Document Query Response Using Microsoft-Phi-3-Mini-4K

M.Krishnaraj
Department of Computer Science and Engineering
St. Joseph's Institute of Technology,
Chennai, India
monykrishnaraj@gmail.com

S.Eben Isaacraj
Department of Computer Science and Engineering
St.Joseph's Institute of Technology,
Chennai, India
ebenisaacraj2003@gmail.com

V.Haris Jai Soorya
Department of Computer Science and Engineering
St. Joseph's Institute of Technology,
Chennai, India
harisjaisooryavasanth@gmail.com*

2025 International Conference on Emerging Smart Computing and Informatics (ESCI) | 979-8-3315-1568-3/25/$31.00 ©2025 IEEE | DOI: 10.1109/ESCI63694.2025.10988177

***Abstract*— The role of automated mechanisms such as chatbots has improved in today's environment of rapid technological progress which is majorly attributed to the growing use of Artificial Intelligence (AI) based solutions. The ability of such systems to interact with questions concerning PDFs can completely transform support and after sales services in the automotive field where time and resources can be saved. Within this context, this work analyses the application of Large Language Models (LLMs) in the development of RAG based applications to create a smarter and confidentiality-driven chatbot to preserve the privacy of individuals and companies alike.**

Index Terms—Large Language Models (LLM),Retrieval Augmented Generation(RAG), embeddings

I. Introduction

Preference for digitalized data over physical data has become the new norm, which has led to an increasing need for efficient and effective methods to manage and retrieve information from digital documents. When faced with large files like PDFs containing hundreds of pages, CSVs with massive datasets, or even simple Word files, manual searching can quickly become overwhelming and time consuming as stated in [1]. To solve this problem, we have developed a system that combines advanced Large Language Models (LLMs) with embedding techniques to streamline data retrieval. Our approach uses Ollama integrated Phi-3 Mini LLM, paired with the HuggingFace embedding model, for retrieval-augmented generation (RAG). This system leverages vector search capabilities powered by VectorStoreIndex and is integrated into a user-friendly interface built on Streamlit. With this setup, users can interact with large documents seamlessly, allowing professionals and researchers to quickly find relevant information, also promoting confidentiality due to the localized nature of the application.

II. Related Works

The field of document query response systems and information retrieval through chatbots has witnessed significant strides owing to the advancement of natural language processing (NLP) as well as large language models (LLMs). The primary focus of these early systems was on the use of keyword-oriented searching where most context was usually lost during a complex request, resulting in many instances of frustration due to undesired or partial outcomes. Similarly with the aid of Empirical Models such as BERT and GPT Models, the embedding based approaches became the vogue for better capturing the meanings of documents. With the use of several techniques, consisting of chunking and sentence embeddings, these systems decompose large documents into smaller parts for easier processing and understanding but while maintaining the context and improving retrieval accuracy.

In recent work, efficient vector-based document search methods have been explored. Talks about the most well-known library FAISS, which allows for cost-effective resemblance searching within clusters, appear through other embedding models meanwhile hugging face's Bge embedding has made use of the more detailed analysis of our features. In that sense, these models are designed for bots more evenly allowing working with many formats at once such as PDF documents..

In contrast, this project integrates embedding based search with the Phi-3-Mini-4k [2] which is a large language model designed to comprehend and respond to users' queries in a cohesive manner. The approach presented in this paper is fundamentally different from conventional ones which employ interaction with FAISS. Rather, it applies document searches using LlamaIndex for index based searching, document incorporation and retrieves additional documents with HuggingFace embeddings.

III. PROBLEM STATEMENT

Text-based datasets come with enormous challenges in research, legal, business-oriented environments. In most cases, traditional methods will involve looking for useful information by physically scanning through many large documents i.e. PDF files or CSV files which can take a lot of time and even lead to the loss of important information. This is made worse in the current days due to the increasing amounts of data and absence of sophisticated tools for quick and precise search and retrieval as every professional understands how challenging it is to get useful information fast and accurately.

The outcome of this project is to address and mitigate these issues through the use of the more recent and sophisticated HuggingFace libraries along with transformers and LLMs. More specifically, this solution relies on LlamaIndex for text manipulation, HuggingFace embeddings for meaning representation, and the Phi-3 Mini 4K Instruct LLM for information completion. This way, it's possible to reduce and almost eliminate the time lost looking for information as well as avoid common mistakes.

In addition, this project focuses on confidentiality of data, because the system is fully on premise and does not imply any document uploads to any external systems such as OpenAI's ChatGPT or other cloud-based solutions. This means that all the sensitive documents are kept safe, providing an answer to the questions related to data safety and misuse of those document's content for model's training or other purposes without a user's permission. This way, while data processing is carried out within local bounds, this solution guarantees not only protection of the sensitive information, but also makes compliance with privacy policies possible for the organizations that deal with private documents.

IV. Methodology

A. System Overview

The LLM which we are using is a recently introduced model known as Phi 3 developed by Microsoft. Reference [2] states that the Phi-3 model is trained on only 3.8 billion parameters, which is extremely small and less resource intensive. This was the primary reason we chose this LLM due to its miniature nature, unlike BERT, llama, etc which require supercomputers. According to [2], the phi-3 can run on mobile phones as it occupies about 1.8 GB of RAM which is a first for LLMs.

Phi-3 interacts with user queries by using its lightweight architecture to process and generate responses quickly, even in low end devices with lack of GPU. The Phi-3 LLM's role is to be an efficient chatbot that generates accurate and quick responses to document-based queries made by the user, while being resource friendly. This is unlike other LLMs like llama, etc.

A general overview of the system is presented in the block diagram of the system as follows:

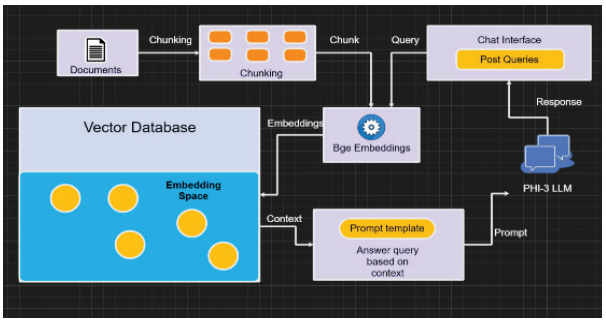


Fig. 1. System overview

B. Data collection

The chatbot leverages the phi-3 model with ollama to provide precise and efficient results, granted there is the required context to answer prompts. This context is taken from the textual data present in the PDFs.

This textual data is extracted using SimpleDirectoryHeader, from the llamaIndex library, which extracts the data by reading the pdf files in the specified directory. SimpleDirectoryReader converts the files with the appropriate extensions into a structured format suitable for indexing and querying.

Before Data collection, the uploaded document via Streamlit undergoes preprocessing steps such as noise removal to filter headers, text segmentation to divide text into smaller chunks and error handling to handle corrupted files.

To commence Data Collection, we create a conda environment with llamaIndex and Streamlit installed. The directory containing the uploaded files is read by SimpleDirectoryReader, which extracts the text from it. Finally, we use VectorStoreIndex to store the embeddings created by HuggingFaceEmbedding (BAAI/bge-large-en-v1.5 model), and prepare the data for querying, enabling similarity search and RAG.

C. Retrieval Augmented Generation

Normally, there are 2 ways to train a LLM to perform a custom task, which is fine-tuning and RAG. This system uses RAG over fine-tuning because, as stated by [3], RAG excels in dynamic environments. Whereas, fine-tuning use unsupervised learning to make LLMs learn factual information, which is very resource intensive and provides inaccurate responses due to the dynamic nature of input (changing PDFs)

Reference [3] made a performance evaluation of LLMs to test RAG and FT. According to the result, RAG consistently outperforms FT. However, this does not mean fine-tuning is bad for LLMs, the choice between RAG and fine-tuning purely depends on the dynamic/static nature of the data, resource limitations and customization.

So, the VectorStoreIndex, which stores the embeddings of the text from document, also accepts the query provided by the user via Streamlit. Due to which, the LLM is able to retrieve the most relevant document chunks based on similarity to the query (Retrieval). As for Augmentation, the retrieved chunks of document text are given to the LLM for additional context. This is done by adding the context in the prompt template which augments in addition to the phi-3's pre-trained knowledge.Finally, the phi-3 LLM generates answers based on the provided query and the context from retrieval and augmentation, forming what we know as Generation. Hence why RAG [3] is being used to create the document chatbot. However, RAG does have its limitations, which include hallucinations and generation difficulty.

To optimize the efficiency of the RAG process, some of the potential ways are to use parallel processing of text extraction and data cleaning at the same time which reduces bottlenecks. We also considered using caching mechanisms to store frequently used query embeddings to minimize time taken to generate a response.

D. Embeddings

Next is the process of embedding text from the PDF. Embedding can be described, in basic terms, as taking text (or any other form of data) and transforming it into vectors that are capable of capturing the meaning inherent in the data. For example, in Natural Language Processing (NLP), word, sentence, or document embeddings would be represented as dense vectors in a vector space.

Despite the inherent difficulties, attempts have been made to convert textual data into mathematical formats that models can work with. For instance, the formats can be suitable for similarity searches, clustering and even as inputs for the further stages in the process (classification, generation) Embeddings have helped bridge this gap by enabling models to such processing of text based content [3].

This PDF bot utilizes an embedding model of Hugging Face named BAAI/bge-large-en-v1.5 to vectorize the contents of the PDF document enabling it to carry out subsequently similarity search queries on an embedded vector space.

There are several embedding models to choose from like OpenAI, BGE, BERT, etc. But some of them have a limitation to tokenize number of texts per day and to tokenize more would be expensive. OpenAI is apt example of being a pricey embedding model.

E. Indexing

Indexing is organizing big text or document collections aiming for ease of search and retrieval [3]. In your case, however, indexing intends to convert the content of PDF files to a searchable feature using embeddings for quicker lookups when engaging with the chatbot. The indexing process is handled mostly by VectorStoreIndex from LlamaIndex (previously known as GPT Index). After the document embeddings are issued, they go to the VectorStoreIndex to construct the index. After the document embeddings are issued, they go to the VectorStoreIndex to construct the index: Now, the query engine is configured to work with the indexed embeddings to answer user issues. The query_engine is the part that performs the similarity queries to the index. It retrieves the parts of your document that are the most relevant to the person's query, allowing precise and fast answering. If indexing is not done, the system will have to search the entire raw text of the document for each and every query commonly resulting in latency. Indexing facilitates the fastening of lookup and retrieval. Embedding indexing enables the system to know what the query and the documents mean making it more precise than keyword-based searches.

F. Query Processing and Response Generation

Processing queries is an essential component of the system that helps in achieving the optimal answer for the user's concern in terms of seeking information and generating a response. Rather than directly acting on the query placed by the user, the system first finds the most relevant document chunks from the indexed content via vector similarity search. Such retrieval makes sure that only relevant parts of the document are scanned for, thus minimizing any non-informative. The system also adopts the Prompt Template to enhance the answer, ensuring that the context is properly briefed prior to the answer. With such arrangement, the resulting query will be response in an organized and direct manner.

In case of response generation, the phi-3 LLM will be given a filtered content that is drawn from the retrieved information for an understandable generation of response. Through methods such as prompt engineering, the system is able to control the LLM and made it answer contextually and straight to the point. This is effective in minimizing the bulk of unnecessary information and maintaining the succinctness and relevance of the answers. By modifying and improving the quality of the prompts, the system effectively manages to prevent the LLM from being provided with too much or too little information which in turn enhances the clarity and accuracy of the answer.

G. Challenges

The most troubling challenge the system may face at times is the concept of hallucinations [4]. Reference [4] states that hallucinations are generation of nonsensical or unfaithful data with context of the provided source content. The chatbot often produces an Input-conflicting hallucination [4], which is content that deviates from the input source material provided by users. Hallucinations are typically a result of RAG, whereas fine-tuning significantly reduces the chance of a hallucination from ever occurring in the context of LLMs.

However, it does not mean the problem of hallucination can't be cured, [4] lists out several solution such as performing Supervised Fine-Tuning, which involves annotation or collecting massive-task instruction-following data. Curation of data has proven to be quite an effective counter to hallucination. But performing SFT could also create an another problem such as behaviour cloning[4], it means the model learns directly from imitating the expert's actions, which could mean that the LLM will no longer derive context from the user's PDF, resulting in inaccurate response generation.

This system utilizes the HuggingFaceEmbedding model BAAI/bge-large-en-v1.5, which is pretty large and has high resource usage. The quality of embeddings and the richness of contextual understanding it can provide are some of the benefits; however, it still poses certain challenges and disadvantages due to its size.High embedding models may consume a lot of memory and compute resources, which may pressure user system, especially when they are run in Streamlit or smaller cloud instances. This can lead to slower processing times and even exhaust memory, thereby negatively affecting the performance of the chatbot. The model is just so large that it takes a considerable amount of time to generate the embeddings. In this way, an enormous amount of time is taken to give responses to questions from users, especially when handling large documents or multiple files. Users would end up waiting for such long periods of time, which ultimately affects the overall user experience and responsiveness of the chatbot. The size of the embedding model increases latency, especially in processing real-time queries. That latency may be frustrating for users when waiting for the embeddings to be created before the response can return if the system is used in a live setting or interactively.Handling such big models calls for proper memory management, as implemented by garbage collection (gc.collect()). Otherwise, there may sometimes occur memory leaks or degradation in the efficiency of performance. In general, the large embedding model is very good at delivering strong contextual embeddings, but it has other disadvantages related to performances, scalability, and resource demand. It's better suited for less resource and real-time-constrained applications. Feel free to experiment with smaller embedding models that achieve a performance versus computational efficiency balance.

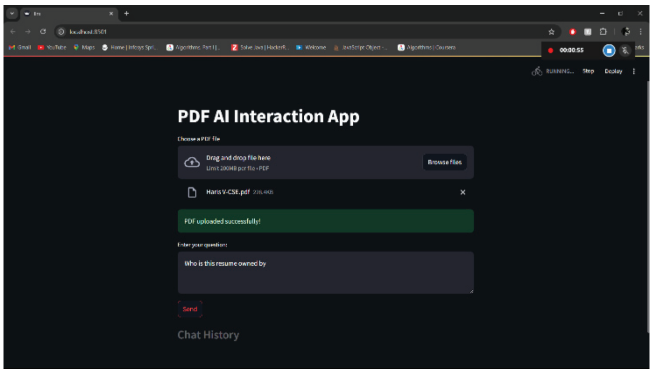
However, to make the chatbot more reliable, we enforced the use of confidence scores to ensure the resultant responses are indeed valid and context-accurate. Confidence scoring can be checked via ROUGE scores. If the scores are low, the chatbot can ask the user for more context to avoid generating inaccurate answers.

H. Deployment

Streamlit is used for the code's deployment. This will enable users to upload and interact with PDF documents in a very interactive and user-friendly interface. The users will use the components of Sidebar provided by Streamlit for intuitive selection between Phi-3 and Llama-3 and, therefore, uploading some documents to process. Following the successful indexing of the document, the chatbot will now interract with the users in real time and use the real-time responses generated from the content of the document. Streamlit's layout system allows for clean organization, so that the PDF is previewed in an embedded viewer, so that users can review the document they're querying. The chat history feature maintains a clear record of all queries users make and responses it gives back, so that there is a smooth conversational flow across it. Besides, session state management features also ensure such an application maintains consistency across interactions even if the page gets refreshed or is revisited. Meanwhile, the use of tempfile in Streamlit allows for efficient temporary storage of documents, and the powerful LLM guarantees proper processing of a user's queries. Furthermore, the availability of a "Clear" button makes it possible to easily reset the chat and begin a new session.

In terms of enhancing the real-time process of the chatbot, we implemented partial streaming of responses or on-the-run responses rather than let the user wait for the whole response to complete. GPUs could also be used to ensure real-time generation of the responses, however using GPUs are completely optional in case of resource limitations.

V. RESULT AND OUTPUT



PDF AI Interaction App

Choose a PDF file

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

Haris V-CSE.pdf

PDF uploaded successfully!

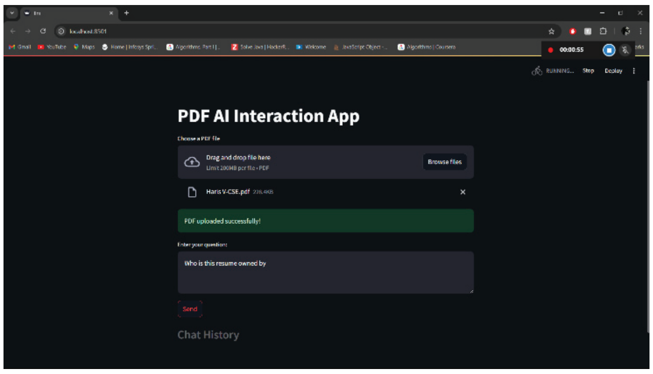
Enter your question:

Who is this resume owned by

Send

Chat History

Fig. 2. Home screen of the pdf chatbot



PDF AI Interaction App

Choose a PDF file

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

Haris V-CSE.pdf

PDF uploaded successfully!

Enter your question:

Who is this resume owned by

Send

Chat History

Fig. 3. Output and resultant query

Fig 2 shows the web application interface for homepage of the PDF chatbot with streamlit UI

Fig 3 shows the web application interface and sidebar to upload PDFs and the generated response to the provided query.

VI. PERFORMANCE ANALYSIS

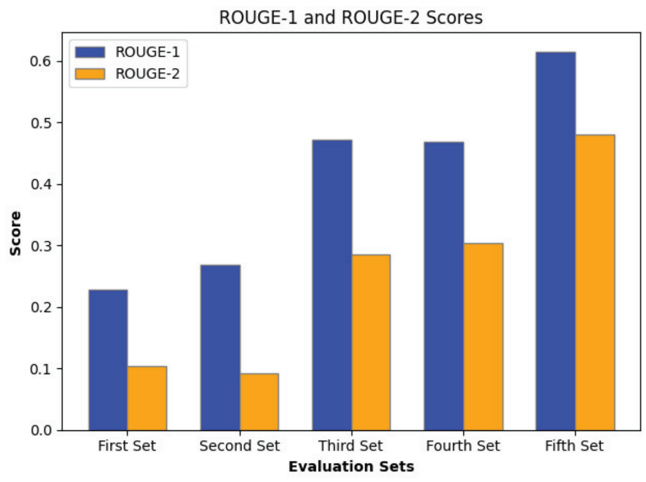


Fig. 4. ROUGE scores

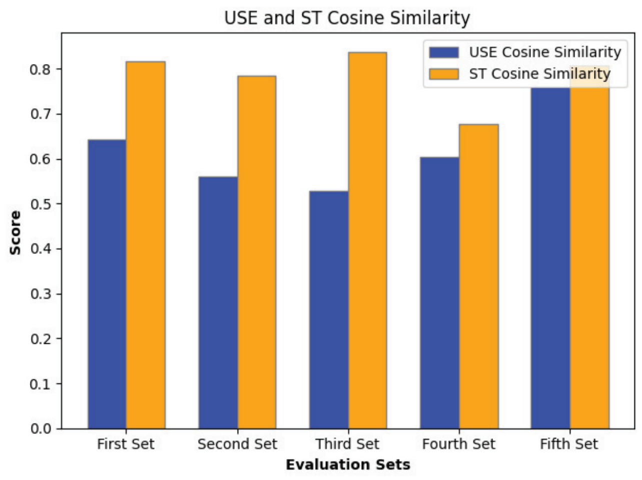


Fig. 5. Cosine similarity scores

The performance of the PDF-based chatbot system is measured using the range of metrics, from ROUGE to cosine similarity metrics based on the Universal Sentence Encoder (USE) and Sentence Transformer (ST) embeddings. Graphs can now display the performance across all of the different evaluation sets. [5] evaluates the LLM performance using E2E metric which uses a pre-defined set of golden answers, golden answers are the expertly written responses to pre-defined queries. So, the LLM's response is tested against the golden answer to evaluate their similarity by comparing ROUGE and cosine similarity scores. For summarization tasks, ROUGE metrics are used to calculate overlap between the predicted and reference sentences. In detail, ROUGE-1 measures overlap using unigrams; in contrast, ROUGE-2 focuses on overlap based on bigrams, meaning it uses two consecutive words.

As for generating the above two figures that is fig 4 and fig 5, we had created a set of golden answers that belong to predefined queries. This is used to compute and compare the cosine similarity and rouge scores between the golden answers and phi-3 generates responses. The cosine similarity scores are computed using python's USE library and the rouge scores are done using rouge_scorer python library. These scores are then stored then in a dictionary for each query and then visualized as a bar chart using python's matplotlib library.

Fig 4 shows that ROUGE-1 scores appear to gradually improve from the first set at around 0.2 and peak at above 0.6 for the fifth set. This would seem to indicate that the system is becoming progressively more effective for extracting relevant single-word content out of the PDF, therefore increased alignment of user queries and document content as evaluation progresses.

Fig 4 also shows that the scores for ROUGE-2 are lower for all sets than the scores of ROUGE-1, and the maximum value is still below 0.4. This is expected as capturing bigram overlap is harder than that of unigram-based overlap. But with an increase in performance between the third set to the fifth by ROUGE-2, it shows there is improvement on managing contextual relationships, which is good for the chatbot in terms of handling more complex user queries as it progresses.

Cosine similarity is measured between two embeddings of text and its range lies between -1 to 1, which means it is equal to 1 for perfect similarity. As with cosine similarity, as depicted in fig 4, this is between 0.6 and 0.7 for all of the evaluation sets. It is much less sharp than ST, at least with regard to one of the axes, yet this does allow consistency of performance over all of the sets. This could provide stable understanding of the meaning of a sentence in the USE model but not necessarily capturing anything like as much detail as some of the other models. The ST model performed better than USE on all its tested datasets, achieving cosine similarities over or equal to 0.8 in both cases. For the most part, outperformance of ST means that it encapsulates more semantics in sentence representations and aligns more with actual user queries. One likely explanation for the outperformance is that the use of transformers could enable it to better understand complex queries and give more accurate answers based on the PDF content. That is the system reliance on ST embeddings and increasing ROUGE scores indicate that it is well-poised for dealing with lengthy, contextual queries-an important feature for any document-based assistant.

VII. CONCLUSION AND FUTURE WORKS

The PDF-based chatbot system presented here performs promising extraction at retrieving relevant information from documents, leveraging the advanced language models and embedding techniques applied. It shows rather consistent performance for different query sets, but outstanding well for sentence-level extraction with Sentence Transformer embeddings. The large number of captured unigrams and bigrams by the ROUGE metrics shows that the system progresses from low-recall early sets to higher-quality later sets in which it captures more complex queries. It also offers stable performance in the Universal Sentence Encoder (USE) while enjoying superior performance in ST embeddings, allowing for the use of chatbots as very reliable tools for navigation and response in document-based queries.

Future update of this system should consider these enhancements. One enhancement is about how contextual relationships of words and sentences are handled. Handling relationship dynamics turned to become weak, especially during relatively low ROUGE-2 scores. As for enhancing the performance of the chatbot, by incorporating OCR error correction for better noise reduction could enhance accuracy and performance. Also, using OpenAI embeddings instead of huggingface would significantly improve tokenization and accuracy, however, it would be expensive and denounce the cost-free characteristic of the chatbot. The last step in expanding the system is expansion of the range of supported file types and the making of the system multi-linguistic would make it more broadly applicable for needs in various professional and academic sectors more intuitive and accessible across various domains.

REFERENCE

[1] N. Kandpal, H. Deng, A. Roberts, E. Wallace, and C. Raffel, "Large Language Models Struggle to Learn Long-Tail Knowledge," arXiv:2211.08411v2 [cs.CL], Jul. 2023.

[2] M. Abdin and S. A. Jacobs, "Phi-3 Technical Report: A Highly Capable Language Model Locally on Your Phone," 2024, arXiv:2404.14219v1.

[3] Y. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, Y. Dai, J. Sun, M. Wang, and H. Wang, "Retrieval-Augmented Generation for Large Language Models: A Survey," 2024, arXiv:2312.10997v5.

[4] Y. Zhang, Y. Li, L. Cui, D. Cai, L. Liu, T. Fu, X. Huang, E. Zhao, Y. Zhang, Y. Chen, L. Wang, A. T. Luu, W. Bi, F. Shi, and S. Shi, "Siren's Song in the AI Ocean: A Survey on Hallucination in Large Language Models," 2023, arXiv:2309.01219v2.

[5] D. Banerjee, P. Singh, A. Avadhanam, and S. Srivastava, "Benchmarking LLM powered Chatbots: Methods and Metrics," 2023, arXiv:2308.04624v1.

[6] D. Danopoulos, C. Kachris, and D. Soudris, "Approximate Similarity Search with FAISS Framework Using FPGAs on the Cloud," in Proc. Int. Conf. Embedded Computer Systems: Architectures, Modeling, and Simulation, Aug. 2019, pp. 373–386. DOI: 10.1007/978-3-030-27562-4_27.

[7] X. Han and L. Wang, "A Novel Document-Level Relation Extraction Method Based on BERT and Entity Information," IEEE Access, vol. 8, pp. 96912 - 96919, May. 2020, DOI: 10.1109/ACCESS.2020.2996642.

[8] S. Liu, W. Lyu, X. Ma, and J. Ge, "An Entity-Relation Joint Extraction Method Based on Two Independent Sub-Modules From Unstructured Text," IEEE Access, vol. 11, pp.122154-122163, Oct. 2023, DOI: 10.1109/ACCESS.2023.3328802.

[9] T. Medeiros, M. Medeiros, M. Azevedo, M. Silva, I. Silva, and D. G. Costa, "Analysis of Language-Model-Powered Chatbots for Query Resolution in PDF-Based Automotive Manuals," Vehicles, vol. 5, pp. 1384–1399, Oct. 2023, DOI: 10.3390/vehicles5040076.

[10] B. Yuan and L. Xu, "DoreBer: Document-Level Relation Extraction Method Based on BernNet," IEEE Access, vol. 11, pp. 136468 - 136477, Nov. 2023, DOI: 10.1109/ACCESS.2023.3337871.

[11] D. Christou and G. Tsoumakas, "Improving Distantly-Supervised Relation Extraction Through BERT-Based Label and Instance Embeddings," IEEE Access, vol. 9, pp. 62574 - 62582, Apr. 2021, DOI: 10.1109/ACCESS.2021.3073428.

[12] S.-K. Liu, R.-L. Xu, B.-Y. Geng, Q. Sun, L. Duan, and Y.-M. Liu, "Metaknowledge Extraction Based on Multi-Modal Documents," IEEE Access, vol. 9, pp. 50050 - 50060, Mar. 2021, DOI: 10.1109/ACCESS.2021.3068728.

[13] X. Hu, Z. Hong, C. Zhang, A. Liu, S. Meng, L. Wen, I. King, and P. S. Yu, "Reading Broadly to Open Your Mind: Improving Open Relation Extraction With Search Documents Under Self-Supervisions," IEEE Trans. Knowl. Data Eng., vol. 36, no. 5, pp. 987–999, May 2024.

[14] Y. Sun, H. Qiu, Y. Zheng, Z. Wang, and C. Zhang, "SIFRank: A New Baseline for Unsupervised Keyphrase Extraction Based on Pre-Trained Language Model," IEEE Access, vol. 8, pp. 10896 -10906, Jan. 2020, DOI: 10.1109/ACCESS.2020.2965087.