# RAG model

Nane Mambreyan

2024-06-19

## SECTION 1: ARCHITECTURE

A chatbot that will let people chat with PDF manuals, following an Retrieval-Augmented Generation model would typically have the following structure (Please find the illustration of this architecture at the end of the section):

### Part 1 - FROM PDF TO DB

1) we begin with the PDF document

2) PDF is passed to a pre-processing(reading, removing stop words, cleaning…) function. I would use PyMuPDF for extracting information because it is highly efficient and versatile, making it ideal for handling the diverse content found in installation manuals for home equipment and not only. PyMuPDF (aka fitz) excels in processing PDF documents quickly, which is crucial when dealing with large manuals. It is capable of extracting text, images, and metadata, ensuring comprehensive information retrieval from the manuals. Additionally, PyMuPDF maintains the structure and layout of the original documents to some extent, which aids in preserving the context of the extracted text.

3) The Preprocessed data from PDF is then passed to a function for splitting the overall content into logicAl and coherent chunks of data. I would use spaCy for segmenting the extracted text because it is a powerful natural language processing (NLP) library that deals with in text segmentation. spaCy's built-in sentence segmentation capabilities allow for dividing the text into meaningful sentences, ensuring each chunk of text is coherent and contextually relevant. This is particularly important for breaking down technical manuals into understandable segments which I will use later on. Moreover, spaCy offers customization options for handling specific domain-related language, such as the technical terms found in installation manuals. Its optimization for performance makes it suitable for efficiently processing large volumes of text, ensuring that the segmentation process is both fast and accurate.

4) So far we would have a data that is

- BIG
- textual
- 'chunked'

The first two criteria are subject to difficulties. We want to optimize the data in terms of the space it occupies (this will also help to ensure the low time complexity associated with further actions to be done with the data). I would use SVD given a numeric data. So one option would be converting textual to numeric data then doing dim. reduction. The reason why I decided not to go with that option is that converting big textual data into numeric is already computationally heavy and would take lots of time.So instead, we can do dim. reduction on textual data firstly. NMF is well-suited for textual data dim reduction tasks due to its ability to decompose non-negative matrices into interpretable parts. This approach would allow the chatbot to extract and comprehend key topics from the manuals, facilitating more accurate responses to user queries. By reducing the dim of the 'word matrix derived from the PDF using NMF, the chatbot can efficiently analyze and categorize information based on the underlying themes and concepts within the manuals. This ensures that the chatbot can provide relevant and contextually meaningful information to users seeking guidance on installation procedures or troubleshooting.

5) So we have relatively small data in size divided into chunks. Now we solve the second problem mentioned in 4, which is textual data. The process of converting textual data to its nuneirc representaiton is called text embedding. I would use Sentence Transformers (e.g., the sentence-transformers library) because it provides high-quality embeddings specifically designed for capturing the meaning of sentences and paragraphs. This is crucial for understanding the detailed and technical content found in installation manuals. Since Sentence Transformers can be resource-intensive with larger models, we did the dim reduction previously. The reason behind it is that the quality of embeddings they that Sentence Transformers produce ensures that the chatbot can efficiently retrieve and generate meaningful responses based on the extracted content from the manuals. So I chose to use Sentence Transformers after dim reduction rather than vice versa.

6) Having all set up we can now store the vectorized information from PDF inside a db so that we can use it during Vector Similarity Search step (discussed later in the document). I would use Pinecone for storing the vector embeddings in a db because it is specifically designed for managing and retrieving vector embeddings efficiently. Pinecone provides a highly optimized, scalable, and easy-to-use solution for working with vector data, which is essential for quickly retrieving relevant information based on a query. It supports advanced indexing and similarity search algorithms, ensuring fast and accurate results. Additionally, Pinecone offers seamless integration with popular machine learning frameworks and embedding tools, making it an ideal choice for our chatbot.

## Part 2 - FROM USER TO LLM

1) User comes up with some question

2) Question is formulated as a prompt, where prompt will consist of

1. General instructions from the user. (E.g. Answer in Russian)
2. Prompt-before-Prompt (discussed in this video[https://www.youtube.com/watch?v=u47GtXwePms]). (E.g. Answer the question according to the information from my db . . . )
3. Prompt or actual quesiton

## Part 3 - FROM LLM TO DB, DB TO LLM, LLM to USER

1) At this point the actual question (3rd point) imposed by the user passes through the preprocessing and text embedding phases. Here we can again use sentence-transformations as described above.

2) LLM TO DB - After having the question in a form of a vector of numbers and a db where there are many such vectors, we do Vector Similarity Search. Meaning that in the storage of many vactors (the db) we look for the vectors(chunks) that are most similar to the given vector(question). After finding the top vectors in similarity, we can say that we managed to find the parts of the manual that contain the answer to the users question.

3) DB TO LL - So we pass the vector from db to the promp in LLM as the second component (prompt-before-prompt)

4) Lastly, combining the instructions from the user, the relevant data from db and the actual question of the user, the LLM (for e.g. we can use

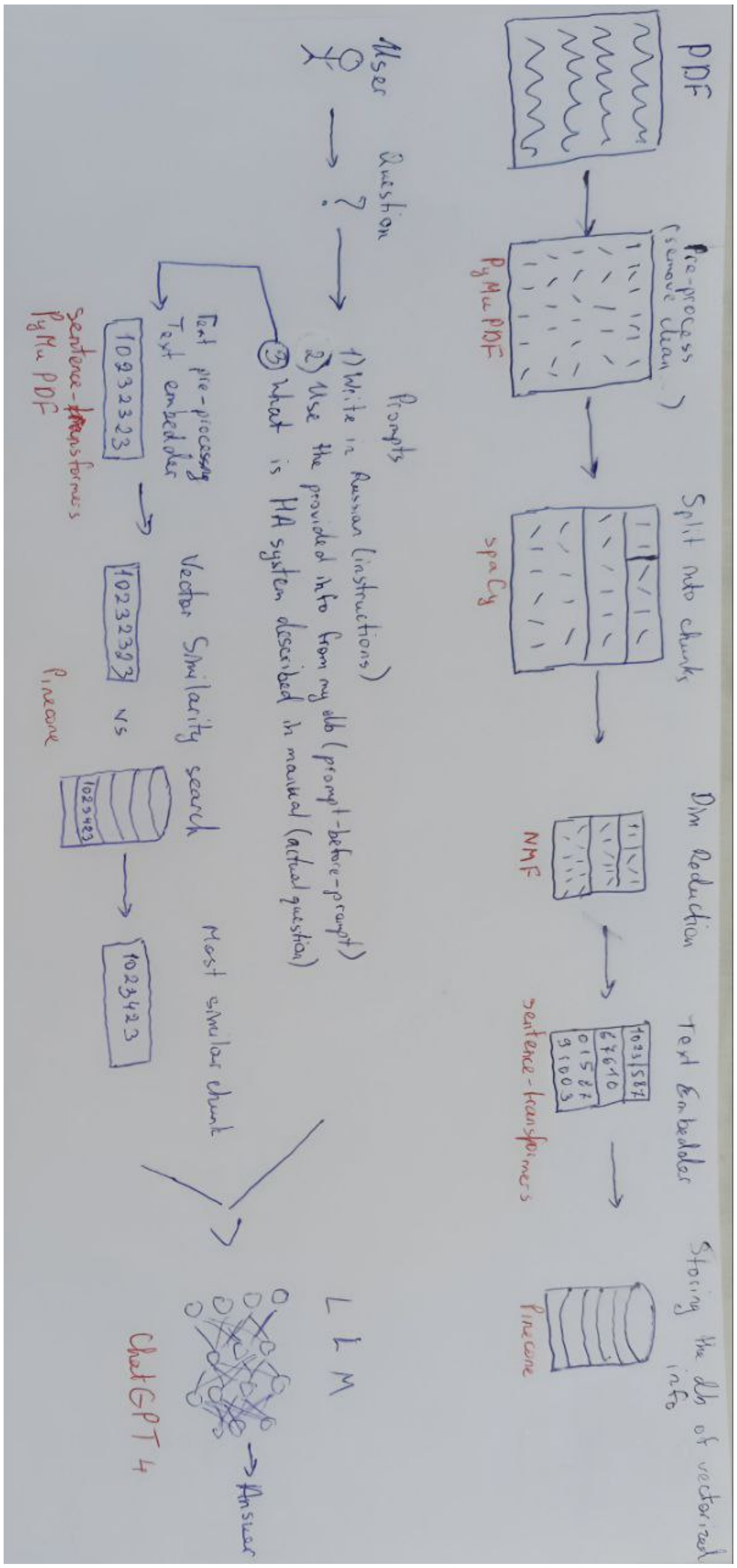

Figure 1: Image Alt Text

## SECTION 2: POTENTIAL PROBLEMS

Potential Problems could be

1) Computational Intensity of Text Embedding with Sentence Transformers which is discussed and solved to some extent above
2) Storing Large Volumes of Vector Data - Vector embeddings, especially when derived from extensive text, can consume significant storage space. For that we can implement advanced indexing techniques offered by Pinecone to efficiently store and retrieve vector embeddings. Techniques like approximate nearest neighbors (ANN) indexing can significantly reduce search times while managing storage effectively.
3) Similarity vector search - With large data, search can be costly. One solution that I can think of is implementing columnar db. In this case, instead of scanning through rows sequentially as we would do during the search in regular row-oriented dbs, we would selectively read only the columns needed for the search in columnar db.

## SECTION 3: USE CASES

### Questions that the chatbot will be able to answer:

1) What steps are involved in securely wiring a multi indoor unit to the outdoor unit according to the installation manual?

The chatbot will retrieve relevant information from the “Wiring” section inside the “Indoor Unit Installation” part of the manual stored in the db. Using vector similarity search, it identifies the most relevant sections based on the prompt. The chatbot then formulates a response using a language model, summarizing the steps involved in securely wiring the multi indoor unit to the outdoor unit ACCORDING TO THE MANUAL

2) What can be the most appropriate place for my air conditioner?

By processing the question through text embedding and vector similarity search, the chatbot extracts information from the “Safety Precautions” and “Choosing an Installation Site” sections of the manual. It gives this data to a language model, which generates a response recommending the most suitable locations for the air conditioner again based on the content retrieved from the manual.

3) How should I handle the refrigerant piping when installing a Daikin split air conditioner, and what precautions are necessary to prevent leaks and ensure safety during installation?

After converting the query into a numerical vector and searching for similar vectors in the db, the chatbot identifies relevant information primarily from the “Safety Precautions” section of the manual. It utilizes this information to formulate its response

4) How should I install the mounting plate for the indoor unit of my air conditioner?

The chatbot locates the answer in the “Installing the Mounting Plate” section within the “Indoor Unit Installation” part of the manual by performing a vector similarity search. It then uses the retrieved information to formulate a response, providing step-by-step instructions as outlined in the manual.

5) How do I perform trial operation and testing for my air conditioner, including checking supply voltage and setting temperatures?

Using text embedding and vector similarity search techniques, the chatbot identifies relevant sections from the “Trial Operation and Testing” part of the manual. It retrieves this information and passes it to a language model, which generates a response detailing the steps.

## Questions that the chatbot will fail to answer:

1) How do I ensure that the refrigerant piping meets local building codes in Armenia?

While the chatbot can retrieve general installation guidelines, it may not be equipped to understand and apply specific regional building codes unless these are explicitly mentioned in the manuals it has processed.

2) Can the Daikin air conditioner be installed in car?"

The chatbot's knowledge is limited to installation guidelines provided in typical residential settings. It does not have information on specialized installations which require different considerationsn.

3) What does the picture on page 10 mean?

The chatbot operates on textual data extracted from the manuals and converted into vector embeddings. It does not have the capability to interpret images

4) What technological advancements can I expect in future models of Daikin air conditioners?

The chatbot might not have enough data or training to be able to make logical assumptions about the future any time it is asked.

5) Is FTXS24LVJU the lastes model of Daiki air conditioners?

The latest models of products can change frequently. What might be the latest model today could be outdated in a few months. Without up-to-date information (which is not always up-to-date in our case but rather limited to the manual and it's recency), it's challenging to definitively state if a specific model is the latest.

## Resources used:

The one and only :) ChatGPT - https://chatgpt.com/share/50931d68-4a5b-4790-9046-528a42a88db3

YouTube - https://www.youtube.com/watch?v=qppV3n3YlF8, https://www.youtube.com/watch?v=u47GtXwePms

The choice of the tools was made according to my previous experience, the official documentations of the packages and ChatGPT