



Hochschule für Technik
und Wirtschaft Berlin

University of Applied Sciences

Hochschule für Technik und Wirtschaft Berlin

Fachbereich 4
Professional IT Business and Digitalization

Master Thesis
Summer 2024

Information Retrieval from Large Collections of Local Documents
Using Large Language Models: RAG Approach

Author
Omkar Dilip Gavali (s0585840)

1st Supervisor: Prof. Dr.-Ing. Thomas Schwotzer

2nd Supervisor: Dr. Cristian Grozea

Date: Berlin, 10 September 2024

Contents

1	Introduction	7
1.1	Structure of the Thesis	7
1.2	Background and Motivation	8
1.3	Research Questions	10
1.3.1	Benefits	10
1.4	Scope of the Thesis	10
2	State of the Art	12
2.1	Related Work- LLMs and 3GPP Specs	12
2.2	Retrieval Augmented Generation (RAG)	12
2.2.1	Pre-retrieval	14
2.2.2	Indexing and Embedding	16
2.2.3	Vector Database	18
2.2.4	Retriever	18
2.2.5	Post Node Processing	19
2.2.6	Generation	21
2.3	Applications of LLMs	21
2.4	LlamaIndex	22
3	Design and Implementation	24
3.1	Design	24
3.2	Data Collection	24
3.3	Data Processing	26
3.3.1	Chunking	28
3.3.2	Embedding	29
3.3.3	Vector Database	29
3.3.4	LLMs	30
3.4	User Interface	30
3.4.1	Key Features	32
4	Evaluation	36

5 Conclusion	43
5.1 Limitations	44
5.2 Future Work	44

List of Figures

1 RAG architecture	14
2 Visual representation of different types of chunking	16
3 NN Router	17
4 Vector database architecture adapted from [Xie+23]	19
5 RAG architecture implemented	25
6 3GPP specifications on ETSI website in PDF format	26
7 3GPP specifications on 3GPP website in ZIP folder	26
8 Number of 3GPP documents in each release	27
9 Page label metadata missing while loading Doc format	27
10 Document converted into nodes after using SimpleDirectoryReader	28
11 Open source embedding models	29
12 Chroma architecture	30
13 Open source LLMs	31
14 UI components	32
15 Top K chunks	33
16 Text content in the chunk	34
17 User Interface	35
18 A good structured output with reference	37
19 Timer	38
20 Rating of Llama3 LLM and gte-large-en-v1.5 Embedding Model Based on 20 Telecommunications Questions Across Key Metrics	39
21 Output containing tabular data	40
22 Different responses from different chat interfaces	41
23 3GPP specification document, snippet of a table	42

24	Representation of data in the table reproduced in Figure 23 after loading and chunking	42
----	---	----

List of Tables

1	Comparison of Vector Databases	18
2	Comparison of Postprocessor Modules in LlamaIndex	20
3	LlamaIndex [Lla] vs Langchain [Lan]	23
4	Prompt for generating answers with correct references	31
5	Rating and Response Time	38
6	Pairwise Comparison of LLMs and Embedding Models with Metrics Ratings	38

Listings

1	Data Loading	27
2	Code example for splitting using semantic chunking	28

Acknowledgement

Writing this thesis has been an interesting journey for both my personal and academic growth. I learned something new every week, then iteratively improved upon it, and finally tried to express those insights. It was a truly enriching experience. I am grateful to everyone who contributed to my journey.

First, I would like to thank my university Hochschule für Technik und Wirtschaft Berlin for providing me with such great experience during my master's journey. With all the support and guidance they provided at every stage I had a wonderful experience here. I am deeply thankful to my first supervisor, Prof. Dr.-Ing. Thomas Schwotzer, for his insightful feedback and encouragement, which greatly enhanced the quality of my work.

I would also like to express my deep appreciation to Dr. Cristian Grozea, my second supervisor, for his guidance during my time at Fraunhofer FOKUS, where I held a student position. His insightful recommendations and constructive criticism greatly impacted my research process. I also extend my gratitude to the Software-based Networks department of Fraunhofer FOKUS (NGNI) for funding my student position and providing access to the infrastructure, particularly the GPU cluster, which was instrumental in my thesis work.

I am truly grateful for the mentorship and support from both of my supervisors, which propelled me to successfully complete this thesis. At last I would like to extend my gratitude to my family, friends and to all those who have supported me in any way during completion of my thesis.

Abstract

This thesis provides a detailed guideline to build a Large Language Model (LLM) based chat bot using Retrieval Augmented Generation (RAG) on a corpus of local documents with a focus on 3GPP technical specification documents. It utilizes existing open-source local models and software for the implementation. This thesis also covers the latest RAG methodologies with different architectures along with different embedding and large language models. It discusses briefly each step involved in the RAG system, including but not limited to extracting the data, data preprocessing, embedding, model selection, and user interface.

1 Introduction

The telecommunication industry produces a vast amount of technical documentation, such as 3rd Generation Partnership Project (3GPP) [3GPnd] specifications, which play a pivotal role in maintaining and developing telecommunication systems. These include but are not limited to 5G, mobile networks, and non-terrestrial communication networks. Due to the domain specific technical language and complex nature of these documents, extracting valuable information from them is a challenging task. This thesis, titled “Information Retrieval from Large Collections of Local Documents Using Large Language Models: RAG Approach,” covers the end to end implementation of Retrieval Augmented Generation (RAG) on these 3GPP documents. It discusses various findings from the process and also addresses results related to existing Large Language Model (LLM) based systems for 3GPP specifications.

1.1 Structure of the Thesis

The structure of the Thesis “Information Retrieval from Large Collections of Local Documents Using Large Language Models: RAG Approach” is as follows:

1. Introduction

- **Structure of the Thesis:** Gives an outline of the thesis structure.
- **Background and Motivation:** Highlights the challenges and the need of using an LLM based chat bot in the telecommunication industry. We will explore why this topic is important and what drives the investigation.
- **Research Questions:** It outline the specific questions my thesis aims to address. We will also highlight the benefits of addressing these questions.
- **Scope of the Thesis:** Defines the boundaries and focus of the thesis.

2. State of the Art

- **Related Work LLMs and 3GPP Specs:** Reviewing existing literature around the same problem.
- **Retrieval Augmented Generation (RAG):** Explain State of the art RAG architectures and break it down into its components, including pre-retrieval, indexing, embedding, retriever, and generation.
- **Applications of LLMs:** This section will dive into various applications of LLMs in different sectors, demonstrating its potential and impact.
- **LlamaIndex** We will introduce LlamaIndex, discussing its features and functionalities.

3. **Design and Implementation:** We will explain the design of the system outlining the key components of the architecture and describe the implementation in detail with a focus on data collection, data processing, and the user interface.
4. **Evaluation:** In this section, we evaluate our NGNI 3GPP chatbot by testing it with a variety of questions using different LLMs and embedding models, measured against various metrics.
5. **Conclusion:** In this section we will discuss the limitations and future work for RAG based NGNI 3GPP chat bot.

1.2 Background and Motivation

Today, AI has reached unprecedented capabilities and is still expanding its horizons. As building such intelligent systems integrated with a Large Language Model (LLM) requires substantial computing power and heavy resource dependency, they often incur high costs that are not feasible for small and mid sized organizations [Tho+22]. Thus, using pre-trained open source models offers a viable alternative. One of the biggest challenges is ensuring

the relevance of the intelligence of the model to the specific industry. For example, a law firm would require an LLM that understands natural language and has up-to-date knowledge of the legal domain, otherwise it would not produce expected results.

To solve this problem, one approach could be to train our model on domain-specific knowledge. However, this introduces two new obstacles: 1) Training a model is expensive as it requires a large number of GPUs and significant computing resources, 2) Even if we train the LLM with the most recent data, new data will continue to come in and we won't be able to retrain the model for every new piece of information. Retrieval Augmented Generation (RAG) [Lew+21] is a popular approach to bring domain specific knowledge to the LLM. This is the topic of my thesis, and the domain I have decided to address is telecommunications. Telecommunication specifications are often very technical and detailed. In order to find particular specifications from standards like the 3rd Generation Partnership Project (3GPP), one might need to read multiple documents to form a conclusion. This is where LLMs come into the picture.

LLMs can be very useful as they can read and understand vast amounts of data in seconds. As part of the thesis I built a NGNI 3GPP chat bot that takes a user query, searches through 3GPP documents, retrieves relevant chunks, and brings them back to the LLM. This retrieved piece of information can be further modified based on defined prompt engineering, to answer the user query ultimately providing a response. The telecommunication community can benefit immensely from this NGNI 3GPP chat bot as it saves significant time and effort in analyzing documents.

The ever-evolving field of telecommunication poses unique challenges when implementing a RAG system due to its complex, high-level technical specifications, as seen in 3GPP documents [Maa+24]. These documents also contain a variety of numerical calculations, tables, and figures, which makes us consider a multi-modal architecture for retrieval. However, as we delve further into this thesis, we will realize the need for a simple structure that extracts only text for RAG.

1.3 Research Questions

- What are the bottlenecks in processing 3GPP specification documents?
- RAG vs Fine-tuning?
- Why RAG for this Problem?
- How to develop a RAG system for 3GPP specification documents?
- How do different text splitting techniques affect retrieval accuracy?

1.3.1 Benefits

- Privacy: Conversing with our local documents without sharing these with an online provider such as OpenAI.
- Saves time by circumventing the time-consuming process of reading multiple technical telecommunication 3GPP documents, thereby increasing productivity.
- Such systems can also be fine tuned for other technical domains.

1.4 Scope of the Thesis

This thesis encompasses two primary objectives. First understanding and solving the challenges posed by the 3GPP documents or similar technical documents while using RAG. It also includes the technical implementation of a RAG system for 3GPP documents covering the below aspects

1. Dataset collection
2. Data processing
3. Selection of the embedding model
4. Selection of the LLM base model
5. Model fine-tuning

6. Evaluation metrics
7. Comparative analysis.

The second part talks about different RAG architectures and its components. We also talk about different applications of RAG, vector databases and LLM frameworks available.

2 State of the Art

This section discusses various topics related to 3GPP specifications, Large language models, RAG, Vector data bases, embedding models.

2.1 Related Work- LLMs and 3GPP Specs

As presented in papers named TSpec-LLM [N BG24] and Telco-RAG [Bor+24] similar approaches have been made to build a RAG system based for 3GPP specification. In Telco-RAG the authors used a technique named “NN routing” to decide which retriever to use from different data sets. They also used Multiple Choice Questions (MCQ’s) to query the LLM. This is where my approach differs. I designed and developed a NGNI 3GPP chat bot where you can have a conversation related to telecommunication. In my solution there is no need to give multiple option for the LLM to choose an answer from. Similar approaches have been made to address this problem and make the labour intensive task of going through different specifications easy [Bor+24].

3GPP specifications cover cellular telecommunication technologies, including radio access, core network, and service capabilities, which provide a complete system description for mobile telecommunications [3GPnd]. The 3GPP specifications also provide hooks for non-radio access to the core network and for inter working with non-3GPP networks [ETS24]. For our NGNI 3GPP chat bot we are going to process release 19 of Series 38 specification documents which covers topics related to 5G radio technology. We will discuss more about this in the data collection section 3.2 of the thesis [Kar+24].

2.2 Retrieval Augmented Generation (RAG)

LLM performs well on general user queries as they are trained on huge and diverse knowledge base. However they fail to include all available knowledge of every particular domain such as telecommunication, which is ever evolving and gets regular updates. RAG [Lew+21] tries to solve this prob-

lem. It helps in knowledge intensive tasks by allowing continuous updates and integration of domain specific information. Additionally, sharing sensitive data with online LLMs providers poses a security threat. It is a good practice to keep local data in a local environment within our trust boundary as sharing such data outside the trust boundary is a security threat. RAG when combined with open source LLMs could also address this issue and will make sure that the data stays within our local environment and trust boundary [Gao+24].

A basic RAG architecture (see Figure 1) consists of a retriever which is responsible for retrieving all relevant pieces of data from the corpus of local documents –in our case 3GPP specification documents. Before retrieving we split the data into chunks and then embed these chunks and store those in a vector database. After retrieving the top k chunks next comes the generator part. When the user queries the LLM, the relevant chunks are retrieved from the vector database and along with the user query are fed to the LLM. The LLM then analyses both the query and the context given by the chunks and answers according to the user query. This is the most basic architecture of RAG [Lew+21]. While it does solve the problem of keeping the system updated with the latest information, it also produces its own challenges. Information overload can be one of those. This happens when we retrieve chunks from the vector database and get an overwhelming number of chunks. Due to this huge number of chunks the LLM might not be able to process and might hallucinate [Gao+24]. Additionally, there is a significant risk of retrieving irrelevant chunks. For example, if a user queries the LLM, “What are NTN channels for FR1?”, the retriever might return chunks simply because they contain the term “NTN,”. Some of these chunks might discuss completely unrelated topics thus providing irrelevant information. To mitigate these challenges we add different components in the basic architecture to make the system more robust.

Each component of the RAG system can be further subdivided into subsections based on different use cases. Adding more components to this basic architecture transforms it into an advanced RAG, that includes components

like pre-retrieval, routing, query rewriting, post-retrieval processing and so on which will be discussed in detail in the next section [HH24].

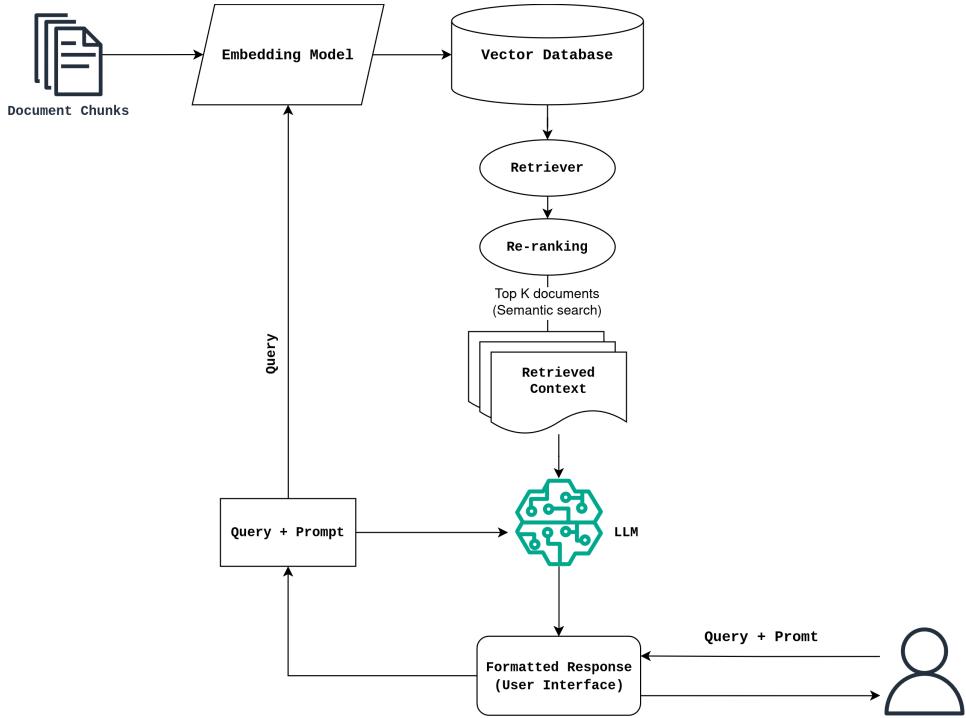


Figure 1: RAG architecture

2.2.1 Pre-retrieval

Any processing done before the retrieval process can be called as pre-retrieval. This includes components like chunking, indexing, query rewriting, query expansion and query transformation. All these components try to address the challenges posed by the basic RAG architecture.

Data Structure: Selecting a dataset in the appropriate format can facilitate smoother handling of subsequent processes. Texts are most easy to handle as they are unstructured and their raw format makes it easy for parsing and later

on embedding. On the other hand a PDF format has its own challenges. It's semi structured format makes it difficult to read tables and images present in the data set. Thus there is a loss of information making the retrieval less efficient [Gao+24]. Another factor that influences the selection of a dataset in a particular format is metadata. Every format provides different metadata for example PDFs provide page number as one of their metadata where as a text file or a doc file may or may not provide it. In some use cases having page number as metadata is more importance so one might prefer PDFs.

Chunking: To ensure that the retrieved chunks are relevant, chunking plays an crucial role in maintaining the meaning of the chunks. Different strategies can be applied for chunking. Below are the most prominent chunking strategies widely used. View Figure 2 for visual representation.

- Fixed token length: This method splits the document into small same length chunks based on predetermined fixed number of tokens (e.g. 100, 256).
- Sliding window: In this method the split is made with fixed token along with some tokens overlapping between the chunks, depending upon the predetermined overlap token value. This makes sure that the chunks boundaries retain some of the contextual information.
- Semantic: An LLM or vector embeddings can be used to split the based on semantic similarities. Now we can keep the most semantically related sentences together in a chunk.

Longer chunks retain context better and are therefore more informative, though they might have some redundant information which could distract the retriever and the LLM. On the other hand smaller chunks might put burden on the retriever and the semantic meaning would be missing. This leads to information loss at the end of each chunk. To address this issue semantic chunking might be helpful [Pra23].

The retrieved chunks are as good as the user query. Since the retrieved chunks are going to be compared against the user query we need to make

sure the user query is grammatically correct and do not lack semantics of the conversation. For this purpose we use query rewriting and query expansion. Query rewriting enhances the user query using LLMs, to make it grammatically sound and able to retrieve relevant chunks. Whereas query expansion gives an idea about previous query to help the LLM give context to build a relevant query, this method makes sure a conversation relevance is met [Fan+24].

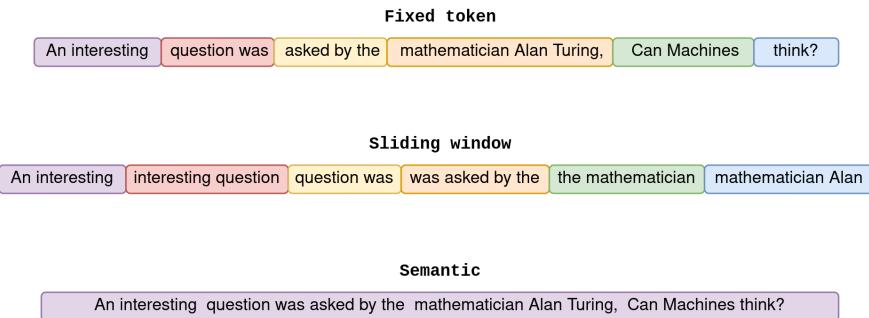


Figure 2: Visual representation of different types of chunking

To improve RAG’s efficiency for large document corpora like 3GPP specification documents in terms of RAM, NN routing can be introduced as one of the solutions. 3GPP specification documents are divided into 18 series. In NN routing the model will be trained such that only the relevant series embedding will be loaded, thus drastically reducing the RAM usage (see Figure 3).

2.2.2 Indexing and Embedding

In any RAG system indexing is very important. Indexing begins with data cleaning via removing stop words, preprocessing on some schema based on a particular use case and tokenization. The final step of indexing involves mapping data into higher-dimensional numerical representations, commonly known as vectors. These vectors are instrumental in retrieval by leveraging cosine similarity to identify relevant data points within the embeddings.

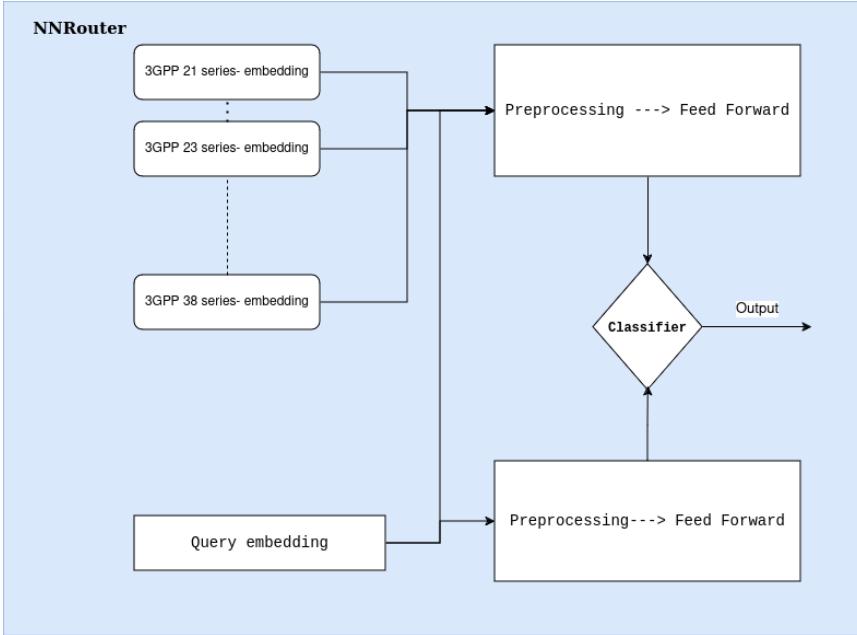


Figure 3: NN Router

Different embedding models are used to transform them into vectors and if required store them in a vector database. The deciding factors for any embedding models are as follows:

- Model size: It is the number of parameters in a model. Larger models are more accurate and usually perform better but require more computational power and memory resources. Small models are efficient but might be less accurate. For example, the model all-mpnet-base-v2 has 420 Mb as model size, which makes it a small, yet powerful embedding model.
- Embedding Dimensions: It refers to the number of features or dimensions generated for each input by the embedding model. For example, the model all-mpnet-base-v2 has embedding dimensions of 768. This means it represents each input as a vector in a 768 dimensional space. A higher dimension usually represents its capabilities of capturing more details of each input.
- Max Tokens: It is the maximum number of tokens used during pro-

cessing. e.g all-mpnet-base-v2 has 514 max tokens. A higher token allows to process larger text which is important while dealing with long context documents.

In summary, an embedding model needs to be selected in a way that balances computational efficiency and performance based on the targeted use case [Xie+23].

2.2.3 Vector Database

Once we have embedded the chunks we need a database to store it so that we can retrieve it multiple times and there is no need of embedding the chunks every time we need to use the retriever. There are many paid as well as open source solutions available in the market [Xie+23].

Vector Database	Open Source	Distance Metrics	ANN Algorithms
Qdrant	Yes	Cosine, Dot Product, L2	HNSW
Chroma	Yes	Cosine, Euclidean, Dot Product	HNSW
Pinecone	No	Cosine, Euclidean, Dot Product	Proprietary (Pinecone Graph Algorithm)

Table 1: Comparison of Vector Databases

As seen in the Table 1 different vector data bases provides different features, while open source vector bases are free of cost other vector data base costs us depending on the volume of data stored in the vector databases. Paid vector databases also provide unique functionalities like fully-managed service for boosting performance and making it more scalable. That's why Pinecone which is scalable cannot be deployed locally where as Chroma and Quadrant can be deployed locally [Xie+23]. The storing, searching and retrieving capabilities have made vector databases popular in the domain of ML/AI. These combined with LLMs makes it possible to build scalable AI applications [HLW23] .

2.2.4 Retriever

A retriever is responsible with the extraction of the query relevant chunks from the vector database. Frameworks like Langchain and LlamaIndex provide different inbuilt retrievers which focus on different retrieving tech-

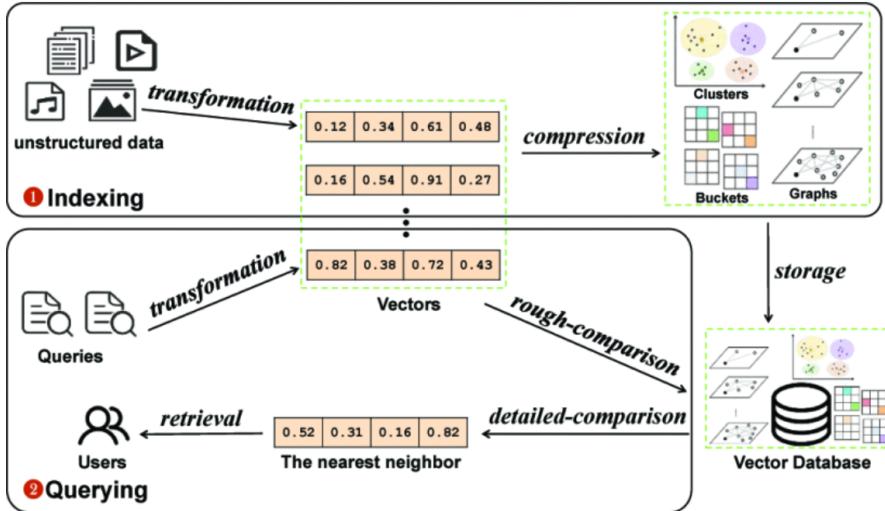


Figure 4: Vector database architecture adapted from [Xie+23]

niques. Usually they are built on top of indexes but the frameworks also provide functionality to build one independently. Different modules are called upon to retrieve the top k chunks depending upon the above frameworks used. The next important step is post retrieval processing, which enhances results of the retrieval passed to the LLM together with the query.

2.2.5 Post Node Processing

Frameworks like LlamaIndex provide different node post processing modules, which helps in preparing the retrieved chunks or nodes for the LLM. See Table 2 for the most commonly used post node processing module techniques.

Table 2 shows different methods to improve the post retrieval phase and explain its functionalities. One can choose one of those or build one of their own. What matters is the core strategy. One such strategy is the re-ranking. It helps to remove irrelevant chunks from the retrieved top k chunks and prioritize the most relevant information using the similarity score between

Postprocessor Module	Functionality	Key Parameters	Use Case
SimilarityPostprocessor	Removes nodes that are below a similarity score threshold.	similarity_cutoff	Ideal for filtering out nodes that don't meet a required level of similarity.
KeywordNodePostprocessor	Ensures certain keywords are either excluded or included.	required_keywords, exclude_keywords	Useful for focusing on or avoiding specific keywords in nodes.
MetadataReplacement	Replaces node content with a field from the node metadata, if available.	target_metadata_key	Best used when nodes need to be replaced with metadata content.
LongContextReorder	Re-orders nodes to prioritize important details at the start or end of the context.	None	Improves performance on long contexts by re-ordering nodes to highlight key information.
SentenceEmbeddingOptimizer	Optimizes token usage by removing irrelevant sentences based on embeddings.	embed_model, percentile_cutoff	Refines nodes by eliminating non-relevant content, keeping only the most pertinent sentences.
CohereRerank	Re-orders nodes using Cohere ReRank API and returns the top N nodes.	top_n, model, api_key	Reranks nodes based on Cohere's API to get the most relevant nodes.
SentenceTransformerRerank	Uses cross-encoders from the sentence-transformer package to re-order nodes.	model, top_n	Reranks nodes with sentence-transformers, balancing speed and accuracy.
LLM Rerank	Uses a Large Language Model (LLM) to re-order nodes, returning the top N nodes.	top_n, service_context	Leverages LLMs to rank node relevance, particularly useful for high-level understanding.
FixedRecencyPostprocessor	Returns the top K nodes sorted by date, assuming a date field in the metadata.	top_k, date_key	Prioritizes nodes based on recency of information, useful for keeping the most recent nodes.

Table 2: Comparison of Postprocessor Modules in LlamaIndex

query and the chunks. We can also use an LLM for selecting the most relevant chunks.

Another post retrieval module is routing, also provided by frameworks like LlamaIndex . (Refer Figure 3) Routing is a technique that is used when we have more than one retriever. Those retrievers might be connected to different data sources. While responding to any user query the router will first decide which retrieval to use or prioritize. Similar approaches have been used in Telco-RAG where the authors implemented neural network routing [Bor+24] [Maa+24].

2.2.6 Generation

As stated in the book Fundamentals of Artificial Intelligence “Natural language processing (NLP) is a collection of computational techniques for automatic analysis and representation of human languages, motivated by theory” [Cho20]. In simple words NLP can be defined as the techniques devised to help computers/machines process human language. On a deeper level, the core of NLP lies in natural language understanding, where semantics and pragmatics plays an important role in achieving natural language understanding [Cho20]. NLP techniques are essential for parsing and understanding the input query, facilitating the retrieval of relevant chunks from external sources which are then used by the generation component to produce accurate and contextually appropriate outputs. Generation completes the RAG system by generating the output based on the chunks retrieved from external data sources. It employs advanced prompt engineering techniques to optimize the interaction between the Large Language Model (LLM) and these chunks ensuring more accurate and contextually relevant responses. Every model can generate texts but they might not necessarily generate correct answers, this is due to the fact that they have different power limited by parameters like model size, context window length, architecture and training set.

2.3 Applications of LLMs

As discussed in section 4 the transition from basic to advanced RAG system enhances LLM capabilities significantly. This advancement is not just limited to telecommunication domain but also has a lot of potential in multiple domains such as education, healthcare, technology, mobility, customer service chat bot etc.

- **Education:** LLM’s huge knowledge base along with RAG can be utilised to improve student’s learning experience and help teaching professionals reach every student by creating different learning support tools. These include development of tools which prioritize per-

sonalized learning experiences, educational content creation and generation platforms, language learning and teaching platforms fostering cross-lingual communication and translation tools [Li+24].

On one hand, the integration of LLMs in education has many advantages. On the other hand their use also presents significant ethical challenges. For instance, students can use them to cheat, instead of using it as a learning tool [Li+24]. To address these issues, another set of tools and frameworks should also be in place to serve as guardrails for the use of LLMs in an educational settings [Had+23] [Lat+24].

- **Medical:** Large Language Models (LLMs) like ChatGPT have proven tremendous potential in improving numerous parts of medical practice, including improving clinical documentation, hospital workflow management, improving diagnostic accuracy, and patient care management[Xie+24].

The paper by [Xie+24] introduces "Me-LLaMA" a novel family of medical LLMs. It is build using large corpus of medical dataset and involved the pre training and fine tuning of LLaMA2 model. Also there are numerous factors to consider while properly implementing LLMs in medicine. Key aspects to consider include transfer learning, domain-specific fine-tuning, domain adaptation, reinforcement learning with expert input, dynamic training, interdisciplinary collaboration, education and training, evaluation metrics and clinical validation [KM23]. Data privacy, ethical consideration and regulatory frameworks also play a pivotal role in shaping the development of these LLMs in the healthcare sector [Had+23].

2.4 LlamaIndex

LlamaIndex [Lla] is an open source data framework developed by Anthropic, designed to build reliable information retrieval systems for LLM applications. It provides an end to end solution for building scalable applications right from injecting data to building chat engine with a user interface. Some

of the features offered by LlamaIndex are mentioned below:

- Provides data connectors for injecting data from wide range of data sources and data formats including APIs, SQL, PDFs, docs, mark-down, json etc.
- Provides options to structure the data through indices and graphs for efficient retrieval and generation.
- Allows easy integration with LLMs, such as Hugging Face Transformers, vLLM, Ollama making it well suited for different AI application usecases.
- Allows easy integration with external frameworks and libraries like LangChain, Flask, Docker, ChatGPT, Gradio and many more.

Feature	LlamaIndex	Langchain
Primary Purpose	Search and retrieval tasks	Building applications powered by large language models (LLMs)
Strengths	Fast, efficient, accurate, ideal for large data sets, simple interface	Flexible, versatile, customizable, supports diverse LLMs, advanced context-awareness
Weaknesses	Limited to search and retrieval tasks, less flexibility in customization	Steeper learning curve, complex for beginners, resource-intensive for complex applications
Use Cases	Document search, code generation, customer service chat bots, content filtering	chat bots, virtual assistants, knowledge bases, personalized learning platforms, creative writing tools
Ease of Use	Easy	Moderate
Documentation	Good	Extensive
Cost	Free	Free

Table 3: LlamaIndex [Lla] vs Langchain [Lan]

3 Design and Implementation

This section covers the design and the steps involved in building a RAG system with LlamaIndex , particularly for 3GPP documents. It discusses each step in detail and the challenges faced during the implementation of each step, from data preprocessing up to integrating the LLM.

3.1 Design

This section covers the design of the RAG system in detail covering every element used in the architecture. Figure 5 depicts the architecture used for our RAG system. We discuss the implementation of these components in detail in the later sections.

We use LlamaIndex as framework to facilitate our RAG system. Table 3 shows us the comparison between two prominently used frameworks. We decided to use LlamaIndex as its features which we have already discussed in the section 2.4 are ideal for our use case. LlamaIndex provides connectors which loads our documents into the vector database. As mentioned in section 2.2.3, we selected Chroma due to its open source nature. When a user submits a query, the LLM receives the query and the retrieved chunks from the vector database. This is carried out by the chat engine components provided by LlamaIndex. Together it is responsible for retrieving, query re-writing, augmentation and finally generating. In the following sections, we will discuss the implementation of each step in detail.

3.2 Data Collection

The 3GPP documents are freely available on official 3GPP and ETSI websites. The challenge arrives when we need to download it in a particular format. The 3GPP website provides different folders from release 9 to release 19 and each release has many series with ZIP folders which hold .doc format files (see Figure 7). On the other hand the ETSI website is the same but provides the 3GPP documents in a PDF form (see Figure 6).

Earlier I used .doc version from 3GPP website, then converted it into

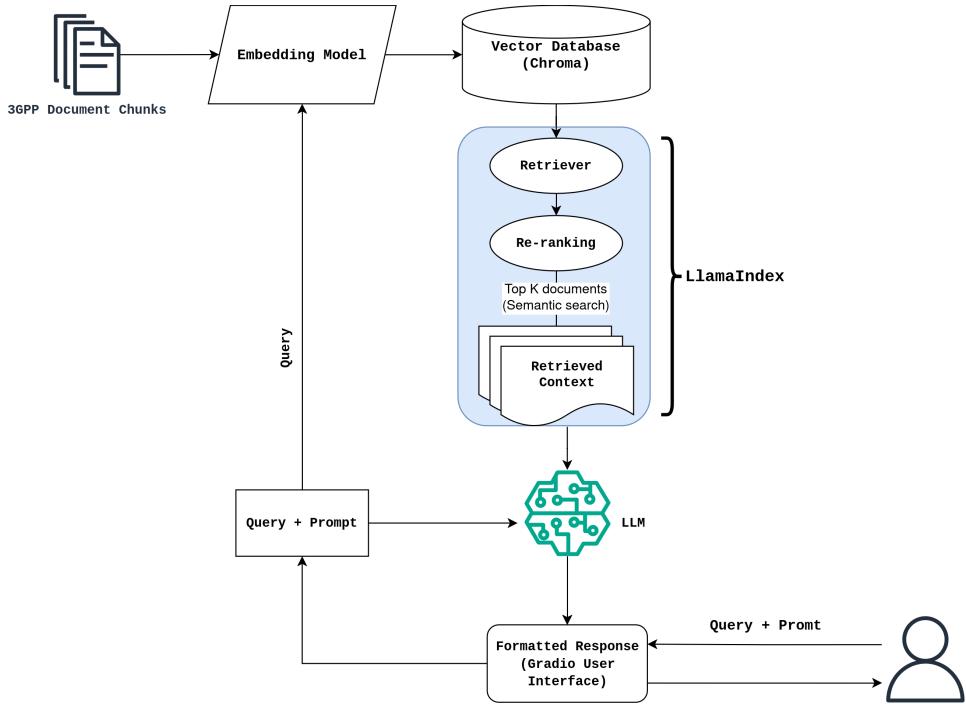


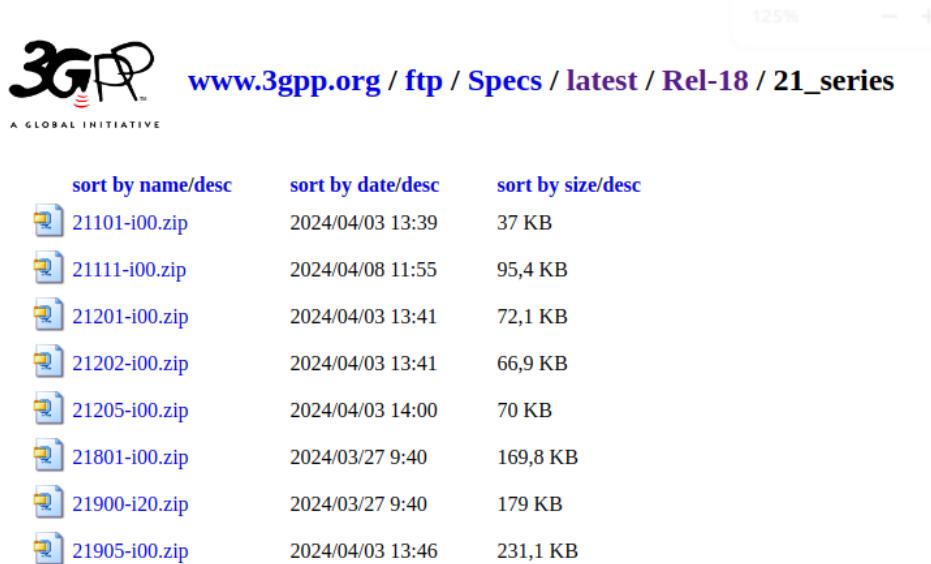
Figure 5: RAG architecture implemented

.docx format for preprocessing it with LlamaIndex as it requires .docx format to convert the document into nodes and then later on store it in vector database. While scaling I realised that .docx format did not include some metadata features like page number which was crucial for our use case, so I opted for the ETSI website which gives us the PDF version of the technical specifications including crucial page number metadata. This is illustrated in Figure 9, which compares the metadata features of PDF and .doc formats. With PDFs one can get meta data information like page label easily while loading the data itself which we will discuss in the next subsection. One might use .doc format from 3GPP website and convert it into PDFs if using a Windows machine, since on linux machines it's a little challenging to convert into PDFs since Libre Office was not able to convert all files into PDFs due to the complexity and size of the documents. Figure 8 shows the statistics of number of documents in each release. Here the Y axis denotes the number of document in thousands and X axis denotes the release number.

**www.etsi.org -
/deliver/etsi_ts/138500_138599/13852103/18.03.00_60/**

[\[To Parent Directory\]](#)
8/7/2024 1:01 PM 5880320 [ts_13852103v180300p.pdf](#)

Figure 6: 3GPP specifications on ETSI website in PDF format



The screenshot shows a list of ZIP files on the 3GPP website. The files are sorted by name in descending order. Each file entry includes a small icon of a ZIP file, the file name, the upload date, and the file size.

sort by name/desc	sort by date/desc	sort by size/desc
21101-i00.zip	2024/04/03 13:39	37 KB
21111-i00.zip	2024/04/08 11:55	95,4 KB
21201-i00.zip	2024/04/03 13:41	72,1 KB
21202-i00.zip	2024/04/03 13:41	66,9 KB
21205-i00.zip	2024/04/03 14:00	70 KB
21801-i00.zip	2024/03/27 9:40	169,8 KB
21900-i20.zip	2024/03/27 9:40	179 KB
21905-i00.zip	2024/04/03 13:46	231,1 KB

Figure 7: 3GPP specifications on 3GPP website in ZIP folder

3.3 Data Processing

LlamaIndex provides a simple directory function which helps in loading the data and converting them into nodes (a format which is easily integrated with other LlamaIndex's functions and processes). Llamaindex provides different connectors for loading data from different sources like web, Discord, Notion and different file types including .pdf, .jpg, .png, .docx. For this implementation I have used all the 3GPP documents from series 38 - release 19. In Listing 2 a snippet of code for loading the data is shown.

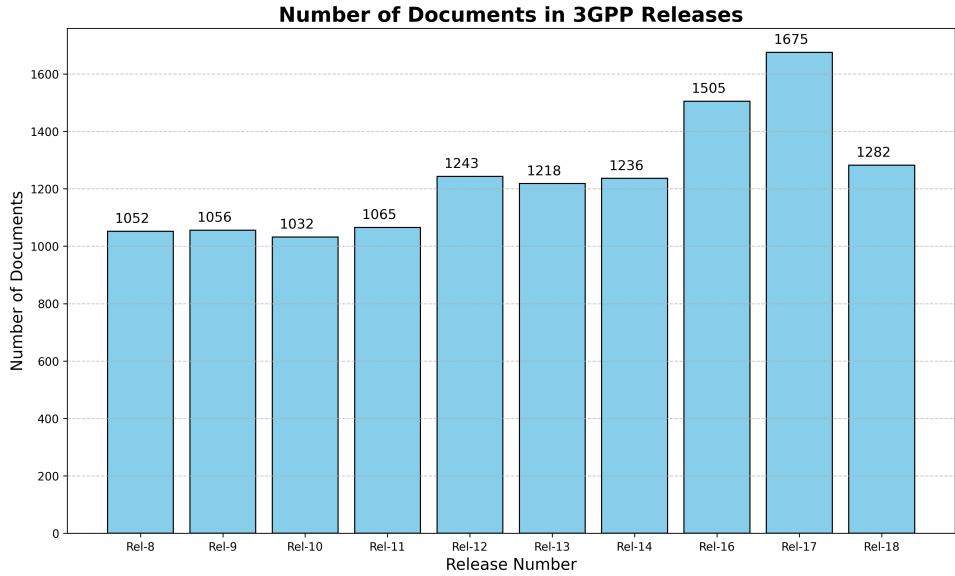


Figure 8: Number of 3GPP documents in each release

```

Loading a document file
Document(id_='ec9cf029-c6f2-4be7-a36b-f356463d709c', embedding=None, metadata={'file_name': '23093-a01.docx', 'file_path': '/home/jovyan/test/master-thesis/other_notebooks/3gpp_18/23093-a01.docx', 'file_type': 'application/vnd.openxmlformats-officedocument.wordprocessingml.document', 'file_size': 898240, 'creation_date': '2024-08-14', 'last_modified_date': '2024-08-14'}, excluded_embed_metadata_keys=['file_name', 'file_type', 'file_size', 'creation_date', 'last_modified_date', 'last_accessed_date'], excluded_llm_metadata_keys=['file_']

Loading a PDF file
Document(id_='856e1d10-569f-4c68-96c1-c89a3cede47f', embedding=None, metadata={'page_label': '2', 'file_name': '3gpp_18.pdf', 'file_path': '/home/jovyan/test/master-thesis/other_notebooks/3gpp_18/3gpp_18.pdf', 'file_type': 'application/pdf', 'file_size': 7393767, 'creation_date': '2024-08-14', 'last_modified_date': '2024-08-14'}, excluded_embed_metadata_keys=['file_name', 'file_type', 'file_size', 'creation_date', 'last_modified_date', 'last_accessed_date'], excluded_llm_metadata_keys=['file_name', 'file_type', 'file_size', 'creation_']

```

Figure 9: Page label metadata missing while loading Doc format

```

1 from llama_index.core import SimpleDirectoryReader
2 loader = SimpleDirectoryReader(input_dir=". ./data/3gpp_docs/")

```

Listing 1: Data Loading

LibreOffice has difficulty processing some files, likely due to their large size, with some documents reaching around 1000 pages.

Figure 10: Document converted into nodes after using SimpleDirectoryReader

3.3.1 Chunking

Chunking plays a crucial role in any RAG system, as its implementation is decisive for the retrieval phase. My first chunking approach was based on a python script which I wrote by setting a token limit of 512 (which can be changed). This approach had a major flaw. This method breaks and creates a new chunk in the middle of a thought process or conversational text, which inturns gives the retriever incomplete chunks and thus the inference would not be accurate. To address this problem, I introduced regex pattern recognition in my script which was able to identify and break at every new section or subsection thus holding the contextual meaning of the chunks. In the later stage while scaling I realised the need of using a framework to streamline whole process so I used LlamaIndex. LlamaIndex provides with many inbuilt chunking functionalities like SentenceSplitter, SentenceWindowNodeParser, SemanticSplitterNodeParser, TokenTextSplitter and many more. I chose SemanticSplitterNodeParser as it provides us with more relevant semantically interconnected sentences in a chunk than SentenceSplitter, which used to cut the sentence in the middle of a topic [Men24].

LlamaIndex provides different types of parsers depending on the file type. It then creates node objects as chunks from the documents. Later on LlamaIndex indexes these nodes using vector embeddings, enabling fast and semantic search. SimpleFileNodeParser is one of those which helps to parse different types of file formats like PDF, Doc, MD etc.

```

1 from llama_index.core.node_parser import SemanticSplitterNodeParser
2 from llama_index.embeddings.openai import OpenAIEmbedding
3
4 embed_model = settings.embed_model()
5 splitter = SemanticSplitterNodeParser(buffer_size=1,
                                         breakpoint_percentile_threshold=95, embed_model=embed_model)

```

Listing 2: Code example for splitting using semantic chunking

3.3.2 Embedding

Embedding can be defined as representation of data into high dimensional vectors. They are useful in RAG system to perform cosine similarity, which is essential to find top K relevant chunks while retrieving [HH24]. Top embeddings could be found at hugging face's MTEB [Mue+22] leaderboard. MTEB stands for Massive Text Embedding Benchmark. It is a comprehensive benchmark to evaluate the performance of text embedding models across various parameters and diverse embedding tasks. It's leader board provides us with the top performing models across variety of task. Those rankings are based on re-ranking, summarization, clustering, retrieval and many more ratings which are tested on a MTEB dataset.

Embedding model	Alibaba-NLP/gte-large-en-v1.5
	thenlper/gte-base
	all-MiniLM-L6-v2
	all-mpnet-base-v2

Figure 11: Open source embedding models

3.3.3 Vector Database

There are many Vector databases that are available in the market most prominent being Milvus ,Weaviate, Pinecone, Chroma. Since my focus was on us-

ing open source software my obvious choice was Chroma [HLW23]. Refer Figure 12 for Chroma architecture.

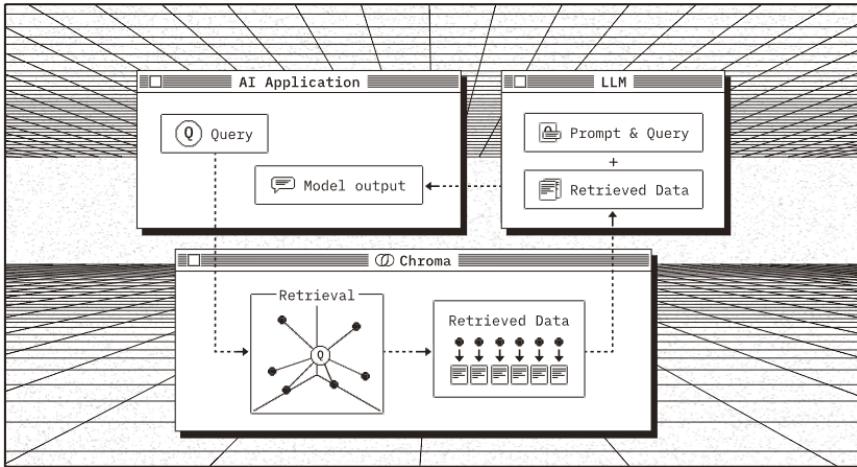


Figure 12: Chroma architecture

3.3.4 LLMs

Open source LLMs were used to keep the data in the trust boundary. See Figure 13 llama3 being the biggest model with 70b parameters. The larger the model, the more context it can understand, and the large models are also better at understanding prompts and follow those strictly, providing thus the expected results. To make sure that the LLMs follow the prompt strictly, the temperature hyper parameter can be tuned to zero. This parameter across various LLMs helps us have more control over the result, thus maintaining the consistency and reproducibility of the responses and making the RAG system more robust.

3.4 User Interface

A functional and user-friendly user interface (UI) is essential for enhancing the user experience and engagement in complex systems like our Retrieval-Augmented Generation (RAG) system. Gradio [Abi+19] and Streamlit are well known user interface frameworks for AI applications. Gradio was used

```

1 "Here are the relevant text chunks for the context:
2   {context_str}
3 For each text chunk you received such location information as
4   in the following example-
5   (page_label:..... ,file_path:..... )
6 You are tasked with answering questions based strictly on the
7   provided text chunks. Follow these specific instructions
8   closely:
9   1. If the information is not in the provided text chunks,
10      respond with 'I couldn't find any relevant information
11      in the
12      provided documents'.
13   2. After each complete answer, on a new line include the
14      reference to the text chunks, specifying the page_label
15      on
16      the relevant text chunk and file_path
17   3. Take a step back and cross-check your answer and
18      citations for accuracy.
19   4. Maintain a consistent and well-structured format in all
20      answers."

```

Table 4: Prompt for generating answers with correct references

Category	Model
LLMs	llama3:70b
	mistral
	command-r
	nous-hermes2-mixtral:8x7b

Figure 13: Open source LLMs

for the NGNI 3GPP chat bot since it had better customization functions. Gradio was chosen also because of its ease of use, flexibility, and several integrations with machine learning frameworks like LlamaIndex and langchain. Another option would be creating the front end with flask, which can offer even more control over how buttons and other features will look.

Considering the complexities of the RAG system, which involves multi-

ple components like loading, indexing, retrieving, generation, the interface needed to accommodate multiple functionalities. These functionalities include query input, multiple buttons, display of generated responses, and visualization of the top retrieved chunks of information. The aim of the user interface is to create an environment and make it easy for the user to interact with the NGNI 3GPP chat bot, to be able to converse about a topic, to check the top k chunks for the queries as well as see the chunk's context.

3.4.1 Key Features

Gradio provides low level API called ‘Blocks’, which was instrumental for implementing our NGNI 3GPP chat bot. Blocks provided good control over the overall layout and interaction of the components of our chat bot. The core components of the user interface include:

- **Text Box:** This is the place where the user inputs his queries. This component was styled to improve user experience and ease of use. See Figure 14 for a visual representation.

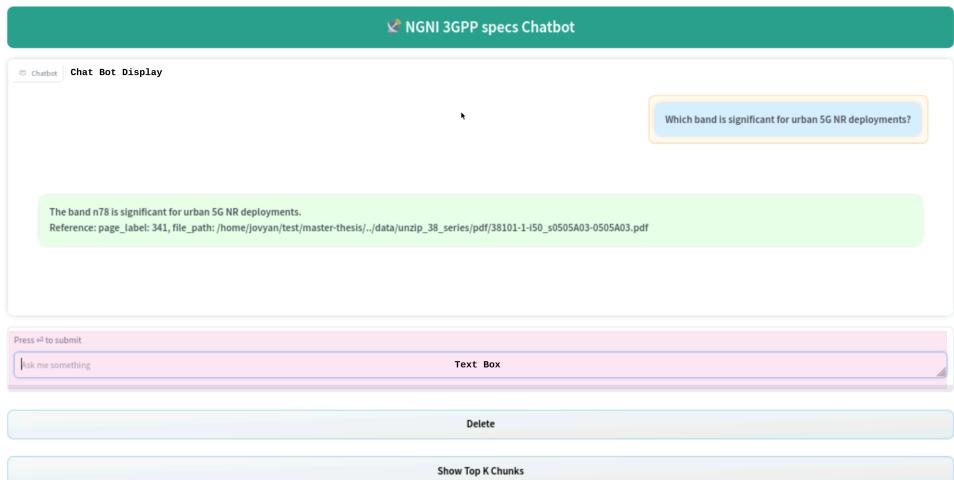


Figure 14: UI components

- **Chat Bot Display:** This is the place where the display for user query and response from LLM is displayed. This also stores the history of

the conversation unless and until delete button is clicked. For a visual representation of the UI, see Figure 14.

- **Top k Chunks Button:** In the backend LlamaIndex sends the top k chunks retrieved from the vector database to the LLM. Since these retrieved information could also hold valuable clues for the user we decided to show the top k chunks as well . For this the user needs to click on this button for the top k chunks metadata like the file location, file name and page number to be visible. Then we use the accordion component which we will discuss next to see the content in that chunk. For a visual example of the button’s functionality, refer to Figure 15.

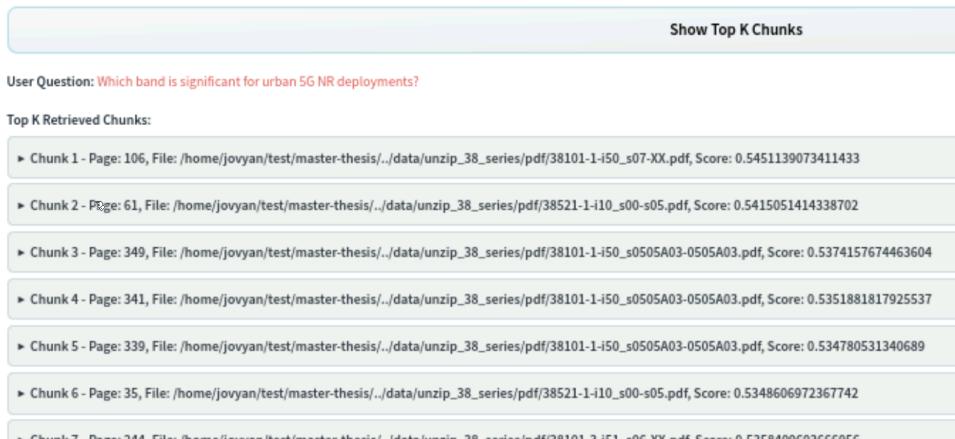


Figure 15: Top K chunks

- **Accordion Component:** The content of the top k chunks is initially hidden with the ‘accordion’ component. The user can then choose between the content’s visibility, which helps to efficiently manage the space and provide a neat and organized interface. Illustrated in Figure 16, the component displays chunk headers that include the page number, file location, and similarity score relative to the query and corresponding content upon clicking the headers.
- **Delete Button:** This button was introduced to reset the conversation history not only on the screen but also from the memory of the chat

engine. This facilitates iterative testing and ensures that the assessment is not compromised by portions that are previously saved in the chat history and benefit from being answered. This button invokes the “chat-engine.reset()” function, ensuring the user interface and the conversation was reset.

▼ Chunk 9 - Page: 531, File: /home/jovyan/test/master-thesis/..../data/unzip_38_series/pdf/38101-3-i51_s00-05.pdf, Score: 0.5428157481230819

For brevity the said configuration for operating bands for DC with Band n90 are not listed in the tables below but are covered by this specification. Non contiguous resource allocation and almost contiguous allocation are not applicable for E UTRA or NR carrier part of intra band EN DC configuration. If the mandatory simultaneous Rx/Tx capability applies for a lower order DC configuration, when the applicable lower order DC configuration is a band pair in a higher order DC configuration, the mandatory simultaneous Rx/Tx capability also applies for the band pair in the higher order DC configuration. For a higher order EN-DC band combination of which DC_20_n28/DC_28_n20/CA_20-28/CA_n20-n28 is a subset, the frequency range in band n28/28 is restricted for the higher order band combination to 703-733 MHz for the UL and 758-788 MHz for the DL. For NR inter-band dual connectivity specified in 5.5B.7, the corresponding NR CA configurations in 5.5A.1, i.e., dual uplink inter-band carrier aggregation between FR1 and FR2 with uplink assigned to two NR bands, are applicable to Dual Connectivity. NOTE 1: Requirements for the dual connectivity configurations are defined in the clause corresponding NR uplink CA between FR1 and FR2 configurations, unless otherwise specified. 3GPP544 Release 18 3GPP TS 38.101-3 V18.5.1 (2024-03)

► Chunk 10 - Page: 247, File: /home/jovyan/test/master-thesis/..../data/unzip_38_series/pdf/38101-3-i51_s06-XX.pdf, Score: 0.5524357943119242

Figure 16: Text content in the chunk

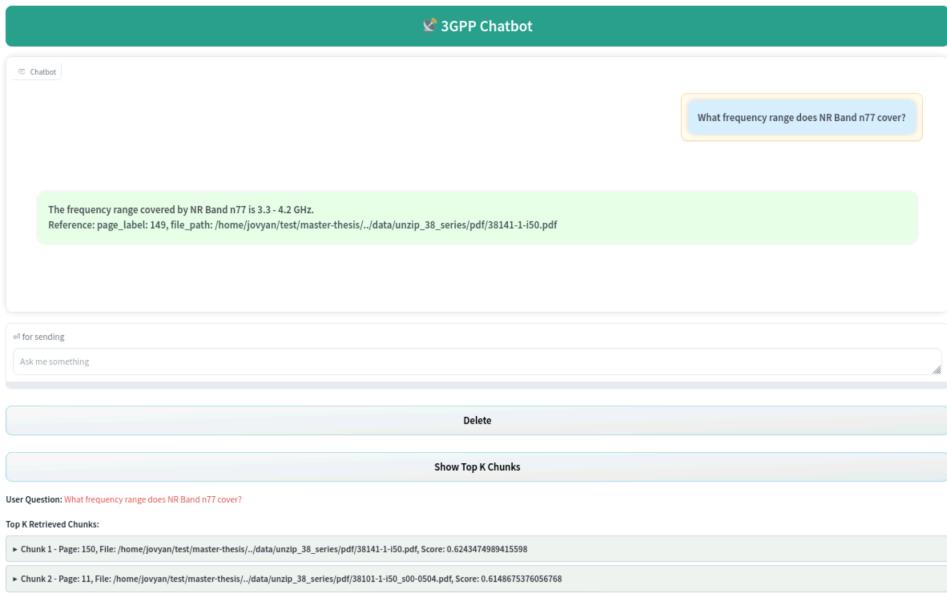


Figure 17: User Interface

4 Evaluation

In this section we discuss about our evaluation method and evaluate responses from different combinations of embeddings and LLMs. Our goal is to analyse and find the ideal combinations that suits our use case, which is telecommunications expert NGNI 3GPP chat bot. Although there are many state of the art quantitative evaluation metrics like precision, F1 score, recall etc, we will focus on qualitative metrics like structure, accuracy, relevance, redundancy, completeness and response time. One of the main reasons for choosing these metrics is our particular use case which also values other parameters of the response like structure, redundancy which makes accessing the chat bot better. These metrics were rated between 1 and 5 where 5 means it has met all the requirements of the metric and 1 means it fails to meet the basics requirements of the metric. Let us start by defining each metric that we are going to use.

- **Structure:** This metric assesses the logical structure and formatting of the response. A well-structured response guarantees that the information given by the chat bot, particularly one created for telecommunications specialists, is straightforward to explore and comprehend. The structure is determined by the prompt we give to the LLM, in our case we have mentioned strict guidelines to follow. In Table 4 we show the prompt template. The structure may be evaluated by examining the usage of headers, references (which includes page number, file location and name) and a logical sequencing of ideas. A well-structured answer improves readability and user experience. Figure 18 illustrates a well-structured answer with one reference.
- **Accuracy:** Accuracy assesses the factual correctness of the information provided. For a telecommunications expert chat bot, this means verifying that the technical details, standards, and protocols mentioned in the response are correct and up-to-date. This can be measured by cross-referencing with the cited reference to check its correctness.
- **Relevance:** This metrics assesses whether the response is able to ad-

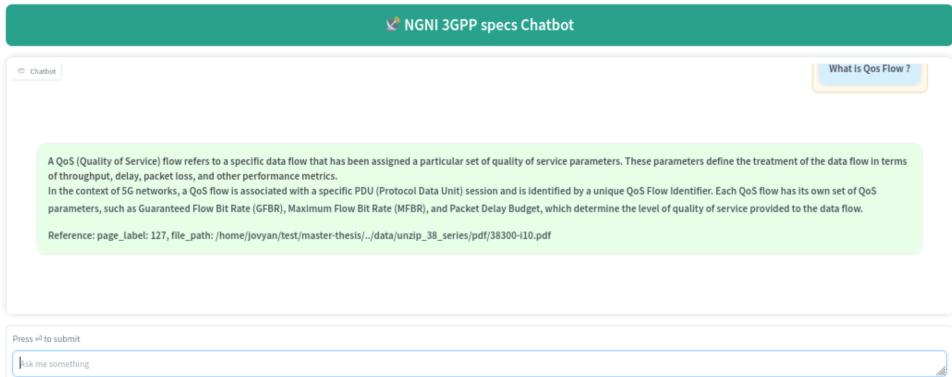


Figure 18: A good structured output with reference

dress the user query and does not reply something unrelated. For example if an user asks “Who discovered an Atom?” and the chat bot responds “An atom is the smallest unit of matter”, the response is irrelevant. Here even if the response is factually correct it did not address the user question about the discovery.

- **Redundancy:** This metric assesses whether the response contains unnecessary repetition or redundant information. This extra information can confuse the user to make a decision. So a 5 rating here would signify that the response does not contain any redundant information.
- **Completeness:** This metrics defines the response as whole, providing explanations which covers all the points. This ensures all aspects of the user’s query are covered or answered.
- **Response Time:** This metric calculates the time taken by the chat bot to answer a user query. The user interface has a timer (see Figure19) on the top right corner which sets on when we hit the send button and stops when the response is generated, this is how the response time was determined. When a response takes more than a minute to reply, it receives a rating of 1, refer Table 5 for response time for corresponding rating.



Figure 19: Timer

Rating	Response Time
1	Greater than 50 seconds
2	41 - 50 seconds
3	31 - 40 seconds
4	21 - 30 seconds
5	0 - 20 seconds

Table 5: Rating and Response Time

LLM	Embedding Model	Structure	Accuracy	Relevance	Redundancy	Completeness	Response Time	Total Score
llama3:70b	all-MiniLM-L6-v2	5	4	3.5	3.5	4	2.5	22.5
llama3:70b	all-mpnet-base-v2	5	4	3.5	3.5	4	2.5	22.5
llama3:70b	Alibaba-NLP/gte-large-en-v1.5	5	4	4	3.5	4	2.5	23
command-r35b	Alibaba-NLP/gte-large-en-v1.5	4	4	3.5	3.5	3.5	3	21.5
nous-hermes2:8x7b	Alibaba-NLP/gte-large-en-v1.5	3	3	3.5	3.5	3	3.5	19.5
mistral:7b	Alibaba-NLP/gte-large-en-v1.5	3	3	3.5	3	3.5	3.5	19.5

Table 6: Pairwise Comparison of LLMs and Embedding Models with Metrics Ratings

Table 6 presents a comparison of various combinations of LLMs and embeddings, summarizing results from testing five different LLMs paired with various embedding models, with each metric score representing the average obtained from 20 telecommunications-related questions. They were a mix of easy, medium and difficult questions. Figure 20 is an example of

the questions which were framed and tested with different ratings for the Llama3 LLM and gte-Large embedding model.

No	Questions	Structure	Accuracy	Relevance	Redundancy	Completeness	Response Time
1	What are the frequency range options defined in TS 38.101 for 5G NR?	5	5	4	5	5	4
2	What is the maximum transmission bandwidth configuration for frequency range FR1 in TS 38.104?	5	5	4	5	5	3
3	How is the physical downlink shared channel (PDSCH) mapped?	5	5	3	4	5	4
4	What are the slot formats for uplink and downlink transmission?	5	4	5	4	4	4
5	What are the measurement events for reporting in?	5	4	4	2	4	4
6	Describe the NR architecture components?	4	4	5	2	3	3
7	Which band is significant for urban 5G NR deployments?	5	4	5	5	4	3
8	What are the MAC control elements?	5	5	4	3	3	3
9	What are the RRC states and state transitions?	5	5	4	3	4	4
10	What are the main interfaces defined between gNodeB and the core network?	5	4	4	4	4	3
11	Which band is significant for urban 5G NR deployments?	5	3	4	4	5	3
12	What is a QoS Flow?	5	5	5	4	4	5
13	How does 5G NR handle dual connectivity with LTE?	4	4	5	3	3	4
14	What are the requirements for the modulation schemes supported by 5G NR User Equipment?	5	4	4	3	4	4
15	What protocol does the SMF use to control the user plane?	5	4	4	2	4	4
16	Who decides when a UE handover procedure should be initiated?	5	4	3	3	4	3
17	How many simultaneous PDU sessions can a UE have?	5	3	4	2	3	4
18	What are the requirements for the modulation schemes supported by 5G NR User Equipment?	5	3	4	3	4	3
19	How does 5G NR handle dual connectivity with LTE?	5	4	3	3	4	5
20	What are the conformance testing requirements for UE?	5	4	4	3	3	4

Figure 20: Rating of Llama3 LLM and gte-large-en-v1.5 Embedding Model Based on 20 Telecommunications Questions Across Key Metrics

As we can see in the Figure 20, which compares the performance of “Llama3” LLM and “gte-large-en-v1.5” embedding model against the metrics we defined earlier. The structure and the adherence to the prompt instructions 4 is very well maintained. As we can see the structure metrics in the Figure 20 larger model adhere to the prompt more strictly which is why in Figure 20 we observe consistent ratings of 5. When we use a LLM as big as Llama3 which has 70 billion parameters, it has bigger context window and is trained on much larger data sets than the other LLMs like mstral, command-r, which makes sure that the instructions are followed strictly. Using models with huge number of parameters as 70 billion posses its challenges, the first and foremost challenge being the need for a high end GPU. In our case our biggest model, which is Llama3, requires around 40 GB of GPU VRAM. I utilized a GPU with around 47 GB of memory, which was sufficient to load and run the model. However, incorporating the embedding model, which demands an additional 2 GB to 5 GB of GPU memory, results in substantial pressure on the GPU resources. Inspite of using a 47 GB GPU, both the models put high usage pressure on the GPU, resulting in increased response times, as illustrated in Table 6. It can be seen in Table

The screenshot shows a web-based chat interface for the NGNI 3GPP specs Chatbot. At the top, there is a green header bar with the text "NGNI 3GPP specs Chatbot". Below the header, there is a text input field with the placeholder "Ask me something" and a "Submit" button. The main content area contains a table titled "Operating bands with radiated requirements table?". The table has a light green background and contains the following data:

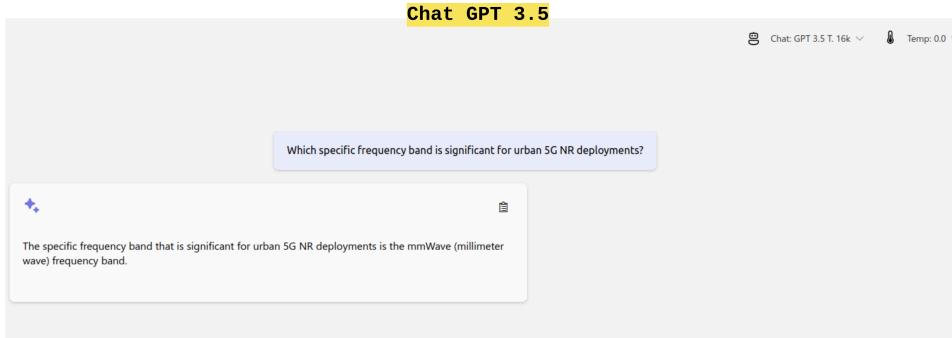
Spurious frequency range	Limit (dBm)	Measurement Bandwidth	Note
30 MHz - 1 GHz	-36	100 kHz	1
1 GHz - 18 GHz	-30	1 MHz	1

Below the table, there is a note: "Here are the operating bands with radiated requirements mentioned in the text: Table 6.7.5.2-1: Radiated Rx spurious emission limits for BS type 2-0". At the bottom of the interface, there is a message "Press ⌘ to submit".

Figure 21: Output containing tabular data

5, the rating of 2.5 indicates that it took an average of 35 to 45 seconds to respond, which is comparatively more than other model pairings. The total score indicate that the combination of “Llama3:70b” and “Alibaba-NLP/gte-large-en-v1.5” delivered the best performance, achieving the highest rating of 23. The “command-r:35b” model also performed well with rating of 21.5, while “nous-hermes2:8x7b” and “mistral:7b” lagged behind with ratings of 19.5.

As we can see in Figure 22 I have presented three responses from one common question, first one with Chat GPT 3.5, with temperature 0. Here to access chat GPT 3.5 I used generative text AI solution ‘fhgenie’, a Fraunhofer internal website based on Open AI’s GPT chat API. The second response is generated by Llama3 with 70 billion parameters, using Ollama inference engine without any RAG system integration. The last response is generated with Llama3 integrated with our RAG system named NGNI 3GPP specs chat bot also known as NGNI 3GPP chat bot. When we query the same question and study the response by all the 3 mentioned interfaces, we see that the most correct answer with expected structure is from our RAG system. While the first and second answers are also correct, we were looking for ‘n78’ in specific. Our RAG system performs well on most of the questions as seen in Figure 20 but struggles with complicated personalised questions, this is due to the fact that it may fail to access information present in the tables, which is often required for precise answers to the queries. Figure 21



Llama3 without RAG

... Which specific frequency band is significant for urban 5G NR deployments?
 For urban 5G New Radio (NR) deployments, the three key frequency bands are:

1. Sub-6 GHz: This includes frequencies from 400 MHz to 6 GHz, which are suitable for covering large areas with good penetration and providing a wider coverage area. The Sub-6 GHz band is essential for providing widespread 5G coverage in urban areas.
2. Mid-band (C-band): The mid-band frequencies range from 3.4 GHz to 5.8 GHz, and they offer a good balance between coverage and capacity. Mid-band frequencies provide faster data rates and lower latency than Sub-6 GHz but do not penetrate buildings as well. They are ideal for urban deployments where high data rates and low latency are essential.
3. Millimeter Wave (mmWave): The mmWave frequencies range from 24 GHz to 100 GHz, providing extremely high bandwidths but with limited coverage range and poor penetration. They are ideal for fixed wireless access and high-density urban deployments where very high data rates and low latency are necessary.

5G NR deployments in urban areas typically use a combination of these bands to provide extensive coverage, high capacity, and fast data rates. However, the exact frequency band used depends on the specific

Llama3 with RAG

NGNI 3GPP specs Chatbot

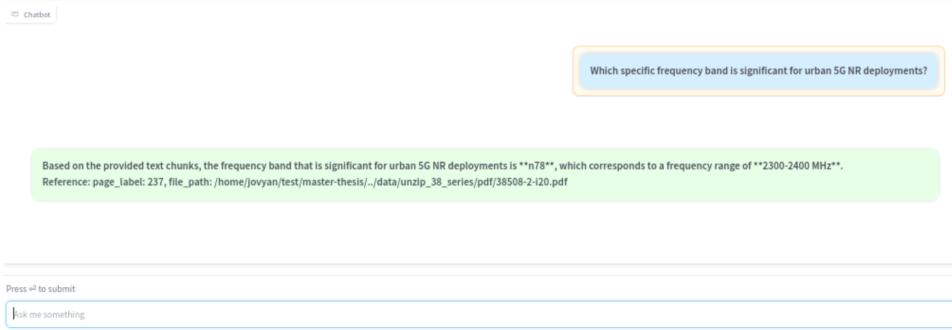


Figure 22: Different responses from different chat interfaces

illustrates that while we can occasionally view tabular data, there are a lot of differences when we compare it to other examples. For example, comparison between Figure 23 and Figure 24 shows that the tabular data has been incorrectly parsed.

5.2.3 Operating bands with radiated requirements

NTN satellite is designed to operate in the operating bands defined in Table 5.2.3-1.

Table 5.2.3-1: Satellite operating bands in FR2-NTN

Satellite operating band	Uplink (UL) operating band SAN receive / UE transmit $F_{UL,low} - F_{UL,high}$	Downlink (DL) operating band SAN transmit / UE receive $F_{DL,low} - F_{DL,high}$	Duplex mode
n512 ¹	27500 MHz - 30000 MHz	17300 MHz - 20200 MHz	FDD
n511 ²	28350 MHz - 30000 MHz	17300 MHz - 20200 MHz	FDD
n510 ³	27500 MHz - 28350 MHz	17300 MHz - 20200 MHz	FDD

NOTE 1: This band is applicable in the countries subject to CEPT ECC Decision(05)01 and ECC Decision (13)01.
 NOTE 2: This band is applicable in the USA subject to FCC 47 CFR part 25.
 NOTE 3: This band is applicable for Earth Station operations in the USA subject to FCC 47 CFR part 25. FCC rules currently do not include ESIM operations in this band (47 CFR 25.202).

Figure 23: 3GPP specification document, snippet of a table

Table 5.2.3-1: Satellite operating bands in FR2-NTN

Satellite	operatin	g band	Uplink (UL) operating band						
			SAN receive / UE transmit						
			$F_{UL,low} - F_{UL,high}$						
			Downlink (DL) operating						
			band						
			SAN transmit / UE receive						
			$F_{DL,low} - F_{DL,high}$						
			Duplex						
			mode						
n512	27500	MHz	- 30000	MHz	17300	MHz	- 20200	MHz	FDD
n511	28350	MHz	- 30000	MHz	17300	MHz	- 20200	MHz	FDD
n510	27500	MHz	- 28350	MHz	17300	MHz	- 20200	MHz	FDD

Figure 24: Representation of data in the table reproduced in Figure 23 after loading and chunking

5 Conclusion

This thesis started by addressing the challenges in extracting critical information from 3GPP specification documents in the telecommunications domain. Later we discussed about existing solutions, which are only helpful in scenarios where structured, multiple-choice queries are available. This limitation led us to explore state of the art Retrieval Augmented Generation (RAG) architectures and breaking down their components and how each contributes to improved information retrieval and response generation. We then explored why RAG is the most effective solution, examining its state of the art architectures and core components. We decided to use one of these architectures for implementation based on this analysis, with a focus on 3GPP specification documents. We compared two popular frameworks and chose LlamaIndex to create our NGNI 3GPP chatbot. A detailed overview of its components was then discussed, including retrieval, chat engine, generation, and user interface.

The evaluation section showed that the ‘llama3:70b’ model performs well across multiple metrics, especially when combined with ‘Alibaba-NLP-gte-large-en-v1.5’ (refer Table 6). This makes it a good option for applications that need responses that are highly accurate, relevant, and complete. The evaluation also indicates that while other models like ‘command-r35b’ performed well across metrics like accuracy and response time, further enhancements could be made to improve their performance metrics, especially in relevance and structure.

This solutions shows us how RAG can improve information retrieval in domains such as telecommunications and assist users navigate through technical specifications like 3GPP and suggest relevant sections with details like page number. RAG has the potential to increase productivity and accuracy for professionals dealing with complex, domain-specific knowledge by making documents like 3GPP more accessible and navigable. This could pave the way for future developments in intelligent information retrieval.

5.1 Limitations

Although providing numerous functionalities, this solution has some limitations that need to be addressed. First and foremost is incomplete parsing of tables, leading to potential loss of critical data. Comparison between Figure 23 and Figure 24 shows that there is a difference between the original table in the PDF and its parsed output. One such example is that ‘n512¹’ is misinterpreted as ‘n5121’, with the superscript ‘1’ incorrectly treated as an independent numeral alongside ‘n512’. To address this Optical Character Recognition (OCR) can be one of the solution but still need to validate it.

Another issue is the system’s limited knowledge about the other series of 3GPP technical specifications. We need our system to be able to draw references across multiple series to better understand and respond to user queries.

5.2 Future Work

To expand the scope of this thesis, we should consider scaling RAG system up to multiple 3GPP document series giving the Large Language Model (LLM) more relevant chunks of information. To address some of the shortcomings of the system, we can delve into a multi-modal approach to effectively incorporate and leverage the information contained within the tables of the source documents. Ultimately, if we could permanently implement it as an online service, that would be very helpful to telecommunications specialists.

Abbreviations

3GPP 3rd Generation Partnership Project

LLM Large Language Model

LLMs Large Language Models

AI Artificial Intelligence

NLP Natural Language Processing

NLU Natural Language Understanding

ANN Artificial Neural Networks

UI User Interface

GPT-3.5 Generative Pre-trained Transformer 3.5

HNSW Hierarchical Navigable Small World

MTEB Massive Text Embedding Benchmark

ML Machine learning

MCQ Multiple Choice Question

Llama Large Language Model Meta AI

NTN Non Terrestrial Networks

ETSI European Telecommunications Standards Institute

PDF Portable Document Format

NN Neural Network

VLLM Virtual Large Language Model

VDMS Vector Database Management System

seq2seq sequence-to-sequence

SQL Structured Query Language

RAG Retrieval Augmented Generation

JSON JavaScript Object Notation

POC Proof of Concept

Declaration of Authorship

I hereby declare that the attached Master's thesis was completed independently and without the prohibited assistance of third parties, and that no sources or assistance were used other than those listed. All passages whose content or wording originates from another publication have been marked as such. Neither this thesis nor any variant of it has previously been submitted to an examining authority or published.



Berlin, 10 September 2024

Signature of the student

References

- [3GPnd] 3GPP. *Introducing 3GPP*. <https://www.3gpp.org/about-us/introducing-3gpp>. Accessed: 2024-07-17. n.d.
- [Abi+19] Abubakar Abid et al. *Gradio: Hassle-Free Sharing and Testing of ML Models in the Wild*. 2019. arXiv: 1906.02569 [cs.LG]. URL: <https://arxiv.org/abs/1906.02569>.
- [Bor+24] Andrei-Laurentiu Bornea et al. *Telco-RAG: Navigating the Challenges of Retrieval-Augmented Language Models for Telecommunications*. 2024. arXiv: 2404.15939 [cs.IR].
- [Cho20] K. R. Chowdhary. “Natural Language Processing”. In: *Fundamentals of Artificial Intelligence*. New Delhi: Springer India, 2020, pp. 603–649. ISBN: 978-81-322-3972-7. DOI: 10.1007/978-81-322-3972-7_19. URL: https://doi.org/10.1007/978-81-322-3972-7_19.
- [ETSI24] ETSI. *3GPP Telecom Management*. 2024. URL: <https://www.etsi.org/technologies/3gpp-telecom-management?jjj=1725956228276> (visited on 09/06/2024).
- [Fan+24] Wenqi Fan et al. *A Survey on RAG Meeting LLMs: Towards Retrieval-Augmented Large Language Models*. 2024. arXiv: 2405.06211 [cs.CL]. URL: <https://arxiv.org/abs/2405.06211>.
- [Gao+24] Yunfan Gao et al. *Retrieval-Augmented Generation for Large Language Models: A Survey*. 2024. arXiv: 2312.10997 [cs.CL]. URL: <https://arxiv.org/abs/2312.10997>.
- [Had+23] M. U. Hadi et al. “A survey on large language models: applications, challenges, limitations, and practical usage”. In: (2023). DOI: 10.36227/techrxiv.23589741.v1.

- [HH24] Yizheng Huang and Jimmy Huang. *A Survey on Retrieval-Augmented Text Generation for Large Language Models*. 2024. arXiv: 2404.10981 [cs.IR]. URL: <https://arxiv.org/abs/2404.10981>.
- [HLW23] Yikun Han, Chunjiang Liu, and Pengfei Wang. *A Comprehensive Survey on Vector Database: Storage and Retrieval Technique, Challenge*. 2023. arXiv: 2310.11703 [cs.DB].
- [Kar+24] Athanasios Karapantelakis et al. *Using Large Language Models to Understand Telecom Standards*. 2024. arXiv: 2404.02929 [cs.CL].
- [KM23] Mehmet Karabacak and Konstantinos Margetis. “Embracing Large Language Models for Medical Applications: Opportunities and Challenges”. In: *Cureus* 15.5 (2023), e39305. DOI: 10.7759/cureus.39305. URL: <https://doi.org/10.7759/cureus.39305>.
- [Lan] Langchain. *Langchain Documentation*. <https://python.langchain.com/v0.2/docs/introduction/>. Accessed: 2023-10-01.
- [Lat+24] Ehsan Latif et al. *Knowledge Distillation of LLM for Automatic Scoring of Science Education Assessments*. 2024. arXiv: 2312.15842 [cs.CL]. URL: <https://arxiv.org/abs/2312.15842>.
- [Lew+21] Patrick Lewis et al. *Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks*. 2021. arXiv: 2005.11401 [cs.CL]. URL: <https://arxiv.org/abs/2005.11401>.
- [Li+24] Qingyao Li et al. *Adapting Large Language Models for Education: Foundational Capabilities, Potentials, and Challenges*. 2024. arXiv: 2401.08664 [cs.AI]. URL: <https://arxiv.org/abs/2401.08664>.
- [Lla] LlamaIndex. *LlamaIndex Documentation*. <https://docs.llamaindex.ai/en/stable/>. Accessed: 2023-10-01.

- [Maa+24] Ali Maatouk et al. *Large Language Models for Telecom: Forthcoming Impact on the Industry*. 2024. arXiv: 2308 . 06013 [cs. IT]. URL: <https://arxiv.org/abs/2308.06013>.
- [Men24] Kunal Menon. *Utilizing Open-Source AI to Navigate and Interpret Technical Documents: leveraging RAG models for enhanced analysis and solutions in product documentation*. Available at: <https://urn.fi/URN:NBN:fi:amk-2024052314858>. 2024. URL: [%5Curl %7Bhttp://www.theseus.fi/handle/10024/858250 %7D](https://hdl.handle.net/10024/858250) (visited on 08/26/2024).
- [Mue+22] Niklas Muennighoff et al. “MTEB: Massive Text Embedding Benchmark”. In: *arXiv preprint arXiv:2210.07316* (2022). DOI: 10 . 48550 / ARXIV . 2210 . 07316. URL: <https://arxiv.org/abs/2210.07316>.
- [NBG24] Rasoul Nikbakht, Mohamed Benzaghta, and Giovanni Geraci. *TSpec-LLM: An Open-source Dataset for LLM Understanding of 3GPP Specifications*. 2024. arXiv: 2406 . 01768 [cs.NI]. URL: <https://arxiv.org/abs/2406.01768>.
- [Pra23] Prasanth. *The Ultimate Guide on Chunking Strategies (RAG Part 3)*. Accessed: 2024-05-26. 2023. URL: <https://prasanth-product.medium.com/the-ultimate-guide-on-chunking-strategies-rag-part-3-82bc7040ab3c#:~:text=Naive%20Chunking, such%20as%20headers%20or%20sections.-->.
- [Tho+22] Neil C. Thompson et al. *The Computational Limits of Deep Learning*. 2022. arXiv: 2007 . 05558 [cs.LG]. URL: <https://arxiv.org/abs/2007.05558>.
- [Xie+23] Xingrui Xie et al. “A Brief Survey of Vector Databases”. In: *2023 9th International Conference on Big Data and Information Analytics (BigDIA)*. 2023, pp. 364–371. DOI: 10 . 1109/BigDIA60676 . 2023 . 10429609.

- [Xie+24] Qianqian Xie et al. *Me LLaMA: Foundation Large Language Models for Medical Applications*. 2024. arXiv: 2402 . 12749 [cs . CL] . URL: <https://arxiv.org/abs/2402.12749>.