



Escape the naivety of RAG

Guidelines linked to specific use
cases of RAG applications

Escape the naiveness of RAG

Guidelines linked to specific use cases of RAG applications

VELIMIR GRAORKOSKI



About the autor

Velimir Graorokoski has been in the software industry for almost 25 years, working on different engineering projects using various technologies. He is a tech expert and founder of the Data Science guild at Intertec. As a pioneer in the first AI/ML projects in the company, he was involved in many research activities for which he actively writes and presents. Even though he holds several AWS certifications, he still remains an open source enthusiast and Transformers (both) fan.

About this content

Origins, what is it?

Article, book or something else, it is just content after all. And as content it brings a value, hopefully a positive one. What is valuable about this and other contents is that they represent the outcome of an effort put forth by someone that had some idea, realized it and wanted to share it with the rest of the world, thus contributing to a chain of events that might lead to other ideas, with even more potential.

The origins of this content can be traced back to the spring of 2023. As the “probability” decided for me, the whole year was filled with personal disturbances for which I am still wondering how they contributed to having increased r&d activities at the same period. Maybe because I was inspired by other people that had attractive ideas that they eventually could not realize. Or perhaps the emergence of **RAG** as a concept that occupied my mind. Or is it both? Got to be the last one.

Who should read it?

I would like to describe this edition as a continuation of the *Road to Preserving Knowledge*. It doesn't mean that I will strongly push you to read that prequel, but at least some experience in **GenAI** and awareness of the related terminology can be of some help. While the article's content may seem utopian and strives towards perfection, I still encourage you to do so in order to demystify the potentials of **GenAI**, especially for knowledge representation. The *Road to Preserving Knowledge* was born in the middle of that year but **GenAI** and **RAG** were just the “strawberries on the cake”, with the cake being my past adventures in the field of knowledge representation that dated even back to my academic years (more than 2 decades ago).

Intended to give the readers an impression that although **RAG** by itself is not naive at all, there are ways that it can be improved in case it does not have sufficient **ROI**. Or rephrased: *There is no such thing as naive **RAG**, as long as it brings money. But there are ways of improvements so that it can bring even more.*

Reasons for the generation of this content

As I mentioned previously, after the spring, the summer of 2023 proved to be even busier. To be more specific, in June 2023 I started working on my first serious **RAG** application. While gaining more experience in the **RAG** field, by September I developed several **RAG** apps and containerized them to be in a production ready state. In the same month at **Intertec** we started our first project involving **RAG**.

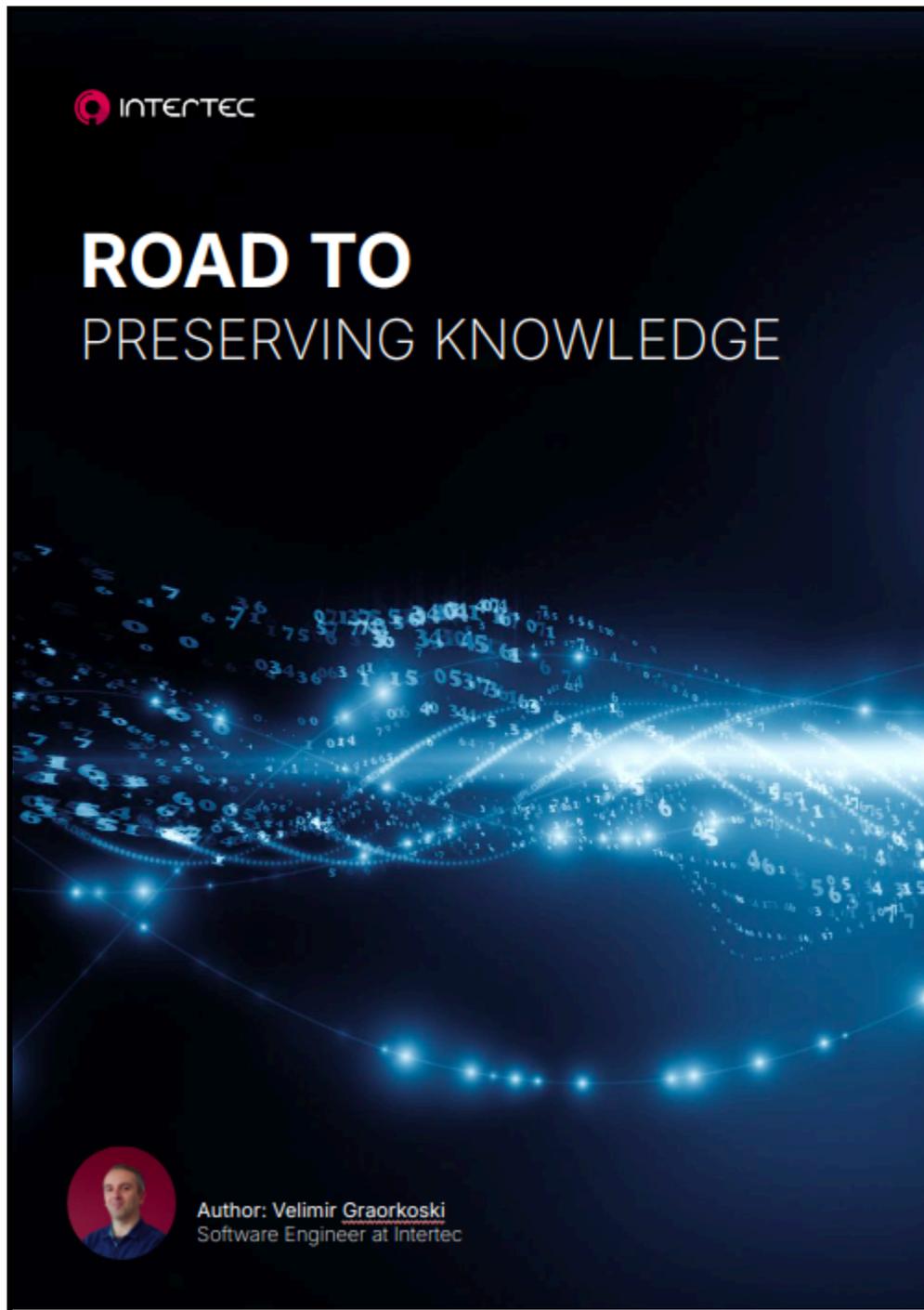


Figure 0: Cover of the *Road to Preserve Knowledge*

Impressed by the possibilities in several domains, I got inspired to finalize the *Road to Preserving Knowledge*. Afterwards I realized that I probably made a mistake because after I faced some problems on the **RAG** applications regardless whether they were at work or on a

private level, I felt that I stripped the readers from the challenges I definitely needed to overcome soon. In the continuation period because I attended local and worldwide sessions about **RAG**, but at the same time I organized workshops and urged the others in the company to participate and research, I promised I will no longer be a presenter in the next year, partially in order to make room for the other researchers. However the biggest reason for that decision was that lately I felt that I can express my thoughts much more efficiently without direct interaction with the audience simply because it lacked individuals with sufficient hands-on experience in **RAG**. I don't fully count out my participation at some events planned to be organized either by the company, by the local tech communities or other third parties, and probably I will do it after this content is published.

From technical aspects, I wanted to present my experience with **RAG** without using any fine-tuned nor trained **LLMs** as part of it. Lately the business started perceiving **RAG** as a savior that will reduce the "time to market" by avoiding long term and expensive (in a different way, not that **RAG** does not bring any costs with it) processes such as fine-tuning and training. And although in some cases it is inevitable to use models trained on domain specific data, I will try to elaborate on techniques that might still lead you to avoid custom models.

With the initial successes of the **RAG** applications I was involved in, came the failures and I worked tirelessly to overcome them. Being also aware of the limitations and some public condemnations of **RAG**, I still think its future is brighter than the darkness it comes with. At the same time I need to be realistic about its potential, especially in the business where the margin of error is much lower than in the academic or community circles, and where implementing a naive **RAG** application is not enough.

Acknowledgments

I feel that some of the people that I really want to acknowledge, do not want to reveal themselves as either direct or indirect contributors to this book. Therefore I will mention only their initials and how they got their hands messy with what is described in the *Escape the naiveness of RAG*.

Colleagues from the **DataNavi** team (A.V., M.T. and S.G.)

Friend and supporter on **Mutavchain** (A.K.)

R&D mates from the **Intertec DS** guild (A.K., M.C. and P.P.)

The other part of **InteRAG** "duo" (M.V.)

Content organization

[1. Kickoff behind RAG](#)

[1.1. The hunt for knowledge](#)

[1.2 RAG applications for observation](#)

[1.3 RAG reminder](#)

[1.3.1 RAG pipeline](#)

1.4 Naive RAG

2. Pipeline improvements

2.1 Prompts

2.1.1 Query expansion

2.1.1.1 Hypothetical questions

2.1.1.2 Multiple queries

2.1.1.3 Cross-encoder reranking

2.1.2 Chain of density

2.2 Data

2.2.1 Correction

2.2.2 Additional content

2.2.3 Content translation

2.3 Embeddings

2.3.1 Embedding adaptors

2.3.2 HyDE

2.4 Vector Store

2.4.1 Management

2.4.2 Retriever tweaks

2.4.2.1 K results

2.4.2.2 Search type

2.4.2.2.1 Similarity search score

2.4.2.2.2 MMR

2.4.2.3 Chunk size

2.4.3 Chunking

2.4.3.1 Fixed size

2.4.3.2 Sliding window

2.4.3.3 Sentence based

2.4.3.4 NSP based

2.4.3.5 Context-aware

2.4.3.6 Semantic

2.4.4 Embedded summarization

3. Application improvements

3.1 Context search

3.2 Temporal RAG

3.3 Solving the contradictions

4. Optimization

4.1 Detecting issues

4.1.1 Generator evaluation

4.1.1.1 Custom evaluation

4.1.1.2 Evaluation with MLOps

[4.2. Cost reduction](#)

[4.2.1 Contextual compression](#)

[4.2.2 Inference constraint](#)

[4.3 Latency reduction](#)

[5. Maintenance](#)

[5.1 Retrieval management](#)

[5.2 Vector Store updates](#)

[5.3 RAG scaling](#)

[5.3.1 Batching](#)

[5.3.2 Scalable retrieval](#)

[5.4 Evaluation](#)

[6. Conclusion and beyond](#)

About the author



Velimir Graorokoski has been in the software industry for almost 25 years, working on different engineering projects using various technologies. He is a tech expert and founder of the **Data Science** guild at **Intertec**. As a pioneer in the first AI/ML projects in the company, he was involved in many research activities for which he actively writes and presents. Even though he holds several AWS certifications, he still remains an open source enthusiast and Transformers (both) fan.

1. Kickoff behind RAG

I will start with slowly diving into some of the problems **RAG** is trying to address and continue with describing several of the small applications that I worked on for each particular use case.

1.1 The hunt for knowledge

Sometimes the thirst for knowledge can be correlated to personal satisfaction. At least this applies only to some humans. If we consider those humans, most of them are involved in a quiet competition of “who will first” and “who will gain the most important knowledge”. Fortunately for them, in the recent past there are relevant strategies that appeared and can be embraced. But most of those strategies required much skills on the knowledge representation part. As mentioned in the *Road to Preserving Knowledge*, it would be almost perfect if we want to find something we are looking for even in the first try or learn something quickly without passing through large amounts of concepts along the way. Let’s go through several examples.

In general it’s harder for a comic to be released than a book. But for the majority of the readers, the comics offer more pleasure than a book with the same topic. I remember I used to like the *Robotech* franchise and all the assets it produced such as the movies, series, comics and novels. When I compare the magnificent novel *End Of the Circle* by Jack McKinney, to the *Prelude of the Shadow Chronicles* by Tommy Yune covering a similar part of the saga, I remember I enjoyed reading the latter more. Probably because the comic stripped some parts from the book that were describing the scene too much, so that they had me defocused from the main goal. Other people enjoy that elaboration, but not me, a teenager who was already familiar with the races, the planets and the mecha used in the book and needed no further explanation.

Something similar is happening when we watch boring movies with interesting resolutions. If someone else that watched a movie spreads the word that only the end scene is worth watching, it will make us skip the rest of the part. For us that are “afraid” to take a risk and watch only specific scenes, we have the luxury to either skip to the desired time of the scene or search for that particular scene on **Youtube**. Here is another example: a historian, an expert of WW2, does not want to watch the whole *Battle of the Bulge* movie because he wants to avoid the scenes with battles and focus on the parts when spying and deception behind the allied lines occur; he prefers to find several short video clips from the movie that only show these actions by the German soldiers disguised as American MPs.

In my initial years as a programmer or software engineer, I relied on reading books of a particular programming language syntax or IT equipment manuals. I also had to extensively digest the codebase I inherited. Over the years I started to use web search engines and lately **GenAI** tools, even some code assistants. On some occasions I feel as if I don’t need to read the

whole documentation nor the entire code base of the project that I'm about to work on. Honestly, this is a practice I don't entirely follow nor recommend, but I realize that a lot of younger engineers that I've worked with are utilizing it to quickly overcome their deficiency in skills.

When we are talking about software development I witness that most of my colleagues prefer attending presentations about a particular topic that occurs within the company rather than spending time reading extensive articles. They want someone (one, maybe two people) to swallow a particular topic by spending some time on it and then share only the most crucial parts with the audience.

In general, wouldn't it be good if you would like to pinpoint what you want to know i.e. to distill the knowledge you want to gain and free your brain from the burden of "garbage" knowledge? Coincidence or not, up to the point of writing I've haven't heard of a good unconventional book for **RAG** with a service like **Amazon Bedrock**. Maybe because the concept or the technology are still evolving, or maybe the authors want to wait a bit more because by the time of release there might be more promising alternatives emerging. Or maybe that nobody prefers to read a huge amount of content sequentially anymore. In that case my intention of telling you what not to spend much time on can backfire.

Remembering phone numbers, mathematical formulas, command line syntax and more what used to be precious knowledge or a skill in the past is not anymore in this present. We delegate all those tasks to our artificial assistants so that we release capacity for our brains to focus on other activities.

1.2 RAG applications for observation

I picked up several **RAG** applications that I was involved in. Here are brief overviews for each of them, by chronological order.

Mutavchain is the first **RAG** app I developed as part of a wider community project for centralizing the knowledge available in written form related to the area around Galichnik settlement. The problem that it solves is to offer a valuable interface for every user that wants to converse with an entity that has access to written content that otherwise can be lost through the generations as older people pass away. It is written in **Python's Streamlit** interactive data application framework, with **LangChain** as an **LLM** framework. It utilizes the **FAISS** vector store for content stored on **Google Drive**, while **Hugging Face LLMs** are used as generators.

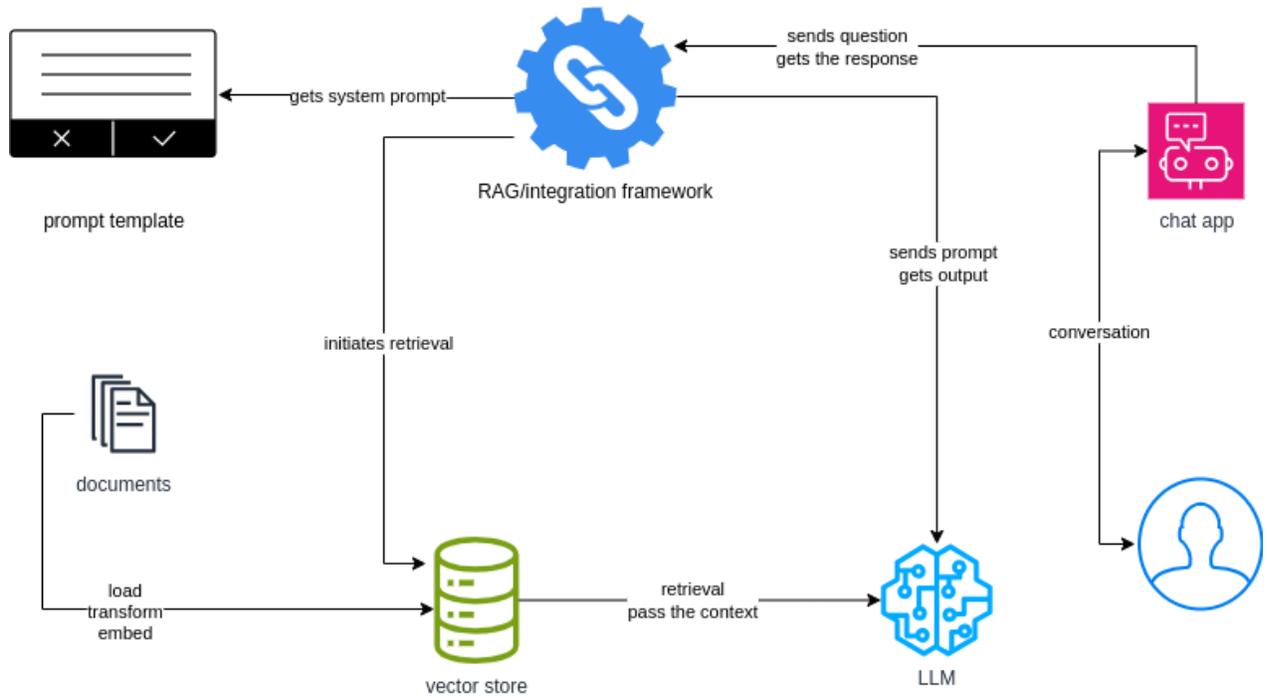


Figure 1.1 **Generalized RAG pipeline used in Mutavchain**

Intertechain is the first **RAG** application started at **Intertec**'s data science guild as part of a large r&d project called **DQA** (*Document Question Answerer*) whose functionality was to answer questions about arbitrary documents in a specific folder. Its main purpose was to ease the research of all the current possibilities of implementing a **RAG** application. It served as a basis for comparison of several technologies for constructing **RAG** pipelines. Written in **Python** it utilized **Haystack**, **LangChain** and **OpenAI** libraries, while it supported loading doc, excel and pdf files formats from **Google Drive**.

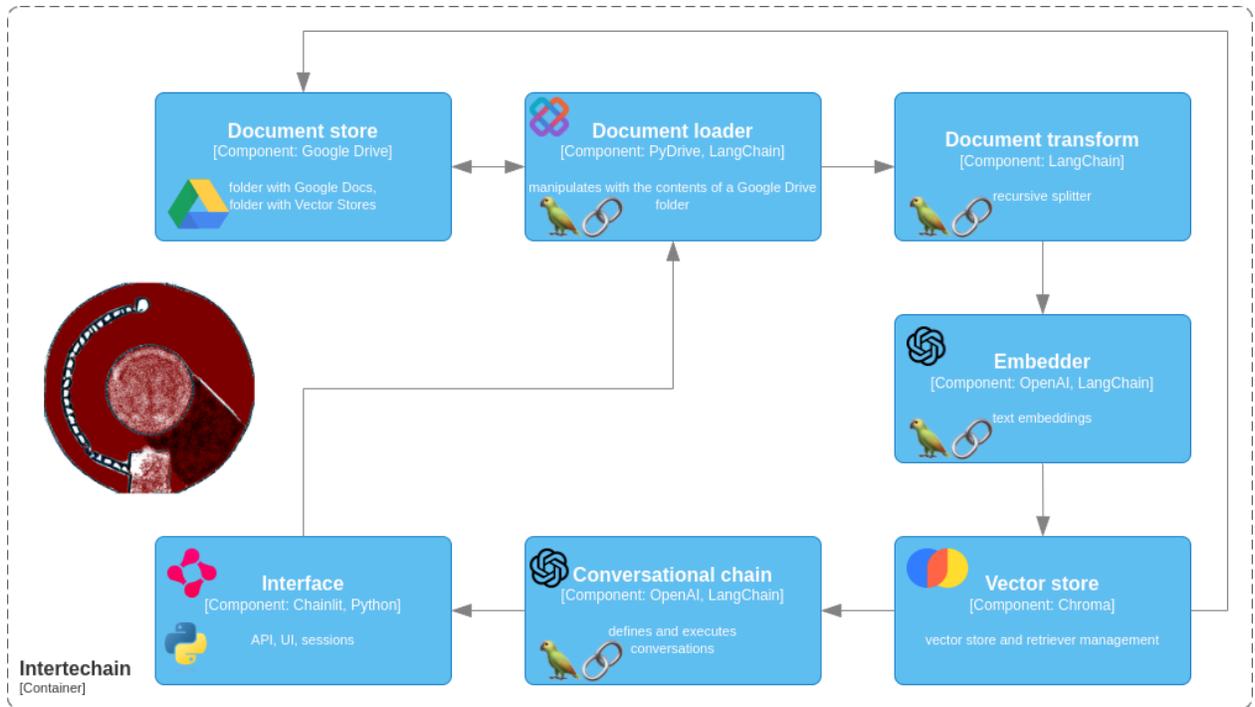


Figure 1.2 Intertechain architecture

DataNavi is the first **RAG** related project intended to satisfy the requirements of a client company whose main problem was accurate retrievals from their spreadsheet dumps on a cloud storage. The biggest advantage of using **DataNavi** is the accurate summarization, aggregation and querying of the tables within the spreadsheet documents. It basically acts as a **document question answering** tool for specific tabular data.

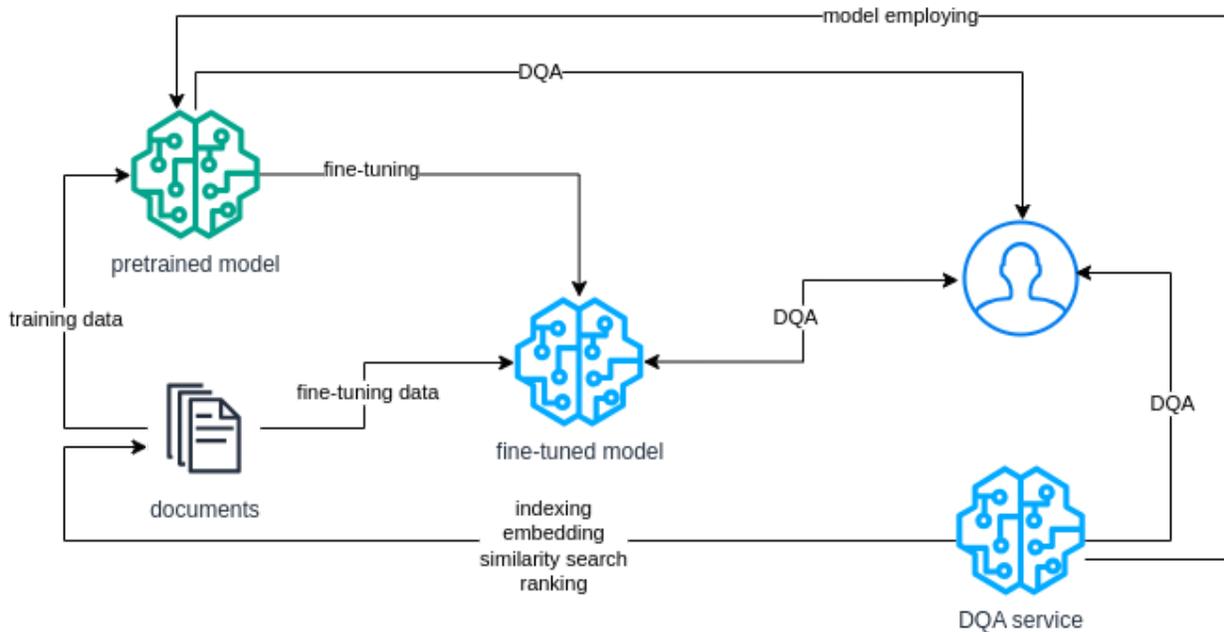


Figure 1.3 General architecture of a document question answering system

One of **Intertec**'s client companies had extensive documentation on **Confluence**. Whenever new members joined the team they needed to be personally onboarded all over again because they struggled to understand the project's documentation which consumed a significant portion of the working time of the "onboarders". At the time Confluence still relied on keyword search and other more static algorithms for retrieving relevant document pages. **ConfluenceRAG** filled that void through syncing the **Chroma** vector database with **Confluence** data and providing a suitable chatbot for conversation.

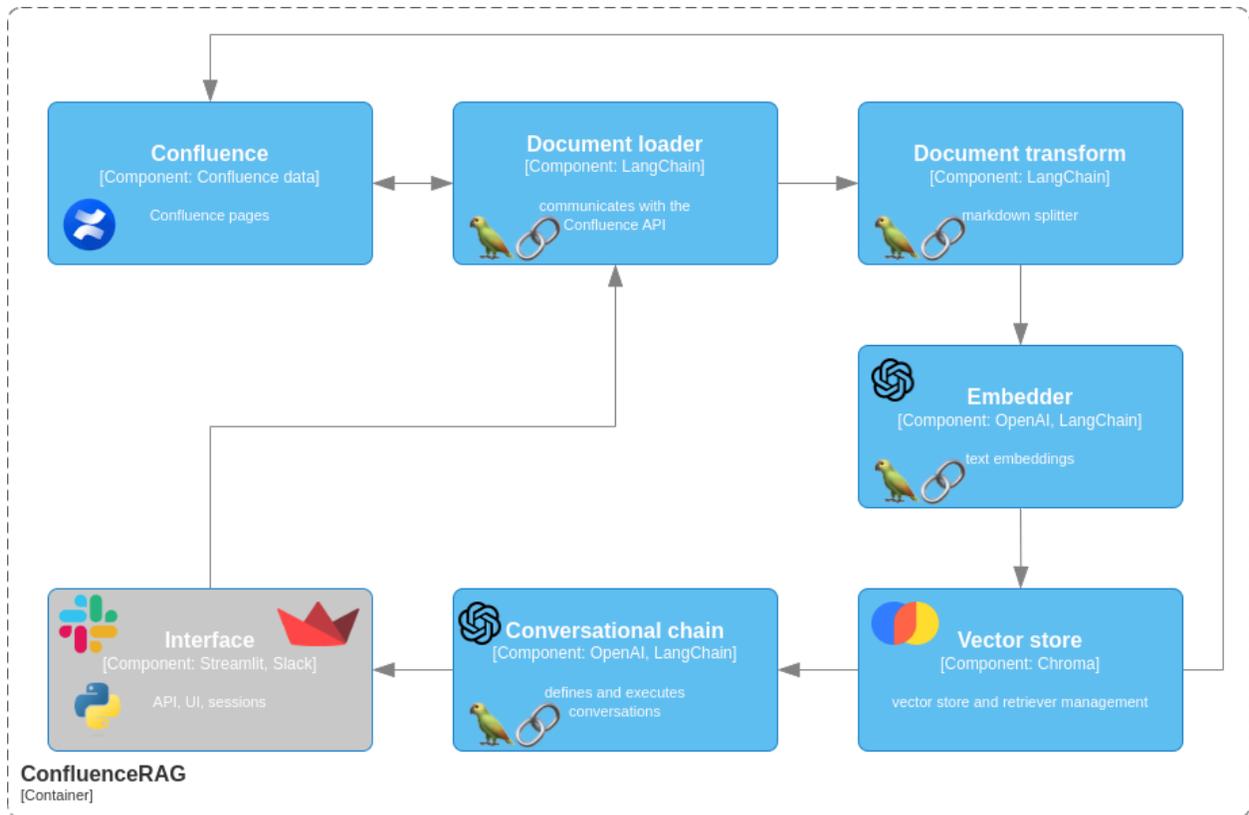


Figure 1.4 **ConfluenceRAG** architecture

InteRAG goes one step further towards satisfying external users, not just within an organization. It adds a conversational option to an existing interface. As such it loads all of the contents of a website, social media and internal documents, to create a vector store that is frequently synced afterwards. It also makes use of **MLflow** for logging the chains

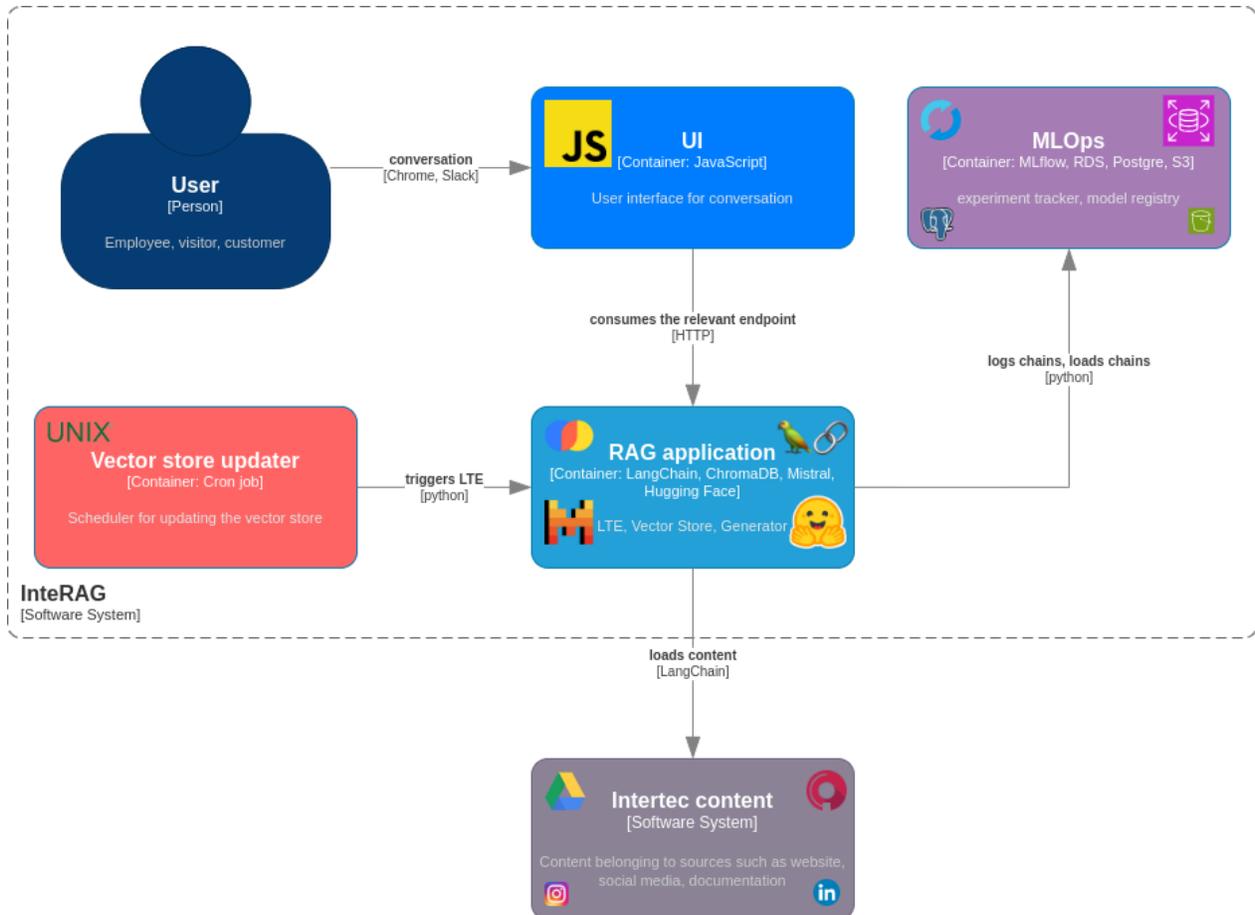


Figure 1.5 High level overview of the containers within InteRAG software system

In the upcoming sections I will not be focusing that much on the programming language nor on the framework, but more on the crucial points that create opportunities for improvements to the **RAG** pipeline resulting in increased user/learner satisfaction.

1.3 RAG reminder

To pay a little respect to those who don't have a clear knowledge of what **Retrieval Augmented Generation** is, I will try to define it as simply as possible. And for the others it will be a small reminder of what the concept of **RAG** represents. As mostly stated on the internet, **RAG** does not represent a classic framework. Nor does it represent a separate model type, as Patrick Lewis named it back in 2020 when he created the first **RAG-Sequence** model. Since then, only the name remained which resembles three separate processes: retrieval (from a data source), augmenting (applying some rules to get the desired results) and generating (output to be more human understandable). It is more like a concept or a paradigm that is conceived to mitigate some of the problems still present in **GenAI** on one side and classic **ML** on the other side.

There are two types of challenges for **GenAI LLMs** that prevent their performance. The first one is the hallucination effect which occurs whenever the model generates something that

does make sense to what the consumer is expecting as an output, but is inaccurate. The generated information may sound plausible but is factually incorrect, nonsensical, or fabricated. Over time, the most advanced **LLMs** improved to avoid hallucination at least for facts that are widely known, however the risk of it still remains. Take a look at the example on **Figure 1.6**.

What was the name of Frank Rosenblatt's and Patrick Lewis's joint paper about autonomous vehicles?

Frank Rosenblatt and Patrick Lewis co-authored a groundbreaking paper titled "The Cognitive Highway: A Neural Approach to Autonomous Navigation." In it, they explored the potential of neural networks in guiding vehicles through complex environments, envisioning a future where cars could communicate with each other and the infrastructure around them. Their ideas were so ahead of their time that they even proposed a "traffic whisperer" AI to enhance road safety!

Figure 1.6 **Example of an LLM hallucination**

Pretty weird, ha? Neither the lifespans of both scientists overlapped, nor their field of interest lay in autonomous vehicles. Not to mention that they did not write a paper with that title at all. Maybe such paper does not even exist. Do not try to reproduce the hallucination because it took me some time to make this **LLM** do so. The frequency of occurrence varies across different **LLMs** and depends on the requests as well. How does the concept of **RAG** mitigate this anomaly? The obvious answer is to make the model "know" the scope of information and direct it to generate negative answers when it has insufficient information. It means that the **LLM** queries a knowledge base, database, or search engine to retrieve relevant, factual information. This ensures that the generated content is grounded in real-world data, reducing the chance of fabricating details. Once the information is retrieved, the model uses it to construct the response, blending retrieved facts with its generative capabilities. This allows the model to stay on topic and avoid creating unsupported or fictional content. Anchoring the generation process to concrete sources, prevents the model from over-relying on its own internal knowledge, which can sometimes lead to hallucinations, especially when the data isn't current or detailed enough. Since the retrieval component (shown as part of the **vector store** on **Figure 1.7**) can access up-to-date and accurate information, the models can adapt and update in real time, correcting inaccuracies or hallucinations that might emerge from relying solely on outdated data they have been trained on.

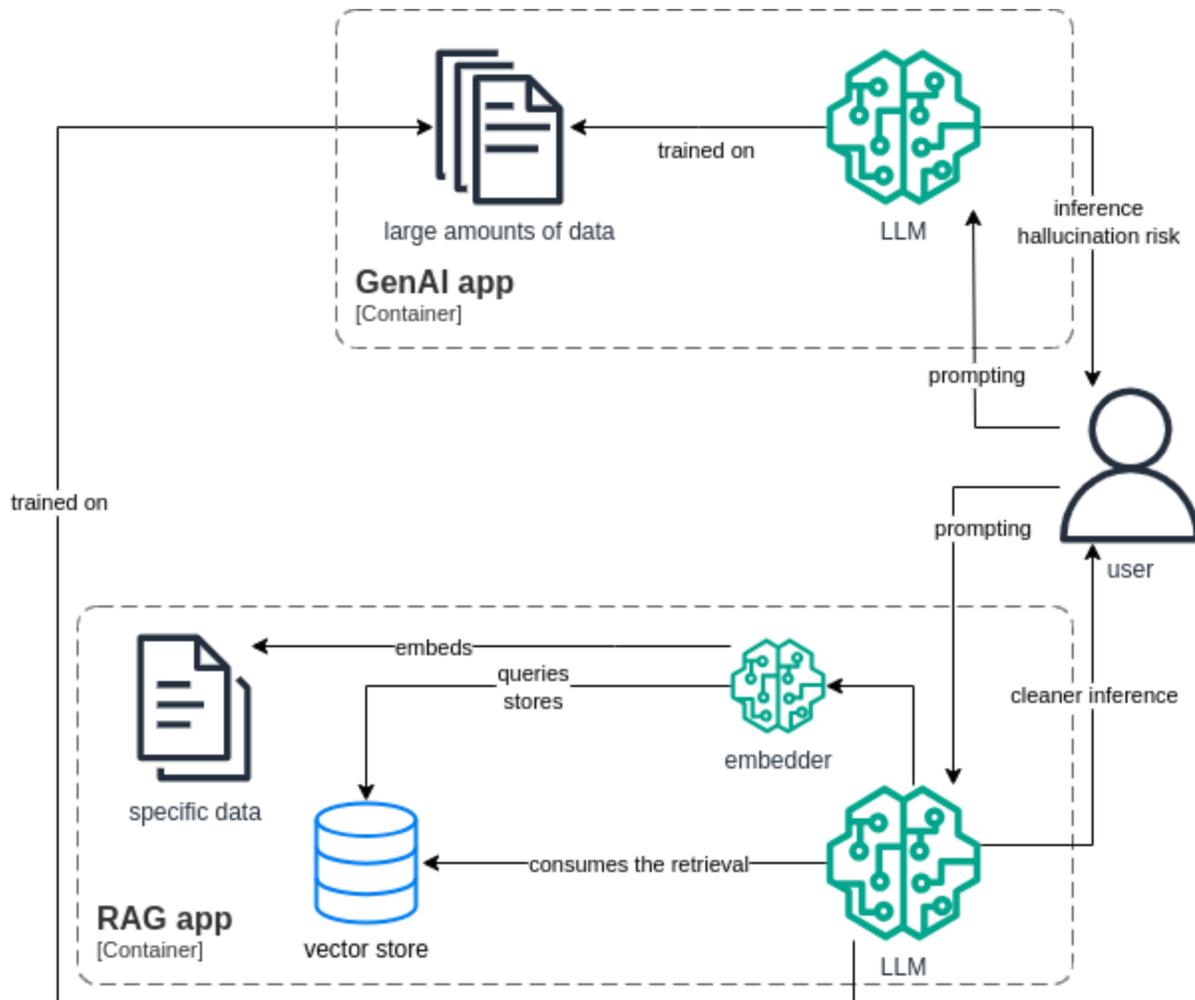


Figure 1.7 classic GenAI LLM prompting vs RAG prompting

If a user wants to prompt the **LLM** for a matter related to some recent information, in most of the cases it will not get an answer at all. This is happening because the **LLM** was not exposed to recent data while trained. Usually most of the popular **LLMs** have a knowledge cut-off period of at least 6 months to up to 1.5 years. **LLMs** as part of the **RAG** pipeline are also suffering from this flaw. But they can retrieve information from external databases, search engines, or other dynamic sources. This bypasses the limitations of their pre-trained knowledge, which has a fixed cut-off point. If the model doesn't know something due to its training data being outdated, it can pull the latest information from an external source during the generation process or in best case scenarios can prioritize it over the information coming from the trained data. By connecting to specific frequently updated knowledge bases, **RAG** allows the **LLM** to stay current with recent events, technological advances, or newly published research. This is crucial for domains where information changes rapidly, such as the use cases I explained previously.

For the data engineers running their workloads on **AWS** cloud, the purpose of **RAG** in the case of document search may seem similar to **Kendra** service. While **Kendra** focuses only on retrieval i.e. answering queries by retrieving documents or specific information, rather than

generating free-form text responses, **RAG** produces human-like natural language that can vary from detailed text to a short summary. In other words, **Kendra** also does not suffer from hallucination nor from knowledge cutoff but it is more strict in the answers and does not support searching through data from sources other than databases. **RAG** goes beyond that and adds ways of customization because of the different units it consists of, plus offering conversational functionalities.

Although **RAG** solves these two issues, I will have to warn the readers that if the **RAG** pipeline is not setup nor handled properly, the same issues of the **GenAI LLMs** will appear as we will take a look in the next sections.

1.3.1 RAG pipeline

As you noticed on some of the previous diagrams, a common **RAG** pipeline consist of the following main units (ordered starting from the consumer to the source):

- **Conversational Interface** - can be a simple **API** or a nice frontend page with all the optional functionalities for providing elegant **UX**, in both cases the intention is to provide the user an effective way of communication with the language model;
- **Generator** - the unit that generates responses, consisting of a language model (foundational **LLM** in most of the cases or a fine-tuned language model), and additional processing logic mainly dedicated to handling prompts;
- **Retriever** - fetches the relevant content by applying techniques such as ranking to ensure the most relevant information is selected;
- **LTE** unit - performs loading of the data, transforming it using chunking, cleaning and other preprocessing techniques, and finally embedding it using embedding model (referred to as **Embedder** in the majority of this content);
- **Vector Store** - specialized data structure that stores and indexes high-dimensional vector representations of data, enabling efficient similarity search and retrieval based on vector embeddings;
- **Data Source** - can be a storage, database or a web page, basically any kind of source that contains text;
- **Evaluator** - combination of human in the loop and **MLOps** techniques that assesses the accuracy and relevance of generated responses by comparing them to the retrieved documents, applied to almost every unit but most of the focus falls on the **generator** and the **LTE**.

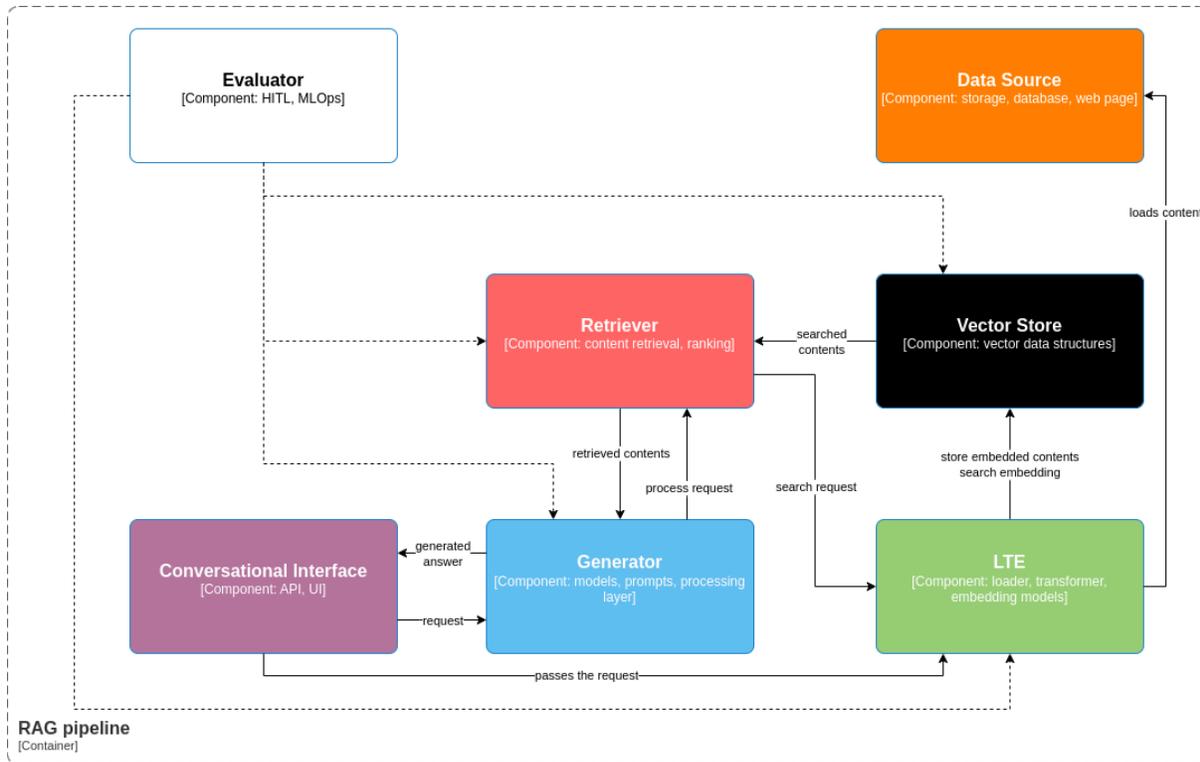


Figure 1.8 Common RAG pipeline

Because the pipeline offers flexibility up to some degree, there are variants that depend on the use case and technology. For example, sometimes the generator can also be used to adjust the search request before sending it to the **LTE** unit. There are also cases when more than one model is used for the **Generator**. We will explore some minor **RAG** variants in the next sections.

1.4 Naive RAG

Some of the **RAG** applications I previously mentioned were started in a totally naive way. It means that it was expected that constructing a **RAG** pipeline will solve the majority of the problems and future maintenance will practically be non-existent. Having all of the units in the **RAG** pipeline working straightforward i.e. in a non-optimized state can solve only trivial problems. This type of **RAG** pipeline is usually referred to as naive, baseline, traditional (even though I do not like this name because **RAG** is still a relatively new paradigm), foundational or vanilla. For our purpose I chose to stick with the first term - **Naive**.

Out of my experience I determined that **Naive RAG** applications can be suitable if the questions are asked for a content chunk that clearly states the answer. For example if the question is about a song named “Azhi Pavle” (in Macedonian „Ажи Павле“) contained in some documents from the **Data Source** unit, the output will present the song without any mistakes because there is a clear correlation between the name of the song and the lyrics as in **Listing 1.1**.

Unset

```
// Part of document contents in the Data Source:  
... Истовремено бил омилен од сите девојки и не случајно за него  
и за убавата Менда е испеана прекрасната љубовна-свадбарска песна  
со милозвучна мелодија по која е препознатлива галичката  
свадба:\r\n\r\n\r\n\r\n„Нарочуват бело Менде, порочуват(x2) како  
зноеш Ажи Павле да си доеш.(x2)\r\nДа си доеш Ажи Павле пред  
Петровден(x2) пред Петровден Ажи Павле две недели.(x2) Ако не  
доеш Ажи Павле пред Петровден(x2) Ка го наеш бело Менде  
оможено.(x2)“ ...
```

Python

```
naive_rag_qa.invoke("Како гласи песната за Ажи Павле?")
```

Unset

```
'Песната за Ажи Павле гласи:\n\n„Нарочуват бело Менде,  
порочуват(x2) како зноеш Ажи Павле да си доеш.(x2)\nДа си доеш  
Ажи Павле пред Петровден(x2) пред Петровден Ажи Павле две  
недели.(x2) Ако не доеш Ажи Павле пред Петровден(x2) Ка го наеш  
бело Менде оможено.(x2)“ '
```

Listing 1.1 Example of a question handled by Naive RAG app

On the other hand, here are some examples of questions where **Naive RAG** applications failed to provide a proper answer:

- What is the year of publication of the book/article? - hard to determine the year below the title or from the book metadata because it is usually not preceded by “title: name of the title”, although in English it sometimes works fine;
- What are the titles/headings inside the book? - for the similar reasons as the previous question;
- Who is the reviewer of the book? - usually the review is in a separate section of a book content where a specific format is used and arbitrary question may not work;
- Who are the writers of the books? - for the similar reasons as the previous question.

In the next core chapters I will show you how I coped with similar problems and also what possible interventions you should be careful with or not spend more effort on.

2. Pipeline improvements

For showcasing what problems can/cannot be improved and which strategies can/cannot be applied in the **RAG** pipeline, I will take two of the previously described **RAG** applications which are similar in implementation but for different purposes.

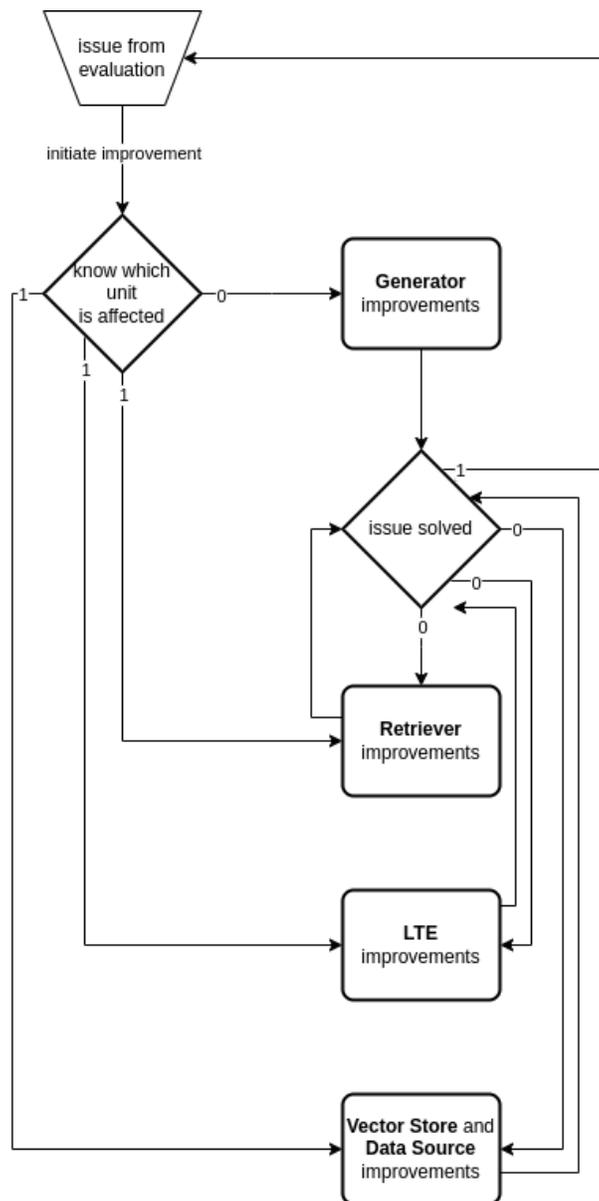


Figure 2.1: Hierarchical flow of RAG improvements

The first choice is **Mutavchain** simply because it is required to support conversation and content in Macedonian language. I will not translate the examples from Macedonian to English because although not a frequently spoken language, it is anyway well covered by the most popular translation tools.

The second one is **InteRAG** whose showcase in the next sections is backed up by its technical implementation. It encompasses **HuggingFace**, **LangServe** and **MLflow** libraries with multiple data sources support.

How/where can you adjust your **RAG** pipeline in order to improve it? One logical answer will be “by taking opportunities on every piece (unit) that forms it”. However, improvements on one **RAG** unit may impact other units but may also not be sufficient. Although the most frequent recommendation that can be found on the internet is to begin with improvements of the **Retriever** and the **LTE** units, I will advise to start with the most obvious or those that usually require the least effort, such as shown on **Figure 2.1**. In the upcoming sections I will not order the improvements according to the hierarchy in the pipeline though, because I want to also give some emphasis on the strategies that are not always working and are not recommended to try them out as first.

2.1 Prompts

Prompt improvements for the **Generator** in the **RAG** pipeline mainly strive to enhance the quality and coherence of the generated responses intended for the user. In the case of prompt improvements, standard **PE (prompt engineering)** techniques apply but I will not entirely focus on those. Instead I will include the part related to the querying of the **Retriever** and the **Embedder** which is an example of how an **LLM** can be used to form better search queries.

Python

```
def setup_generator(retriever):
    prompt = hub.pull("rlm/rag-prompt")
    llm = ChatOpenAI(model_name="gpt-4o-mini", temperature=0)

    rag_chain = ({
        "context": retriever | format_docs,
        "question": RunnablePassthrough()
    } | prompt | llm | StrOutputParser())

    return rag_chain
```

Python

```
def setup_generator(retriever):
```

```

llm = ChatOpenAI(model_name="gpt-4o-mini", temperature=0)
template = """Use the following pieces of context to answer the question
at the end. If you don't know the answer, just say that you don't know,
don't try to make up an answer. Use three sentences maximum and keep the
answer as concise as possible.

{context}

Question: {question}

Helpful Answer: """

custom_rag_prompt = PromptTemplate.from_template(template)

rag_chain = ({
    "context": retriever | format_docs,
    "question": RunnablePassthrough()
} | custom_rag_prompt | llm | StrOutputParser())

return rag_chain

```

Listing 2.1: **Standard vs custom prompt declaration in Mutavchain**

One basic example is given on **Listing 2.1** where the standard prompt is replaced with a custom one, in order to adjust the output. One of the crucial parts is that the generated answer must consist of no longer than three sentences. One special kind of technique is **CRP (Contextual Retrieval Process)** that uses custom prompts for every chunk that is retrieved from the embedded content, but I will not focus on it since it depends on factors related to the organization of the data and the embeddings as well, although one example is available on **Mutavchain's** repository.

2.1.1 Query Expansion

Query Expansion in a **RAG** refers to the process of enriching the original user query to improve the relevance and diversity of the retrieved documents. The goal is to ensure that the **Retriever** identifies more matching documents which are therefore used by the generator. In this case the **Generator** has a wider pool of information from which it can generate the final answer. I will present two such techniques that also include some negative examples as a warning that there is no “silver bullet” to “kill” the problem of formulating an efficient query.

2.1.1.1 Hypothetical Questions

Also called **Generated**, the strategy of **Hypothetical Questions** aims to make the **LLM** generate questions for each chunk. After the embedding, these generated questions are transformed into vectors and stored in the **Vector Store**. Therefore during retrieval of text chunks from the documents, the relevance of the user's question against the **Hypothetical**

Question embeddings can be matched. This enhances the search quality, as the semantic similarity between the query and the **Hypothetical Question** is higher than with an original text chunk from the document. The strategy is shown in **Figure 2.2**.

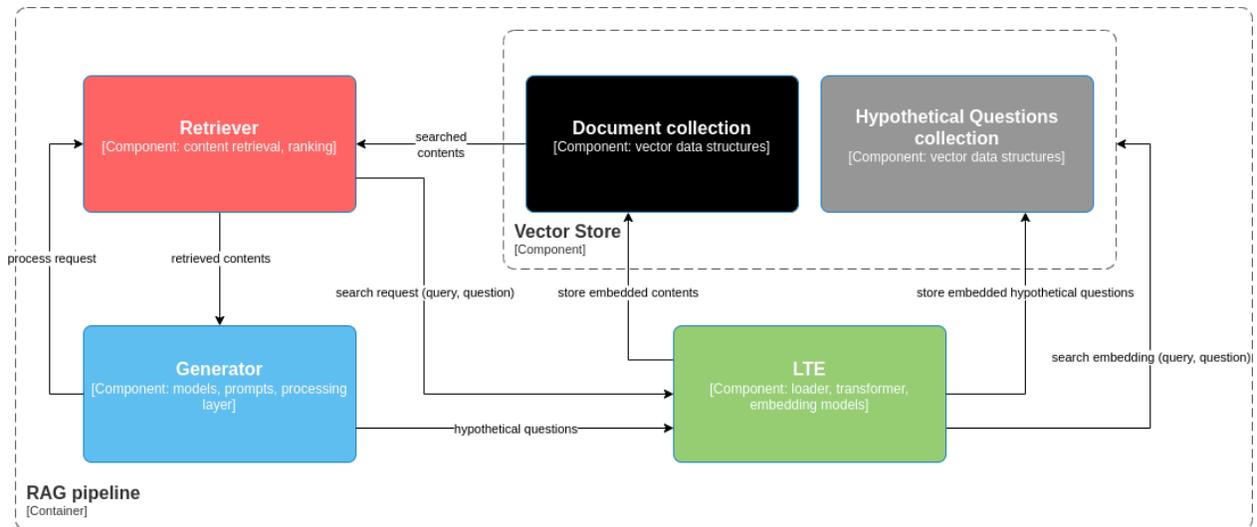


Figure 2.2: Hypothetical Questions strategy as part of the RAG pipeline

The strategy of **Hypothetical Questions** may seem weird but it helps in some cases. **Listing 2.2**. shows how it can be initialized in **Mutavchain** on a set of documents with **LangChain**. The initialization contains definition of specific functions as well controlling the number of generated questions using an adequate instruction in the prompt.

```

Python
functions = [{
    "name": "hypothetical_questions",
    "description": "Generate hypothetical questions",
    "parameters": {
        "type": "object",
        "properties": {
            "questions": {
                "type": "array",
                "items": {
                    "type": "string"
                }
            }
        }
    },
    "required": [
        "questions"
    ]
}]
}
  
```

```

Python
chain = (
    {"doc": lambda x: x.page_content}
    | ChatPromptTemplate.from_template(
        "Генерирај листа од 3 хипотетички прашања што следниов документ ќе ги
искористи за одговор:\n\n{doc}"
    )
    | ChatOpenAI(max_retries=0, model="gpt-4o-mini").bind(
        functions=functions, function_call={"name": "hypothetical_questions"}
    )
    | JsonKeyOutputFunctionsParser(key_name="questions")
)

```

```

Python
chain.invoke(documents[0])

```

```

Unset
['Што го поттикнало авторот да напише книга за Галичник?', 'Како 33 „Бистра“
придонела за развојот на Галичник?', 'Кои мерки ги предложил авторот за
ревитализација на овчарството во Македонија?']

```

Listing 2.2: Initialization of Hypothetical Questions in LangChain

Besides generating three questions for a single document, the previous listing also shows the potential to cover the whole initial state of the **Vector Store** with embeddings for the extracted **Hypothetical Questions**. **Listing 2.3** provides how such a **Vector Store** is created for **Mutavchain** with an example how to search for the questions for a given query.

```

Python
hypothetical_questions = chain.batch(documents, {"max_concurrency": 5})

vectorstore = Chroma(
    collection_name="hypo-questions",
    embedding_function=OpenAIEmbeddings()
)

store = InMemoryByteStore()
id_key = "doc_id"

retriever = MultiVectorRetriever(
    vectorstore=vectorstore,
    byte_store=store,

```

```

        id_key=id_key)

doc_ids = [str(uuid.uuid4()) for _ in documents]

question_docs = []
for i, question_list in enumerate(hypothetical_questions):
    question_docs.extend([
        Document(
            page_content=s,
            metadata={
                id_key: doc_ids[i]
            }
        ) for s in question_list
    ])

retriever.vectorstore.add_documents(question_docs)
retriever.docstore.mset(list(zip(doc_ids, documents)))

```

Python

```

sub_docs = vectorstore.similarity_search("Стојче Росоклија")
sub_docs

```

Unset

```

[Document(page_content="Koj e значењето на орото 'Тешкото' и каде е родено?",
metadata={'doc_id': 'e8130eba-3a02-4a29-b84a-ed9f3c3f66d9'}),
Document(page_content='Како Галичник го инспирирал Стојче Дојчиноски-Росоклија
за неговите дела?', metadata={'doc_id':
'94ac2882-2e8a-4433-a2c1-ff56d8ee7f5b'}), Document(page_content='Што е значаен
придонес на Стојче Дојчиноски-Росоклија за Галичник?', metadata={'doc_id':
'e8273dd2-d75a-45ee-850a-81fc9d619211'})]

```

Listing 2.3: Vector Store of Hypothetical Questions and its usage

All is not bright in this strategy though, as I experienced. One downside of this approach is that for some languages (in this case Macedonian), the hypothetical questions may not be well formulated as it can be noticed in the output of **Listing 2.3** where the first question is grammatically incorrect because it starts with “Koj” instead of “Koe”. A consequence of this can be that the similarity search does not retrieve many relevant documents which means that the **Vector Store** allocates space for data that might not be useful for the **RAG** pipeline. For the previous example, there is not much negative impact because the formulation error is small, hence the question can still prove to be useful.

Over the course of implementing this strategy, I encountered some technical limitations as well, that I hope by the time of the release of this book will be resolved:

- **Hugging Face Hub** does not support it, workarounds are there but personally I do not find them useful enough and will require a lot of effort to include them in the codebase;
- **FAISS** does not support it out of the box as other **Vector Stores** such as **Chroma** do, workarounds for implementation exist but are not recommended.

2.1.1.2 Multiple Queries

You don't need to imagine the following scenario because it usually happens. A user asks the application a question and gets an unsatisfying answer. Then he tries to rephrase the question and repeats the paraphrasing until it gets the desired answer. All these steps of creating rephased queries can be automated, by leaving only the formulation of the first query the user's commitment.

Python

```
multiquery_retriever = MultiQueryRetriever.from_llm(retriever=retriever,
llm=llm)
multiquery_retriever
```

Unset

```
MultiQueryRetriever(retriever=VectorStoreRetriever(tags=['Chroma',
'HuggingFaceEmbeddings'],
vectorstore=<langchain_community.vectorstores.chroma.Chroma object at
0x77ff2282ca10>), llm_chain=PromptTemplate(input_variables=['question'],
template='You are an AI language model assistant. Your task is \n
to generate 3 different versions of the given user \n
question to retrieve relevant documents from a vector
database. \n
By generating multiple perspectives on the user question, \n
your goal is to help the user overcome some of the limitations \n
of distance-based similarity search. Provide these alternative \n
questions separated by newlines. Original question:
{question}')
|
HuggingFaceHub(client=<InferenceClient(model='mistralai/Mistral-7B-Instruct-v0.
1', timeout=None)>, repo_id='mistralai/Mistral-7B-Instruct-v0.1',
task='text-generation', model_kwargs={'max_length': 800, 'temperature': 0.2})
| LineListOutputParser())
```

Listing 2.4: **Default MultiQueryRetriever in InterAG**

LangChain library offers this out of the box through its class **MultiQueryRetriever**. As you can see from the example on **Listing 2.4**, the default chain for the **from_llm** function, uses

a built-in prompt that generates 3 different versions of the original query before pinging the **Retriever**. You can replace the default prompt with a custom one as well, but the key point remains the class unless you want to customize the **Multiple Queries** strategy out of the **Retriever** restricting it only to the **Generator**. Take a look at **Listing 2.5** where based on the prompt you can guess that the additional two queries that were automatically generated are closely related to “*What is the current number of clients?*” and “*Note two names of Intertec’s clients!*”.

```
Python
custom_prompt = PromptTemplate(
    input_variables=["question"],
    template="""You are an AI language model assistant. Your task is to
generate two
different versions of the given user question to retrieve relevant
documents from a vector database. By generating multiple perspectives on the
user question, your goal is to help the user overcome some of the limitations
of the distance-based similarity search. Provide these alternative questions
separated by newlines. If the original question is not related to a quantity of
the topic, generate such a question. If the original question is not related to
specific names or examples, generate such a question. Otherwise generate an
arbitrary question but different from the original one.
Original question: {question}""",
)

llm = HuggingFaceHub(repo_id=llm_repo, model_kwargs={"max_length": 800,
"temperature": 0.2})

llm_chain = custom_prompt | llm | StrOutputParser()

multiquery_retriever = MultiQueryRetriever(
    retriever=vectordb.as_retriever(), llm_chain=llm_chain, parser_key="lines"
)
```

```
Python
multiquery_retriever.invoke("Which companies are clients of Intertec?")
```

```
Unset
Intertec is a tech partner to both enterprises and promising startups. The
current number of clients is 15. Notable examples are BMW and Atolls.
```

Listing 2.5: Using custom prompt for MultiQueryRetriever in InteRAG

To sum it up, by issuing multiple variants of a query, the system can retrieve documents or passages that may not have been found with a single query. Different phrasing or rewording of the query can yield a broader range of relevant information.

2.1.1.3 Cross-encoder reranking

One intervention that can be combined with expansion of the queries is the document reranking of the retrieved documents. It comes as a consequence of having an enlarged pool of retrieved documents due to the increased number of queries. Even though on paper, this technique can improve the order of the retrieved documents, it does not always work properly or at least it represents a challenge.

```
Python
original_query = "Кои личности Веле први ги спомнува како најзаслужни за
Галичник?"
generated_queries = [
    "Кои личности имаат најголема заслуга за Галичник?",
    "Дали постојат личности што се најзаслужни за Галичник?",
    "Кои се најзаслужните личности во Галичник?",
]

queries = [original_query] + generated_queries

results = chroma_collection.query(query_texts=queries, n_results=10,
include=['documents', 'embeddings'])
retrieved_documents = results['documents']

unique_documents = set()
for documents in retrieved_documents:
    for document in documents:
        unique_documents.add(document)
unique_documents = list(unique_documents)

cross_encoder = CrossEncoder('cross-encoder/ms-marco-MiniLM-L-6-v2')

pairs = []
for doc in unique_documents:
    pairs.append([original_query, doc])

scores = cross_encoder.predict(pairs)

print("New Ordering (worse than the original):")
for o in np.argsort(scores)[::-1]:
    print(o)
```

```
Unset
New Ordering:
18
3
10
0
6
11
9
15
19
5
13
1
16
4
7
2
17
14
12
8
```

Listing 2.6: **Cross-encoder reranking in Mutavchain**

First, a model (shallow logic alternative is not recommended), needs to be invoked which has to operate on a lower level, on the **Vector Store** itself, instead of the chain, to retrieve a document array with improved order. Second, this reranking often backfires and produces worse ranking than the original, as shown in **Listing 2.6** which was confirmed by the **HITL**. Therefore I would not strongly recommend it, especially when dealing with content in low frequently spoken languages.

2.1.2 Chain of Density

The way of progressively refining the queries and the retrieval results to improve the quality of the generated responses is known as **Chain of Density (COD)** technique in **PE** that comes useful in some **RAG** applications. This strategy involves multiple steps where the system retrieves information, generates intermediate responses, and iteratively refines those responses for a more accurate and detailed final output that in most of the cases acts as a summary.

```
Python
def setup_generator(_retriever):
    llm = ChatOpenAI(model_name="gpt-4o-mini", temperature=0)
```

```
template = """Article: {context}
```

You will generate increasingly concise, entity-dense summaries of the above article.

Repeat the following 2 steps 5 times.

Step 1. Identify 1-3 informative entities (";" delimited) from the article which are missing from the previously generated summary.

Step 2. Write a new, denser summary of identical length which covers every entity and detail from the previous summary plus the missing entities.

A missing entity is:

- relevant to the main story,
- specific yet concise (5 words or fewer),
- novel (not in the previous summary),
- faithful (present in the article),
- anywhere (can be located anywhere in the article).

Guidelines:

- The first summary should be long (4-5 sentences, ~80 words) yet highly non-specific, containing little information beyond the entities marked as missing. Use overly verbose language and fillers (e.g., "this article discusses") to reach ~80 words.
- Make every word count: rewrite the previous summary to improve flow and make space for additional entities.
- Make space with fusion, compression, and removal of uninformative phrases like "the article discusses".
- The summaries should become highly dense and concise yet self-contained, i.e., easily understood without the article.
- Missing entities can appear anywhere in the new summary.
- Never drop entities from the previous summary. If space cannot be made, add fewer new entities.

Remember, use the exact same number of words for each summary.

Answer in JSON. The JSON should be a list (length 5) of dictionaries whose keys are "Missing_Entities" and "Denser_Summary".

If you don't know the answer, just say that you don't know, don't try to make up an answer.

Use three sentences maximum and keep the answer as concise as possible.

```
Question: {question}
```

```
Helpful Answer:""
```

```
custom_rag_prompt = PromptTemplate.from_template(template)
```

```
rag_chain = (  
    {"context": retriever | format_docs, "question": RunnablePassthrough()}  
    | custom_rag_prompt
```

```

        | llm
        | StrOutputParser()
    )
    return rag_chain

cod_chain = setup_generator(retriever)

```

```

Python
cod_chain.invoke("Како гласи песната за Ажи Павле?")

```

```

Unset
„Нарочува бело менде“ е стара народна македонска песна испеана од легендарниот
Александар Сариески.

```

Listing 2.7: **COD** example in Mutavchain

One example of **COD** prompt can be found on the **LangSmith** hub (under the path *lawwu/chain_of_density*) which includes instructions for dense summarization as seen in **Listing 2.7**. The listing also shows why this strategy is not suitable when you need a precise answer to a question where the loss of information cannot be tolerated at all, like in this case the missing lyrics of the song. You can always engineer a custom **COD** prompt but it usually takes a lot of time and there is a tradeoff risk between the costs (increased because of the size of the prompt) and the loss of information which correlates to the level of needed summarization.

2.2 Data

Over time as your **RAG** application improves, there is a chance it will “hit a wall” regarding the possibilities of further improving the architecture, the used software and the parameters. After all the possible options have been exploited, it will all come down to the quality of the data again. Improving the data consists of many techniques that are also used in **NLP (Natural Language Processing)** in general, starting from manual interventions, to automatic transformation such as cleaning, summarizing or tagging. I will mention only several of them which are important for **RAG**.

2.2.1 In-place correction

In-place correction of the content as part of the **Data Source** ensures that errors or outdated information in the documents are corrected before being loaded into the **Vector Store**. This implies that the inaccurate information will not be present in the retrieved context, and the

likelihood of generating incorrect responses will disappear in the **Generator**, improving the overall trustworthiness of the system.

Python

```
qa_chain.invoke("Колку посетители имал музејот на галичкото културно наследство во Галичник?")
```

Unset

```
'Музејот на галичкото културно наследство во Галичник имал околу 7.000 посетители.'
```

Python

```
# in-place correction to the relevant information in the document for Galichnik's cultural heritage
...
# recreating the Retriever and the Generator
retriever = setup_openai_embeddings_and_chroma_retriever(documents)
qa_chain = setup_generator(retriever)
```

Python

```
qa_chain.invoke("Колку посетители имал музејот на галичкото културно наследство во Галичник?")
```

Unset

```
'Музејот на галичкото културно наследство во Галичник во јули 2006 година имал околу 2.000 посетители.'
```

Python

```
qa_chain.invoke("Колку посетители имал музејот на галичкото културно наследство во Галичник во јули во 2006 година?")
```

Unset

```
'Околу 2.000 посетители.'
```

Listing 2.8: Example of In-place correction in Mutavchain

The example on **Listing 2.8** shows how a small intervention in the original document can lead to accurate response which is valuable to the user. First there is a broad question about the number of visitors without specifying the time period. The first answer gives a specific number that creates an impression that it is about the total number of visits since the museum's opening. However the information about the 7000 overall visits does not exist at all. In the museum's original documents there were no mentions of number 7000 related to the visits at all. Instead, the museum noted the number of visits per month. After the first correction which states that during July, 2006, the number of visits was 2000, the next answer is far more accurate, including the period as well even if the question does not provide a specific period of time.

This is probably the most promising technique of improvement of the **RAG** pipeline, but at the same time not easy to handle because of its manual nature and the necessity to update its embedding in the **Vector Store** or in some cases even recreate the whole **Vector Store**. That will be discussed in one of the next sections dedicated to the **Vector Store** improvements.

2.2.2 Additional content

There is often a necessity for new content to be added within the existing or as a separate document in the **Data Source**. This is because **RAG** applications rely on external knowledge sources, which means that over time the content can become limited if the users are seeking answers to their new questions. By adding new content regularly, the **RAG** pipeline can keep the knowledge up-to-date. Making sure you widen the **Data Source** is not a difficult task on the condition that access is given for the **Data Source** and/or there is a proper interface available.

```
Python
qa_chain.invoke("Кoj бил д-р Васил Гривчев?")
```

```
Unset
'Не знам кој бил д-р Васил Гривчев'
```

```
Python
# Vector Store updated with additional content about Vasil Grivchev
...
qa_chain.invoke("Кoj бил д-р Васил Гривчев?")
```

Unset

'Д-р Васил Гривчев-Гривчески бил истакнат македонски економист, правник и општествено-политички работник. Тој имал богата кариера, вклучувајќи позиции како член на Извршниот совет на СР Македонија, сојузен секретар за надворешна трговија и судија на Уставниот суд на Југославија. Исто така, бил професор по финансиско право и автор на бројни трудови.'

Listing 2.9: Improvement of the response with adding new content after a previously negative answer in Mutavchain

Listing 2.9 shows an example from **Mutavchain** where additional content was included in the dataset after there were multiple requests from users about a certain person that yielded negative answers. If you include even more content related to that person, the **RAG** application will become more diverse, covering information about the person from different angles for similar questions. And for more detailed questions it will add more depth to the responses.

The strategy of **Additional content** is particularly important in **RAG** applications covering customer support, research, adaptive learning, and knowledge management, where having multiple perspectives can enhance decision-making, increase engagement, and improve the overall user experience.

2.2.3 Content translation

One of the key purposes of **Mutavchain** was to access contents from the Data Source that has documents written only in Macedonian and to provide answers to Macedonian as well. Because the **LLMs** used in the application support English as the most widely used language in the world as well, the communication with the user can be done in English with similar performance as provided in **Listing 2.10**. The answer in English differs a little bit from the one in Macedonian but both answers are correct and are based on the same piece of content from the **Data Source**.

Python

```
qa_chain.invoke("Кoj бил Арсениј?")
```

Unset

"Арсениј од Галичник бил архимандрит и иницијатор на изградбата на црквата Св. Петка во Галичник во 1809 година. Тој се одолжил на своето родно село како еден од донаторите за изградбата на црковниот храм. Во тој период, манастирот Св. Јован Бигорски бил една од најзначајните духовни институции во Македонија."

Python

```
qa_chain.invoke("Who was Arsenij?")
```

Unset

```
"Arsenij was a builder, archimandrite, and hegumen of the Bigorski Monastery. He contributed to the construction and artistic works of the monastery, including zografy, iconography, and woodcarving. Arsenij's leadership and vision greatly influenced the cultural identity and growth of the Bigorski Monastery complex."
```

Listing 2.10: **Communication in Macedonian and in English for the same question in Mutavchain**

Even though there are many **LLMs** that successfully handle Macedonian and other low-resource languages, the reality is that English is still the dominant language in many knowledge sources. Hence unless a model is trained specifically for Macedonian, most of the models will always perform much better in English regardless of whether they are used as part of the **Generator** or in the **LTE** unit in the **RAG** pipeline.

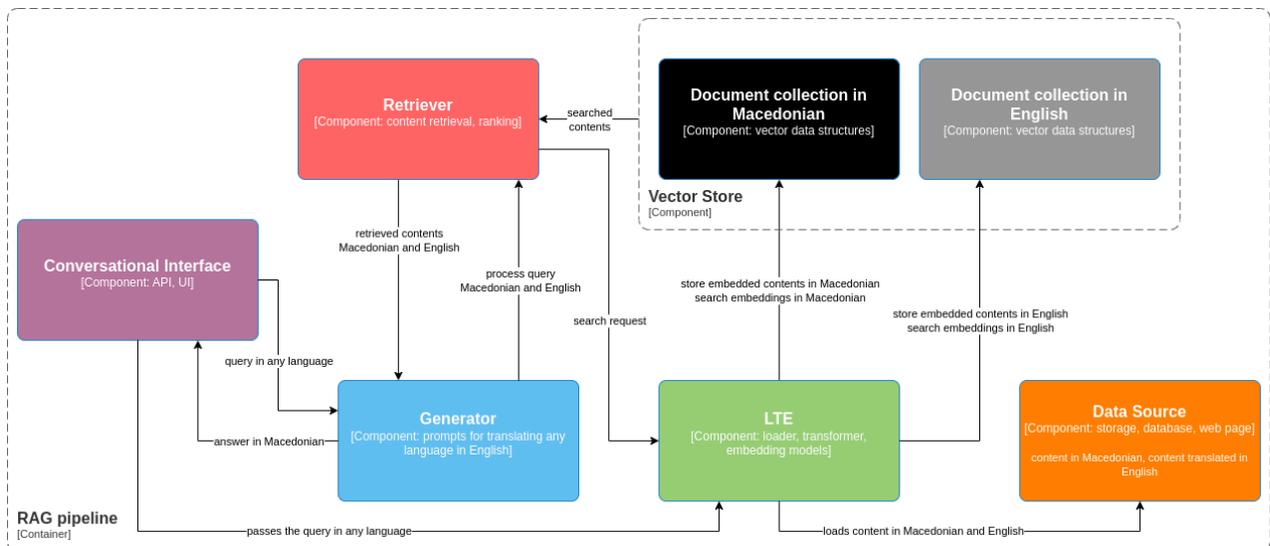


Figure 2.3: **RAG pipeline supporting only Macedonian for communication but improved with content translated in English**

However, it is not enough for the **LLM** only to support the user's request in English and provide answers in the same language. The **RAG** application still needs to be able to embed and retrieve relevant content from the **Vector Store**. This implies that the **Vector Store** should contain embeddings from the English content as well because most of the embedders are more optimized for English. By translating the content to English, you enable the **RAG** application to

access, index, and retrieve a much larger corpus of global information to combine it with the one from the **Data Source**. Combined with changes in the prompt to translate the query to English, as well as to translate the answer back to the original language, the RAG pipeline can achieve overall improvement even if it does not seem to the user that English is supported at all. Take a look at the diagram in **Figure 2.3**, for an idea how to construct the **RAG** pipeline for applications with use cases similar to **Mutavchain**.

2.3 Embeddings

Improvement of handling the embeddings as part of the **LTE** unit, positively impacts the retrieval process in the **RAG** pipeline. Unlike improvements in the **Generator**, the strategies considering the **Embedder** are generally harder to be applied. Usual techniques of fine-tuning an **Embedder** or other customizations on the model itself, I keen to avoid in the next sections for the reasons I stated in this book's introduction part. Instead I will focus on the techniques that can potentially make good use of pretrained models, especially on the risks they bring with them.

2.3.1 Embedding adaptors

If during the **HITL** evaluation or user's feedback it is detected that the **Embedder** can do a better job for some parts of the content, you don't need to switch the embedding model, nor to fully fine-tune/train it. Instead you can adjust it by adding another layer of transformation to the **LTE** unit in the form of **Embedding adaptors** (or in some literature often mentioned as adaptors), kind of an addition to the **Embedder** in the form of another model. Although theoretically the **Embedding adaptors** are extending the **Embedder**, in practice they are attached to the **Vector Store** functions as shown in the next listing.

Python

```
def load_chroma(document, collection_name, embedding_function):
    chunks = _chunk_texts(document)

    chroma_client = chromadb.Client()
    chroma_collection = chroma_client.get_or_create_collection(
        name=collection_name, embedding_function=embedding_function
    )

    ids = [str(i) for i in range(len(chunks))]

    chroma_collection.add(ids=ids, documents=chunks)

    return chroma_collection
```

Python

```
def project_embeddings(embeddings, umap_transform):
    umap_embeddings = np.empty((len(embeddings), 2))
    for i, embedding in enumerate(tqdm(embeddings)):
        umap_embeddings[i] = umap_transform.transform([embedding])
    return umap_embeddings
```

Python

```
embedding_function = SentenceTransformerEmbeddingFunction()

chroma_collection = load_chroma(
    document=formatted_documents,
    collection_name="mutavchain_embedding_adaptor",
    embedding_function=embedding_function,
)

embeddings = chroma_collection.get(include=['embeddings'])['embeddings']
umap_transform = umap.UMAP(random_state=0, transform_seed=0).fit(embeddings)
projected_dataset_embeddings = project_embeddings(embeddings, umap_transform)
```

Python

```
def generate_queries(model="gpt-4o-mini"):
    messages = [
        {
            "role": "system",
            "content": "You are a semantic search engine for article and book content in Macedonian language. You provide users answers based on their questions. "
            "Suggest 10 to 15 short questions that are important to ask when searching for content. "
            "Do not output any compound questions (questions with multiple sentences or conjunctions). "
            "Output each question on a separate line divided by a newline."
        },
    ]

    response = openai_client.chat.completions.create(
        model=model,
        messages=messages,
    )
    content = response.choices[0].message.content
    content = content.split("\n")
```

```
return content
```

Python

```
generated_queries = generate_queries()  
for query in generated_queries:  
    print(query)
```

Unset

Која е темата на содржината?

Кoj е авторот на текстот?

Која е главната идеја на делото?

Каде е објавен текстот?

Кога е објавена содржината?

Кои клучни зборови го опишуваат текстот?

Која е целната публика на авторот?

Дали има релевантни цитати или извори?

Кои се основните аргументи на авторот?

Дали текстот содржи визуелни помагала?

Кои се заклучоците на текстот?

Дали има контрааргументи?

Кои се користените методи на истражување?

Какви примери или студии се користени?

Каков тон користи авторот?

Python

```
results = chroma_collection.query(query_texts=generated_queries, n_results=10,
include=['documents', 'embeddings']) retrieved_documents = results['documents']
```

Python

```
def evaluate_results(query, statement, model="gpt-4o-mini"):
    messages = [
        {
            "role": "system",
            "content": "You are a semantic search engine for article and book
content in Macedonian language. You provide users answers based on their
questions. "
            "For the given query, evaluate whether the following satement is
relevant."
            "Output only 'yes' or 'no'.",
        },
        {"role": "user", "content": f"Query: {query}, Statement: {statement}"},
    ]

    response = openai_client.chat.completions.create(
        model=model, messages=messages, max_tokens=1
    )
    content = response.choices[0].message.content
    if content == "yes":
        return 1
    return -1
```

Python

```
retrieved_embeddings = results["embeddings"]
query_embeddings = embedding_function(generated_queries)

adapter_query_embeddings = []
adapter_doc_embeddings = []
adapter_labels = []

for q, query in enumerate(tqdm(generated_queries)):
    for d, document in enumerate(retrieved_documents[q]):
        adapter_query_embeddings.append(query_embeddings[q])
        adapter_doc_embeddings.append(retrieved_embeddings[q][d])
        adapter_labels.append(evaluate_results(query, document))

adapter_query_embeddings = torch.Tensor(np.array(adapter_query_embeddings))
```

```

adapter_doc_embeddings = torch.Tensor(np.array(adapter_doc_embeddings))
adapter_labels = torch.Tensor(np.expand_dims(np.array(adapter_labels), 1))

dataset = torch.utils.data.TensorDataset(
    adapter_query_embeddings, adapter_doc_embeddings, adapter_labels
)

```

Python

```

def model(query_embedding, document_embedding, adaptor_matrix):
    updated_query_embedding = torch.matmul(adaptor_matrix, query_embedding)
    return torch.cosine_similarity(updated_query_embedding, document_embedding,
dim=0)

```

Python

```

def mse_loss(query_embedding, document_embedding, adaptor_matrix, label):
    return torch.nn.MSELoss()(
        model(query_embedding, document_embedding, adaptor_matrix), label
    )

```

Python

```

mat_size = len(adapter_query_embeddings[0])
adaptor_matrix = torch.randn(mat_size, mat_size, requires_grad=True)

min_loss = float("inf")
best_matrix = None

for epoch in tqdm(range(100)):
    for query_embedding, document_embedding, label in dataset:
        loss = mse_loss(query_embedding, document_embedding, adaptor_matrix,
label)

        if loss < min_loss:
            min_loss = loss
            best_matrix = adaptor_matrix.clone().detach().numpy()

    loss.backward()
    with torch.no_grad():
        adaptor_matrix -= 0.01 * adaptor_matrix.grad
        adaptor_matrix.grad.zero_()

```

Python

```
query_embeddings = embedding_function(generated_queries)
adapted_query_embeddings = np.matmul(best_matrix,
np.array(query_embeddings).T).T

projected_query_embeddings = project_embeddings(query_embeddings,
umap_transform)
projected_adapted_query_embeddings =
project_embeddings(adapted_query_embeddings, umap_transform)
```

Python

```
query_embeddings = embedding_function(generated_queries)
adapted_query_embeddings = np.matmul(best_matrix,
np.array(query_embeddings).T).T

projected_query_embeddings = project_embeddings(query_embeddings,
umap_transform)
projected_adapted_query_embeddings =
project_embeddings(adapted_query_embeddings, umap_transform)
```

Python

```
def generate_answer(query, model="gpt-4o-mini",
chroma_collection=chroma_collection):
    results = chroma_collection.query(query_texts=query, n_results=30,
include=['documents', 'embeddings'])
    retrieved_documents = results['documents'][0]

    context = "\n\n".join(retrieved_documents)

    messages = [
        {
            "role": "system",
            "content": "Use the following pieces of context to answer the
question at the end. If you don't know the answer, just say that you don't
know, don't try to make up an answer. Use three sentences maximum and keep the
answer as concise as possible.",
        },
        {
            "role": "user",
            "content": f"{context}\n\nQuestion: {query}\n\nHelpful Answer:",
        }
    ]
```

```
    },  
  ]  
  
  response = openai_client.chat.completions.create(  
    model=model,  
    messages=messages,  
  )  
  content = response.choices[0].message.content  
  return content
```

Python

```
output = generate_answer(  
  query="Кои ликови Веле први ги спомнува како најзаслужни за развојот на  
Галичник?",  
  chroma_collection=chroma_collection  
)  
output
```

Unset

Веле ги спомнува како најзаслужни за Галичник: Василески Алексов Георги, Веноски Душан-Дуко, Владически Влатко, Владически Спасе, Влческа Сирма, Влчески Насте, Влчески Рафаил, Гегоски Ѓорѓе и Гегоски Иван.

Python

```
chroma_collection_2 = load_chroma(  
  document=formatted_documents,  
  collection_name="mutavchain_embedding_adaptor",  
  embedding_function=projected_adapted_query_embeddings  
)
```

Python

```
output = generate_answer(  
  query="Кои ликови Веле први ги спомнува како најзаслужни за развојот на  
Галичник?",  
  chroma_collection=chroma_collection_2  
)  
output
```

Unset

Не знам

Listing 2.11: Embedding adaptors using generated questions as an intermediary step for their training in Mutavchain

It is worth trying this technique even if it doesn't work on several tries. However it may consume a significant amount of time and you should be prepared for such a tradeoff. I used this technique in **Mutavchain** where it did not produce satisfying results and it did not pay off for trying more, which does not mean that you should neglect it for other use cases where you are not time/cost pressured. The negative outcomes can often resemble the hallucination effect as shown in **Listing 2.11**. It can be noticed that in order to implement **Embedding adaptors**, the embedding function (in this case **SentenceTransformers**) should be supported by the **Vector Store** (in this case **ChromaDB**), which is not always the case for other combinations of embeddings and vector stores. In the example in the listing it is clear that the adaptors broke the query search because it re-embed the relevant documents and others cannot be found that will help in answering the question. Not even manipulating with the **n_results** argument could improve the matter, except that it avoided providing a misleading answer by generating *"He знам (I don't know)"*. If there was an improved embedding model that is capable of effectively embedding all the domain specific text in the data source of **Mutavchain** it would probably have performed well.

2.3.2 HyDE

Although hypothetical in its nature as well as the **Hypothetical Questions** strategy for improving the **RAG** pipeline on prompt level, the **HyDE (Hypothetical Document Embeddings)** actually does the opposite. It uses prompting an **LLM** to create a hypothetical answer that does not exist in the **Data Source** which is afterwards embedded and used for the retrieval.

Python

```
system = """You are an expert for giving correct answer to questions about books and articles written in Macedonian."""
prompt = ChatPromptTemplate.from_messages(
    [
        ("system", system),
        ("human", "{question}"),
    ]
)
llm = ChatOpenAI(model="gpt-4o-mini", temperature=0)
qa_no_context = prompt | llm | StrOutputParser()
```

Python

```
answer = qa_no_context.invoke({"question": "Што напишал Славко Брезоски?"})
```

Unset

Славко Брезоски е македонски писател и поет. Тој е познат по своите дела во кои често се обработуваат теми од секојдневниот живот, љубовта и природата. Некои од неговите познати дела вклучуваат поетски збирки и раскази. За конкретни наслови на неговите дела, би било најдобро да се консултирате со библиотека или книжарница која има македонска литература.

Python

```
hyde_chain = RunnablePassthrough.assign(hypothetical_document=qa_no_context)
```

Python

```
hyde_chain.invoke({"question": "Кoj бил д-р Васил Гривчев?"})
```

Unset

```
{  
    "question": "Кoj бил д-р Васил Гривчев?",  
    "hypothetical_document": "Д-р Васил Гривчев бил истакнат македонски лекар и општественик. Роден е во 1877 година во Велес, Македонија. Тој е познат по својата активност во борбата за ослободување на Македонија од османлиската власт и за неговата улога во македонското револуционерно движење. Д-р Гривчев бил член на Внатрешната македонска револуционерна организација (ВМРО) и учествувал во Илинденското востание во 1903 година. По востанието, тој продолжил да работи како лекар и да се залага за подобрување на здравствените услови во Македонија."  
}
```

Python

```
prompt_template = """Use the following pieces of context to answer the question at the end.
```

```
If you don't know the answer, just say that you don't know, don't try to make up an answer.
```

Use three sentences maximum and keep the answer as concise as possible.

Question: {question}

Helpful Answer:""

```
prompt = PromptTemplate(input_variables=["question"], template=prompt_template)
llm_chain_hyde = LLMChain(llm=llm, prompt=prompt)
```

Python

```
hyde_embeddings = HypotheticalDocumentEmbedder(
    llm_chain=llm_chain_hyde, base_embeddings=OpenAIEmbeddings()
)
text_splitter = RecursiveCharacterTextSplitter(
    chunk_size=1000, chunk_overlap=200, add_start_index=True
)
splits = text_splitter.split_documents(documents)
chroma_store = Chroma.from_documents(documents=splits,
embedding=hyde_embeddings)
```

Python

```
search_result = chroma_store.similarity_search("Кoj бил д-р Васил Гривчев?")
search_result
```

Unset

```
[
  Document(
    page_content="многу тешки и се наметнува потреба за итни мерки за да се подобри состојбата.",
    metadata={
      "start_index": 156692,
      "title": "Galichka Mozaichna Soznajba",
      "when": "2024-06-16T13:14:04.006Z",
    },
  ),
  Document(
    page_content="шестотините златни наполеони. Тој само потврдил, дека доколку дојде до враќање на неговиот имот, посебно земјоделските посеци, односно ако има денационализација земјиштето ќе им го препише на Миле и Петре
```

```

Желчески. За оваа печалбарска судбина-невидена кражба постојат над стотици
страници документи, докази и пишани материјали. Но што фајде од тоа.",
  metadata={
    "start_index": 73513,
    "title": "Galichka Mozaichna Soznajba",
    "when": "2024-06-16T13:14:04.006Z",
  },
),
]

```

Listing 2.12: **Example of HyDE applied in Mutavchain**

As defined, the strategy feels pretty simple to implement, but there are some risks as well because the hypothetical document needs to be related and embedded accurately. The example on **Listing 2.12** is valid, but the results are not encouraging because the retrieved document chunks do not contain information related to the query. Therefore some time needs to be spent on generating a good enough hypothetical document or applying additional techniques such as generating multiple hypothetical documents.

2.4 Vector Store

Besides the choosing and maintaining a more performant type/library, improvements on the **Vector Store** side are usually coupled with the **Retriever** and mainly refer to the tweaks of the **Retriever's** arguments and the chunking strategy, as shown on **Figure 2.4**.

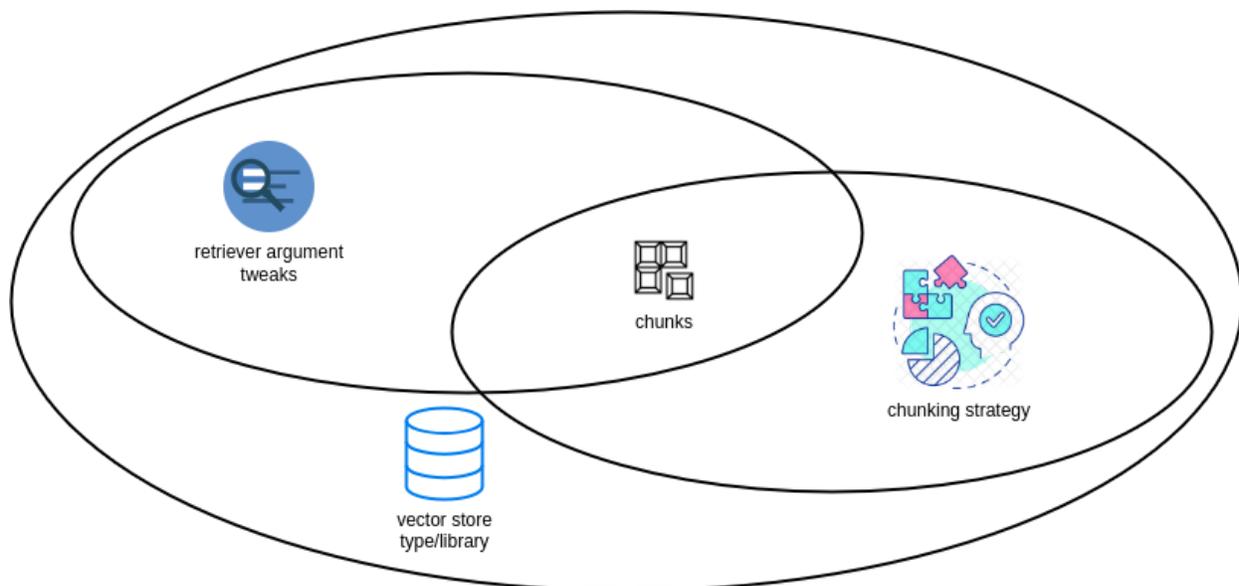


Figure 2.4: **Three aspects of Vector Store improvements**

There are however different application requirements that alter the direction of the improvements of the **Vector Store**. For instance, the ideal **Vector Store** for a **RAG** pipeline that supports multiple languages is the one that can handle multilingual embeddings and cross-lingual document retrieval. In the case of **Mutavchain**, if a question is asked in English but the relevant documents are primarily in Macedonian, a well-optimized vector store will facilitate retrieval of the Macedonian documents thus impacting high-quality, cross-lingual retrieval. However, proper management of the content in the **Vector Store**, which can include timely synchronization of content in one language to the one in another, is a precondition for making other improvements. Therefore I will start with some suggestions on how to properly manage a **Vector Store**.

2.4.1 Management

If you use a cloud service such as **Amazon's Bedrock** that provides management of the **Vector Store** you can skip this section. Still I believe that the majority of the researchers want to start by exploring the possibilities of an **OSS Vector Store** or one that has to be maintained by someone. Managing a **Vector Store** may seem straightforward but it is not just updating the versions and refactoring. It is more about how to update the content and when.

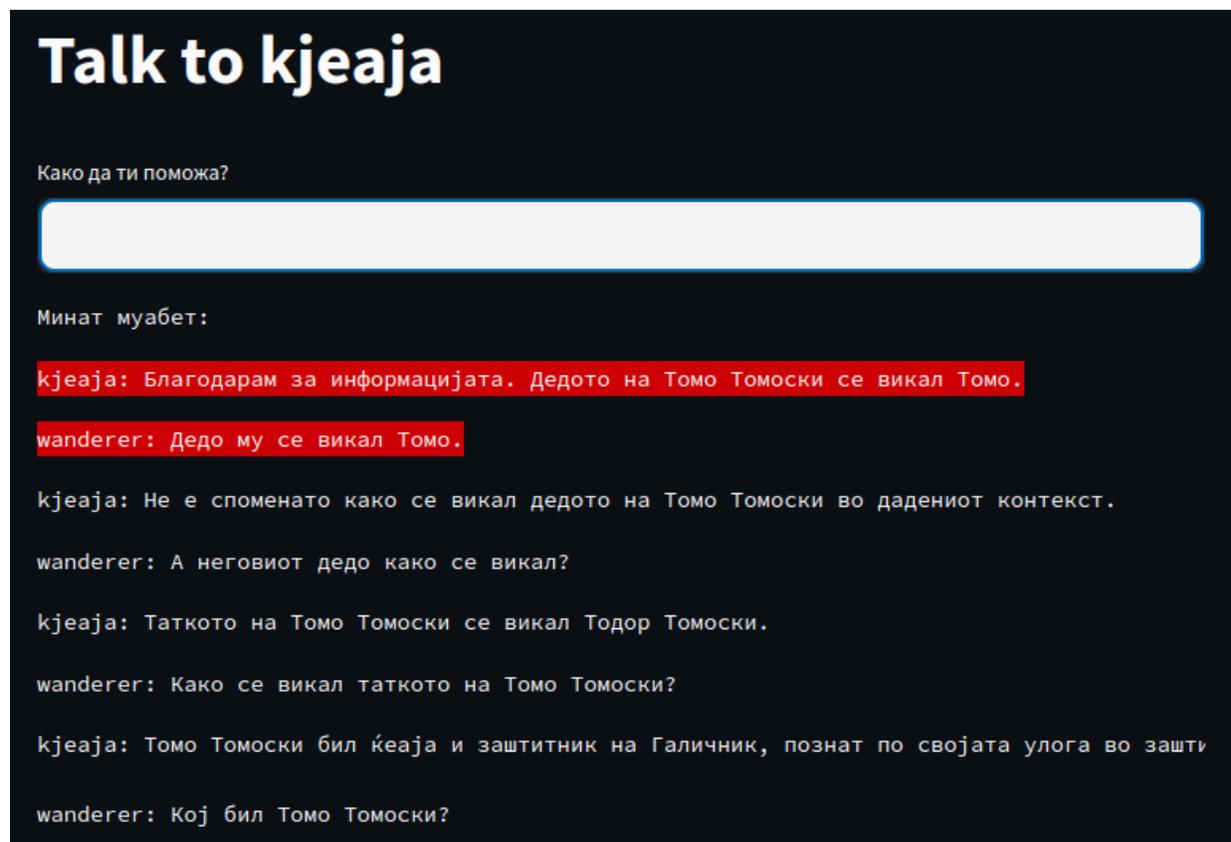


Figure 2.5: Part of conversation in Mutavchain where potential new content is provided by the user

Since the **Vector Store** consists of embeddings of the content coming from the **Data Source**, it is necessary to sync it as much as possible whenever a change in the **Data Source**'s content occurs. Therefore there are two ways of doing that: updating an existing **Vector Store** or recreating the **Vector Store** from scratch.

Whenever a **Data Source** is updated with new data and/or old data is removed, by default the standard **RAG** pipeline does not propagate that change to the **Vector Store**. In order to stay relevant, there are two options that can be used, one of which is very risky i.e. to update the **Vector Store** when the user is providing the missing information. In such a case, as shown in **Figure 2.5**, the user may provide wrong information which the **LLM** takes for granted. This risk can be mitigated with a separate interface, combined with providing access only for relevant users before an approval step is conducted. Error handling of nonexistent embeddings whenever a content chunk needs to be replaced also needs to be introduced.

The other option is to make the user initiate updating the **Vector Store** whenever it encounters no answer, but not by providing the new content. Instead the idea is to initiate an update process coming from a potentially relevant document in the **Data Source** that is not yet part of the **Vector Store**. This means that the list of document names or other metadata need to be updated and the **RAG** application needs to access it in order to choose a relevant document that will be added to the **Vector Store** as shown in one part of the diagram in **Figure 2.6**.

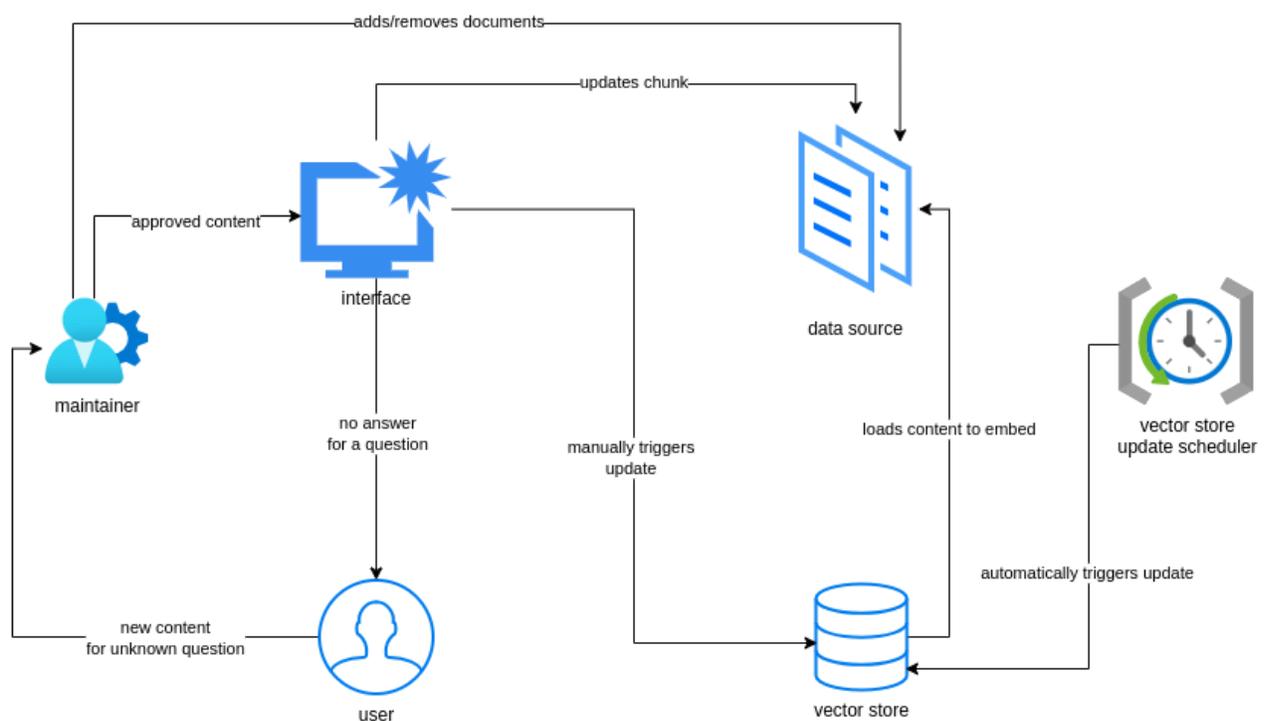


Figure 2.6: **Possible improvement options as part of the Vector Store management**

Updating a **Vector Store** can be a costly operation because some time will pass before the **Vector Store** is updated, especially if the amount of content that needs to be re-embedded is high. Even costlier is the recreation of the **Vector Store** from scratch, and that is the only major reason why this strategy of improvement is avoided. However in cases where it is

determined that the **Vector Store** contains errors that are either coming from wrong data or wrong embeddings, it becomes inevitable to do so. Main options (also shown as part of the diagram on **Figure 2.6**) for recreating the whole **Vector Store** are to schedule (ex. **cron job**) the recreation, or to manually trigger it whenever the maintainer feels it is needed while monitoring the user demands and the content available in the **Data Source**. In the advanced **RAG** pipelines often the combination of both options is used for recreating the **Vector Store**.

2.4.2 Retriever tweaks

Some of the libraries that support **RAG** integration such as **LangChain**, provide classes and functions that expose arguments with which you can tweak the results retrieved from the **Vector Store**. These arguments control various aspects of the retrieval process, such as the number of retrieved documents, similarity search algorithm, or similarity search threshold, each of which affects how well the retrieved documents match the user query. By changing these parameters, you can optimize the **RAG** application to return more relevant, diverse, and useful document chunks for the **Generator** to use in the answers. This section will cover some of them which I find useful.

2.4.2.1 K results

The **k** argument controls the number of results (documents) retrieved from the **Vector Store**. It determines how many documents are retrieved based on the similarity search of their embeddings when compared to the query. Lower **k** implies that the **Generator** will yield shorter answers, with less diversity and chance for mistakes, but at the same time with lower costs and latency. On the other hand, higher **k** means that the answers the users get will have more diversity, more chance for irrelevant content to be included, but also higher costs and higher latency.

Python

```
def setup_generator(_retriever):
    prompt = hub.pull("rlm/rag-prompt")
    llm = ChatOpenAI(model_name="gpt-4o-mini", temperature=0)

    rag_chain = (
        {"context": _retriever | format_docs, "question":
    RunnablePassthrough()}
        | prompt
        | llm
        | StrOutputParser()
    )
    return rag_chain

def setup_rag_chain_with_arguments(_docs, k):
    text_splitter = RecursiveCharacterTextSplitter(
        chunk_size=1000, chunk_overlap=200, add_start_index=True
```

```
)
splits = text_splitter.split_documents(_docs)

vectorstore = Chroma.from_documents(documents=splits,
embedding=OpenAIEmbeddings())
retriever = vectorstore.as_retriever(
    search_type="similarity", search_kwargs={"k": k}
)

return retriever
```

Python

```
retriever = setup_rag_chain_with_arguments(documents, 1)
rag_chain = setup_generator(retriever)
rag_chain.invoke("Кoj бил Арсениј?")
```

Unset

Арсениј бил игумен и архимандрит на Бигорскиот манастир, кој управувал со манастирот од 1807 до 1839 година.

Python

```
retriever = setup_rag_chain_with_arguments(documents, 10)
rag_chain = setup_generator(retriever)
rag_chain.invoke("Кoj бил Арсениј?")
```

Unset

Арсениј бил архимандрит и игумен на манастирот Св. Јован Бигорски од 1807 до 1839 година. Тој бил познат градител и ктитор, кој значително придонел за развојот на манастирот и културниот идентитет на Македонија. Неговото место на погребување не е прецизно утврдено, но се претпоставува дека е во родниот Галичник.

Python

```
retriever = setup_rag_chain_with_arguments(documents, 50)
```

```
rag_chain = setup_generator(retriever)
rag_chain.invoke("Кoj бил Арсениј?")
```

Unset

Арсениј бил архимандрит и игумен на манастирот Св. Јован Бигорски од 1807 до 1839 година. Тој бил роден во Галичник и значително придонел за развојот на манастирот преку архитектонски и уметнички проекти. Неговото место на погребување не е прецизно утврдено, но се претпоставува дека е погребан во Галичник.

Listing 2.13: Examples of retrievals in Mutavchain with k set to 1, 10 and 50 respectively

I advise you not to set **k** too high because there is a chance that the maximum number of relevant documents that can be retrieved for a query to be far too low than the **k** value which can lead to getting similar answers as for lower **k** values but can also potentially pollute the retrieval with irrelevant content. A similar example with a value of 50 is shown on **Listing 2.13** for **Mutavchain** where it can be noticed that it returned the same answer as for **k=50**.

2.4.2.2 Search type

Choosing a suitable type of search for your **RAG** pipeline can also improve its performance in terms of the retrieved content. There are many search types, predominantly utilizing vector similarity. Most of the libraries supporting **RAG** expose the **search type** as a separate argument which can be used in combination with other arguments, such as the threshold, without much limitation that can tweak the **Retriever** and the **Vector Store**.

2.4.2.2.1 Similarity search score

One of the most widely used **search types** is the **similarity search score** which is supported by most of the **Vector Stores**. The only difference across the **Vector Stores** comes from the similarity algorithm that is used (for example **FAISS** by default uses **Approximate Nearest Neighbour - ANN**), but choosing different algorithms does not make a large impact like using a threshold for the **similarity search score**.

Python

```
def setup_openai_embeddings_and_chroma_retriever_with_arguments(
    _docs, k, score_threshold
):
    text_splitter = RecursiveCharacterTextSplitter(
        chunk_size=1000, chunk_overlap=200, add_start_index=True
```

```
)
splits = text_splitter.split_documents(_docs)

vectorstore = Chroma.from_documents(documents=splits,
embedding=OpenAIEmbeddings())
retriever = vectorstore.as_retriever(
    search_type="similarity_score_threshold",
    search_kwargs={"k": k, "score_threshold": score_threshold},
)

return retriever
```

Python

```
retriever =
setup_openai_embeddings_and_chroma_retriever_with_arguments(documents, 1, 0.5)
rag_chain = setup_generator(retriever)
rag_chain.invoke("Кoj бил Арсениј?")
```

Unset

Арсениј бил игумен и архимандрит од Бигорскиот манастир, кој управувал со манастирот од 1807 до 1839 година.

Python

```
retriever =
setup_openai_embeddings_and_chroma_retriever_with_arguments(documents, 1, 0.9)
rag_chain = setup_generator(retriever)
rag_chain.invoke("Кoj бил Арсениј?")
```

Unset

Не знам.

Python

```
retriever =
setup_openai_embeddings_and_chroma_retriever_with_arguments(documents, 10, 0.9)
```

```
rag_chain = setup_generator(retriever)
rag_chain.invoke("Кoj бил Арсениј?")
```

```
Unset
He знам.
```

Listing 2.14: Examples of similarity search score tweaks with values set to 0.5 and 0.9 in Mutavchain

Tweaking the **similarity search score** threshold has a similar impact as tweaking the **k** argument. The higher the value for the **similarity search score** threshold, the lower the number of the retrieved results is for the **Generator** to use. If the threshold is set too high then there is a chance that no context is retrieved and subsequently the **Generator** needs to have a properly customized prompt so that it avoids inaccurate answers being presented to the user. **Listing 2.14** shows some examples of tweaking the **similarity search score** argument combined with **k**. You can notice that if the **similarity search score** threshold is set too high, not even setting the **k** to retrieve more documents can help provide a suitable answer.

2.4.2.2.2 MMR

Using classic **similarity search** type or **similarity search score** by threshold focuses on the most similar content for retrieval. As most libraries for **RAG** offer tweaking the search type via special argument and reserved constants, you can easily choose the desired search type depending on the situation. There is another widely supported search type that also can be used for tweaking, that presents a tradeoff between the similarity and diversity of the retrieved documents, known as **Maximal Marginal Relevance (MMR)**. By applying this search type on the **Retriever**, you can make it select documents based on a combination of which of them are similar to the inputs, while also optimizing for diversity by penalizing the most similar content.

```
Python
def setup_openai_embeddings_and_chroma_retriever_with_arguments(
    _docs, search_type, k, score_threshold
):
    text_splitter = RecursiveCharacterTextSplitter(
        chunk_size=1000, chunk_overlap=200, add_start_index=True
    )
    splits = text_splitter.split_documents(_docs)

    vectorstore = Chroma.from_documents(documents=splits,
        embedding=OpenAIEmbeddings())
```

```
retriever = vectorstore.as_retriever(  
    search_type=search_type,  
    search_kwargs={"k": k, "score_threshold": score_threshold},  
)  
  
return retriever
```

Python

```
retriever =  
setup_openai_embeddings_and_chroma_retriever_with_arguments(documents, 'mmr',  
1, 0.5)  
rag_chain = setup_generator(retriever)
```

Python

```
rag_chain.invoke("Кoj бил Арсениј?")
```

Unset

Арсениј бил игумен и архимандрит од Бигорскиот манастир, кој управувал со манастирот од 1807 до 1839 година.

Python

```
rag_chain.invoke("Каква врска имал Арсениј со Бигорски?")
```

Unset

Арсениј имал врска со Бигорскиот манастир преку неговата творечка иницијатива и работа на иконостасот.

Python

```
retriever =  
setup_openai_embeddings_and_chroma_retriever_with_arguments(documents, 'mmr',  
1, 0.9)  
rag_chain = setup_generator(retriever)
```

Python

```
rag_chain.invoke("Каква врска имал Арсениј со Бигорски?")
```

Unset

Арсениј имал амбициозна творечка иницијатива за растежот на Бигорскиот манастирски комплекс и организирал најдобрите градители, иконописци и резбари за тоа.

Python

```
retriever =  
setup_openai_embeddings_and_chroma_retriever_with_arguments(documents, 'mmr',  
20, 0.5)  
rag_chain = setup_generator(retriever)
```

Python

```
rag_chain.invoke("Каква врска имал Арсениј со Бигорски?")
```

Unset

Арсениј имал големо ктиторско значење за Бигорскиот манастир, бил игумен во манастирскиот комплекс Св. Јован Бигорски и раководел со горните набавки, живопис и опрема на црквата.

Listing 2.15: Examples of MMR search type usage in Mutavchain

MMR was used in **Mutavchain** to deal with more complex queries coming from the user in the form of questions about two or more topics. When used on simple questions, there is a small risk that it can reduce the pool of relevant documents. Also you need to pay attention when combining it with other arguments, such as the ones mentioned in the previous sections. For example, even when the similarity score is set (does not even matter how high), **MMR** will ignore it, because the score threshold is related to the **similarity search score** type. Also even when **k** is set too high, **MMR** overrides it by trying to penalize the most relevant documents returned. **Listing 2.15** covers some of these consequences with several examples.

2.4.2.3 Chunk size

Changing the **chunk size** before creating the **Vector Store** can significantly alter the retrieval in the **RAG** pipeline. The **chunk size** refers to the number of characters into which a document is divided before being embedded and stored in the **Vector Store**. It is usually combined with the argument for the size of the overlap in order to avoid losing the relationship between the chunks.

Python

```
def setup_openai_embeddings_and_chroma_retriever_with_arguments(
    _docs, chunk_size, chunk_overlap, search_type, k, score_threshold
):
    text_splitter = RecursiveCharacterTextSplitter(
        chunk_size=chunk_size, chunk_overlap=chunk_overlap,
        add_start_index=True
    )
    splits = text_splitter.split_documents(_docs)

    vectorstore = Chroma.from_documents(documents=splits,
        embedding=OpenAIEmbeddings())
    search_args = {"k": k}
    if search_type == "similarity_score_threshold" and score_threshold:
        search_args["score_threshold"] = score_threshold
    retriever = vectorstore.as_retriever(
        search_type=search_type, search_kwargs=search_args
    )

    return retriever
```

Python

```
retriever =
setup_openai_embeddings_and_chroma_retriever_with_arguments(documents, 800,
200, 'similarity_score_threshold', 1, 0.5)
rag_chain = setup_generator(retriever)
```

Python

```
rag_chain.invoke("Каква врска имал Арсениј со Бигорски?")
```

Unset

Арсениј бил архимандрит кој имал амбициозна творечка иницијатива и големи визии за растежот на Бигорскиот манастирски комплекс.

Python

```
retriever =  
setup_openai_embeddings_and_chroma_retriever_with_arguments(documents, 100, 50,  
'similarity_score_threshold', 1, 0.5)  
rag_chain = setup_generator(retriever)
```

Python

```
rag_chain.invoke("Каква врска имал Арсениј со Бигорски?")
```

Unset

Арсениј бил епископ во Бигорски манастир.

Listing 2.16: Examples of chunk size manipulation in Mutavchain

What value should you set the **chunk size** argument to and when? Smaller chunks allow the system to retrieve highly relevant parts of the document that directly match the query. This is especially useful when the query targets very specific information, such as the cases when the user uses shorter questions. Even though the **Vector Store** is created slower with reduced **chunk size** because of many separate embeddings, it also results in faster retrieval since there are more, smaller pieces of content that are easier to compare to the query vector. Another benefit of having lower **chunk size** is that the chances of including irrelevant information in the retrieved results are reduced. Some examples are shown on **Listing 2.16**.

Python

```
retriever =  
setup_openai_embeddings_and_chroma_retriever_with_arguments(documents, 1000,  
10, 'similarity_score_threshold', 1, 0.5)  
rag_chain = setup_generator(retriever)
```

```
Python
```

```
rag_chain.invoke("Каква врска имал Арсениј со Бигорски?")
```

```
Unset
```

```
Петровиќ не нуди доволно контекст за да се одговори на прашањето. Немам информации.
```

Listing 2.17: Example of unsuccessful answer with low chunk overlap and high chunk size in Mutavchain

On the other hand, having higher **chunk size** keeps more of the original context intact, which can improve the quality of the response, especially for queries that require understanding the broader scope of a topic. For higher **chunk size**, the ability to set higher **chunk overlap** values also increases, which creates other opportunities for improvement but it also opens up some risks. Keeping low **chunk overlap** for higher **chunk size** can result in losing the relationship between the chunks i.e. some questions coming from the user might be unanswered such as the example on **Listing 2.17**.

2.4.3 Chunking

In the previous section I showed how changing the **chunk size** and the **overlap** can impact the effectiveness of the **RAG** pipeline. Now I will continue presenting my experience with chunks, but from a different angle - how changing the chunking strategy can enhance your **RAG** application. There are numerous ways of breaking down large documents into smaller, manageable pieces before they are stored in the **Vector Store**. I will mention only a subset that I believe consists of the most important strategies for chunking.

2.4.3.1 Fixed size

This strategy requires the least effort to implement because it simply chops the content without worrying about the tokens, sentences or other logical text units. It results in very quick splitting without much code base behind the used library. For example, you can use pure **Python** instead of **LangChain**'s **CharacterTextSplitter** to divide the content into chunks with fixed length without much problems. **LangChain**'s functions and classes for splitting the loaded documents also rely on the **separator** argument which if found in the content, uses it to split before the **chunk size** value is reached. As shown in **Listing 2.18**, it extracts the first chunk on the 192nd character when the last space character is reached before 200.

```
Python
```

```
text_splitter = CharacterTextSplitter(
```

```
separator=" ",
chunk_size=200,
chunk_overlap=0,
length_function=len
)
splits = text_splitter.split_documents(documents)
```

```
Python
splits[0]
```

```
Unset
Document(
  metadata={
    "title": "арсениј2",
    "when": "2023-08-08T07:46:03.525Z",
  },
  page_content="\ufe0fГенијалниот Арсениј \r\nградител, архимандрит и игумен
\r\n од Бигорскиот манастир\r\n\r\n\r\n\r\nНеповторливиот мијачки центар Галичник
во текот на минатите векови ни подари низа истакнати знаменити личности, кои",
)
```

```
Python
splits[1]
```

```
Unset
Document(
  metadata={
    "title": "арсениј2",
    "when": "2024-08-08T07:46:03.525Z",
  },
  page_content="ни оставија врвни дела од областа на градителството,
зографството, иконописот, ликовната уметност и копаничарството. Во таа редица
великани, посебно место му припаѓа на градителот и ктитор Арсениј.",
)
```

Listing 2.18: Fixed size chunking in the loaded documents of Mutavchain

A huge disadvantage of this approach for chunking is that there is no semantic relationship between the chunks which can result in significant loss of information and huge negative implications when the desired documents are retrieved. Therefore this chunking strategy is rarely recommended except in the cases where the documents follow strict formats or templates.

2.4.3.2 Sliding window

In order to mitigate the negative impact the **Fixed size chunking** strategy has regarding the semantic relationship of the chunks, a separate argument is introduced (**chunk overlap**) as I mentioned in one of the previous sections. Here I will only add that when combined with the **Fixed size chunking** it is known as **Sliding window** chunking which can provide surprisingly good results. However it requires many trial and error iterations, which usually consumes a lot of time, hence it possesses quite a risk for the business.

Python

```
text_splitter = CharacterTextSplitter(
    separator=" ",
    chunk_size=200,
    chunk_overlap=10,
    length_function=len
)
splits = text_splitter.split_documents(documents)
```

Python

```
splits[0]
```

Unset

```
Document(
  metadata={
    "title": "арсениј2",
    "when": "2023-08-08T07:46:03.525Z",
  },
  page_content="\ufffdГенијалниот Арсениј \r\nградител, архимандрит и игумен\r\nод Бигорскиот манастир\r\n\r\n\r\nНеповторливиот мијачки центар Галичник во текот на минатите векови ни подари низа истакнати знаменити личности, кои",
)
```

```
Python  
splits[1]
```

```
Unset  
Document(  
    metadata={  
        "title": "арсениј2",  
        "when": "2024-08-08T07:46:03.525Z",  
    },  
    page_content="кои ни оставија врвни дела од областа на градителството,  
зографството, иконописот, ликовната уметност и копаничарството. Во таа редица  
великани, посебно место му припаѓа на градителот и ктитор",  
)
```

Listing 2.19: Sliding window chunking in the loaded documents of Mutavchain

Listing 2.19, shows the same example of Fixed size chunking from **Listing 2.18** combined with a **chunk overlap**. it extracts the first chunk on the 192nd character and the second chunk starts with the word that is found in the 10 characters before the end of the first chunk, meaning that it counts the **separator** before the 10 character length of the overlap is reached. This points to a certain connection between the chunks, although the word *кои* is relatively frequent in Macedonian which means that if there are other chunks starting with the same word, they can be mistakenly treated as the continuation of the sentence instead of the second chunk. Therefore, more trials are needed towards increasing the **chunk overlap** before an optimal **Sliding window chunking** is found. With addition of more content to the **Document Source** and hence the **Vector Store**, the **Sliding window chunking** strategy, if continued to be applied, should be reevaluated for potential adjustment of the **chunk overlap** value.

2.4.3.3 Sentence based

Whenever the **separator** is set to a string that usually marks the end of a sentence, it results in chunking that uses the sentence as a content unit. The previously mentioned **CharacterTextSplitter** can be used for this purpose with **separator** values such as full stop, question mark, exclamation mark or in some extreme cases even a new line. Besides the **CharacterTextSplitter**, **LangChain** library provides the **RecursiveCharacterTextSplitter** which tries to split at more meaningful semantic boundaries and falls back to smaller units if needed. It also produces more coherent chunks by prioritizing larger, semantically meaningful splits before falling back on smaller units.

```
Python
text_splitter = RecursiveCharacterTextSplitter(
    chunk_size=200, chunk_overlap=10, length_function=len
)
splits = text_splitter.split_documents(documents)
```

```
Python
splits[0]
```

```
Unset
Document(
  metadata={
    "title": "арсениј2",
    "when": "2023-08-08T07:46:03.525Z",
  },
  page_content="\uffeffГенијалниот Арсениј \r\nградител, архимандрит и игумен
\r\n од Бигорскиот манастир",
)
```

```
Python
splits[1]
```

```
Unset
Document(
  metadata={
    "title": "арсениј2",
    "when": "2023-08-08T07:46:03.525Z",
  },
  page_content="Неповторливиот мијачки центар Галичник во текот на минатите
векови ни подари низа истакнати знаменити личности, кои ни оставија врвни дела
од областа на градителството, зографството, иконописот,",
)
```

```
Python
splits[2]
```

```

Unset
Document(
  metadata={
    "title": "арсениј2",
    "when": "2023-08-08T07:46:03.525Z",
  },
  page_content="ликовната уметност и копаничарството. Во таа редица великани,
  посебно место му припаѓа на градителот и ктитор Арсениј, архимандрит и игумен
  во манастирот Св. Јован Бигорски. Арсениј со својот",
)

```

```

Python
splits[3s]

```

```

Unset
Document(
  metadata={
    "title": "арсениј2",
    "when": "2023-08-08T07:46:03.525Z",
  },
  page_content="со својот генијален, возвишен, молитвен и креативен дух за 32
  години раководење (1807-1839) допринел небоземниот центар од Река да ги доживее
  своите најсвездени мигови и за навек да биде едно од",
)

```

Listing 2.20: Sentence based chunking with RecursiveCharacterTextSplitter

An example of **RecursiveCharacterTextSplitter** is given on **Listing 2.20** where the differences in chunking compared to the previous listing and **CharacterTextSplitter** can clearly be noticed in the form of having smaller chunks (the title at the first chunk) at the expense of larger ones that fill up to the **chunk size**. Although more performant, the **Sentence based** chunking strategy is more complex than the previous ones and results in additional latency which certainly pays off for more complex queries and content.

2.4.3.4 NSP based

One problem with the **Sentence based chunking** approach is its reliance on content that is sequentially correct i.e. the flow of the context is logical. But what happens if the original content is out of order? In that case there will be relationships between sentences that are far away from each other which can ultimately affect the way the chunks are embedded. There is

one solution to this however - using a **Next Sentence Predictor (NSP)** model to predict the most probable continuation of a sentence.

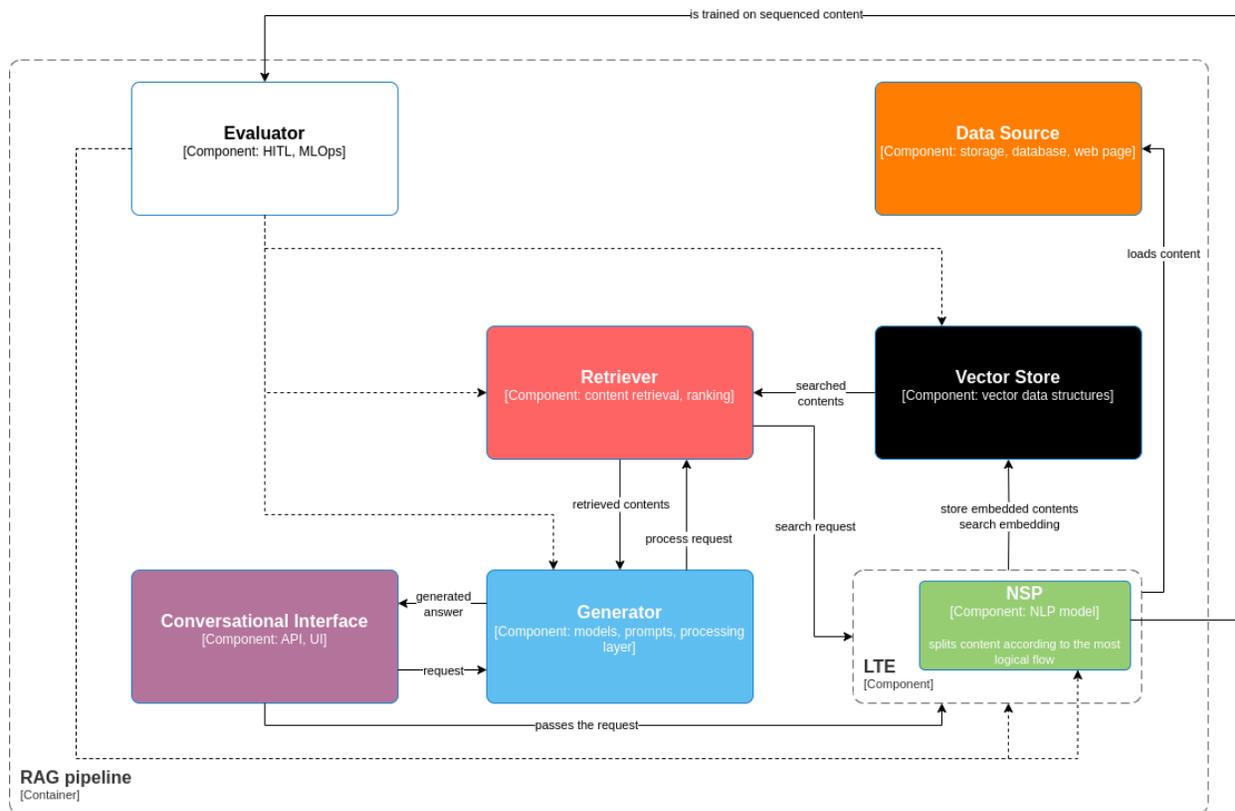


Figure 2.7: NSP chunker as part of the LTE unit in the RAG pipeline

Introducing an **NSP** in the **RAG** pipeline means that the **LTE** unit will include a model for transformation that includes splitting the content according to what the model predicts to be the most natural sequential of information. The adjusted diagram of a common **RAG** pipeline that includes the **NSP** chunker is shown on **Figure 2.7**.

Though advantageous, **NSP** is not that easy to integrate in the **RAG** pipeline. First it requires additional time for splitting because the model can introduce additional latency when splitting, something that the simpler chunkers do not. Second, in most of the cases, such as dealing with content written in low frequently used languages (Macedonian in the case of **Mutavchain**), **NSP** requires high quality and quantity data which results in significant time necessary to fine-tune a model or often to train it from scratch. This implies maintenance costs just for the **NSP** model. The trained **NSP** model itself needs to provide high performance otherwise predicting the next sentence can proceed into a totally different direction which will lead to degrading the chunking to a level worse than the simpler strategies are chunking.

Python

```
def nsp_chunker(content, max_chunk_size=1000, similarity_threshold=0.2):
```

```

pattern = r"(?<=[!\?\.\.])\s+"
sentences = re.split(pattern, content)

nsp_template = """Predict the next sentence.

Sentence: {sentence}
Next sentence: """
nsp_prompt = PromptTemplate.from_template(nsp_template)
nsp_llm = ChatOpenAI(model_name="gpt-4o-mini", temperature=0)
nsp_chain = nsp_prompt | nsp_llm

embeddings = OpenAIEmbeddings()
openai_evaluator = load_evaluator(
    "embedding_distance",
    distance_metric=EmbeddingDistance.COSINE,
    embeddings=embeddings,
)

splits = []
current_chunk = ""
index = 0
for sentence in sentences:
    chunk_to_add = sentence
    next_sentence = nsp_chain.invoke(sentence)
    if index:
        similarity_score = openai_evaluator.evaluate_strings(
            prediction=sentences[index - 1],
            reference=next_sentence.content
        )["score"]
        print(similarity_score)
        if similarity_score < similarity_threshold or (
            len(current_chunk) + len(chunk_to_add) > max_chunk_size
        ):
            splits.append(current_chunk)
            current_chunk = chunk_to_add
        else:
            current_chunk += f". {chunk_to_add}"
    else:
        current_chunk = chunk_to_add
    index += 1
splits.append(current_chunk)

return splits

```

```
Python
```

```
document_text = "Впрочем пред крајот на мојов живот најмногу во главата ми се  
вртеа моите родители, другари, комшии, бележити галичани, мојата животна  
определба да се очуваат Галичник и Мијачијата, човечноста, добрината,  
овчарството. Свесен сум дека овој материјал на некого нема да му се допадне.  
Сепак, сметам дека откако ќе се прочита ова дело, може да се констатира дека  
сте доловиле барем дел од аманетните загледи на еден човек кој целиот свој  
живот го посвети на Галичник и Бистра."
```

```
chunks = nsp_chunker(document_text)  
chunks
```

```
Unset
```

```
['Впрочем пред крајот на мојов живот најмногу во главата ми се вртеа моите  
родители, другари, комшии, бележити галичани, мојата животна определба да се  
очуваат Галичник и Мијачијата, човечноста, добрината, овчарството. Свесен сум  
дека овој материјал на некого нема да му се допадне.', 'Сепак, сметам дека  
откако ќе се прочита ова дело, може да се констатира дека сте доловиле барем  
дел од аманетните загледи на еден човек кој целиот свој живот го посвети на  
Галичник и Бистра.']
```

Listing 2.21: Example of utilizing an NSP chunker in Mutavchain

Because **Mutavchain** relies on content that has logical sequences (articles and short books), involving **NSP** can be redundant. **Listing 2.21** shows an example of creating an **NSP** chunker in **Mutavchain** based on content in Macedonian. A special evaluator function from **LangChain** is used along with the help of an **OpenAI LLM** and using prompting instead of training/fine-tuning. As the example shows, because of the correct sequence of the original content, the **NSP** chunker splits the content in the same way as the **Sentence based chunker** does.

2.4.3.5 Context-aware

Another case of not having clean content can be texts that include **Markdown**, **HTML** or other markup languages. They pollute the content and include unnecessary embeddings in the **Vector Store**. The most simple way to avoid treating such reserved strings by the **Embedder** is to strip them with regular expressions. But that will not always work because there can be a variety of markup formats and often they can include errors such as inappropriately closed tags or wrongly used syntax.

Python

```
splitter = SentenceSplitter()  
chunking_agent = T5ChunkerModel("t5-small")
```

```
document_text = "Ракописот е еден од ретките потфати и специфично фактографско творештво, приказ од животот и работата на авторот и неговите аманети. Ракописот се однесува за еден простор, за времето минато и идно, содржини кои малку им се познати и на самите галичани. Фактите кои ни ги дава авторот за времето по Втората светска војна, имаат цел да нè предизвикаат и да нè обврзат да направиме уште повеќе за ревитализација на родниот Галичник.\r\nВаква книга-животоопис може да напише Велимир Граоркоски-Веле поради неговото долго присуство и работа во Галичник, затоа што тој беше бескрајно вљубен во единствениот Галичник, познат по неговата богата историја и природни убавини.\r\nВелимир Граоркоски е човек кој ги сакаше најдлабоко своите Мијаци, кои се отргнале од своите огништа и кои сметаа дека никогаш не престанал животот во Галичник и Мала Река. Оттаму и оваа книга е повик што треба да се следи за да се разгори жарта на старите огништа, а за тоа, смета авторот, постојат вистински личности за кои кажува и самиот.\r\nЗошто авторот се зафатил да пишува за ваква тема? Основните побуди што го поттикнале да се зафати со пишување на овој мошне значаен текст за историјата на Галичник по ослободувањето на земјата и за многу галичани, ги наоѓам во неговиот долг престој и работа во Галичник сè до неговото пензионирање и познанствата со голем број галичани.\r\nГаличник во текот на Втората светска војна претрпе катастрофални последици и промени во негативна смисла. Многу фамилии го напуштија селото. Сточарите со своите стада останаа на окупираниот дел на Македонија од страна на Бугарија. Западна Македонија и Галичник беа под окупација на Италија и Албанија. Галичник беше блокиран и без услови за нормален опстанок. Тоа беше и причина што многу семејства го напуштија. По ослободувањето, со настанатите промени на политичкиот и економскиот систем, уште повеќе се промени ликот и занимањата во Галичник. Приватниот сектор, сточарството и печалбарството се заменија со општествен. Галичник како најразвиен стопански центар, остана без својот најкрупен капитал-сточарството. Затоа Галичник како и многу други села, беше принуден да формира задруги со цел да го вработи работоспособното население кое остана без своите средства за работа (овците и печалбата). Во Галичник се формираа две задруги. Подоцна двете задруги се обединија под име\r\n\r\n\r\nЗемјоделско-сточарска задруга „Бистра“.
```

```
split_content = splitter.split_text(document_text, merge_sentences=1)  
chunks = chunking_agent.chunk(split_content)  
chunks
```

Unset

```
['Ракописот е еден од ретките потфати и специфично фактографско творештво, приказ од животот и работата на авторот и неговите аманети. ', 'Ракописот се
```

однесува за еден простор, за времето минато и идно, содржини кои малку им се познати и на самите галичани. ', 'Фактите кои ни ги дава авторот за времето по Втората светска војна, имаат цел да нè предизвикаат и да нè обврзат да направиме уште повеќе за ревитализација на родниот Галичник.\r', 'Ваква книга-животоопис може да напише Велимир Граоркоски-Веле поради неговото долго присуство и работа во Галичник, затоа што тој беше бескрајно вљубен во единствениот Галичник, познат по неговата богата историја и природни убавини.\r', 'Велимир Граоркоски е човек кој ги сакаше најдлабоко своите Мијаци, кои се отргнале од своите огништа и кои сметаа дека никогаш не престанал животот во Галичник и Мала Река. ', 'Оттаму и оваа книга е повик што треба да се следи за да се разгори жарта на старите огништа, а за тоа, смета авторот, постојат вистински личности за кои кажува и самиот.\r', 'Зошто авторот се зафатил да пишува за ваква тема? ', 'Основните побуди што го поттикнале да се зафати со пишување на овој мошне значаен текст за историјата на Галичник по ослободувањето на земјата и за многу галичани, ги наоѓам во неговиот долг престој и работа во Галичник сè до неговото пензионирање и познанствата со голем број галичани.\r', 'Галичник во текот на Втората светска војна претрпе катастрофални последици и промени во негативна смисла. Многу фамилии го напуштија селото. ', 'Сточарите со своите стада останаа на окупираниот дел на Македонија од страна на Бугарија. ', 'Западна Македонија и Галичник беа под окупација на Италија и Албанија. Галичник беше блокиран и без услови за нормален опстанок. ', 'Тоа беше и причина што многу семејства го напуштија. ', 'По ослободувањето, со настанатите промени на политичкиот и економскиот систем, уште повеќе се промени ликот и занимањата во Галичник. ', 'Приватниот сектор, сточарството и печалбарството се заменија со општествен. Галичник како најразвиен стопански центар, остана без својот најкрупен капитал-сточарството. ', 'Затоа Галичник како и многу други села, беше принуден да формира задруги со цел да го вработи работоспособното население кое остана без своите средства за работа (овците и печалбата). ', 'Во Галичник се формираа две задруги. Подоцна двете задруги се обединија под име Земјоделско-сточарска задруга „Бистра“. ']

Listing 2.22: Using context-aware-chunker library for clean content in Mutavchain

Luckily, there are some **NLP** models that can parse the whole content and extract only the context part. One such utilization of **Context-aware** models is the **context-aware-chunker** library for **Python** that takes advantage of **T5** and **BERT** models to extract only the meaningful part of the content. **Listing 2.22** shows its unnecessary usage in **Mutavchain** where the documents are mostly clean and do not contain any markup languages which does not mean that there will not be any documents in the future that include. Until such documents occur, the **Context aware** chunker by default splits the content by the new line special characters or spaces, somewhat similar to the **Sentence based** chunker, while also taking advantage of the **merge_sentences** argument for the number of sentences that should be included in a single chunk.

2.4.3.6 Semantic

When chunking large content, similar sentences can be encountered based on the semantics. Those sentences when embedded have a small distance compared to the rest of the sentences in the content. Therefore those sentences are grouped and take part of a single chunk. Defined as such, the **Semantic** chunking strategy introduces another embedder in the **LTE** unit as shown in **Figure 2.8**.

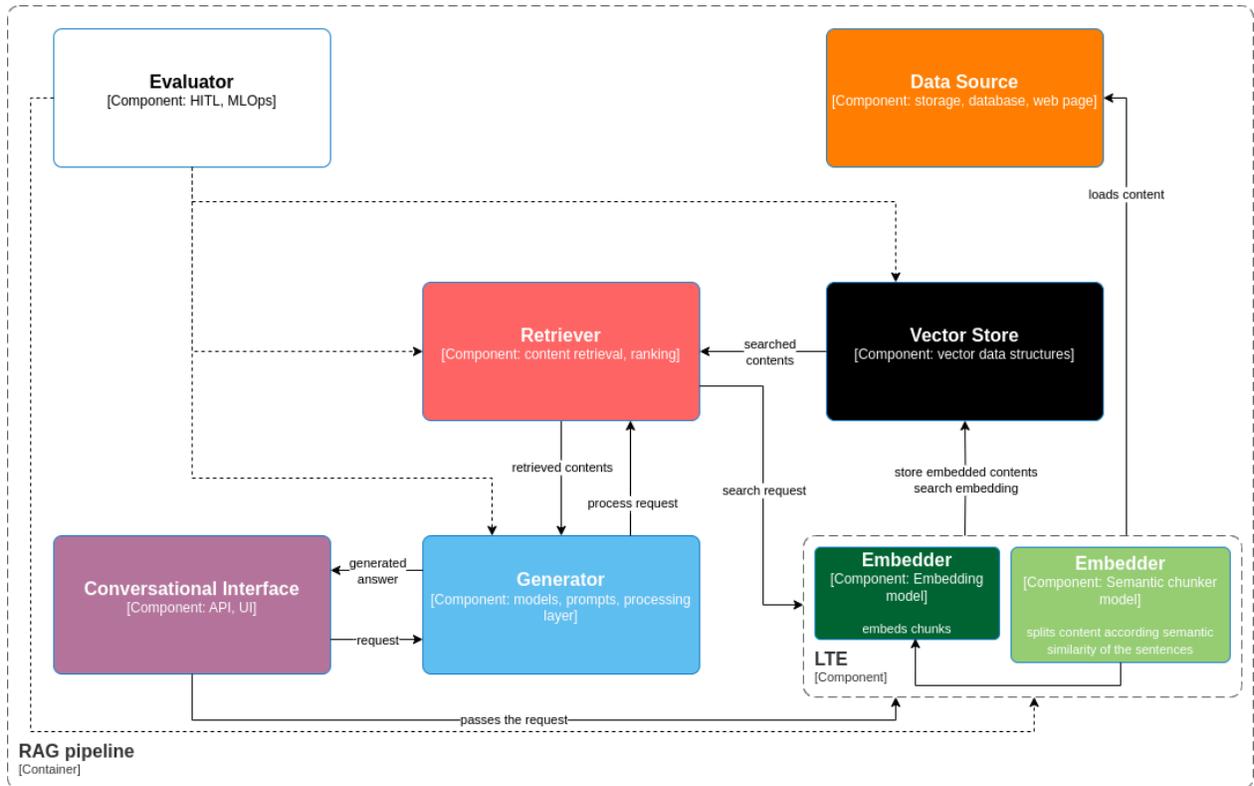


Figure 2.8: **Semantic chunker as part of the LTE unit in the RAG pipeline**

LangChain supports **Semantic** chunking, although by the time of writing this paragraph it was still in experimental state, therefore not recommended for production usage. When tried to be used for **Mutavchain** purposes, it did not bring much improvements as examples in **Listing 2.23** show.

```
Python
def setup_retriever(_docs, breakpoint_threshold_type, search_type, k,
score_threshold):
    text_splitter = SemanticChunker(
        OpenAIEmbeddings(), breakpoint_threshold_type=breakpoint_threshold_type
    )
    splits = text_splitter.split_documents(_docs)
```

```
vectorstore = Chroma.from_documents(documents=splits,
embedding=OpenAIEmbeddings())
search_args = {"k": k}
if search_type == "similarity_score_threshold" and score_threshold:
    search_args["score_threshold"] = score_threshold
retriever = vectorstore.as_retriever(
    search_type=search_type, search_kwargs=search_args
)

return retriever, len(splits)
```

Python

```
retriever, splits_length = setup_retriever(documents, 'percentile',
'similarity', 20, 0.2)
rag_chain = setup_rag_chain(retriever)
answer = rag_chain.invoke("Како се поврзани Галичник и Јанче?")
print(f'Total chunks: {splits_length}; Answer: {answer}')
```

Unset

```
'Total chunks: 278; Answer: Галичник и Јанче се поврзани преку патот што води
преку месноста Скала, кој поминува низ природната зоолошка градина на Бистра.'
```

Python

```
retriever, splits_length = setup_retriever(documents, 'standard_deviation',
'similarity', 20, 0.2)
rag_chain = setup_rag_chain(retriever)
answer = rag_chain.invoke("Како се поврзани Галичник и Јанче?")
print(f'Total chunks: {splits_length}; Answer: {answer}')
```

Unset

```
'Total chunks: 127; Answer: Галичник и Јанче се поврзани преку патот што води
преку месноста Скала, кој поминува низ природната зоолошка градина на Бистра.'
```

Python

```
retriever, splits_length = setup_retriever(documents, 'interquartile',  
'similarity', 20, 0.2)  
rag_chain = setup_rag_chain(retriever)  
answer = rag_chain.invoke("Како се поврзани Галичник и Јанче?")  
print(f'Total chunks: {splits_length}; Answer: {answer}')
```

Unset

```
'Total chunks: 278; Answer: Галичник и Јанче се поврзани преку патот што води  
преку месноста Скала и преку природната зоолошка градина на Бистра.'
```

Listing 2.23: Semantic chunking examples in Mutavchain

Semantic chunker disregards other arguments such as the **chunk size** and the **chunk overlap**. It also brings along another argument (**breakpoint threshold type**) that can be tweaked by choosing either **percentile** (default), **standard_deviation**, **interquartile** or **gradient** which results in a different number of chunks (coincidentally **percentile** and **interquartile** have the same number in the examples in the listing). Although useful for experimental purposes, you should keep an eye on any additional costs it brings along, because it relies on an embedding model.

2.4.4 Embedded summarization

It is obvious that the larger the content, the larger the **Vector Store**. Larger content has a higher probability to include redundant or unnecessary information which does not positively affect the overall **RAG** pipeline, but can have a negative impact on the retrieval's latency given the increased **Vector Store**'s size. One solution to avoid this is to summarize the content before being embedded and included in the **Vector Store**.

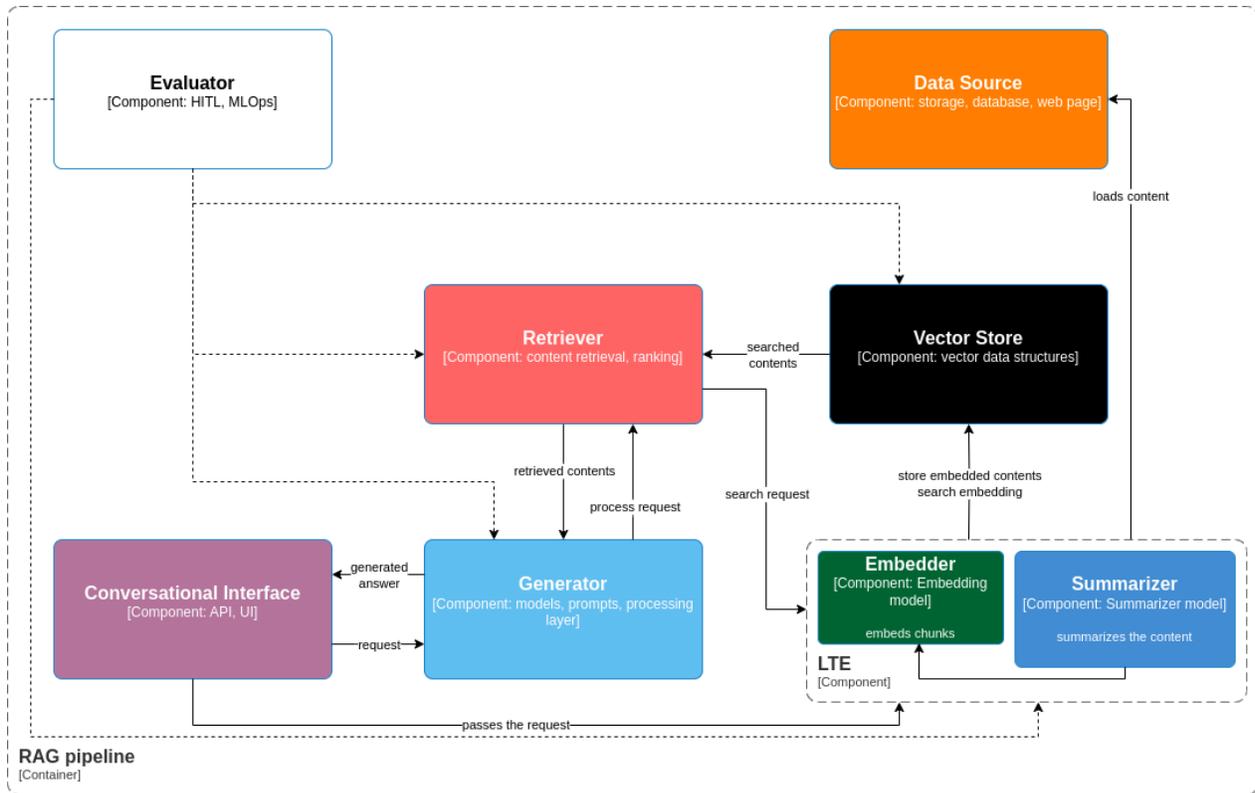


Figure 2.9: **Embedded summarization as part of the LTE unit in the RAG pipeline**

Embedding the summarization needs a summarizer model to perform the summarization task as part of the transformation step in the **LTE** unit, before the embedding happens. This process is shown on **Figure 2.9**.

Python

```
llm_summarizer = ChatOpenAI(temperature=0, model_name="gpt-4o-mini")
chain_summarizer = load_summarize_chain(llm_summarizer, chain_type="stuff")
```

Python

```
summarized_documents = chain_summarizer.invoke(content)
summarized_documents
```

Unset

```
{
  "input_documents": [
    Document(
```

```

        metadata={"start_index": 0},
        page_content="\ufffdСтојче Дојчиноски-Росоклија е рецензент на
книгата Галичка мозаична сознајба.\r\nГодината на издавање на книгата Галичка
мозаична сознајба е 2010.",
    )
],
    "output_text": 'Стојче Дојчиноски-Росоклија рецензира книгата "Галичка
мозаична сознајба", издадена во 2010 година.',
}

```

Listing 2.24: **Embedded summarization example in Mutavchain**

Embedded summarization techniques come with some risks though depending on the performance of the summarizer. Loss of information, that later can prove to be valuable, can occur. In addition, since the most performant summarizer **LLMs** favor the English language, when used on content written in other languages, some of them tend to directly generate summarization in English instead of the original language. In order to avoid this, the prompts need to be adequately engineered so that there is no undesired translation occurring. In general, employing **Embedded summarization** for use cases like **Mutavchain**, where the original content is in low frequently used languages and almost every piece of information is valuable, is unnecessary. **Listing 2.24** shows an example in **Mutavchain** of utilizing the **LangChain's load_summarize_chain** function for a single document chunk that can fit in the context window of the **gpt-4o-mini** model used as a summarizer. For larger document content that cannot be handled by the summarizer model, instead of using **stuff chain type** to “squeeze” the whole content in the prompt, **map_reduce** alternative can be used that does the summarization in several steps in order to overcome the **LLM** limitation.

3. Application improvements

When a **RAG** application is mentioned, it usually refers to the **RAG** pipeline plus some additions that make it complete for using. This chapter includes some of the external improvements of **RAG** application that can impact not just the **RAG** pipeline but the whole user experience.

3.1 Context search

One reason why **RAG** applications are so powerful lies in the **Context search**, as opposed to the more traditional **Keyword search** engines. The search is based on more semantic comparison instead on syntactic. I already presented this behavior as a consequence of searching through a corpus of vectors as a result of accurate embedding where the semantically similar pieces of content are represented close to each other in the vector space. However this is not enough to have a smooth interfacing because one conversation represented in a session can refer to multiple topics and the attention keyword will not always be part of the queries. Most of the **LLMs** can take the responsibility of understanding the context of the conversation and there are libraries such as **LangChain** that provide relatively simple ways of handling the chat history.

```
Python
contextualized_system_prompt = """Given a chat history and the latest user
question \
which might reference context in the chat history, formulate a standalone
question \
which can be understood without the chat history. Do NOT answer the question, \
just reformulate it if needed and otherwise return it as is."""
contextualized_prompt = ChatPromptTemplate.from_messages(
    [
        ("system", contextualized_system_prompt),
        MessagesPlaceholder("chat_history"),
        ("human", "{input}"),
    ]
)
history_aware_retriever = create_history_aware_retriever(
    llm, retriever, contextualized_prompt
)
```


г.)\r\n_____ \r\nМакедонска народна архитектура\r\n\r\n\r\n\r\nАко сакаме со малку зборови да ја претставиме Македонија, тогаш би рекле: тоа е библиска земја, со различни карактеристики, со природни убавини, со големо природно богатство - ретки руди, ориз, тутун. Тоа е земја со богато минато, за чија висока култура сведочат многубројните и разновидни споменици: стари ископини, тврдини, цркви, манастири, хармоничните населби и градови и над сè пластичната и пејзажна народна куќа. Во неа сè уште постојат живите траги на големите култури, со посебен зачуван стандард на високо ниво на архитектонската свест.\r\n\r\n\r\n\r\n \r\n\r\n\r\n\r\n\r\nКуќата на Робевци",

```
    metadata={
      "title": "Galichnik vo mojot kolazh",
      "when": "2024-06-09T19:24:35.112Z",
      "start_index": 30558,
```

```
    },
```

```
  ),
```

```
  Document(
```

```
    page_content="Круна на неговата градителска кариера е изградбата на црквата „Св. Петар и Павле“ во неговиот роден Галичник, која е ремек-дело на мијачката градителска рака и невидено мајсторство на луковските каменоресци. Таа прекрасна градба, завршена во 1930 година, а осветена во 1931 година, плени со својата импозантна камена структура, богатиот ентериер со орнаменти и, пред сè, со одличната акустичност. Стана препознатлив белег на Галичник и на манифестацијата „Галичка свадба“, светилиште во кое се венчаваат на најоригинален македонски обичај многу брачни двојки и светилиште во кое богослужеле сите досегашни македонски архиепископи.\r\n\r\n\r\n\r\n \r\n\r\n\r\n\r\n\r\nТеатарот во Скопје",
```

```
    metadata={
      "title": "Galichnik vo mojot kolazh",
      "when": "2024-06-09T19:24:35.112Z",
      "start_index": 29491,
```

```
    },
```

```
  ),
```

```
  Document(
```

```
    page_content="Театарот во Скопје\r\n\r\n\r\n\r\nТокму ова сеќавање нека биде и повод, неговите роднокрајци достоино да ја одбележат 120-годишнината од раѓањето на градителот Јордан Стојанчески и 75 години од изградбата на неговиот прекрасен објект црквата „Св. Петар и Павле“ со една пригодна спомен-плоча со името на градителот, годината на раѓањето - 1885 и годината на неговата смрт - 1943, која ќе биде поставена во објектот.\r\n\r\n\r\n\r\n \r\n\r\n\r\n\r\n\r\nЦрквата „Св. Петар и Павле“ во Галичник\r\n\r\n\r\n\r\n \r\n\r\n\r\n\r\n\r\nСтанбена зграда во Скопје\r\n\r\n\r\n\r\n\r\n(„Дневник“, 18.3.2006 г.)\r\n_____ \r\nМакедонска народна архитектура",
```

```
    metadata={
      "title": "Galichnik vo mojot kolazh",
```

```

        "when": "2024-06-09T19:24:35.112Z",
        "start_index": 30137,
    },
),
Document(
    page_content="„Св. Петар и Павле“ во својот роден Галичник, која е ремек-дело на мијачката градителска рака и невидено мајсторство на луковските каменорезачи. Таа прекрасна градба чичко Јордан ја градеше три години и беше завршена во 1930 година, а осветена во 1931 година. Плени со својата импозантна камена структура, богатиот ентериер со орнаменти и пред сè, со одличната акустичност. Денес на неговата градба која не е поместена ниту за еден милиметар се воодушевуваат сите посетители.\r\nНормално, морам да ги издвојам и тројцата мои прочуени роднокрајци за кои е многу пишувано и затоа само на кратко ќе се осврнам на нив и нивните дела.",
    metadata={
        "title": "Galichka Mozaichna Soznajba",
        "when": "2024-06-13T06:12:44.964Z",
        "start_index": 217692,
    },
),
Document(
    page_content="Со изградбата на црквата „Света Петка“, галичаните уште еднаш покажаа како сами ги градат своите светилишта, како сами сликаат фрески и икони, покажаа како на својата татковина и на својот народ се подарува драгоцен духовен архитектонски објект со посебна изворност, со творечки израз, со рационалност и со амбиенталност.\r\nУште еднаш галичаните ја докажаа својата храброст, достоинственост, мудрост и дипломатија, потврдија како се чуваат и како се сакаат сопствената вера и народ и како се бори за нив. Ја надмудрија силната турска и тврдокорна власт и со ферман добија дозвола да изградат верски објекти со невообичаени димензии за тоа време.",
    metadata={
        "title": "Galichnik vo mojot kolazh",
        "when": "2024-06-09T19:24:35.112Z",
        "start_index": 75323,
    },
),
Document(
    page_content="C:\\Users\\Administrator\\Desktop\\s5.jpg
\r\n\r\n\r\n\r\n\r\n\r\nЦрковниот комплекс „Света Петка“, во далечната 1809 година, бил изграден со креативно чувство за избор на местото (локацијата), каде што доминира духовна и природна амбиентална средина. Тоа е центарот на Галичник (во тоа време), односно во најстарото маало Станишмало-Долномало, поточно под патот што ги поврзувал плоштадот и панаѓурот во правец од Шулевци и од Шаклевци кон Једиковци и кон Чапаровци. Галичките градители со духовно и со интуитивно

```



```
    },
  ),
  Document(
    page_content="Историското гледање за изградбата на црквата во Галичник е показ за наследството во секоја смисла на зборот, зашто историјата треба да ја гледаме како творечки процес и како дел од сопственото постоење. Всушност, со изградбата на оваа црква, како и со изградбата на подоцнежните цркви во другите мијачки населби, ќе се создадат услови за формирање автономна мијачка - македонска црква. Таа во втората половина на 19 век ќе биде најтрајниот и најцврстиот браник за запирање на навлегувањето на туѓите црковни влијанија и на туѓите пропаганди од страна на сестринските цркви. Предводници на формирањето на таа автономна црква, би ја нарекол црковна матица, ќе бидат токму мијачките првенци, свештени лица и учители од Лазарополе, од Галичник и од другите рекански села, воспитани и пораснати со духот на игуменот и архимандрит Арсениј од Галичник.",
    metadata={
      "title": "Galichnik vo mojot kolazh",
      "when": "2024-06-09T19:24:35.112Z",
      "start_index": 74476,
    },
  ),
  Document(
    page_content="Црковниот комплекс „Света Петка“ во Галичник, кој се гради и опстојува во клучното време за буђење на македонскиот народ, на почетокот на 19 век, се појавува како карактеристичен културен и просветителен стожер, во чие дејствување се рефлектираат и се пројавуваат борбата за самостојна црква, различни настани и личности, значајни за македонската богослужба, архитектура, уметност и просвета во соодветниот историски период. „Света Петка“ е галичкото битие и душа на Мијачијата, виста кон далечното минато, од која се простира најраскошниот недоглед на преродбата на Македонците во 19 век. „Света Петка“ е светилиште во чии ѕидини и покриви е никната, живеела и траела 200 години галичката мисла, мијачката духовност, односно македонската култура.",
    metadata={
      "title": "Galichnik vo mojot kolazh",
      "when": "2024-06-09T19:24:35.112Z",
      "start_index": 79615,
    },
  ),
  Document(
    page_content="Еден од иницијаторите и донаторите за изградба на црквата во Галичник е прочуениот архимандрит Арсениј - игумен во манастирот „Свети Јован Бигорски“. Ова свето лице со редок ум и идеја верно да му служи на Бога, го обмислил прекрасниот и многусодржаен црковен комплекс „Света Петка“ во
```

родниот Галичник. Раководејќи 32 години со манастирот, од 1807 до 1839 година, тој нераскинливо бил поврзан со богослужбата и со растежот на парохиската црква во Галичник. Со својот возвишен, молитвен и креативен ум, придонел двете небоземни светилишта да ги доживеат своите најсветли мигови и засекогаш да останат патоказ во разбирањето на верата и во воспоставувањето на посебен однос кон минатото на еден народ.\r\nНа првата страна од споменикот за градба на црквата е запишано следното:",

```
    metadata={
      "title": "Galichnik vo mojot kolazh",
      "when": "2024-06-09T19:24:35.112Z",
      "start_index": 72397,
    },
```

```
  ),
```

```
  Document(
```

```
    page_content="- црквата Св. Петка во Галичник - \r\n24 \r\nАрсениј од Галичник\r\nРезбите, живописот и опремувањето \r\nво манастирот Св. Јован Бигорски \r\nза време на игуменот Арсениј од Галичник \r\nПо проширувањето на манстирската црква со заслуга на монахот Митрофан кој починал во 1807 година со манастирот раководи неговиот современик, игуменот и архимандрит Арсениј. Тогаш започнува внатрешното уредување и декорација на манстирската црква. Потекнувајќи од татковината на копаничарството и зографството - Галичник, Арсениј во себе го носи исконскиот порив и чувство за уметност и тоа всушност ќе биде причина да знае да ги ангажира најдобрите мијачки резбари кои што ги платил за изработка на нов иконостас, балдахин, голем змиски крст, владички и игуменски столови, амвон, и столови на јуж",
```

```
    metadata={
      "title": "Arsenij Kniga",
      "when": "2024-05-14T18:56:42.636Z",
      "start_index": 20232,
    },
```

```
  ),
```

```
  Document(
```

```
    page_content="и во Србија и на Косово, посебно во градовите Урошевац и Призрен. Во Македонија работи во црквите во Штип, Кочани, Струга, Гостивар, Ростуше, Кичево, Скопје и други места. Посебно треба да се истакне неговото дело во црквата „Свети Петар и Павле“, комплетно насликана со фрески, иконостас и икони. Тоа е дело со посебен колорит кое и ден-денес ги плени погледите на секој посетител на светиот храм во Галичник. Будеше длабоки емоции посебно кај нас, кои бевме",
```

```
    metadata={
      "title": "Galichka Mozaichna Soznajba",
      "when": "2024-06-13T06:12:44.964Z",
      "start_index": 216561,
    },
```


во Русија, на Емпаер стејт билдинг во Менхетен.“ Вака пишува Луј Адамич во далечната 1931 година за градителите од Галичник.”,

```
    metadata={
        "title": "Galichnik vo mojot kolazh",
        "when": "2024-06-09T19:24:35.112Z",
        "start_index": 88594,
    },
),
Document(
    page_content="„Света Петка“ во близина на Старо Село. Во тоа со свои средства учествуваа тој и неговите браќа. Беше претседател на Месната заедница Галичник, кога реализира многу проекти и достоино го афирмираше Галичник. Постојано обезбедува средства од разни донатори за одржување на Галичката свадба. Ја сочува и обнови својата родна куќа и таа денес е најубава и најголема камена палата во Галичник од непроценлива вредност. Со грижата за најблиската и подалечна фамилија, со смиреноста, остроумност и комуникација со секого, потсетува на своите предци-галичките првенци.\r\nФилипоски Јосе - директор на една од поуспешните фирми за ортопедски помагала во „Славеј“, Скопје.\r\nФилипоски Кире - сточар од познатата фамилија Филипоски, мирен и кроток човек, искрен другар и работник, еден од најдобрите бригадири на бачилата на Бистра.\r\nФилипоски Косто - сточар и бригадир во бачилата на Бистра.",
    metadata={
        "title": "Galichka Mozaichna Soznajba",
        "when": "2024-06-13T06:12:44.964Z",
        "start_index": 313933,
    },
),
],
"answer": "Црквата „Св. Петар и Павле“ во Галичник е изградена од мијачки градители и луковски каменоресци, завршена е во 1930 година и осветена во 1931 година. Таа се одликува со импозантна камена структура, богат ентериер со орнаменти и одлична акустичност. Црквата е препознатлив белег на Галичник и е важен дел од манифестацијата „Галичка свадба“.",
}
```

Python

```
question = "Кoj ја изградил црквата?"
rag_output = qa_chain.invoke({"input": question, "chat_history": chat_history})
chat_history.extend([HumanMessage(content=question), rag_output["answer"]])
rag_output
```



```

        metadata={
            "title": "Galichnik vo mojot kolazh",
            "when": "2024-06-09T19:24:35.112Z",
            "start_index": 30558,
        },
    ),
    Document(
        page_content="Театарот во Скопје\r\n\r\n\r\nТокму ова сеќавање нека биде и повод, неговите роднокрајци достоино да ја одбележат 120-годишнината од раѓањето на градителот Јордан Стојанчески и 75 години од изградбата на неговиот прекрасен објект црквата „Св. Петар и Павле“ со една пригодна спомен-плоча со името на градителот, годината на раѓањето - 1885 и годината на неговата смрт - 1943, која ќе биде поставена во објектот.\r\n\r\n\r\n\r\n \r\n\r\n\r\n\r\n\r\nЦрквата „Св. Петар и Павле“ во Галичник\r\n\r\n\r\n\r\n \r\n\r\n\r\n\r\n\r\nСтанбена зграда во Скопје\r\n\r\n\r\n\r\n(„Дневник“, 18.3.2006 г.)\r\n_____\r\nМакедонска народна архитектура",
        metadata={
            "title": "Galichnik vo mojot kolazh",
            "when": "2024-06-09T19:24:35.112Z",
            "start_index": 30137,
        },
    ),
    Document(
        page_content="„Св. Петар и Павле“ во својот роден Галичник, која е ремек-дело на мијачката градителска рака и невидено мајсторство на луковските каменорезачи. Таа прекрасна градба чичко Јордан ја градеше три години и беше завршена во 1930 година, а осветена во 1931 година. Плени со својата импозантна камена структура, богатиот ентериер со орнаменти и пред сè, со одличната акустичност. Денес на неговата градба која не е поместена ниту за еден милиметар се воодушевуваат сите посетители.\r\nНормално, морам да ги издвојам и тројцата мои прочуени роднокрајци за кои е многу пишувано и затоа само на кратко ќе се осврнам на нив и нивните дела.",
        metadata={
            "title": "Galichka Mozaichna Soznajba",
            "when": "2024-06-13T06:12:44.964Z",
            "start_index": 217692,
        },
    ),
    Document(
        page_content="Сакам да споменам некои работи за другите двајца славни галичани, едниот градител, другиот зограф и иконописец. Не случајно градоначалникот на Галичник Тихон Крцкоски-Михајлоски, кога требаше да се гради новата црква\r\n„Свети Петар и Павле“, ги покани овие двајца врвни мајстори да ги изведуваат градителските и уметничките работи. Тоа беше еден поинаков стил и

```

во однос на градбата и во однос на декорацијата. За прв пат во мијачкиот крај беше градена црква со цементен малтер, бетонска плоча, а иконостасот не беше дрвен. Растев со нивните дела и нивните човечки добрини.",

```
        metadata={
            "title": "Galichka Mozaichna Soznajba",
            "when": "2024-06-13T06:12:44.964Z",
            "start_index": 215195,
        },
    ),
    Document(
        page_content="C:\\Users\\Administrator\\Desktop\\s5.jpg
\r\n\r\n\r\n\r\n\r\n\r\nЦрковниот комплекс „Света Петка“, во далечната 1809 година, бил
изграден со креативно чувство за избор на местото (локацијата), каде што
доминира духовна и природна амбиентална средина. Тоа е центарот на Галичник (во
тоа време), односно во најстарото маало Станишмало-Долномало, поточно под патот
што ги поврзувал плоштадот и панаѓурот во правец од Шулевци и од Шаклевци кон
Једиковци и кон Чапаровци. Галичките градители со духовно и со интуитивно
чувство и студиозност, го изградиле комплексот на стрмен терен со извонредни
визури кон поблиската и кон подалечната околина. Црквата и другите придружни
објекти формираат едно архитектонско-урбанистичко гнездо, како што би рекол
мојот драг професор и пријател архитектот Петар Муличкоски.",
        metadata={
            "title": "Galichnik vo mojot kolazh",
            "when": "2024-06-09T19:24:35.112Z",
            "start_index": 76448,
        },
    ),
    Document(
        page_content="Со изградбата на црквата „Света Петка“, галичаните
уште еднаш покажаа како сами ги градат своите светилишта, како сами сликаат
фрески и икони, покажаа како на својата татковина и на својот народ се подарува
драгоцен духовен архитектонски објект со посебна изворност, со творечки израз,
со рационалност и со амбиенталност.\r\nУште еднаш галичаните ја докажаа својата
храброст, достоинственост, мудрост и дипломатија, потврдија како се чуваат и
како се сакаат сопствената вера и народ и како се бори за нив. Ја надмудрија
силната турска и тврдокорна власт и со ферман добија дозвола да изградат верски
објекти со невообичаени димензии за тоа време.",
        metadata={
            "title": "Galichnik vo mojot kolazh",
            "when": "2024-06-09T19:24:35.112Z",
            "start_index": 75323,
        },
    ),
    Document(
```



```

        "title": "Galichnik vo mojot kolazh",
        "when": "2024-06-09T19:24:35.112Z",
        "start_index": 46580,
    },
),
Document(
    page_content="Историското гледање за изградбата на црквата во
Галичник е показ за наследството во секоја смисла на зборот, зашто историјата
треба да ја гледаме како творечки процес и како дел од сопственото постоење.
Всушност, со изградбата на оваа црква, како и со изградбата на подоцнежните
цркви во другите мијачки населби, ќе се создадат услови за формирање автономна
мијачка - македонска црква. Таа во втората половина на 19 век ќе биде
најтрајниот и најцврстиот браник за запирање на навлегувањето на туѓите црковни
влијанија и на туѓите пропаганди од страна на сестринските цркви. Предводници
на формирањето на таа автономна црква, би ја нарекол црковна матица, ќе бидат
толку мијачките првенци, свештени лица и учители од Лазарополе, од Галичник и
од другите рекански села, воспитани и пораснати со духот на игуменот и
архимандрит Арсениј од Галичник.",
    metadata={
        "title": "Galichnik vo mojot kolazh",
        "when": "2024-06-09T19:24:35.112Z",
        "start_index": 74476,
    },
),
Document(
    page_content="- црквата Св. Петка во Галичник - \r\n24 \r\n\r\nАрсениј
од Галичник\r\n\r\nРезбите, живописот и опремувањето \r\n\r\nво манастирот Св. Јован
Бигорски \r\n\r\nза време на игуменот Арсениј од Галичник \r\n\r\nПо проширувањето на
манстирската црква со заслуга на монахот Митрофан кој починал во 1807 година
со манастирот раководи неговиот современик, игуменот и архимандрит Арсениј.
Тогаш започнува внатрешното уредување и декорација на манстирската црква.
Потекнувајќи од татковината на копаничарството и зографството - Галичник,
Арсениј во себе го носи исконскиот порив и чувство за уметност и тоа всушност
ќе биде причина да знае да ги ангажира најдобрите мијачки резбари кои што ги
платил за изработка на нов иконостас, балдахин, голем змиски крст, владички и
игуменски столови, амвон, и столови на јуж",
    metadata={
        "title": "Arsenij Kniga",
        "when": "2024-05-14T18:56:42.636Z",
        "start_index": 20232,
    },
),
Document(

```

page_content="Црковниот комплекс „Света Петка“ во Галичник, кој се гради и опстојува во клучното време за будење на македонскиот народ, на почетокот на 19 век, се појавува како карактеристичен културен и просветителен стожер, во чие дејствување се рефлектираат и се пројавуваат борбата за самостојна црква, различни настани и личности, значајни за македонската богослужба, архитектура, уметност и просвета во соодветниот историски период. „Света Петка“ е галичкото битие и душа на Мијачијата, виста кон далечното минато, од која се простира најраскошниот недоглед на преродбата на Македонците во 19 век. „Света Петка“ е светилиште во чии ѕидини и покриви е никната, живеела и траела 200 години галичката мисла, мијачката духовност, односно македонската култура.",

```
metadata={  
  "title": "Galichnik vo mojot kolazh",  
  "when": "2024-06-09T19:24:35.112Z",  
  "start_index": 79615,  
},
```

```
),
```

```
Document(  
  page_content="и во Србија и на Косово, посебно во градовите
```

Урошевац и Призрен. Во Македонија работи во црквите во Штип, Кочани, Струга, Гостивар, Ростуше, Кичево, Скопје и други места. Посебно треба да се истакне неговото дело во црквата „Свети Петар и Павле“, комплетно насликана со фрески, иконостас и икони. Тоа е дело со посебен колорит кое и ден-денес ги плени погледите на секој посетител на светиот храм во Галичник. Будеше длабоки емоции посебно кај нас, кои бевме",

```
metadata={  
  "title": "Galichka Mozaichna Soznajba",  
  "when": "2024-06-13T06:12:44.964Z",  
  "start_index": 216561,  
},
```

```
),
```

```
Document(  
  page_content="„Света Петка“ во близина на Старо Село. Во тоа со
```

свои средства учествуваа тој и неговите браќа. Беше претседател на Месната заедница Галичник, кога реализира многу проекти и достоино го афирмираше Галичник. Постојано обезбедува средства од разни донатори за одржување на Галичката свадба. Ја сочува и обнови својата родна куќа и таа денес е најубава и најголема камена палата во Галичник од непроценлива вредност. Со грижата за најблиската и подалечна фамилија, со смиреноста, остроумност и комуникација со секого, потсетува на своите предци-галичките првенци.\r\nФилипоски Јосе - директор на една од поуспешните фирми за ортопедски помагала во „Славеј“, Скопје.\r\nФилипоски Кире - сточар од познатата фамилија Филипоски, мирен и кроток човек, искрен другар и работник, еден од најдобрите бригадири на

бачилата на Бистра.\r\nФилипоски Косто - сточар и бригадир во бачилата на Бистра.",

```
    metadata={
      "title": "Galichka Mozaichna Soznajba",
      "when": "2024-06-13T06:12:44.964Z",
      "start_index": 313933,
    },
```

```
  ),
```

```
  Document(
```

```
    page_content="по обем најголемото црковно кубе во Романија. Познат е и градителот Стамат Алаутоски, кој гради, исто така, низ Балканот, а особено за истакнување е куќата на неговата фамилија во Галичник изградена како камена палата со атриум, бањи, камини и сл. Кога зборуваме за архитектонското творештво не смее да се заборава улогата на големиот ктитор, игуменот и архимандритот Арсениј од Галичник, кој е роден во шеесеттите години на 18 век, а починал во 1839 година, под чие раководство и со чија идеја се изградени манастирските конаци во „Св. Јован Бигорски“ и голем број други објекти, ентериерот на црквата, трпезариите, тогаш, всушност, манастирот ги доживува своите ѕвездени мигови. Ја гради црквата во Рајчица и парохиската црква „Св. Петка“ во Галичник.",
```

```
    metadata={
      "title": "Galichnik vo mojot kolazh",
      "when": "2024-06-09T19:24:35.112Z",
      "start_index": 89883,
    },
```

```
  ),
```

```
  Document(
```

```
    page_content="Впрочем, не е за изненадување галичката архитектура, ако се знае дека: „Сидарите и каменорезбарите од Галичник работеле на палатите во Версај, на мостовите во Будимпешта, на катедралата ‘Свети Стефан’ во Виена, на големите хотели што гледаат кон Неаполскиот Залив, на кралските палати во Белград и Букурешт, на Мормонскиот храм во Јутаха, во колосалниот Дњепрострој во Русија, на Емпаер стејт билдинг во Менхетен.“ Вака пишува Луј Адамич во далечната 1931 година за градителите од Галичник.",
```

```
    metadata={
      "title": "Galichnik vo mojot kolazh",
      "when": "2024-06-09T19:24:35.112Z",
      "start_index": 88594,
    },
```

```
  ),
```

```
  Document(
```

```
    page_content="Една од прочуените градителски тајфи е онаа на Ѓорѓија Пулевски (1818 - 1894), која во Романија гради мостови и палати. Градителот Кара Иван работел на утврдувањето на градот Видин во Бугарија. Илија
```

Јовановски градел куќи во Солун. Стојан Томовски (1862 - 1934) градел куќи во Романија и изработувал ентериери, изградил модерна камена палата за својата фамилија. Неговиот син Јордан Стојанчески (1885 - 1943) е градител на црквата во Галичник „Св. Петар и Павле“, учесник во изградбата на стариот театар во Скопје, градител на станбени куќи во Скопје и на пругата Велес - Штип. Фамилиите Јурукоски од Тресонче, Колоски и Личеноски од Галичник, изградиле стотина цркви, повеќето во Романија. Особено се истакнуваат Кузман Колоски и Матеј-Моте Велов Личеноски, за кого се знае дека го конструирал, по обем најголемото црковно кубе во Романија. Познат е и градителот Стамат Алаутоски, кој гради, исто така, низ Балканот, а особено за истакнување е куќата на неговата фамилија во Галичник изградена",

```
metadata={
  "title": "Galichnik vo mojot kolazh",
  "when": "2024-06-09T19:24:35.112Z",
  "start_index": 89086,
},
```

```
),
```

```
Document(
```

```
  page_content="Сè уште се надевам дека со сесрдна помош на донатори од бизнис-групацјата, како и со поддршка на учениците на Личеноски, овој меморијален музеј ќе биде реализиран. За овој објект се потребни мали средства (колку една помала скулптура), но тој ќе биде голема надградба за нашето културно архитектонско наследство и доказ дека со слободен израз се доаѓа до објекти што ќе нè надминат и што ќе бидат архитектура на нашето време и на новиот милениум.\r\n\r\n\r\n(„Репер“, 2013 г.)\r\n_____\r\nДвесте години од изградбата на првата парохиска црква во Мијачијата „Света Петка“ во Галичник",
```

```
metadata={
  "title": "Galichnik vo mojot kolazh",
  "when": "2024-06-09T19:24:35.112Z",
  "start_index": 69620,
},
```

```
),
```

```
Document(
```

```
  page_content="Откако закрепнал, Партениј од Галичник заминал во Цариград, каде што од руската влада бил назначен за духовен претставник на руското претставништво. По формирањето на богословското училиште во 1851 година во Зографскиот манастир, Партениј бесплатно им предавал на десет сиромашни ученици. Покрај црковните предмети, тој предавал граматика и тука е неговиот зародок за учебничарската и на граматичката ангажираност. Кон крајот на 1853 година, од страна на Цариградската патријаршија, Партениј бил назначен за професор по црковнословенски јазик во богословското училиште на островот Халки кај Цариград, а по две години предавање тој во јавноста излегол со своите први статии од областа на јазикот и со свои учебници на македонски јазик.",
```

```

        metadata={
            "title": "Galichnik vo mojot kolazh",
            "when": "2024-06-09T19:24:35.112Z",
            "start_index": 204395,
        },
    ),
],
"answer": "Црквата „Св. Петар и Павле“ во Галичник ја изградил Јордан
Стојанчески, познат градител од Галичник.",
}

```

Listing 3.1: **Example of Context search improvements by handling the chat history in Mutavchain**

Proper handling of the chat history in order to construct more queries for retrieving the most appropriate answers is one way to improve the overall performance of the **RAG** application as shown in **Listing 3.1** for **Mutavchain**. There is also another possible way of **Context search** utilization in terms of having to validate the correctness of the content if the user provides additional information as feedback or counter answer. Implementing this brings certain risks because the user might possess wrong information, make typos or will simply lie.

3.2 Temporal RAG

What if the **Data Source** possesses content that keeps information about events that reoccurred through time? Moreover, if the content is about documentation that contains matter that keeps getting updated over the years with the old copies remaining, how will you get an answer based on the desired copy? The logical solution will be to include the period in the prompt when the content was published but that will work only if there is a time attribute attached to the content copy itself. This problem can be solved by a **RAG** variant which incorporates time-sensitive or temporal information into the retrieval and generation processes. It allows the system to handle information that evolves over time, such as news, events, updates in a particular domain, or policies that change periodically. In **Temporal RAG**, the system retrieves relevant documents or information while taking into account the time relevance of the data, ensuring that the generated responses are accurate and current.

Temporal RAG can be treated as a special case of **Data Source**, **Prompt** and **Vector Store** improvements, combined with improvements on the **Conversational Interface** where specific UI elements can be added for handling the time constraint and integrate it in the user query itself. Lack of enhancements on the **Conversational Interface** of the **RAG** application means that the burden of handling the time will fall on the user's shoulders that must manually prompt for a specific time period in order to hope to get the desired answer.

```
Python
loader = ConfluenceLoader(
    url=CONFLUENCE_SPACE_NAME, username=CONFLUENCE_USERNAME,
    api_key=CONFLUENCE_API_KEY
)

docs = loader.load(
    space_key=CONFLUENCE_SPACE_KEY, limit=100, max_pages=500,
    keep_markdown_format=True
)
```

```
Python
print("Content: \n ----- \n" + docs[-1].page_content)
```

```
Unset
Content:
-----
Owned by Vellimir Graorkoski, created
with a template

Feb 01, 2024
4 min read

5 people viewed

# Table of contents

- RAG and Agent in the Machine Learning lifecycle
- LangChain overview
- Talking to Athena
- Challenges

*
*

This document describes the experience of creating LangChain Agent for Athena
data analytics service. An Agent is used as an interface for communication
mainly with databases or storage, but also with APIs or any kind of other
software that exposes data. In a way, it represents a special case of RAG
(Retrieval Augmented Generation) paradigm by which the LLMs (Large Language
```

Models) are becoming “aware” of the data that they have not been trained on, thus allowing possible bypassing the fine-tuning and the training processes.

More high level overview of this concept known as of Talk to Data, can be found on the following slides:

Listing 3.2: Loaded content with time in ConfluenceRAG

One **Data Source** that keeps time related data is **Confluence**. Remember the **ConfluenceRAG** application? As I describe it then, one benefit that it brings is that it helps the other employees to get specific answers about portions of documentation that otherwise is hard to locate or it takes minutes of reading in order to determine it. However, the company that owned the **Confluence** space for years, had procedures and algorithms that changed over time but the pattern was to keep the old versions for tracking purposes. This creates a potential problem when the embedding of an outdated documentation is closer to the user’s query than the latest documentation content that is relevant. Since **Confluence** pages keep time of their publication/update as shown in Listing , it can be embedded along with the rest of the content and make a difference when the retrieval occurs on the **Vector Store** to get the latest relevant information.

Handling the time integrated in the retrieval depends on the type of **Vector Store** as well. Some **Vector Stores** have better predispositions for retrieving time related queries such as **pgvector** combined with **Timescale Vector** although it takes additional efforts to properly set it up. Another high-performance **Vector Store** that supports handling the time metadata is **Qdrant**.

Python

```
def setup_temporal_retriever(_docs):
    text_splitter = RecursiveCharacterTextSplitter(
        chunk_size=1000, chunk_overlap=200, add_start_index=True
    )
    splits = text_splitter.split_documents(_docs)

    vectorstore = Chroma.from_documents(documents=splits,
embedding=OpenAIEmbeddings())
    # retriever = vectorstore.as_retriever(search_type="similarity",
search_kwargs={"k": 6, 'filter': {'when': '2024-06-24T09:59:51.975Z'}})
    retriever = vectorstore.as_retriever(
        search_type="similarity",
        search_kwargs={"k": 6, "filter": {"when": {"$gt": "2024-06-23"}}},
    )
```

```
return retriever
```

Python

```
def setup_temporal_rag_chain(_retriever):  
    llm = ChatOpenAI(model_name="gpt-4o-mini", temperature=0)  
  
    template = """Use the following pieces of context to answer the question at  
the end.  
If you don't know the answer, just say that you don't know, don't try to  
make up an answer.  
Use three sentences maximum and keep the answer as concise as possible.  
If there is no time specified in the question, order the results from the  
query by the time metadata of the document.  
  
{context}  
  
Question: {question}  
  
Helpful Answer: """  
    custom_rag_prompt = PromptTemplate.from_template(template)  
  
    rag_chain = (  
        {"context": retriever | format_docs, "question": RunnablePassthrough()}  
        | custom_rag_prompt  
        | llm  
        | StrOutputParser()  
    )  
    return rag_chain
```

Python

```
temporal_qa_chain = setup_turbo_llm_and_temporal_qa_chain(temporal_retriever)  
temporal_qa_chain.invoke("Која содржина датира од јуни, 2024 година?")
```

Unset

```
'Не знам.'
```

Python

```
temporal_qa_chain.invoke("What do you know about any context close to that date?")
```

Unset

```
'The context mentions "ДЕЛ ОД МОЕТО ТВОРЕШТВО ИНСПИРИРАНО, ПРЕД СЕЌ, ОД МОЈОТ ГАЛИЧНИК," which translates to "A PART OF MY CREATION INSPIRED, ABOVE ALL, BY MY GALICHNIK." It also references Aleksandar-Aleksa Ćinoski, a key figure in Macedonian photography, with a date of May 15, 2013. The repeated elements suggest a focus on personal artistic inspiration and the significance of Galichnik in the creator's work.'
```

Listing 3.3: Example of time handling in Mutavchain

I will leave you to experiment with different prompts that can help handle the time better because this belongs to the prompt improvements. But I will refer to one negative example from **Mutavchain** with **Chroma Vector Store**, shown in **Listing 3.3** where the time constraint is set on the **Vector Store** and the **Generator** but the retrieved results are not improved resulting in an unsatisfying answer because there was no indication that the embedded document contained the corresponding time. Also the way the time constraint was set on the **Vector Store** level does not scale because it implies resetting the retriever on each query that contains a different time.

3.3 Solving the contradictions

In practice, the **Data Source** can hide contradictory information, a state that is hard to detect before the user interaction with the **RAG** application occurs. The additional problem is how to determine which of the pieces of information is wrong. For simplicity purposes I am going to refer to a scenario where one piece is a contradiction to the others, otherwise there can be more than one contradiction.

Python

```
### A sentence stating wrongly that Arsenij was in charge of Zrze monastery is included among a significant pool of sentences that Arsenij was the head of Bigorski monastery
```

Python

```
rag_chain.invoke("Со кој манастир раководел Арсениј?")
```

Unset

'Арсениј раководел со Бигорскиот манастир од 1807 до 1839 година. За време на неговото игуменување, манастирот доживеал значајни уметнички и архитектонски подобрувања. Тој ангажирал најдобрите мајстори на резбата и живописот, што резултирало со блескава внатрешност на манастирската црква.'

Python

```
rag_chain.invoke("Со кои манастири раководел Арсениј?")
```

Unset

'Арсениј раководел со Бигорскиот манастир од 1807 до 1839 година. Тој бил одговорен за бројни архитектонски и уметнички зафати, вклучувајќи ангажирање на мајстори за резба и живопис. Неговиот придонес значително ја подобрил духовната и културната вредност на манастирот.'

Python

```
rag_chain.invoke("На кој манастир Арсениј бил архимандрит?")
```

Unset

'Арсениј бил архимандрит во манастирот „Свети Јован Бигорски“. Тој управувал со манастирот од 1807 до 1839 година. Неговото раководство значително придонело за развојот и реafirмацијата на манастирот.'

Python

```
rag_chain.invoke("Дали Арсениј раководел со манастирот Зрзе?")
```


candidates, with the lower being the score, the higher the relevancy the results, hence the detected contradiction has to have lower values. **Listing 3.6** shows locating the document where the wrong sentence is included having lower score. Like in the previous example, the list of the retrieved documents is ordered by the score, starting from the highest. Here however, the potentially wrong information will be found in the last part of the list. Therefore I recommend when looking for possible contradictions, you can start from the end of the list.

The previous techniques of locating the contradictory data are based on the assumption that the wrong information rarely appears within the content of the **Data Source**. Although that often happens, there are edge cases where due to accumulation of incorrect data information, the correct facts can be rare. One example can be documents affected by politics that interfere with historical facts such as the fake theory elaboration that the earth is flat. In those cases by applying the previously mentioned strategies we will remove the rare pieces of information that are relevant. The conclusion is that solving the contradictions sometimes cannot be fully prevented i.e. if someone elaborates deeply on a particular subject no matter how wrong it is, the relevance score can still be higher.

4. Optimization

Optimizing a **RAG** application involves improvements in terms of performance which we mainly discussed in the previous sections. In this chapter I will try to show you how to detect those performance issues while focusing on **RAG** evaluation. Furthermore I will try to provide some tips for reducing the application's latency and costs.

4.1 Detecting issues

When testing your **RAG** application before deploying on production, a simple way to validate the generated output for each test query is to check the retrieved documents on either **Vector Store** or on **Retriever** level. It requires manual checking at the end of the day but you can write your code in a way that it helps your eye to detect potential problems. **Listing 4.1** shows examples of listing documents in **Mutavchain** from the **Vector Store** and from the **Retriever**.

Python

```
def pretty_print_docs_vectorstore(docs):
    print(
        f"\n{'-' * 100}\n".join(
            [f"Document {i+1}:\n\n" + d[0].page_content for i, d in
             enumerate(docs)]
        )
    )
```

Python

```
def pretty_print_docs_retriever(docs):
    print(
        f"\n{'-' * 100}\n".join(
            [f"Document {i+1}:\n\n" + d.page_content for i, d in
             enumerate(docs)]
        )
    )
```

Python

```
search_result = vectorstore.similarity_search("Со кои манастири раководел  
Арсениј?", k=5)  
pretty_print_docs_vectorstore(search_result)
```

Unset

Document 1: Потекнувајќи од татковината на копаничарството и зографството - Галичник, Арсениј во себе го носи исконскиот порив и чувство за уметност и тоа всушност ќе биде причина да знае да ги ангажира најдобрите мијачки резбари кои што ги платил за изработка на нов иконостас, балдахин, голем змиски крст, владички и игуменски столови, амвон, и столови на јужната, западната и северната страна во наосот на црквата за веропоклонците, потоа певници и друго. Не само на почетокот, кога Арсениј раководи со манастирот, туку и понатаму ќе биде најзаслужен за ангажирањето на големите мајстори на резбата и живописот. Во времето на Арсениј внатрешноста на манастирската црква и другите простории ќе блеснат засекогаш во невиден сјај од делата на мијачките уметници. По украсувањето на манастирската црква со балдахинот, кај Арсениј се јавува уште посилен жед за опремување на црквата и патува за Света Гора и Цариград каде набавил крстови, свеќници и нарачал голем кивот.

----- Document 2: ната, западната и северната старна во наосот на црквата за веропоклонците, потоа певници и друго. Не само на почетокот кога Арсениј раководи со манастирот, туку и понатаму ќе биде најзаслужен за ангажирањето на големите мајстори на резбата и живописот. Во времето на Арсениј внатрешноста на манастирската црква и другите простории ќе блеснат засекогаш во невиден сјај од делата на мијачките уметници. Во 1810 година Арсениј со нарачката на балдахинот од Елбасан, на манастирот ќе му донесе едно извонредно уметничко дело од мајсторите Филиповци од Осој кои во тоа време работеле на територијата на Матија, денешна Албанија. Таму мијачките копаничари имале сопствена работилница, а манастирот Бигор ски свои имоти со насади од маслинки, лозја и друго. Ова зборува за живата врска на манастирот СВ. Јован Бигорски со Елбасан. Архимандритот Арсениј за првата манастирска резбарска работа 25 Арсениј од Галичник

----- Document 3: духовните чеда на Бигорскиот манастир ќе запише: "Архимандритот Арсениј е навистина еден од ретките игумени судејќи според зачуваните документи, кој се потрудил, повику вајќи се на народното предание раширено сред народот од мијачкиот крај, да го "осветли" и приспомне темното минато на историјата на манастирот - неговото подалечно и поблиско минато" Од раката на Арсениј за 32 години игуменување останати се голем број записи и документи за манастирот кои се зачувани до ден денес како што е на пример Великиот поменик или писмото за собирање на прилог за манастирот и др.

----- Document 4: Арсениј ќе управува со манастирот од 1807 до 1839 година, поточно до 27 јуни 1839 година кога тој умира. Во досегашните научни истражувања нема прецизни податоци каде бил погребан Арсениј, како впрочем и за другите игумени. Можно е тој да е погребан во родниот Галичник, што било негова последна желба, во црквата за која дал идеја за градба и голем придонес со сопствени средства за реализација. Имено, во дворот на старата црква „Св. Петка“ на северниот ѕид од авлијата се наоѓа гроб, каде што на надгробната плоча е врежано името Арсениј. Можеби овој податок до кој дојдов ќе биде предизвик за нашите истражувачи да го потврдат местото каде што почива најзаслужниот игумен и архимандрит Арсениј од Бигорскиот манастир.

----- Document 5: Арсениј превземал бројни нови архитектонски - градежни зафати и содржини во манастирот. Исто така тој ги реafirмирал ликовната зографска префинетост, копаничарството-резбата, книжевната традиција, изградбата на нови духовни храмови, проширување на манастирските имоти и секако, пред сè, невидената теолошка духовна продлабоченост која ги надминувала дотогашните дострели на игумените. Завршен резултат од игуменувањето на Арсениј е восхитувањето и почитувањето на сите посетители, кои откриваат духовни вредности, а денес истите ги споредуваме барајќи ја во нив силната творечка вистина и инспирација за слични потфати ако воопшто се повторно можни. Ваква богата и величествена духовна и градителска култура можеше да ни остане во наследство само благодареејќи на Арсениј, духовникот кој знаеше и имаше чувство да ги ангажира и да ги организира најдобрите градители, иконописци, резбари, игуменот кој имаше визији за растежот на бигорскиот манастирски комплекс.

Python

```
retriever = vectorstore.as_retriever(search_type="similarity",  
search_kwargs={"k": 5})
```

```
retriever_results = retriever.invoke("Со кои манастири раководел Арсениј?")  
pretty_print_docs_retriever(retriever_results)
```

Unset

Document 1: Потекнувајќи од татковината на копаничарството и зографството - Галичник, Арсениј во себе го носи исконскиот порив и чувство за уметност и тоа всушност ќе биде причина да знае да ги ангажира најдобрите мијачки резбари кои што ги платил за изработка на нов иконостас, балдахин, голем змиски крст, владички и игуменски столови, амвон, и столови на јужната, западната и северната страна во наосот на црквата за веропоклонците, потоа певници и друго. Не само на почетокот, кога Арсениј раководи со манастирот, туку и

понатаму ќе биде најзаслужен за ангажирањето на големите мајстори на резбата и живописот. Во времето на Арсениј внатрешноста на манастирската црква и другите простории ќе блеснат засекогаш во невиден сјај од делата на мијачките уметници. По украсувањето на манастирската црква со балдахинот, кај Арсениј се јавува уште посилен жед за опремување на црквата и патува за Света Гора и Цариград каде набавил крстови, свеќници и нарачал голем кивот.

----- Document 2: ната, западната и северната старна во наосот на црквата за веропоклонците, потоа певници и друго. Не само на почетокот кога Арсениј раководи со манастирот, туку и понатаму ќе биде најзаслужен за ангажирањето на големите мајстори на резбата и живописот. Во времето на Арсениј внатрешноста на манастирската црква и другите простории ќе блеснат засекогаш во невиден сјај од делата на мијачките уметници. Во 1810 година Арсениј со нарачката на балдахинот од Елбасан, на манастирот ќе му донесе едно извонредно уметничко дело од мајсторите Филиповци од Осој кои во тоа време работеле на територијата на Матија, денешна Албанија. Таму мијачките копаничари имале сопствена работилница, а манастирот Бигор ски свои имоти со насади од маслинки, лозја и друго. Ова зборува за живата врска на манастирот СВ. Јован Бигорски со Елбасан. Архимандритот Арсениј за првата манастирска резбарска работа 25 Арсениј од Галичник

----- Document 3: духовните чеда на Бигорскиот манастир ќе запише: “Архимандритот Арсениј е навистина еден од ретките игумени судејќи според зачуваните документи, кој се потрудил, повику вајќи се на народното предание раширено сред народот од мијачкиот крај, да го “осветли” и приспомне темното минато на историјата на манастирот – неговото подалечно и поблиско минато” Од раката на Арсениј за 32 години игуменување останати се голем број записи и документи за манастирот кои се зачувани до ден денес како што е на пример Великиот поменик или писмото за собирање на прилог за манастирот и др.

----- Document 4: Арсениј ќе управува со манастирот од 1807 до 1839 година, поточно до 27 јуни 1839 година кога тој умира. Во досегашните научни истражувања нема прецизни податоци каде бил погребан Арсениј, како впрочем и за другите игумени. Можно е тој да е погребан во родниот Галичник, што било негова последна желба, во црквата за која дал идеја за градба и голем придонес со сопствени средства за реализација. Имено, во дворот на старата црква „Св. Петка“ на северниот ѕид од авлијата се наоѓа гроб, каде што на надгробната плоча е врежано името Арсениј. Можеби овој податок до кој дојдов ќе биде предизвик за нашите истражувачи да го потврдат местото каде што почива најзаслужниот игумен и архимандрит Арсениј од Бигорскиот манастир.

----- Document 5: Арсениј превземал бројни нови архитектонски - градежни зафати и содржини во манастирот. Исто така тој ги реafirмирал ликовната зографска префинетост, копаничарството-резбата, книжевната традиција,

изградбата на нови духовни храмови, проширување на манастирските имоти и секако, пред сè, невидената теолошка духовна продлабоченост која ги надминувала дотогашните дострели на игумените. Завршен резултат од игуменувањето на Арсениј е восхитувањето и почитувањето на сите посетители, кои откриваат духовни вредности, а денес истите ги споредуваме барајќи ја во нив силната творечка вистина и инспирација за слични потфати ако воопшто се повторно можни. Ваква богата и величествена духовна и градителска култура можеше да ни остане во наследство само благодареејќи на Арсениј, духовникот кој знаеше и имаше чувство да ги ангажира и да ги организира најдобрите градители, иконописци, резбари, игуменот кој имаше визији за растежот на бигорскиот манастирски комплекс.

Listing 4.1: Using suitable formatting functions for listing the documents from the Vector Store and from the Retriever

Besides the **Vector Store** and the **Retriever**, there can be other **RAG** units prone to optimization as I already gave examples in the previous book sections. After going live with your **RAG** application, you cannot expect all the users to act like **SMEs**, nor to report the issues immediately if they somehow spot them. For that purpose you need to enable logging or tracing on each **RAG** unit where possible. Some **RAG** libraries and frameworks support it out of the box. If the **RAG** framework does not provide logging on some of the units, then you have to invest some effort into writing your own code for logging purposes.

Python

```
os.environ['LANGCHAIN_TRACING_V2'] = 'true'  
os.environ['LANGCHAIN_ENDPOINT'] = 'https://api.smith.langchain.com'  
os.environ['LANGCHAIN_PROJECT'] = 'interag'  
os.environ['LANGCHAIN_API_KEY'] = getpass("LANGCHAIN_API_KEY")  
  
retriever.invoke("Which team members from the company provider side  
should take part in the problem-framing discussions?")
```

Unset

```
[Document(metadata={'description': 'The benefits from using MLOps software for  
data science have been addressed and recognized for years, reg', 'language':  
'en', 'source':  
'https://www.intertec.io/resources/blogs/how-building-an-open-source-ml-ops-pla  
tform-on-aws-can-improve-your-company-efficiency', 'title': 'How building an  
open-source MLOps platform on AWS can improve your companyâ\x80\x99s
```

```

efficiency'}, page_content='Usually after recognizing the theoretical
advantages of using MLOps, when an inexperienced data scientist wants to use it
in its everyday work, it turns to small open-source software tools that have a
wide community support, and on top of it can be used on their local
environments or within their notebooks. In general, mainly for personal
purposes. '), Document(metadata={'content_type': 'text/html', 'description':
'The benefits from using MLOps software for data science have been addressed
and recognized for years, reg', 'language': 'en', 'source':
'https://www.intertec.io/resources/blogs/how-building-an-open-source-ml-ops-pla
tform-on-aws-can-improve-your-company-efficiency', 'title': 'How building an
open-source MLOps platform on AWS can improve your company's efficiency'},
page_content='Usually after recognizing the theoretical advantages of using
MLOps, when an inexperienced data scientist wants to use it in its everyday
work, it turns to small open-source software tools that have a wide community
support, and on top of it can be used on their local environments or within
their notebooks. In general, mainly for personal purposes. '),
Document(metadata={'content_type': 'text/html', 'description': 'The benefits
from using MLOps software for data science have been addressed and recognized
for years, reg', 'language': 'en', 'source':
'https://www.intertec.io/resources/blogs/how-building-an-open-source-ml-ops-pla
tform-on-aws-can-improve-your-company-efficiency', 'title': 'How building an
open-source MLOps platform on AWS can improve your company's efficiency'},
page_content='Usually after recognizing the theoretical advantages of using
MLOps, when an inexperienced data scientist wants to use it in its everyday
work, it turns to small open-source software tools that have a wide community
support, and on top of it can be used on their local environments or within
their notebooks. In general, mainly for personal purposes. '),
Document(metadata={'content_type': 'text/html', 'description': 'The benefits
from using MLOps software for data science have been addressed and recognized
for years, reg', 'language': 'en', 'source':
'https://www.intertec.io/resources/blogs/how-building-an-open-source-ml-ops-pla
tform-on-aws-can-improve-your-company-efficiency', 'title': 'How building an
open-source MLOps platform on AWS can improve your company's efficiency'},
page_content='Usually after recognizing the theoretical advantages of using
MLOps, when an inexperienced data scientist wants to use it in its everyday
work, it turns to small open-source software tools that have a wide community
support, and on top of it can be used on their local environments or within
their notebooks. In general, mainly for personal purposes.')]

```

Listing 4.2: Enabling LangSmith tracing in InterAG

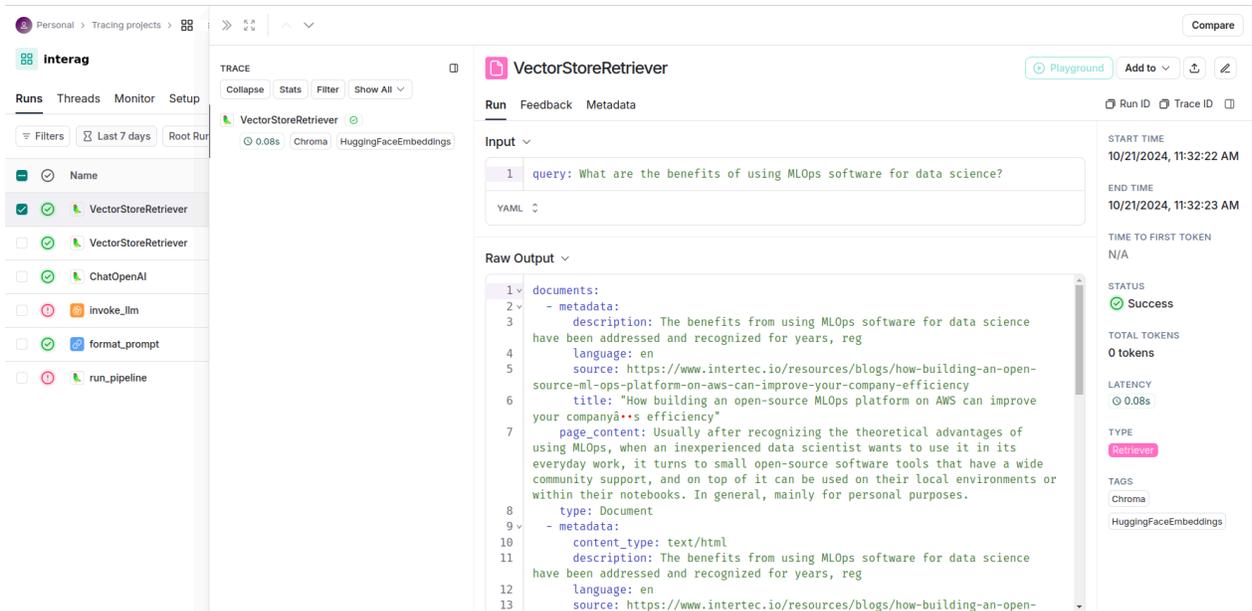


Figure 4.1: LangSmith dashboard for tracing the retriever in InteRAG

One library that supports tracing out of the box is **LangChain** through its developer platform **LangSmith**. **Listing 4.2** shows how to enable this from your code base and **Figure 4.1** presents a screenshot of how the tracing looks like on the **LangSmith** dashboard for the **InteRAG** project. With these examples most of the possibilities for automating the detection of potential issues are exploited. Afterwards you must observe the logs looking for wrong content and subsequently initiate the evaluation process.

```
Python
add_routes(app, qa_chain, path="/mistral",
enable_public_trace_link_endpoint=True)
```

Listing 4.3: Setting `enable_public_trace_link_endpoint` argument to trace the activity on a LangServe route in InteRAG

Because **InteRAG** uses **LangServe** for its **API**, it offers automatic tracing on the routes by exposing a simple argument as shown in **Listing 4.3**. You need to keep in mind though that this sets the logging information to be publicly available, so use it only for testing purposes with dummy data.

4.1.1 Generator evaluation

Evaluating the **Generator** involves actions similar to evaluating any **Foundational Model (FM)**, **LLM** or in general any **GenAI** related model that is used in a non **RAG** application.

Taking that into account, the evaluation can be custom or **MLOps** based. I will briefly explain both evaluation principles in terms of **RAG Generator**.

4.1.1.1 Custom evaluation

As every action/operation that is performed manually, the custom evaluation of the **Generator** is really hard to perform because of the fact that multiple outputs need to be reviewed for a single prompt and, validate them separately and finally make an overall estimation about the performance (usually a metric such as the percentage of satisfying answers out of a total number of outputs). Unfortunately the custom evaluation is still needed as part of **HITL** because using **Superalignment** models for cognitive evaluation of a variable output is still in the experimental phase and the standard **NLP** metrics such as **BLEU** and **ROUGE** cannot capture the overall quality of the retrieved answers.

| Query | Generation | Estimation | Comment |
|--|---|------------|--|
| Bulletpoint me some interesting things for the company | Intertec has delivered projects in key industries such as HealthTech, Telemedicine, e-Commerce, FinTech, SportsTech, Affiliate, Chemicals, and Automotive. They have built a customer portal for a polymer trading client, an MVP solution for Industry 4.0 laboratory, and 5-star custom projects. | 1 | Good enough |
| Bulletpoint me some interesting things for the company | By aligning with our partners' values and aspirations, Intertec ensures that its collaborative journey is a symphony of shared purpose. Borce Cvetkovski is a Tech Manager at Intertec | 0 | Not quite relevant, exposing employee info |

Figure 4.2: Custom evaluation of the Generator in InteRAG

The custom evaluation of the **Generator** is an expensive process in terms of time and human resources, similar to the data labeling in the traditional **ML**. One example of how the process looks is given on **Figure 4.2** for InteRAG where a spreadsheet is used for describing the problem with the output.

4.1.1.2 Evaluation with MLOps

With the appearance of **GenAI** and **RAG**, the **MLOps** software evolved from supporting the operationalization of the traditional **ML lifecycle** to providing support for large **Foundation Models (FMs)** such as the pretrained **LLMs** as well. Furthermore, the main **MLOps** platforms started supporting logging, versioning and evaluation of parts of the **RAG pipeline** which confirmed its continued key importance for applying **RAG** in the business.

Python

```
llm = HuggingFaceHub(repo_id="mistralai/Mistral-7B-Instruct-v0.1",
                    model_kwargs={"max_length": 800, "temperature": 0.2}
)
rag_chain = RetrievalQA.from_llm(llm=llm, retriever=retriever)
```

Python

```
mlflow.set_tracking_uri(uri="http://0.0.0.0:5000/")
```

```

mlflow.set_experiment("InteRAG chain")

with mlflow.start_run(description="Mistral v1") as run:
    model_info = mlflow.langchain.log_model(
        chain,
        artifact_path="interag_chain",
        loader_fn=load_retriever,
        persist_dir=persist_directory
    )

```

```

Python
model_name = "interag"
model_version = "4"

interag_model =
mlflow.pyfunc.load_model(model_uri=f"models:{model_name}/{model_version
}")

```

Listing 4.4: logging, versioning and loading of part of the RAG pipeline in InteRAG using MLflow

Listing 4.4 shows how a chain encompassing the **Retriever** and the **Generator** in **InteRAG** can be logged and then loaded by version using **MLflow**. The versioning part is done automatically with each new chain logged under the same experiment.

| | |
|--|-----------------------|
| ari_grade_level/v1/p90 | 18 |
| ari_grade_level/v1/variance | 12.26834576 |
| flesch_kincaid_grade_level/v1/mean | 10.2769 |
| flesch_kincaid_grade_level/v1/p90 | 13.8 |
| flesch_kincaid_grade_level/v1/variance | 8.112096390000001 |
| latency/mean | 1.3152061591148376 |
| latency/p90 | 1.5179993629455566 |
| latency/variance | 0.12305331553936126 |
| rouge1/v1/mean | 0.6338462959102232 |
| rouge1/v1/p90 | 0.8076923076923077 |
| rouge1/v1/variance | 0.01969228327208603 |
| rouge2/v1/mean | 0.5265395181504999 |
| rouge2/v1/p90 | 0.7589490968801315 |
| rouge2/v1/variance | 0.029030409700764002 |
| rougeL/v1/mean | 0.6086899706211222 |
| rougeL/v1/p90 | 0.7936507936507937 |
| rougeL/v1/variance | 0.020044627759471828 |
| rougeLsum/v1/mean | 0.6191123197988869 |
| rougeLsum/v1/p90 | 0.8 |
| rougeLsum/v1/variance | 0.02070022378685399 |
| toxicity/v1/mean | 0.0012323756200639765 |

Figure 4.3: OpenAI gpt-4o-mini evaluation example with standard metrics in MLflow

Platforms and services such as **MLflow**, **LangSmith** and **Bedrock** provide **LLM** evaluation with standard metrics by which you can compare the models' performance on the domain specific data. **Figure 4.3** shows several metrics used for one **OpenAI** model in **ConfluenceRAG**.

4.2 Cost reduction

In the previous sections I mentioned that the summarization of the content can indirectly reduce costs for your **RAG** pipeline. There are other techniques solely dedicated to making your **RAG** application more cost effective than it was earlier. I will focus on the two I find have the best potential to achieve cost reduction on your **RAG** application.

4.2.1 Contextual compression

Contextual compression is a technique that enhances the efficiency of the **RAG** application by summarizing, condensing, or filtering retrieved documents before passing them to the **Generator**. The goal is to reduce the size of the retrieved context while retaining the most important and relevant information for generation. By effectively compressing the content while preserving most of the crucial context, the **RAG** system can focus on key details and produce more accurate, relevant, and efficient outputs. In **LangChain** the **Contextual compression** is provided by the **ContextualCompressionRetriever** wrapper for the base **Retriever**.

Python

```
llm = ChatOpenAI(model_name="gpt-4o-mini", temperature=0)
llm_chain_extractor = LLMChainExtractor.from_llm(llm)
llm_chain_compression_retriever = ContextualCompressionRetriever(
    base_compressor=llm_chain_extractor, base_retriever=retriever
)
```

Python

```
llm_chain_compression_retriever_results =
llm_chain_compression_retriever.invoke("Со кои манастири раководел
Арсениј?")
pretty_print_docs_retriever(llm_chain_compression_retriever_results)
```

Unset

Document 1:

Архимандритот Арсениј за првата манастирска резбарска работа
Арсениј од Галичник

Listing 4.5: LLM chain extractor for Contextual compression in Mutavchain

There are several ways to implement **Contextual compression**. The first one is by using **LLM chain extractor**. It works by extracting only relevant parts of a document content. **LangChain** provides **LLMChainExtractor** as a base compressor class for the **ContextualCompressionRetriever**. An example of how to use it is shown on **Listing 4.5**. However, using it on large content at the same time written in Macedonian language, such as in **Mutavchain**, does not yield satisfying results as shown in the final output (the document contents are shrunk drastically and there is only a single document considered for retrieval).

Python

```
llm_chain_filter = LLMChainFilter.from_llm(llm)
llm_chain_filter_compression_retriever = ContextualCompressionRetriever(
    base_compressor=llm_chain_filter, base_retriever=retriever
)
```

Python

```
llm_chain_filter_compression_retriever_results =
llm_chain_filter_compression_retriever.invoke("Со кои манастири
раководел Арсениј?")
pretty_print_docs_retriever(llm_chain_filter_compression_retriever_results)
```

Unset

Document 1:

ната, западната и северната старна во наосот на црквата за веропоклонците, потоа певници и друго. Не само на почетокот кога Арсениј раководел со манастирот, туку и понатаму ќе биде најзаслужен за ангажирањето на големите мајстори на резбата и живописот. Во времето на

Арсениј внатрешноста на манастирската црква и другите простории ќе блеснат засекогаш во невиден сјај од делата на мијачките уметници. Во 1810 година Арсениј со нарачката на балдахинот од Елбасан, на манастирот ќе му донесе едно извонредно уметничко дело од мајсторите Филиповци од Осој кои во тоа време работеле на територијата на Матија, денешна Албанија. Таму мијачките копаничари имале сопствена работилница, а манастирот Бигорски свои имоти со насади од маслинки, лозја и друго. Ова зборува за живата врска на манастирот СВ. Јован Бигорски со Елбасан. Архимандритот Арсениј за првата манастирска резбарска работа

25

Арсениј од Галичник

Document 2:

духовните чеда на Бигорскиот манастир ќе запише: “Архимандритот Арсениј е навистина еден од ретките игумени судејќи според зачуваните документи, кој се потрудил, повику вајќи се на народното предание раширено сред народот од ми јачкиот крај, да го “осветли” и приспомне темното минато на историјата на манастирот - неговото подалечно и поблиско минато”

Од раката на Арсениј за 32 години игуменување останати се голем број записи и документи за манастирот кои се зачувани до ден денес како што е на пример Великиот поменик или писмото за собирање на прилог за манастирот и др.

Listing 4.6: LLM chain extractor for Contextual compression in Mutavchain

Another strategy that fares a bit better for the same use case, but still not satisfying enough is to use **LLM chain filter** instead of the **LLM chain extractor**. Unlike the previous option, the **LLM chain filter** works not by manipulating the content of the original document but completely filtering out the documents that are irrelevant. **Listing 4.6** shows how to use it with **LangChain** and **Python**. The retrieved results contain more content from more than one document but still not quite as good as the ones that would have been returned by the base **Retriever**.

```
Python
embeddings_filter = EmbeddingsFilter(
```

```
        embeddings=OpenAIEmbeddings(), similarity_threshold=0.89
    )
    embeddings_filter_compression_retriever =
    ContextualCompressionRetriever(
        base_compressor=embeddings_filter, base_retriever=retriever
    )
```

Python

```
embeddings_filter_compression_retriever_results =
embeddings_filter_compression_retriever.invoke("Со кои манастири
раководел Арсениј?")
pretty_print_docs_retriever(embeddings_filter_compression_retriever_resu
lts)
```

Unset

Document 1:

ната, западната и северната старна во наосот на црквата за веропоклонниците, потоа певници и друго. Не само на почетокот кога Арсениј раководи со манастирот, туку и понатаму ќе биде најзаслужен за ангажирањето на големите мајстори на резбата и живописот. Во времето на Арсениј внатрешноста на манастирската црква и другите простории ќе блеснат засекогаш во невиден сјај од делата на мијачките уметници. Во 1810 година Арсениј со нарачката на балдахинот од Елбасан, на манастирот ќе му донесе едно извонредно уметничко дело од мајсторите Филиповци од Осој кои во тоа време работеле на територијата на Матија, денешна Албанија. Таму мијачките копаничари имале сопствена работилница, а манастирот Бигор ски свои имоти со насади од маслинки, лозја и друго. Ова зборува за живата врска на манастирот СВ. Јован Бигорски со Елбасан. Архимандритот Арсениј за првата манастирска резбарска работа 25 Арсениј од Галичник

Document 2:

Потекнувајќи од татковината на копаничарството и зографството - Галичник, Арсениј во себе го носи исконскиот порив и чувство за уметност

и тоа всушност ќе биде причина да знае да ги ангажира најдобрите мијачки резбари кои што ги платил за изработка на нов иконостас, балдахин, голем змиски крст, владички и игуменски столови, амвон, и столови на јужната, западната и северната страна во наосот на црквата за веропоклонците, потоа певници и друго. Не само на почетокот, кога Арсениј раководи со манастирот, туку и понатаму ќе биде најзаслужен за ангажирањето на големите мајстори на резбата и живописот. Во времето на Арсениј внатрешноста на манастирската црква и другите простории ќе блеснат засекогаш во невиден сјај од делата на мијачките уметници. По украсувањето на манастирската црква со балдахинот, кај Арсениј се јавува уште посилна жед за опремување на црквата и патува за Света Гора и Цариград каде набавил крстови, свеќници и наачал голем кивот.

Listing 4.7: Embedding filters for Contextual compression in Mutavchain

In order to improve the selection of the document candidates for retrieval for the purpose of **Contextual compression**, you can always turn to using embeddings models. However making an extra call to that kind of model can be slow and costly. Instead, applying a filtering logic based on the similarity of document embeddings to a query, can ensure that only the most relevant, diverse, and concise information is selected for the final context. This method is called **Embeddings filter** and takes advantage of a similarity threshold that is set for returning only the content that satisfies its value. An example of using **Embeddings filter** in **Mutavchain** is shown in **Listing 4.7**.

```
Python
splitter = CharacterTextSplitter(chunk_size=300, chunk_overlap=0,
separator=". ")
redundant_filter =
EmbeddingsRedundantFilter(embeddings=OpenAIEmbeddings())
relevant_filter = EmbeddingsFilter(
    embeddings=OpenAIEmbeddings(), similarity_threshold=0.89
)
pipeline_compressor = DocumentCompressorPipeline(
    transformers=[splitter, redundant_filter, relevant_filter]
)
pipeline_compression_retriever = ContextualCompressionRetriever(
    base_compressor=pipeline_compressor, base_retriever=retriever
)
```

Python

```
pipeline_compression_retriever_results =  
pipeline_compression_retriever.invoke("Со кои манастири раководел  
Арсениј?")  
pretty_print_docs_retriever(pipeline_compression_retriever_results)
```

Unset

Document 1:

Не само на почетокот, кога Арсениј раководи со манастирот, туку и понатаму ќе биде најзаслужен за ангажирањето на големите мајстори на резбата и живописот

Listing 4.8: **Compression pipeline example in Mutavchain**

To take advantage of multiple **Contextual compression** methods, you can use a **Compression pipeline** such as **DocumentCompressorPipeline**, provided by **LangChain**. Even more, with the **DocumentCompressorPipeline** you can combine **Contextual compression** with document transformation. You can see in the **Listing 4.8** how it works for **Mutavchain** - obviously the retrieved content is still too reduced because of a combination of factors (chunk related arguments and similarity threshold).

4.2.2 Inference constraint

As in any **GenAI** application, costs for **RAG** applications can be reduced on the **Generator** level by limiting the length of the answer. It is logical that adding more instructions in the prompt in order to affect the size of the output will lead to decreasing the cost because of the lower number of output tokens but it may also slightly increase the costs related to the number of input tokens.

Python

```
def setup_rag_chain_with_inference_constraint(_retriever):  
    llm = ChatOpenAI(model_name="gpt-4o-mini", temperature=0)  
  
    template = """Use the following pieces of context to answer the  
question at the end.  
If you don't know the answer, just say that you don't know, don't  
try to make up an answer.
```

```
Use three sentences maximum and keep the answer as concise as possible.
```

```
The answer should not exceed 100 characters in length.
```

```
{context}
```

```
Question: {question}
```

```
Helpful Answer:""
```

```
custom_rag_prompt = PromptTemplate.from_template(template)
```

```
rag_chain = (  
    {"context": retriever | format_docs, "question":  
RunnablePassthrough()}  
    | custom_rag_prompt  
    | llm  
    | StrOutputParser()  
)  
return rag_chain
```

```
Python
```

```
rag_chain = setup_rag_chain_with_inference_constraint(retriever)  
rag_chain.invoke("Какво гласи песната за Ажи Павле?")
```

```
Unset
```

```
'„Нарочуват бело Менде, порочуват...“'
```

Listing 4.9: Inference constraint on the prompt level in Mutavchain

One example of implementing the **Inference constraint** on prompt level is given on **Listing 4.9** where additional instruction is provided that constrains the answer to 100 characters regardless of the number of retrieved documents. This results in increasing the number of input tokens by 10 and hence the costs for the input but reduces the size of the output, thus preventing the costs measured by output tokens to skyrocket. Also taking into account that for most of the **LLMs** the prices for the output tokens used are higher than the ones for the input tokens, it pays to invest into increasing the prompt in order to constrain the output. In the example of the listing, it cuts the output way before the 100th character because the

answer is about the song and it does not make sense to cut it in the middle of the second sentence. This way it is assured that the answer is understandable enough without creating an impression that it is a mistake.

Python

```
def setup_rag_chain_with_inference_constraint(_retriever):
    llm = ChatOpenAI(model_name="gpt-4o-mini", temperature=0,
max_tokens=100)

    template = """Use the following pieces of context to answer the
question at the end.
    If you don't know the answer, just say that you don't know, don't
try to make up an answer.
    Use three sentences maximum and keep the answer as concise as
possible.

    {context}

    Question: {question}

    Helpful Answer: """
    custom_rag_prompt = PromptTemplate.from_template(template)

    rag_chain = (
        {"context": retriever | format_docs, "question":
RunnablePassthrough()}
        | custom_rag_prompt
        | llm
        | StrOutputParser()
    )
    return rag_chain
```

Python

```
rag_chain = setup_rag_chain_with_inference_constraint(retriever)
rag_chain.invoke("Какво гласи песната за Ажи Павле?")
```

Unset

```
'Песната за Ажи Павле гласи:\n\n„Нарочуват бело Менде, порочуват(x2)  
како зноеш Ажи Павле да си доеш.(x2) \nДа си доеш Ажи Павле пред  
Петровден(x2) пред Петровден Ажи Павле две недели.(x2) \nАко не доеш Ажи  
Павле пред Петровден(x2) Ќа го на'
```

Listing 4.10: Inference constraint with reduced maximum number of tokens in Mutavchain

If you want a more precise way of controlling the generated answer, there is an argument that you can take advantage of and it is supported for most of the current **LLMs**. I already used the **max_tokens** argument in one of the previous examples (take a look at **Listing 2.11** for **Embedding adaptors**). By reducing the maximum number of output tokens you can limit the costs more accurately than by intervening on prompt level. However, you have to be extremely careful because setting the value too low can result in cutting the sentences to being not meaningful at all and even applying a post processing logic will not save you in such situations as in **Listing 4.10**.

4.3 Latency reduction

The **LLM** in the **Generator** unit of the **RAG** pipeline is doing some **NLP** tasks on the retrieved content from the **Retriever** such as summarizing, paraphrasing etc. Those tasks are not quite complex anymore for most of today's **NLP** models regardless of their size. But at the start of my hands-on adventure with **RAG** I was reluctant to avoid using state of the art and extremely large models because I thought that for the **Generator** it was a strong requirement based on those examples from the frameworks that I used.

Python

```
def setup_rag_chain(_retriever, openai_model_name):  
    prompt = hub.pull("rlm/rag-prompt")  
    llm = ChatOpenAI(model_name=openai_model_name, temperature=1)  
  
    rag_chain = (  
        {"context": _retriever | format_docs, "question":  
RunnablePassthrough()}  
        | prompt  
        | llm  
        | StrOutputParser()  
    )  
    return rag_chain
```

```
Python
```

```
rag_chain = setup_rag_chain(retriever, 'gpt-4o-mini')
```

```
Python
```

```
%%time
```

```
rag_chain.invoke("Кoj бил Арсениј?")
```

```
Unset
```

```
CPU times: user 23.6 ms, sys: 6.06 ms, total: 29.7 ms
```

```
Wall time: 2.88 s
```

```
'Арсениј бил архимандрит и игумен на манастирот „Св. Јован Бигорски“,  
управувајќи со него од 1807 до 1839 година. Роден е во Галичник во  
втората половина на 18 век и се замонашил во младост. Познат е по својот  
придонес во уметноста и украсувањето на манастирот, ангажирајќи  
талентирани резбари и уметници.'
```

```
Python
```

```
rag_chain = setup_rag_chain(retriever, 'o1-preview')
```

```
Python
```

```
%%time
```

```
rag_chain.invoke("Кoj бил Арсениј?")
```

```
Unset
```

```
CPU times: user 31.4 ms, sys: 4.13 ms, total: 35.5 ms
```

```
Wall time: 15 s
```

```
'Архимандритот Арсениј, роден во Галичник во втората половина на 18 век,  
бил игумен на манастирот „Св. Јован Бигорски“ од 1807 до 27 јуни 1839
```

година. Тој значително придонел за украсувањето на манастирот, ангажирајќи ги најдобрите мијачки резбари за изработка на иконостас, балдахин, крстови и други уметнички дела. Арсениј патувал во Света Гора и Цариград за да набави црковни предмети и да нарача голем кивот, збогатувајќи ја духовната и културната вредност на манастирот.'

Listing 4.11: **Comparison of a smaller (gpt-4o-mini) and larger model (o1-preview) used in Mutavchain's Generator**

After some time I visited a conference where a guy presented examples of usages of lighter models as part of the **Generator**. They were working fine for his use cases and I decided to try smaller models in my applications as well and I ended up with mixed results. But one benefit of using smaller models of the same type aside from the reduced costs for the commercial models, was that they were a bit faster which is a logical consequence of the reduced size, although not always the case (for example, **gpt-4o** is a bit faster than **gpt-4o-mini**). For instance, for the **OpenAI** models, **gpt-3.5-turbo** performed similarly to **gpt-4-turbo** but faster, as well as **gpt-4o-mini** compared to **o1-preview** larger model. Also for the **OOS Mistral** models, when I replaced the larger **Mixtral-8x7B-Instruct** model with the smaller **Mistral-7B-Instruct**, I did not notice any degrading in the performance but the latency was reduced a bit. Of course some of the smaller models that I tried in **Mutavchain** were not handling the language as well as their larger counterparts because of the nature of the application (retrieving results in Macedonian). One such example is shown in **Listing 4.11**.

Latency can be reduced on the **Vector Store** level as well. For instance, **FAISS** is known to provide faster retrieval and storing than **Chroma**, although the difference can be measured in milliseconds which in some cases does not matter much. **Bedrock's Knowledge Bases** also have very low latency, especially when it scales as the quantity of the data rises significantly.

5. Maintenance

Most of the improvement points I have presented in the previous chapters can be treated as maintenance steps. Furthermore, maintaining a **RAG** application involves continuous monitoring, regular updates to the **Data Source**, suitable management of the **Vector Store**, proper scaling, while also applying robust testing and feedback mechanisms. These practices help ensure that a **RAG** application remains accurate, efficient, and relevant, providing users with reliable, high-quality interaction.

5.1 Retrieval management

There are two main ways of managing the data coming from the **Data Source** through the **LTE** unit, the **Vector Store** and finally the **Retriever** in the **RAG** applications - custom and built-in. Custom management can be implemented in many ways depending on how the data is loaded and retrieved. I will show two variants of custom management.

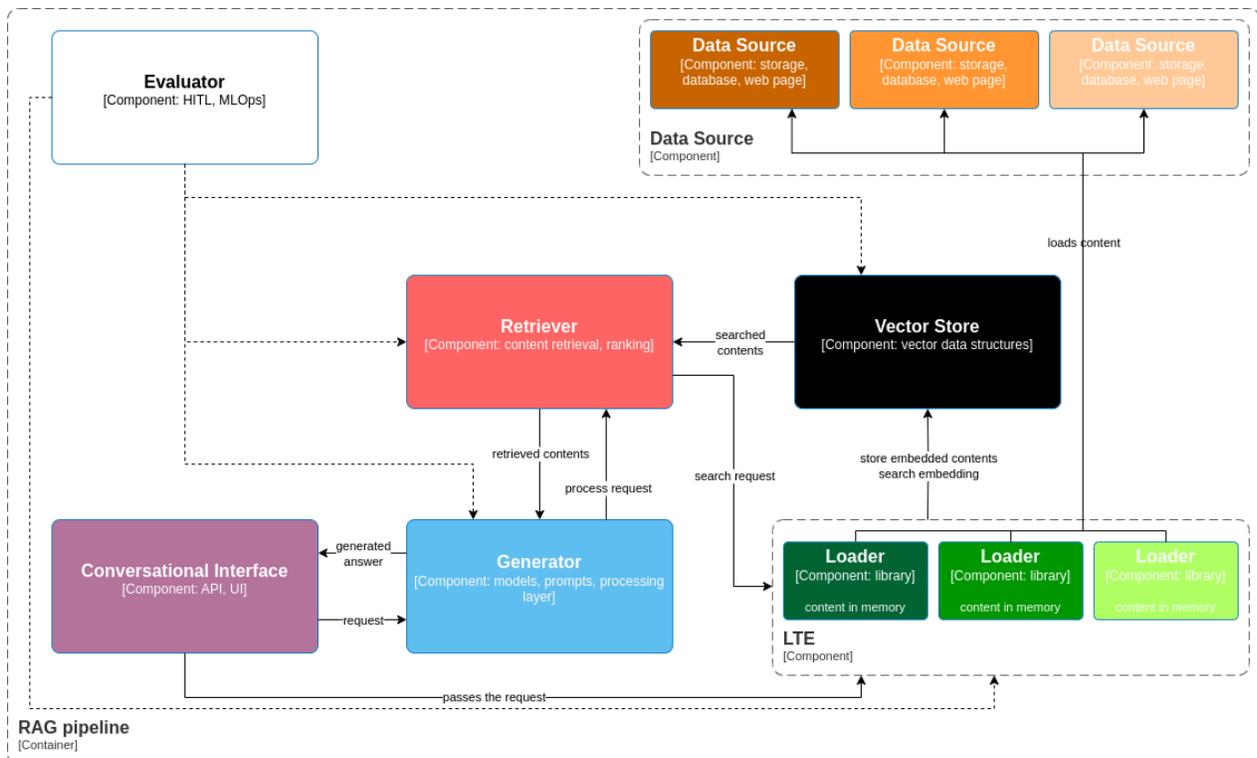


Figure 5.1: multiple loaders single retriever custom retrieval management

If you have data stored in more than one **Data Source**, and want to use them in the **RAG** pipeline, you can use multiple loaders in the **LTE** unit to load the data from all the **Data Sources**. Then after creating a **Vector Store** out of the loaded content, you can use a single **Retriever** to return the desired results. This is called multiple loaders single retriever management. This variant is implemented in **InteRAG** in a similar way as the one shown on the diagram in **Figure 5.1**.

```
Python
url_aws_mlops =
"https://www.intertec.io/resources/blogs/how-building-an-open-source-ml-
ops-platform-on-aws-can-improve-your-company-efficiency"
url_ai_manufacturing = (
"https://www.intertec.io/resources/blogs/10-use-cases-ai-manufacturing"
)

loader_aws_mlops = WebBaseLoader(url_aws_mlops)
docs_aws_mlops = loader_aws_mlops.load()

loader_ai_manufacturing = WebBaseLoader(url_ai_manufacturing)
docs_ai_manufacturing = loader_ai_manufacturing.load()

for doc in docs_aws_mlops:
    for md in doc.metadata:
        doc.metadata[md] = str(doc.metadata[md])

for doc in docs_ai_manufacturing:
    for md in doc.metadata:
        doc.metadata[md] = str(doc.metadata[md])
```

```
Python
embedding_model = "sentence-transformers/all-mpnet-base-v2"
persist_directory_aws_mlops = "../db/chroma_aws_mlops"
persist_directory_ai_manufacturing = "../db/chroma_ai_manufacturing"
llm_mistral7binstructv3 = "mistralai/Mistral-7B-Instruct-v0.3"

prompt = hub.pull("rlm/rag-prompt")
llm = HuggingFaceHub(repo_id=llm_mistral7binstructv3,
model_kwargs={"max_length": 800, "temperature": 0.2})
```

```

text_splitter = RecursiveCharacterTextSplitter(
    chunk_size=500, chunk_overlap=100, separators=["\n\n", "\n", "(?<=\.
)", " "]
)

all_splits_aws_mlops =
clean_documents(text_splitter.split_documents(docs_aws_mlops))
all_splits_ai_manufacturing =
clean_documents(text_splitter.split_documents(docs_ai_manufacturing))

vectorstore_aws_mlops =
Chroma.from_documents(documents=all_splits_aws_mlops,
embedding=embeddings, persist_directory=persist_directory_aws_mlops)
vectorstore_ai_manufacturing =
Chroma.from_documents(documents=all_splits_ai_manufacturing,
embedding=embeddings,
persist_directory=persist_directory_ai_manufacturing)

retriever_aws_mlops = vectorstore_aws_mlops.as_retriever()
retriever_ai_manufacturing = vectorstore_ai_manufacturing.as_retriever()

ensemble_retriever = EnsembleRetriever(
    retrievers=[retriever_aws_mlops, retriever_ai_manufacturing],
    weights=[0.5, 0.5]
)

```

Python

```

combined_results = ensemble_retriever.invoke("machine learning at
Intertec")
combined_results

```

Unset

```

[
    Document(
        page_content="Using MLOps software in data science has become a
recognized standard, freeing data scientists from manual tracking,
versioning, and code maintenance. Newcomers often start with open-source

```

tools for personal use, like MLflow, which enhances model evaluation, packaging, and deployment. For instance, teams at Intertec leverage MLflow to streamline compatibility with AWS Sagemaker, simplifying model deployment. Explore more in Intertec's guide to the machine learning lifecycle.",

```
        metadata={
            "description": "The benefits from using MLOps software for
data science have been addressed and recognized for years, reg",
            "language": "en",
            "source":
"https://www.intertec.io/resources/blogs/how-building-an-open-source-ml-
ops-platform-on-aws-can-improve-your-company-efficiency",
            "title": "How building an open-source MLOps platform on AWS
can improve your company's efficiency",
        },
    ),
```

```
    Document(
        page_content="Machine learning: a subset of AI, that involves
learning autonomously from data and algorithms without the need for
explicit programming. It mirrors human learning, where machines enhance
their performance through experience.",
```

```
        metadata={
            "description": "Artificial intelligence is more than just a
buzzword. It's a groundbreaking force in today's technologica",
            "language": "en",
            "source":
"https://www.intertec.io/resources/blogs/10-use-cases-ai-manufacturing",
            "title": "10 ways that AI can transform manufacturing
processes.",
        },
    ),
```

```
    Document(
        page_content="Building an OSS MLOps platform through machine
learning development services as an AWS partner",
        metadata={
            "description": "The benefits from using MLOps software for
data science have been addressed and recognized for years, reg",
            "language": "en",
            "source":
"https://www.intertec.io/resources/blogs/how-building-an-open-source-ml-
ops-platform-on-aws-can-improve-your-company-efficiency",
```

```

        "title": "How building an open-source MLOps platform on AWS
can improve your company's efficiency",
    },
),
Document(
    page_content="AI can extract, classify, and enter data from
documents into systems, streamlining operations and reducing
inefficiencies.",
    metadata={
        "description": "Artificial intelligence is more than just a
buzzword. It's a groundbreaking force in today's technologica",
        "language": "en",
        "source":
"https://www.intertec.io/resources/blogs/10-use-cases-ai-manufacturing",
        "title": "10 ways that AI can transform manufacturing
processes.",
    },
),
]

```

Listing 5.1: Ensemble Retriever with two Vector Stores in InteRAG

Another trick that can be applied on top of the previous custom retrieval management is to allow multiple **Vector Stores** to be created for multiple **Data Sources**. This means that a **Vector Store** can be created for one or more loaded content by the loader of the **LTE** unit. This variant is called multiple loaders multiple retrievers management. Some **RAG** libraries and frameworks can also support having multiple **Vector Stores** of different types for a single **RAG** application. **LangChain** goes even beyond and provides an **Ensemble Retriever** that combines the results from multiple retrievers which on the other hand get the results from multiple **Vector Stores**. **Listing 5.1** shows how this is implemented in **InteRAG**.

```

Python
bedrock_runtime = boto3.client(
    service_name="bedrock-runtime",
    region_name="eu-central-1",
)

model_id = "anthropic.claude-3-haiku-20240307-v1:0"

```

```

model_kwargs = {
    "max_tokens": 800,
    "temperature": 0.2,
    "top_p": 1,
    "stop_sequences": ["\n\nHuman"],
}

template = """Answer the question based only on the following context:
{context}

Question: {question}"""

prompt = ChatPromptTemplate.from_template(template)

retriever = AmazonKnowledgeBasesRetriever(
    knowledge_base_id="YQER3PBCYU",
    retrieval_config={"vectorSearchConfiguration": {"numberOfResults":
4}},
)

model = ChatBedrock(
    client=bedrock_runtime,
    model_id=model_id,
    model_kwargs=model_kwargs,
)

chain = (
    RunnableParallel({"context": retriever, "question":
RunnablePassthrough()})
    .assign(response=prompt | model | StrOutputParser())
    .pick(["response", "context"])
)

```

Python

```

response = chain.invoke("Which team members from the company provider
side should take part in the problem-framing discussions?")
response

```

Unset

```
{
  "response": "Based on the given context, there is no information about problem-framing discussions or the involvement of specific team members from the company provider side. The documents provided discuss remote working practices and the use of the Jitsi platform for doctor-patient communication. There is no mention of any problem-framing discussions or the roles of different team members. Therefore, I cannot answer the question based solely on the provided context.",
  "context": [
    Document(
      metadata={
        "location": {
          "type": "WEB",
          "webLocation": {
            "url":
"https://www.intertec.io/resources/blogs/most-effective-practices-remote-working"
          },
        },
        "score": 0.37671039,
        "source_metadata": {
          "x-amz-bedrock-kb-source-uri":
"https://www.intertec.io/resources/blogs/most-effective-practices-remote-working",
          "x-amz-bedrock-kb-chunk-id":
"1%3A0%3AuNS48ZEBdCof1bTcHB1K",
          "x-amz-bedrock-kb-data-source-id": "JCA8KK0D82",
        },
      },
      page_content="In fact, if you want your employees to move forward and improve their performance, you would need to include and provide regular and meaningful feedback. You can start by setting up a process of providing feedback, whether it is a weekly or monthly one-on-one online meeting or just a simple feedback app. Once you've done that, you should encourage your team members to give feedback to each other, as well as to you as their manager. While remote work can be solely viewed as an emerging trend, the results from Buffer's The 2020 State of Work report, show that in fact, this way of working is here to stay. *If you're on the path of scaling your business, hiring a remote team is something you should definitely consider.* *Contact us* *and our consultants will help you understand how remote setup can be beneficial
```

for your business.* Tanja Zlatanovska May 19, 2020 ## You may also like
View all posts

Towards_Perfect_ES_Query_Generation_with_Elastic_builder.w
ebp ### Towards Perfect ES Query Generation with Elastic-builder With
all due respect to what ES (ElasticSearch) brings to the querying of the
data compared to the other storage engines, one of its disadvantages is
the potential complexity of the queries.",

```
    ),  
    Document(  
      metadata={  
        "location": {  
          "type": "WEB",  
          "webLocation": {  
            "url":  
"https://www.intertec.io/resources/blogs/using-jitsi-for-doctor-patient-  
communication"  
          },  
        },  
        "score": 0.37531075,  
        "source_metadata": {  
          "x-amz-bedrock-kb-source-uri":  
"https://www.intertec.io/resources/blogs/using-jitsi-for-doctor-patient-  
communication",  
          "x-amz-bedrock-kb-chunk-id":  
"1%3A0%3ADNy48ZEBTRfKsCuLFoVH",  
          "x-amz-bedrock-kb-data-source-id": "JCA8KK0D82",  
        },  
      },  
      page_content="Using Jitsi For Doctor-Patient Communication  
Curalie\_Blog\_image\_1.webp # Using Jitsi For Doctor-Patient  
Communication The relevance of telemedicine has increased significantly  
during the pandemic, but even before the pandemic it was gaining  
popularity because it helps solve several challenges, especially in  
rural areas, where there may be shortages of specialists and the nearest  
hospital or doctor's office may be far away. Telemedicine services such  
as video consultations have become much more important and popular in  
the wake of the Covid-19 pandemic. One of our clients in the healthcare  
industry wanted to build a video consultation platform that lets you get  
professional help from medical experts in just a few clicks -- from  
anywhere, without the hassle of getting to a doctor's office, with no  
wait and no risk of infection. ##### **What were the next steps** First,
```

we researched the WebRTC standard, and how it works and then explored six existing video conferencing platforms: **Jitsi:** Jitsi is an open-source communication platform that employs WebRTC standards to facilitate real-time audio and video communication. It is designed for a range of collaborative applications, with a focus on video conferencing and instant messaging. **Apache OpenMeetings:** OpenMeetings is a free web conferencing software that allows multiparty conferencing via audio/video, drawing board, and shared desktop. Both webinars with one presenter and conferences with an arbitrary number of participants interacting with each other are supported.",

```
    ),
    Document(
      metadata={
        "location": {
          "type": "WEB",
          "webLocation": {
            "url":
"https://www.intertec.io/resources/blogs/using-jitsi-for-doctor-patient-communication"
          },
        },
        "score": 0.37477598,
        "source_metadata": {
          "x-amz-bedrock-kb-source-uri":
"https://www.intertec.io/resources/blogs/using-jitsi-for-doctor-patient-communication",
          "x-amz-bedrock-kb-chunk-id":
"1%3A0%3AD9y48ZEBTRfKsCuLFoVH",
          "x-amz-bedrock-kb-data-source-id": "JCA8KK0D82",
        },
      },
      page_content="Among closed-source cloud solutions, the most promising is Twilio Video with Amazon Chime SDK being a close second (the tiebreaker in favor of Twilio is the community, documentation, and focus on APIs). #### Final decision and recommendation Considering the requirements for a certified video consultation platform we decided to use Jitsi as our base platform. We decided to use Jitsi because of several technical and practical reasons: * Open-source: Jitsi's open-source code lets us inspect and customize the code to our needs. * Self-hosting: Hosting Jitsi on our own (client's) server, gave us complete control over data security and privacy. * Ease of use: Even
```

non-technical users find Jitsi intuitive and easy to use. No downloads or plugins needed, making it seamless for users. This translates to smooth collaboration and fewer support requests. * Web-based accessibility: The web-based nature of Jitsi aligns with modern development practices. Participants can join meetings directly from their web browsers, simplifying the user experience and making it accessible across different platforms. * Scalability: Jitsi can scale to accommodate hundreds of participants. * API integrations: Jitsi has support for integration through APIs. This feature enabled us to seamlessly integrate video consultation into our workflow. * Customization options: Allowed us to tweak the UI/UX to fit the client's brand and needs.",

```
    ),
    Document(
      metadata={
        "location": {
          "type": "WEB",
          "webLocation": {
            "url":
"https://www.intertec.io/resources/blogs/using-jitsi-for-doctor-patient-communication"
          },
        },
        "score": 0.37450394,
        "source_metadata": {
          "x-amz-bedrock-kb-source-uri":
"https://www.intertec.io/resources/blogs/using-jitsi-for-doctor-patient-communication",
          "x-amz-bedrock-kb-chunk-id":
"1%3A0%3AEdy48ZEBTRfKsCuLFoVH",
          "x-amz-bedrock-kb-data-source-id": "JCA8KK0D82",
        },
      },
      page_content="To generalize, if you want to send a post object to an ES endpoint, the construction of that object Velimir Graorkoski Velimir Graorkoski Article Elastic\\_Builder\\_To\\_The\\_Rescue.webp ### Elastic Builder To The Rescue Previously, we talked about the birth and the agony of expanding complex ES queries when the custom generation is applied with all of its disadvantages. In this post, we will show examples of how easily the elastic-builder can handle the generation and which are the po Velimir
```

```

Graorkoski Velimir Graorkoski Article
Understanding\the\Most\Effective\Practices\of\Remote\Working.webp ### Understanding the Most Effective Practices of Remote Working
Intertec has been delivering services by working remotely for clients in
multiple business domains for almost 10 years. By using the latest
remote-first methodologies we have helped companies either grow their
tech teams or turn their ideas into reality. In recent times, remote
working ha Tanja Zlatanovska Tanja Zlatanovska Article ## Let's talk.
Name Email Phone Submit By submitting this, you agree to our Terms and
Conditions & Privacy Policy",
    ),
    ],
}

```

Listing 5.2: Integrating Bedrock's Knowledge Bases in InterRAG

If you want to avoid the burden of custom management of the **RAG** pipeline despite the useful features of the **RAG** libraries, there are services that have built-in management with little to no code effort, such as **Amazon's Bedrock**. This cloud service provides a relatively easy click based interface that supports a variety of **Data Sources** and stores the loaded data in **Vector Stores** of different types that the consumer does not even have to be aware of, known as **Knowledge Bases**. All you have to do later is to communicate with them in the code in a similar way as given on **Listing 5.2**.

5.2 Vector Store content updates

For a **RAG** application to consider the latest state of the content in the **Data Source**, the **Vector Store** needs to stay up to date with the content's embedded representation. On paper, this seems simple to implement but it can result in a resource consuming operation that will take additional development and monitoring of the activity of the whole application. I will give several examples of how to deal with maintaining the **Vector Store** in terms of content updates.

The most ideal way of keeping the **Vector Store** contents synced with the **Data Source** is to propagate the content through the **LTE** unit and finally to the **Vector Store** every time there is a change. In reality this is very challenging to be implemented in a single application because it will consume some capacity for listening to changes in the **Data Source** as well as increased usage of the **LTE** unit and the **Vector Store** that in the meantime has to be focused on retrieval instead.

On the other hand the most naive way of handling the **Vector Store** contents is to recreate it every time the application starts. This will increase the start time of the application and if there are other changes that need to be deployed on production, the downtime will be significant. Hence the consumers will be blocked until the new **Vector Store** is ready for

retrieval. The downtime will be increased as the size of the content in the **Data Source** that is loaded increases as well.

```
Python  
vectorstore = Chroma.from_documents(documents=document_chunks,  
embedding=embedding_model, persist_directory=vector_store_location)
```

Listing 5.3: Loading of Chroma Vector Store from persistent location using LangChain in InteRAG

Slight improvement of this **Vector Store** management option is to use libraries that handle the **Vector Store** update by considering only the changed content. This decreases the down time a bit because the Vector Store is not recreated entirely, but only a part of it on top of what is stored already. **Listing 5.3** shows how **Chroma Vector Store** is loaded from a persistent location before updating it with new content.

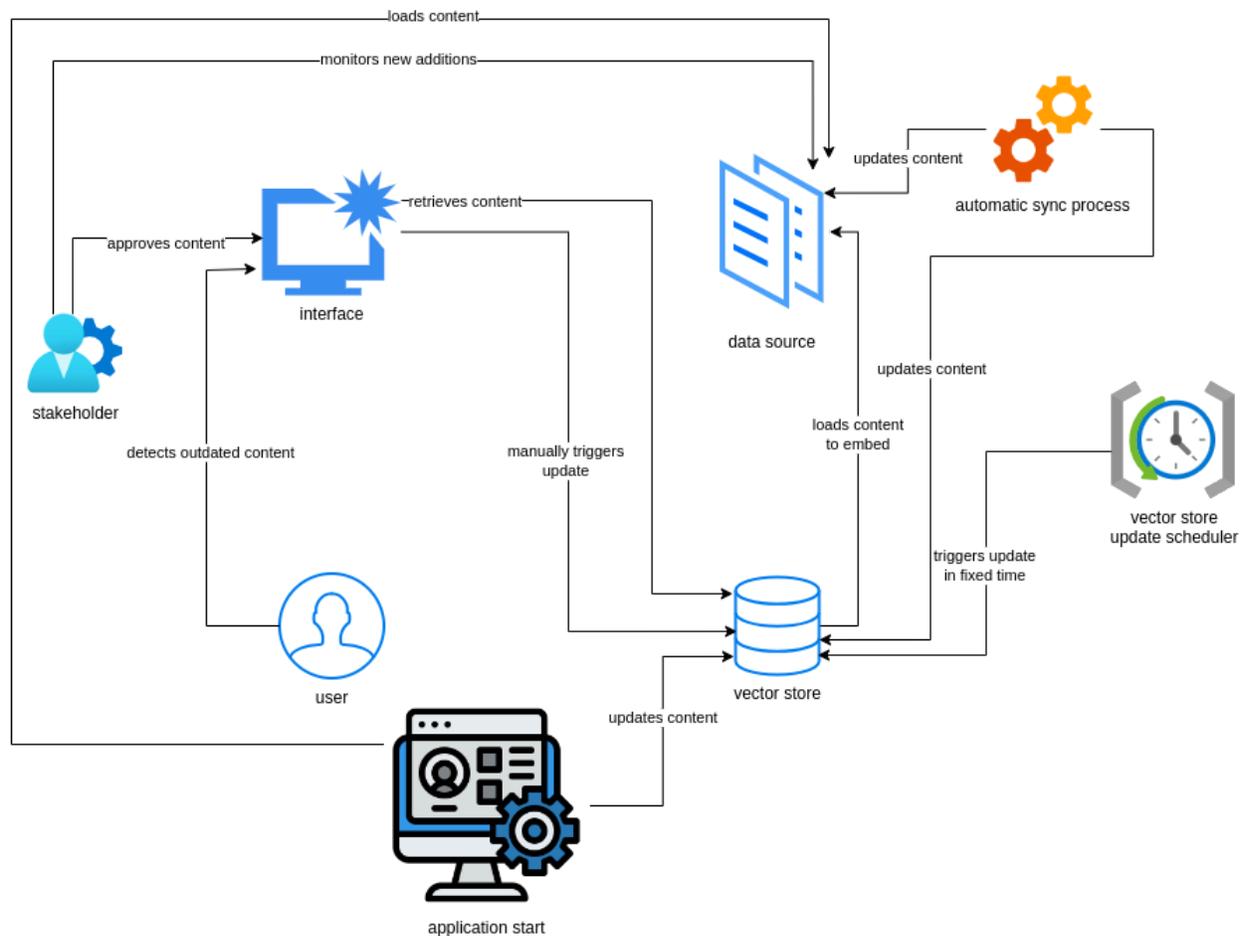


Figure 5.2: Vector Store content update options

To relieve the **RAG** application of having to sync the **Vector Store** with the **Data Source**, a scheduler (cron job, separate application) can take care of updating the **Vector Store**. This option represents sort of a tradeoff between the first two options depending on the frequency of update in the **Data Source**. For example, the content synchronization can be triggered at a specified period of time (at 11 o'clock) and/or at a specific time interval (every 3 days).

Manual interventions are often applied because usually the stakeholders want to control the data that should be considered by the **RAG** application. This option results in having an approval step on every change in the **Data Source**. According to it, the content that is not approved does not trigger an update of the **Vector Store**.

Sometimes even the users are allowed to manually trigger the **Vector Store** update when they feel like the data presented by the application is out of date, but mostly in the cases where there is a dedicated **Vector Store** for each user.

All of these **Vector Store** content update options are given on the diagram on **Figure 5.2**. Please keep in mind that although not clearly stated, updating the **Vector Store** content covers also new content addition as well as content removal.

5.3 RAG scaling

RAG applications can be implemented to serve single or multiple requests at a time according to their purpose or the domain they are used for. The more consumers (users, other services) they have, the more effectively you need to manage them. In order to make the management easier, similar to other applications in the software world, they have to be scalable. In terms of **RAG** it means that the applications can cope without many problems with the increased query load from the **Conversational Interface** side and increased content quantity from the **Data Source** side that affect the whole pipeline.

For a single API, since more requests coming from the consumers can increase the latency because it will force the application to perform more retrievals, in the worst case scenario it will result in failure to process the requests. In order to prevent such cases as much as possible, you can use scalable OSS API frameworks such as **FastAPI** or if your application is written with **LangChain**, **LangServe** can be a good alternative as used for the purpose of **InteRAG**. Other software that can be used to deal with the scalability of the **RAG** application are the cloud platform based **RAG** services such as **Bedrock** from **Amazon**.

5.3.1 Batching

Often in cases of service consumers and sometimes in cases of users of **RAG** applications, the queries/questions are related. In those cases there is an opportunity to treat all those queries at once in a batch, instead of sequentially dealing with them. Besides on the **Retriever**, the batching concept can be applied while embedding the content. I already showed it in the example for creating a **Vector Store** with **Hypothetical questions** where it totally makes sense to tackle multiple questions at once instead of embedding them one by one.

Python

```
combined_results = ensemble_retriever.batch(['machine learning at Intertec', 'artificial intelligence at Intertec'])
combined_results
```

Unset

```
[
  [
    Document(
      page_content="Using MLOps software in data science has become a recognized standard, freeing data scientists from manual tracking, versioning, and code maintenance. Newcomers often start with open-source tools for personal use, like MLflow, which enhances model evaluation, packaging, and deployment. For instance, teams at Intertec leverage MLflow to streamline compatibility with AWS Sagemaker, simplifying model deployment. Explore more in Intertec's guide to the machine learning lifecycle.",
      metadata={
        "description": "The benefits from using MLOps software for data science have been addressed and recognized for years, reg",
        "language": "en",
        "source": "https://www.intertec.io/resources/blogs/how-building-an-open-source-ml-ops-platform-on-aws-can-improve-your-company-efficiency",
        "title": "How building an open-source MLOps platform on AWS can improve your company's efficiency",
      },
    ),
    Document(
      page_content="Machine learning: a subset of AI, that involves learning autonomously from data and algorithms without the need for explicit programming. It mirrors human learning, where machines enhance their performance through experience.",
      metadata={
        "description": "Artificial intelligence is more than just a buzzword. It's a groundbreaking force in today's technologica",
        "language": "en",
        "source": "https://www.intertec.io/resources/blogs/10-use-cases-ai-manufacturing",

```

```

        "title": "10 ways that AI can transform manufacturing
processes.",
    },
),
Document(
    page_content="Building an OSS MLOps platform through machine
learning development services as an AWS partner",
    metadata={
        "description": "The benefits from using MLOps software
for data science have been addressed and recognized for years, reg",
        "language": "en",
        "source":
"https://www.intertec.io/resources/blogs/how-building-an-open-source-ml-
ops-platform-on-aws-can-improve-your-company-efficiency",
        "title": "How building an open-source MLOps platform on
AWS can improve your company's efficiency",
    },
),
Document(
    page_content="AI can extract, classify, and enter data from
documents into systems, streamlining operations and reducing
inefficiencies.",
    metadata={
        "description": "Artificial intelligence is more than
just a buzzword. It's a groundbreaking force in today's technologica",
        "language": "en",
        "source":
"https://www.intertec.io/resources/blogs/10-use-cases-ai-manufacturing",
        "title": "10 ways that AI can transform manufacturing
processes.",
    },
),
[
    Document(
        page_content="Join us in embracing the future of MLOps,
where technology meets possibility, and together, let's redefine what's
achievable in AI and machine learning.Velimir GraorkoskiJul 03, 2024You
may also likeView all postsTowards Perfect ES Query Generation with
Elastic-builderWith all due respect to what ES (ElasticSearch) brings to
the querying of the data compared to the other storage engines, one of

```

its disadvantages is the potential complexity of the queries. To generalize, if you want to send a",

```
        metadata={
            "description": "The benefits from using MLOps software
for data science have been addressed and recognized for years, reg",
            "language": "en",
            "source":
"https://www.intertec.io/resources/blogs/how-building-an-open-source-ml-
ops-platform-on-aws-can-improve-your-company-efficiency",
            "title": "How building an open-source MLOps platform on
AWS can improve your company's efficiency",
        },
    ),
    Document(
```

```
        page_content="Scarcity of skilled professionals IBM's Global
AI Adoption Index 2023 reported that 33% of enterprises
face\u202flimited in-house AI skills and expertise. AI projects need a
diverse team, including data scientists, machine learning engineers,
software architects, business intelligence analysts, and subject matter
experts. Additionally, manufacturing will face a significant workforce
shortage in the next decade as Baby Boomers retire.",
```

```
        metadata={
            "description": "Artificial intelligence is more than
just a buzzword. It's a groundbreaking force in today's technologica",
            "language": "en",
            "source":
"https://www.intertec.io/resources/blogs/10-use-cases-ai-manufacturing",
            "title": "10 ways that AI can transform manufacturing
processes.",
        },
    ),
```

```
    Document(
        page_content="Looking forward, as MLflow introduces new
GenAI features and AWS expands its support with services like Bedrock,
we're poised to harness these advancements. Our focus remains on
developing innovative AI applications and adapting to emerging
technologies, ensuring that our clients benefit from cutting-edge
solutions.",
```

```
        metadata={
            "description": "The benefits from using MLOps software
for data science have been addressed and recognized for years, reg",
```

```

        "language": "en",
        "source":
"https://www.intertec.io/resources/blogs/how-building-an-open-source-ml-
ops-platform-on-aws-can-improve-your-company-efficiency",
        "title": "How building an open-source MLOps platform on
AWS can improve your company's efficiency",
    },
),
Document(
    page_content="Computer vision: It allows AI systems to
examine and interpret visual input, thus playing a pivotal role in their
comprehension of the surroundings.",
    metadata={
        "description": "Artificial intelligence is more than
just a buzzword. It's a groundbreaking force in today's technologica",
        "language": "en",
        "source":
"https://www.intertec.io/resources/blogs/10-use-cases-ai-manufacturing",
        "title": "10 ways that AI can transform manufacturing
processes.",
    },
),
],
]

```

Listing 5.4: **Batch inference on a LangChain Retriever in InterRAG**

Libraries such as **LangChain** support batching on several levels. For a simple **Batch inference** regardless of how the other units of the **RAG** pipeline are set, you can use the **batch** function with multiple queries instead of the **invoke** for a single query. **Listing 5.4** shows how this is done on the same documents used in example in **Listing 5.1**.

For user facing applications where the relationship of a certain number of questions is known beforehand or at least can be predicted, a technique called **Continuous batching** can be implemented. Each incoming request is added to a queue and when the queue reaches the maximum size, **Batch inference** is triggered for all of the queries. It can be implemented in a custom way using special libraries or API frameworks, such as **Asyncio** for **Python**. By aggregating queries into batches continuously, you reduce the number of calls to the retrieval and generation models, which improves overall throughput. While the latency compared to the individual requests is increased, by applying **Continuous batching** the overall latency decreases. Furthermore there are fewer total calls to the **Retriever** which optimizes resource allocation and results in lower operational costs.

5.3.2 Scalable retrieval

To efficiently handle large content from **Data Sources** and high query volumes on the **Vector Stores**, providing quick and relevant responses as the data grows can be mitigated with scalable **Retrievers**. Increased amount of data that needs to be embedded results in larger **Vector Stores** which in the long run increases the latency of the retrieval and can also affect the overall performance of the **RAG** pipeline. Scaling manually is too complicated as you probably already understood from the **Maintenance** chapter, it is more or less related to every aspect of managing a **RAG** application.

The screenshot displays the AWS Bedrock console interface for managing a Knowledge Base. The main content area is titled 'intertec-blog' and includes several sections:

- Knowledge base overview:** Shows the Knowledge base name (intertec-blog), Knowledge base ID (YQER3PBCYU), Log Deliveries (Configure log deliveries and event logs in the Edit page), Status (Ready), and Created date (September 14, 2024, 19:34 (UTC+02:00)).
- Tags (1):** A table with columns for Key and Value. The entry is 'Owner' with the value 'velimir.graorkoski@intertec.io'.
- Data source (1):** A table with columns for Data source, Status, Data source, Account ID, Source Link, Last sync, Last sync, and Chunking. The entry is 'intertec-...' with status 'Available', source 'Web Craw...', account ID '22059492...', source link 'https://w...', last sync 'Septembe...', and chunking 'Default'.
- Embeddings model:** A section for selecting an embeddings model.

On the right side, there is a 'Test knowledge base' sidebar with a 'Generate responses' toggle and a 'Claude Instant v1.2 | On-demand throughput' model selection. A chat interface is shown with a prompt: 'What are the top blog posts in intertec?' and a response: 'Based on the search results, some of the top blog posts from intertec are about moving to a new office space, using elastic builders to generate Elasticsearch queries, and understanding effective practices for remote working.' A 'Run' button is visible at the bottom of the chat interface.

Figure 5.3: Bedrock's Knowledge Base feature for InterRAG shown on AWS console

The best way to handle the problem of **Scalable retrieval** is to use cloud services that support automatically scaling of the **Vector Stores**. Such services as in the case of **Bedrock**, offer a variety of ways to create **Vector Stores** or to connect to other services that manage **Vector Stores**. **Bedrock** manages this through its **Knowledge Base** feature that I introduced you to earlier. As shown on **Figure 5.3**, it provides an interface with easy click based connections to a variety of **Data Sources**, configuring listing of those connections, along with testing it on a playground widget.

5.4 Evaluation

Even if your **RAG** application scales without any major technical problems it still does not mean that you can avoid maintaining it. Maintenance also covers evaluation of the **RAG** pipeline which is an operation still largely manually performed because it requires human judgment of

the applications performance that I mentioned earlier as **HITL**. You also witnessed how evaluation can be performed on the **Generator** unit of the **RAG** pipeline in **Chapter 4.1.1**. In this last section however, I will try to generalize the term. Almost everything I covered in this book can become a target for evaluation.

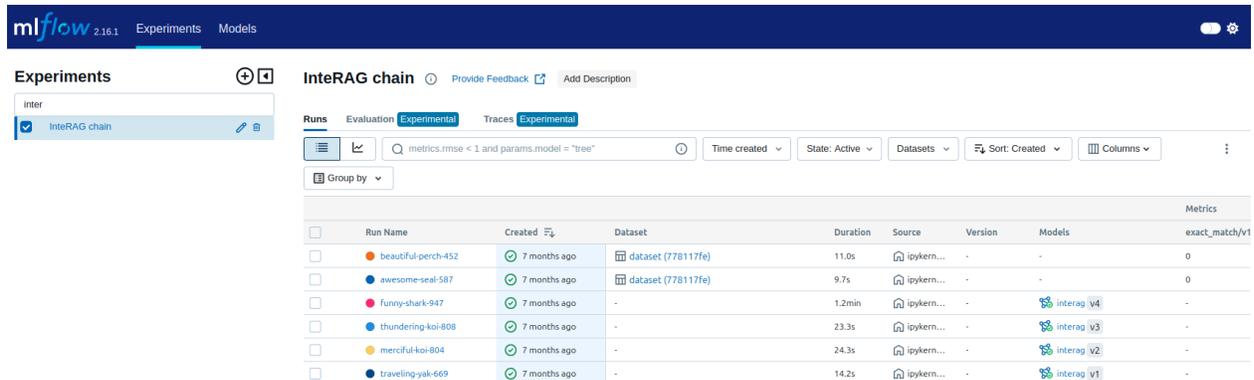


Figure 5.4: Experiment tracking dashboard on MLflow for InteRAG

Evaluation of **RAG** is flexible, you can choose any kind of feedback from **HITL** to include or any **RAG** unit to cover. The more you include in your coverage, the more work for evaluation you will have. But that leads to more opportunities for catching previously undetected problems and potentially improving the state of your **RAG** application. There is one piece of software that you can count on to assist you in the evaluation - **MLOps**.

Python

```
def load_retriever(persist_directory):
    embeddings = HuggingFaceEmbeddings(model_name=embedding_model)
    vectorstore = Chroma(persist_directory=persist_directory,
        embedding_function=embeddings)
    return vectorstore.as_retriever()
```

Python

```
llm = HuggingFaceHub(repo_id="mistralai/Mistral-7B-Instruct-v0.3",
    model_kwargs={"max_length": 800, "temperature": 0.2}
)
generator_chain = RetrievalQA.from_llm(llm=llm, retriever=retriever)
```

Python

```
mlflow.set_experiment("InteRAG chain")
```

```
with mlflow.start_run(description="Mistral v3") as run:
    model_info = mlflow.langchain.log_model(
        chain,
        artifact_path="interag_chain",
        loader_fn=load_retriever,
        persist_dir=persist_directory
    )
```

Listing 5.5: **Logging of the Generator and the Retriever using MLflow in InteRAG**

MLOps platform is already considered as an integral part of the **Evaluator** unit of the current **RAG** pipelines. It is preferable to use it, not just because of the built in metrics for the **LLMs** you can take advantage on for **Generator** evaluation, but because you can define what metric or parameter of the **RAG** pipeline you want to monitor and perform more operations such as triggering on demand testing, versioning of parts of the pipeline, comparison of those versions and many more. An overview of the dashboard of **MLflow**, an **MLOps** tool that is used for **InteRAG** purposes is shown in **Figure 5.4**, with the first two rows representing the dataset experiments, while the rest of the rows show the tracking of the logging of the versions of part of the **RAG** pipeline, in this case for the **Generator** and the **Retriever**. The code to enable logging of the **Generator** and the **Retriever** is shown on **Listing 5.5**.

My advice is to immediately test every change you apply to the **RAG** pipeline because even the smallest one, no matter how irrelevant it seems to be (adding a new word or a sentence, changing the embedding model, changing the standard chunking with **NSP**, ...), can affect the output in a different way from the one that the pipeline was projected to generate. Define and monitor only the main metrics such as latency and performance, but for the latter try to define it to be as much understandable for both the engineering team and the stakeholder. The latter is usually crucial for the **HITL** evaluation, hence he/she needs to understand the metric in order to better estimate the overall performance, point out the possible flaws and update the **ROI** projections.

6. *Conclusion and beyond*

As you approach the end of this book's content, your expectations might not be fulfilled as you hoped at the beginning. I did not write the book with the intention that it will solve all of your problems you might have with utilizing the **RAG** concept. Nor did I intend to make you strongly consider the improvements for which I gave negative examples. I just wanted to warn you that all is not ideal in the **RAG** world and definitely not black/white, because what did (not) work for me, will (not) work for you depending on the domain you are researching/developing. Because of this, at the end of the day, **RAG** will not relieve you from the challenge of estimating the effort you use on developing a "smart" application, that the traditional **ML** suffered from.

Since I gave lots of negative examples for implementing advanced **RAG** techniques, you might feel disappointed that you will inevitably have to start from the basic **RAG** setup. And you are not far from the truth. At the end of the day there is no escaping the naive **RAG** because you cannot just skip it and immediately apply most of the advanced improvement techniques that I mentioned. And if you do so you can open up more tech problems instead of solving the ones your domain requires you to mitigate.

In the near future, I expect more content to be published on the internet describing the experiences of developing **RAG** applications for different domains and use cases because the challenges can vary, something similar to the **Domain Driven Design** pattern for standard application development. One positive side effect of adopting **RAG** for website development is that it shifts the focus from the design to the API i.e. why put efforts on the design and web pages structuring to improve the visitors' navigation when you can just have a **Conversational Interface** instead. Furthermore, the **Agentic RAG** concept is quickly gaining hype and importance because of the ability of systems which employ it to integrate with various **Data Sources** and make decisions based on user's necessities to solve these problems in a more efficient way than the traditional **Virtual Assistants** used to do.

If you don't quite get all the information presented in this book, please try to load it with a **RAG** framework and develop a naive **RAG** application using the mine or other recommendations. Then just ask questions based on this content. If the answers start to not make sense, please try to use some of the techniques for improvement that I presented or similar ones. At least you will be able to detect if and where I made some mistakes that the reviewers and proofreaders were not able to catch. Extensive content such as this book needs to be stored for the readers like people or consumers like the models to digest it. After all, something needs to act as a core knowledge in order to derive another more useful knowledge from it.