrm{rank}_q^{(1)}} \tag{6.5}$$

where $\mathrm{rank}_q^{(1)}$ is the position of the first relevant document for query q. The maximum value is 1, achieved when every query's first result is relevant; the minimum approaches 0 as relevant documents are retrieved only at deep ranks or not at all. MRR is especially relevant for tasks requiring a single early correct answer.

6.1.2 Evaluation Framework

By using binary relevance, it is specified that each document is either fully relevant to a query or not, simplifying the ground truth setup. Each metric ranges from 0 to 1, with higher values indicating better performance. This multi-metric framework ensures a comprehensive understanding of retrieval accuracy and effectiveness. Recall@k evaluates coverage, MAP and NDCG@k assess both accuracy and ranking order, and MRR reflects the ability to retrieve at least one correct answer early. This multi-metric approach is employed on all three retriever configurations (sparse, dense, and ensemble/hybrid), thus supporting component-level diagnosis and retrieval comparison.[Gan+25; Yu+25]. Such multi-metric assessments are essential for identifying retrieval limitations, quantifying improvements, and correlating retrieval performance with downstream LLM answer quality [Gan+25; Yu+25; MGG25].

In order to pinpoint the strengths and weaknesses of different retrieval setups for the RAG system, all three retriever configurations are evaluated: sparse (BM25), dense (FAISS Flat IP Index with all-miniLM-l6-v2 normalized embeddings), and a hybrid ensemble using RRF on both sparse and dense methods. Each configuration is assessed at each level of retrieval granularity (chunks, sentences, and propositions) to understand how unit size influences evidence coverage and precision. At the chunk level, evaluation gauges the system's ability to surface broader contextual passages, which typically improves recall but introduces more non-essential text or outright noise. At the sentence level, evaluation examines more fine-grained units obtained by splitting chunks, which retain only limited local context and typically reduce noise compared to whole chunks, but may also lack essential context and therefore require more precise retrieval. At the proposition level, evaluation targets fully decontextualized, atomic facts that are well suited for precise answer grounding, though multi-part answers often require retrieving and composing several propositions [Che+24]. Examining results across these granularities, therefore, makes explicit the recall–precision trade-off that different unit sizes impose on the retriever.

By measuring Recall@k, MAP@k, NDCG@k, and MRR for each retriever and granularity, it becomes possible to diagnose whether lexical matching or semantic searching drives performance gains and also determine the optimal configuration for knowledge-intensive QA in a resource-constrained setting.

The results of the retrieval evaluation per granularity for each retriever on both datasets are presented below. *Note that all values reported in all the result tables are the metric averages across all queries of a dataset.*

Tab. 6.1.: Retrieval Metrics on the Synthetic QA Set (Chunk Level, 212 chunks in total)

Retriever	R@1	R@5	R@10	R@25	R@50	R@100	MRR	MAP@100	NDCG@100
VectorstoreRetriever	0.53	0.76	0.84	0.94	0.97	0.99	0.64	0.64	0.72
BM25Retriever	0.57	0.81	0.88	0.93	0.96	0.97	0.67	0.67	0.74
EnsembleRetriever	**0.63**	**0.87**	**0.93**	**0.97**	**0.99**	**1.00**	**0.74**	**0.74**	**0.80**

Tab. 6.2.: Retrieval Metrics on the Real-World QA Set (Chunk Level, 212 chunks in total)

Retriever	R@1	R@5	R@10	R@25	R@50	R@100	MRR	MAP@100	NDCG@100
VectorstoreRetriever	**0.52**	**0.69**	**0.80**	0.86	**0.93**	**0.98**	**0.67**	**0.63**	**0.72**
BM25Retriever	0.24	0.42	0.46	0.57	0.68	0.79	0.35	0.34	0.44
EnsembleRetriever	0.32	0.56	0.68	**0.90**	0.91	0.95	0.48	0.46	0.58

6.1.3 Retrieval Evaluation Results per Granularity

Analysis of Chunk-Level Retrieval Results

For the Synthetic QA set at the chunk level, the EnsembleRetriever achieved the highest performance, consistently surpassing both BM25 and VectorstoreRetriever across all metrics. This highlights the advantage of combining lexical and semantic retrieval methods at chunk granularity for synthetic data. The BM25 retriever performed well but was slightly behind the Ensemble, while VectorstoreRetriever showed competitive but slightly lower performance.

For the Real-world QA set at the chunk level, the VectorstoreRetriever notably outperformed both BM25 and the EnsembleRetriever at chunk granularity. BM25's performance significantly declined, reflecting the limitations of lexical matching when handling the complexity and variability inherent in real-world questions. The EnsembleRetriever, while still effective, did not reach the performance level of the VectorstoreRetriever in this context.

Tab. 6.3.: Retrieval Metrics on the Synthetic QA Set (Sentence Level, 2554 sentences in total)

Retriever	R@1	R@5	R@10	R@25	R@50	R@100	MRR	MAP@100	NDCG@100
VectorstoreRetriever	**0.21**	**0.34**	**0.39**	**0.49**	**0.59**	**0.68**	**0.43**	**0.30**	**0.41**
BM25Retriever	0.15	0.25	0.29	0.36	0.41	0.49	0.33	0.21	0.30
EnsembleRetriever	0.01	0.21	0.33	0.46	0.56	0.63	0.15	0.10	0.24

Tab. 6.4.: Retrieval Metrics on the Real-World QA Set (Sentence Level, 2554 sentences in total)

Retriever	R@1	R@5	R@10	R@25	R@50	R@100	MRR	MAP@100	NDCG@100
VectorstoreRetriever	**0.20**	**0.42**	**0.47**	**0.57**	**0.59**	**0.69**	**0.48**	**0.33**	**0.46**
BM25Retriever	0.08	0.16	0.20	0.25	0.30	0.39	0.19	0.13	0.20
EnsembleRetriever	0.02	0.19	0.28	0.44	0.56	0.62	0.14	0.11	0.25

Analysis of Sentence-Level Retrieval Results

For the Synthetic QA set at the sentence level, VectorstoreRetriever consistently outperformed BM25 and EnsembleRetriever. BM25 showed notably weaker performance, demonstrating clear limitations in lexical retrieval for shorter, context-dependent segments. The EnsembleRetriever also did not achieve significant improvements, suggesting challenges in effectively integrating multiple retrieval methods at this granularity. This can in part be attributed to the choice of weights assigned to each subretriever. As a result, the ensemble may not have been able to fully leverage the complementary strengths of its components, potentially limiting its overall effectiveness. A more systematic approach to hyperparameter tuning, specifically regarding the contribution of each retriever, could further enhance ensemble performance and represents a promising direction for future work.

For the Real-world QA set at the sentence level, VectorstoreRetriever again delivered the best results, significantly surpassing both BM25 and EnsembleRetriever. BM25 retrieval quality was consistently low, highlighting its inherent weaknesses in real-world sentence-level retrieval. Similarly, the EnsembleRetriever underperformed, confirming the difficulties associated with hybrid methods at finer granularities.

Tab. 6.5.: Retrieval Metrics on the Synthetic QA Set (Proposition Level, 6625 propositions in total)

Retriever	R@1	R@5	R@10	R@25	R@50	R@100	MRR	MAP@100	NDCG@100
VectorstoreRetriever	**0.13**	**0.33**	**0.46**	0.60	0.70	0.76	**0.47**	**0.33**	0.50
BM25Retriever	0.10	0.24	0.32	0.44	0.56	0.65	0.34	0.25	0.39
EnsembleRetriever	**0.13**	**0.33**	0.43	**0.61**	**0.73**	**0.81**	0.46	**0.33**	**0.51**

Tab. 6.6.: Retrieval Metrics on the Real-World QA Set (Proposition Level, 6625 propositions in total

Retriever	R@1	R@5	R@10	R@25	R@50	R@100	MRR	MAP@100	NDCG@100
VectorstoreRetriever	**0.10**	**0.31**	**0.43**	**0.57**	**0.66**	**0.75**	**0.38**	**0.29**	**0.45**
BM25Retriever	0.08	0.09	0.14	0.18	0.23	0.32	0.18	0.11	0.18
EnsembleRetriever	0.06	0.19	0.25	0.48	0.62	0.74	0.24	0.18	0.34

Analysis of Proposition-Level Retrieval Results

For the Synthetic QA set at the proposition level, VectorstoreRetriever demonstrated clear superiority, indicating its effectiveness in capturing detailed semantic nuances. BM25 retrieval was consistently weaker, reflecting challenges at fine-grained semantic retrieval. The EnsembleRetriever provided competitive results but did not exceed the performance of the dense retrieval method.

For the Real-world QA set at the proposition level, VectorstoreRetriever maintained dominance, reinforcing its strong semantic understanding capability. BM25 showed severely limited performance, indicating significant shortcomings in processing finely detailed semantic content. The EnsembleRetriever offered moderate results, revealing complexities in effectively combining retrieval signals at the proposition level.

6.1.4 Summary of Observations

VectorstoreRetriever consistently exhibited superior performance across all granularities, with its advantage becoming especially pronounced in real-world scenarios. Its robustness in semantic retrieval allowed it to achieve the highest scores in nearly all metrics—particularly Recall@k, MAP, and NDCG@100—across both sentence and proposition-level evaluations. The chunk-level performance of VectorstoreRetriever on the Real-world QA set emerged as the single most effective retrieval configuration, combining high MRR and top-k recall, hinting that the custom parsing process following the strict structure of the Studies Guide document aids retrieval as it contains dense context about a specific theme, even if it may also carry some non-essential information.

BM25Retriever demonstrated only limited competitiveness, achieving relatively strong performance in the synthetic chunk-level setting, likely due to a substantial lexical similarity between synthetic questions and indexed content. However, its performance sharply declined in finer granularities and in the real-world QA set, where semantic variability and paraphrasing reduced its effectiveness. Notably, its poor MAP@100 and NDCG@100 scores at the sentence and proposition levels reflect an inability to rank relevant evidence effectively under these conditions.

The EnsembleRetriever, which integrates lexical and dense signals via RRF, performed best in the synthetic chunk-level setting, outperforming both individual retrievers in terms of MRR and Recall@k. Nevertheless, its effectiveness diminished at the sentence and proposition levels, particularly in real-world queries. This suggests that while fusion methods can yield gains under controlled input distributions, they struggle to generalize when retrieval units become finer and queries more context-dependent—possibly due to suboptimal fusion weighting or redundancy amplification.

Overall, chunk-level retrieval consistently provided the best trade-off between contextual sufficiency and retrieval accuracy. When paired with VectorstoreRetriever, it not only achieved the highest aggregate scores but also exhibited stable performance across metrics and datasets. Sentence and proposition-level retrievals, although valuable for reducing context length and targeting atomic information, underperformed. These findings collectively underscore that, under current constraints, chunk-level semantic retrieval constitutes the most effective configuration for accurate and contextually appropriate evidence retrieval from the knowledge base.

6.2 Generation Evaluation

The systematic evaluation of generated answers in knowledge-intensive QA systems is a multifaceted challenge that necessitates both a quantitative and a qualitative approach to account for the diversity of language output. To comprehensively assess the effectiveness of the answer generation module within the RAG-based student assistant, a comprehensive suite of evaluation metrics is employed.

These include the traditional reference-based, the embedding-based, and the more modern LLM-based text generation metrics. **Reference-based metrics** compare model outputs to human-written or LLM-generated correct answers, called references or reference answers, using n-gram overlap and surface-level similarity. **Embedding-based metrics** assess semantic alignment in high-dimensional representation spaces, and **LLM-based metrics** leverage the reasoning abilities of advanced LLMs to provide holistic assessments of answer quality, guided by structured scoring guidelines.

6.2.1 Traditional Generation Evaluation Metrics

Traditional generation evaluation metrics fall into two complementary paradigms: classic NLG (Natural Language Generation) metrics, such as BLEU, METEOR, and ROUGE, that quantify surface-level token overlap between generated and ground-truth answers, and embedding-based metrics like BERTScore, which measure semantic similarity by comparing their contextual embeddings from pre-trained language models.

Classic NLG metrics, sometimes referred to as string-based or overlap-based evaluation metrics [Hwa+23], constitute the most established framework for the automatic assessment of machine-generated text. The central premise of these metrics is to compare the model's output, the generated answer, also known as the candidate answer, against one or more human-written correct answers (i.e., treated as ground-truth answers), known as reference answers. Due to recent advancements in LLM capabilities, LLM-generated answers can be used as a substitute in cases where human annotators are unavailable or prohibitively

expensive. Answer quality is then judged in terms of the degree of overlap, measured at the level of surface forms such as words, n-grams, subsequences, or exact tokens.

The paradigm these metrics follow is grounded in the tradition of machine translation and summarization evaluation, where reference texts are treated as gold standards, and model outputs are evaluated based on their ability to reproduce the wording, phraseology, or ordering of the reference texts. Metrics like BLEU, ROUGE, and METEOR are thus said to follow a token-matching paradigm, which prioritizes lexical similarity over semantic equivalence or factual correctness. As a result, these metrics are most effective in settings where the space of acceptable answers is relatively constrained and reference texts provide good coverage of valid variations.

A series of systematic studies have demonstrated that their correlation with human judgments is generally weak, especially in open-ended generation tasks, dialog systems, or knowledge-intensive question answering, where there may be many equally valid phrasings or correct answers not present in the reference set [Liu+16; Low+17; NK18]. Furthermore, these metrics are inherently limited to scenarios where at least one high-quality reference is available for each input, which is often not the case for real-world, domain-specific, or conversational datasets.

BLEU

The **BLEU (Bilingual Evaluation Understudy)** metric introduced by Papineni et al. [Pap+02] is a corpus-level n-gram precision score combined with a brevity penalty to discourage overly short hypotheses. It calculates the geometric mean of modified n-gram precisions (usually uni- to 4-grams, called BLEU-4) as follows:

$$\text{BLEU} = \text{BP} \cdot \exp\left(\sum_{n=1}^{4} \frac{1}{4} \ln p_n\right) \tag{6.6}$$

where p_n is the modified (clipped) n-gram precision and BP is the brevity penalty as defined by Papineni et al. [Pap+02]. Its strength lies in its efficiency, language independence, and strong correlation with human judgments across diverse language pairs [Pap+02].

Modified n-gram precision refers to clipping the count of each n-gram in the candidate translation to the maximum number of times it appears in any reference translation; this prevents inflated precision from over-repetition of n-grams [Pap+02].

Segment-level BLEU applies the metric at the sentence level rather than across a corpus. Since higher-order n-grams may be missing in short sentences, this often leads to zero scores even when the overall quality is acceptable [Pap+02]. Chen and Cherry [CC14] addressed this issue by introducing smoothing techniques, such as epsilon smoothing

(which adds a small constant ε to unseen n-gram counts) or floor smoothing (a lower bound on precision estimates). These smoothing techniques were adopted by NLTK to produce more stable sentence-level BLEU results.

These techniques form the basis of the `nltk.translate.bleu_score` module, in particular the `sentence_bleu` function and the `SmoothingFunction` class, which are employed in the evaluation scheme.

ROUGE

The **ROUGE[20] (Recall-Oriented Understudy for Gisting Evaluation)** metrics were introduced by Lin [Lin04] as a package for automatic summarization evaluation. These metrics focus on coverage—the degree to which the generated answer overlaps with reference answers—using various n-gram and sequence measures.

ROUGE-N computes the recall of overlapping n-grams between a candidate answer and one or more references:

$$\text{ROUGE-}N = \frac{\sum_{S \in \text{Refs}} \sum_{gram_n \in S} \text{Count}_{\text{match}}(gram_n)}{\sum_{S \in \text{Refs}} \sum_{gram_n \in S} \text{Count}(gram_n)} \tag{6.7}$$

where N is the n-gram length. In the following generation evaluation metric tables, ROUGE-1, ROUGE-2, ROUGE-3, and ROUGE-4 (i.e., unigram to 4-gram recalls) are all reported to capture increasing spans of lexical overlap [Lin04].

ROUGE-L is based on the **longest common subsequence (LCS)** between the candidate and reference, capturing sentence-level structure without requiring strict adjacency. It combines LCS-based recall and precision into an F-measure, rewarding in-sequence matches even across gaps [Lin04].

METEOR

The **METEOR (Metric for Evaluation of Translation with Explicit Ordering)** metric introduced by Lavie and Agarwal [LA07] was designed to address certain limitations of BLEU, particularly its limited use of recall and weak sentence-level correlation with human judgment. METEOR aligns unigrams between system-generated output and reference answers using multiple matching modules (exact surface form, stemming, synonymy) and favors alignments with fewer word order violations.

[20] In the following metric tables, all ROUGE metrics are abbreviated as "R".

Once an optimal alignment is established, METEOR computes unigram precision $P = \frac{m}{w_t}$ and recall $R = \frac{m}{w_r}$, where m is the number of matched unigrams, w_t is the hypothesis token count, and w_r is the reference token count. These are combined using a weighted harmonic mean $F_{\text{mean}} = \frac{PR}{\alpha P + (1-\alpha)R}$, with greatest emphasis on recall (commonly $\alpha = 0.1$, giving recall nine times more weight).

To account for fluency and word order, METEOR introduces a penalty based on fragmentation: the ratio of aligned chunks to matches, elevated and scaled by tunable parameters β and γ. The final score for each segment is:

$$\text{METEOR} = F_{\text{mean}} \cdot \left(1 - \gamma \cdot \left(\frac{\text{chunks}}{m}\right)^{\beta}\right) \tag{6.8}$$

Lavie and Agarwal [LA07] showed METEOR yields significantly higher sentence-level correlation with human judgments than BLEU (up to 0.40 vs. 0.22), and tunable parameters ($\alpha = 0.1$, $\beta = 3$, $\gamma = 0.5$) and matching modules enhance its effectiveness. It remains one of the most reliable metrics for evaluating answer generation in QA systems, particularly when grammatical ordering and semantics are important.

BERTScore

BERTScore, introduced by Zhang* et al. [Zha+20], computes semantic similarity between a candidate answer and its reference by leveraging contextual embeddings from a pre-trained BERT model. Rather than relying solely on exact n-gram matches, BERTScore aligns tokens via cosine similarity in the embedding space. Specifically, the process begins by passing both the candidate and reference answers through the model to obtain their token-level vector representations. Let $\mathbf{r}$ denote the sequence of embeddings for the reference answer, and $\mathbf{c}$ for the candidate answer. BERTScore then defines precision and recall at the level of contextualized tokens by measuring the maximum *cosine similarity* between each token embedding in one sequence and all token embeddings in the other [Zha+20]:

$$P_{\text{BERT}} = \frac{1}{|c|} \sum_{c_j \in c} \max_{r_i \in r} \cos(r_i, c_j), \tag{6.9}$$

$$R_{\text{BERT}} = \frac{1}{|r|} \sum_{r_i \in r} \max_{c_j \in c} \cos(r_i, c_j), \tag{6.10}$$

where $\cos(\cdot, \cdot)$ denotes cosine similarity between token embeddings. The overall BERTScore F_1 is then computed as the harmonic mean of P_{BERT} and R_{BERT}, analogous to the traditional F_1 formula:

$$BERTScore\text{-}F_1 = 2 \cdot \frac{P_{BERT} \cdot R_{BERT}}{P_{BERT} + R_{BERT}} \tag{6.11}$$

BERTScore values theoretically range from -1 to 1, corresponding to the cosine similarity interval, but according to Hanna and Bojar [HB21], empirical scores for high-quality outputs typically occupy the upper end of this spectrum. This range reflects the metric's underlying sensitivity to both semantic alignment and the quality of the learned representations.

This metric correlates more strongly with human judgments in generation tasks, such as machine translation and image captioning, than traditional metrics like BLEU and ROUGE, offering robustness to paraphrasing and semantic variation [Zha+20].

In the evaluation setup, BERTScore-F_1 is computed using the BERT-BASE-UNCASED model via the Hugging Face TRANSFORMERS and BERT_SCORE libraries. This approach ensures a semantically grounded evaluation of generated answers, which is particularly important for assessing deeper meaning and paraphrase fidelity beyond surface-level matches.

6.2.2 LLM-based Generation Evaluation Metrics

LLMs enable two complementary paradigms for evaluating generated answers: intrinsic confidence estimation via perplexity and external judgment via LLM-as-Judge.

Perplexity

Perplexity (commonly abbreviated as **PPL**) is widely used to quantify how well a language model predicts a given sequence. In RAG scenarios, it is used to complement reference-based metrics, as, unlike them, it does not directly measure the quality of generated text. Instead, it assesses the "confidence" or "surprise" of a language model in predicting the next word in a sequence of words. It is defined as the exponentiated average negative log-likelihood (i.e., average cross-entropy [MMZ23]) of a sequence of tokens. Formally:

$$\mathrm{PPL}(X) = \exp\left(-\frac{1}{N}\sum_{i=1}^{N}\log p_\theta(x_i \mid x_{<i})\right), \tag{6.12}$$

where N is the number of tokens and $p_\theta(x_i \mid x_{<i})$ is the model probability for token x_i given its preceding context. Equivalently, it can be viewed as the $\exp(\textsc{CrossEntropy})$ per token, with lower PPL indicating that the model is assigning higher probability mass to the correct tokens [CS24].

In the evaluation scheme, PPL is computed as the exponentiated average cross-entropy loss over the ground-truth answer tokens, excluding the query and ground-truth context tokens, which are masked in the input. The model's per-token cross-entropy loss on

the answer is weighted by the number of answer tokens to recover the total negative log-likelihood for each QA pair. These totals are then summed across all QA pairs and divided by the overall number of answer tokens, yielding the average cross-entropy loss $\bar{\ell}$. Finally, PPL is obtained as $\exp(\bar{\ell})$. In this thesis, PPL essentially reports how confident, on average, the model was in producing the gold answers, given the right context. Under this gold-context setup, the calculation is independent of the retriever configuration or the value of k.

As future work, the evaluation framework could also be extended to compute PPL under a full RAG setup, where the retrieved context directly influences the probabilities the LLM assigns to the ground truth answer tokens. Another direction closely aligned with this is the Semantic Perplexity (SePer) framework introduced by Dai et al. [Dai+25], which quantifies retrieval efficacy in RAG systems by measuring the reduction in semantic perplexity resulting from the retrieved context.

Perplexity offers an intrinsic, language-agnostic, and computationally efficient metric, reflecting a model's internal confidence over its generation. However, it is influenced by tokenization, context length, and vocabulary size, making it unsuitable as a standalone metric to evaluate the generation quality, but useful nonetheless.

Moreover, while PPL is ideal for assessing generation fluency and model calibration, Hu et al. [Hu+24] showed that it does not reliably reflect an LLM's ability to understand long texts, as it focuses on the predictability of individual words within a sequence, essentially measuring how well the model captures local language patterns. Thus, it can yield low values while the model still struggles with complex, lengthy passages.

For these reasons, generation evaluation is complemented with traditional NLG metrics, such as BLEU, ROUGE, and METEOR, and embedding-based metrics such as BERTScore. Last but not least, LLM-based metrics are also employed to capture other nuances that affect the quality of generated answers, like factual accuracy, helpfulness, and coherence.

LLM-as-Judge

The fundamental premise of LLM-as-Judge is to leverage the contextual reasoning and task generalization of state-of-the-art models (such as GPT-4) to assess generated answers based on a set of human-interpretable criteria. These typically include relevance to the user query, factual accuracy, fluency, coherence, and helpfulness, which are either explicitly provided in the evaluation prompt or learned through fine-tuning [Li+24]. In pointwise evaluation, the LLM is asked to independently grade each answer, assigning scores or labels according to each criterion; in pairwise evaluation, it compares two answers and determines which

one is superior or declares a tie; listwise and more advanced multi-turn settings are also increasingly explored for ranking and conversation evaluation tasks [Li+24; Zhe+23].

The evaluation scheme employs a pointwise LLM-as-Judge tactic, where the LLM, GPT-4o-mini in the present case, is presented with the user query, the retrieved context, and the generated answer, and instructed via a carefully designed prompt to provide a structured, multi-dimensional evaluation. This setup enables fine-grained, scalable, and cost-effective assessment, especially when expert human annotation is prohibitively expensive or infeasible for large evaluation sets.

Specifically, through the structured prompt, the LLM is instructed to grade each generated answer based on five distinct metrics, on a scale from 1 through 5, each targeting a critical aspect of answer quality. **Relevance** assesses whether the response directly addresses the user's query and utilizes the provided context appropriately. **Factual Accuracy** judges the correctness and verifiability of the information presented, penalizing hallucinations or unsupported statements. **Fluency** measures the grammatical correctness and the well-formedness of the answer, ensuring that the response is both easy to read and natural in its phrasing. **Coherence** evaluates the logical flow and structural organization of the response, measuring whether the answer logically follows from both the query and the retrieved context. Lastly, **Helpfulness** reflects the practical utility of the answer, emphasizing whether it provides actionable, informative, or otherwise useful guidance to the student. For each generated answer, the LLM is instructed to output a JSON-formatted object with a discrete score (1–5) for every metric, as well as a brief explanatory comment justifying its ratings. This approach is intended not only to support quantitative aggregation and comparison across models or systems but also to enable qualitative error analysis, making the evaluation process both scalable and interpretable. Incorporating the LLM-as-Judge paradigm in the evaluation scheme enables comprehensive, fine-grained evaluation at scale, serving as a practical proxy for expert human annotation.

The full instruction prompt used for LLM-based generation evaluation can be found at Appendix A.6.

The different values of top-k presented in the tables, specifically top-k $= 10$ for chunks and top-k $= 100$ for sentences and propositions, are a direct consequence of hardware limitations. While evaluating generation performance, significant GPU memory limitations that restricted the chunk-level experiments to top-k $= 10$ were encountered. In contrast, due to the smaller memory footprint required by sentence and proposition-level representations, it was possible to conduct more extensive evaluations with top-k $= 100$, and would have further increased this value if additional computational resources and time had been available.

The results of the generation evaluation (both Traditional metrics, LLM-based metrics, and PPL values) per granularity for each retriever on both datasets are presented below:

Chunk Granularity

Synthetic QA Set

Tab. 6.7.: Traditional Generation Metrics on the Synthetic QA Set (Chunk Level, top k = 10)

Retriever	BLEU	METEOR	R-1	R-2	R-3	R-4	R-L	BERTScore
BM25Retriever	0.26	0.60	0.61	0.44	0.34	0.27	0.54	0.75
VectorStoreRetriever	**0.33**	**0.65**	**0.66**	**0.51**	**0.41**	**0.33**	**0.60**	**0.79**
EnsembleRetriever	0.26	0.61	0.62	0.46	0.35	0.28	0.55	0.76

Tab. 6.8.: LLM-as-Judge Generation Metrics on the Synthetic QA Set (Chunk Level, top k = 10)

Retriever	Factual Accuracy	Relevance	Fluency	Coherence	Helpfulness
BM25Retriever	4.52	4.58	4.97	4.61	4.50
VectorStoreRetriever	4.39	4.50	4.97	4.53	4.39
EnsembleRetriever	**4.59**	**4.66**	**4.98**	**4.68**	**4.58**

Real-World QA Set

Tab. 6.9.: Traditional Generation Metrics on the Real-World QA Set (Chunk Level, top k = 10)

Retriever	BLEU	METEOR	R-1	R-2	R-3	R-4	R-L	BERTScore
BM25Retriever	0.11	0.37	0.40	0.23	0.15	0.11	0.34	0.63
VectorStoreRetriever	**0.17**	**0.45**	**0.47**	**0.30**	**0.21**	**0.16**	**0.40**	**0.68**
EnsembleRetriever	0.13	0.39	0.41	0.24	0.15	0.11	0.35	0.64

Tab. 6.10.: LLM-as-Judge Generation Metrics on the Real-World QA Set (Chunk Level, top k = 10)

Retriever	Factual Accuracy	Relevance	Fluency	Coherence	Helpfulness
BM25Retriever	3.73	4.20	4.93	4.21	3.88
VectorStoreRetriever	**4.66**	**4.86**	**4.98**	**4.87**	**4.75**
EnsembleRetriever	4.14	4.52	4.95	4.54	4.29

Analysis of Chunk-Level Generation Results

For the Synthetic QA set at the chunk level, the VectorStoreRetriever achieved the best results across traditional metrics, highlighting its superior ability in semantic retrieval. However, for LLM-as-Judge metrics, the EnsembleRetriever notably outperformed both VectorStoreRetriever and BM25 in factual accuracy, relevance, fluency, coherence, and helpfulness, demonstrating its strength in combining semantic and lexical retrieval methods to enhance overall retrieval results, and by extension, answer generation quality.

For the Real-world QA set at the chunk level, the VectorStoreRetriever consistently achieved top performance across all traditional metrics and LLM-as-Judge metrics, significantly outperforming both BM25 and EnsembleRetriever. The BM25 retriever demonstrated substantial limitations in generating accurate and relevant content, particularly reflected in very low BLEU and METEOR scores and lower factual accuracy and helpfulness scores.

6.2.3 Generation Evaluation Results per Granularity

Sentence Granularity

Synthetic QA Set

Tab. 6.11.: Traditional Generation Metrics on the Synthetic QA Set (Sentence Level, top-k = 100)

Retriever	BLEU	METEOR	R-1	R-2	R-3	R-4	R-L	BERTScore
BM25Retriever	0.18	0.43	0.48	0.32	0.25	0.20	0.42	0.69
VectorStoreRetriever	**0.29**	**0.54**	**0.57**	**0.42**	**0.32**	**0.25**	**0.49**	**0.74**
EnsembleRetriever	0.20	0.46	0.50	0.35	0.26	0.20	0.44	0.70

Tab. 6.12.: LLM-as-Judge Generation Metrics on the Synthetic QA Set (Sentence Level, top k = 100)

Retriever	Factual Accuracy	Relevance	Fluency	Coherence	Helpfulness
BM25Retriever	3.26	3.73	4.86	3.82	3.37
VectorStoreRetriever	**4.30**	**4.55**	**4.96**	**4.49**	**4.32**
EnsembleRetriever	3.75	4.21	4.89	4.07	3.88

Real-World QA Set

Tab. 6.13.: Traditional Generation Metrics on the Real-World QA Set (Sentence Level, top k = 100)

Retriever	BLEU	METEOR	R-1	R-2	R-3	R-4	R-L	BERTScore
BM25Retriever	0.11	0.34	0.39	0.23	0.15	0.12	0.33	0.63
VectorStoreRetriever	**0.17**	**0.41**	**0.45**	**0.28**	**0.20**	**0.15**	**0.38**	**0.67**
EnsembleRetriever	0.13	0.36	0.40	0.24	0.16	0.12	0.34	0.64

Real-World QA Set

Tab. 6.14.: LLM-as-Judge Generation Metrics on the Real-World QA Set (Sentence Level, top k = 100)

Retriever	Factual Accuracy	Relevance	Fluency	Coherence	Helpfulness
BM25Retriever	3.33	3.83	4.78	3.91	3.46
VectorStoreRetriever	**4.44**	**4.74**	**4.98**	**4.70**	**4.54**
EnsembleRetriever	3.98	4.37	4.94	4.45	4.16

Analysis of Sentence-Level Generation Results

For the Synthetic QA set at the sentence level, VectorStoreRetriever once again led across traditional metrics, outperforming both BM25 and EnsembleRetriever significantly. This trend is further reflected in LLM-as-Judge metrics, where VectorStoreRetriever scored highest in factual accuracy, relevance, coherence, and helpfulness. BM25 notably underperformed across both sets of metrics, indicating poor retrieval quality and lower semantic coherence.

For the Real-world QA set at the sentence level, VectorStoreRetriever maintained its leading position, clearly outperforming BM25 and EnsembleRetriever in all metrics. BM25 showed persistently low performance, highlighting its ineffectiveness in capturing relevant semantic context, while EnsembleRetriever presented moderate performance improvements but remained behind VectorStoreRetriever.

Proposition Granularity

Synthetic QA Set

Tab. 6.15.: Traditional Generation Metrics on the Synthetic QA Set (Proposition Level, top k = 100)

Retriever	BLEU	METEOR	R-1	R-2	R-3	R-4	R-L	BERTScore
BM25Retriever	0.19	0.46	0.50	0.34	0.26	0.20	0.42	0.70
VectorStoreRetriever	**0.30**	**0.57**	**0.58**	**0.43**	**0.33**	**0.26**	**0.50**	**0.75**
EnsembleRetriever	0.20	0.48	0.52	0.36	0.27	0.21	0.44	0.71

Tab. 6.16.: LLM-as-Judge Generation Metrics on the Synthetic QA Set (Proposition Level, top-k = 100)

Retriever	Factual Accuracy	Relevance	Fluency	Coherence	Helpfulness
BM25Retriever	3.81	4.15	4.90	4.21	3.90
VectorStoreRetriever	**4.35**	**4.59**	**4.95**	**4.51**	**4.40**
EnsembleRetriever	4.01	4.33	4.91	4.35	4.09

Real-World QA Set

Tab. 6.17.: Traditional Generation Metrics on the Real-World QA Set (Proposition Level, top k = 100)

Retriever	BLEU	METEOR	R-1	R-2	R-3	R-4	R-L	BERTScore
BM25Retriever	0.09	0.31	0.35	0.19	0.12	0.09	0.29	0.62
VectorStoreRetriever	**0.13**	**0.38**	**0.41**	**0.25**	**0.18**	**0.14**	**0.34**	**0.66**
EnsembleRetriever	0.09	0.33	0.36	0.20	0.13	0.10	0.30	0.63

Tab. 6.18.: LLM-as-Judge Generation Metrics on the Real-World QA Set (Proposition Level, top k = 100)

Retriever	Factual Accuracy	Relevance	Fluency	Coherence	Helpfulness
BM25Retriever	3.00	3.53	4.83	3.66	3.17
VectorStoreRetriever	**4.40**	**4.89**	**4.99**	**4.72**	4.55
EnsembleRetriever	4.34	4.75	4.89	4.71	**4.57**

Analysis of Proposition-Level Generation Results

For the Synthetic QA set at the proposition level, VectorStoreRetriever delivered the highest scores across all reference-based metrics, demonstrating effective semantic alignment and detailed context retrieval. In LLM-as-Judge metrics, VectorStoreRetriever again ranked highest, especially in factual accuracy and helpfulness, confirming its superior semantic understanding capabilities. EnsembleRetriever performed moderately, whereas BM25 consistently underperformed, reflecting significant shortcomings in capturing detailed semantic information.

For the Real-world QA set at the proposition level, VectorStoreRetriever again clearly dominated both reference-based and LLM-based generation metrics. The EnsembleRetriever showed competitive performance in LLM-as-Judge metrics, closely following VectorStoreRetriever in factual accuracy, relevance, and helpfulness, yet still behind in traditional metrics. BM25 remained notably inadequate, emphasizing its significant limitations in precise semantic retrieval.

Perplexity Tables

Synthetic QA Set

Tab. 6.19.: Perplexity Values for Each Granularity on the Synthetic QA Set

Granularity	Perplexity
Chunks	**6.44**
Sentences	6.63
Propositions	6.63

Real-World QA Set

Tab. 6.20.: Perplexity Values for Each Granularity on the Real-World QA Set

Granularity	Perplexity
Chunks	**8.52**
Sentences	8.68
Propositions	8.72

Analysis of Perplexity Results

Perplexity scores across both Synthetic and Real-world QA sets highlighted the advantage of chunk-level granularity, achieving lower perplexity values compared to sentence and proposition granularities. The consistently lower perplexity at chunk granularity underscores its optimal balance between context depth and semantic clarity, enhancing the generative performance across retrieval configurations.

6.2.4 Summary of Observations

Across all evaluated configurations, the VectorStoreRetriever consistently yielded the strongest performance in generation quality, as measured by both traditional reference-based metrics and LLM-as-Judge scores. This dominance was observed at all levels of retrieval granularity and across both the synthetic and real-world QA datasets, reinforcing the importance of semantic retrieval in effectively grounding answers and facilitating meaningful generation.

For chunk-level retrieval, which benefits from richer context windows, the VectorStoreRetriever led in traditional metrics such as BLEU, ROUGE, and METEOR, especially on the real-world QA set where semantic understanding is crucial. However, in the synthetic QA set, the EnsembleRetriever marginally outperformed in LLM-as-Judge metrics, particularly in factual accuracy and helpfulness. This suggests that combining lexical and dense retrieval signals may sometimes enhance generation when queries align more closely with indexed content. Nevertheless, this advantage did not generalize to real-world inputs, where the VectorStoreRetriever outperformed all others by a substantial margin across all evaluation dimensions.

At sentence-level granularity, the VectorStoreRetriever maintained its lead in both metric categories. Its ability to assemble coherent answers from context-fragmented sentences underlines its robustness, especially in handling diverse real-world phrasing. BM25Retriever consistently underperformed, and while the EnsembleRetriever closed the gap slightly in synthetic tasks, it could not surpass the semantic retriever in either dataset.

The proposition-level results further confirmed this pattern. On both QA sets, the VectorStoreRetriever scored highest across the board, with LLM-as-Judge evaluations particularly favoring it for factual accuracy and relevance. The fine granularity of propositions likely demands accurate semantic alignment, a task that lexical-based retrieval inherently struggles with. Moreover, BM25Retriever's default hyperparameters—which were not tuned in this work—are typically optimized for longer documents and may therefore perform suboptimally on short segments such as sentences and propositions. While

the EnsembleRetriever did improve in the real-world QA set under LLM-based evaluation, its traditional metric scores remained inferior to those of the semantic retriever, suggesting that score fusion at the proposition level may still fall short of exploiting contextual density optimally.

Perplexity results mirrored the broader findings: chunk-level retrieval consistently yielded the lowest values on both datasets, indicating higher model confidence and better contextual predictability. This affirms that the chunk representations strike an effective balance between information richness and answer relevance, enhancing the model's ability to generate accurate and fluent answers. In contrast, sentences and propositions, while more fine-grained, did not improve perplexity.

Overall, these observations point to VectorStoreRetriever with chunk-level retrieval as the most reliable configuration for the task of this thesis, offering optimal performance in both controlled (synthetic) and unconstrained (real-world) settings. Sentence- and proposition-level retrieval offer valuable avenues for further refinement, particularly when combined with semantic reranking or dynamic context selection, but may require advanced post-retrieval strategies to rival chunk-level results. Importantly, the results also underscore the limited scalability of BM25Retriever in text generation tasks and the inconsistent gains of hybrid retrieval in real-world QA, thereby motivating more adaptive fusion mechanisms or reranking pipelines for future development.

7 Conclusions

7.1 Conclusions

Overall, VectorStoreRetriever consistently emerged as the superior retriever across all granularities and datasets in both reference-based and LLM-based generation metrics, for the particular task of this thesis. Chunk granularity, in particular, also provided the optimal generative performance, evident through the lowest perplexity values and consistently higher generation metric scores. BM25 showed severe limitations across most conditions, as further discussed in Section 7.2, while the EnsembleRetriever provided moderate improvements, especially notable in LLM-based metrics at chunk granularity. These results suggest that currently, VectorStoreRetriever combined with the chunk granularity represents the most effective configuration for the proposed RAG system.

Additionally, the experiments emphasized the utility of synthetic question-answer generation as a practical method for preliminary system evaluation, particularly valuable in low-resource environments lacking extensive real-world datasets. However, observed discrepancies between synthetic and authentic question-answer datasets indicate that synthetic datasets alone might not fully capture the complexity and subtlety inherent in real user queries. Therefore, a balanced combination of synthetic and authentic data is recommended to ensure evaluation results more closely reflect actual performance scenarios.

Furthermore, the cost-performance trade-off analysis conducted in this thesis highlighted how, despite constraints in computational resources—particularly hardware limitations and model size—the developed RAG system achieved commendable performance. This demonstrates that an efficient and thoughtful system design, incorporating optimal retrieval architecture and careful granularity selection, can effectively offset resource constraints, reinforcing the potential for robust yet resource-efficient RAG system deployments.

7.2 Limitations

It must be noted that in the current implementation of the system, the document collection size remains small and lacks diversity, potentially affecting generalizability. Furthermore,

the quality of decontextualization during proposition creation, which primarily depends on the model and the prompt used, may also impact retrieval and generation accuracy. The significant limitations of BM25, especially in the sentence and proposition granularities, can be attributed not only to its inability to capture semantic similarity but also to the use of untuned hyperparameter values. Since these default values are generally optimized for longer documents, their direct application to much shorter units of text likely reduced retrieval effectiveness, as also evidenced in prior work on BM25 tuning [Cha+21]. The advantages observed at the chunk granularity likely stem from the highly structured nature and inherent contextual completeness of the Studies Guide document. However, most real-world documents rarely exhibit such characteristics, suggesting the potential value of proposition-level decomposition to handle less structured, more diverse corpora effectively. Expanding the knowledge base to include documents of varying structural complexity could further validate the effectiveness and generalizability of proposition-level retrieval.

7.3 Future Work

Future work will focus on several critical improvements and experiments to enhance system performance and robustness. Expanding the corpus by indexing additional university-related documents, including regulatory documents, professor CVs, and Studies Guides from other departments, will address limitations regarding corpus size and diversity. Implementing pre-retrieval strategies such as metadata filtering and post-retrieval reranking could further improve retrieval accuracy.

Exploring more advanced embedding models and stronger generator LLMs represents a promising direction for improving semantic searching and answer generation quality. Future work could also include tuning the BM25 hyperparameters to better accommodate shorter retrieval units. Moreover, experimenting with multiple granularities simultaneously or dynamically selecting granularity during query processing could optimize retrieval performance, as suggested by Zhong et al. [Zho+25]

Finally, expanding and enhancing the real-world QA dataset will provide a more robust basis for evaluating and improving the system, ensuring it remains responsive and effective in addressing realistic academic queries and information needs. Additionally, initiating trial deployments of the system and engaging students to interact with it, providing feedback on missing features and their overall satisfaction, would further inform iterative enhancements and ensure alignment with user expectations and practical usability.

Bibliography

[AD19] Hiteshwar Kumar Azad and Akshay Deepak. "Query expansion techniques for information retrieval: A survey". In: *Information Processing and Management* 56.5 (Sept. 2019), pp. 1698–1735.

[Ain+23] Joshua Ainslie, James Lee-Thorp, Michiel de Jong, et al. "GQA: Training Generalized Multi-Query Transformer Models from Multi-Head Checkpoints". In: *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. Ed. by Houda Bouamor, Juan Pino, and Kalika Bali. Singapore: Association for Computational Linguistics, Dec. 2023, pp. 4895–4901.

[Ant+24] Chiara Antico, Stefano Giordano, Cansu Koyuturk, and Dimitri Ognibene. *Unimib Assistant: designing a student-friendly RAG-based chatbot for all their needs*. 2024. arXiv: 2411.19554 [cs.HC].

[CC14] Boxing Chen and Colin Cherry. "A Systematic Comparison of Smoothing Techniques for Sentence-Level BLEU". In: *Proceedings of the Ninth Workshop on Statistical Machine Translation*. Ed. by Ondřej Bojar, Christian Buck, Christian Federmann, et al. Baltimore, Maryland, USA: Association for Computational Linguistics, June 2014, pp. 362–367.

[CCB09] Gordon V. Cormack, Charles L. A. Clarke, and Stefan Büttcher. "Reciprocal rank fusion outperforms condorcet and individual rank learning methods". In: *Proceedings of the 32nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2009, Boston, MA, USA, July 19-23, 2009*. Ed. by James Allan, Javed A. Aslam, Mark Sanderson, ChengXiang Zhai, and Justin Zobel. ACM, 2009, pp. 758–759.

[Cha+21] Ilias Chalkidis, Manos Fergadiotis, Nikolaos Manginas, Eva Katakalou, and Prodromos Malakasiotis. "Regulatory Compliance through Doc2Doc Information Retrieval: A case study in EU/UK legislation where text similarity has limitations". In: *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*. Ed. by Paola Merlo, Jorg Tiedemann, and Reut Tsarfaty. Online: Association for Computational Linguistics, Apr. 2021, pp. 3498–3511.

[Che+24] Tong Chen, Hongwei Wang, Sihao Chen, et al. "Dense X Retrieval: What Retrieval Granularity Should We Use?" In: *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*. Ed. by Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen. Miami, Florida, USA: Association for Computational Linguistics, Nov. 2024, pp. 15159–15177.

[CR12] Claudio Carpineto and Giovanni Romano. "A Survey of Automatic Query Expansion in Information Retrieval". In: *ACM Comput. Surv.* 44.1 (Jan. 2012).

[CS24] Nathan Cooper and Torsten Scholak. *Perplexed: Understanding When Large Language Models are Confused*. 2024. arXiv: 2404.06634 [cs.SE].

[Dai+22] Zhuyun Dai, Arun Tejasvi Chaganty, Vincent Zhao, et al. "Dialog Inpainting: Turning Documents to Dialogs". In: *International Conference on Machine Learning (ICML)*. PMLR. 2022.

[Dai+25] Lu Dai, Yijie Xu, Jinhui Ye, Hao Liu, and Hui Xiong. *SePer: Measure Retrieval Utility Through The Lens Of Semantic Perplexity Reduction*. 2025. arXiv: 2503.01478 [cs.CL].

[Dev+19] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Ed. by Jill Burstein, Christy Doran, and Thamar Solorio. Minneapolis, Minnesota: Association for Computational Linguistics, June 2019, pp. 4171–4186.

[FPC21] Thibault Formal, Benjamin Piwowarski, and Stéphane Clinchant. "SPLADE: Sparse Lexical and Expansion Model for First Stage Ranking". In: *SIGIR '21: The 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. Virtual Event, Canada: ACM, July 2021, pp. 2288–2292.

[FS93] Edward A. Fox and Joseph A. Shaw. "Combination of Multiple Searches". In: *TREC*. Vol. 500-215. NIST Special Publication. National Institute of Standards and Technology (NIST), 1993, pp. 243–252.

[Gan+25] Aoran Gan, Hao Yu, Kai Zhang, et al. *Retrieval Augmented Generation Evaluation in the Era of Large Language Models: A Comprehensive Survey*. 2025. arXiv: 2504.14891 [cs.CL].

[Gao+24a] Yunfan Gao, Yun Xiong, Xinyu Gao, et al. *Retrieval-Augmented Generation for Large Language Models: A Survey*. 2024. arXiv: 2312.10997 [cs.CL].

[Gao+24b] Yunfan Gao, Yun Xiong, Meng Wang, and Haofen Wang. *Modular RAG: Transforming RAG Systems into LEGO-like Reconfigurable Frameworks*. 2024. arXiv: 2407.21059 [cs.CL].

[GRS24] Shailja Gupta, Rajesh Ranjan, and Surya Narayan Singh. *A Comprehensive Survey of Retrieval-Augmented Generation (RAG): Evolution, Current Landscape and Future Directions*. 2024. arXiv: 2410.12837 [cs.CL].

[Has24] Jahid Hasan. *Optimizing Large Language Models through Quantization: A Comparative Analysis of PTQ and QAT Techniques*. 2024. arXiv: 2411.06084 [cs.LG].

[HB21] Michael Hanna and Ondřej Bojar. "A Fine-Grained Analysis of BERTScore". In: *Proceedings of the Sixth Conference on Machine Translation*. Ed. by Loic Barrault, Ondrej Bojar, Fethi Bougares, et al. Online: Association for Computational Linguistics, Nov. 2021, pp. 507–517.

[HH24] Yizheng Huang and Jimmy Huang. *A Survey on Retrieval-Augmented Text Generation for Large Language Models*. 2024. arXiv: 2404.10981 [cs.IR].

[Hu+24] Yutong Hu, Quzhe Huang, Mingxu Tao, Chen Zhang, and Yansong Feng. *Can Perplexity Reflect Large Language Model's Ability in Long Text Understanding?* 2024. arXiv: 2405.06105 [cs.CL].

[Hwa+23] Yerin Hwang, Yongil Kim, Hyunkyung Bae, et al. "Dialogizer: Context-aware Conversational-QA Dataset Generation from Textual Sources". In: *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. Ed. by Houda Bouamor, Juan Pino, and Kalika Bali. Singapore: Association for Computational Linguistics, Dec. 2023, pp. 8806–8828.

[Jag+23] Rolf Jagerman, Honglei Zhuang, Zhen Qin, Xuanhui Wang, and Michael Bendersky. *Query Expansion by Prompting Large Language Models*. 2023. arXiv: 2305.03653 [cs.IR].

[JDJ21] Jeff Johnson, Matthijs Douze, and Hervé Jégou. "Billion-Scale Similarity Search with GPUs". In: *IEEE Transactions on Big Data* 7.3 (July 2021), pp. 535–547.

[Ke22] Weimao Ke. "Alternatives to Classic BM25-IDF based on a New Information Theoretical Framework". In: *2022 IEEE International Conference on Big Data (Big Data)*. Dec. 2022, pp. 36–44.

[Kur+25a] Gustavo Kuratomi, Paulo Pirozelli, Fabio G. Cozman, and Sarajane M. Peres. *A RAG-Based Institutional Assistant*. 2025. arXiv: 2501.13880 [cs.CL].

[Kur+25b] Eldar Kurtic, Alexandre Marques, Shubhra Pandit, Mark Kurtz, and Dan Alistarh. *"Give Me BF16 or Give Me Death"? Accuracy-Performance Trade-Offs in LLM Quantization*. 2025. arXiv: 2411.02355 [cs.LG].

[Kuz+20] Saar Kuzi, Mingyang Zhang, Cheng Li, Michael Bendersky, and Marc Najork. *Leveraging Semantic and Lexical Matching to Improve the Recall of Document Retrieval Systems: A Hybrid Approach*. 2020. arXiv: 2010.01195 [cs.IR].

[KZ20] Omar Khattab and Matei Zaharia. "ColBERT: Efficient and Effective Passage Search via Contextualized Late Interaction over BERT". In: *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2020, Virtual Event, China, July 25-30, 2020*. Ed. by Jimmy X. Huang, Yi Chang, Xueqi Cheng, et al. ACM, 2020, pp. 39–48.

[LA07] Alon Lavie and Abhaya Agarwal. "Meteor: an automatic metric for MT evaluation with high levels of correlation with human judgments". In: *Proceedings of the Second Workshop on Statistical Machine Translation*. StatMT '07. Prague, Czech Republic: Association for Computational Linguistics, 2007, pp. 228–231.

[Lew+20] Patrick Lewis, Ethan Perez, Aleksandra Piktus, et al. "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks". In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin. Vol. 33. Curran Associates, Inc., 2020, pp. 9459–9474.

[Li+24] Haitao Li, Qian Dong, Junjie Chen, et al. *LLMs-as-Judges: A Comprehensive Survey on LLM-based Evaluation Methods*. 2024. arXiv: 2412.05579 [cs.CL].

[Lin04] Chin-Yew Lin. "ROUGE: A Package for Automatic Evaluation of Summaries". In: *Text Summarization Branches Out*. Barcelona, Spain: Association for Computational Linguistics, July 2004, pp. 74–81.

[Liu+16] Chia-Wei Liu, Ryan Lowe, Iulian Serban, et al. "How NOT To Evaluate Your Dialogue System: An Empirical Study of Unsupervised Evaluation Metrics for Dialogue Response Generation". In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Ed. by Jian Su, Kevin Duh, and Xavier Carreras. Austin, Texas: Association for Computational Linguistics, Nov. 2016, pp. 2122–2132.

[Liu+23] Nelson F. Liu, Kevin Lin, John Hewitt, et al. *Lost in the Middle: How Language Models Use Long Contexts*. 2023. arXiv: 2307.03172 [cs.CL].

[Lla24] AI@Meta Llama Team. *The Llama 3 Herd of Models*. https://llama.meta.com/. arXiv:2407.21783. 2024.

[Low+17] Ryan Lowe, Michael Noseworthy, Iulian Vlad Serban, et al. "Towards an Automatic Turing Test: Learning to Evaluate Dialogue Responses". In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Ed. by Regina Barzilay and Min-Yen Kan. Vancouver, Canada: Association for Computational Linguistics, July 2017, pp. 1116–1126.

[Ma+23] Xinbei Ma, Yeyun Gong, Pengcheng He, Hai Zhao, and Nan Duan. "Query Rewriting in Retrieval-Augmented Large Language Models". In: *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. Ed. by Houda Bouamor, Juan Pino, and Kalika Bali. Singapore: Association for Computational Linguistics, Dec. 2023, pp. 5303–5315.

[MA02] Mark H. Montague and Javed A. Aslam. "Condorcet fusion for improved retrieval." In: *CIKM*. ACM, 2002, pp. 538–548.

[Mek+25] Anmol Mekala, Anirudh Atmakuru, Yixiao Song, Marzena Karpinska, and Mohit Iyyer. *Does quantization affect models' performance on long-context tasks?* 2025. arXiv: 2505.20276 [cs.CL].

[MGG25] Chandana Sree Mala, Gizem Gezici, and Fosca Giannotti. *Hybrid Retrieval for Hallucination Mitigation in Large Language Models: A Comparative Analysis*. 2025. arXiv: 2504.05324 [cs.IR].

[MMZ23] Anqi Mao, Mehryar Mohri, and Yutao Zhong. *Cross-Entropy Loss Functions: Theoretical Analysis and Applications*. 2023. arXiv: 2304.07288 [cs.LG].

[Nas+21] Shahrzad Naseri, Jeffrey Dalton, Andrew Yates, and James Allan. "CEQE: Contextualized Embeddings for Query Expansion". In: *Advances in Information Retrieval: 43rd European Conference on IR Research, ECIR 2021, Virtual Event, March 28 – April 1, 2021, Proceedings, Part I*. Berlin, Heidelberg: Springer-Verlag, 2021, pp. 467–482.

[NC19] Rodrigo Nogueira and Kyunghyun Cho. *Passage Re-ranking with BERT*. arXiv: 1901.04085. Feb. 2019.

[Neu+24] Subash Neupane, Elias Hossain, Jason Keith, et al. *From Questions to Insightful Answers: Building an Informed Chatbot for University Resources*. 2024. arXiv: 2405.08120 [cs.ET].

[NK18] Preksha Nema and Mitesh M. Khapra. "Towards a Better Metric for Evaluating Question Generation Systems". In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Ed. by Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii. Brussels, Belgium: Association for Computational Linguistics, 2018, pp. 3950–3959.

[Nog+20] Rodrigo Nogueira, Zhiying Jiang, Ronak Pradeep, and Jimmy Lin. "Document Ranking with a Pretrained Sequence-to-Sequence Model". In: *Findings of the Association for Computational Linguistics: EMNLP 2020*. Ed. by Trevor Cohn, Yulan He, and Yang Liu. Online: Association for Computational Linguistics, Nov. 2020, pp. 708–718.

[Ouy+22] Long Ouyang, Jeffrey Wu, Xu Jiang, et al. "Training language models to follow instructions with human feedback". In: *Advances in Neural Information Processing Systems*. Ed. by S. Koyejo, S. Mohamed, A. Agarwal, et al. Vol. 35. Curran Associates, Inc., 2022, pp. 27730–27744.

[Pap+02] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. "BLEU: a method for automatic evaluation of machine translation". In: *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*. ACL '02. Philadelphia, Pennsylvania: Association for Computational Linguistics, 2002, pp. 311–318.

[PBF24] Arjun Panickssery, Samuel R. Bowman, and Shi Feng. *LLM Evaluators Recognize and Favor Their Own Generations*. 2024. arXiv: 2404.13076 [cs.CL].

[PMM24] Aleksandr V. Petrov, Sean MacAvaney, and Craig Macdonald. "Shallow Cross-Encoders for Low-Latency Retrieval". In: *Advances in Information Retrieval: 46th European Conference on Information Retrieval, ECIR 2024, Glasgow, UK, March 24–28, 2024, Proceedings, Part III*. Glasgow, United Kingdom: Springer-Verlag, 2024, pp. 151–166.

[Raf+23] Rafael Rafailov, Archit Sharma, Eric Mitchell, et al. "Direct preference optimization: your language model is secretly a reward model". In: *Proceedings of the 37th International Conference on Neural Information Processing Systems*. NIPS '23. New Orleans, LA, USA: Curran Associates Inc., 2023.

[RG19] 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 and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Ed. by Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 3982–3992.

[RZ09] Stephen Robertson and Hugo Zaragoza. "The Probabilistic Relevance Framework: BM25 and Beyond". In: *Found. Trends Inf. Retr.* 3.4 (Apr. 2009), pp. 333–389.

[Sam+25] Saron Samuel, Dan DeGenaro, Jimena Guallar-Blasco, et al. *MMMORRF: Multimodal Multilingual Modularized Reciprocal Rank Fusion*. 2025. arXiv: 2503.20698 [cs.CV].

[SG25] Jakub Swacha and Michał Gracel. "Retrieval-Augmented Generation (RAG) Chatbots for Education: A Survey of Applications". In: *Applied Sciences* 15.8 (2025).

[Sha+17] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, et al. *Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer*. 2017. arXiv: 1701.06538 [cs.LG].

[Sha+24] Sanat Sharma, David Seunghyun Yoon, Franck Dernoncourt, et al. *Retrieval Augmented Generation for Domain-specific Question Answering*. 2024. arXiv: 2404.14760 [cs.CL].

[Sha25] Chaitanya Sharma. *Retrieval-Augmented Generation: A Comprehensive Survey of Architectures, Enhancements, and Robustness Frontiers*. 2025. arXiv: 2506.00054 [cs.IR].

[Sou+24] Heydar Soudani, Roxana Petcu, Evangelos Kanoulas, and Faegheh Hasibi. *A Survey on Recent Advances in Conversational Data Generation*. 2024. arXiv: 2405.13003 [cs.CL].

[Su+21] Jianlin Su, Yu Lu, Shengfeng Pan, Bo Wen, and Yunfeng Liu. "RoFormer: Enhanced Transformer with Rotary Position Embedding". In: *CoRR* abs/2104.09864 (2021). arXiv: 2104.09864.

[Vas+17] Ashish Vaswani, Noam Shazeer, Niki Parmar, et al. “Attention is All you Need”. In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon, U. Von Luxburg, S. Bengio, et al. Vol. 30. Curran Associates, Inc., 2017.

[Ver24] Sourav Verma. *Contextual Compression in Retrieval-Augmented Generation for Large Language Models: A Survey*. 2024. arXiv: 2409.13385 [cs.CL].

[Vla+25] Christos Vlachos, Nikolaos Stylianou, Alexandra Fiotaki, et al. *Building Open-Retrieval Conversational Question Answering Systems by Generating Synthetic Data and Decontextualizing User Questions*. 2025. arXiv: 2507.04884 [cs.CL].

[Wan+13] Yining Wang, Liwei Wang, Yuanzhi Li, Di He, and Tie-Yan Liu. “A Theoretical Analysis of NDCG Type Ranking Measures”. In: *Proceedings of the 26th Annual Conference on Learning Theory*. Ed. by Shai Shalev-Shwartz and Ingo Steinwart. Vol. 30. Proceedings of Machine Learning Research. Princeton, NJ, USA: PMLR, June 2013, pp. 25–54.

[Wan+20] Wenhui Wang, Furu Wei, Li Dong, et al. “MiniLM: Deep Self-Attention Distillation for Task-Agnostic Compression of Pre-Trained Transformers”. In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin. Vol. 33. Curran Associates, Inc., 2020, pp. 5776–5788.

[Wu+22] Zeqiu Wu, Yi Luan, Hannah Rashkin, et al. “CONQRR: Conversational Query Rewriting for Retrieval with Reinforcement Learning”. In: *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*. Ed. by Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang. Abu Dhabi, United Arab Emirates: Association for Computational Linguistics, Dec. 2022, pp. 10000–10014.

[WZZ21] Shuai Wang, Shengyao Zhuang, and Guido Zuccon. “BERT-based Dense Retrievers Require Interpolation with BM25 for Effective Passage Retrieval”. In: *Proceedings of the 2021 ACM SIGIR International Conference on Theory of Information Retrieval*. ICTIR ’21. Virtual Event, Canada: Association for Computing Machinery, 2021, pp. 317–324.

[XZ25] Tong Xiao and Jingbo Zhu. *Foundations of Large Language Models*. 2025. arXiv: 2501.09223 [cs.CL].

[Ye+24] Jiayi Ye, Yanbo Wang, Yue Huang, et al. *Justice or Prejudice? Quantifying Biases in LLM-as-a-Judge*. 2024. arXiv: 2410.02736 [cs.CL].

[Yor+24] Ori Yoran, Tomer Wolfson, Ori Ram, and Jonathan Berant. “MAKING RETRIEVAL-AUGMENTED LANGUAGE MODELS ROBUST TO IRRELEVANT CONTEXT”. In: Publisher Copyright: © 2024 12th International Conference on Learning Representations, ICLR 2024. All rights reserved.; 12th International Conference on Learning Representations, ICLR 2024 ; Conference date: 07-05-2024 Through 11-05-2024. 2024.

[Yu+25] Hao Yu, Aoran Gan, Kai Zhang, et al. “Evaluation of Retrieval-Augmented Generation: A Survey”. In: *Big Data*. Springer Nature Singapore, 2025, pp. 102–120.

[Zha+20] Tianyi Zhang*, Varsha Kishore*, Felix Wu*, Kilian Q. Weinberger, and Yoav Artzi. "BERTScore: Evaluating Text Generation with BERT". In: *International Conference on Learning Representations*. 2020.

[Zhe+23] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, et al. *Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena*. 2023. arXiv: 2306.05685 [cs.CL].

[Zho+25] Zijie Zhong, Hanwen Liu, Xiaoya Cui, Xiaofan Zhang, and Zengchang Qin. *Mix-of-Granularity: Optimize the Chunking Granularity for Retrieval-Augmented Generation*. 2025. arXiv: 2406.00456 [cs.LG].

[Zie+20] Daniel M. Ziegler, Nisan Stiennon, Jeffrey Wu, et al. *Fine-Tuning Language Models from Human Preferences*. 2020. arXiv: 1909.08593 [cs.CL].

Prompt Templates

A

This appendix presents the prompt templates used throughout the development of the AUEBot assistant. These prompts were submitted to LLMs to enable document transformation and synthetic data generation, and are central to the system's preprocessing and evaluation pipeline.

A.1 Decontextualized Proposition Creation

Chunk Decomposition to Decontextualized Propositions Prompt

I am giving you a text chunk extracted from the Studies Guide of the Informatics Department of Athens University of Economics and Business.

Your task is to decompose the chunk into clear, simple, and decontextualized propositions.

Follow these instructions:
1. Split compound sentences into simple sentences. Maintain the original phrasing from the input whenever possible.
2. For any named entity that is accompanied by additional descriptive information, separate this information into its own distinct proposition.
3. Always decontextualize each proposition by adding necessary modifiers to nouns or entire sentences and replacing pronouns (e.g., "it", "he", "she", "they", "this", "that") with the full name of the entities they refer to. This is imperative!
4. Every proposition MUST BE DECONTEXTUALIZED FULLY, SO THAT IT CAN BE INTERPRETABLE ON ITS OWN.
5. Ensure the JSON output is encoded in UTF-8 without Unicode escape sequences.

```
{
  "{num}": {
    "chunk_id": <chunk_id>,
    "propositions": [
      {
        "prop_id": 1,
```

```
        "text": "<Proposition 1>"
      },
      {
        "prop_id": 2,
        "text": "<Proposition 2>"
      }
      ...
    ]
  }
}

<chunk>
{chunk}
</chunk>
```

A.2 QA Pairs Generation

Synthetic QA Pairs Generation from Chunks Prompt

You are given a set of text chunks extracted from the Studies Guide of the Informatics Department of Athens University of Economics and Business.

Your task is to read the given chunks and generate question-answer pairs between a user and a virtual assistant based on one or more chunks, where the user asks certain questions, and the assistant tries to provide answers.

Follow these instructions:
1. Your response should be a JSON structure of the following format:

```
{
  "{num}": {
    "user": "Your question here",
    "chunks_used_ids": [4, 5, ...],
    "system": "Your answer here"
  },
  "{next_num}": {
    "user": "Your question here",
    "chunks_used_ids": [1, 3, ...],
    "system": "Your answer here"
  }
  ...
}
```

2. You MUST generate at least one question-answer pair for EACH of the provided chunks.

3. The ‘chunks_used_ids‘ field must include the exact ‘chunk_id‘ of the chunk the QA pair is based on.
4. The system must answer helpfully, carefully, politely, impartially, honestly, and respectfully to the user.
5. The user’s question must be a self-contained, standalone question without the need to refer to any previous context.
6. You may combine two or more chunks to answer a more complex question, but make sure that each chunk is still used in at least one distinct question-answer pair.
7. In each question-answer pair, the system answers the user’s question based only on information from the retrieved chunks.
8. Answers must form a complete sentence or paragraph.
9. Create as many high-quality question-answer pairs as are reasonably supported by the given chunks, with a minimum of one per chunk.
10. You can use the same chunk in multiple question-answer pairs, but make sure every chunk is used in at least one.
11. You can generate more than one question-answer pair per chunk if meaningful, but one is the minimum for each.
12. Questions should be phrased as if asked by an undergraduate Informatics student at AUEB.
13. Generate at least one question-answer pair for each chunk given.
14. Each question and answer should be a single-line string.
15. For this subset, you MUST generate at least 15 question-answer pairs in total.

```
<chunks>
{docs}
</chunks>
```

A.3 QA Pairs Annotation Sentence Annotation

QA Pair Annotation from Chunk IDs to Sentence IDs Prompt

You are given a set of question-answer pairs generated from text chunks of the Studies Guide of the Informatics Department of Athens University of Economics and Business.

Each pair includes:
- A user question,
- The assistant’s answer,

- The chunk ids from which the answer was derived.

Your task is to annotate each QA pair by identifying the specific sentence IDs that best support or justify the assistant's answer.

Follow these instructions carefully:
1. You will be provided with:
- A set of question-answer pairs.
- A set of sentences, each associated with a unique ‘chunk_id‘ from the ‘chunks_used_ids‘ field in the QA pairs.
2. Your output must be a valid JSON object of the following form:

```
{
  "{num}": {
    "user": "Original question here",
    "sentences_used_ids": [5, 10, ...],
    "system": "Original answer here"
  }
  ...
}
```

3. For each QA pair, use the sentences that most directly support the assistant's answer.
4. Be precise — select only the relevant sentences, not all from a chunk.
5. The ‘sentences_used_ids‘ must belong to the ‘chunks_used_ids‘ used in the original answer.
6. Do NOT change the original 'user' question or 'system' answer. Only insert the 'sentences_used_ids' field for each QA.
7. Return only the updated JSON structure as your output — do not include any other text.

```
<question_answer_pairs>
{qa_pairs}
</question_answer_pairs>

<sentences>
{sentences}
</sentences>
```

A.4 QA Pairs Proposition Annotation

QA Pair Annotation from Chunk IDs to Proposition IDs Prompt

You are given a set of question-answer pairs generated from text chunks of the Studies Guide of the Informatics Department of Athens University of Economics and Business.

Each pair includes:
- A user question,
- The assistant's answer,
- The chunk ids from which the answer was derived.

Your task is to annotate each QA pair by identifying the specific proposition IDs that best support or justify the assistant's answer.

Follow these instructions carefully:
1. You will be provided with:
- A set of question-answer pairs.
- A set of propositions, each associated with a unique 'chunk_id' from the 'chunks_used_ids' field in the QA pairs.
2. Your output must be a valid JSON object of the following form:

```
{
  "{num}": {
    "user": "Original question here",
    "propositions_used_ids": [5, , 10 ...],
    "system": "Original answer here"
  }
  ...
}
```

3. For each QA pair, use the propositions that most directly support the assistant's answer.
4. Be precise — select only the relevant propositions, not all from a chunk.
5. The 'propositions_used_ids' must belong to the 'chunks_used_ids' used in the original answer.
6. Do NOT change the original 'user' question or 'system' answer. Only insert the 'propositions_used_ids' field for each QA.
7. Return only the updated JSON structure as your output — do not include any other text.

<question_answer_pairs>
{qa_pairs}

```
</question_answer_pairs>

<propositions>
{propositions}
</propositions>
```

A.5 System Prompt for AUEBbot

System Prompt

"You are AUEBbot, an assistant mainly for undergraduate students of the Department of Informatics at the Athens University of Economics and Business, also known as AUEB. Your task is to answer a student's question based only on valid information from the department's current Studies Guide.
You are particularly helpful to the student and provide concise but adequately comprehensive answers. Focus on answering as accurately as possible. Respond carefully, politely, impartially, honestly, and respectfully towards the student.
You will only accept to answer questions that are relevant to AUEB and the Department of Informatics.
If you cannot deduce a valid answer for the student, simply answer «I am afraid I do not know the answer to your question.»"

A.6 LLM-based Generation Evaluation

LLM-as-Judge Evaluation Prompt

Evaluate the following response based on the user's query and the provided context documents.

Rate the response on the following criteria:
- Relevance: Does it address the user's query? (1-5)
- Factual Accuracy: Is the information correct? (1-5)
- Fluency: Is the response well-written and grammatically correct? (1-5)
- Coherence: Does it logically follow from the query and context? (1-5)
- Helpfulness: Does it provide actionable or useful information for the user? (1-5)

Query:
{query}

Retrieved Context:
{context}

Generated Response:
{response}

Output: Provide scores for each metric and a brief comment in a single-line string justifying your decision.

- Return your answer **only** in the following JSON format.
- Do not include any extra commentary or text outside the JSON block.

Your answer should be in the following JSON format:

```
{
  "{query_id}": {
    "query_id": {query_id},
    "evaluation": {
      "Relevance": <score>,
      "FactualAccuracy": <score>,
      "Fluency": <score>,
      "Coherence": <score>,
      "Helpfulness": <score>,
      "Comments": "<optional additional comments>"
    }
  }
  ...
}
```

List of Acronyms

AUEB	Athens University of Economics and Business
API	Application Programming Interface
AI	Artificial Intelligence
NLP	Natural Language Processing
QA	Question-Answering
ConvQA	Conversational Question-Answering
IR	Information Retrieval
TREC	Text Retrieval Conference
LLM	Large Language Model
RoPe	Rotary Positional embeddings
SFT	Supervised Fine-Tuning
RS	Rejection Sampling
DPO	Direct Preference Optimization
RAG	Retrieval-Augmented Generation
TF	Term Frequency
IDF	Inverse Document Frequency

BM25 Best Match 25

BERT Bidirectional Encoder Representations from Transformers

SBERT Sentence BERT

FAISS Facebook AI Similarity Search

IP Inner Product

RRF Reciprocal Rank Fusion

AP Average Precision

MAP Mean Average Precision

NDCG Normalized Discounted Cumulative Gain

MRR Mean Reciprocal Rank

BLEU Bilingual Evaluation Understudy

ROUGE Recall-Oriented Understudy for Gisting Evaluation

METEOR Metric for Evaluation of Translation with Explicit Ordering

PPL Perplexity

JSON JavaScript Object Notation

List of Figures

List of Tables