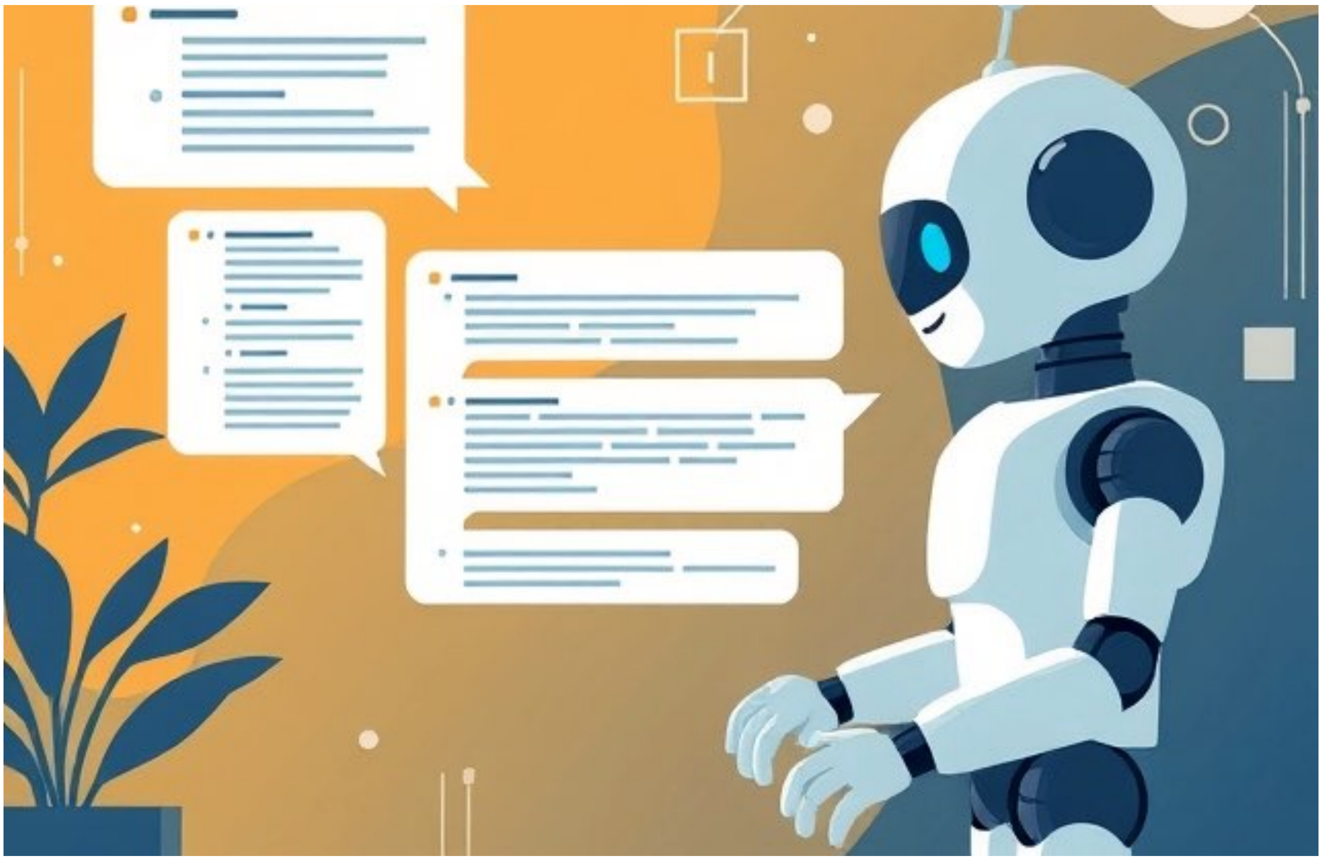
RETRIEVAL-AUGMENTED GENERATION (R.A.G.) WITH OLLAMA

AN INTERACTIVE CHATBOT FOR DOCUMENT ANALYSIS



Kamal Thampi

Project Overview

Goals

The primary goal of this project is to create a versatile, AI-driven chatbot that leverages Retrieval-Augmented Generation (RAG) techniques to analyse, summarise, and interact with uploaded PDF documents. This chatbot not only facilitates document-based query resolution but also serves as a powerful tool to test and optimise RAG model performance by adjusting a variety of parameters, including language model (LLM) selection, embedding models, chunk size, and LLM-specific hyperparameters.

Additionally, the tool can operate as a standalone chatbot, enabling general interactions without document uploads and making it adaptable for broader use cases.

Data Sources

The core data source for this chatbot comprises uploaded PDF documents. These documents are parsed, segmented into manageable chunks, and stored in a vector database. The vector database allows for efficient retrieval of content relevant to the user's query. Key aspects of the data flow include:

- **User-Uploaded PDF Files:** The app supports a wide range of documents, parsed into text chunks for processing.
- **External API (Ollama):** Ollama's LLM and embedding models provide contextual embeddings and generate responses based on retrieved document segments.
- **ChromaDB:** This vector database handles the embeddings for document chunks, supporting semantic search within document content.

Project Purpose and Additional Use Cases

In addition to enabling rich interaction with uploaded documents, the tool can:

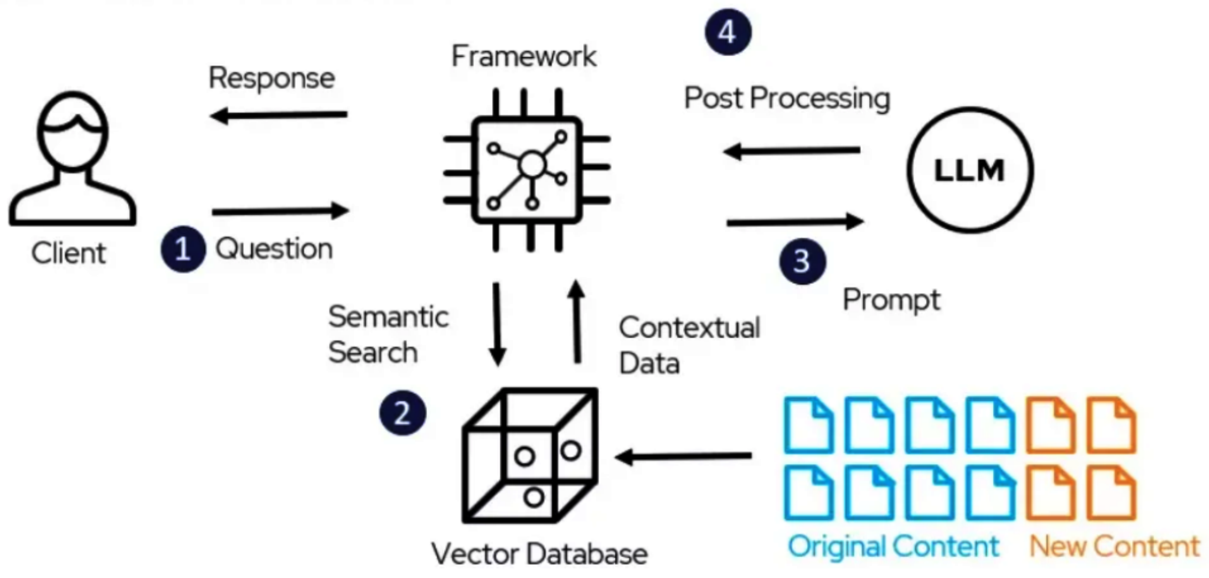
- Serve as a **testing platform** to evaluate RAG performance across various settings by switching LLMs, embedding models, chunk sizes, and LLM parameters.
- Facilitate conversations without document uploads, allowing users to interact with the LLM directly for general conversational purposes.

System Design

Architecture and Workflow

The RAG application employs a carefully structured architecture to balance efficient data processing with a smooth user experience. The architecture includes:

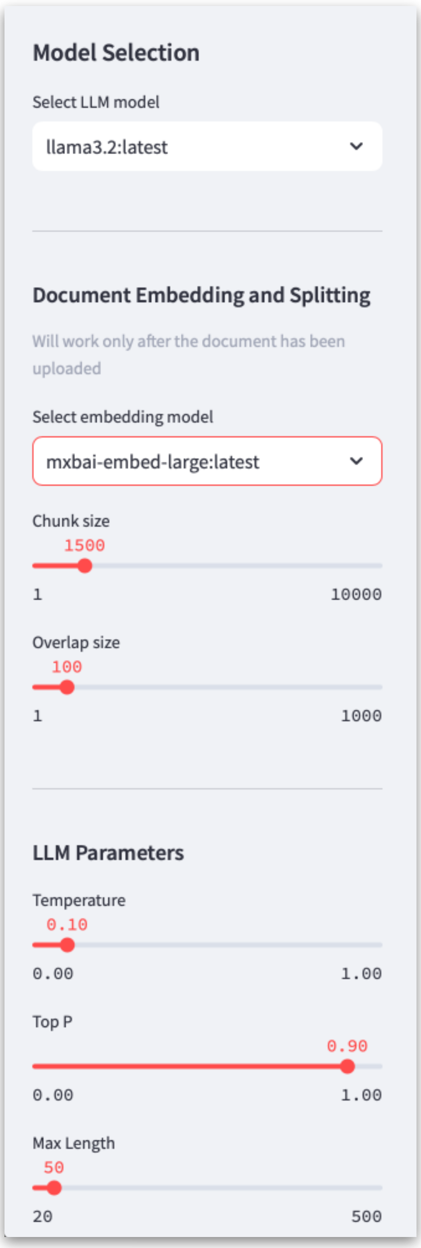
- **Document Parsing and Chunking:** Uploaded documents are parsed into text, split based on user-configurable chunks and overlap sizes, and then processed for embeddings. Recursive splitting enables the chatbot to work effectively with large documents.
- **Dynamic Vector Database Creation:** A unique vector database is created when a document is uploaded, or the embedding model is changed. This database stores chunks of text as vectorised embeddings, allowing for fast similarity searches on user queries.
- **Embeddings and Semantic Search:** Semantic embeddings from the selected model are used to compare the query with document segments, enabling contextual responses.
- **LLM Query Processing:** After retrieving the relevant document segments, the app passes these to the LLM. Various configurable parameters fine-tune response generation to meet user requirements.



Innovative Features

- **Parameter Flexibility:** The user can adjust parameters, such as chunk size and model settings, allowing for deep experimentation and insights into how different models perform on specific tasks. (Figure 1)
- **Side-by-Side Comparison:** The app enables users to view parsed document content alongside the original file, ensuring transparency in how the LLM interprets the text. (Figure 2)
- **Standalone Chatbot Mode:** Beyond document analysis, the tool can also engage in open-ended conversation, demonstrating its flexibility. (Figure 3)

Fig 1. -
Parameter Flexibility



Model Selection

Select LLM model: llama3.2:latest

Document Embedding and Splitting

Will work only after the document has been uploaded

Select embedding model: mxbai-embed-large:latest

Chunk size: 1500 (1 – 10000)

Overlap size: 100 (1 – 1000)

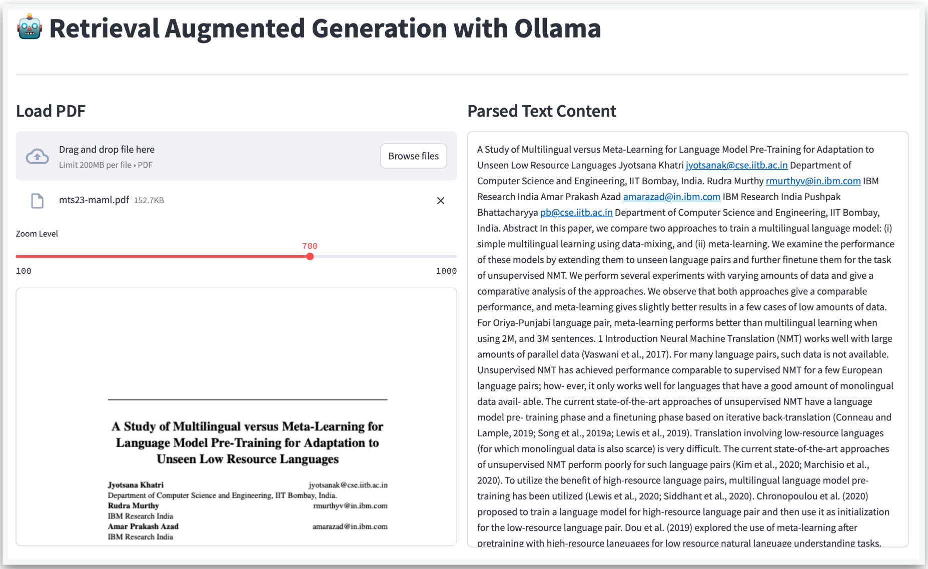
LLM Parameters

Temperature: 0.10 (0.00 – 1.00)

Top P: 0.90 (0.00 – 1.00)

Max Length: 50 (20 – 500)

Fig 2. -
Side-by-Side Comparison



🤖 Retrieval Augmented Generation with Ollama

Load PDF

mts23-maml.pdf 152.7KB

Zoom Level: 700 (100 – 1000)

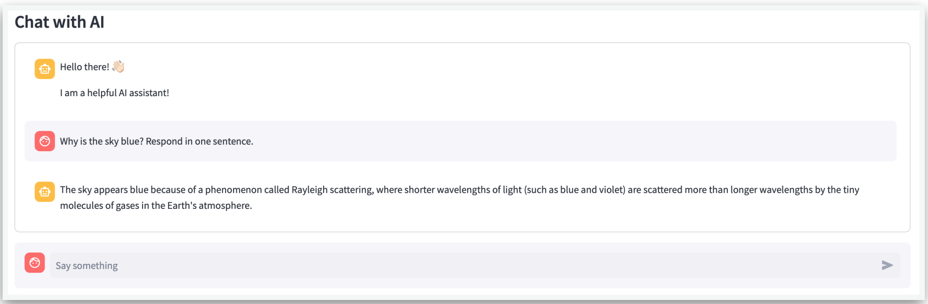
A Study of Multilingual versus Meta-Learning for Language Model Pre-Training for Adaptation to Unseen Low Resource Languages

Jyotsana Khatri jyotsanak@cse.iitb.ac.in
Department of Computer Science and Engineering, IIT Bombay, India.
Rudra Murthy rmurthyv@in.ibm.com
IBM Research India
Amar Prakash Azad amarazad@in.ibm.com
IBM Research India

Parsed Text Content

A Study of Multilingual versus Meta-Learning for Language Model Pre-Training for Adaptation to Unseen Low Resource Languages Jyotsana Khatri jyotsanak@cse.iitb.ac.in Department of Computer Science and Engineering, IIT Bombay, India. Rudra Murthy rmurthyv@in.ibm.com IBM Research India Amar Prakash Azad amarazad@in.ibm.com IBM Research India Pushpak Bhattacharyya pb@cse.iitb.ac.in Department of Computer Science and Engineering, IIT Bombay, India. Abstract In this paper, we compare two approaches to train a multilingual language model: (i) simple multilingual learning using data-mixing, and (ii) meta-learning. We examine the performance of these models by extending them to unseen language pairs and further finetune them for the task of unsupervised NMT. We perform several experiments with varying amounts of data and give a comparative analysis of the approaches. We observe that both approaches give a comparable performance, and meta-learning gives slightly better results in a few cases of low amounts of data. For Oriya-Punjabi language pair, meta-learning performs better than multilingual learning when using 2M, and 3M sentences. 1 Introduction Neural Machine Translation (NMT) works well with large amounts of parallel data (Vaswani et al., 2017). For many language pairs, such data is not available. Unsupervised NMT has achieved performance comparable to supervised NMT for a few European language pairs; how- ever, it only works well for languages that have a good amount of monolingual data avail- able. The current state-of-the-art approaches of unsupervised NMT have a language model pre- training phase and a finetuning phase based on iterative back-translation (Conneau and Lample, 2019; Song et al., 2019a; Lewis et al., 2019). Translation involving low-resource languages (for which monolingual data is also scarce) is very difficult. The current state-of-the-art approaches of unsupervised NMT perform poorly for such language pairs (Kim et al., 2020; Marchisio et al., 2020). To utilize the benefit of high-resource language pairs, multilingual language model pre-training has been utilized (Lewis et al., 2020; Siddhant et al., 2020). Chronopoulou et al. (2020) proposed to train a language model for high-resource language pair and then use it as initialization for the low-resource language pair. Dou et al. (2019) explored the use of meta-learning after pretraining with high-resource languages for low resource natural language understanding tasks.

Fig 3. -
Standalone Chatbot Mode



Chat with AI

Hello there! 👋🏻

I am a helpful AI assistant!

Why is the sky blue? Respond in one sentence.

The sky appears blue because of a phenomenon called Rayleigh scattering, where shorter wavelengths of light (such as blue and violet) are scattered more than longer wavelengths by the tiny molecules of gases in the Earth's atmosphere.

Technical Implementation

AI Model Quality

- **Embedding Models:** The app uses Ollama's embedding models to vectorise document chunks, enabling the chatbot to retrieve contextually relevant information with high accuracy.
- **Language Model (LLM):** Various LLMs are available for query processing, allowing users to test and optimise for response quality. Each LLM can be fine-tuned with parameters such as temperature, top-k sampling, and penalties, refining response generation.

Code Efficiency

- **Session State Management:** By leveraging Streamlit's `st.session_state`, the app optimises performance by reusing existing databases and embedding models where possible. This approach reduces redundant processing and improves runtime efficiency.
- **Efficient Data Handling with ChromaDB:** ChromaDB manages vector embeddings and retrieval, ensuring fast and accurate responses even for large documents.

Seamless Integration with APIs

The app's seamless interaction with Ollama and ChromaDB APIs highlights the system's adaptability. Document uploads, embeddings, LLM queries, and vector searches occur with minimal latency, ensuring a responsive user experience.

Source code

The code is published on GitHub at
https://github.com/khamsakamal48/Local-RAG-with-Ollama/

Pre-requisites

The application expects that Ollama is installed on your environment and accessible on http://localhost:11434 for API connections. Please refer to the documentation of Ollama, as mentioned on https://ollama.com/, to install it in your environment.

Load at least one Large Language Model and appropriate Embedding Model from https://ollama.com/library

User Experience.

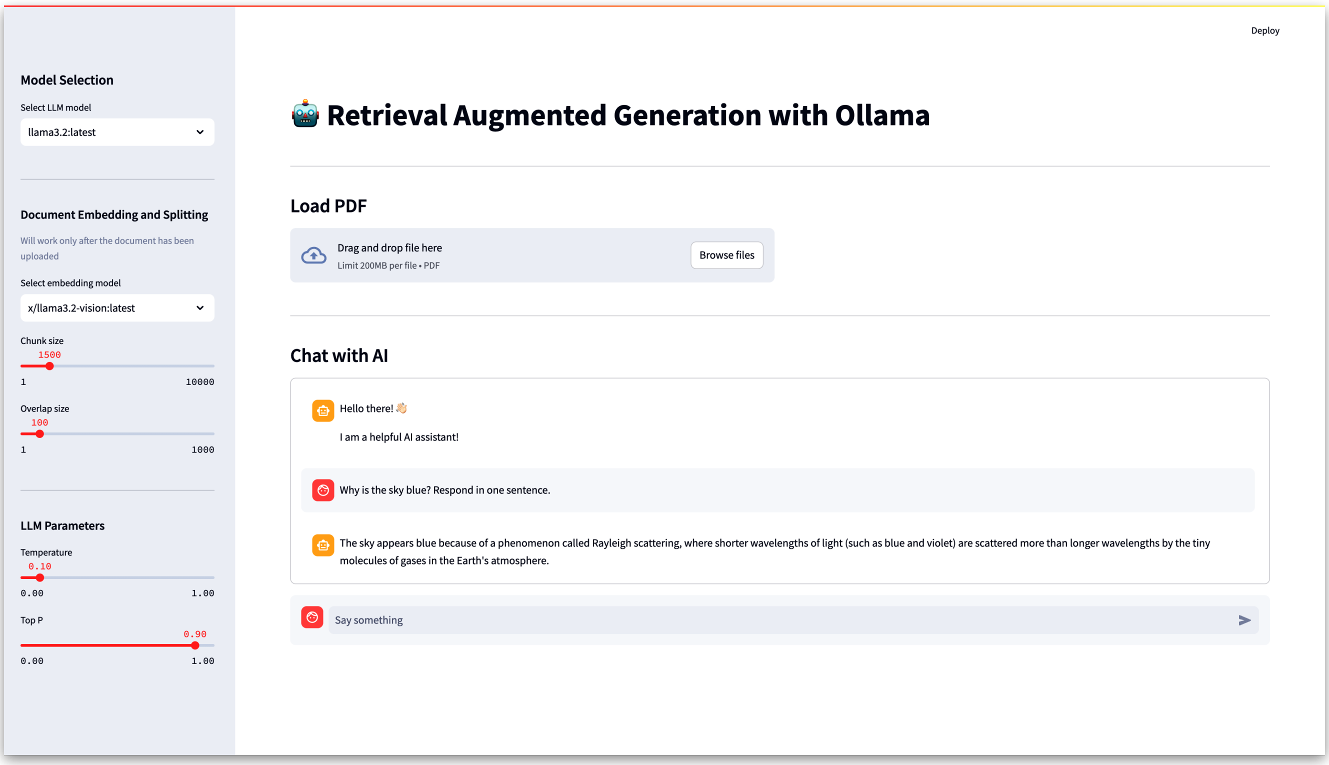
User-Friendliness and Interactivity

The app's layout and sidebar options make it easy to navigate:

- **Intuitive Sidebar Options:** Model selection, chunk size, overlap, and LLM parameters are available in the sidebar, enabling real-time adjustments.
- **Document Upload and Comparison:** Users can upload documents and view parsed content side-by-side with the original file, promoting trust in the AI's data handling.
- **Flexible LLM Configuration:** The extensive range of parameters allows users to customise LLM output. These parameters include temperature, top-k, and top-p, making it ideal for those testing response precision.

Video Demo

Asking ChatBot:
"Why is the sky blue? Respond in one sentence."



Model Selection

Select LLM model: llama3.2:latest

Document Embedding and Splitting

Will work only after the document has been uploaded

Select embedding model: x/llama3.2-vision:latest

Chunk size: 1500 (1 – 10000)

Overlap size: 100 (1 – 1000)

LLM Parameters

Temperature: 0.10 (0.00 – 1.00)

Top P: 0.90 (0.00 – 1.00)

Retrieval Augmented Generation with Ollama

Load PDF

Drag and drop file here
Limit 200MB per file • PDF

Chat with AI

Hello there! 👋
I am a helpful AI assistant!

Why is the sky blue? Respond in one sentence.

The sky appears blue because of a phenomenon called Rayleigh scattering, where shorter wavelengths of light (such as blue and violet) are scattered more than longer wavelengths by the tiny molecules of gases in the Earth's atmosphere.

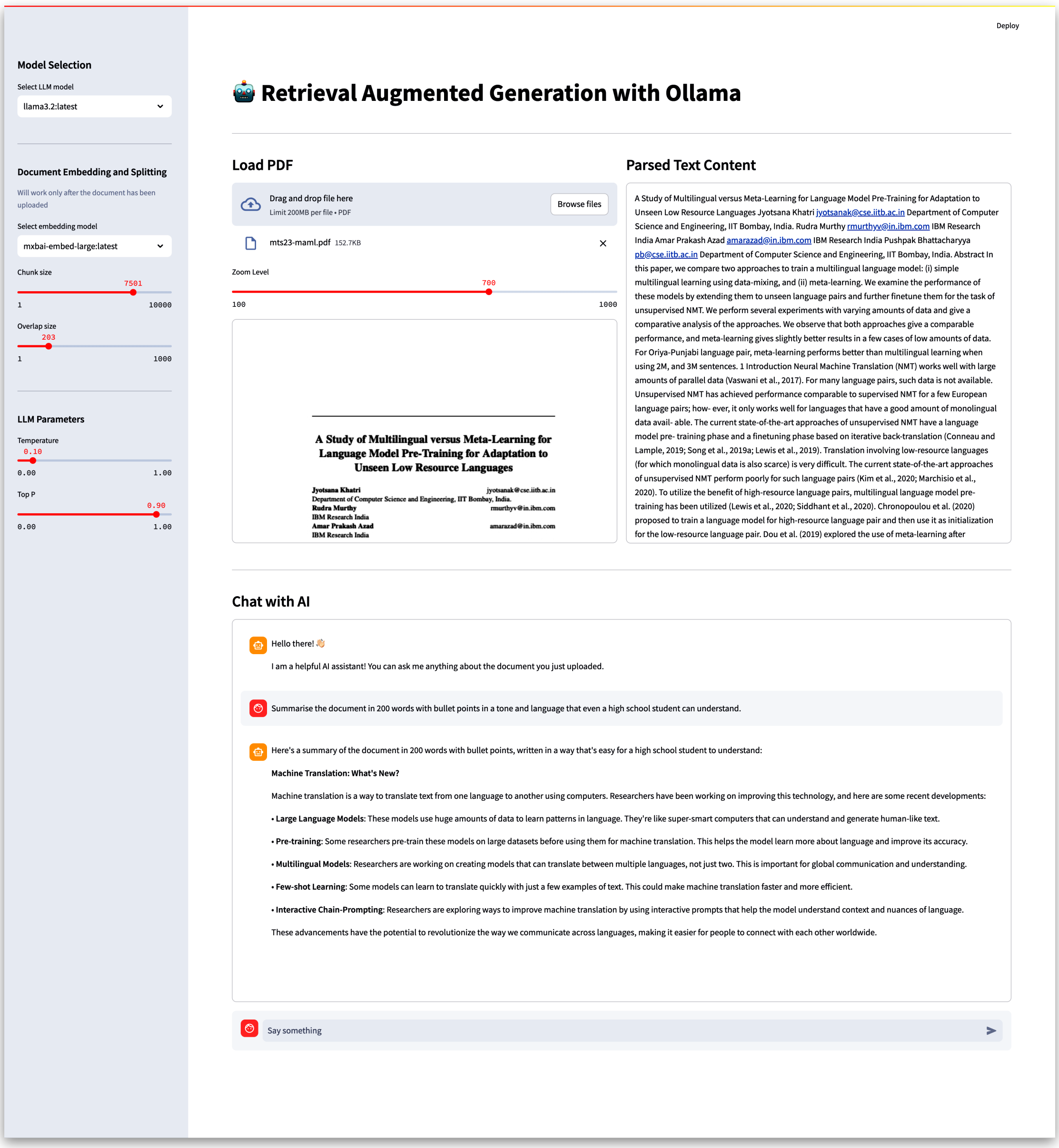
Used LlaMa 3.2 as LLM with temp: 0.1 and top-p: 0.9 & max_length: 50

Clear and Contextual Responses

The app provides clear responses relevant to user queries, emphasising content extracted from the document or conversational context. For example, users querying about specific sections of the document receive responses directly related to those sections. This clarity makes the chatbot highly usable for document analysis.

Video Demo

Asking ChatBot:
"Summarise the <u>document</u> in 200 words with bullet points in a tone and language that even a high school student can understand."


Deploy

Model Selection

Select LLM model: llama3.2:latest

Document Embedding and Splitting

Will work only after the document has been uploaded

Select embedding model: mxbai-embed-large:latest

Chunk size: 7501 (1 – 10000)

Overlap size: 203 (1 – 1000)

LLM Parameters

Temperature: 0.10 (0.00 – 1.00)

Top P: 0.90 (0.00 – 1.00)

Retrieval Augmented Generation with Ollama

Load PDF

Drag and drop file here
Limit 200MB per file • PDF
Browse files

mts23-maml.pdf 152.7KB

Zoom Level: 700 (100 – 1000)

A Study of Multilingual versus Meta-Learning for Language Model Pre-Training for Adaptation to Unseen Low Resource Languages

Jyotsana Khatri jyotsanak@cse.iitb.ac.in
Department of Computer Science and Engineering, IIT Bombay, India.
Rudra Murthy rmurthyv@in.ibm.com
IBM Research India
Amar Prakash Azad amarazad@in.ibm.com
IBM Research India

Parsed Text Content

A Study of Multilingual versus Meta-Learning for Language Model Pre-Training for Adaptation to Unseen Low Resource Languages Jyotsana Khatri jyotsanak@cse.iitb.ac.in Department of Computer Science and Engineering, IIT Bombay, India. Rudra Murthy rmurthyv@in.ibm.com IBM Research India Amar Prakash Azad amarazad@in.ibm.com IBM Research India Pushpak Bhattacharyya pb@cse.iitb.ac.in Department of Computer Science and Engineering, IIT Bombay, India. Abstract In this paper, we compare two approaches to train a multilingual language model: (i) simple multilingual learning using data-mixing, and (ii) meta-learning. We examine the performance of these models by extending them to unseen language pairs and further finetune them for the task of unsupervised NMT. We perform several experiments with varying amounts of data and give a comparative analysis of the approaches. We observe that both approaches give a comparable performance, and meta-learning gives slightly better results in a few cases of low amounts of data. For Oriya-Punjabi language pair, meta-learning performs better than multilingual learning when using 2M, and 3M sentences. 1 Introduction Neural Machine Translation (NMT) works well with large amounts of parallel data (Vaswani et al., 2017). For many language pairs, such data is not available. Unsupervised NMT has achieved performance comparable to supervised NMT for a few European language pairs; how- ever, it only works well for languages that have a good amount of monolingual data avail- able. The current state-of-the-art approaches of unsupervised NMT have a language model pre- training phase and a finetuning phase based on iterative back-translation (Conneau and Lample, 2019; Song et al., 2019a; Lewis et al., 2019). Translation involving low-resource languages (for which monolingual data is also scarce) is very difficult. The current state-of-the-art approaches of unsupervised NMT perform poorly for such language pairs (Kim et al., 2020; Marchisio et al., 2020). To utilize the benefit of high-resource language pairs, multilingual language model pre- training has been utilized (Lewis et al., 2020; Siddhant et al., 2020). Chronopoulou et al. (2020) proposed to train a language model for high-resource language pair and then use it as initialization for the low-resource language pair. Dou et al. (2019) explored the use of meta-learning after

Chat with AI

Hello there! 👋

I am a helpful AI assistant! You can ask me anything about the document you just uploaded.

Summarise the document in 200 words with bullet points in a tone and language that even a high school student can understand.

Here's a summary of the document in 200 words with bullet points, written in a way that's easy for a high school student to understand:

Machine Translation: What's New?

Machine translation is a way to translate text from one language to another using computers. Researchers have been working on improving this technology, and here are some recent developments:

• **Large Language Models**: These models use huge amounts of data to learn patterns in language. They're like super-smart computers that can understand and generate human-like text.

• **Pre-training**: Some researchers pre-train these models on large datasets before using them for machine translation. This helps the model learn more about language and improve its accuracy.

• **Multilingual Models**: Researchers are working on creating models that can translate between multiple languages, not just two. This is important for global communication and understanding.

• **Few-shot Learning**: Some models can learn to translate quickly with just a few examples of text. This could make machine translation faster and more efficient.

• **Interactive Chain-Prompting**: Researchers are exploring ways to improve machine translation by using interactive prompts that help the model understand context and nuances of language.

These advancements have the potential to revolutionize the way we communicate across languages, making it easier for people to connect with each other worldwide.

Say something

LLM: LlaMa 3.2, **Embedding Model:** Mixed Bread AI, **Chunk Size:** 7500
Document: <u>https://www.cse.iitb.ac.in/~pb/papers/mts23-maml.pdf</u>

Challenges Faced

Embedding Dimension Mismatch

Switching embedding models with differing vector dimensions initially resulted in dimension mismatch errors. Addressing this required dynamically resetting the vector database whenever embedding models were changed to ensure consistency.

Efficient Session Management

Efficiently managing session states was crucial due to the app's flexible parameter settings. Implementing logic to reset or retain databases and session states depending on the changes made (e.g., document upload, embedding model switch, chunk size change) improved efficiency and resolved frequent database errors.

Parameter Tuning for Optimal Response Generation

The variety of LLM parameters posed a challenge in maintaining response quality. Iterative testing and feedback loops helped optimise defaults that balance coherence, relevance, and user control in response generation.

Conclusion

This RAG-based chatbot application leverages the power of Ollama's embedding models and LLMs to provide a robust document analysis and conversational tool. By enabling dynamic parameter adjustment, the app allows users to explore RAG's full potential and engage with uploaded documents or directly with the LLM for broader interactions. The carefully designed user interface, efficient session management, and seamless API integration make it a highly adaptable solution suitable for a wide range of applications in document analysis and interactive AI.

This tool demonstrates the versatility of RAG and its potential in real-world applications, making it an invaluable resource for both technical exploration and end-user engagement.