CS6120 Final Project Report: Book Tutor

Utkarsh Gogna, Leonel Senai, Sara Spasojevic
Northeastern University, gogna.u, senai.l, spasojevic.s@northeastern.edu

***Abstract* - This project aims to develop a conversational question-answering (QA) system that enables users to interact with PDF documents through a chatbot. Users will be able to upload a PDF and ask questions about its contents. This tool is designed to streamline information extraction from documents such as research papers, legal documents, and business reports. The system efficiently processes and understands the document by leveraging natural language processing (NLP) techniques allowing users to retrieve precise and contextually relevant answers. The application is not limited to a specific use case/field. It significantly reduces the time required for information retrieval and improves document accessibility for users with varying levels of expertise.**

Index Terms - Machine Learning, Natural Language Processing (NLP), Retrieval Augmented Generation (RAG)

INTRODUCTION

Navigating large PDF documents to find specific information can be a slow and tedious task, especially for researchers, legal professionals, and business analysts. This project aims to make that process easier by developing a system that allows users to simply upload PDFs and ask questions in plain language. The system searches the document for relevant sections using NLP techniques and generates accurate answers using Retrieval-Augmented Generation (RAG) approach, saving time and effort. This approach helps users quickly find the information they need, whether it's for research papers, books, or other documents, improving efficiency and accessibility.

INTRODUCTION

Conversational question-answering (QA) systems have become more advanced especially with transformer-based models like BERT (Devlin et al., 2018) [1], and GPT (Brown et al., 2020) [2]. Retrieval-Augmented Generation (RAG), introduced by Lewis et al. (2020) [3], combines retrieval based on semantic search and these generative techniques to improve answers' precision. This approach narrows the focus to relevant document sections before they are passed to LLMs to use.

Chunks are embedded into dense vectors and put into efficient similarity search frameworks like FAISS (Johnson et al., 2017) [4]. These methods enable fast and accurate retrieval of contextually relevant sections. This project aims to generalize such capabilities for diverse PDF documents, enhancing accessibility and usability across various fields.

METHODOLOGY

I. RAG Pipeline

RAG works by splitting the document into chunks. These are embedded into vector representations and stored in a database. When a user submits a query, it is also embedded, and the most similar chunks are retrieved based on a similarity score. In the second stage, the top n similar chunks are combined to craft a prompt, which is then passed to a large language model (LLM) to generate a contextually accurate response. This two-step process ensures the response is both relevant and grounded in the source material.

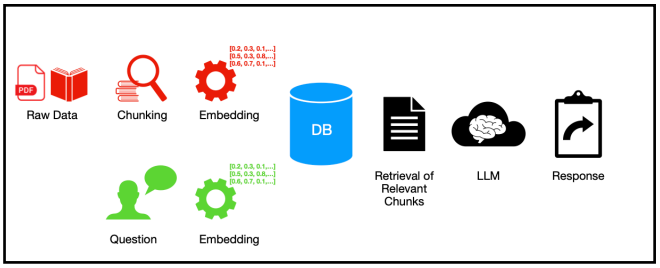

FIGURE I
DIAGRAM OF RAG PIPELINE

II. Data Preprocessing

IIa. Loading the PDF

The first step in the pipeline is loading the PDF document. Using the PyMuPDF library, the PDF is read, and the text is parsed. Each text segment and the page it appears on is saved inside of a dictionary. This mapping of text to a page it came from allows the system to reference specific pages when generating responses. This way, the system provides accurate answers and gives users a page within the document to trace responses back to.

IIb. Chunking

The chunking process divides the document into smaller, manageable segments for embedding and retrieval. We tested four configurations for chunking: splitting by fixed character lengths, by a fixed number of sentences, by paragraphs and by pages. Each chunk undergoes preprocessing to clean the text, normalize spaces, and standardize newlines. The resulting chunks are stored in a DataFrame along with metadata, such as the total number of words, sentences, and characters per chunk, to provide insights into their structure. Selecting the right chunk size is critical—chunks that are too short may lack sufficient

context, while overly long chunks can dilute relevance and increase computational overhead.

IIc. Embedding

During the embedding process, we convert text chunks into dense vector representations that carry the semantic meanings of those chunks. We tested multiple pre-trained models: "all-MiniLM-L6-v2", "all-mpnet-base-v2", and "multi-qa-mpnet-base-cos-v1". Each model offers different trade-offs between accuracy, embedding size, and computational efficiency. After experimentation, the embedding model was chosen based on the correctness of the chunks retrieved to match the query and its ability to effectively capture the context. These embeddings are the most important part of the retrieval step, matching the user queries with the most relevant chunks in the document.

IId. Storing

Once we generate the embeddings, we store them in a vector database. We use FAISS (Facebook AI Similarity Search) which stores and organizes embeddings in a way that supports fast nearest-neighbor lookups. This makes it work very well with large-scale datasets, and helps with fast retrieval.

III. Retrieval

For the second part of the process, we embed the user's query and compare it to the embeddings of the chunks in the database. FAISS supports retrieving those similar chunks using cosine similarity and Euclidean distance. Cosine similarity measures the angle between two vectors in an n-dimensional space, with focus on their orientation and not magnitude. Euclidean distance, however, calculates the actual straight-line distance between these vectors. We chose to use FAISS with Euclidean distance because of its computational efficiency, as chunks retrieved with both cosine and Euclidean distance were fairly similar.

IV. Template and prompt engineering

Crafting a perfect template that works with each model determines the quality of the response. Template is engineered with specific instructions on what the model should/should not do and then populated with the context and query. These carefully crafted prompts can affect the length of the response, complexity of the words used, format of delivery, and they can also prevent the model from hallucinating facts or using its previous knowledge to respond to queries. Not having a specific enough query (not giving the model strong enough guidelines) may leave the model open to vulnerabilities which can further cause some debatable outcomes.

V. Models

The models chosen are as follows: GPT2, Microsoft's Phi-3.5-mini-Instruct, Mistral 8x7B-Instruct (v 0.1), and Qwen2.5 72B-Instruct. Aside from GPT2, the remaining three models are larger models that are tuned for instruction. This pretraining makes them inherently more suitable for RAG and tasks that require structured prompt engineering and generating context-specific responses. Each of these models was selected to evaluate the effectiveness of different architectures, sizes, and tuning strategies in the context of a conversational QA system that utilizes RAG.

Due to the space and computational resources required to run these three instruction-tuned models locally, we utilized Hugging Face's inference API to experiment with them. This allowed us to leverage powerful models like Qwen and Mistral without incurring the overhead to deploy them locally.

Va. GPT2

GPT2 (Generative Pre-Trained Transformer 2) is the smallest model used in the project. It was pre-trained on a dataset of 8 million web pages and has 1.5 billion parameters [5]. Due to GPT2 being originally trained on a large corpus of text scraped from the web, its main focus is predicting the next word in a sequence. Due to this model not being trained on a QA dataset, when passed context chunk and a query as a prompt, GPT2 continued the sequence in style of chunk passed and started "making up" the rest of the text.

In order to explore the capabilities of GPT2 further, we have attempted to train the model on a QA dataset: The Stanford Question Answering Dataset (SQuAD) [6]. We optimized the dataset so it is passed to the model in the format context, question, answer. We also limited the number of examples to 20,000 due to hardware limitations, which took several hours of training. Model was then saved and model was then passed the same queries. It started making connections between the chunks and the queries, answering questions to the best of its abilities, with sentences sometimes not making much grammatical sense, which is why we have decided to move on to more complex models. This successful training however, shows the ability of the model to be further fine-tuned for a particular use-case.

Vb. PHI

Phi-3.5-mini-instruct (Phi-3.5 for short) is the most lightweight of the remaining three models, making it more suitable for resource-constrained use cases, while still being tuned for instruction. The model is a Small Language Model (SLM) with 3.8 billion parameters. Phi-3.5 was trained on 3.4 trillion tokens; the training data includes filtered high-quality documents, synthetic textbook-like data for math, coding, and reasoning, and supervised chat data optimized for instruction-following, truthfulness, and helpfulness [7].

Phi-3.5 supports a vocabulary size of 32,096 tokens and has a context length of 128,000 tokens, allowing it to handle moderately long inputs and outputs. As an instruction-tuned model, its developers recommended using prompts with the following chat format:

```
<|system|>
You are a helpful assistant.<|end|>
<|user|>
How to explain Internet for a medieval knight?<|end|>
<|assistant|>
```

FIGURE II
PHI PROMPT TEMPLATE

Vc. Mistral

Mistral 7B-Instruct is a large language model designed to be a demonstration model by Mistral with 7.3 billion parameters [8]. It supports a context length of 8,192 tokens and has a vocabulary size of 32,768 [9]-[10]. The model was trained on public online data (though further details are not shared). Its instruction-tuned version is optimized for following prompts and delivering context-aware responses. The recommended instruction format is suggested as:

```
<s>[INST] Instruction [/INST] Model answer</s>
[INST] Follow-up instruction[/INST]
```

FIGURE III
MISTRAL PROMPT TEMPLATE

Furthermore, the model can be prompted with different roles such as 'system', 'user', and 'assistant', which is described as a 'few-shot setting', allowing users to steer the model response more directly within a single prompt text block. An example is included below:

```
<|system|>
You are a helpful assistant that reformats dates into the YYYY-MM-DD format.<|end|>
<|user|>
Convert the following date: "April 5th, 2023"<|end|>
<|assistant|>
2023-04-05<|end|>
<|user|>
Convert the following date: "January 15th, 2024"<|end|>
```

FIGURE IV
MISTRAL PROMPT TEMPLATE FOR DIFFERENT ROLES

Vd. QWEN

Qwen2.5 72B-Instruct (Qwen2.5) is a causal language model with 72.7 billion parameters. It was pretrained on up to 18 trillion tokens, which is significantly larger than Phi-3.5's training set (Mistral 8x7B training set size does not appear publicly available). Qwen2.5 has a relatively large vocabulary size of 151,646 tokens, which is significantly larger than the other two instruction-tuned models. The model supports a context length of up to 32,768 tokens by default, but can extend up to 131,072 tokens with specific configurations (YaRN) [11]-[13].

Due to its large parameter size, the model may present comparative scalability challenges compared to the smaller models, especially if the YaRN configuration is enabled and the context length happens to be relatively small.

Like the other instruction-tuned models, Qwen2.5 has a recommended format and an apply_chat_template method for structuring prompts:

```
prompt = "What is the capital of Japan?"
messages = [
        {"role": "system", "content": "You are a polite assistant."},
        {"role": "user", "content": "What is the capital of France?"},
        {"role": "assistant", "content": "The capital of France is Paris."},
]
text = tokenizer.apply_chat_template(
    messages,
    tokenize=False,
    add_generation_prompt=True
)
```

FIGURE V
QWEN PROMPT FORMAT

Below is a table summarizing the key features of each of the three models:

TABLE I
KEY FEATURES OF EACH MODEL

Model	Parameters	Training Tokens	Vocab Size	Context Length
Phi-3.5-mini-Instruct	3.8B	3.4T	32,096	128,000 tokens
Mistral 7B-Instruct	7.3B	Public online data (details unknown)	32,768	8,192 tokens
Qwen2.5 72B-Instruct	72.7B	18T	151,646	32,768 tokens (up to 131,072)

ABLATION STUDIES - EVALUATING PIPELINE CONFIGURATIONS

Due to the lack of numerical metrics to use for evaluating our choices, we have resorted to comparing the chunks to ground truth top picks, as well as manually grading and evaluating the quality of responses. We took into consideration how well the models responded to queries, but also how well they followed direction specifying what to do and what not to do.

I. Chunking methods

Without additional architecture, LLM's experience difficulty processing large documents in their entirety and memorizing important concepts due to token limitations. Hence, it is important to divide it in chunks and retrieve the appropriate piece when needed. We explored chunking the text four ways:

- **Entire page** - while it is better than having an entire document and it preserves context within a single page, it may exceed the model's token limit. Additionally, when a page contains multiple topics (e.g., the start of a new chapter), the model might infer irrelevant connections, leading to inaccurate interpretations.
- **Fixed Number of Characters** - this can lead to unintended disruptions in the text. Thoughts may get cut

off mid-sentence, as well as words, which impacts the flow and prevents us from providing the model with the full meaning.

- **Paragraph-Based Chunking** - this method may suffer from issues such as standalone titles or questions, especially in documents like textbooks or instructional materials. For example, a chunk might consist of just a title or a question, lacking the necessary context for deeper understanding, making it unsuitable for certain tasks like semantic search or question answering.
- **Fixed Number of Sentences** - the most effective method we found. This method resolves the problem of words and thoughts being cut off, and avoids issues like standalone titles and questions. By keeping related sentences together, it ensures that each chunk maintains semantic coherence while staying within token limits for effective processing. The number of sentences we extract at a time is 10. Having more results in coming closer to a full page, while having less results in not having enough context to answer the questions.

II. Embedding Models

This exploration is meant to look into indices of the chunks retrieved by all three embedding models: "all-MiniLM-L6-v2", "all-mpnet-base-v2", "multi-qa-mpnet-base-cos-v1".

- **"all-MiniLM-L6-v2"** - this model has the smallest number of embedding dimensions (384), which resulted in lower accuracy of chunks retrieved when the chunks passed to it were longer.
- **"all-mpnet-base-v2"** and **"multi-qa-mpnet-base-cos-v1"** have 768 embedding dimensions and handled the inputs much better resulting in more precise retrieved chunks.

When embedding chunks of Driver Manual.pdf (a driver's learning manual in NYS) the following table shows an example of 4 questions and how the models performed. This sample is representative of the entire exploration for the 3 models.

TABLE II
EXAMPLE OF EMBEDDING MODELS' PERFORMANCE

	all-MiniLM-L6-v2	all-mpnet-base-v2	multi-qa-mpnet-base-cos-v1
What do I do if my turn lights are not working?	2/3	3/3	2/3
Who has the right of way in an intersection?	1/3	3/3	2/3
Where am I not allowed to park?	3/3	3/3(in order)	3/3
What am I not allowed to do while driving?	1/3	2/3	2/3(in order)

After running this test in a similar way and manually comparing against the ground truth, we have decided to use all-mpnet-base-v2 as the embedding model.

III. Similarity

Cosine similarity measures the angle between two vectors in an n-dimensional space, with focus on their orientation and not magnitude. Euclidean distance, however, calculates the actual straight-line distance between these vectors. We chose to use FAISS with Euclidean distance because of its computational efficiency, as chunks retrieved with both cosine and Euclidean distance were fairly similar.

IV. Templates

Crafting a template that will work with the model is of utmost importance for getting the appropriate answer. There are several factors to consider: total token limit (length of input (prompt) plus the generated output (answer)) which will dictate how much of the context we can give to the model; special beginning and end tokens for each model along with formatting that the model is used to for optimal responses; and instructions passed to the model such as what to do and what not to do.

We started with a template that gave some instructions to the model utilizing role-playing (telling the model that they are a tutor), and explaining what their task is. The template was then populated with the context chunks retrieved and passed into the model. This proved too ambitious for GPT2 (limit of 1024 input tokens), as the context/prompt was too large making the model mis-perform and not generate output. For testing with GPT2, we crafted a template that would use one context and one query. Once trained on a QA dataset as described above, this made the model perform better and give some answers to questions.

Moving to other models that are able to take in larger prompts and were trained on various tasks, we crafted and compared the performance of two templates - one outlining the QA process (in form of context, query, answer with one example where the answer is present in the context and another where it is not) and basic role-play instructions; and the second template with distinct guidelines on what to do, but no QA examples.

Through comparison, template 2 showed better performance and more precise question-answering, even though it contained no QA examples for the model to follow - this shows the significance of model's pre-training on a QA dataset so that, once it's presented with a task, it can successfully perform. Not having the QA examples in the second template resulted in a shorter and more precise input to the model containing only the essentials.

Templates contain the instruction not to answer the query using previous knowledge. Template 1 was given an example in form of context, query, answer with answer being:

"Unfortunately, the context provided does not contain the answer to your inquiry."

Template 2 has explicit instructions:

"Only use the context to answer and do not answer the question if the answer is not in the context. If the context does not contain the answer, say that you cannot deduce the

answer from the context."

Models followed template 2 instructions better across the board and did not respond to the question in such cases. Stripping away the instructions of not to answer when context does not mention the topic resulted in the models using its previous knowledge and training to answer questions not covered by the pdf document. To the question "How do I get a private pilot license?", following is an example of a response gotten from the model (Mistral, with template 1):

> *"To obtain a private pilot license (PPL), follow these steps: 1. Find a flight school (Page [10]): You can start by finding a certified flight instructor (CFI)..."*

FIGURE VI
MODEL RESPONSE TO A QUESTION NOT COVERED BY MATERIAL - WITHOUT INSTRUCTIONS

Response with explicit instructions not the answer the question using previous knowledge yielded the response as demonstrated below (QWEN, with template 2):

> *"I cannot deduce the answer from the context provided. The given pages focus on obtaining a driver's license and endorsements in New York..."*

FIGURE VII
MODEL RESPONSE TO A QUESTION NOT COVERED BY MATERIAL - WITH INSTRUCTIONS

Stripping the special beginning and end tokens for each model resulted in models mis-performing (repeating half of the question, continuing to hallucinate and generate new context in style of template passed, repeat words or whole sentences over and over etc.). One such example is:

> *"? ate pilot license? To get a private pilot license pilot license ? license..."*

FIGURE VIII
MODEL RESPONSE WITH PROMPT NOT PADDED WITH TOKENS

The above process showed the strength of template 2, producing optimal results.

V. Models

With the exception of GPT2, the models shared the same set of parameters when using the inference API. Each of which are described below:

- **max_new_tokens**: this specifies the maximum number of tokens the model can generate in response to the input. Though our prompts requested brevity, we chose 500 tokens as a balance between computational cost and the ability to produce sufficiently detailed responses, especially when prompts required more elaboration.
- **temperature**: temperature is the most important parameter regarding the quality of responses, especially for a QA system. Temperature controls the randomness or determinism of the model's response when generating text. Inside the models, this parameter relies on the SoftMax function, which takes a vector of numbers corresponding to the logits of what word to pick next, and converts them to a probability distribution. A lower temperature value makes the SoftMax distribution sharper, meaning the model's output is more deterministic. Thus it will strongly favor words with the highest probabilities. A higher temperature flattens the distribution, which makes less likely next words (or tokens) more probable - introducing randomness, and therefore more creativity. For the purpose of a QA system that is answering targeted questions about a given context, it is important to keep temperature low, so as to ensure a proper, contextually relevant conversation. We decided on a temperature level of 0.2.
- **return_full_text**: this parameter simply determines whether the response includes the input prompt. For testing and experimentation it was helpful to set this parameter to 'True' because we discovered that some models may rephrase the input prompt or query slightly. This finding allowed us to accordingly adjust our prompting strategy to instruct the models to not change the user's queries at all. For end users, and to align with the chatbot use-case, the parameter was set to 'False'.

VI. PDF Length

We tested our Book Tutor against documents of sizes 5 to 80 pages. There was no increase or decrease in quality of responses with change of document size. The only thing the document size affected is the time it took to embed the chunks of the document.

VI. Querying

All three models were tested on queries covering a range of cases all shown in the figure below. The model that performed consistently well over all edge cases was QWEN paired with template 2.

TABLE III
EXAMPLE OF MODELS COVERING A RANGE OF QUERY CASES

	Best Performing Model (in order)	Best template
Answer present in the context	QWEN, Mistral	Template 2
Answer not present in the context	QWEN	Template 2
Answer requires bulletpoints (itemization & procedural explanations)	QWEN, Mistral (only with template 2)	Template 2
Query not detailed enough/ synonyms used	Mistral, QWEN	Template 2
Hypothetical/what-if questions about the context	QWEN (does not hypothesize as design choice)	Template 2
Asking for opinion on the topic	QWEN	Template 1
Trick questions (not mentioned or false statement)	QWEN	Template 2
Inference from the context	QWEN	Template 2

RESULTS

Realizing the goal of the project, we have successfully developed a QA chatbot capable of answering questions from the uploaded PDF documents with high accuracy. After experimenting with multiple models and configurations, we found that the combination of all-mpnet-base-v2 and Qwen2.5 72B-Instruct gave best responses with most relevant chunks, working well in all edge cases. Engineering the prompt template was critical to the success of the system since it helped the model extract answers from meaningful chunks retrieved using similarity search, and guided the model to respond within the context of the document and not using its pre-training knowledge. Similarly, choosing the chunk size for dividing the document was essential: chunks that were too short did not carry enough context, while chunks that were too big resulted in hallucinations and answers containing irrelevant information. The combination of these strategies ensured the system's ability to deliver reliable, context-aware responses across a variety of documents.

WINNING CONFIGURATION

After conducting ablation studies, the winning configuration is:

- **Chunking method**: sentences (n=10)
- **Embedding model**: all-mpnet-base-v2
- **Similarity style**: FAISS (using Euclidean distance)
- **Template**: template 2
- **Model**: QWEN ("max_new_tokens": 500, "temperature": 0.2, "return_full_text": False)

CONCLUSION

The project successfully explored and demonstrated the capability of RAG systems in creating a conversational QA chatbot that surpasses keyword-based search. By combining chunking, embedding, and retrieval techniques with advanced language models, we created a chatbot capable of extracting accurate, contextually relevant answers directly from uploaded PDF documents. This approach improves efficiency by avoiding the need to ingest entire PDFs and reduces the likelihood of model hallucinations by constraining responses to the provided context.

The original inspiration for the project was to create a tutor-like chatbot that can answer queries about a document and cite exactly where in the document its responses came from. This approach highlights the potential for NLP in improving efficiency and accessibility of information retrieval tasks, making it a valuable tool for researchers, students, and professionals alike.

Our experimentation allowed us to learn and explore the importance of carefully selecting system components - such as chunking size (sentence, paragraph, etc.), embedding models, and prompt engineering - for optimal performance. Even smaller models like Phi-3.5, with only 5% of Qwen2.5's parameters, proved effective for QA tasks when coupled with a focused RAG approach, demonstrating the versatility and efficiency of locally run SLMs. The winning configuration reliably addressed the challenges of maintaining context relevance, following explicit instructions, and minimizing hallucination, demonstrating the potential for real-world applications in education, legal, and business domains.

FUTURE WORK

Our team identified several areas to enhance and expand the system's capabilities, and delve deeper into understanding and improving RAG workflows for QA chatbot applications:

- **Improved queries / query expansion**: In many ways, the quality of an answer is directly related to the quality of a query. While we cannot control exactly what a user asks from a chatbot, there are avenues to explore how to improve the quality of final queries being passed to the chatbot. This could include consideration of synonymous words/phrases, or employ methods of query expansion. The models could be used to better capture user intent, and thus improve retrieval quality/ accuracy.
- **Full-text summarization capabilities**: While the project primarily focused on RAG, thus breaking PDF into relevant chunks based on retrieved embeddings, a more complete chatbot product would have summarization capabilities to provide context overview of entire documents. We would propose exploring models such as Longformer or BigBird for this feature, as these models are optimized for processing entire documents of thousands of tokens.
- **Expanded chatbot features**: In keeping with the spirit of the original 'tutor' concept, we would like to explore introducing new features such as generating multiple-choice questions for educational purposes, to test areas

such as reading comprehension. We also discussed implementing a persistent conversation feature to store response and chat history, enabling follow-up queries and enhanced user-experience.

REFERENCES

[1] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," arXiv preprint arXiv:1810.04805, 2018.

[2] A. Radford, J. Wu, D. Amodei, et al., "Language Models are Unsupervised Multitask Learners," Proceedings of the 2019 OpenAI Conference, 2019.

[3] P. Lewis, E. Perez, A. Piktus, et al., "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," arXiv preprint arXiv:2005.11401, 2020.

[4] M. Douze, A. Guzhva, C. Deng, et al., "Faiss: A Library for Efficient Similarity Search and Clustering of Dense Vectors," arXiv preprint arXiv:1702.08734, 2017.

[5] A. Radford, J. Wu, D. Amodei, et al., "Language Models are Unsupervised Multitask Learners," Proceedings of the 2019 OpenAI Conference, 2019.

[6] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, "SQuAD: 100,000+ Questions for Machine Comprehension of Text," Stanford Question Answering Dataset (SQuAD), 2016. [Online]. Available: https://rajpurkar.github.io/SQuAD-explorer/

[7] Hugging Face, "Phi-3.5-mini-instruct," GitHub repository, [Online]. Available: https://huggingface.co/microsoft/Phi-3.5-mini-instruct/blame/8ddd775b1e1405e0585cfe7cf242e79132df5144/README.md. [Accessed: Dec. 8, 2024].

[8] A. Q. Jiang, A. Sablayrolles, A. Mensch, et al., "Mistral 7B," arXiv preprint arXiv:2310.06825v1, 2023.

[9] Mistralai, "Mistral 7B v0.3," Hugging Face repository, [Online]. Available: https://huggingface.co/mistralai/Mistral-7B-v0.3. [Accessed: Dec. 8, 2024].

[10] Mistralai, "Mistral 7B Instruct v0.1," Hugging Face repository, [Online]. Available: https://huggingface.co/mistralai/Mistral-7B-Instruct-v0.1/blob/main/README.md. [Accessed: Dec. 8, 2024].

[11] Qwen Team, "Qwen2.5: A Party of Foundation Models!" Qwen Blog, Sep. 19, 2024. [Online]. Available: https://qwenlm.github.io/blog/qwen2.5/. [Accessed: Dec. 8, 2024].

[12] Qwen, "Qwen2.5-72B-Instruct," Hugging Face repository, [Online]. Available: https://huggingface.co/Qwen/Qwen2.5-72B-Instruct. [Accessed: Dec. 8, 2024].

[13] Qwen Team, "Key Concepts," Qwen Documentation, [Online]. Available: https://qwen.readthedocs.io/en/latest/getting_started/concepts.html. [Accessed: Dec. 8, 2024].