Improving RAG Performance with Re-Ranking & Spacy:-

As the dataset grows in size and complexity, selecting relevant information for returning tailored answers to complex queries becomes crucial. For this purpose, a family of techniques called Re-Ranking allows you to understand which chunks are important within the text. They reorder and filter the documents, ranking the most relevant ones first.Using Re-Ranking models as alternative techniques to embedding models. They take as input the query and the context and return similarity scores instead of embeddings

Using LLMs to capture the semantic information within the documents more efficiently.

Before applying these Re-Ranking approaches, let's evaluate what the baseline RAG system returns as the top three chunks for our second query:

Re-Ranking with FlagEmbeddingReranker

To retrieve the relevant chunks, we can use an open-source Re-Ranking model from Hugging Face, called the bge-ranker-base model.

Just like the OpenAI API, using Hugging Face requires you to get a user access token. You can create a user access token from Hugging Face .

Code Sample:-

HF_TOKEN = userdata.get('HF_TOKEN')

os.environ['HF_TOKEN'] = HF_TOKEN

2nd Technique:-

Of the Two innovative techniques spacy is one technique will reduce the unnecessary things in the text& reduces chunk size , where Chunks serve as the building blocks for various NLP tasks, enhancing(ner) named entity recognition, relation extraction, text summarization, and sentiment analysis. They also boost information retrieval, text classification, and keyword extraction. In essence, spaCy's chunking is like a blacksmith, forging the raw iron of text into the fine steel of meaningful phrases.There by optimizing the RAG , by reducing the chunk size & removing the non-required texts in the documents.

Below Code Snippet:-

```
text_splitter = SpacyTextSplitter(
    chunk_size = 150,
```

```
    chunk_overlap = 100
)

docs = text_splitter.create_documents([text])
```