German Mastery with AI Chatbot and Roman Urdu: A1-A2 Proficiency

Project Team

Jawad Ahmed 20P-0165
Muhammad Hassan Shah 20P-0025

Session 2020-2024

Supervised by

Shahzeb Khan

Department of Computer Science

National University of Computer and Emerging Sciences
Peshawar, Pakistan

June, 2024

Student's Declaration

We declare that this project titled "*German Mastery with AI Chatbot and Roman Urdu: A1-A2 Proficiency*", submitted as requirement for the award of degree of Bachelors in Computer Science, does not contain any material previously submitted for a degree in any university; and that to the best of our knowledge, it does not contain any materials previously published or written by another person except where due reference is made in the text.

We understand that the management of Department of Computer Science, National University of Computer and Emerging Sciences, has a zero tolerance policy towards plagiarism. Therefore, We, as authors of the above-mentioned thesis, solemnly declare that no portion of our thesis has been plagiarized and any material used in the thesis from other sources is properly referenced.

We further understand that if we are found guilty of any form of plagiarism in the thesis work even after graduation, the University reserves the right to revoke our BS degree.

Jawad Ahmed Signature: _____

Muhammad Hassan Shah Signature: _____

_____

Verified by Plagiarism Cell Officer
Dated:

Certificate of Approval

The Department of Computer Science, National University of Computer and Emerging Sciences, accepts this thesis titled *German Mastery with AI Chatbot and Roman Urdu: A1-A2 Proficiency*, submitted by Jawad Ahmed (20P-0165), and Muhammad Hassan Shah (20P-0025), in its current form, and it is satisfying the dissertation requirements for the award of Bachelors Degree in Computer Science.

Supervisor

Shahzeb Khan Signature: _____

_____

Haroon Zafar

FYP Coordinator
National University of Computer and Emerging Sciences, Peshawar

_____

Fazle Basit

HoD of Department of Computer Science
National University of Computer and Emerging Sciences

Acknowledgements

We would like to express our gratitude to all those who have contributed to the successful completion of this project.

First we extend our deepest thanks to our advisor, Mr. Shahzeb Khan, for their invaluable guidance, support, and encouragement throughout the project. We also wish to thank FAST for providing the necessary resources and facilities to carry out this project. Special thanks to Mr. Haroon Zafar for their assistance and collaboration.

Thank you all for your contributions and support.

Jawad Ahmed

Muhammad Hassan Shah

Abstract

Learning the German language presents a significant challenge for many individuals, often hindered by barriers such as limited access to resources and ineffective learning materials. In response to this issue, our research project employs an innovative methodology: the development of an AI-powered chatbot designed to offer interactive German language lessons in Roman Urdu, covering proficiency levels A1 and A2.

The core focus of this project is to address the shortcomings of existing language learning approaches by providing an engaging, and highly effective language learning experience. Accessible through a user-friendly web application, this AI chatbot is poised to transform language education.

Our project aims to deliver a solution that bridges the gap in German language education, making it accessible to a broader audience. By leveraging the power of conversational AI and offering Roman Urdu translations tailored to individual needs and pacing, we endeavor to facilitate high-quality German language education at scale.

This innovative approach not only enhances accessibility but also promises to revolutionize the way individuals acquire proficiency in the German language, offering a more effective, adaptable, and user-centered learning experience.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Many people aspire to learn the German language, but traditional methods often pose significant challenges. These methods can be expensive, inflexible, and ineffective. To simplify this process, we are developing an AI-powered chatbot that facilitates German language learning from Roman Urdu. The core purpose of this project is to revolutionize language learning by addressing the limitations of traditional methods and offering an improved approach.

Learning German using traditional methods such as reading books and watching videos can be hard and slow. This poses a significant problem for many learners.

Some universities and companies in Germany require proficiency in the German language as a criterion for scholarships, admissions, or job opportunities. The existing solutions, primarily books and YouTube videos, often teach German through English, which can be challenging for individuals whose first language is not English.

Our thesis and approach involve providing a web application that enables users to learn German from Roman Urdu. The application will feature an AI-powered Retrieval Augmented Generation (RAG) based chatbot that responds to user queries in real-time, enhancing the learning experience.

The success of our application will be determined primarily by how well users can learn from the provided content and how accurately the chatbot replies to user queries.

1.1 Target Audience

The target audience for this German language learning web application comprises individuals seeking to build German language proficiency for academic or professional reasons.

More specifically, key target user groups include:

University students aiming to pursue higher education opportunities in Germany through scholarships that require German language skills. Professionals looking to relocate to Germany for job assignments that necessitate German fluency. The app enables convenient self-paced learning. Language learners who desire to learn German but lack access to quality localized instruction. The solution makes German language education more broadly accessible. The application covers two levels A1 and A2 levels.

With its customized Roman Urdu-translated content and interactive chatbot features, the application provides an intuitive German learning experience tailored to target users.

1.2 Methodology

To address the challenges in making German language learning accessible from Roman Urdu, we have designed the following methodology:

1.2.1 Dataset Creation

Our web application will offer two primary features to enhance the learning experience. The first feature is Learning Content, where users will be presented with material to learn German. This content will be available in both German and Roman Urdu, making it accessible and easy to understand for Urdu-speaking learners. The second feature is a Chatbot that will assist users by answering their queries as they learn. This interactive tool is designed to provide immediate support and clarification, ensuring a smooth and efficient learning process.

To built these features we need two datasets for both the learning and for the chatbot. So we have created two datasets, The learning content dataset and The chatbot dataset. The following methodology we used to create these datasets.

1.2.1.1 Learning Dataset Creation

The creation of the learning dataset involved several steps. Initially, we gathered English books that teach German through English and selected a suitable book for our purposes. We then scraped the dataset from the chosen book. Following this, we translated the English dataset into Roman Urdu to cater to our target audience. To enhance readability and structure, we converted the translated content into JSON format using pre-trained transformers.

1.2.1.2 Chatbot Dataset

The creation of the chatbot dataset involved a few key steps. We utilized the same translated content and employed pre-trained transformers to generate potential question-and-answer pairs that users might ask. This dataset was then formatted into a CSV file, with questions and answers organized into respective columns for easy access and use by the chatbot.

1.2.2 Chatbot Creation: Enhancing LLMs Knowledge using RAG

For the creation of the chatbot, we employed a technique called Retrieval Augmented Generation. Initially, we converted our chatbot dataset into embeddings, which are numerical representations of our data. This dataset was then stored in a vector database known as FAISS (Facebook AI Similarity Search). When a user poses a question, we use a retriever to fetch similar questions from the vector database. The retrieved context is then passed to a large language model, which generates answers based on the provided context. If the answer is present in our dataset, the chatbot will provide it; otherwise, it will respond with 'Question asked is not related to the text provided.'

1.2.3 Web Application

We have built applications using the MERN stack (MongoDB, Express, React and NodeJS). Each user will log in/signup in to the web application and will be shown the learning content and he can interact with the chatbot. Each user profile will be saved so he can view his

history. We have built the backend in nodejs and then fetched the content one by one for each document on the front end. We have utilized the REST APIs in making connections between the front end (react) and back end (nodejs).

Chapter 2

Review of Literature

This literature review provides a critical overview of recent research on leveraging artificial intelligence, specifically transformer-based models, for developing multilingual conversational agents focused on language learning. The goal is to synthesize key findings and limitations from current studies to identify opportunities and challenges in applying state-of-the-art AI to create an effective German learning chatbot. The reviewed studies encompass relevant areas like machine translation, chatbots for education, multilingual models, and conversational agents. By surveying the existing landscape of models, methods, and datasets, this review aims to contextualize and inform the application of modern natural language processing techniques to facilitate personalized, adaptive German language acquisition through conversational interaction.

2.1 Summaries

1. In 2019 [1], a study by Godwin-Jones looked at chatbots for learning languages. They found that chatbots can be helpful because they offer personalized lessons that match your needs. They also give you feedback right away, helping you fix mistakes. But sometimes, using chatbots can be tricky due to technology, and it might make learning a bit harder because you need to juggle between learning and using the computer.

2. In 2017 [2], an important study by Vaswani and their team introduced something

revolutionary in the world of computers and language: attention mechanisms. These clever mechanisms allowed computers to understand and generate text, making them much smarter when it came to translating languages. They could look at different parts of a sentence and understand how they all fit together, making translations more accurate. This was a big deal because it made text generation, like writing and translating, much more powerful. But, there was a catch. While these computers were great at putting words together, they didn't always fully understand the big picture. Sometimes they'd get things a bit wrong because they focused on small parts of sentences. So, while they were super helpful, they still had some limitations when it came to fully understanding the context.

3. In 2018 [3], Devlin and team came up with a smart computer program called bert. BERT was super good at understanding and classifying text, especially in English. But the problem was that BERT was only good with English and not other languages, which made it less useful for learning different languages like German.

4. In 2020 [4], Devlin and others made BERT speak many languages. This was great because it could help people learn different languages. However, because it had to work with so many languages, it lost a little bit of the special details in each language.

5. In 2022 [5], a computer model called LLaMA could talk really well, especially in English. But it wasn't so great at talking in many other languages. This was a problem for people who wanted to learn other languages using a chatbot.

6. Also in 2022 [6], a big dataset called RefinedWeb helped computer models like Falcon become better at understanding many languages. This was good news. However, even with this dataset, the models still needed some extra training to be their best.

7. In recent years, significant advancements have been made in the field of natural language processing (NLP) and large language models (LLMs), with a particular focus on improving translation and dialogue generation capabilities. In 2023 [7] enhanced encoder-decoder models with retrieval augmentation in their work

"Raven," improving in-context learning for better translation and dialogue generation. Similarly, In 2022 [8] explored few-shot learning techniques using retrieval-augmented models, which are beneficial for developing responsive chatbots with limited data. In 2023 [9] conducted a survey on techniques for maximizing LLM performance, including retrieval-augmented generation (RAG) approaches, which are relevant for chatbot development. In 2023 [10] investigated prompt compression techniques to accelerate the inference of large language models, optimizing chatbot response times in their study "LLMLingua." In 2023 [11] discussed active retrieval mechanisms to enhance RAG models, thereby improving the accuracy of responses generated by chatbots.

8. In 2023 [12] highlighted the challenges large language models face in learning long-tail knowledge, emphasizing the need for more effective strategies to improve the models' understanding of less common information. In 2023 [13] proposed knowledge graph-augmented language models to enhance knowledge-grounded dialogue generation, demonstrating improved dialogue coherence and informativeness. These studies collectively contribute to advancing the capabilities of NLP and LLMs in various applications.

9. In 2023 [14] introduced the "Tree of Clarifications," a method for answering ambiguous questions using retrieval-augmented large language models, which helps in providing more accurate responses. In 2022 [15] presented the "Demonstrate-Search-Predict" framework, combining retrieval and language models for knowledge-intensive NLP tasks, enhancing the overall performance of these systems. In 2023 [16] introduced the "Tree of Clarifications," a method for answering ambiguous questions using retrieval-augmented large language models. This approach helps in providing more accurate and contextually relevant responses by iteratively refining the query and retrieving pertinent information, thereby improving the overall performance and reliability of LLMs in handling ambiguous questions.

10. In 2020, [17] researchers developed the Dense Passage Retrieval (DPR) system for open-domain question answering, leveraging dense text representations to retrieve relevant information effectively. DPR combines neural networks with dense vector

search techniques, significantly improving the accuracy and efficiency of answering questions in open-domain settings.

11. In 2019 [18] introduced the concept of Nearest Neighbor Language Models (NNLMs), which improve generalization by leveraging memorization through nearest neighbor search in the training data. In 2023 [19] presented "Mamba," a linear-time sequence modeling framework utilizing selective state spaces to enhance efficiency and scalability in sequence modeling tasks. In 2019 [20] proposed the "Mask-Predict" approach for parallel decoding in conditional masked language models, significantly accelerating the decoding process while maintaining high generation quality.

12. In 2023 [21] explored model selection and decoding strategies for keyphrase generation using pre-trained sequence-to-sequence models, proposing methods to enhance the accuracy and relevance of generated keyphrases. In 2020 [22] investigated pre-training techniques for multilingual neural machine translation by leveraging alignment information, improving translation quality across multiple languages. In 2019 [23] introduced BART, a denoising sequence-to-sequence pre-training model designed for natural language generation, translation, and comprehension, demonstrating state-of-the-art performance across various NLP tasks.

13. In 2019 [24] introduced RoBERTa, a robustly optimized BERT pretraining approach, which enhances the performance of BERT by employing larger training data and longer training times. In 2019 [25] presented the concept of generative pre-training for improving language understanding, laying the groundwork for models like GPT that leverage unsupervised learning to achieve superior results in various NLP tasks. In 2019 [26] developed fairseq, a fast and extensible toolkit for sequence modeling, facilitating efficient training and deployment of state-of-the-art NLP models.

To wrap up, these smart computer models are enhancing multilingual communication and learning, including German. Continued improvements and training will address existing challenges, making language learning more engaging and effective.

Chapter 3

Project Vision

3.1 Problem Statement

The traditional methods of teaching German lack real-time feedback and interactivity, creating significant barriers to proficiency, particularly for Roman Urdu speakers.

3.2 Business Opportunity

This project holds the potential to provide an accessible and engaging solution for learning German by harnessing the power of AI. It addresses the diverse needs of learners seeking to acquire language skills effectively.

3.3 Objectives

Our primary objective is to develop a web application that is going to offer content to learn german from roman urdu and a chabot that is going to answer to user queries.

3.4 Project Scope

The project's initial focus is on the core functionality of AI-driven German language learning through conversations. Future enhancements may include user profiles, progress tracking, interactive exercises, and seamless integration of custom content.

3.5 Constraints

Our project is subject to several constraints, including access to multilingual datasets, model training durations, and the associated web hosting costs.

3.6 Stakeholders Description

Students require an engaging and effective approach to learning German that adapts to their unique proficiency levels, ensuring that they can learn at their own pace and retain the language skills they acquire.

German universities seek applicants who can demonstrate a high level of German proficiency, as this is crucial for their academic success and integration into the educational environment.

Businesses operating in Germany need employees with strong German language skills to meet communication demands and effectively engage with clients, partners, and colleagues.

Language learning companies are in search of innovative solutions to remain competitive in the evolving market, aiming to offer advanced and efficient tools that enhance language acquisition for their users.

3.6.1 Stakeholders Summary

Students aim to achieve fluency in German quickly and effectively through engaging and interactive learning experiences. **German universities** aim to attract more applicants with the German proficiency needed for success in higher education. **Businesses in Germany** are striving to meet the demand for employees who are fluent in German. Lastly, **Language Learning Companies** are focused on leading the market by incorporating cutting-edge AI innovations into their language learning products.

Chapter 4

Software Requirements Specifications

4.1 List of Features

Our web application offers comprehensive user account creation and management, allowing users to create accounts and manage their profiles effortlessly.

We provide comprehensive lessons covering levels A1 and A2, ensuring users have access to structured and progressive learning materials.

Users can request translations from Roman Urdu to German, enhancing their understanding and learning efficiency.

An integrated chatbot is available to answer user queries, providing real-time assistance and support throughout the learning process.

Users can also track their progress, monitor their advancements, and identify areas for improvement to stay motivated and focused on their language learning journey.

Functional Requirements

User Registration and Profiles

Users can create and manage their accounts, setting their language proficiency and starting level to tailor their learning experience. This feature ensures that users begin at the appropriate level and can update their profiles as they progress.

Learning Modules

The application offers structured lessons from beginner (A1) to proficiency (A2), covering essential areas such as grammar, vocabulary, and writing. These comprehensive modules provide a solid foundation for learning German effectively.

Chatbot

An integrated chatbot is available to answer user queries, providing real-time assistance and support throughout the learning process. This feature enhances the interactive learning experience, ensuring users receive timely help.

Progress Tracking

Users can monitor their learning progress through detailed tracking features. This allows learners to see their advancements, identify areas for improvement, and stay motivated throughout their language-learning journey.

Access to Up-to-Date Content

The application ensures that all learning content remains current and relevant. Regular updates to the material help users stay informed about the latest language usage and trends.

User Authentication and Security

The platform ensures secure user account management and data protection, safeguarding user information and maintaining privacy. Robust security measures are in place to protect against unauthorized access.

Storing User History

The application stores user interactions with the chatbot and tracks user progress, enabling a personalized learning experience. This historical data helps tailor future lessons and responses to better suit individual user needs.

4.2 Quality Attributes

Reliability

The chatbot system should operate with high reliability, ensuring consistent availability to users. It must have robust error handling to provide users with informative feedback when issues occur, maintaining a seamless user experience even in the face of potential disruptions.

Scalability

The system should be designed to accommodate increasing numbers of users, capable of handling concurrent users without significant degradation Quain performance. Load balancing mechanisms will be implemented to distribute user requests across multiple servers, maintaining responsiveness and ensuring smooth operation during high-traffic periods.

Security

User data should be encrypted to protect sensitive information from unauthorized access. Secure user authentication mechanisms will be in place to ensure that only authorized users can access the system, thereby safeguarding user information and maintaining data privacy.

Performance

The chatbot system should offer fast response times, particularly for interactive exercises and chat interactions. This ensures a smooth and efficient user experience, minimizing delays and enhancing the overall functionality of the system.

Usability

The user interface should be intuitive and user-friendly, allowing users to easily navigate and use the system. This enhances user satisfaction and encourages continued use of the application by making it accessible to users of all skill levels.

Maintainability

The system's codebase and infrastructure should be well-documented and modular to facilitate maintenance and updates. Version control systems will be used to track code changes, allowing for easier collaboration among development teams and ensuring that updates and improvements can be implemented smoothly and efficiently.

4.3 Non-Functional Requirements

Functional Requirement	**Non-Functional Requirement**
User account creation and management	User data should be encrypted to ensure data privacy and security.
Lessons and exercises for levels A1 and A2	Data will be correct and updated.
Translation from Roman Urdu to German	The generated translation will be accurate.
Chatbot	The chatbot should have high availability to ensure uninterrupted user access.
Progress tracking and storing user history	The web application will maintain user history and store user interactions with the chatbot.
Customizable study plans	Load balancing mechanisms should distribute user requests evenly across multiple servers to maintain system responsiveness.

Table 4.1: Comparison of Functional and Non-Functional Requirements

4.4 UML Diagrams

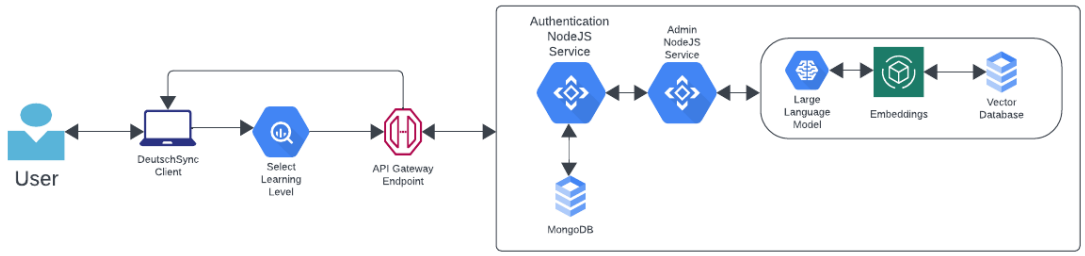

Figure 4.1: Architecture Diagram

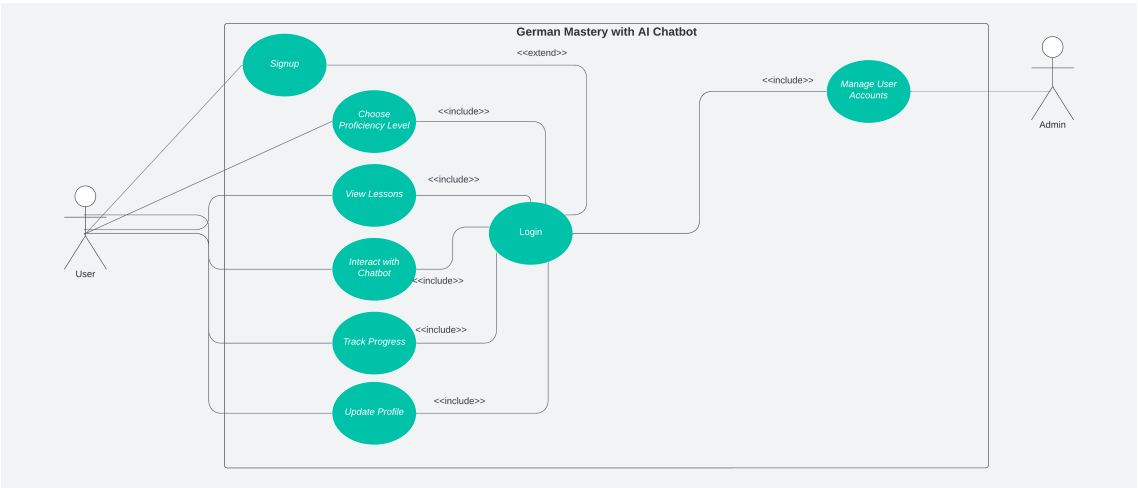

Figure 4.2: Use Case Diagram

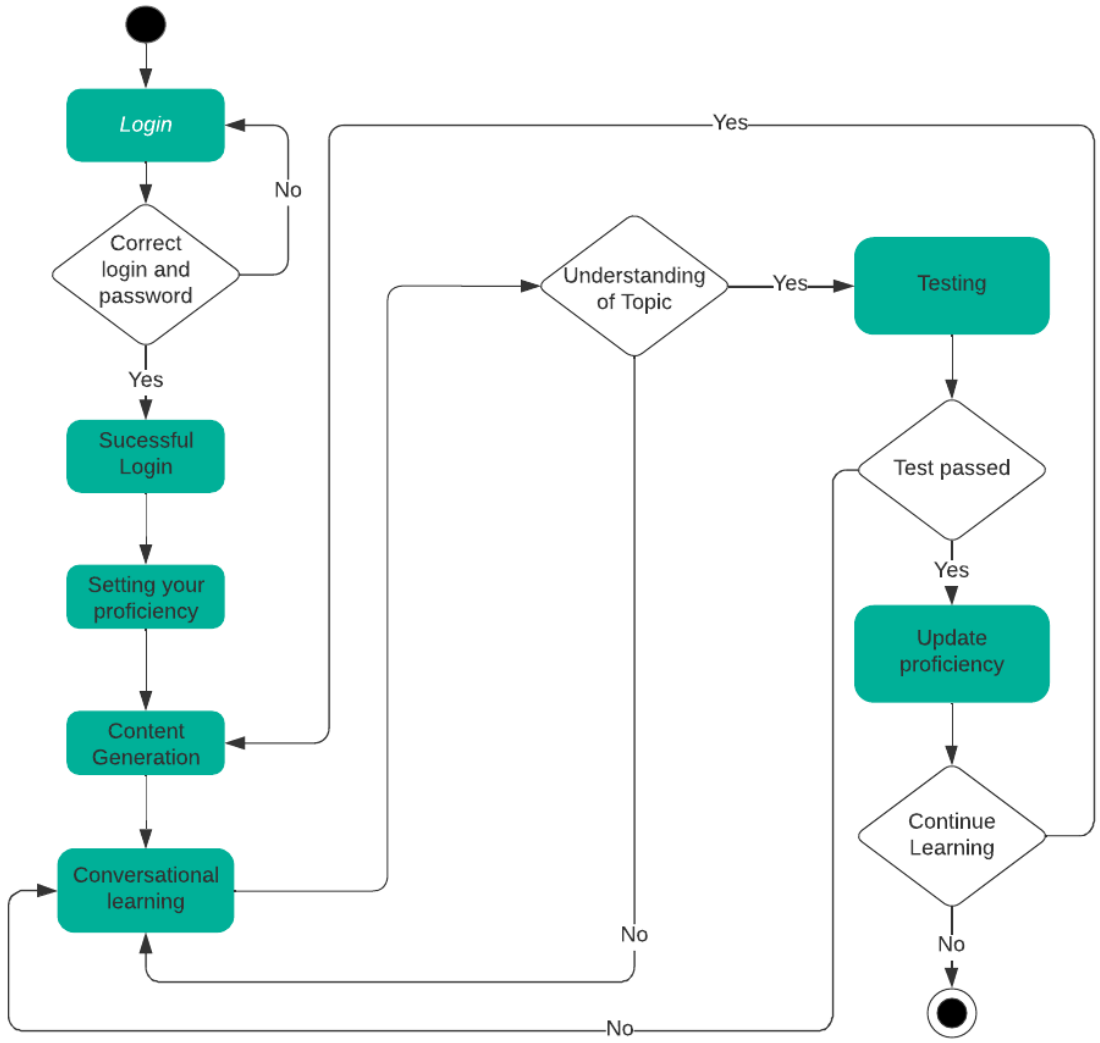

Figure 4.3: Activity Diagram

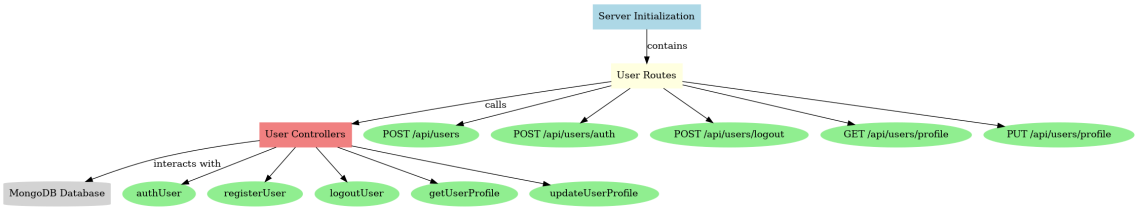

Figure 4.4: Server API diagram

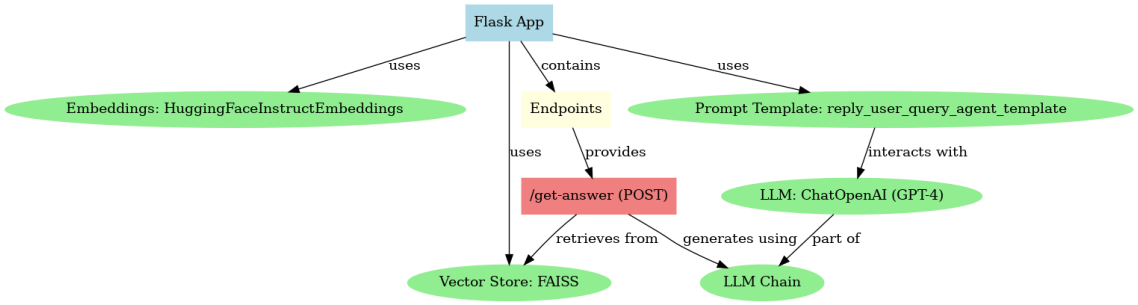

Figure 4.5: RAG Chatbot API diagram

Chapter 5

Iteration Plan

5.1 Midterm FYP 1

During Iteration 1, we conducted comprehensive research aimed at understanding various Large Language Models (LLMs) available in the academic and industry domains, as well as exploring resources for teaching German using Roman Urdu. This phase served as a foundational step in informing subsequent project activities and decisions.

5.2 Final FYP 1

In this iteration, we have successfully created the MERN-based web application. We have created a frontend in react and a backend server in node js and express. We are using the MongoDB database to store user data. We have used REST APIs to create a connection between the backend (server) and the frontend(client).

5.3 Midterm FYP 2

We conducted research to identify relevant books aligning with our use case requirements. Following this, we extracted data from these books and processed it through a Language Model (LLM). The LLM translated the data from English to Roman Urdu and structured it into a suitable JSON format for further analysis and utilization.

5.4 Final FYP 2

In this iteration, we are going to first Improve the dataset. Second, we are going to test our application. Third we are going to add a new feature translation from Roman Urdu to german chatbot and Lastly, we are going to deploy the application on the cloud.

Chapter 6

Iteration 1

6.1 Introduction

During Iteration 1, we researched Large Language Models (LLMs) and explored resources for teaching German using Roman Urdu. This foundational step informed subsequent project activities and decisions.

6.1.1 Research Process

The research process involved thorough exploration of academic journals and conferences, with a focus on understanding the characteristics and functionalities of prominent LLMs. Additionally, we researched available books that teach German using English as there were no resources found for teaching German using Roman Urdu.

6.1.2 Selected LLMs

The following LLMs were identified and studied as part of our research endeavors:

- Attention is All You Need
- BERT (Bidirectional Encoder Representations from Transformers)
- mBERT (Multilingual BERT)
- Falcon

- Llama2

6.1.3 Literature Review

Detailed findings from the research conducted on the aforementioned LLMs have been documented in the Literature Review section of Chapter 2 of this report. This section provides an in-depth analysis of each model and available teaching resources, including their underlying architectures, key features, strengths, weaknesses, and notable research contributions.

6.1.4 Diagrams

6.1.4.1 Use Case Diagram

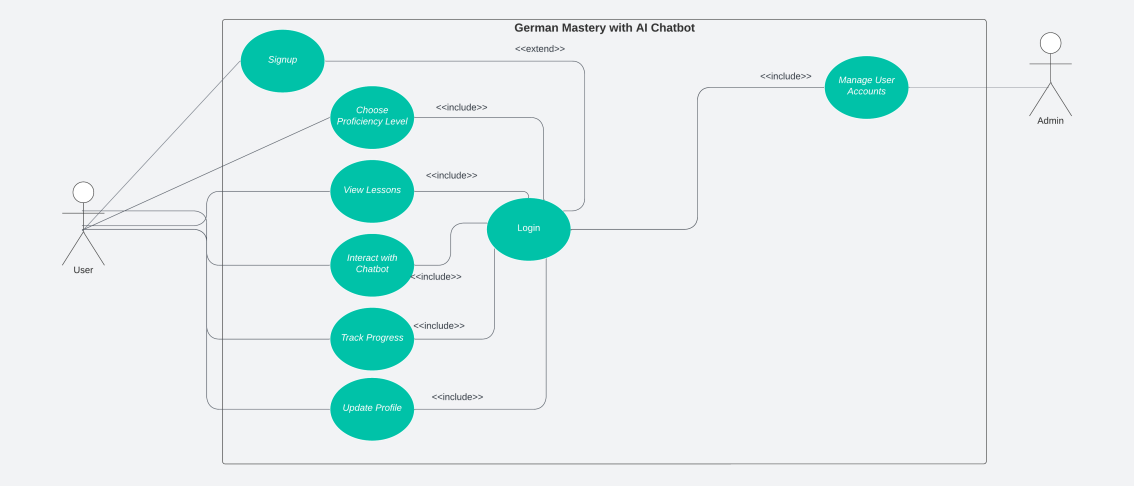

Figure 6.1: Use Case Diagram

The Use Case diagram illustrates user interactions within the system, encompassing account creation, login, proficiency level selection, lesson viewing, progress tracking, chatbot interaction for queries related to educational content, and profile updates. This diagram succinctly captures the essential functionalities available to users, streamlining their educational journey and enhancing overall user experience.

6.1.4.2 Activity Diagram

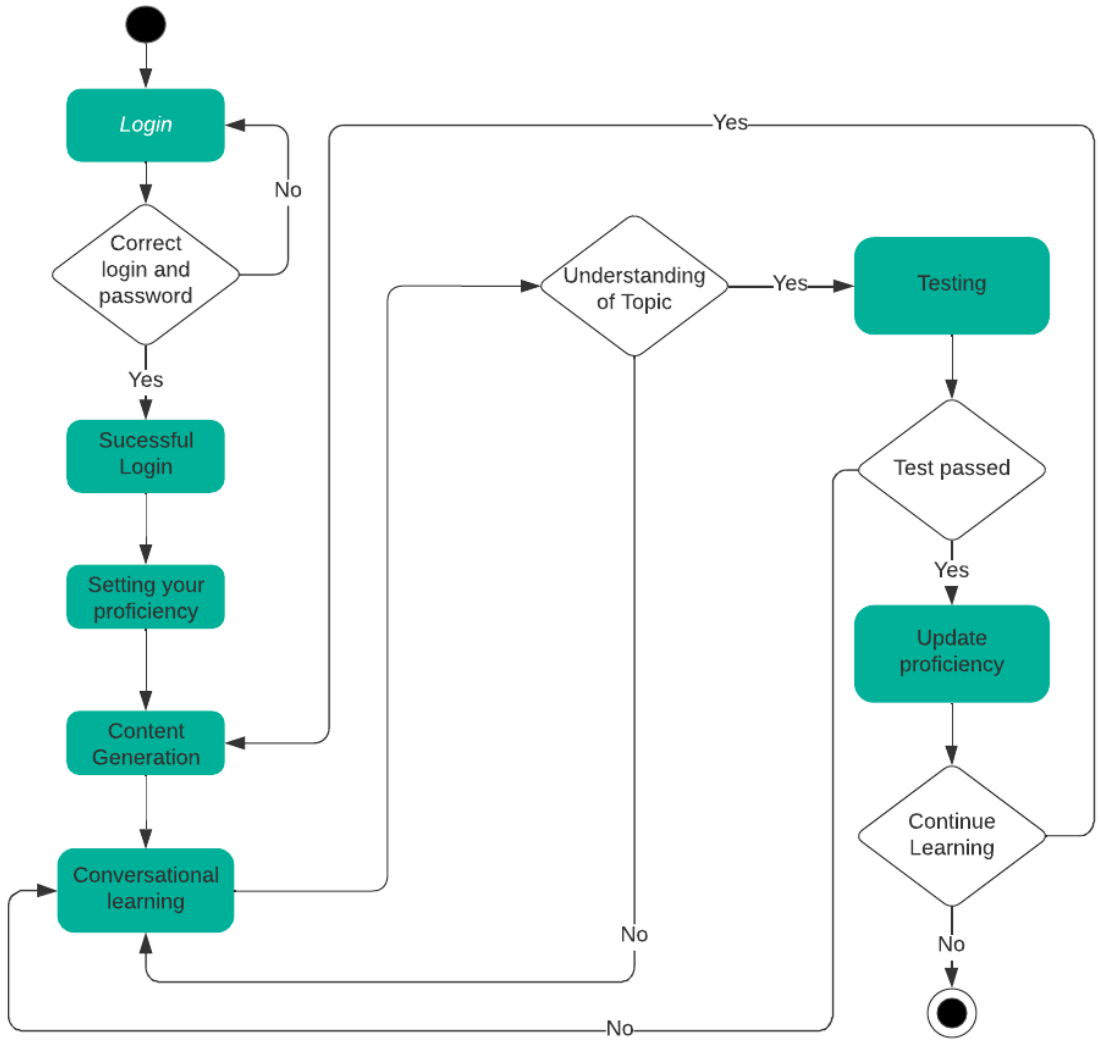

Figure 6.2: Activity Diagram

This activity diagram illustrates the flow after a user successfully logs in, allowing them to select their proficiency level. The application then generates tailored content for the user's learning, during which they can interact with a chatbot to ask questions related to the passage. Upon successful completion of the learning phase, users proceed to take a proficiency test to advance to a higher level of proficiency.

Chapter 7

Iteration 2

7.1 Prototype

In this iteration, we successfully developed a web application using the MERN stack, which comprises MongoDB, Express, React, and Node.js. The application features a React-based frontend and a backend built with Node.js and Express. For data storage, we utilized MongoDB to manage user information. We also established REST APIs to ensure seamless communication between the backend and frontend, resulting in a fully integrated and functional web application.

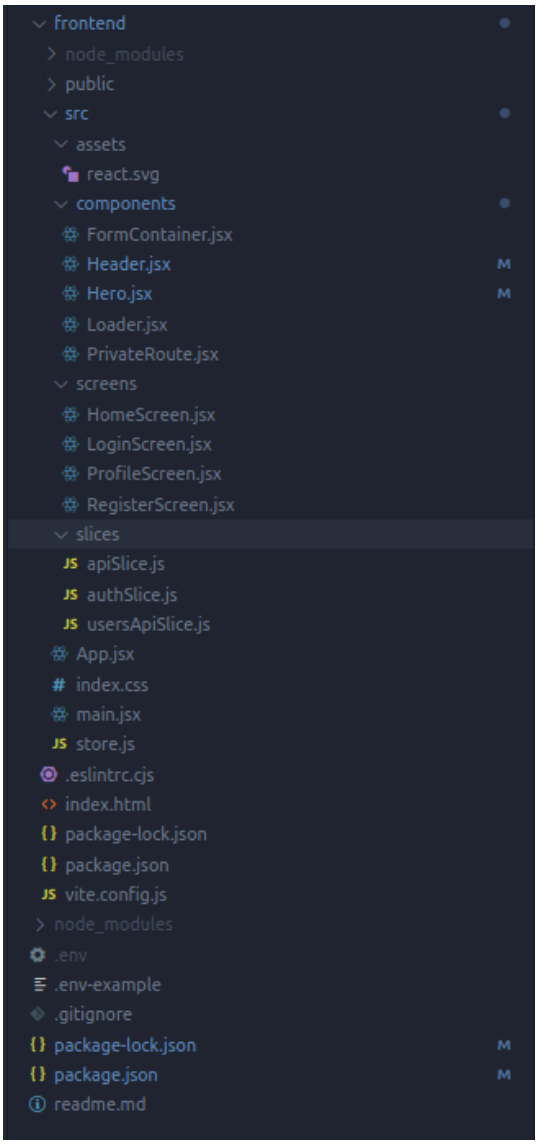

Figure 7.1: Structure of front-end

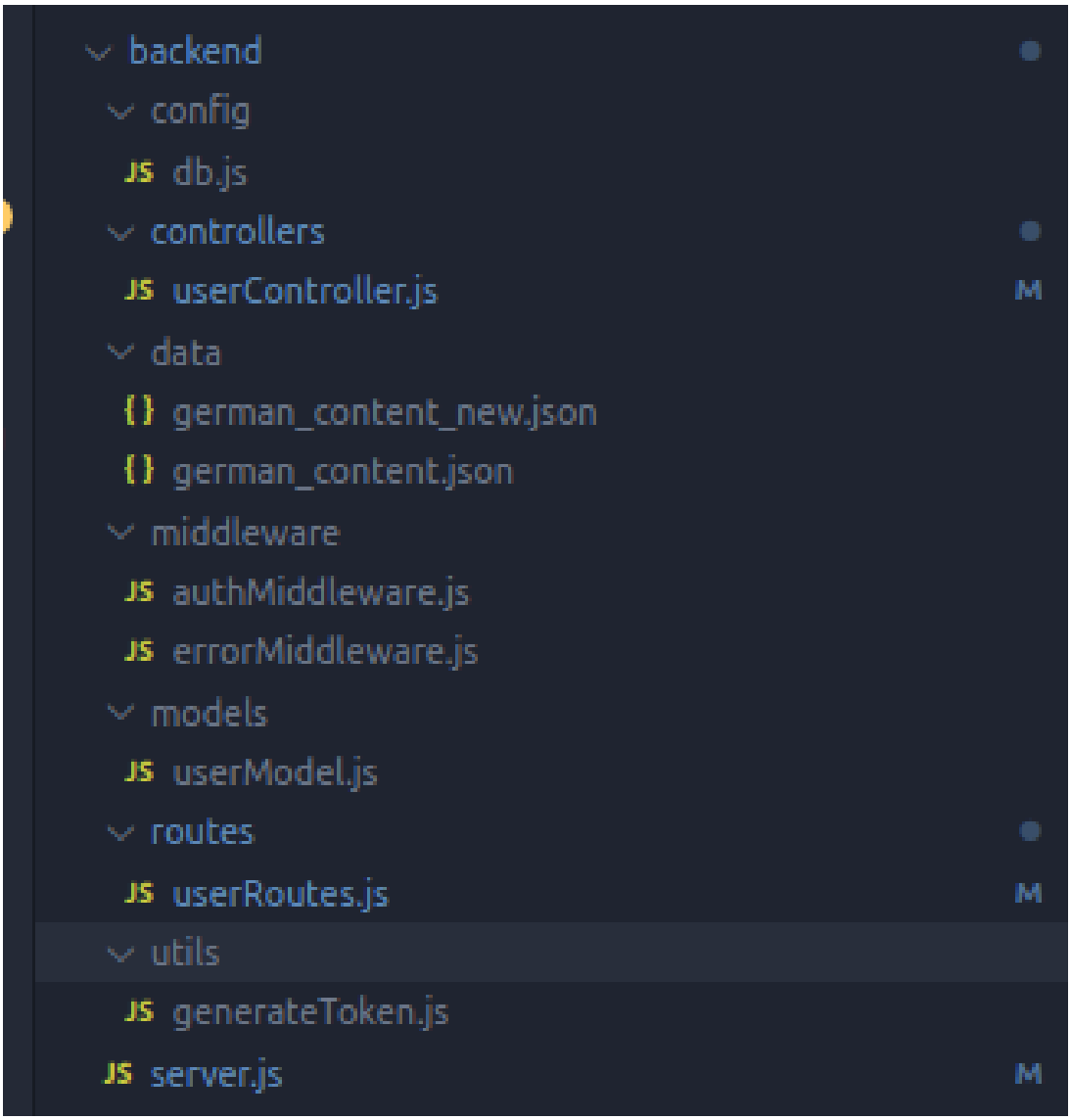

Figure 7.2: Structure of back-end

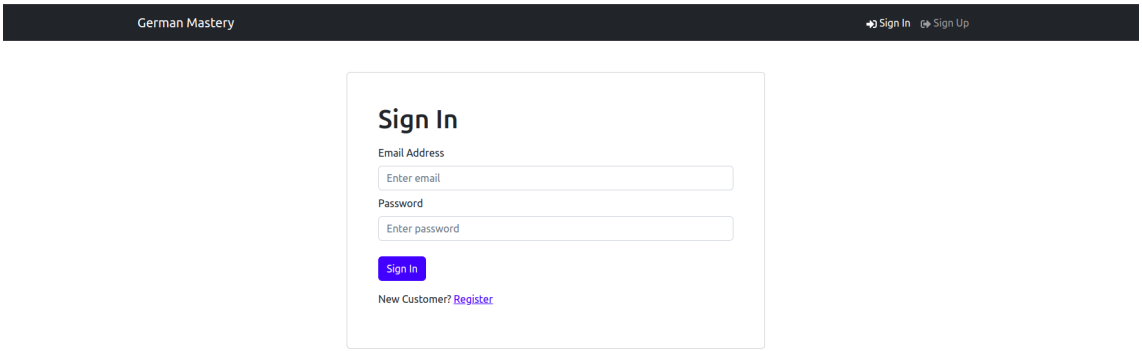

Figure 7.3: Signin

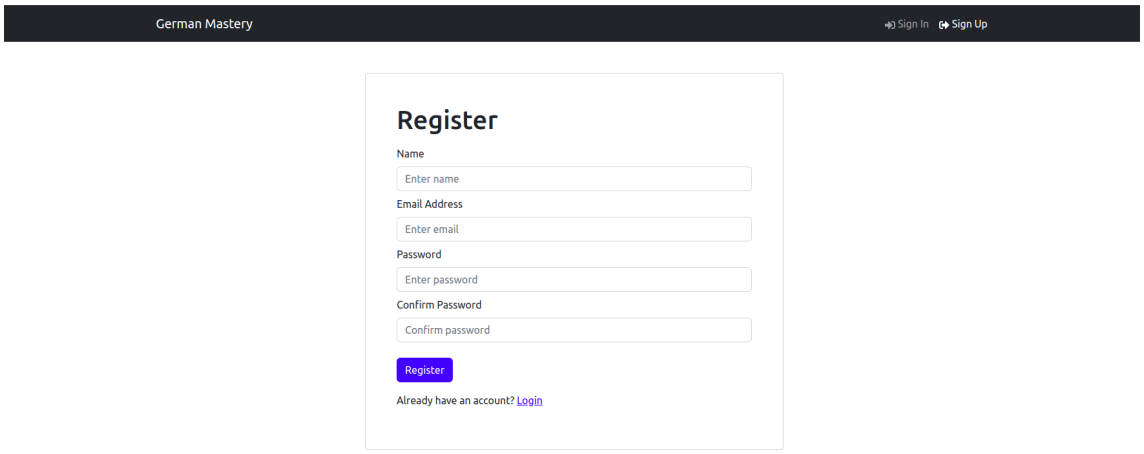

Figure 7.4: Signup

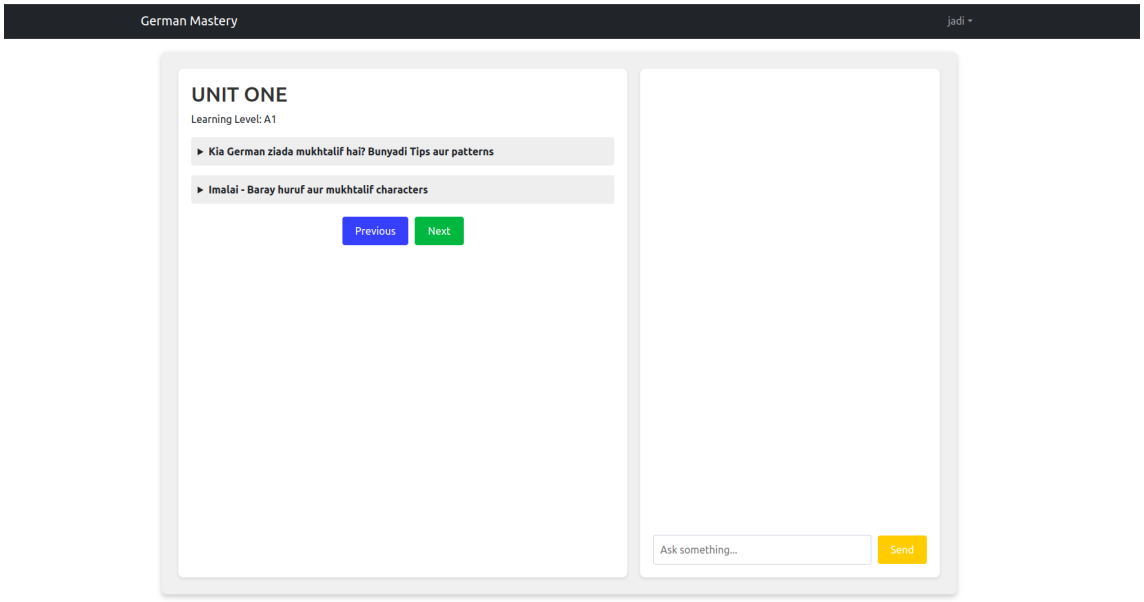

Figure 7.5: Dashboard

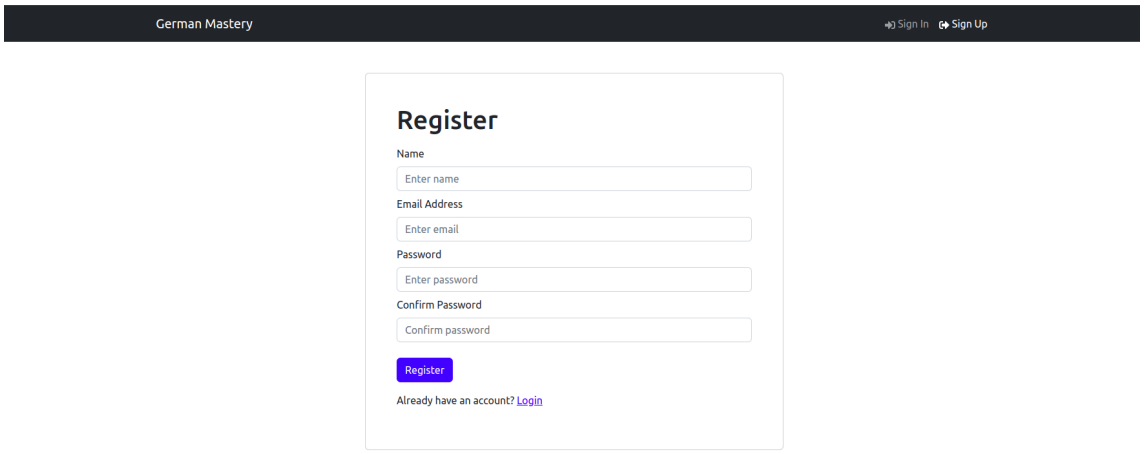

Figure 7.6: Update profile

Chapter 8

Iteration 3

8.1 Preparation Of Dataset

8.1.1 Preparing Data

We conducted research to identify relevant books aligning with our use case requirements. Following this, we extracted data from these books and processed it through a Language Model (LLM). The LLM translated the data from English to Roman Urdu and structured it into a suitable JSON format for further analysis and utilization.

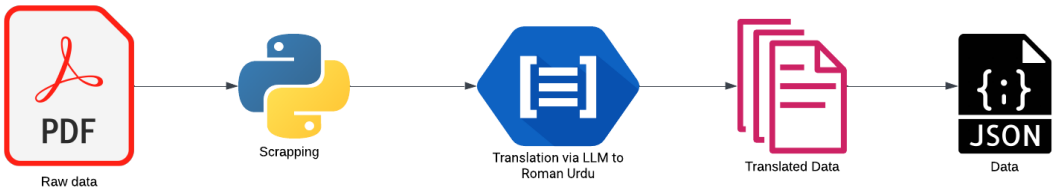

Figure 8.1: Preparing Data

8.1.2 Preparing Data for RAG

Utilizing the JSON file, we generated all feasible questions and answers, consolidating them into a CSV file for our question generation (QG) model.

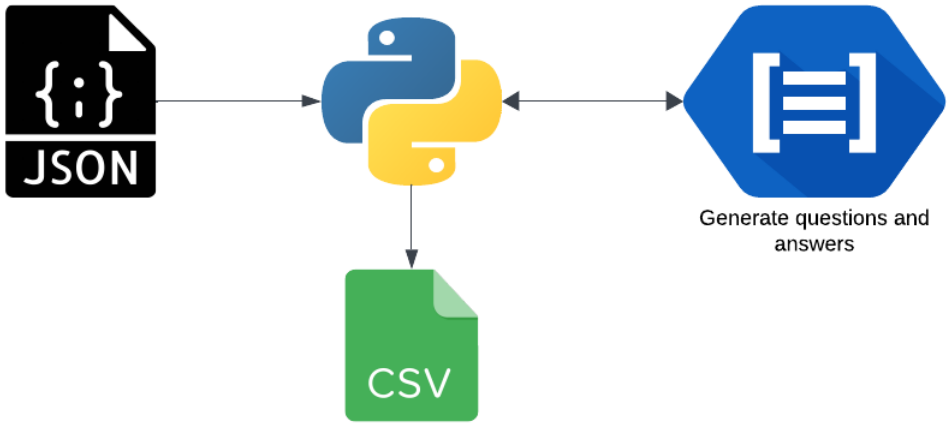

Figure 8.2: Preparing Questions

8.2 Preparation of RAG Chatbot

We have prepared the RAG-based chatbot and trained on the chatbot dataset. The chatbot is capable of understanding user queries and based on that responding to the user. We have used the Hugging face instruction embeddings and for storing those embeddings we have used the FAISS (Facebook AI similarity search) database. Here is the architecture diagram for our chatbot, its structure and code.

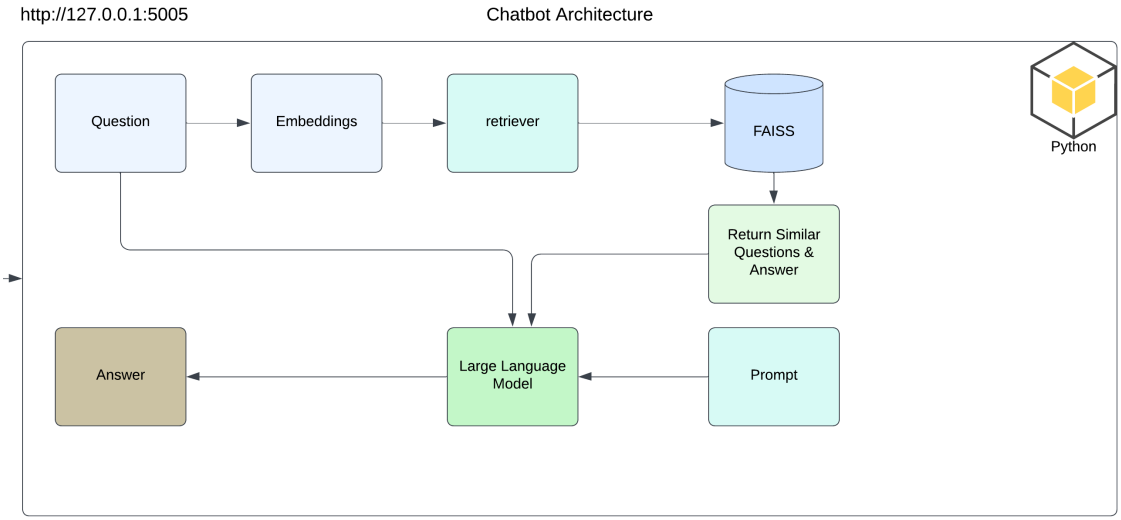

Figure 8.3: RAG chatbot architecture

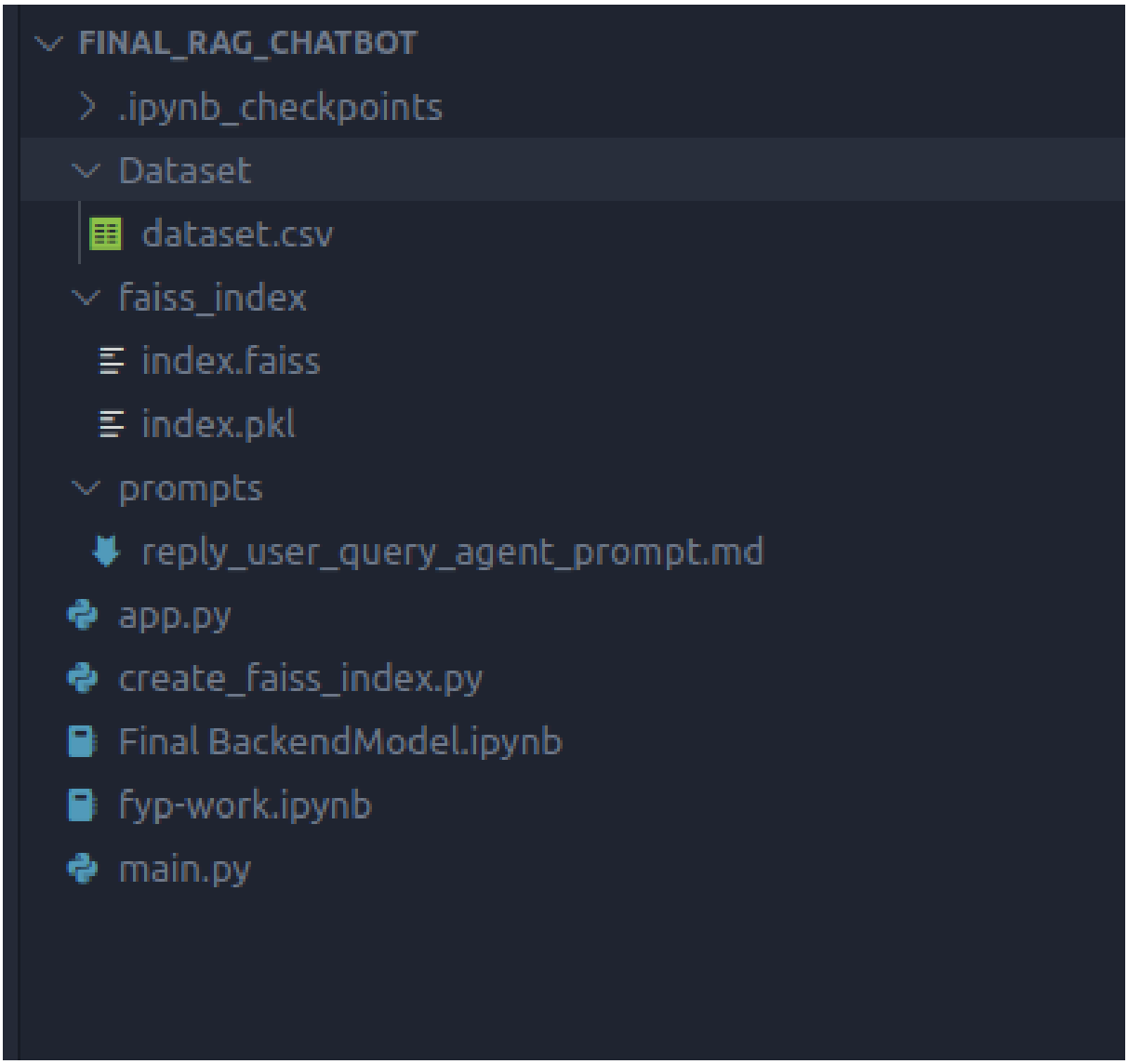

Figure 8.4: RAG file structure

```python
from flask import Flask, request, jsonify
from langchain.embeddings import HuggingFaceInstructEmbeddings
from langchain.vectorstores import FAISS
from flask_cors import CORS
from langchain.document_loaders.csv_loader import CSVLoader
from langchain.chains import RetrievalQA
from langchain.prompts import PromptTemplate
from langchain_openai import ChatOpenAI
from langchain import LLMChain
import os

app = Flask(__name__)
CORS(app)

# Setup the embeddings and FAISS vector store
instructor_embeddings = HuggingFaceInstructEmbeddings()
vectordb_file_path = "faiss_index"
vectordb = FAISS.load_local(vectordb_file_path, instructor_embeddings, allow_dangerous_deserialization=True)

# Setup the OpenAI API key
os.environ["OPENAI_API_KEY"] = "sk-proj-4eiNQl3DAi2HGj8kaH68T3BlbkFJRfe6LO4R7wMHramAdGZu"

# Determine the absolute path to the markdown file
base_dir = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))
markdown_file_path = os.path.join(base_dir, "Final_Rag_Chatbot", 'prompts', 'reply_user_query_agent_prompt.md')

print(markdown_file_path, 'hello')
# Load the prompt from the Markdown file
with open(markdown_file_path, 'r') as file:
    reply_user_query_agent = file.read()

# Create the prompt template
reply_user_query_agent_template = PromptTemplate(
    template=reply_user_query_agent,
    input_variables=["query", "context"]
)

# Create the LLMChain with OpenAI GPT-4 model
llm = ChatOpenAI(model_name="gpt-4", temperature=0.5)
chain = LLMChain(llm=llm, prompt=reply_user_query_agent_template)

@app.route('/get-answer', methods=['POST'])
def ask():
    # Get the query from the POST request
    data = request.json
    query = data.get("query")

    # Check if query was provided
    if not query:
        return jsonify({"error": "No query provided"}), 400

    # Retrieve relevant documents
    retriever = vectordb.as_retriever(score_threshold=0.7)
    rdocs = retriever.get_relevant_documents(query)
    context = rdocs

    # Generate the response
    response = chain.predict(query=query, context=context)
    return jsonify({"answer": response})

if __name__ == '__main__':
    app.run(debug=True, port=5005)

```

Figure 8.5: RAG code

Chapter 9

Iteration 4

In this iteration, we will test the application, add a feature for translating Roman Urdu to German in the chatbot, and deploy the application on the cloud.

9.1 Cloud Deployment

To make our platform accessible globally, we deployed it on Amazon Web Services (AWS) using separate Elastic Compute Cloud (EC2) instances for different components.

For the main web application, we launched an EC2 instance with an Apache web server to host the application code, serving as the front-end for users. A separate EC2 instance was provisioned for the chatbot component, where we installed the necessary libraries and deployed the chatbot code. Similarly, another EC2 instance was set up for the translation model component, with appropriate libraries and the translation model code.

Each instance was configured with suitable compute resources (CPU, memory, storage) to ensure optimal performance. To distribute traffic and ensure high availability, we implemented AWS Elastic Load Balancing, which automatically routes incoming requests to healthy instances.

The chatbot and translation model instances are integrated with the web application via APIs. User interactions with the chatbot or translation requests are handled through these APIs, fetching the required data or services from the respective instances.

Deploying our platform components on separate EC2 instances improved resource allocation, scalability, and maintainability. Each component can be independently scaled and updated without affecting the entire system.

9.2 Translation Model

We developed a sequence-to-sequence model for language translation using a dataset of 40,000 data pairs, 23,000 unique words, and 17,000 unique sentences. Due to hardware limitations, we trained the model on 9,000 sentences. The dataset was preprocessed with tokenization, vocabulary building, and one-hot encoding. Our model features an encoder-decoder architecture with LSTM layers (256 units each), converting sequences to probability distributions via a dense softmax layer. Training involved 200 epochs with a batch size of 256, using the Adam optimizer and categorical cross-entropy loss. We tracked validation loss and accuracy to evaluate performance. Next steps include extensive testing, adding Roman Urdu to German translation, and deploying the application on the cloud. An image of validation loss and accuracy metrics will be included to show model performance.

```
import tensorflow as tf
from tensorflow import keras
from keras.layers import Input, LSTM, Dense
from keras.models import Model
from keras.callbacks import ModelCheckpoint, EarlyStopping

#Dimensionality
dimensionality = 256
#The batch size and number of epochs
batch_size = 256
epochs = 100
#Encoder
encoder_inputs = Input(shape=(None, num_encoder_tokens))
encoder_lstm = LSTM(dimensionality, return_state=True)
encoder_outputs, state_hidden, state_cell = encoder_lstm(encoder_inputs)
encoder_states = [state_hidden, state_cell]
#Decoder
decoder_inputs = Input(shape=(None, num_decoder_tokens))
decoder_lstm = LSTM(dimensionality, return_sequences=True, return_state=True)
decoder_outputs, decoder_state_hidden, decoder_state_cell = decoder_lstm(decoder_inputs,
                                                    initial_state=encoder_states)
decoder_dense = Dense(num_decoder_tokens, activation='softmax')
decoder_outputs = decoder_dense(decoder_outputs)

# #Model
training_model = Model([encoder_inputs, decoder_inputs], decoder_outputs)
# #Compiling
training_model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])


checkpoint = ModelCheckpoint(filepath='model.h5', save_best_only=True, verbose=1)
early_stopping = EarlyStopping(monitor='val_loss', patience=3, verbose=1)

# Now, callbacks is a list of the defined callbacks
callbacks = [checkpoint, early_stopping]
#Training
# history = training_model.fit([encoder_input_data, decoder_input_data], decoder_target_data, batch_size = batch_size, epochs = epochs, validation_split = 0.2, callbacks=[my_callbacks])
history = training_model.fit([encoder_input_data, decoder_input_data],
                             decoder_target_data, batch_size = batch_size, epochs = epochs,
                             validation_split = 0.2, callbacks=[callbacks])
#training_model.save('training_model.h5')
```

Figure 9.1: Encode Decoder

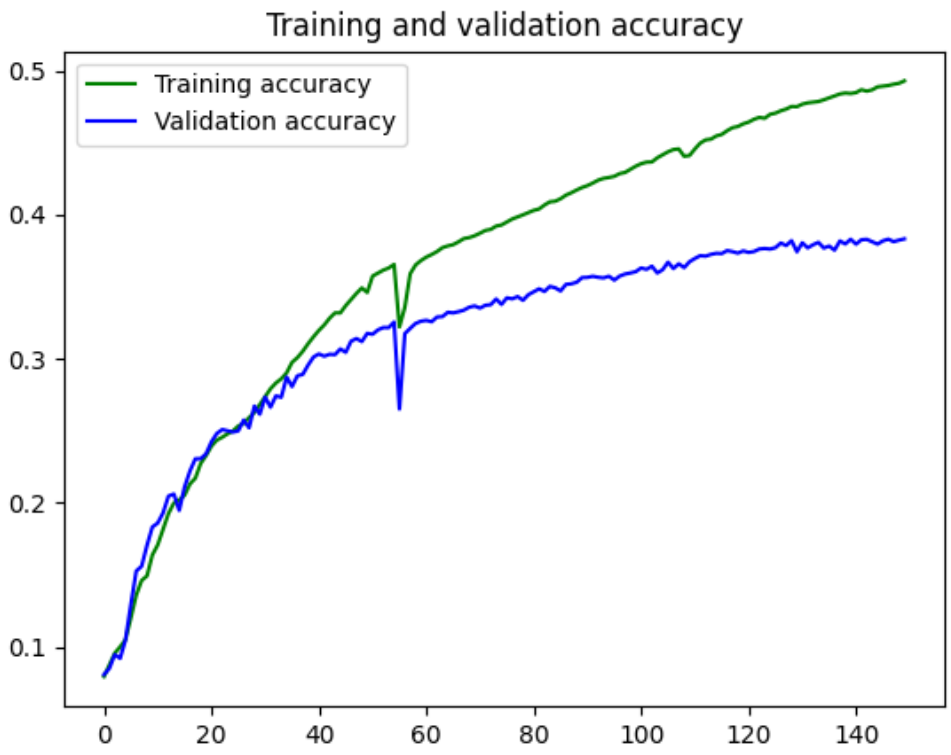

Figure 9.2: Sequence to Sequence

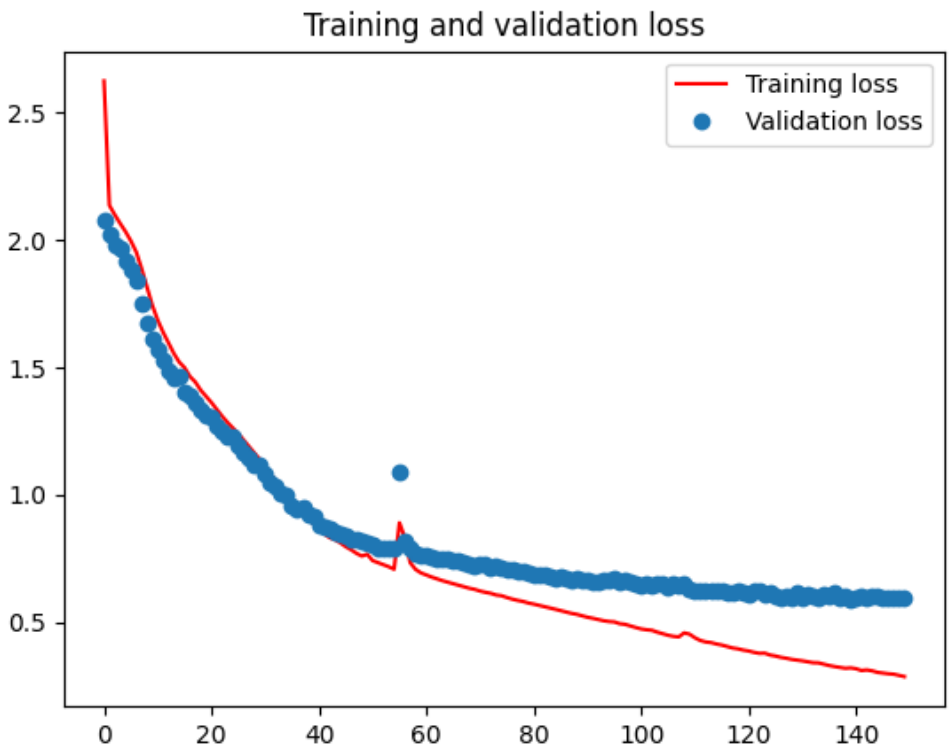

Figure 9.3: Sequence to Sequence

9.3 Testing

Feature	Test Cases	Results
User Login/Signup	Tested valid and invalid login credentials	All test cases passed
	Tested user registration with valid and invalid data	Some issues found with invalid data handling, now resolved
Content Showing	Tested displaying content for levels A1 and A2	Content displayed correctly for both levels
	Tested content pagination	Pagination implemented correctly
Generative Chatbot	Tested chatbot responses for various query types	Most queries handled well, some improvements needed for ambiguous queries
	Tested handling of ambiguous or irrelevant queries	Added better fallback responses for irrelevant queries
Translation Chatbot	Tested translation from Roman Urdu to German for different sentences	Translation quality is fair, but only 50
	Tested translation quality and accuracy	Accuracy meets some requirements but needs further improvement
Profile Management	Tested updating user profile information	Profile updates working correctly
	Tested tracking and displaying user progress	Progress tracking and display implemented as per requirements
	Tested completion of lessons or modules	Lesson/module completion status updated accurately

Table 9.1: Feature Testing Summary

Chapter 10

Implementation Details

10.1 Tools and Programming Languages

10.1.1 MERN Stack

For the creation of the web application, we have utilized the MERN stack which is MongoDB, Expressjs, React, and nodejs. Here are details of for what purpose we have utilized each tool in our project.

- **Reactjs:** We have used reactjs for the creation of the responsive frontend. We have used the redux toolkit to maintain and update data across our application for multiple components to share, all while remaining independent of the components.

- **Expressjs and Nodejs:** We have used expressjs and nodejs for the creation of the backend. We have used the Mongoose node package to create connection with database. User login, signup, content fetching and showing, and user profile management we have done using the expressjs and nodejs.

- **MongoDB:** We have utilized the NoSQL database MongoDB for storing the user data.

10.1.2 Git and Github

We utilized Git and GitHub, both distributed version control systems, to facilitate collaboration and track changes throughout our project. These tools allowed multiple team members to work simultaneously on different aspects of the project, efficiently manage code versions, and maintain a comprehensive history of all modifications. This approach ensured seamless collaboration and streamlined our development process.

10.1.3 Python, Flask

We have used Python for the following purposes.

- **Dataset Creation:** We have utilized Python for the creation of our dataset. We have used Python different libraries for interacting with the text data and converting that into the required dataset format.

- **Chatbot:** For the creation of the chatbot we have utilized Python libraries and then we created a POST rest API in Flask that is going to take user questions and then respond to user queries. That is how the backend chatbot server is going to communicate with the front end.

- **Language Translation Model:** We developed a sequence-to-sequence language translation model using Python, TensorFlow, and Keras. The application will be deployed using Flask for creating a REST API, enabling communication between the backend and frontend.

10.1.4 Vector Databases and Hugging Face

We utilized Hugging Face's instructor embeddings to convert our CSV question-answer dataset into a numerical form, making it comprehensible for the computer. These numerical embeddings were then stored in the FAISS (Facebook AI Similarity Search) database, facilitating efficient similarity search and retrieval operations.

10.1.5 Cloud Technologies

We utilized various Amazon Web Services (AWS) technologies to ensure scalability, reliability, and performance for our platform. Amazon EC2 (Elastic Compute Cloud) was employed to launch and manage virtual servers, hosting different components of our platform, including the main web application, chatbot, and translation model. Amazon S3 (Simple Storage Service) was used for storing and retrieving large amounts of data, such as user-generated content and datasets. To manage relational databases in the cloud, we utilized Amazon RDS (Relational Database Service), providing a robust and scalable solution for storing structured data.

AWS Elastic Load Balancing was implemented to automatically distribute incoming application traffic across multiple EC2 instances, ensuring high availability and fault tolerance. For monitoring and logging, we employed Amazon CloudWatch to track the application's performance and health metrics, enabling proactive management and troubleshooting. AWS Lambda was leveraged to run serverless functions in response to specific events, allowing for scalable and efficient processing without the need to manage servers.

Additionally, Amazon Route 53 was used for DNS web service, ensuring reliable routing of end users to the application. Finally, AWS IAM (Identity and Access Management) was implemented for secure management of access to AWS services and resources, ensuring proper authentication and authorization protocols. These technologies collectively contributed to the seamless operation and scalability of our platform.

Chapter 11

Results

11.1 RAG Chatbot results

The RAG chatbot is performing exceptionally well in understanding Roman Urdu and generating responses in both Roman Urdu and German. It demonstrates remarkable proficiency in comprehending user queries and applying its knowledge efficiently to provide accurate answers. Based on human evaluations, the chatbot has achieved outstanding accuracy in furnishing users with correct and relevant responses to their inquiries.

11.2 Translation Model result

The sequence-to-sequence (seq2seq) model, trained on a dataset of nine thousand samples, has achieved an accuracy of fifty percent in translating Roman Urdu text to German. This model demonstrates the capability to effectively translate content from Roman Urdu into the German language.

Chapter 12

User manual

12.1 User Manual

To get started, sign up or log in to create your account and set your German proficiency level to either A1 or A2. Once logged in, you can access structured lessons that cover grammar, vocabulary, and writing tailored to your selected proficiency level. You can interact with the AI chatbot to ask questions related to the lesson content in Roman Urdu. Additionally, you can track your learning progress in the profile section and update your profile settings as needed.

Chapter 13

Conclusions and Future Work

13.1 Conclusion

We developed an AI chatbot web app designed to help users learn German from Roman Urdu, specifically targeting A1-A2 levels. To support this, we created custom datasets and employed retrieval-augmented generation for the chatbot. The app was built using the MERN stack, offering interactive learning and Q&A capabilities. This project addresses the scarcity of accessible German language resources for Roman Urdu speakers, providing a valuable tool for language learners.

13.2 Future Work

To enhance our AI chatbot web app, we plan to improve the dataset further and expand it to include proficiency levels from B1 to C2. Additionally, we aim to extend support to more languages beyond German and Roman Urdu. We will focus on improving the chatbot's accuracy and integrating more visual content to enrich the learning experience. Our development process will be continuously refined based on user feedback and advancements in AI technology, ensuring the app remains effective and up-to-date.

Bibliography

[1] Weijiao Huang, Khe Foon Hew, and Luke K.Fryer. Chatbots for language learning—are they really useful? a systematic review of chatbot-supported language learning. *Journal of Computer Assisted Learning*, 38(1):237–257, 2022.

[2] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. 2017.

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[4] Hila Gonen, Shauli Ravfogel, Yanai Elazar, and Yoav Goldberg. It's not greek to mbert: Inducing word-level translations from multilingual bert. *arXiv preprint arXiv:2010.08275*, 2020.

[5] Hugo Touvron et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

[6] Guilherme Penedo et al. The refinedweb dataset for falcon llm: Outperforming curated corpora with web data, and web data only. *arXiv preprint arXiv:2306.01116*, 2023.

[7] Ping W. Xu P. Shoeybi M. Chang K.-C. Catanzaro B. Huang, J. In-context learning with retrieval augmented encoder-decoder language models. *arXiv:2308.07922.*, 2023.

[8] Lewis P. Lomeli M. Hosseini L. Petroni F.-Schick T. Dwivedi-Yu J. Joulin A. Riedel S. Grave E. Izacard, G. Few-shot learning with retrieval augmented language models. *arXiv:2208.03299*, 2022.

[9] Allard J. Jarvis, C. A survey of techniques for maximizing llm performance. *OpenAI*, 2023.

[10] Wu Q. Lin C.-Y. Yang Y. Qiu L. Jiang, H. Llmlingua: Compressing prompts for accelerated inference of large language models. *arXiv:2310.05736.*, 2023.

[11] Xu F. F. Gao-L. Sun Z. Liu Q. Dwivedi-Yu-J. Yang Y. Callan J. Neubig G. Jiang, Z. Active retrieval augmented generation. *arXiv:2305.06983.*, 2023.

[12] Deng H. Roberts A.-Wallace E. Raffel C. Kandpal, N. Large language models struggle to learn long-tail knowledge. *International Conference on Machine Learning.*, 2023.

[13] Kwak J. M. Baek-J. Hwang S. J. Kang, M. Knowledge graph-augmented language models for knowledge-grounded dialogue generation. *arXiv:2305.18846.*, 2023.

[14] Kim S. Jeon B.-Park J. Kang J. Kim, G. Tree of clarifications: Answering ambiguous questions with retrieval-augmented large language models. *arXiv:2301.06800*, 2023.

[15] Santhanam K. Li X. L. Hall D. Liang P. Potts-C.- Zaharia M. Khattab, O. Demonstrate-search-predict: Composing retrieval and language models for knowledge-intensive nlp. *arXiv:2212.14024.*, 2022.

[16] et al. Kim, G. Tree of clarifications: Answering ambiguous questions with retrieval-augmented large language models. *arXiv:2301.06800.*, 2023.

[17] Oğuz B. Min S. Lewis P. Wu L. Edunov S.-Chen-D. Yih W.-T. Karpukhin, V. Dense passage retrieval for open-domain question answering. *arXiv:2004.04906.*, 2020.

[18] Levy O. Jurafsky D. Zettlemoyer L. Lewis M. Khandelwal, U. Generalization through memorization: Nearest neighbor language models. *arXiv:1911.00172*, 2019.

[19] Tri Dao Albert Gu. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv:2312.00752*, 2023.

[20] Yinhan Liu Luke Zettlemoyer Marjan Ghazvininejad, Omer Levy. Mask-predict: Parallel decoding of conditional masked language models. *arXiv:1911.02054.*, 2019.

[21] Kai-Wei Chang Di Wu, Wasi Uddin Ahmad. Rethinking model selection and decoding for keyphrase generation with pre-trained sequence-to-sequence models. *arXiv:2310.06374*, 2023.

[22] Mingxuan Wang-Xipeng Qiu Jiangtao Feng Hao Zhou-Lei Li Zehui Lin, Xiao Pan. Pre-training multilingual neural machine translation by leveraging alignment information. *arXiv:2010.03142*, 2020.

[23] Naman Goyal. et all Mike Lewis, Yinhan Liu. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv:1910.13461*, 2019.

[24] Naman Goyal. et all Yinhan Liu, Myle Ott. Roberta: A robustly optimized bert pretraining approach. *arXiv:1907.11692*, 2019.

[25] Narasimhan K. Salimans T. Sutskever I. Radford, A. Improving language understanding by generative pre-training. *OpenAI*, 2019.

[26] Edunov S. Baevski A. et al. Ott, M. fairseq: A fast, extensible toolkit for sequence modeling. *NAACL*, 2019.