Proceedings of the International Conference on Electronics and Renewable Systems (ICEARS 2025)
IEEE Xplore Part Number: CFP25AV8-ART; ISBN: 979-8-3315-0967-5

DocGenie: A Retrieval-Augmented Generation Chatbot for Interactive PDF Exploration

Palomi Gawli
Department of Artificial Intelligence and Data Science
Vishwakarma Institute of Technology
Pune, India
palomi.gawli@vit.edu

Gauri Kulkarni
Department of Management
Vishwakarma University
Pune, India
gauri.kulkarni@vupune.ac.in

Swapnil More
Department of Artificial Intelligence and Data Science
Vishwakarma Institute of Technology
Pune, India
swapnil.more22@vit.edu

Mitali Mukkawar
Department of Artificial Intelligence and Data Science
Vishwakarma Institute of Technology
Pune, India
mitali.mukkawar22@vit.edu

Om Munde
Department of Artificial Intelligence and Data Science
Vishwakarma Institute of Technology
Pune, India
om.munde22@vit.edu

Saloni Nachankar
Department of Artificial Intelligence and Data Science
Vishwakarma Institute of Technology
Pune, India
saloni.nachankar22@vit.edu

2025 International Conference on Electronics and Renewable Systems (ICEARS) | 979-8-3315-0967-5/25/$31.00 ©2025 IEEE | DOI: 10.1109/ICEARS64219.2025.10940621

***Abstract* — In order to overcome the drawbacks of conventional PDF readers, this article presents DocGenie, a cutting-edge tool that allows for dynamic and interactive document study. DocGenie improves user engagement and accessibility in document-intensive workflows by enabling effective querying, summarization, and contextual interaction through the use of Retrieval-Augmented Generation (RAG) and Google Gemini Pro. The accuracy, efficiency, and user satisfaction of DocGenie surpass those of traditional tools because to the use of cutting-edge techniques including document chunking, FAISS-based embedding storage, and real-time natural language processing. The platform is a major breakthrough in AI-driven document interaction because of its distinctive characteristics, which include multi-document querying, adaptive summarization, and multilingual support. This paper also identifies opportunities for improvement, such as support for a variety of file formats and improved multilingual capabilities, while highlighting DocGenie's potential uses in fields including education, legal research, and digital libraries.**

Keywords— Retrieval-Augmented Generation, PDF Interaction, AI Chatbot.

I. INTRODUCTION

Nowadays, libraries have transformed from physical locations to online hubs that give patrons access to a vast array of online materials in the digital age. The Portable Document Format (PDF), one of the most widely used formats for digital documents, maintains the structure and layout of documents, making it perfect for reports, academic papers, and other kinds of information. The limited interaction of PDFs in library systems is a major problem, though. User irritation and disengagement may result from PDFs' inability to search for specific information, annotate text, or browse the content with ease, in contrast to web pages or other digital forms.

DocGenie offers a solution to this problem by allowing users to communicate with PDF documents using an easy-to-use chat interface. DocGenie, which is powered by the Google Gemini API, improves the user experience by enabling users to pose queries, look up specific information, and get relevant answers. The system offers more dynamic and customized interactions with digital resources by utilizing cutting-edge strategies including document chunking, adaptive summarization, and real-time natural language processing. Because of this, DocGenie is a useful tool for updating library systems, improving user interaction, and expanding the availability of digital content. DocGenie's multi-document querying and support for multiple languages makes it easier to use PDF documents in library systems and has added an interactive interface to the usage of academic papers, research reports and other documents. With the help of DocGenie integration, libraries will be able to deliver a better experience to the users while assuring availability and usability of digital resources[7].

II. LITERATURE REVIEW

Over the years, there has been a mushrooming of research into challenges in document interaction, question answering (QA), and information retrieval, especially in the domain of PDF documents. These studies have shaped the progress toward developing tools like DocGenie, using advanced natural language processing techniques to improve user experience in accessing PDFs.

A. Over informative Question Answering by Humans and Machines (2023)

This paper delves into methods for question answering (QA) over PDF documents, specifically focusing on techniques that enable systems to extract information and respond to user queries effectively. The authors explore the use of text extraction, natural language processing (NLP), and

document understanding techniques to build robust QA systems. These insights contribute to the design of DocGenie, where similar NLP methods are leveraged to enable users to ask questions and receive precise answers from multi-document PDFs, enhancing document interactivity[4].

B. Information Retrieval Meets Large Language Models (2023)

The authors probe the capability of LLMs to improve upon information retrieval tasks. In this research, they show the possibility of adapting fine-tuned transformer-based models toward existing information retrieval frameworks for efficiency in effectiveness. DocGenie applies an analogue principle by engaging the Google Gemini API-LLM to enhance the features of document search and retrieval. Utilizing LLMs such as Gemini, DocGenie can give context-aware responses and provide summary and query functionalities across multiple documents, helping raise the bar for traditional document search[9].

C. Conversational Agents: Theory and Applications (2022)

It is noteworthy in this paper because it has directed attention to conversational agents, especially chatbots or virtual assistants, which are designed to help users in document understanding tasks. This puts forth the integration of ideas of natural language understanding, dialogue management, and document summarization into conversational interfaces. Building from such thoughts, DocGenie uses a chatbot-based interface that allows users to interact with documents conversationally. This approach not only enhances the ease with which information can be accessed but also encourages user engagement and productivity, particularly in academic and research settings[11].

D. Vector Space Model: An Information Retrieval System (2022)

Vector Space Model of IR This is a fundamental framework in IR where both documents and queries can be mapped in a high-dimensional space as vectors. The model is further useful for calculating the similarity among documents and the corresponding user queries. DocGenie uses the same vector-space-based techniques for storage of document chunking and provides better matching of relevant sections of documents with user queries using FAISS. The VSM principles help ensure effective and efficient information retrieval within large-scale PDF collections[2].

E. Creating Large Language Model Applications Utilizing LangChain: Primer on Developing LLM Apps Fast (2023)

This paper introduces LangChain, a framework for fast development of LLM applications. Best practices are mentioned for leveraging LLMs to build robust applications, along with practical examples of its usage. DocGenie takes inspiration from LangChain, where it integrates Google Gemini Pro to dynamically power interactions with documents. The paper highlights that very efficient integration of the LLMs with document-based applications is crucial to the success of DocGenie, enabling features like adaptive summarization and real-time query responses[3].

III. DESIGN AND METHODOLOGY

- **System Architecture overview** :

The components of Proposed System are as follows:

- **User Interface :**

The web-based interface allows users to easily upload PDF documents and submit queries. Users interact through a conversational interface with the chatbot for retrieving information from the uploaded PDFs. It is designed such that users can upload files, submit queries, and explore document content with minimal hassle[12].

PDF Processing Module :

When files are uploaded, PyPDF2 is applied to the documents to extract text, which is then preprocessed to remove all extraneous characters or formatting so the text can be read cleanly and in a structured form. It is then divided into sentences or paragraphs for easier processing[10].

Text Embedding Module :

The system utilizes the Google Gemini Pro API through LangChain to create high dimensional embeddings for every text chunk; such embeddings represent the semantic context of the text and thereby allow similarity calculations to be precise and retrieve information efficiently[6].

Indexing and Retrieval Module :

In this way, the FAISS library indexes those embeddings to create a searchable database for chunks of text. Upon receiving queries from a user, it generates embeddings for the queries and uses FAISS to retrieve the best matching chunks by computing similarity scores. Advanced indexing techniques are used in order to optimize retrieval speed and resource usage.

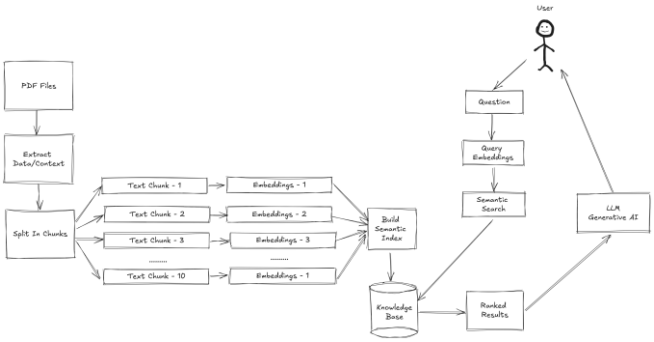

Figure 1: Proposed system architecture

Conversational AI Module :

The system combines Google Gemini Pro with conversational AI that can understand natural language inputs from the user. The chatbot analyzes user inputs, searches for the most relevant information in indexed content, and provides brief and accurate responses. Additionally, the system offers iterative query refinement and contextually suitable suggestions[8].

Real-Time Updates and Feedback Mechanism :

Real time updating keeps the user updated about the system's current processing and retrieval results. It presents a very dynamic user interface experience. A feedback system can be in place to rate the response's relevance and accuracy. Such ratings help refine the indexing and retrieval algorithms to improve system performance over time.

- **User Workflow:**

Step 1: Upload PDF Files : Users upload one or multiple PDF files via the web-based interface.

Step 2: Text Extraction and Embedding : The system extracts text from uploaded files and generates embeddings for individual text chunks.

Step 3: Indexing and Retrieval : The text embeddings are indexed using FAISS, enabling efficient retrieval. When users submit queries, the system retrieves the most relevant text chunks based on similarity calculations.

Step 4: Conversational Interaction : The chatbot engages users in a conversational manner, providing answers and suggestions based on retrieved information from the PDFs.

Step 5: Display Results : The chatbot presents retrieved information through the interface and facilitates further interactions for query refinement or document exploration.

- **Additional Features :**

- **Real-Time Updates** : The system ensures a seamless and engaging user experience by providing dynamic, real-time updates throughout the information retrieval process. As the user uploads PDF files and submits queries, the system visually indicates the progress of text extraction, embedding generation, and content retrieval. These updates keep the user informed at every stage, reducing uncertainty and enhancing overall interactivity.

- **Comprehensive Document Summarization** : The system includes a summarization feature that highlights the most important information within the uploaded PDFs. By condensing large volumes of text into concise summaries, this feature helps users quickly grasp key insights and locate relevant sections of the document. This is particularly useful for navigating lengthy academic papers, research reports, or technical manuals, saving time and improving accessibility to crucial content.

- **Gaps between Existing System and Proposed system :**

1. Functionality :

Existing : Handles general institute-related inquiries.

Proposed : Facilitates queries based on uploaded PDF content.

2.Data Input :

Existing : Takes user queries directly.

Proposed : Incorporates PDF uploads as an additional data source.

3. Processing :

Existing : Processes predefined knowledge-based queries.

Proposed : Extracts and processes text from PDFs to handle document-based queries.

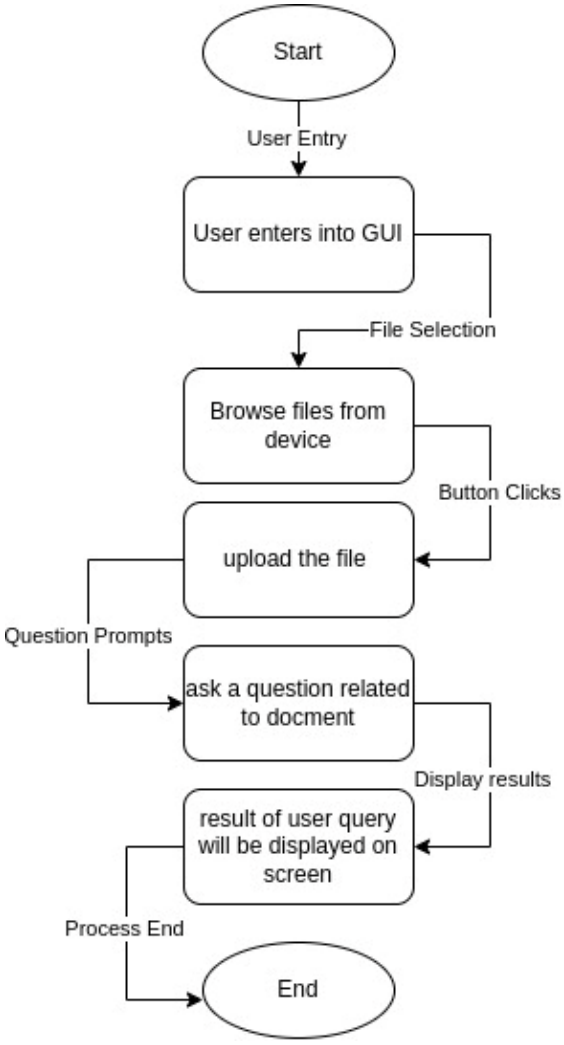

Figure 2: User Workflow

4. Interaction Flow :

Existing : Handles general institute-related inquiries.
Proposed : Facilitates queries based on uploaded PDF content.

IV. Results and Discussion

Approaches and technologies have been developed to address the challenge of extracting information from textual documents, including PDFs, in the realms of information retrieval and natural language processing. The approaches range in complexity from manual methods to sophisticated automated techniques. We will review relevant work and look into the technologies, algorithms, and methodologies used in this section.

1. PDF Parsing Technologies : This project critically parses PDFs and extracts their text content. Tools such as PyPDF2 enable automated processing of PDF files. PyPDF2 allows for the efficient extraction of text and forms part of the core library used for handling PDFs in this system.

- Importing the Library: PyPDF2 is initiated at the beginning of the code.

- Extracting text from PDF files: This function, extract_pdf_text, reads in a user-uploaded PDF, and processes each page to extract text using PyPDF2.

-Text Chunking: Lang chain's Recursive CharacterTextSplitter takes the raw text and breaks it down into palatable chunks. This function accepts defined parameters, including chunk size for the maximum length of a segment and chunk overlap to define overlap between segments. This balances the chunk size for ideal analysis and maintains context across boundaries.

2. Text Processing and Embeddings: To make the chunks easier to understand and retrieve, this framework utilizes the Langchain processing of the text chunks with Google Generative AI Embeddings. The embeddings are a numeric representation of their semantic meaning, ensuring an effective similarity search and contextual understanding.

For instance, consider the following text chunk: "A rising tide lifts all boats, signifying mutual benefits from shared growth."

- Tokenization: The chunk is tokenized into words or subwords -

Tokens: ["A", "rising", "tide", "lifts", "all", "boats", ",", "signifying", "mutual", "benefits", "from", "shared", "growth", "."]

- Embedding Lookup: Each token is mapped to its vector embedding using a pre-trained matrix from Langchain.

Example token embeddings (3D representation for simplicity):
"A": [0.12, 0.34, 0.45]

"rising": [0.56, 0.78, 0.67]

"tide": [0.45, 0.23, 0.54]

(And so on for all other tokens.)

- Aggregation: Individual token embeddings are summed together into a single vector representing the text chunk. For example, using averaging:

Aggregated embedding = (, ,] = [0.42, 0.45, 0.52]

- Normalization: The aggregated vector is normalized to unit length:

Norm = =

Thus , Norm = 0.79

Normalized embedding = [, ,] = [0.53, 0.57, 0.66]

Output : The final normalized embedding vector encapsulates the semantic meaning of the text chunk in a high-dimensional space. In this space, similar meaning text chunks have close embeddings, which could naturally lead to effective information retrieval and similarity calculations[13].

For instance, the normalized embedding vector [0.53, 0.57, 0.66] represents the semantic meaning of the text chunk: "A rising tide lifts all boats, signifying mutual benefits from shared growth.". This 3-dimensional embedding represents the semantics of the text chunk. Such chunks sharing the same semantic meaning like: "Collective progress ensures mutual prosperity," would yield very close distance embeddings to the one given above, and this way the system would be able to look for correct and contextually relevant content. Overall, Text Embedding Module uses Langchain's Google Generative AI Embeddings to convert text blocks into embeddings, capturing the semantic richness of any text and forming a backbone for efficient retrieval within the project framework for handling PDF documents.

3. Similarity Search: FAISS, or Facebook AI Similarity Search, is a fundamental module of this project that allows for fast and accurate retrieval of information from indexed PDF content. As part of the similarity search and clustering tools designed specifically for dense vectors, FAISS is a pretty powerful tool for managing embeddings from text chunking [15].

- Indexing :

Once the chunks of text are extracted from PDF documents and embedded with Langchain's Google Generative AI Embeddings model, FAISS indexes them. Product quantization is used to quantize embeddings to optimize the efficiency of storage and retrieval. This reduces the dimensionality of these embeddings while preserving their representational power. FAISS orders the quantized embeddings into inverted lists according to the centroids, created during the process of quantization. Inside every inverted list, it stores the IDs of text chunks associated with the particular centroid. This structure allows for rapid identification and retrieval of embeddings similar to a given query[5].

- Retrieval :

When a query is submitted, it generates its embeddings in exactly the same way used at index time. In the quantized form, query embeddings are then used to match against inverted lists in the index. FAISS then computes the similarity score between query embeddings and the text chunk embeddings it retrieved. The chunks with the highest similarity scores are presented as the most relevant results[2].

Numerical example using FAISS with simplification for the illustration of the process of indexing and retrieval :

Let's consider three chunks of texts, T1, T2, T3 and present each of them as 2-dimensional vectors. Applying product quantization, the 2-dimensional subspace is divided into two 1-dimensional subspaces and two centroids are created per each subspace, hence four centroids in total.

- Embeddings for text chunks quantized:
  - T1: [0.2, 0.1] → [0, 1]
  - T2: [0.4, 0.3] → [1, 0]
  - T3: [0.6, 0.5] → [1, 1]
- FAISS groups these embeddings into inverted lists:
  - Centroid [0, 1] contains T1
  - Centroid [1, 0] contains T2
  - Centroid [1, 1] contains T3

Retrieval process : A user submits a query embedding [0.25, 0.15]. The embedding is quantized into [0, 1], which corresponds to the centroid for T1. FAISS retrieves T1 from the inverted list and computes a similarity score to determine whether it is relevant. T1 should be given as the best chunk to answer the query.

4. Conversational AI : Conversational AI is a significant enabler of allowing the chatbot to interact with a user in a natural and intuitive manner. This project equips the chatbot with advanced conversational capabilities via Google Generative AI, enabling it to understand user queries and respond precisely based on indexed PDF content. This turns the user experience as they communicate seamlessly through a conversational interface and get information easily retrieved[15].
- Indexed PDF Content : Take two PDF documents D1 and D2 and parse them into a few chunks of texts, say T1, T2, T3. For every single text chunk, generate vector representations with these embeddings indexed using FAISS for efficient similarity searches.

- User Query : A user submits a query, such as "Explain Data Structures," which is converted to an embedding vector-for example, [0.2, 0.3].

- Answer Generation: The FAISS index retrieves relevant text chunks according to the query embedding. Google Generative AI processes the retrieved chunks and query context to generate an informative, coherent response.

- Example Interaction: A user asks the following: "Elucidate machine learning models used in healthcare." The module determined that the most appropriate chunk with regard to the question is T5 extracted from document D3, which has the following text: "Machine learning models include decision trees and neural networks. Such models have become highly applied in healthcare in predictive analytics, patient diagnostics, and treatment plans tailored for different patients." Using this retrieved content, the Conversational AI module produces the response: "Machine learning models like decision trees and neural networks are widely used in healthcare for tasks including predictive analytics, diagnostics, and personalized treatments. For further information on the same, refer to the indexed PDF documents."[5].

5. Vector Store : A critical component, where the vector store is created to manage storage and retrieval of embeddings for chunks of text extracted from PDFs. On uploading a PDF, the backend server location will be accessible, depending on the architecture of the project, so it can be stored on disk. Each PDF uploaded is assigned an identifier, like a filename or ID, and to ensure easy retrieval whenever such could be required. The vector store ensures effective indexing and retrieval of embeddings for effective information processing[14].

6. Interaction with Streamlit : The user interface framework for this project is Streamlit, providing an interactive and user-friendly environment. This allows the end users to upload PDF documents and converse with the chatbot through an intuitive interface. Streamlit streamlines the integration of components such as file uploaders a interfaces and thus forms an excellent choice for developing applications of highest concern in terms of user experience.

- *Implementation* :

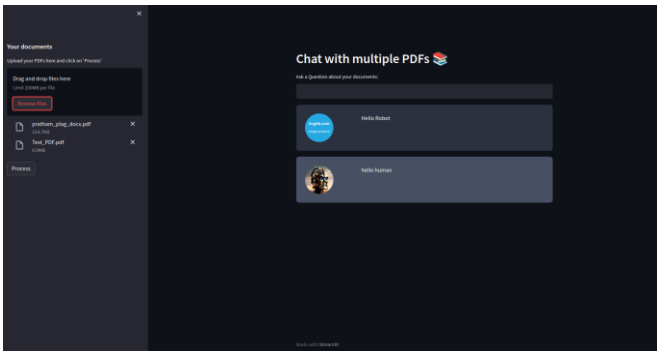

Figure 4: Uploading Multiple pdfs

V. CONCLUSION

It has the effective demonstration of how an interactive chat interface meant for answering questions from uploaded PDFs can be produced. Using Langchain, the application creates a customized question-answering workflow, integrates with an LLM for generating contextually appropriate answers to a variety of user queries. Users can upload PDFs in this system and process them into manageable chunks, embed into vectors of high dimensions, and then index for efficient retrieval. Streamlit is the user interface, so interaction with the chatbot is intuitive, and the entire process of querying PDFs is simple and entertaining.

The core strengths of the project reside in the modular architecture that clearly separates text extraction, embedding generation, similarity search, and the generation of responses. Langchain handles the core task of text processing and embedding generation, and FAISS ensures fast similarity search of high accuracy. Integration of Streamlit helps enhance user interaction even further by allowing for easy uploading and querying of content without any hassle. Although the system has proven quite effective, there are still aspects that have room for improvement in terms of error handling, search optimization, context management, and implementation of security best practices for robust performance in real-world applications[1].

This project therefore opens new possibilities for the enhancement of document interaction by successfully bridging the gap between static document content and dynamic user queries. The idea demonstrates how NLP techniques coupled with processing tools can be used to create engaging as well as an efficient system for information retrieval. As the project emerges, it has the potential to provide more features that will improve accuracy in the understanding of the document, improve the management of context, and offer a more subtle user experience, making it a powerful tool for navigating and extracting insights from complex PDF documents.

REFERENCES

[1] Ritendu Bhattacharyya, Sharat Chandra K.Manikonda, Bharani Kumar Depuru3 "An AIDriven Interactive Chatbot: A Well Trained Chatbot that Communicates with the Users and Reduces the Man- ual Interaction", Vol.9, Feb 2024.

[2] Vaibhav Kant Singh, Vinay Kumar Singh "VECTOR SPACE MODEL: AN INFORMATION RETRIEVAL SYSTEM", July 2022

[3] Topsakal, Oguzhan and Akinci, T. Cetin "Creating Large Language Model Applications Utilizing LangChain: A Primer on Developing LLM Apps Fast", July 2023

[4] Wenhu Chen and Ming-Wei Chang and Eva Schlinger and William Wang and William W. Cohen "Open Question Answering over Tables and Text,2021

[5] Singla, Samriddhi and Eldawy, Ahmed and Diao, Tina and Mukhopadhyay, Ayan and Scudiero, Elia "Experimental Study of Big Raster and Vector Database Systems",April 2021.

[6] Adewumi, T.P., Liwicki, F., Liwicki, M.: Word2vec: Optimal hyper-parameters and their impact on nlp downstream tasks. arXiv preprint arXiv:2003.11645 (2020).

[7] Grave, E., Bojanowski, P., Gupta, P., Joulin, A., Mikolov, T.: Learning word vectors for 157 languages. arXiv preprint arXiv:1802.06893 (2018).

[8] Korat AS (2024) AI-Augmented LangChain: Facilitating natural language SQL queries for Non-Technical users. Journal of Artificial Intelligence & Cloud Computing 3: 1-5.

[9] Wang A, Singh A, Michael J, Hill F, Levy O, et al. (2019) GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. arXiv preprint arXiv:1804.07461.

[10] Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, et al. (2017) Attention is All You Need. Advances in Neural Information Processing Systems 30.

[11] Devlin J, Chang MW, Lee K, Toutanova K (2019) BERT: Pretraining of Deep Bidirectional Transformers for Language Understanding. arXiv preprint arXiv:1810.04805

[12] Kwon ON, Lee N, Shin B (2020) Data-driven cognitive chatbot using transformer and domain knowledge. IEEE Access 8: 45094-45103.

[13] Brown TB, Mann B, Ryder N, Subbiah M, Kaplan J, et al. (2020) Language models are few-shot learners. arXiv preprint arXiv:2005.14165.

[14] Peng B, Li C, Gao J, Chen W, Wong KF, et al. (2020) Reinforced multi-task knowledge base question answering. Proceedings of the 28th International Conference on Computational Linguistics 5653-5665.

[15] Chowdhery A, Narasimhan N, Mishra R, McCallie D (2022) LLM.int8(): 8-bit Matrix Multiplication for Transformers at Scale. arXiv preprint arXiv:2208.07339.