Informatics

Automatische Evaluierung und Paramter Optimierung für Retrieval Augmented Generation Systeme

MASTERARBEIT

zur Erlangung des akademischen Grades

Master of Science

im Rahmen des Studiums

Computational Science and Engineering

eingereicht von

Simon König, Bsc.
Matrikelnummer 11702826

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Andreas Rauber
Mitwirkung: Pretitle Forename Surname, Posttitle
Pretitle Forename Surname, Posttitle
Pretitle Forename Surname, Posttitle

Wien, 17. Jänner 2025

Simon König Andreas Rauber

Informatics

Automated evaluation and parameter optimisation for retrieval augmented generation systems

MASTER'S THESIS

submitted in partial fulfillment of the requirements for the degree of

Master of Science

in

Computational Science and Engineering

by

Simon König, Bsc.
Registration Number 11702826

to the Faculty of Informatics

at the TU Wien

Advisor: Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Andreas Rauber
Assistance: Pretitle Forename Surname, Posttitle
Pretitle Forename Surname, Posttitle
Pretitle Forename Surname, Posttitle

Vienna, January 17, 2025

Simon König   Andreas Rauber

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.at

Erklärung zur Verfassung der Arbeit

Simon König, Bsc.

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.
Ich erkläre weiters, dass ich mich generativer KI-Tools lediglich als Hilfsmittel bedient habe und in der vorliegenden Arbeit mein gestalterischer Einfluss überwiegt. Im Anhang „Übersicht verwendeter Hilfsmittel“ habe ich alle generativen KI-Tools gelistet, die verwendet wurden, und angegeben, wo und wie sie verwendet wurden. Für Textpassagen, die ohne substantielle Änderungen übernommen wurden, haben ich jeweils die von mir formulierten Eingaben (Prompts) und die verwendete IT- Anwendung mit ihrem Produktnamen und Versionsnummer/Datum angegeben.

Wien, 17. Jänner 2025

Simon König

Abstract

English Abstract

Retrieval-Augmented Generation (RAG) systems connect language models to external knowledge bases. The rapid advancements in natural language processing (NLP) and machine learning, particularly transformer-based architectures, have led to the widespread development of RAG systems. While the field is trending in a research-oriented direction, the design of such system often remains more of an art than a science. This work aims to contribute to a more scientific approach for the design and evaluation of such systems. Recent research has introduced evaluation frameworks that heavily depend on labeled datasets to provide guidance for RAG system design. However, these frameworks often fall short in scenarios where labeled datasets are unavailable. This study addresses these limitations by proposing a lightweight RAG evaluation framework capable of handling unlabeled datasets. A RAG pipeline is developed and configured using the proposed evaluation system. The evaluation framework makes use of two complementary scoring metrics - the lightweight ROUGE metric and the elaborate LLM-judge metric. This study introduces a novel, ready-to-use RAG evaluation framework and offers general guidelines for improved RAG system design. Additionally, the RAG Triad approach is proposed as a method for effectively handling datasets without ground-truth labels. The findings of this work contribute to the rapidly evolving ecosystem of natural language AI by offering a robust framework for the evaluation, recommendation and innovation of RAG systems.

German Abstract

Retrieval-Augmented Generation (RAG)-Systeme verbinden Sprachmodelle mit externen Wissensdatenbanken. Die rapiden Fortschritte bei der Verarbeitung natürlicher Sprache (NLP) und maschinelles Lernen, insbesondere transformatorbasierte Architekturen, haben zur weit verbreiteten Entwicklung von RAG-Systemen geführt. Während das Feld in eine forschungsorientierte Richtung tendiert, bleibt der Entwurf solcher Systeme oft eher eine Kunst als eine Wissenschaft. Diese Arbeit soll zu einem wissenschaftlicheren Ansatz für den Entwurf und die Bewertung solcher Systeme beizutragen. In der jüngeren Forschung wurden Evaluierungs-Systeme eingeführt, die stark von gelabelten Datensätzen abhängen, um Anhaltspunkte für den Entwurf von RAG-Systemen zu liefern. Diese Systeme sind jedoch oft in Szenarien, in denen markierte Datensätze nicht verfügbar sind, nicht anwenbar. Diese Arbeit adressiert diese Problem und stellt ein leicht gewichtiges RAG-Evaluierungsframework vor, das auch mit unmarkierten Datensätzen umgehen kann. Zwei komplementäre Bewertungsmetriken finden Anwerndung in dieser Arbeit, die leichtgewichtige ROUGE-Metrik und die aufwendige LLM-Judge-Metrik. Diese Studie stellt ein neuartiges, gebrauchsfertiges RAG-Bewertungssystem vor und bietet allgemeine Richtlinien für ein verbessertes RAG Design. Zusätzlich wird der RAG-Triad-Ansatz als Methode zur effektiven Handhabung von Datensätzen ohne „ground-truth labels“ vorgeschlagen. Die Ergebnisse dieser Arbeit tragen zu dem, sich schnell entwickelnden Ökosystem der natürlichsprachlichen KI bei, indem sie einen robusten Rahmen für die Bewertung von RAG Systemen liefern.

Contents

CHAPTER 1

Introduction

1.1 Motivation

The landscape of Large Language Models (LLMs) and associated fields is undergoing rapid evolution, driven by ongoing advancements and innovations. As LLMs gain relevance in modern private and business use, the major issues of such models are the lack of domain-specific knowledge and hallucinations. One of the leading methods to improve on these problems is Retrieval Augmented Generation (RAG). This approach integrates retrieval mechanisms with generative natural language models to produce responses which are domain specific and take into account the most recent developments and changes within the domain.

This new mechanism opens up vast possibilities, but also proposes a lot of questions. How can a RAG system be comprehensively evaluated? It is more a methodology than an algorithm, and thus not trivial to evaluate. What are the objectives in method optimisation and which subcomponent *R, A or G* has the greatest impact? The purpose of this thesis is to investigate methods for evaluating the RAG pipeline itself and to provide guidelines on system design.

1.2 State of the Art

In the following, a brief summary of current advances in RAG developments is described, along with an outlook and impact on this thesis. Gao et al. recently provided a comprehensive overview of the latest developments in RAG ecosystems [GXG$^+$24]. Their survey encapsulates various methodologies, techniques and applications, shedding light on the state-of-the-art advancements and potential future directions. The paper outlines various RAG approaches, generally categorised into three distinct methods:

- **Naive RAG**

The naive RAG system goes through three main steps: indexing, retrieval, and generation. It starts by turning different document formats into plain text, breaking them into smaller parts, and converting these into searchable vectors. When a user asks a question, it finds the most similar resources, combines them with the user query and prompts a large language model to generate an answer based on the learned context. However, the main issues of naive RAG systems are imprecise retrieval, inaccurate utilisation of the retrieved resources and hallucinations of the LLM.

- **Advanced RAG**

  The advanced RAG system enriches the naive methodology by a pre-retrieval and post-retrieval step. During the pre-retrieval process, the main goal is to improve how the system organizes and understands the original question. This involves improving the search process by organising data more effectively and modifying the query to make it clearer. After finding relevant information, it is important to blend it smoothly with the original question. This might mean reshuffling the retrieved info to highlight the most important parts and making sure there is no unnecessary information. These advancements help the system to focus on concise and important information and avoid context window overflow of the large language model.

- **Modular RAG**

  The modular RAG architecture improves upon previous versions by being more adaptable and versatile, incorporating various strategies like adding a search module or fine-tuning the retriever. It is gaining popularity since it can handle both step-by-step processing and training the system based upon a specific domain and task.

In parallel, the need for robust evaluation frameworks and benchmarking standards within the RAG domain has garnered significant attention. Current RAG evaluations and benchmark aspects have been proposed by RAGAS and ARES [EJEAS23],[SFKPZ24]. These initiatives aim to facilitate fair comparisons between different RAG systems.

This thesis aims to incorporate the building blocks of both ARES and RAGAS. The metrics in this thesis are inspired by the ones proposed in ARES and RAGAS, namely context relevance, answer faithfulness, and answer relevance. While ARES pre-trains LLM judges along the evaluation process, RAGAS uses computational methods to evaluate a given system. This work integrates both approaches in to offer a more streamlined approach. Current RAG evaluation frameworks include RaLLE, focusing on transparent prompt engineering optimisation [HMN+23], RAGGED concerning the number of retrieved documents [HSWN24], RagBench introducing the metrics context utilization and answer completeness [FBS24] and BERGEN, analysing the usability of datasets for RAG [RDC+24]. They introduce different ideas and methods for evaluating question-answer systems and retrieval of various kinds. All of the above have been developed concurrently with respect to this project, hence have similarities to this framework.

The framework proposed in this thesis tries to offer a different, more lightweight approach, while also providing a solution for ground-truth-less applications. Additionally, the building blocks for an advanced RAG system are provided accompanied by the associated code repository.

1.3 Challenges and aim of thesis

In the following, the current challenges in RAG evaluation are discussed along with possible solutions proposed in this work.

From naive to advanced RAG

There is limited understanding of the impact of each of the different steps in the RAG engineering process. RAG usage is not streamlined, and selecting a design for an existing RAG system offers various possibilities. This work proposes an advanced RAG which is based on the analysis of the results yielded by the developed evaluation framework. An advanced RAG system is designed, improving upon the naive RAG design.

Building an evaluation framework

As the field continues to evolve, developing methods to automatically evaluate RAG systems is crucial to sustain high quality scientific standards. Evaluating a RAG ecosystem is not straight-forward due to the complex nature of the system. It is also not clear which metrics are most suitable for addressing the performance of a RAG system. Thus, a framework will be introduced to systematically evaluate RAG ecosystems based on appropriate metrics. Current RAG evaluation systems and metrics focus on performance of complete RAG ecosystems, without diving into specific components or interactions between them. To address this, the analysis of the pairwise relationship between subsystems and their impact on the overall system performance will be the subject of this thesis. The impact of micro-adjustments to components of the system on the macro-performance of the RAG environment will be quantified.

Real domain application

When building a question answer system for a specific domain, field or business, there is rarely a dataset providing ground truth answers. Hence, it is quite difficult to evaluate such a system, given that RAG system responses cannot be compared to a true answer. This thesis introduces the RAG Triad as a remedy. The underlying hypothesis postulates, that when the main components of a RAG system yield positive results based on a metric, then the overall answer of the system tends to be correct.

CHAPTER 2

Background

This chapter is concerned with explaining the basic RAG process schematically. The naive RAG process shown schematically in figure 2.1 gained popularity shortly after the rise of LLMs like ChatGPT to general awareness [Ray23]. The basic process is structured into four parts. (i) First domain data is processed and stored in a vector index for later retrieval. (ii) Second, the system receives a query/question and retrieves information from a database. (iii) Third, the query is fused with the retrieved information via an instructive prompt. (iv) Fourth, based on query, information and prompt, a LLM produces the final rag output [ZZY+24]. This chapter is concerned about the building blocks of a RAG system, without talking about possible improvements. An advanced RAG approach is addressed in the succeeding chapter.

2.1 Information Storage

This section discusses the first rag component, information retrieval. This involves creating a database, from a given text corpus, data collection or repository. Then, the retrieval model retrieves information from the pre-built database with respect to the given query [ZDD+22]. Below is an explanation of how a vector database is designed and how indexing to the vector database works.

2.1.1 Vector Database — Overview

Vector databases offer several advantages over traditional databases. One key benefit is fast and accurate similarity search and retrieval, as they can identify the most relevant data based on vector distance or similarity. This is essential for applications involving natural language processing, computer vision, and recommendation systems. Traditional databases, by contrast, rely on exact matches or predefined criteria, which may not capture the full semantic or contextual meaning of the data. Additionally, vector databases

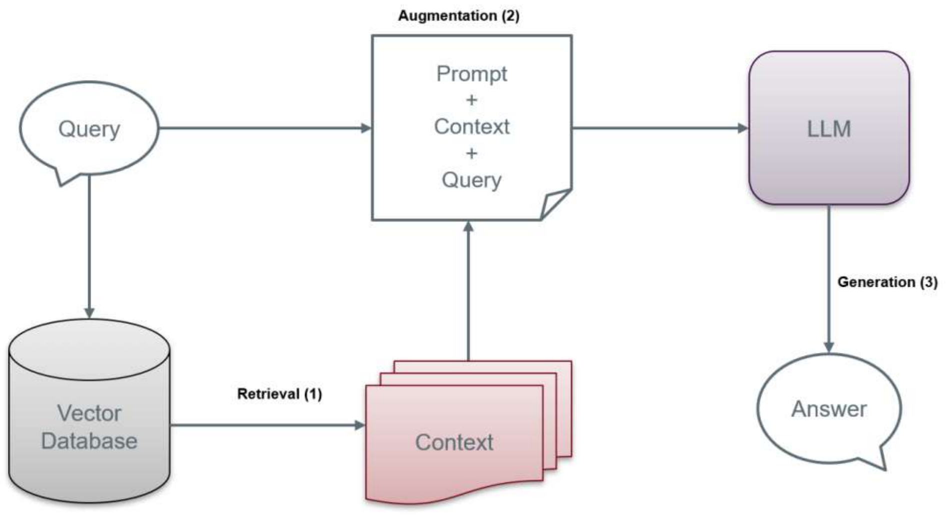


Figure 2.1: Naive RAG

are well-suited for handling complex and unstructured data, such as text, images, or multimedia, which don not conform to the rigid schema of traditional databases. They convert such data into high-dimensional vectors, preserving key features and attributes. Vector databases also excel in scalability and performance, managing large-scale, real-time data processing crucial for modern data science and AI applications [HLW23]. The main principle of a vector database is vectorization [KKB$^+$23]. It aims at converting rich, unstructured data like natural language text, image, videos into numerical vectors, capturing features of the data. Features include language, data size, colours and semantic meaning among others. Unlike traditional databases where a query might ask for the number of players on a specific football team, a query to a vector database can, for example ask, what are the instructions I have to follow when I want to buy a ticket for a football game. Managing the vector databases has gained a lot of traction recently, with the advent of vector database management systems (VDBMS) also called vector indexes. A vector index manages storage and retrieval of the vectorised data efficiently. Vectors are related inherently by their distance to each other. At the indexing step, each data point is inserted into a specified data structure. At the retrieval step, naively an exhaustive search is necessary and the distance between the query vector and all database vector would be computed. A vector index accelerates this process by using appropriate data structures. [Tai24].

A vector index manages the vector database on creation, updating and retrieval. This work uses the open source software *Marqo* [Mar] to manage a custom vector index. The underlying engine is provided by *Vespa* [Ves]. In this tech stack, *Marqo* serves as high level interaction interface, allowing embedding-generation, querying and enriched information retrieval. The *Vespa* vector index supplies the low level foundation by providing a vector store alongside an approximate nearest neighbour storage schema.

2.1.2 Database entries - vectors

The database entries are high-dimensional vectors, encode features such as meaning, word count, sentiment, language, etc. mathematically. The number of dimensions in each vector can vary from single digits to thousands, depending on the chosen embedding mechanism. Vectors are typically generated by applying a transformation or embedding function to an input [CPL19]. The embedding function uses natural language machine learning models, such as a sentence-transformer models. The process of creating such a vector embedding of text involves tokenisation and attention. First, a text is tokenised, meaning it is split up into smaller sub parts. Tokens are similar to words but can also be syllables or include special characters.

$$t_i = [t_1, t_2, t_3, \ldots, t_n] = f_{\text{tokenize}}(\text{input}) \tag{2.1}$$

where

t_i ... The i-th token of the tokenized sequence.

$f_{\text{tokenize}}(\text{input})$ ... A function that takes the input text and splits it into a sequence of tokens.

n: The total number of tokens in the sequence.

Then the tokenized text is parsed by a sentence transformer model and mapped to a high dimensional vector. The output vector is a numerical representation of the features of the input text.

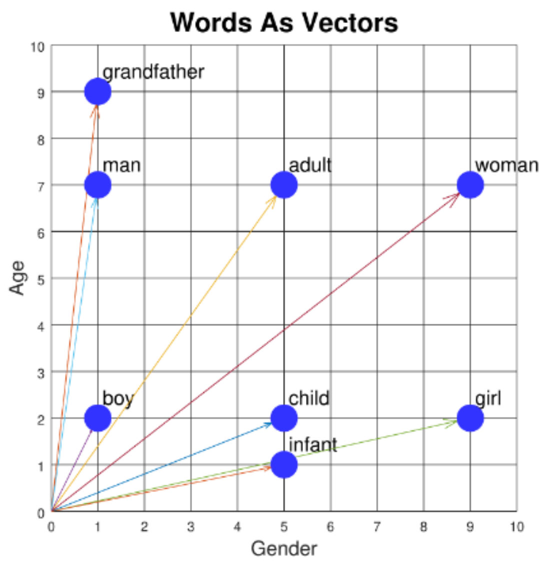
$$\mathbf{v}_i = g_{\text{st}}(t_i) \tag{2.2}$$

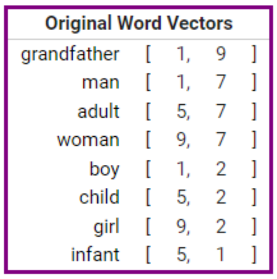
where

$\mathbf{v}_i$ ... The dense embedding vector for the input text

$g_{\mathbf{st}}(t_i)$ ... A sentence transformer function that takes the token t_i and converts it into an embedding vector

An intuitive view of a vector representation has been formulated by Mikikol et al. One, algebraic operations can be performed on vector representation. For instance, when looking for a vector representation of the word “smallest”. One can calculate X = vector(“biggest”) - vector(“big”) + vector(“small”), where X results in the vector





Original Word Vectors		
grandfather	[1,	9]
man	[1,	7]
adult	[5,	7]
woman	[9,	7]
boy	[1,	2]
child	[5,	2]
girl	[9,	2]
infant	[5,	1]

Figure 2.2: Simple word as vector source [Tou]

representation for smallest. Second, they found that, when high dimensional word vectors are trained on a large amount of data, even subtle semantic relationships can be encoded. An example here is that a vector representation of a city includes information on the country it belongs to, e.g. Vienna is in Austria or Paris is in France [MCCD13]. These embeddings enable the computation of distances between data points within the vector space, analogous to calculating distances in three-dimensional Euclidean space. The underlying assumption is that if two data points are close to each other in the vector space, they are similar in semantic meaning [sub].

The main reason for mapping text to a vector is that in a vector space, a distance metric can be defined. The clear mathematical formulation of distance for finite vector spaces allows for the calculation of distance between two vectors, with **cosine similarity** or variations of it, being the most commonly applied method [PWL23]. This distance metric is the basis of measuring similarity within high-dimensional vector spaces. The intuition here is, that if the distance between two points is small, then the difference in semantic meaning is small. The cosine similarity, like most methods used in NLP similarity measures, is based on the dot product [Jur24],

$$\text{dot-product}(\mathbf{v}, \mathbf{w}) = \mathbf{v} \cdot \mathbf{w} = \sum_{i=1}^{N} v_i w_i = v_1 w_1 + v_2 w_2 + \cdots + v_N w_N \tag{2.3}$$

where

$\mathbf{v}$, $\mathbf{w}$ ... embedding vectors

N ... vector dimension

where the cosine follows the normalized dot product,

$$\cos(\mathbf{v}, \mathbf{w}) = \frac{\mathbf{v} \cdot \mathbf{w}}{|\mathbf{v}||\mathbf{w}|} = \frac{\sum_{i=1}^{N} v_i w_i}{\sqrt{\sum_{i=1}^{N} v_i^2}\sqrt{\sum_{i=1}^{N} w_i^2}} \tag{2.4}$$

measuring the alignment of the two vectors. Intuitively, this can be thought of as resembling the inner product of two-dimensional vectors, where a perfect alignment results in an inner product of 1. Whereas, if two vectors are normal to each other, the result is zero.

2.1.3 Vector database — storage

Once data is converted to vector embeddings, it needs to be stored in a so-called vector index in order to have efficient information retrieval later. Due to the high dimensionality of the vector embeddings and the potentially large number of stored vectors n, it would scale with O(n) to calculate the similarity between a query vector and all stored data entries. This is where vector database management system or so-called vector indexes come into place. These systems enable storing vector embeddings in data structures, providing a trade-off between precision and latency. Some popular algorithms are Product Quantization [YCX20], [JDS11], Locality-Sensitive Hashing [ZZW$^+$20] and the Hierarchical Navigable Small World approach [MY18], which creates a hierarchical graph with fast neighbourhood exploration by building a small world network [Tai24]. The vector index used in this work uses the latter HNSW approach. The hierarchical navigable small world (HNSW) algorithm is an k- approximate nearest neighbour algorithm based on the navigable small word (NSW) approach. [HLW23] The navigable small world algorithm uses greedy graph routing to select the k nearest neighbours. The hierarchical extension to the HNSW uses a multi-layer graph, where the one layer contains all data entries and additional layers contain nested subsets of all elements. The multi-layer graph resembles the structure of a skip list. When searching for the elements closest to the element representing the query embedding, a random entry point in the most sparse (top) layer is selected and then greedily routed to the data point of its friends list which is closest to the query embedding. Then, edges are traversed in each layer until a local minimum is found in the most dense (lowest) layer. The k data point closest to the query data point are returned [MY18] [PWL23]. Figure 2.3 shows an NSW graph, while 2.4 shows the evolved multi-layer HNSW graph, schematically.

2.1.4 Vector database — indexing

All tough multimodal data (strings, image, etc.) can be embedded as vectors, this work is concerned with exploring the textual data only. Initially, the data is pre-processed. For the case of textual, the text corpus is divided into smaller segments, or chunks.

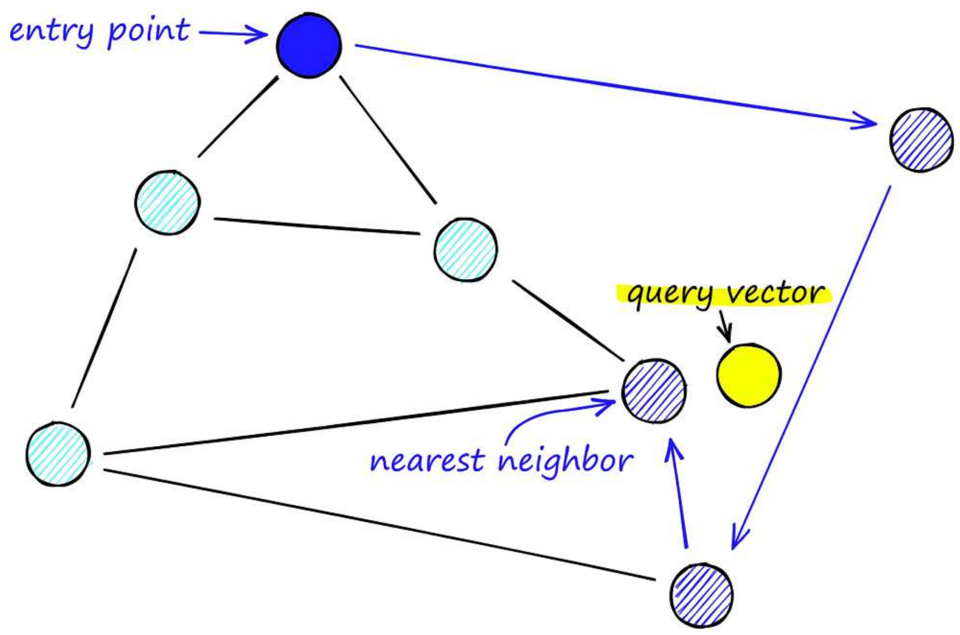


Figure 2.3: NSW, source [Pin]

A chunk is a subsection of a full document; for example, if the document is a single page, a chunk might correspond to a paragraph. Each chunk is represented as a distinct vector embedding within the vector database. Importantly, all vector embeddings are of uniform dimension, regardless of the chunk size. The vector dimension is predefined when initializing the index. This work uses the *flax-sentence-embeddings/all_datasets_v4 _mpnet-base* model to create the embeddings [Hug]. This model was chosen based on [MTMR23] as a trade-off between performance and resource utilisation. The model maps each chunk to a 768 dimensional vector. Then the embedding is stored in the vector index using the HNSW algorithm.

Chunking is done to optimise the semantic distinction of each embedded vector. Note that smaller chunks are not necessarily better, but rather an appropriate, domain dependent size has to be found. For example, a PDF document containing 10 pages and information about various topics is likely too large to be encoded with a distinct and precise semantic meaning and a single word or sentence is probably too small to capture enough complementary context. Remember that no matter the chunk size, the embedding vector always has constant size. The aim is to find a chunking mechanism such that each chunk has a distinct semantic meaning from the surrounding chunks, while still capturing enough macro context. A chunking example is given in Figure 2.5

A chunking function is applied, to extract chunks from a text corpus. This function takes in a text document and outputs text chunks, while fulfilling certain criteria. The chunking strategy determines the size of the chunks and thus also the memory implications. There are many design choices on how to define this strategy, this work focuses on splitting on

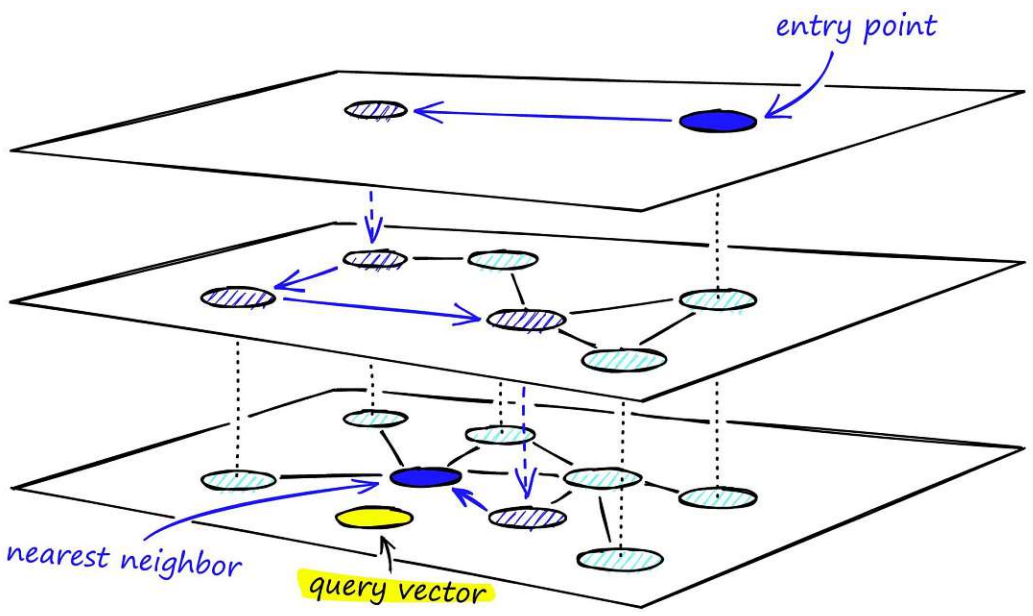
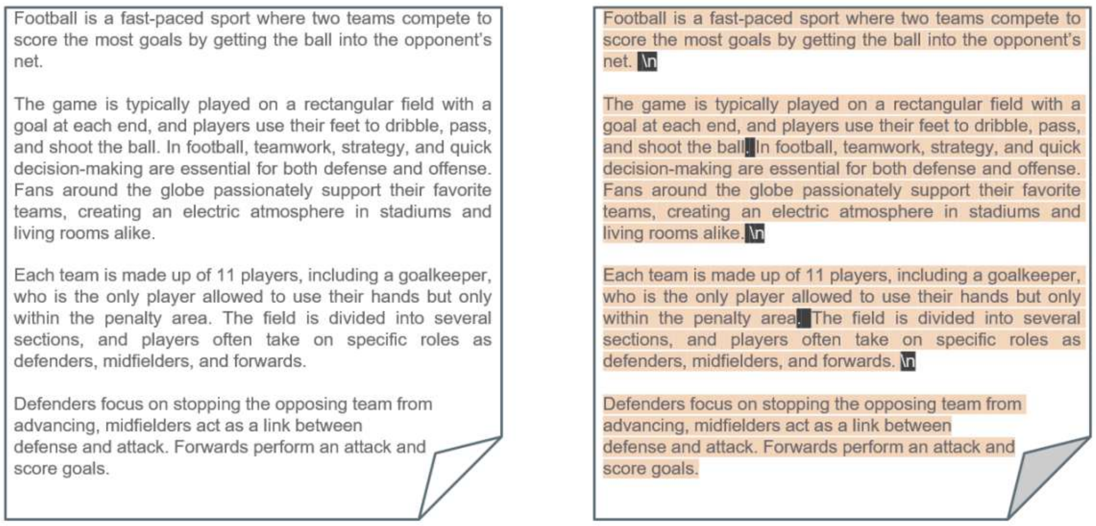


Figure 2.4: HNSW, [Pin]

"separators" while not exceeding a fixed character size for the chunk. A separator is a special textual character, usually resembling a semantic break in written speech, like the end of a sentence or a paragraph. Generic configurations used in this work are:

- Maximal number of characters in a chunk: 256
- Separators: "\n\n" "\n" , " . " , " ! " and " ? "

The number of characters is a chunk is not only an important marker in capturing semantic and related meaning in text, but also has a big impact on the memory footprint of the application. While the exact memory footprint depends on numerous conditions, the chunk size is a defining one. Doubling the chunk size, cuts the number of total chunks which need to be embedded in half and vice versa. Studies have shown, that chunk character counts of 256 and 512 are among the best performing sizes, with respect to accuracy, BLEU and ROUGE retrieval [YYM$^+$24]. The output chunks of the chunking algorithm are subsequently converted to vector embeddings and added to the vector index., using the HNSW schema.



Football is a fast-paced sport where two teams compete to score the most goals by getting the ball into the opponent's net.

The game is typically played on a rectangular field with a goal at each end, and players use their feet to dribble, pass, and shoot the ball. In football, teamwork, strategy, and quick decision-making are essential for both defense and offense. Fans around the globe passionately support their favorite teams, creating an electric atmosphere in stadiums and living rooms alike.

Each team is made up of 11 players, including a goalkeeper, who is the only player allowed to use their hands but only within the penalty area. The field is divided into several sections, and players often take on specific roles as defenders, midfielders, and forwards.

Defenders focus on stopping the opposing team from advancing, midfielders act as a link between defense and attack. Forwards perform an attack and score goals.

Football is a fast-paced sport where two teams compete to score the most goals by getting the ball into the opponent's net. \n

The game is typically played on a rectangular field with a goal at each end, and players use their feet to dribble, pass, and shoot the ball. In football, teamwork, strategy, and quick decision-making are essential for both defense and offense. Fans around the globe passionately support their favorite teams, creating an electric atmosphere in stadiums and living rooms alike. \n

Each team is made up of 11 players, including a goalkeeper, who is the only player allowed to use their hands but only within the penalty area. The field is divided into several sections, and players often take on specific roles as defenders, midfielders, and forwards. \n

Defenders focus on stopping the opposing team from advancing, midfielders act as a link between defense and attack. Forwards perform an attack and score goals.

Figure 2.5: Chunking

2.2 Retrieval

In the retrieval stage, resources are retrieved from a vector index with respect to a given user query. The retrieved context should be as similar as possible to the given query. It is not clear what similar means. The vector index is capable of performing two variants of look up - i.e. two forms of similarity. First, lexical search based on classical word matching search mechanisms. Second, semantic search, also called vector or tensor search, returning results by maximising a semantic similarity measure between the query and the search corpus.

Lexical similarity is only concerned with matching letters, words, and sentences, whereas semantic similarity relates a text to another if they are similar in meaning but not necessarily using the same wording. For example, the word "football" in the North American region means "American football" whereas "football" in Europe is related to what is also known as "soccer". In this case, on the one hand, two articles about football can have high similarity when only concerning lexical similarity but might talk about two different sports where the semantic similarity would be significantly lower as the lexical one. On the other hand, the words football and soccer can have the same semantic meaning, but the lexical similarity is zero, as illustrated in Figure 2.6. Note that the semantic similarity is not maximal here, since the word football can have the meaning of American football and does not have to mean soccer.

The vector index, applied in this work, uses the well studied and widely applied *Okapi BM25* ranking function to perform lexical search. The Okapi BM 25 (BM = best matching) is a function used by search engines to find the best matches to a given search query [RZ09]. The function is based on inverse document frequency (IDF), term frequency (TF) and document length [SMS24]. It scales proportional to the number of

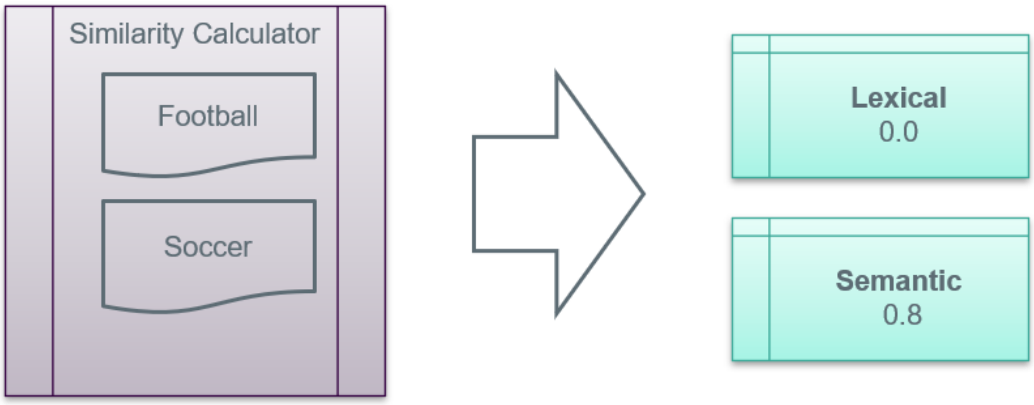


Figure 2.6: Lexical and semantic similarity comparison

exact word matches from words in the query and words in a document. It is defined as,

$$\text{score}(D,Q) = \sum_{i=1}^{n} \text{IDF}(q_i) \cdot \frac{f(q_i,D)\cdot(k_1+1)}{f(q_i,D)+k_1\cdot\left(1-b+b\cdot\frac{|D|}{\text{avgdl}}\right)} \tag{2.5}$$

where

D ... document being scored against the query Q.

Q ... The query containing keywords $\{q_1, q_2, \ldots, q_n\}$.

$f(q_i, D)$ ... The term frequency, of or the number of q_i in document D.

$|D|$ ... The length of the document D in terms of total words.

avgdl ... The average document length in the entire collection of documents.

k_1 ... parameter controlling term frequency saturation, often set between 1.2 and 2.0.

b ... parameter that adjusts the influence of document length, often set to 0.75.

with the inverse document frequency computed as:

$$\text{IDF}(q_i) = \ln\left(\frac{N - n(q_i) + 0.5}{n(q_i) + 0.5} + 1\right) \tag{2.6}$$

where

N ... The total number of documents in the collection.

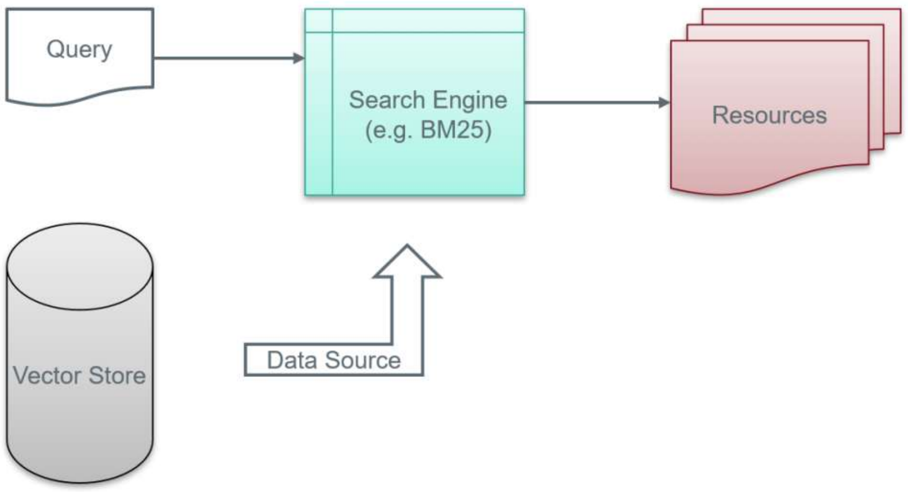


Figure 2.7: Lexical Retrieval

$n(q_i)$ ... The number of documents that contain the keyword q_i.

On the other hand, similarity search is conducted by comparing the pre-indexed vector representations of the data corpus and the online vector representation of the query. An embedding of the query is created using an embedding model, as it was done for all data embedded in the index. The query embedding is projected onto the vector space of the indexed data corpus. Then an approximate KNN algorithm (HNSW in this work) based on a distance metric (e.g. cosine similarity) is applied to retrieve k documents similar to the query [TDVS19].

The vector-database-management-system (VDBMS) at hand allows for retrieval of semantic and lexical resources. One or both can be then be further used in the RAG system. Up to k semantic and up to l lexical sources are retrieved. In a RAG system, a number of sources used downstream is defined. Here, the number of sources used later on by the LLM to generate a system answer is denoted as m. The contextual information downstream can include either lexical, semantic, both sources.

$$S_k = \{\text{Top } k \text{ semantic sources}\} \tag{2.7}$$

$$L_l = \{\text{Top } l \text{ lexical sources}\} \tag{2.8}$$

$$D_m = \text{Top } m \text{ sources from } S_k \cup L_l \tag{2.9}$$

$$k + l \geq m \tag{2.10}$$

An example configuration is:

- $k = 8$
- $l = 8$
- $m = 4$

In this example the l best lexical and the k best semantic sources are combined to the m best overall sources which are then further used in the augmentation. It is not clear how to combine lexical and semantic sources, and also not clear what "best" overall sources means. In the naive RAG, both are used to an equal extent. Thus, in the example above, two semantic and two lexical sources are extracted for further use. An innovative approach is discussed in this work when building an advanced RAG system in the next chapter. This is what is called a hybrid approach, combining both dense and sparse retrieval. Both sparse and dense retrieval sources can benefit the search outcome by leveraging complementary features while making the system more robust to a change in domain [SMS24] [GXG$^+$24].

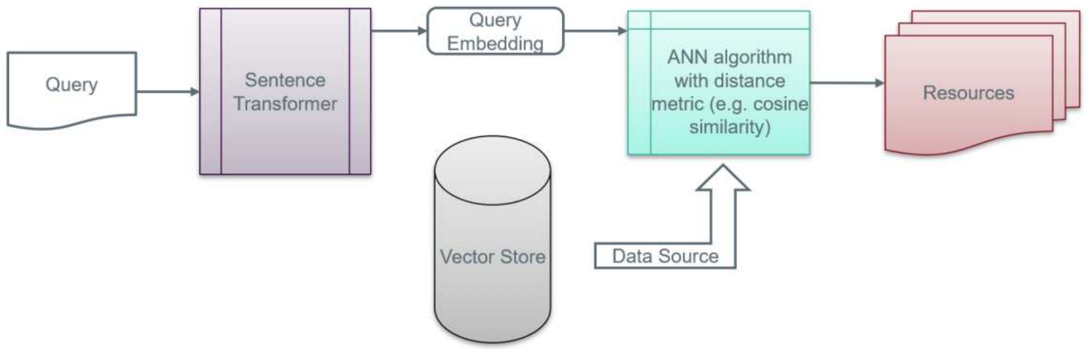


Figure 2.8: Semantic retrieval

2.3 Augmentation

A hand-crafted prompt, along with a user query and selected contextual information, is sent to a large language model (LLM) to generate an answer that integrates information from the provided contexts. While the query and contexts are automatically selected based on the user's input, the design of the prompt itself is tailored by the system designer. The prompt instructs the LLM on how to integrate contextual information into its response to the query, using the principles of prompt engineering. This approach leads the LLM to effectively combine the retrieved resources with respect to the given query, promoting relevant answers. As shown in the prompt engineering literature, structuring prompts in this way is essential for optimising LLM responses [CZLZ24].

The prompt is predefined by the system designer but can be adjusted by the user of the retrieval-augmented generation (RAG) system to suit specific tasks. This prompt serves as a template that directs the LLM on how to address the query in light of the retrieved contexts. In this work, the following generic prompt,

Augment prompt

You are a helpful assistant. Context information is given in the following text. Use the information from the context instead of pre-trained knowledge to answer the question. The answer has to mention explicit details, be explanatory and be short. The answer should refer to the query. If it is a yes or no question, answer with yes or no and then give a brief reasoning. If the answer is not provided in the context. You must say that you do not know the answer. Given the context information and not prior knowledge, answer the following user query: {query}

gives basic guidelines and instruction on how to handle the input and structure the output.

2.4 Generation

The final step in the RAG process is answer generation, where the prompt, query, and retrieved sources are combined and sent to a large language model (LLM) to generate a response to the original query. The response is based on all the design choices made in the system, the underlying data and the quality of the RAG components. The Large Language model (LLM) response, which is the answer to the given query, based on information retrieved from a vector index and fused via an instructive prompt. All LLMs used to generate answers used in this work are based on the transformer architecture and are decoder only [Dah24] [VSP$^+$23].

The context window of a large language model (LLM) can act as a significant constraint in RAG applications. Specifically, it imposes two key limitations. First, the context window defines the total number of tokens an LLM can process in a single inference step. Any tokens exceeding this limit are ignored, meaning the model generates responses based only on the tokens within its allowable context. This restricts the number of retrieved resources that can be incorporated into the response.

Second, when the LLM processes many resources alongside a complex prompt and a lengthy query, it may encounter what is known as the *lost-in-the-middle* problem. Intuitively, this is similar to a human reader focusing primarily on the beginning and end of a long text, the LLM may assign less importance to information in the middle of its input. This issue increases along the number of tokens, potentially leading to a semantic downgrading of information in the middle portion of the input.

CHAPTER 3

Method — Advanced RAG

One of the goals of this work is to move beyond naive RAG to advanced RAG by introducing enhancements on different levels in the RAG process. In the figure 3.1 the figure from the previous section 2.1 is augmented with areas of interest. The figure shows where possible upgrades in the RAG process are identified.

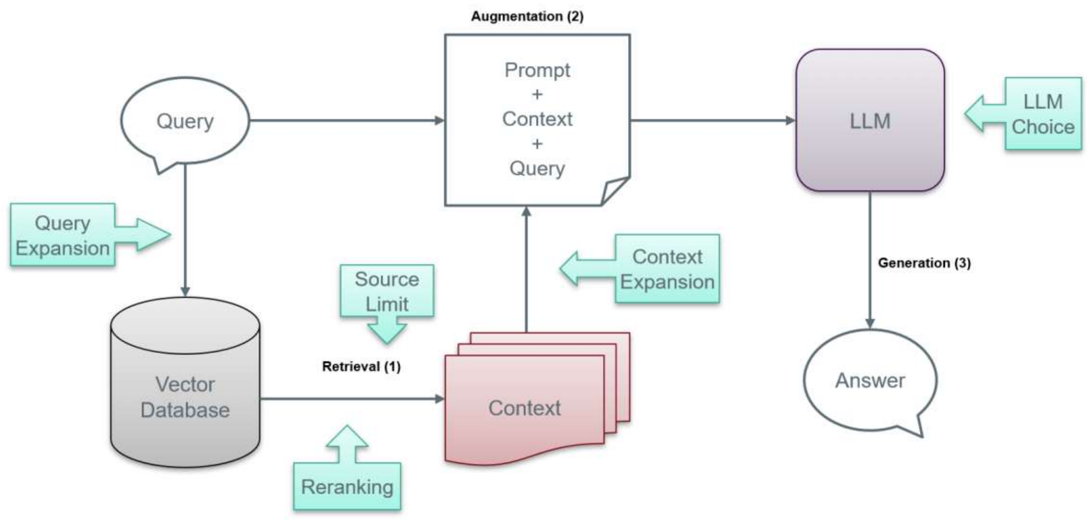


Figure 3.1: Advanced RAG

The advanced RAG pipeline updates the naive RAG pipeline, enriching the RAG ecosystem with additional options. This work discusses the following RAG methods, depicted in Figure 3.1:

- Query Expansion

- Re-ranking
- Context Expansion

Additionally, the upgraded implementation allows for design choices of various system parameters, illustrated in Figure 3.1, of which the following are elaborated:

- Number of retrieved sources
- LLM for answer generation

The optimisation paradigms in this work cover two main approaches: (1) improvements in specific steps, such as methodically selecting sources or enriching contextual information, and (2) parameter tuning, where specific values are chosen from a set of options. Although a range of design choices is possible, this study focuses on five key features, outlined and discussed in detail below.

3.1 Query Expansion

A human written query is possibly not in the optimal shape and form for a language model to extract the necessary information out of the query [MGH+23]. Thus, the goal of query expansion is to create additional queries resembling the meaning of the original query.[GXG+24]. The original query plus the additional queries are then used to retrieve data from the index. The schematic idea of query expansion is shown in Figure 3.2.

The hypothesis here is that different queries lead to different search results, where the search results yielded by the add-on queries could be beneficial to answering the original question. The search space is enlarged, which increases the possibility of retrieving valuable information. For example, the question, “How many people live in the largest city of Austria?” might be enhanced by additional questions like “How many people live in Vienna?”, “What is the population of Austria”, “How many inhabitants does the largest city of Austria have?”. If query expansion is enabled, the query is initially sent to a large language model with a prompt giving the LLM instruction on how to create similar queries. The LLM then returns a list of queries, including the original query.

The following prompt is used to multiply a query:

Query expansion prompt

You are an information system that helps process user questions. Provide information such that a vector-database retrieval system can find the most relevant documents.

Expand the following query: query to n relevant queries which are close in meaning but are simplified. Focus on the main topic of the query and avoid unnecessary details. Focus on nouns and verbs in the query and expand them. Use the same Python list structure in your answer as in the examples below. The first query in the list should be the original query.

Here are some examples:

Query: What is the capital of Norway? Example expansion if the number of queries needed is 2: ["What is the capital of Norway", "City Norway"]

Query: Drugs for cancer treatment? Example expansion if the number of queries needed is 3: ["Drugs for cancer treatment?", "Cancer drugs", "Health anti-cancer drugs"]

Query: What positions are there in a football team? Example expansion if the number of queries needed is 4: ["What positions are there in a football team?", "Football team positions", "Football team", "Rules in football"]

Structure your response as a Python list of strings, where each string is a query. The length of the Python list is n. The answer should be just this list and nothing else.

Here, explanatory rules and simple examples are provided to guide the generator. All queries are then sent individually to the index and context information is retrieved. The contexts are then processed further as if they would have been retrieved with just a single query.

3.2 Re-ranking

Re-ranking means taking the sources retrieved from the vector index with respect to a given query and ranking them according to a re-ranking mechanism, schematically shown in Figure 3.3. This ranking is then used to select the most relevant sources for the RAG system. Sources retrieved from the index are ranked based on a score assigned to each. These ranked results inform the RAG process. If the number of sources required for the final answer generation is less than the total retrieved, the re-ranked list serves as a guide for cut-off selection.

Typically, the combined count of lexical and semantic sources exceeds the number allowed for downstream use in generating the RAG answer. Determining the optimal mix of semantic versus lexical sources can be challenging. Without a systematic approach to

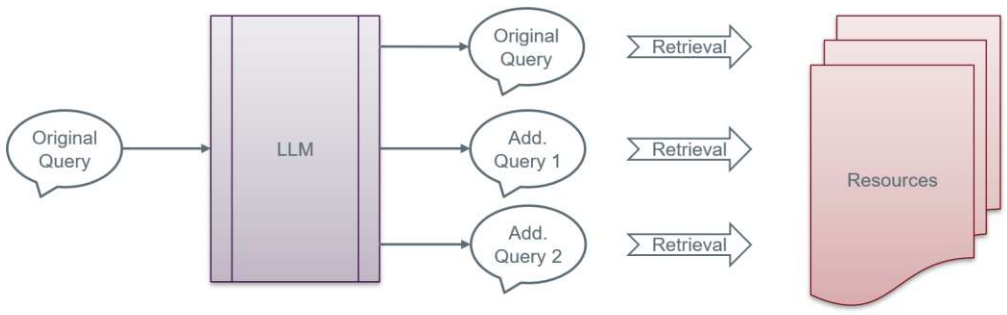


Figure 3.2: Query expansion

prioritizing contextual information, the system relies on a predefined balance between lexical and semantic inputs—often an even split. However, some queries or domains may perform better with an uneven distribution. By applying a re-ranking mechanism, the system gains flexibility, allowing context selection to adapt based on the specific task at hand [GRC+22]. Two main scoring methods are used for re-ranking: semantic re-ranking and reciprocal rank fusion (RRF).

Semantic re-ranking uses a BERT-based model to assess the similarity between the query and each context by calculating the distance between vector embeddings. This model generates a similarity score by comparing each context to the query, and the contexts are then ranked in order of similarity [NC20].

Reciprocal rank fusion (RRF) integrates two sets of ranked lists—in this case, semantic and lexical search results. The RRF function maintains the integrity of each list while combining them into a final ranking. Additionally, if a context appears in both lists, it is promoted in the final ranking. RRF orders the contexts, generally referred to as documents, using a straightforward ranking formula [CCB09]:

$$\text{RRF-score}(d \in D) = \sum_{r \in R} \frac{1}{k + r(d)} \tag{3.1}$$

where:

$d \in D$ ... document d within the set of documents D.

R ... set of ranked lists or sources from which documents are retrieved.

$r(d)$ ... rank of document d in a given ranked list r.

k ... a constant used to prevent division by zero and adjust the weight of rankings.

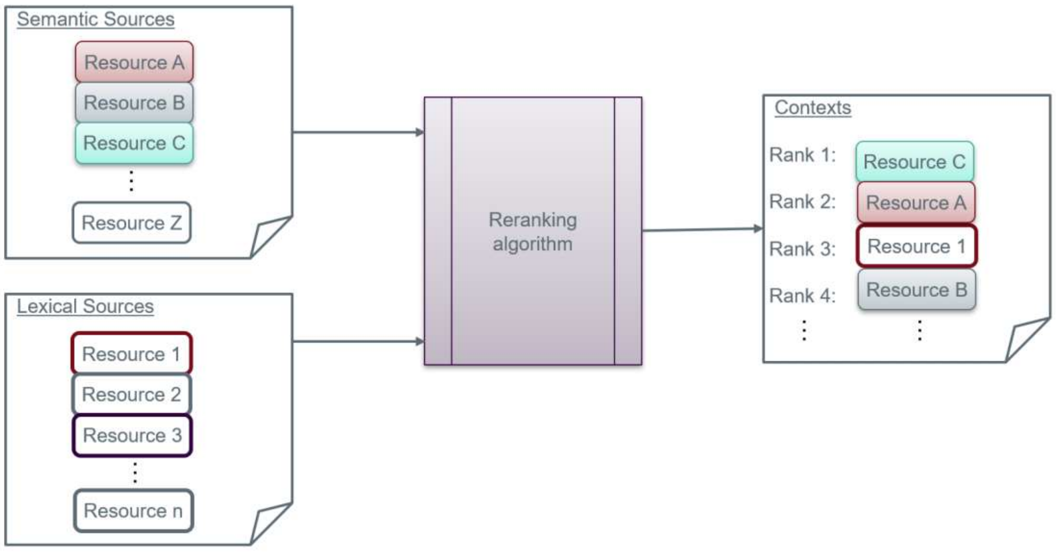
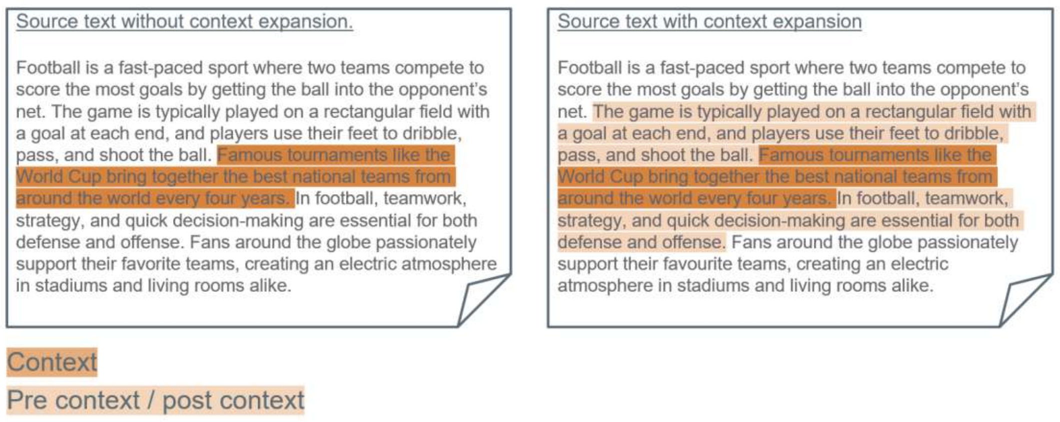


Figure 3.3: Re-rank engine

3.3 Context Expansion

Context expansion, also referred to as sentence-window retrieval, involves enriching a retrieved resource by adding related information. Specifically, when a particular text chunk is retrieved, the preceding and succeeding chunks are also included. This approach leverages the improved retrieval performance achieved with smaller text chunks. Importantly, context expansion does not increase computation during the retrieval phase, instead, it distributes the computational load to the indexing step. Since chunking algorithms retrieve only parts of a larger context, including surrounding information can enhance the relevance of the retrieved data.The surrounding chunks might be farther away in the vector database than they are in the original text, so they may not appear in the initial retrieval but could still provide valuable context. Retrieving smaller chunks also accelerates the process. Context expansion resolves this by adding context to the retrieved chunk.

The pre- and post-context setup is established during the document chunking stage. As each chunk, enriched with additional fields as it is being created iteratively by scanning over the text corpus, as illustrated in Figure 3.4. The additional fields are pre context and post context. Later, when vector search is applied, the pre- and post-contexts are automatically included with each retrieved chunk without requiring extra computation [ENFO24].



Source text without context expansion.

Football is a fast-paced sport where two teams compete to score the most goals by getting the ball into the opponent's net. The game is typically played on a rectangular field with a goal at each end, and players use their feet to dribble, pass, and shoot the ball. Famous tournaments like the World Cup bring together the best national teams from around the world every four years. In football, teamwork, strategy, and quick decision-making are essential for both defense and offense. Fans around the globe passionately support their favorite teams, creating an electric atmosphere in stadiums and living rooms alike.

Source text with context expansion

Football is a fast-paced sport where two teams compete to score the most goals by getting the ball into the opponent's net. The game is typically played on a rectangular field with a goal at each end, and players use their feet to dribble, pass, and shoot the ball. Famous tournaments like the World Cup bring together the best national teams from around the world every four years. In football, teamwork, strategy, and quick decision-making are essential for both defense and offense. Fans around the globe passionately support their favourite teams, creating an electric atmosphere in stadiums and living rooms alike.

Figure 3.4: Context Expansion

3.4 Source Count

The source count indicates the number of sources, after retrieval, that are ultimately used for answer generation. Although many sources may be retrieved, re-ranked, and enriched, the source count reflects the actual number of sources utilized in the RAG pipeline.

The number of sources used can significantly affect the quality of the LLM-generated answer. Too few sources may not provide enough information, while too many can introduce noise to the context or exceed the context window of the LLM. Consequently, the number of sources is balanced to ensure the LLM receives the right amount of information in order to answer the original query and not overflow the LLM. Lexical sources (k) and semantic sources (l) are combined, ranked, context expanded, and filtered through a selection process. The top l sources from this process are then used in downstream tasks. The number of sources l is the *source count.*

3.5 Language Model Choice

Different models are designed with varying architectures, including differences in layer design, the number of parameters per layer, the number of layers, attention mechanism parameters, text embedding methods and size of context window. Additionally, each model is pre-trained on different datasets, contributing to unique strengths and weaknesses across LLMs [NKQ$^+$24]. The specific LLM used in the RAG process may vary depending on the requirements of the task. The LLM selected to answer the original query is the *language model choice.*

CHAPTER 4

Evaluation framework — Evaluating RAG

This work introduces a comprehensive evaluation framework for RAG systems, offering multiple retrieval metrics to support the analysis of various performance indicators. The framework empowers the evaluation from naive to advanced RAG, by enabling data-driven decision-making. The framework addresses the system using the following key performance indicators:

- **Correctness**: Evaluates whether the RAG system's answer aligns with the ground truth.
- **Gold standard context matches**: Determines if the contextual information retrieved from the vector index matches the provided gold standard contexts.
- **RAG Triad**: Assesses whether the system's sub-results are consistent and in line with the overall system answer.

The following scoring metrics are applied:

- **ROUGE** score: Measures the overlap of subsequences in candidate and reference texts
- **LLM-judge**: Analyses texts with respect to an instructive and domain specific prompt
- **Recall@k**: Calculates the proportion of k retrieved sources over all relevant sources.

A graphic overview shown in Figure 4.1 illustrates the evaluation workflow: starting with a given dataset, progressing through the RAG pipeline, entering the evaluation framework and yielding results.

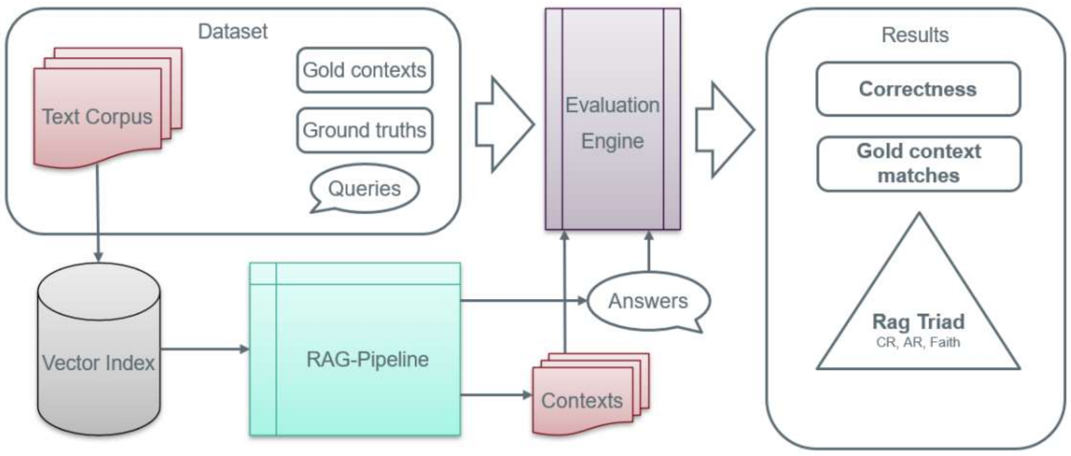


Figure 4.1: Evaluation workflow

4.1 Performance indicators

Evaluating a complex RAG system presents several challenges. It is not obvious how to approach this evaluation due to multiple intermediate steps and correlated sub-results that influence each other. Although data preparation, indexing, retrieval, and large language model (LLM) answer generation are implemented as separate processes, each part impacts the system's final outcome. The main challenge, therefore, is determining how to effectively evaluate both individual components and the overall system. To resolve this, the following performance indicator are introduced to classify the performance of the system as a whole along with its subcomponents.

4.1.1 Correctness

Correctness evaluates how well the RAG system answer aligns with the ground truth answer. While this concept is straightforward, it is challenging to measure, as natural language allows for semantically identical answers to be phrased differently. The closer the RAG system's answer is to the ground truth answer in meaning, the better the RAG system is performing.

4.1.2 Gold standard context matches

Gold context matches assess how well the retrieved contexts align with a gold standard set of retrievable contexts. Not all datasets provide gold standard resources, especially in real-world applications. A match between the ground truth resources and those retrieved from the vector database suggests a well-functioning system. Classical retrieval metrics can be applied here to measure how well ground truth contexts are retrieved from the vector index. The wording gold standard is chosen to clarify that the metric is not of ground truth nature. It is more appropriate since the optimal retrieved contexts may be the gold standard, but they do not have to contain the (full) ground truth. Gold standard contexts here means, they are deemed “optimal” for answering the question at hand.

4.1.3 RAG Triad (Context Relevance, Faithfulness, Answer Relevance)

The RAG Triad is a performance indicator addressing the relationships between RAG subcomponents and intermediate results without relying on ground truth answers or gold standard resources. The schema is illustrated in Figures 4.2 and 4.3.

The RAG Triad in this work is an evolution, and formalisation, of the RAG Triad idea, first proposed by *truera* [Mad]. The Triad hypothesis proposed in this work is that if all subsystems produce high-quality intermediate results, the overall system answer will be high-quality accordingly, assuming the originally indexed corpus is accurate. This is not necessarily true for information management systems and thus requires further examination and proof [Hua10]. In return, poor intermediate results, yield a low-quality outcome. The aim is to find a correlation between the overall correctness scores and the RAG scores across a given dataset.

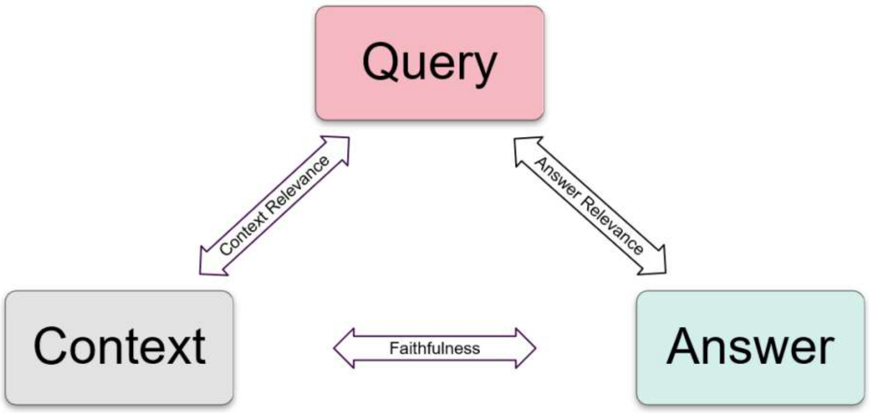


Figure 4.2: Triad schema

The RAG Triad can be formally defined, let the sets be

- Q: the set of given queries, where each query is denoted as q_i.

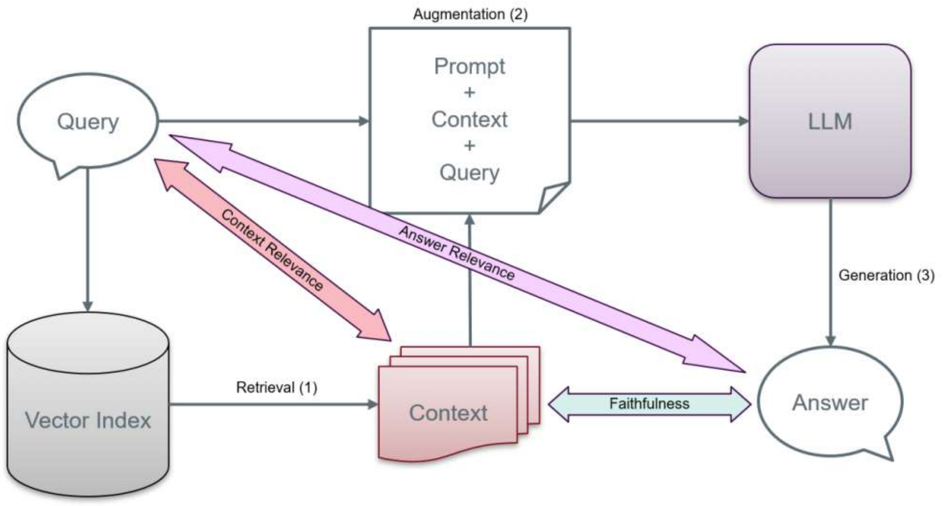


Figure 4.3: Triad in RAG workflow

- A: the set of answers, generated by the system, where each answer is denoted as a_i.
- C: the set of retrieved items or contexts associated with each query, where each context is denoted as c_j, each query has multiple related contexts
- G: the set of given ground truth answers, where each ground truth answer is denoted as g_i.

The evaluation functions are defined as

- Answer Relevance: $\mathrm{AR}(q_i, a_i)$ that measures how relevant an answer a_i is to a query q_i.
- Context Relevance: $\mathrm{CR}(q_i, c_i)$ that measures the relevance of a context c_i to a query q_i.
- Faithfulness: $\mathrm{F}(a_i, p_i)$ that evaluates how faithful an answer a_i is to the contexts it draws information from c_i.

The RAG Triad is computed by summing the maximum scores over all queries q_i and dividing by the number of queries in the dataset:

$$\text{RAG Triad} = \frac{1}{|Q|} \sum_{q_i \in Q} \left(\max_{c_i} \mathrm{CR}(q_i, c_i) + \mathrm{AR}(q_i, a_i) + \max_{c_i} \mathrm{F}(a_i, c_i) \right) \tag{4.1}$$

Here:

- $\sum_{q_i \in Q}$: The sum over all queries in the dataset.

Using the maximum function here is an arbitrary choice and stated here as an example. The framework also implements options for substituting $\max_{x_i}$ with other options, such as the arithmetic mean or a function that selects the first element of the set. Here, “first element” refers to the highest ranked element returned by the RAG pipeline (not by the evaluation framework). The RAG Triad score aggregates the top scores for context relevance, faithfulness, and answer relevance across the all queries, providing a measure of the overall quality of the important sub-results of the RAG pipeline.

Note that the RAG Triad computation does not involve any computation concerning the ground truth. The hypothesis assumes a correlation between intermediate results and end result. A possible correlation between the RAG Triad scores and the answer correctness scores is examined. Here q_i is the $i-th$ query and g_i is the $i-th$ ground truth. This study provides insights into potential correlations and evaluates the validity of the hypothesis.

4.2 Scoring metrics

Quantifying quality in natural language processing and retrieval remains an active research, with no single generally valid approach. Typically, a combination of sequence matching and semantic matching techniques is used. In the following, three scoring metrics are introduced to establish a systematic way of categorizing the results. These scoring metrics are then applied to evaluate system answer correctness based on a ground truth, calculate retrieval recall as well as to verify the RAG triad hypothesis.

4.2.1 LLM-judge

An LLM can be utilized to evaluate how well a candidate text (e.g. a RAG answer) corresponds to a reference text (a given query). Both text segments, along with a tailored prompt, are input into the LLM in few-shot fashion, which assesses the relationship between the texts based on the prompt’s instructions. It is important to note that the LLM-judge metric is computationally expensive compared to the following metrics, since for every score between any two strings, a new LLM call has to be issued.

The LLM can provide a detailed evaluation of the alignment between a candidate and a reference text, especially when guided by an instructive prompt. LLM-based judgments have demonstrated reasonable agreement with human assessments, as noted by Zheng et al. [ZCS+23].

In contrast, methods based purely on semantic similarity scores or word matching algorithms apply a uniform rule set when computing a score between two texts, regardless of task-specific distinctions. LLM-judges, however, allow for customisable comparisons: a prompt can specify the particular conditions under which two texts should be compared. For instance, one might prompt the LLM-judge with “Does text A answer the question posed by Query using the information in text B?" or "Is text B grounded in the content of text A?" Such instructive prompts enable the metric to adjust to the task while keeping the underlying language model unchanged.

For each specific task, a customized prompt is designed, combined with the texts to be evaluated, and then forwarded to the LLM-judge. The LLM returns a score between 1 and 5, with 1 representing the lowest rating and 5 representing the highest. For compatibility purposes, the score is normalized to a range between 0 and 1 during post-processing. The prompts to instruct the LLMs on scoring are to be found in the appendix.

4.2.2 ROUGE score

ROUGE, or (Recall-Oriented Understudy for Gisting Evaluation) is a recall focused metric in the field of natural language processing [Lin04b]. Initially developed for text summarisation and translation tasks, it compares subsequences of a given text with respect to a reference text. More precisely, it measures the overlap of n-grams between a candidate (generated) text and a reference (ground truth) text. All tough there are many variations of ROGUE scoring, the most common one is ROUGE-n, concerning overlap of word n-grams.

It is given by the following formula,

$$\text{ROUGE-N} = \frac{\text{Count}_{\text{match}}(\text{gram}_n)}{\text{Count}_{\text{ref}}(\text{gram}_n)} \tag{4.2}$$

where,

- n ... length of the n-gram and
- $\text{Count}_{\text{match}}(\text{gram}_n)$ ... is the number of n-grams co-occurring in the candidate text and the reference text and
- $\text{Count}_{\text{ref}}(\text{gram}_n)$ ... is the number of n-grams occurring in the reference text.

In this work,ROUGE-1 (n=1) recall is used. This metric is chosen because short RAG answers and ground truths are common, and ROUGE captures these well when n is small.

The ROUGE score metric offers advantages as a scoring metric compared to semantic similarity scorers, (such as LLM-judge or cosine similarity based on embeddings), primarily due to its computation speed. On the one hand, ROUGE scores have shown strong correlation with human evaluations for translation tasks and text summaries [Lin04b]. On the other hand, studies have shown that ROUGE does not always align with human summaries, especially on texts with differing length, grammar and wording. Thus, ROUGE scores have to be related to other metrics in order to check for consistency.

Alignment of ROUGE scores with human judgment increases for short sentences with similar wording. This is particularly useful when rating the correctness of RAG answers compared to a ground truth, since candidate and reference texts tend to be of similar size [SRBR23] [Lin04a]. Both reference and candidate sentence are processed by the ROGUE scoring algorithm, yielding a score between 0 and 1.

4.2.3 Scoring Resources with gold standard, recall@k

Retrieved resources are compared against gold standard resources, using recall@k metric, with Petroni et al. [PPF⁺21]. In this work, the number of retrieved items is also referred to as the number of sources or contexts.

$$\text{Recall@k} = \frac{|R_k|}{|R|} \tag{4.3}$$

where:

- $|R_k|$ is the number of relevant items retrieved within the retrieved k items.
- $|R|$ is the total number of relevant items to the query.

4.3 Experiment Design

This section describes the experiment setup, covering dataset selection, parameter space choices, and the workflow of the RAG evaluation method.

4.3.1 Datasets

When selecting a dataset for RAG evaluation, memory requirements, time efficiency, and suitability for the RAG method were considered. To evaluate RAG, the dataset must include a text corpus that can be indexed, along with a set of questions for querying the system. To address the system correctness, ground truth answers are necessary. Additionally, to evaluate retrieval quality with traditional methods, the dataset has to provide a set of gold standard retrieval documents alongside each question that serve as ground truth for answering each question. Datasets with these characteristics are referred to as question-answer-passage datasets. They form a special kind of dataset class, especially useful for RAG evaluation.

Let C represent the text corpus, where each element $c_i \in C$ is a document or text passage in the corpus.

$$C = \{c_1, c_2, \ldots, c_n\} \tag{4.4}$$

Let Q represent the set of questions, where each element $q_i \in Q$ is a question that can be sent to the system.

$$Q = \{q_1, q_2, \ldots, q_m\} \tag{4.5}$$

Let A be the set of ground truth answers corresponding to each question. This is designed as a set of tuples, where each tuple (q_i, a_i) pairs a question $q_i \in Q$ with its correct answer a_i.

$$A = \{(q_i, a_i) \mid q_i \in Q\} \tag{4.6}$$

Set of Ground Truth Documents (Optional): To evaluate retrieval quality, let D represent a set of ground truth documents. This set is composed of tuples (q_i, D_i), where $D_i \subseteq C$ is the subset of documents from C that ideally answer question q_i.

$$D = \{(q_i, D_i) \mid q_i \in Q, D_i \subseteq C\} \tag{4.7}$$

Given these sets, the question-answer-passage dataset QAP is defined as a collection of tuples, each of which contains a question q_i, its ground truth answer a_i, and optionally, a set of ground truth documents D_i:

$$\text{QAP} = \{(q_i, a_i, D_i) \mid q_i \in Q, a_i \in A, D_i \subseteq C\} \tag{4.8}$$

The defined QAP dataset provides all the necessary components for evaluating both answer and retrieval quality for a given RAG system. The selected datasets have been crafted for this purpose and can be found in the hugging-face library [Hug24]. The chosen datasets are:

- **Mini-Wiki Dataset** The dataset draws its content from a Wikipedia question answer passage set [SHH08]. It features ground truth answers, but no gold standard retrieval passages.
- **Mini-BioASQ QA Dataset** This dataset is derived from the BioASQ-QA dataset and represents the complete collection for the year 2023. It is a subset of the larger BioASQ dataset, residing in the biomedical domain from 2014 to 2024 [KNBP23]. It provides ground truth answers, as well as gold standard retrieval passages.

They have the following specifications:

Category	**Mini-Wiki**	**Mini-BioASQ-QA**
Corpus	3000 documents $\simeq$ 18,000 chunks	40,000 documents $\simeq$ 230,000 chunks
Question-Answer	918 QA pairs	4,700 QA pairs

Table 4.1: RAG datasets scope

In recent RAG literature and analyses, datasets deriving from the KILT [PPF+21] corpus are frequently used. The KILT corpus, in its raw text form (char) demands 29.37 GB of storage. This expands to approximately 300 GB of memory when indexed as a vector embedding, based on an estimated six kilobytes per embedding and a factor of two with respect to the HNSW data structure. These datasets include the well known Hotpot-QA [YQZ+18], Truthful-QA [LHE22] and Natural-Questions [KPR+19] among others. However, these datasets were not chosen for this study due to computational feasibility, more specifically due to high processing times and vast memory requirements. The computational resources available in this work are a single Nvidia A16 GPU and a single Nvidia A40 GPU. The former is in a memory share between the vector index (14 of 16 GB) and the semantic re-ranker (2 of 16 GB). Whereas, the latter A40 GPU is used for LLM inference, albeit not exclusively for this work. The equipment at hand is sufficient to carry out meaningful studies, but it does not scale to Nvidia's A100 and RTX4090 GPUs, which are dedicated to AI computations and are often used when performing analysis on the KILT datasets. A more detailed overview of the computational resources available in this project are in Appendix III: Hardware. The memory specifications, for the KILT corpus and Q-A sets, are given in Table 4.2.

Category	KILT
Corpus	6 million Wikipedia pages $\simeq$ 50 million chunks
Question-Answer	Trivia-QA: 95,000 QA pairs Kilt-Natural Questions: 307,000 QA pairs

Table 4.2: KILT dataset scope

4.3.2 Evaluation space

The RAG pipeline was constructed by making specific design choices and parameter settings. This work focuses on the evaluation space outlined in Table 4.3, detailing the scope and dimension of each method.

The evaluation process is conducted in two stages. First, a grid search is performed, considering all possible parameter permutations. Second, a subset of parameters is identified as having the most potential to reduce hallucinations and improve system correctness. This subset is then further examined on both full datasets.

RAG design choice	Possible Values	Dimension
Context Expansion	Off, On	2
Number of Sources	1, 2, 3, 4, 5, 6	6
Re-ranking	Off, RRF, Semantic	3
Query Expansion	Off, 2, 3	3
LLM for Answer Generation	Llama3.1, Gemma2, Mixtral, Llama3.2	4

Table 4.3: Parameter space and dimension

Details and specifications on the examined LLMs can be found in Appendix II: software.

4.3.3 Experiment run

The experiment setup was conducted using the following methodology:

1. Pipeline and evaluation run for subset of the dataset but full RAG parameter space.
2. Determine influential RAG parameters.
3. Run parameter evaluation for full datasets with selected parameter setting to determine the impact of specific interesting parameters.
4. Run RAG Triad evaluation
5. Run classical retrieval evaluation (gold passages) to determine benefit of re-ranking method in full Mini-BioASQ dataset.

CHAPTER 5

Results

This section presents the results generated by the evaluation framework, capturing the influence of various parameter settings and design choices in RAG process through multiple metrics. Two data sources were used in the RAG pipeline, and results were automatically processed through the evaluation framework.

5.1 Method influence

This subsection examines how variations in RAG methodology impact overall correctness, retrieval quality, and how sub systems influence each other. It aims at identifying whether parameter or method changes have a significant impact on the performance of the system and, if so, quantify the impact. The following sections discuss the influence of each specific method or parameter, including re-ranking, query expansion, context expansion, and the number of sources used downstream. The experiments explore the full parameter space, covering all possible parameter combinations: re-ranking (3 options), query expansion (3 options), context expansion (2 options), and number of sources (6 options), resulting in a total of 108 parameter combinations.

The results are visualized in histogram plots, filtered according to each method. Each histogram entry represents a single pipeline and evaluation run. Due to computational constraints, both datasets were downsampled to 100 query-answer tuples (Wiki) and query-answer-passage tuples (BioASQ), respectively. The tuples were selected at evenly spaced intervals to ensure a roughly uniform representation. Experiments for each design choice were conducted using the Mini-Wiki and Mini-BioASQ datasets and evaluated with ROUGE-1 recall and LLM-based judges. Thus, four distinct plots to assess the impact of each parameter on the RAG pipeline were produced. These plots exemplify the possible types of evaluations by the developed RAG evaluation system. All histogram plots follow the same structure, the y-axis indicates the frequency of items in each bin, while the x-axis displays the mean correctness. Here, mean correctness refers to the arithmetic mean correctness across all items in a single pipeline run (i.e., a specific parameter setting).

5.1.1 Re-ranking

The retrieved sources are utilized in one of three ways: directly without re-ranking, re-ranked using reciprocal rank fusion (RRF), or re-ranked semantically using a sentence transformer model. In the Mini-Wiki dataset, the choice of re-ranking method significantly enhances retrieval accuracy, with both re-ranking approaches outperforming the use of non-ranked sources. Similarly, in the Mini-BioASQ dataset, re-ranking demonstrates a clear advantage, with an increased improvement compared to retrieval without re-ranking. Across all datasets and evaluation metrics, semantic re-ranking with a sentence transformer consistently performs as well as, or better than, RRF re-ranking. The semantic re-ranker outperforms the RRF re-ranker in most cases, with respect to ground truth answer correctness. An exception to this trend is observed in the Mini-Wiki dataset when evaluated with the ROUGE scoring method, where RRF and semantic re-ranking yield comparable results.

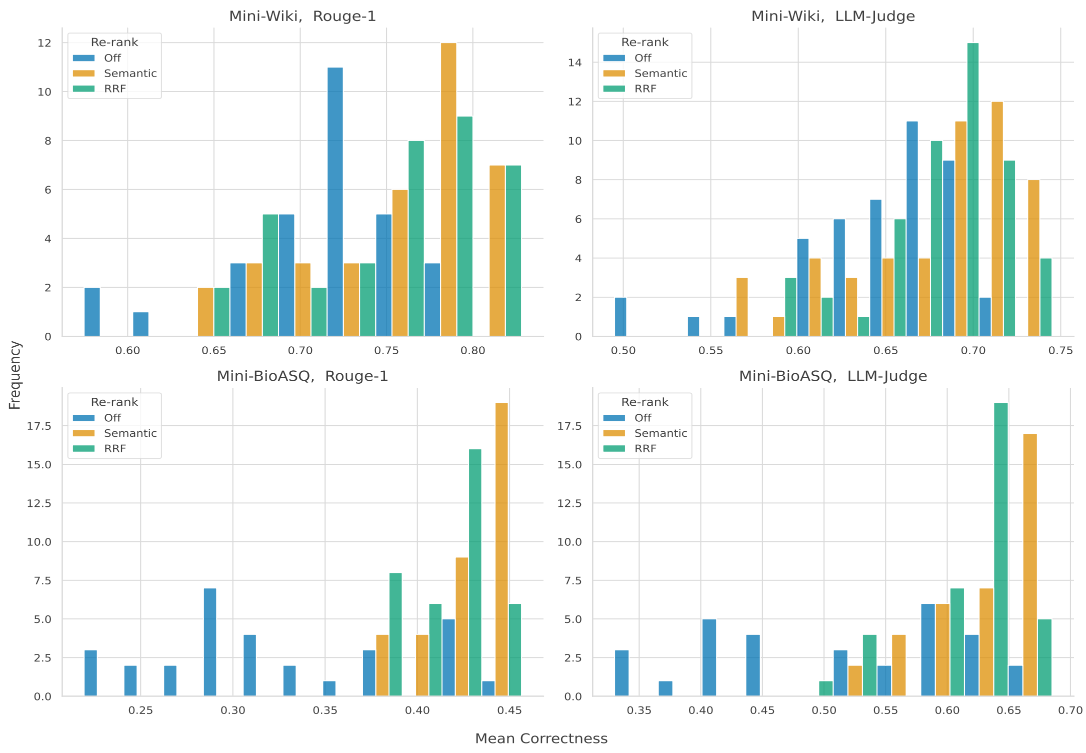


Figure 5.1: Re-rank - All parameter permutations

5.1.2 Query expansion

The impact of query expansion on answer correctness was examined. A query expansion factor of 1 represents no expansion, where the query is forwarded unchanged. In contrast, query expansion factors of 2 and 3 generate multiple artificial queries to potentially enhance retrieval performance. In the Mini-Wiki dataset, query expansion does not improve correctness for either metric. On the contrary, the results are more closely aligned with the ground truth when no query expansion is applied. A similar trend is observed in the Mini-BioASQ dataset, where the differences in correctness between no expansion and expansion factors of 2 or 3 are negligible.

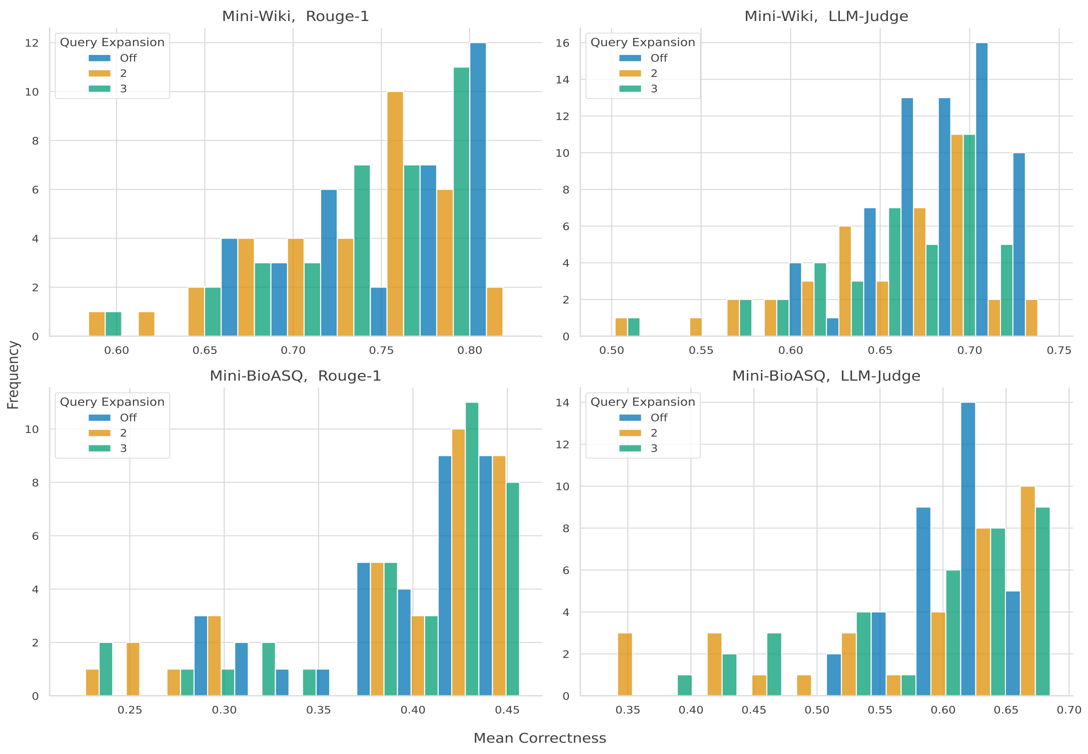


Figure 5.2: Query Expansion - All parameter permutations

5.1.3 Context Expansion

The impact of context expansion on correctness is consistent across both LLM-based judges and ROUGE-1 recall metrics, with minor variations between the datasets. In the Mini-Wiki dataset, enabling or disabling context expansion yields nearly identical correctness results. Whereas for the Mini-BioASQ dataset, a significant improvement in correctness is observed for RAG systems when context expansion is enabled. This could be attributed to the increased complexity of questions and contextual information in the latter dataset. The Mini-BioASQ dataset includes larger resources compared to the Mini-Wiki dataset and requires reasoning across multiple resources.

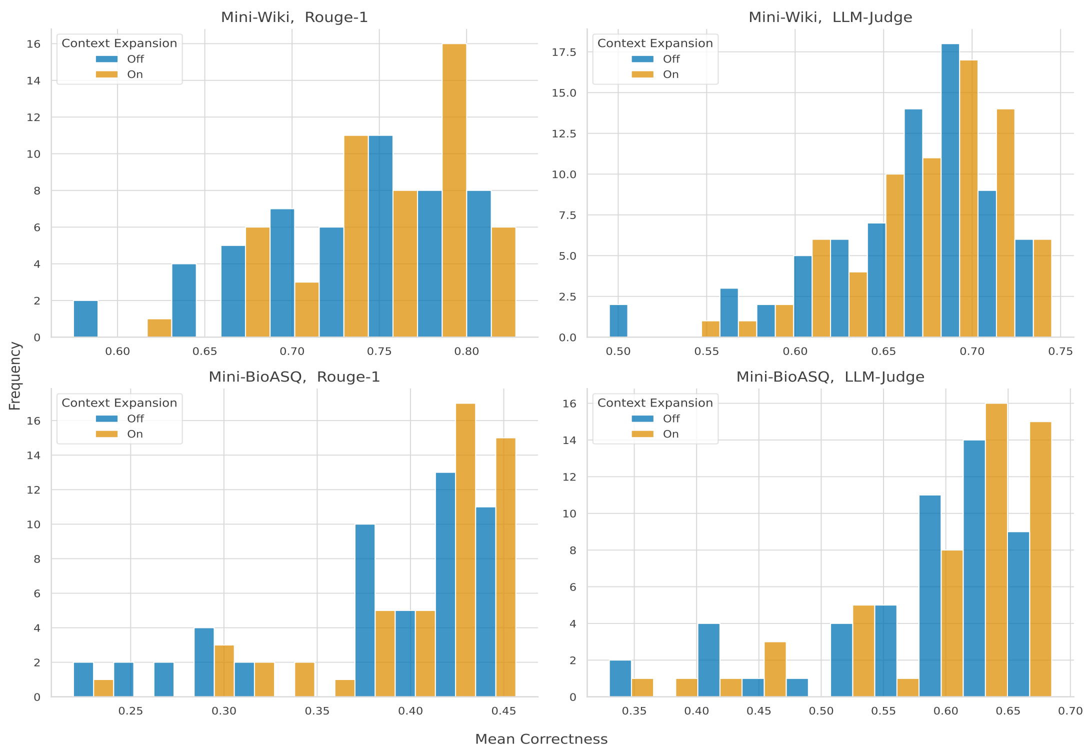


Figure 5.3: Context Expansion - All parameter permutations

5.1.4 Number of Sources

The relationship between the number of sources and answer correctness is illustrated in the following figures. Correctness shows a strong positive correlation with the number of sources used to generate the RAG response. For both the Mini-Wiki and Mini-BioASQ datasets, correctness improves as the number of sources increases. In the Mini-Wiki dataset, this improvement plateaus at approximately three to four sources. In contrast, while the increase in correctness for the Mini-BioASQ dataset is less pronounced with fewer sources, it reaches its peak at around five sources. This difference can be attributed to the Mini-BioASQ dataset's greater reliance on reasoning across multiple sources compared to the Mini-Wiki dataset.

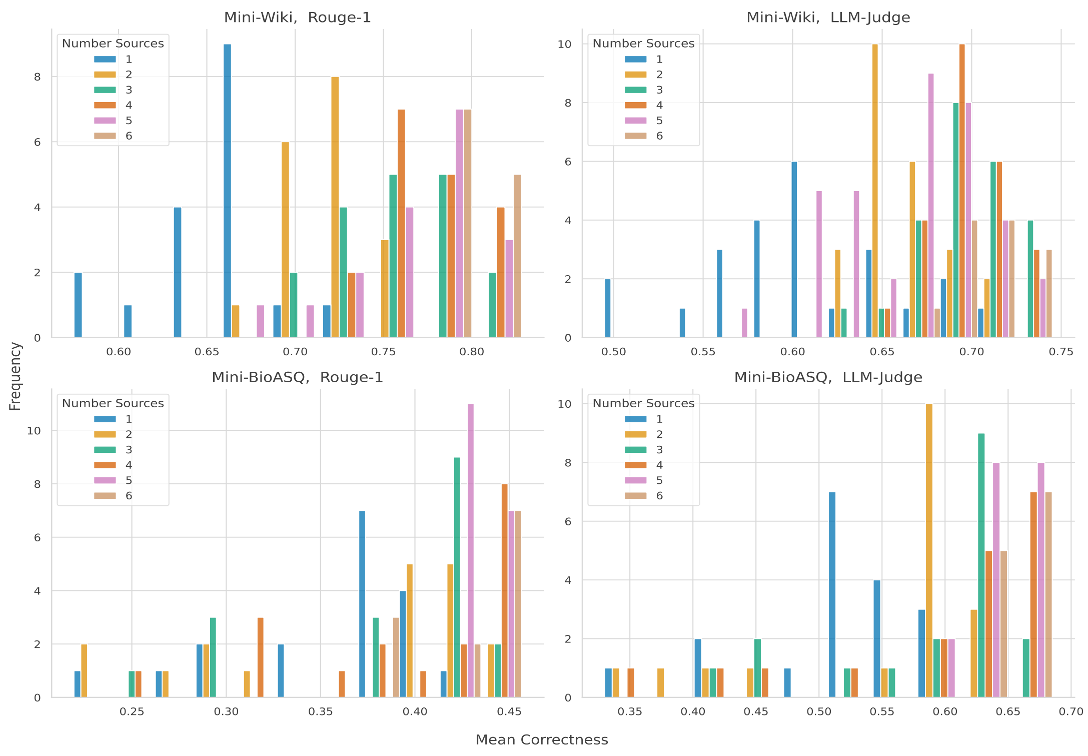
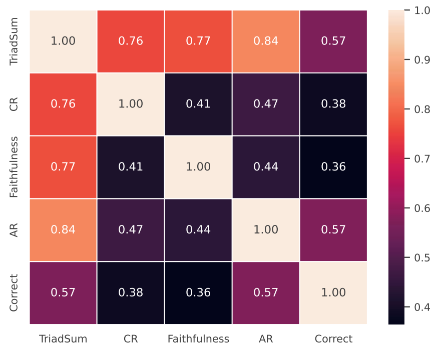


Figure 5.4: Number sources All parameter permutations

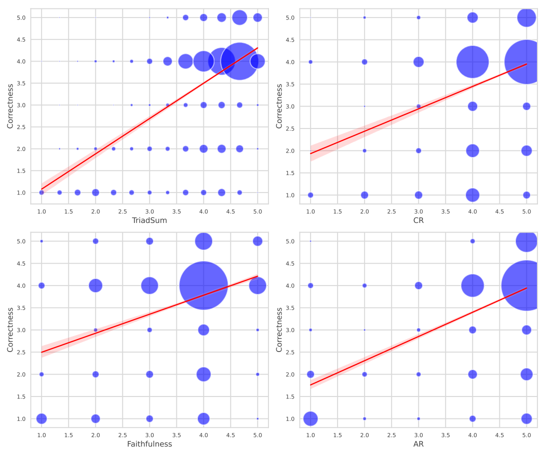
5.2 RAG Triad Correlation

This section addresses the correlation between correctness and the components of the RAG Triad. It aims to provide insights into the system by exploring the relationship between the intermediate outputs of the RAG system, the combined RAG Triad score (Triad Sum), and the overall correctness scores evaluated by an LLM-judge.

The correlation matrix in Figure 5.5a demonstrates a positive correlation between correctness and each of the following relations: context relevance (CR), faithfulness, answer relevance (AR) and the Triad um. Additionally, the scatter plot in Figure 5.5b illustrates the pairwise correlation between context relevance, faithfulness, answer relevance, Triad Sum and correctness. The size of the data points represents their magnitude of occurrence in the evaluated data. The x and y-axis show the LLM judgment score. The regression line for all plots shows a clear positive correlation between the Variables.



(a) Correlation matrix



(b) Scatter plot

Figure 5.5: RAG Triad correlation for Mini-BioASQ and LLM-Judge.

5.3 In depth parameter results

This section explores the results for both datasets using two selected RAG methods, based on the subset results discussed earlier for the downsampled datasets. The primary objective is to identify the RAG methods with the most positive impact on system correctness. The experiments are conducted on the full Mini-BioASQ dataset, which provides 4,012 query-answer-passage triples.

The analysis identifies re-ranking and context expansion as key parameters with the most significant positive impact on system performance. In contrast, query expansion does not contribute to performance gain, and the number of sources reaches a plateau at around 3–5 sources. For this evaluation, re-ranking and context expansion are applied while keeping other RAG system settings fixed as follows:

- Number of sources : 3
- LLM : Mixtral 7x8B
- Query Expansion : Inactive
- Dataset : Mini-BioASQ
- Re-rank: Inactive (when context expansion examined)
- Context Expansion: Inactive (when re-ranking examined)

5.3.1 Re-ranking

The re-ranking mechanism shows an improvement in correctness across both methods (RRF and semantic), with the semantic re-ranking outperforming the RRF method with respect to correctness. While the LLM-judge metric, indicates a modest correctness increase, the ROUGE-1 metric, shown in Figure 5.6 reveals a more significant correctness increase compared to inactive re-ranking.

5.3.2 Context Expansion

The impact of context expansion integration into the RAG pipeline is presented in Figure 5.7, where the LLM-judge metric and the ROUGE-1 metric are applied, respectively. The results are consistent across both metrics, demonstrating a significant increase of correctness for the full Mini-BioASQ dataset.

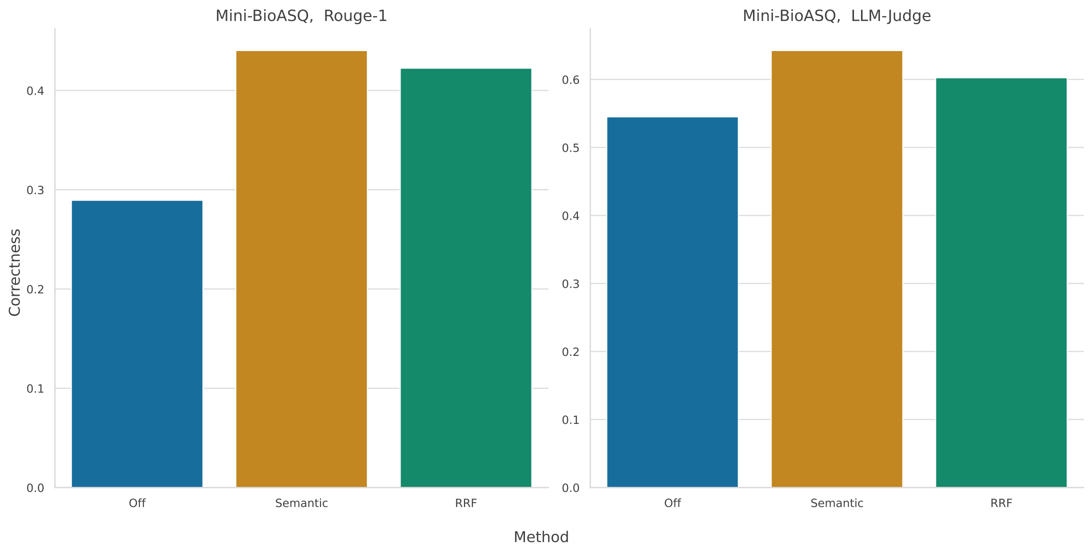


Figure 5.6: Re-ranking correctness

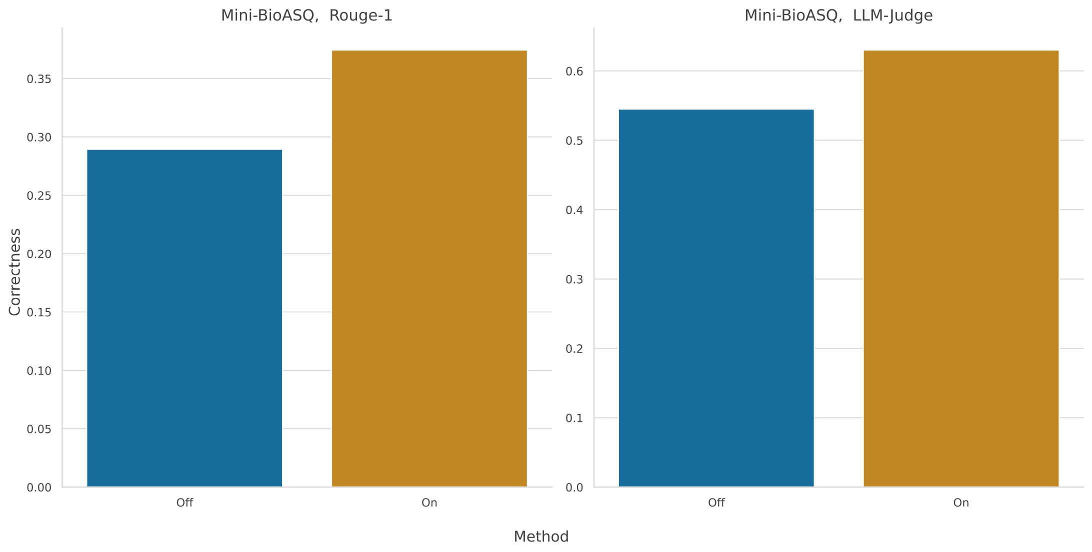


Figure 5.7: Context Expansion correctness

5.3.3 LLM Choice

Four models with distinct specification, strengths, and weaknesses were selected for evaluation. The results, displayed in Figure 5.8 were evaluated by the LLM-judge metric and the ROUGE-1 recall metric respectively, indicate that the Mixtral model achieves the highest correctness scores across both methods. In contrast, the Llama3.1 model yields the lowest performance. The Llama3.2 and Gemma2 alternate their rankings depending on the evaluation metric. Overall, Mixtral consistently outperforms the other models under examination, in part due to its greater number of model parameters compared to the other models.

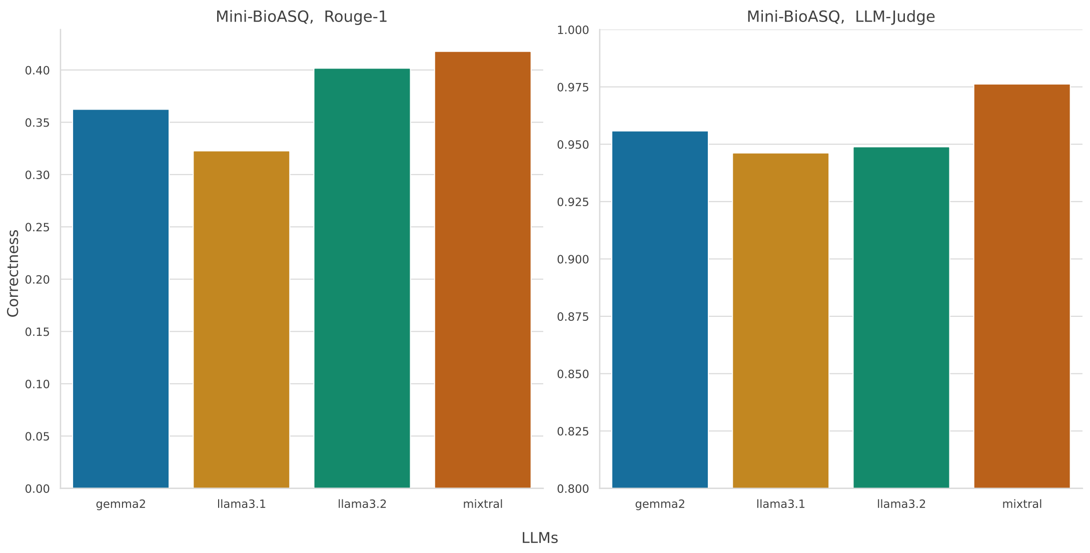


Figure 5.8: Model correctness

5.4 Gold Standard retrieval

This section is focuses on classical retrieval. The impact of re-ranking on the retrieval performance is illustrated in the figures below. Gold standard retrieval sources are only available for the Mini-BioASQ dataset, enabling the use of the recall@k metric evaluating retrieval quality. The plot in Figure 5.9 shows the effect of the re-ranking methods on recall in retrieval performance. Re-ranking results in a higher recall score compared to inactive re-ranking, thereby improving retrieval quality. Among the methods, semantic re-ranking outperforms RRF re-ranking. This is in line with the correctness results for the dataset, where the semantic re-ranking consistently outperforms reciprocal rank fusion re-ranking in similar fashion. The overall recall@5 score settles down at around 0.4, this is due to the number of gold standard passages varying from query to query and often exceeding 5 resources. Additionally, recall metrics on similar datasets in the literature centre around the same rate.

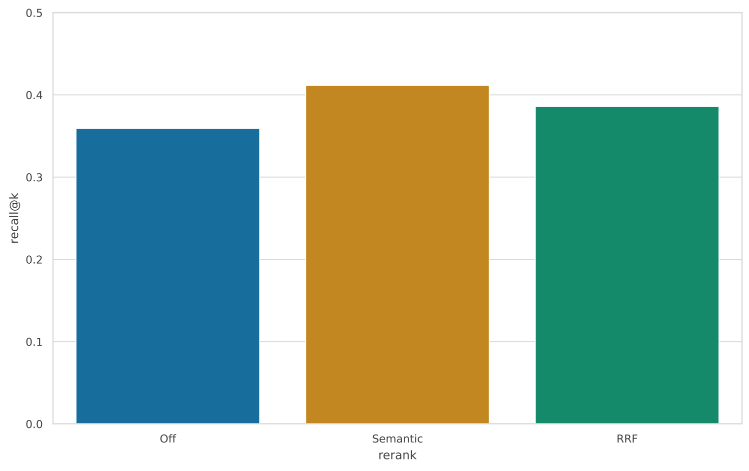


Figure 5.9: Recall@5, for re-rank methods

5.5 Parameter Permutation Results

The tables below summarize the relationship between RAG parameter configurations and corresponding correctness scores. Each table displays the top 10 entries, sorted in ascending order by mean correctness. Full tables are provided in the appendix.

For the Mini-Wiki dataset, the RAG pipeline achieves higher correctness scores when query expansion is disabled. In contrast, for the Mini-BioASQ dataset, enabling query expansion leads to improved performance. Both datasets show better evaluation results when either re-ranking option is applied. Systems with context expansion enabled yield the highest mean correctness across both datasets and scoring methods.

Query Expansion	Re-rank	Context Expansion	Number Sources	Mean Correctness
3	RRF	On	4	0.685000
2	Semantic	Off	6	0.682500
3	Semantic	Off	6	0.680000
2	RRF	On	3	0.675000
2	RRF	On	5	0.672500
2	Semantic	Off	4	0.670000
2	Semantic	Off	5	0.670000
2	Semantic	On	6	0.665000
2	Semantic	On	5	0.665000
3	Semantic	Off	5	0.662500

Table 5.1: Data: Mini-BioASQ, Metric: LLM-judge

Query Expansion	Re-rank	Context Expansion	Number Sources	Mean Correctness
3	Semantic	On	6	0.456490
2	Semantic	Off	6	0.452220
Off	Semantic	Off	4	0.450930
3	Semantic	Off	6	0.448400
2	Semantic	On	4	0.448020
Off	Semantic	Off	5	0.447600
2	Semantic	On	5	0.447550
3	RRF	On	6	0.447140
3	Semantic	On	5	0.444850
Off	Semantic	Off	6	0.444770

Table 5.2: Data: Mini-BioASQ, Metric: ROUGE-1

Query Expansion	Re-rank	Context Expansion	Number Sources	Mean Correctness
2	Semantic	On	5	0.742500
Off	RRF	Off	6	0.740000
2	Semantic	Off	3	0.737500
Off	Semantic	Off	6	0.737500
Off	Semantic	On	4	0.735000
Off	RRF	Off	3	0.732500
Off	RRF	On	4	0.730000
Off	Semantic	On	5	0.730000
Off	Semantic	On	6	0.725000
3	RRF	Off	5	0.720000

Table 5.3: Data: Mini-Wiki, Metric: LLM-judge

Query Expansion	Re-rank	Context Expansion	Number Sources	Mean Correctness
Off	Semantic	On	4	0.828220
Off	Semantic	Off	6	0.825820
Off	RRF	Off	6	0.821770
Off	RRF	Off	3	0.819060
Off	RRF	On	3	0.816540
Off	Semantic	On	6	0.815790
Off	RRF	On	4	0.814130
Off	RRF	On	6	0.812690
2	Semantic	On	5	0.810940
Off	RRF	Off	5	0.810170

Table 5.4: Data: Mini-Wiki, Metric: ROUGE-1

CHAPTER 6

Conclusions

In this section, the results are reviewed, and key takeaways are identified. Notable RAG methods and the capabilities of the evaluation framework are discussed.

6.1 Re-ranking of Sources

Both re-ranking functions outperform the search results where re-ranking is inactive. This highlights the positive impact of high quality search results and indicate that both re-ranking functions perform well when tasked with finding valuable context for a given query. Both re-ranking methods enhance system correctness compared to inactive re-ranking. However, the computational costs of both methods differ significantly. Semantic re-ranking, scales linearly with the number of retrieved resources, as it involves pairwise comparisons between each resource and the query.

The semantic re-ranking process requires embedding generation and similarity scoring, making it computationally expensive. In contrast, reciprocal rank fusion (RRF) adds minimal computation while improving the mean answer correctness, albeit less than semantic re-ranking. The relative performance of the two methods varies depending on the dataset and scoring metric used. The evaluation results confirm that both semantic re-ranking and RRF enhance the system's answer correctness. These findings show the importance of adding re-ranking mechanisms into RAG pipelines to improve retrieval quality and conclusively system answer correctness.

6.2 Query Expansion

The intended benefit of query expansion is to retrieve additional documents by generating queries similar to the original one. However, this process significantly increases the number of retrieved resources. For instance, if two additional queries are formulated, the total number of queries used for the search becomes three, tripling the volume of retrieved documents. The resource pool grows rapidly, increasing the complexity of downstream processing, leading to increased room for selecting the wrong resources while adding computational cost.

The results indicate that query expansion does not achieve improvements in retrieval quality. This can be attributed to the additional noise introduced in the expanded pool of retrieved documents, when selecting which resources, to pass to the LLM for answer generation. In many cases, no additional useful documents are retrieved, while in others, the re-ranker struggles to handle the enlarged pool of resources. This can lead to subpar document ranking. Overall, for the evaluated datasets and domains, query expansion fails to enhance retrieval.

6.3 Context Expansion

The benefits of context expansion are visible across both datasets. Expanding the context to include preceding and succeeding text, provides additional information (which is most probably related), enabling the system to interpret the retrieved resource more accurately within the broader scope of the original document. Even if the surrounding text added during context expansion does not directly contain the answer to the query, it can help the LLM parse the context more accurately. Furthermore, the pre- and post-context possibly hold crucial information themselves, needed to answer the question. A significant advantage of context expansion is its low computational cost. The additional context (pre- and post-text) is added during the indexing phase of the RAG pipeline. Once the documents are indexed, this expanded context is automatically delivered alongside the resource hit via semantic or lexical search. The computational overhead for context expansion is outsourced to the vector index creation. Thus, it's precomputed, making it an efficient enhancement to the retrieval process.

6.4 Number of Sources

The number of sources used in retrieval plays a relevant role in system performance, with clear trade-offs in the number of sources passed on to the LLM for answer generation. On one hand, retrieving too few sources may result in insufficient information to fully address complex queries. A low number of sources may lead to incomplete answers, especially for complex questions that require information from multiple documents. On the other hand, too many sources can hold redundant information and thus introduce additional noise. Additionally, conflicting information across multiple documents can be problematic, in the sense that the downstream LLM is left with a decision on which information to trust. This is a task the LLM is not designed to perform, as it is instructed by the RAG system design to avoid subjective judgment. Such cases might lead to system answers of lower quality. The evaluation results underline the importance of balancing the number of sources to ensure sufficient information while also limiting noise and conflicting information. The outcome of the studies in this work indicate an optimum somewhere between 3–5 sources.

6.5 LLM Choice

The selection of a large language model (LLM) significantly impacts the performance of the RAG system, as it serves as the bridge between context retrieval and final answer generation. It directly touches all components of the RAG pipeline — retrieval, augmentation, and generation — thus making it an integral part of the RAG system as a whole. Among the evaluated models, Mixtral7x8 is superior for the examined domains and datasets. This is likely due to its advanced capabilities in retrieval processing, question answering, context understanding and efficient mixture of experts architecture. It also features a greater number of model weights. Note, that no single LLM can be universally recommended for all tasks and domains. The performance of a specific model depends heavily on the specific problem requirements — dataset domain, problem size, data quality and user needs. In conclusion, the active research fields of computational natural language processing, data science and AI, continuously introduces new models and is evolving rapidly. The dynamic field of language models thus makes it necessary to perform extensive testing of specific problems and making design choices based on evaluation results.

6.6 RAG Triad

The hypothesis of the RAG Triad states that if the intermediate results among the subcomponents of a RAG system — query, resources, and answer — are sufficiently correct, then the RAG system yields sufficiently correct answers.

The presented results indicate a strong positive correlation between RAG answer correctness and the RAG Triad sum. Specifically, a high RAG Triad score for a given query is often associated with a correct answer, while a lower RAG Triad score suggests an incorrect answer. This observation holds significant potential for in real-world applications, since most scenarios do not involve datasets holding ground-truth answers, much less ground truth retrieval passages. Instead, often a document corpus to be indexed and a set of typical queries to the system is are provided. The results presented in this thesis indicate that the RAG Triad can serve as a useful evaluation metric for ground-truth less datasets. However, further testing across a broader range of domains and varying datasets is necessary to show the reliability of the approach to strengthen the case of the RAG Triad hypothesis. Nevertheless, it provides a simple way of computing the performance of a RAG system in the absence of any ground truth. The RAG Triad additionally proved scores fur subcomponents of the system. These can be instrumental in identifying issues when developing a complete RAG system, such as implementation errors, suboptimal parameter settings, incomplete domain corpora, or problems within the retrieval system. In conclusion, the RAG Triad can be a valuable tool for diagnosing problems within the RAG system.

6.7 Evaluation Framework

The proposed evaluation framework delivers promising results, proposing areas of improvement within the RAG process. It is possible to process datasets across multiple domains and is capable of handling datasets with and without ground truths answers or documents. The evaluation system can measure the overall RAG system answer correctness, while also providing results on individual subcomponents of the system via the RAG Triad.

6.8 Lessons learned

The optimal approach involves providing a small number of high quality resources (3-5) to the LLM along with the query. The information content of resources throughout the RAG process is crucial. Re-ranking is pivotal in selecting the resources of highest quality.Context expansion is an effective and low-cost strategy to enhance retrieved resources. The RAG Triad offers an alternative to answer correctness computation, which is of great importance for real-world applications. ROUGE recall produces results comparable to the LLM-judge metric, while being a cost-effective alternative for evaluation. Mean correctness scores differ significantly between the two datasets. The Mini-Wiki dataset yields higher correctness scores on average compared to the Mini-BioASQ dataset. This discrepancy can be attributed to the complexity of the domains relative to one another. The Mini-Wiki dataset features a more straightforward corpus and simpler questions, often requiring fewer sources to generate correct answers. In contrast, the Mini-BioASQ dataset draws question and answers from a complex biomedical domain, including specific terminology and queries that often require contextual information from multiple sources (3+). The Mixtral model outperforms Llama3.1, Llama3. and Gemma2 for the examined datasets. The developed evaluation framework offers multiple evaluation metrics alongside two scoring metrics to quantify system performance.

CHAPTER 7

Outlook

This work presents valuable insights into RAG systems and their evaluation. However, due to limitations in computational resources, it has several limitations.

First, the designed framework is restricted to text-based question answering tasks. The system is not concerned with multi-modal data beyond text, nor does it integrate broader, more adaptable systems, such as graph or agent-based approaches. Second, the experiments were conducted with a specific set of datasets and large language models. Further, limitations are the use of a single vector database and a single embedding model. While these choices present valuable insight, they represent only a small subset of the available resources in the field. Thus, the analysis offers recommendations and identifies trends in RAG design but does not provide a comprehensive guide employable to all scenarios. Third, only a limited number of parameters and methods within the RAG system were examined within the parameter space. Given the complexity of RAG systems — with complex components, multiple design choices and parameter settings — there remains potential to further investigate and optimise. Fourth, the evaluation framework relies on two semantic scoring metrics, LLM-judges and ROUGE recall, each with their own limitations and biases. While these metrics provide meaningful results, other evaluation approaches may capture different nuances and interpretations of the evaluated texts. Additionally, the RAG triad — as a ground-truth-free evaluation metric — shows promising results but requires additional testing across diverse datasets, domains, and languages to establish generalisability. Finally, this work lays the foundation for research in this field. Future areas of exploration include the evaluation of larger datasets, additional LLMs and embedding models while adding more resource intensive evaluation metrics, like human judges.

Concluding, the proposed evaluation framework, together with the analysis on RAG system design and parameter choices, offers a practical tool for addressing RAG systems. It offers a ready-to-use framework and recommendations for a RAG system configuration, providing a reduction in hallucination for resource based answer generation. However, this work does not and cannot claim to define the “perfect” RAG setup or provide a universally applicable evaluation framework. Future studies have to be conducted to explore alternative evaluation methods, delve into diverse domains and examine a wider range of possible RAG tasks.

Appendix I: Results

Significance Results

The ANOVA analysis results are presented, for both datasets and both scoring methods. The results are from drawn from a grid search across the whole examined parameter space with 100 elements subsample per dataset.

Method	p-value	f-value
Number of Sources	0.000002	8.118252
Query Expansion	0.123111	2.139620
Re-rank	0.00000002	26.039115
Context Expansion	0.246011	1.361739

Table 1: ANOVA, Dataset: Mini-BioASQ, Scoring: LLM-Judge

Method	p-value	f-value
Number of Sources	0.000534	4.850804
Query Expansion	0.831789	0.184520
Re-rank	0.0000002	62.508190
Context Expansion	0.067195	3.424304

Table 2: ANOVA, Dataset: Mini-BioASQ, Scoring: ROUGE-1

Method	p-value	f-value
Number of Sources	0.0000002	50.105502
Query Expansion	0.021014	4.017242
Re-rank	0.011682	4.655856
Context Expansion	0.144764	2.160220

Table 3: ANOVA, Dataset: Mini-Wiki, Scoring: LLM-Judge

Method	p-value	f-value
Number of Sources	0.000000	24.313682
Query Expansion	0.000060	10.424392
Re-rerank	0.001304	6.964930
Context Expansion	0.386710	0.753897

Table 4: ANOVA, Dataset: Mini-Wiki, Scoring: ROUGE-1

LLM-Judge Evaluation prompts

The instructive prompts for the LLM-Judges are given below. The prompts guide the LLMs to judge with respect to the requested evaluation metric - context relevance, faithfulness, answer relevance or correctness.

Correctness evaluation prompt

Given the following answer and ground-truth, give a rating from 1 to 5.
Respond with 1 if the answer is not correct based on the ground-truth at all.
Respond with 2 if the answer is slightly correct based on the ground-truth.
Respond with 3 if the answer is moderately correct based on the ground-truth.
Respond with 4 if the answer is mostly correct based on the ground-truth.
Respond with 5 if the answer is completely correct based on the ground-truth.
Your response must strictly and only be a single integer from "1" to "5" and no additional text.

Adhere to the examples:
If the answer is "yes, the shirt is dark blue" and the ground-truth is "yes", your response should be "5".
If the answer is "yes" and the ground-truth is "yes", your response should be "5".
If the answer is "The sky is clear and blueish" and the ground-truth is "The sky is blue", your response should be "4".
If the answer is "The sky is blueish" and the ground-truth is "The sky is blue", your response should be "4".
If the answer is "The sky is somewhat blue" and the ground-truth is "The sky is blue", your response should be "3".
If the answer is "The sky is blue with some clouds" and the ground-truth is "The sky is almost clear", your response should be "3".
If the answer is "The sky is cloudy" and the ground-truth is "The sky is blue", your response should be "2".
If the answer is "The sky is blue" and the ground-truth is "The sky is clear", your response should be "2".
If none of the nouns in the answer are present in the ground-truth, the answer is not correct. Thus, your response should be "1".
If the answer is "yes" and the ground-truth is "no", your response should be "1".
If the answer is "no" and the ground-truth is "yes", your response should be "1".
If the answer is "there is no mention of ... in the given context" and the ground-truth is "yes" or "no", your response should be "1".
If the answer is "I do not know", your response should be "1".

Here are the Answer: "{answer}" and the ground-truth: "{ground-truth}".

Context relevance evaluation prompt

Given the following context and query, give a rating from 1 to 5.
Respond with 1 if the context is not relevant to the query at all.
Respond with 2 if the context is slightly relevant to the query.
Respond with 3 if the context is moderately relevant to the query.
Respond with 4 if the context is mostly relevant to the query.
Respond with 5 if the context is completely relevant to the query.
Your response must strictly and only be a single integer from "1" to "5" and no additional text.

Adhere to the examples:
If the context is "The pandemic, spanning the whole globe started in 2019." and the query is "What year did the corona pandemic start?", your response should be "5".
If the context is "The sky is blue" and the query is "What color is the sky?", your response should be "5".
If the context is "The sky is blue" and the query is "What is the weather like?", your response should be "4".
If the context is "Water boils at 100 degrees Celsius" and the query is "What happens to water at high temperatures?", your response should be "4".
If the context is "The sky is blue" and the query is "What color is the sky usually?", your response should be "3".
If the context is "Water boils at 100 degrees Celsius" and the query is "At what temperature does water usually boil?", your response should be "3".
If the context is "The sky is blue" and the query is "What color is the ocean?", your response should be "2".
If the context is "Water boils at 100 degrees Celsius" and the query is "What is the boiling point of water in Fahrenheit?", your response should be "2".
If none of the nouns in the query are present in the context, the context is not relevant and your response should be "1".
If the context is "The sky is blue" and the query is "What color is the grass?", your response should be "1".
If the context is "The pandemic, was a global event and lead to many deaths." and the query is "What year did the corona pandemic start?", your response should be "1".

Here are the Context: "{context}" and the Query: "{query}".

Faithfulness evaluation prompt

Given the following context and answer, give a rating from 1 to 5.
Respond with 1 if the answer is not sufficiently grounded in the context at all.
Respond with 2 if the answer is slightly grounded in the context.
Respond with 3 if the answer is moderately grounded in the context.
Respond with 4 if the answer is mostly grounded in the context.
Respond with 5 if the answer is completely grounded in the context.
Your response must strictly and only be a single integer from "1" to "5" and no additional text.

Adhere to the examples:
If the context is "The sky is blue" and the answer is "The sky is blue", your response should be "5".
If the context is "Water boils at 100 degrees Celsius" and the answer is "Water boils at 100 degrees Celsius", your response should be "5".
If the context is "Water boils at 100 degrees Celsius" and the answer is "Water boils at around 100 degrees Celsius", your response should be "4".
If the context is "The pandemic, spanning the whole globe started in 2019." and the answer is "The pandemic started in late 2019.", your response should be "4".
If the context is "Water boils at 100 degrees Celsius" and the answer is "Water boils at a high temperature", your response should be "3".
If the context is "The sky is blue" and the answer is "The sky is somewhat blue", your response should be "3".
If the context is "The pandemic, was a global event and lead to many deaths." and the answer is "The pandemic was a significant global event.", your response should be "2".
If the context is "The pandemic, was a global event and lead to many deaths." and the answer is "The pandemic caused many deaths.", your response should be "2".
If the context is "Water boils at 100 degrees Celsius" and the answer is "Water freezes at 0 degrees Celsius", your response should be "1".
If the context is "The pandemic, was a global event and lead to many deaths." and the answer is "The pandemic started in 2019.", your response should be "1".
If none of the nouns in the answer are present in the context, the answer is not grounded and your response should be "1".
If the answer is "I do not know" or "there is no mention of ... in the given context", the answer is not grounded and your response should be "1".

Here are the Context: "{context}" and the Answer: "{answer}".

Answer relevance evaluation prompt

Given the following query and answer, give a rating from 1 to 5.
Respond with 1 if the answer is not relevant to the query at all.
Respond with 2 if the answer is slightly relevant to the query.
Respond with 3 if the answer is moderately relevant to the query.
Respond with 4 if the answer is mostly relevant to the query.
Respond with 5 if the answer is completely relevant to the query.
Your response must strictly and only be a single integer from "1" to "5" and no additional text.

Adhere to the examples:
If the query is "What color is the sky?" and the answer is "The sky is blue", your response should be "5".
If the query is "At what temperature does water boil?" and the answer is "Water boils at 100 degrees Celsius", your response should be "5".
If the query is "What is the capital of Germany?" and the answer is "Berlin is a major city in Germany.", your response should be "4".
If the query is "What is the capital of Germany?" and the answer is "Berlin is a city in Germany.", your response should be "4".
If the query is "What is the capital of Germany?" and the answer is "Berlin is a large city.", your response should be "3".
If the query is "How many moons does the Earth have?" and the answer is "The Earth has at least one moon", your response should be "3".
If the query is "What is the capital of Germany?" and the answer is "Berlin is known for its history.", your response should be "2".
If the query is "At what temperature does water boil?" and the answer is "Water boils at its boiling point.", your response should be "2".
If the query is "What year did the corona pandemic start?" and the answer is "The pandemic, was a global event and lead to many deaths.", your response should be "1".
If the query is "What is the capital of Germany?" and the answer is "Germany is a country in Europe.", your response should be "1".
If none of the nouns in the answer are present in the query, the answer is not relevant, your response should be "1".
If the answer is "I do not know" or "there is no mention of ... in the given context", your response should be "1".

Here are the Query: " {query}" and the Answer: "{answer}".

Appendix II: Software

The foundational building block of this work is natural language software. The integral parts are presented in this section.

LLMs

This subsection provides a comprehensive overview of the selected large language models (LLMs), including their specifications and notable features. All models are open-source and free for personal use. A summary of the examined model and their properties is provided.

Llama3.1

- **Parameter Size:** 8.0B
- **Quantization Level:** Q4_0
- **Model:** Standard decoder-only transformer
- **Pre-training:** Trained on web data composed of 50% general knowledge, 25% mathematics/reasoning, 17% code, and 8% multilingual tokens
- **Context Window:** 130k tokens
- **Developer:** Meta
- **Info:** A lightweight model optimized for low latency [Met24a] .

Llama3.2

- **Parameter Size:** 3.2B
- **Quantization Level:** Q4_K_M
- **Model:** Standard decoder-only transformer

- **Pre-training:** Built on Llama3.1 and enhanced with multimodal training data (e.g., image-text tuples)
- **Context Window:** 128k tokens
- **Developer:** Meta
- **Info:** A super lightweight model with very low latency [Met24b] .

Mixtral

- **Parameter Size:** 47B
- **Quantization Level:** Q4_0
- **Model:** Sparse mixture-of-experts decoder-only model
- **Pre-training:** Trained on open web data (specifics unknown)
- **Context Window:** 32k tokens
- **Developer:** Mistral AI
- **Info:** This lightweight model uses a sparse mixture-of-experts approach. During generation, the feedforward blocks dynamically select from 8 groups of parameters. A router network chooses two "experts" per token at each layer, reducing latency and computational cost by utilizing only a fraction of the total parameters for each token [AI23].

Gemma2

- **Parameter Size:** 9.2B
- **Quantization Level:** Q4_0
- **Model:** Text-to-text decoder-only model
- **Pre-training:** Trained on 8 billion tokens, including web data, code, and mathematics
- **Context Window:** 8k tokens
- **Developer:** Google
- **Info:** Designed for tasks such as question answering, summarization, and reasoning [AI].

Vector index

The vector index software employed in this work is the open source framework marqo [Mar]. Marqo serves as a wrapper (management system) for the underlying vector database structure built on Vespa [Ves].

Two vector indexes were initialised, one for the Mini-BiosQA dataset and another one for the Mini-Wiki dataset. At its core it uses the sentence transformer "flax-sentence-embeddings/all_datasets_v4_mpnet-base" [Hug], which is based on the pre-trained MPnet-base model [Mic]. This pre-trained model is then fine-tuned using a diverse collection of multiple datasets, mostly from question answer domains, involving around 1 billion sentence pairs. The vector dimension of the embedded documents, i.e. the models output, is consistently 768. The hierarchical navigable small world (HNSW) graph, serving as the data structure of the index, is configured with the following parameter specifications:

- Ef - Construction: 512. This is the length of the dynamic list of the nearest neighbors during construction. A longer list produces a more accurate index, but consumes more time and memory.
- M: 16. This parameter controls the maximum static number of nearest neighbor for each node entry in the graph. Similar to ef-construction, a larger number leads to an increased memory footprint.
- Ef-Search: 2000. This is a search parameter relevant at retrieval time and can be set individually for each query. It controls the size of a dynamic list of nearest neighbors kept during the search process. In the approximate nearest neighbor algorithm, a larger number indicates a more precise albeit a more resource intensive search.

Versions

- Marqo: 2.10
- Ollama (LLM manager): 0.4.1

Code

Repository: https://github.com/simon-koenig/RAG

Appendix III: Hardware

The vector index, semantic re-ranker and the LLMs were installed on the following GPU architectures.

Vector index and semantic re-ranker:

- Name: NVIDIA A16
- GPUs: 4 GPUs per card: Each GPU is independent and isolated.
- Memory: 16 GB GDDR6 per GPU: Total of 64 GB across the card.
- CUDA Cores per GPU: 1,792 (7,168 total across 4 GPUs).
- Vector Index: 14 of 16 GB memory
- Semantic Re-ranker: 2 of 16 GB memory

LLMs:

- Name: Nvidia A40
- GPUs: Single GPU
- Memory: 48 GB GDDR6
- CUDA Cores per GPU: 10,752
- Multiple LLMs loaded and infered with by multiple clients simultaneously

List of Figures

List of Tables

Bibliography

[AI] Google AI. Gemma-Modelle von Google AI | Google für Entwickler.

[AI23] Mistral AI. Mixtral of experts, December 2023.

[CCB09] Gordon V. Cormack, Charles L A Clarke, and Stefan Buettcher. Reciprocal rank fusion outperforms condorcet and individual rank learning methods. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 758–759, Boston MA USA, July 2009. ACM.

[CPL19] Qingyu Chen, Yifan Peng, and Zhiyong Lu. BioSentVec: creating sentence embeddings for biomedical texts. In *2019 IEEE International Conference on Healthcare Informatics (ICHI)*, pages 1–5, Xi'an, China, June 2019. IEEE.

[CZLZ24] Banghao Chen, Zhaofeng Zhang, Nicolas Langrené, and Shengxin Zhu. Unleashing the potential of prompt engineering in Large Language Models: a comprehensive review, September 2024. arXiv:2310.14735 [cs].

[Dah24] Younes Dahami. Understanding How ChatGPT Uses the Decoder-Only Transformer Architecture, June 2024.

[EJEAS23] Shahul Es, Jithin James, Luis Espinosa-Anke, and Steven Schockaert. RAGAS: Automated Evaluation of Retrieval Augmented Generation, September 2023. arXiv:2309.15217 [cs].

[ENFO24] Matouš Eibich, Shivay Nagpal, and Alexander Fred-Ojala. ARAGOG: Advanced RAG Output Grading, April 2024. arXiv:2404.01037 [cs].

[FBS24] Robert Friel, Masha Belyi, and Atindriyo Sanyal. RAGBench: Explainable Benchmark for Retrieval-Augmented Generation Systems, June 2024. arXiv:2407.11005 [cs].

[GRC$^+$22] Michael Glass, Gaetano Rossiello, Md Faisal Mahbub Chowdhury, Ankita Rajaram Naik, Pengshan Cai, and Alfio Gliozzo. Re2G: Retrieve, Rerank, Generate, July 2022. arXiv:2207.06300 [cs].

[GXG+24] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. Retrieval-Augmented Generation for Large Language Models: A Survey, March 2024. arXiv:2312.10997 [cs].

[HLW23] Yikun Han, Chunjiang Liu, and Pengfei Wang. A Comprehensive Survey on Vector Database: Storage and Retrieval Technique, Challenge, October 2023. arXiv:2310.11703 [cs].

[HMN+23] Yasuto Hoshi, Daisuke Miyashita, Youyang Ng, Kento Tatsuno, Yasuhiro Morioka, Osamu Torii, and Jun Deguchi. RaLLe: A Framework for Developing and Evaluating Retrieval-Augmented Large Language Models, October 2023. arXiv:2308.10633 [cs].

[HSWN24] Jennifer Hsia, Afreen Shaikh, Zhiruo Wang, and Graham Neubig. RAGGED: Towards Informed Design of Retrieval Augmented Generation Systems, August 2024. arXiv:2403.09040 [cs].

[Hua10] Xiangji Jimmy Huang, editor. *CIKM'10: proceedings of the 19th International Conference on Information & Knowledge Management and Co-Located Workshops; Oktober 26 - 30, 2010, Toronto, Ontario, Canada*. ACM, New York, NY, 2010.

[Hug] Huggingface. flax-sentence-embeddings/all_datasets_v4_mpnet-base · Hugging Face.

[Hug24] Huggingface. RAG Datasets, June 2024.

[JDS11] H Jégou, M Douze, and C Schmid. Product Quantization for Nearest Neighbor Search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(1):117–128, January 2011.

[Jur24] Jurafsky. Speech and Language Processing, August 2024.

[KKB+23] Sanjay Kukreja, Tarun Kumar, Vishal Bharate, Amit Purohit, Abhijit Dasgupta, and Debashis Guha. Vector Databases and Vector Embeddings-Review. In *2023 International Workshop on Artificial Intelligence and Image Processing (IWAIIP)*, pages 231–236, Yogyakarta, Indonesia, December 2023. IEEE.

[KNBP23] Anastasia Krithara, Anastasios Nentidis, Konstantinos Bougiatiotis, and Georgios Paliouras. BioASQ-QA: A manually curated corpus for Biomedical Question Answering. *Scientific Data*, 10(1):170, March 2023.

[KPR+19] Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei

Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. Natural Questions: A Benchmark for Question Answering Research. *Transactions of the Association for Computational Linguistics*, 7:453–466, November 2019.

[LHE22] Stephanie Lin, Jacob Hilton, and Owain Evans. TruthfulQA: Measuring How Models Mimic Human Falsehoods, May 2022. arXiv:2109.07958 [cs].

[Lin04a] Chin-Yew Lin. Looking for a Few Good Metrics: ROUGE and its Evaluation. 2004.

[Lin04b] Chin-Yew Lin. ROUGE: A Package for Automatic Evaluation of Summaries. July 2004.

[Mad] Lofred Madzou. What is the RAG Triad?

[Mar] Marqo. Marqo | Train and Deploy Embedding Models.

[MCCD13] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient Estimation of Word Representations in Vector Space, September 2013. arXiv:1301.3781 [cs].

[Met24a] Meta. llama3.1, July 2024.

[Met24b] Meta. llama3.2, September 2024.

[MGH$^+$23] Xinbei Ma, Yeyun Gong, Pengcheng He, Hai Zhao, and Nan Duan. Query Rewriting for Retrieval-Augmented Large Language Models, October 2023. arXiv:2305.14283 [cs].

[Mic] Microsoft. microsoft/mpnet-base · Hugging Face.

[MTMR23] Niklas Muennighoff, Nouamane Tazi, Loïc Magne, and Nils Reimers. MTEB: Massive Text Embedding Benchmark, March 2023. arXiv:2210.07316 [cs].

[MY18] Yu A. Malkov and D. A. Yashunin. Efficient and robust approximate nearest neighbor search using Hierarchical Navigable Small World graphs, August 2018. arXiv:1603.09320 [cs].

[NC20] Rodrigo Nogueira and Kyunghyun Cho. Passage Re-ranking with BERT, April 2020. arXiv:1901.04085 [cs].

[NKQ$^+$24] Humza Naveed, Asad Ullah Khan, Shi Qiu, Muhammad Saqib, Saeed Anwar, Muhammad Usman, Naveed Akhtar, Nick Barnes, and Ajmal Mian. A Comprehensive Overview of Large Language Models, October 2024. arXiv:2307.06435 [cs].

[Pin] Pinecone. Hierarchical Navigable Small Worlds (HNSW).

[PPF+21] Fabio Petroni, Aleksandra Piktus, Angela Fan, Patrick Lewis, Majid Yazdani, Nicola De Cao, James Thorne, Yacine Jernite, Vladimir Karpukhin, Jean Maillard, Vassilis Plachouras, Tim Rocktäschel, and Sebastian Riedel. KILT: a Benchmark for Knowledge Intensive Language Tasks, May 2021. arXiv:2009.02252 [cs].

[PWL23] James Jie Pan, Jianguo Wang, and Guoliang Li. Survey of Vector Database Management Systems, October 2023. arXiv:2310.14021 [cs].

[Ray23] Partha Pratim Ray. ChatGPT: A comprehensive review on background, applications, key challenges, bias, ethics, limitations and future scope. *Internet of Things and Cyber-Physical Systems*, 3:121–154, 2023.

[RDC+24] David Rau, Hervé Déjean, Nadezhda Chirkova, Thibault Formal, Shuai Wang, Vassilina Nikoulina, and Stéphane Clinchant. BERGEN: A Benchmarking Library for Retrieval-Augmented Generation, July 2024. arXiv:2407.01102 [cs].

[RZ09] Stephen Robertson and Hugo Zaragoza. The Probabilistic Relevance Framework: BM25 and Beyond. *Foundations and Trends® in Information Retrieval*, 3(4):333–389, 2009.

[SFKPZ24] Jon Saad-Falcon, Omar Khattab, Christopher Potts, and Matei Zaharia. ARES: An Automated Evaluation Framework for Retrieval-Augmented Generation Systems, March 2024. arXiv:2311.09476 [cs].

[SHH08] Noah Smith, Michael Heilman, and Rebecca Hwa. Question Generation as a Competitive Undergraduate Course Project, 2008.

[SMS24] Kunal Sawarkar, Abhilasha Mangal, and Shivam Raj Solanki. Blended RAG: Improving RAG (Retriever-Augmented Generation) Accuracy with Semantic Search and Hybrid Query-Based Retrievers, August 2024. arXiv:2404.07220 [cs].

[SRBR23] Bianca Steffes, Piotr Rataj, Luise Burger, and Lukas Roth. On evaluating legal summaries with ROUGE. In *Proceedings of the Nineteenth International Conference on Artificial Intelligence and Law*, pages 457–461, Braga Portugal, June 2023. ACM.

[sub] subramanian. vector-database-benchmark/README.pdf at main · sueszli/vector-database-benchmark.

[Tai24] Toni Taipalus. Vector database management systems: Fundamental concepts, use-cases, and current challenges. *Cognitive Systems Research*, 85:101216, June 2024. arXiv:2309.11322 [cs].

[TDVS19] Kashvi Taunk, Sanjukta De, Srishti Verma, and Aleena Swetapadma. A Brief Review of Nearest Neighbor Algorithm for Learning and Classification. In *2019 International Conference on Intelligent Computing and Control Systems (ICCS)*, pages 1255–1260, Madurai, India, May 2019. IEEE.

[Tou] Dave Touretzky. Word Embedding Demo: Tutorial.

[Ves] Vespa. Vespa.ai.

[VSP$^+$23] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention Is All You Need, August 2023. arXiv:1706.03762 [cs].

[YCX20] Jiacheng Yang, Bin Chen, and Shu-Tao Xia. Mean-removed product quantization for large-scale image retrieval. *Neurocomputing*, 406:77–88, September 2020.

[YQZ$^+$18] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A Dataset for Diverse, Explainable Multi-hop Question Answering, September 2018. arXiv:1809.09600 [cs].

[YYM$^+$24] Antonio Jimeno Yepes, Yao You, Jan Milczek, Sebastian Laverde, and Renyu Li. Financial Report Chunking for Effective Retrieval Augmented Generation, March 2024. arXiv:2402.05131 [cs].

[ZCS$^+$23] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena, December 2023. arXiv:2306.05685 [cs].

[ZDD$^+$22] Hamed Zamani, Fernando Diaz, Mostafa Dehghani, Donald Metzler, and Michael Bendersky. Retrieval-Enhanced Machine Learning. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2875–2886, Madrid Spain, July 2022. ACM.

[ZZW$^+$20] Bolong Zheng, Xi Zhao, Lianggui Weng, Nguyen Quoc Viet Hung, Hang Liu, and Christian S. Jensen. PM-LSH: A fast and accurate LSH framework for high-dimensional approximate NN search. *Proceedings of the VLDB Endowment*, 13(5):643–655, January 2020.

[ZZY$^+$24] Penghao Zhao, Hailin Zhang, Qinhan Yu, Zhengren Wang, Yunteng Geng, Fangcheng Fu, Ling Yang, Wentao Zhang, Jie Jiang, and Bin Cui. Retrieval-Augmented Generation for AI-Generated Content: A Survey, April 2024. arXiv:2402.19473 [cs].