University of Debrecen

Faculty: Faculty of Informatik
Title: Automatic Quiz Generation System (AQGS)
Student name: Sanaa Harun Abdurrahman Al -Ahdal
Supervisors:
Prof. Dr. András Hajdu – Head of Department, Full Professor
Róbert Lakatos - Assistant Lecturer

Debrecen – Hungary - 2024

I agree to submit it for review ……

Table of Content

Chapter 1: Introduction

1.1 Introduction ..... Page 2

Chapter 2: Discussion

2.1 Architecture Overview ..... Page 5

2.2 Technical Implementation..... Page 7

2.2.1Algorithms and Techniques

2.2.2 Tools and Technologies

2.2.3Implementation

2.3 Challenges and Solutions ..... Page 21

2.4 Future Directions …… Page 23

Chapter 3: Thesis Summary

3.1 Summary ……. Page 25

Description of own work ……. Page 26

Appendix ……..…… Page 27

Acknowledgments ……..…… Page 29

References ……..……Page 30

Chapter 1

1.1 Background

In recent years, advancements in artificial intelligence (AI) and natural language processing (NLP) have revolutionized various fields, including education. As educational institutions increasingly embrace digital resources, there is a growing demand for innovative tools that can enhance the learning experience and streamline educational processes. With the digitization of educational materials, there arises a pressing need to efficiently extract, organize, and utilize knowledge from vast repositories of textual data. In response to this need, the Automatic Quiz Generation System emerges as a promising solution, leveraging advancements in Natural Language Processing (NLP) and Artificial Intelligence (AI) to automate the process of quiz creation from textual resources.

Despite the abundance of educational content available in PDF format or any other digital format, the process of creating quizzes based on this content remains predominantly manual. Educators and instructional designers often spend significant time and effort extracting relevant information and formulating quiz questions, which can be a daunting task, especially for large volumes of documents. Moreover, manual quiz creation may limit the scalability and adaptability of assessments, hindering the potential for personalized learning experiences.

1.2 Problem Statement

Despite the widespread adoption of digital resources in education, the Conventional process of quiz creation from Digital documents persists as a significant bottleneck. Educators, content creators, and instructional designers face daunting challenges in extracting relevant information and formulating engaging quiz questions from voluminous Digital materials. This labor-intensive process not only consumes valuable time and resources but also inhibits the scalability and adaptability of assessments, relying heavily on manual effort and expertise. Furthermore, the lack of automated tools for quiz generation leads to a dearth of diverse and contextually relevant assessment materials, potentially undermining the effectiveness of learning experiences. As a result, there is a pressing need for innovative solutions that harness the power of AI and NLP to automate quiz generation from Digital documents, thereby alleviating the burden on educators and enhancing the quality and efficiency of educational assessments.

By automating the quiz generation process, this system addresses these challenges, offering educators and learners alike a scalable, personalized, and dynamic means of assessing comprehension reinforcing learning objectives, and achieving greater scalability and adaptability in educational assessments.

1.3 Objectives:

The primary objective of this research is to develop an Automatic Quiz Generation System (AQGS) capable of intelligently extracting information from Digital documents and employing prompt engineering techniques to formulate contextually relevant quiz questions aligned with learning objectives. By leveraging state-of-the-art NLP models, such as Large Language Models (LLMs), the AQGS will analyze the textual content of Digital files, identify key concepts, and generate a variety of quiz questions, including multiple-choice, true/false, and short-answer questions.

Prompt engineering techniques will be utilized to design prompts that effectively elicit desired responses from learners, thereby enhancing the quality and relevance of generated quizzes. This approach not only streamlines the process of quiz creation but also ensures that the assessments are tailored to the specific content and objectives of each document or course. Additionally, the AQGS aims to provide educators with the flexibility to generate a variable number of quizzes, enabling them to adapt assessments to diverse learning needs and preferences while maintaining efficiency and effectiveness.

1.4 Scope and Limitations

The scope of this research encompasses the design, development, and evaluation of the AQGS, focusing on its effectiveness in generating quizzes across diverse subject areas and educational levels. However, it is important to acknowledge certain limitations, including the potential challenges in accurately interpreting complex or domain-specific content within PDF documents, as well as constraints on the diversity and quality of generated quiz questions.

By addressing these challenges and limitations, this thesis seeks to contribute to the advancement of educational technology and pedagogy by providing educators with a powerful tool for automating quiz generation and enhancing the learning experience for students.

Chapter 2

2.1 Architecture Overview:

Architecture Overview:

To design and implement our Automatic Quiz Generation System, we followed a systematic approach consisting of several key steps. Firstly, we extracted all relevant data from the input files, converting them into text format for further processing. Next, we identified important information and headlines from the text and compiled them into a structured list, ensuring that key concepts are captured for quiz generation.

To manage the potentially large volume of text data, we segmented the extracted text into smaller, manageable 'chunks', facilitating efficient processing and analysis. Each chunk was then embedded into a numerical representation using advanced embedding techniques, enabling semantic understanding and comparison.

To enable efficient indexing and retrieval of information, we constructed a Semantic Index, which serves as a repository for the embedded

text chunks. This index streamlines the search process and enhances the accuracy of information retrieval.

The core functionality of our system revolves around the interaction between two main components: the AI model and the knowledge database. The AI model utilizes the semantic information encoded in the text embeddings to generate quiz questions. Meanwhile, the knowledge database stores and retrieves relevant information based on user queries, enhancing the contextual relevance of the generated questions.

The interaction between these components is orchestrated as follows: First, the system sends a query to the knowledge database to retrieve relevant information based on the input text. Simultaneously, the query is also forwarded to the AI model, which leverages the retrieved knowledge to generate quiz questions. By incorporating prompt templates, the AI model generates questions that are closely aligned with the content extracted from the input files. See flowchart 2.1

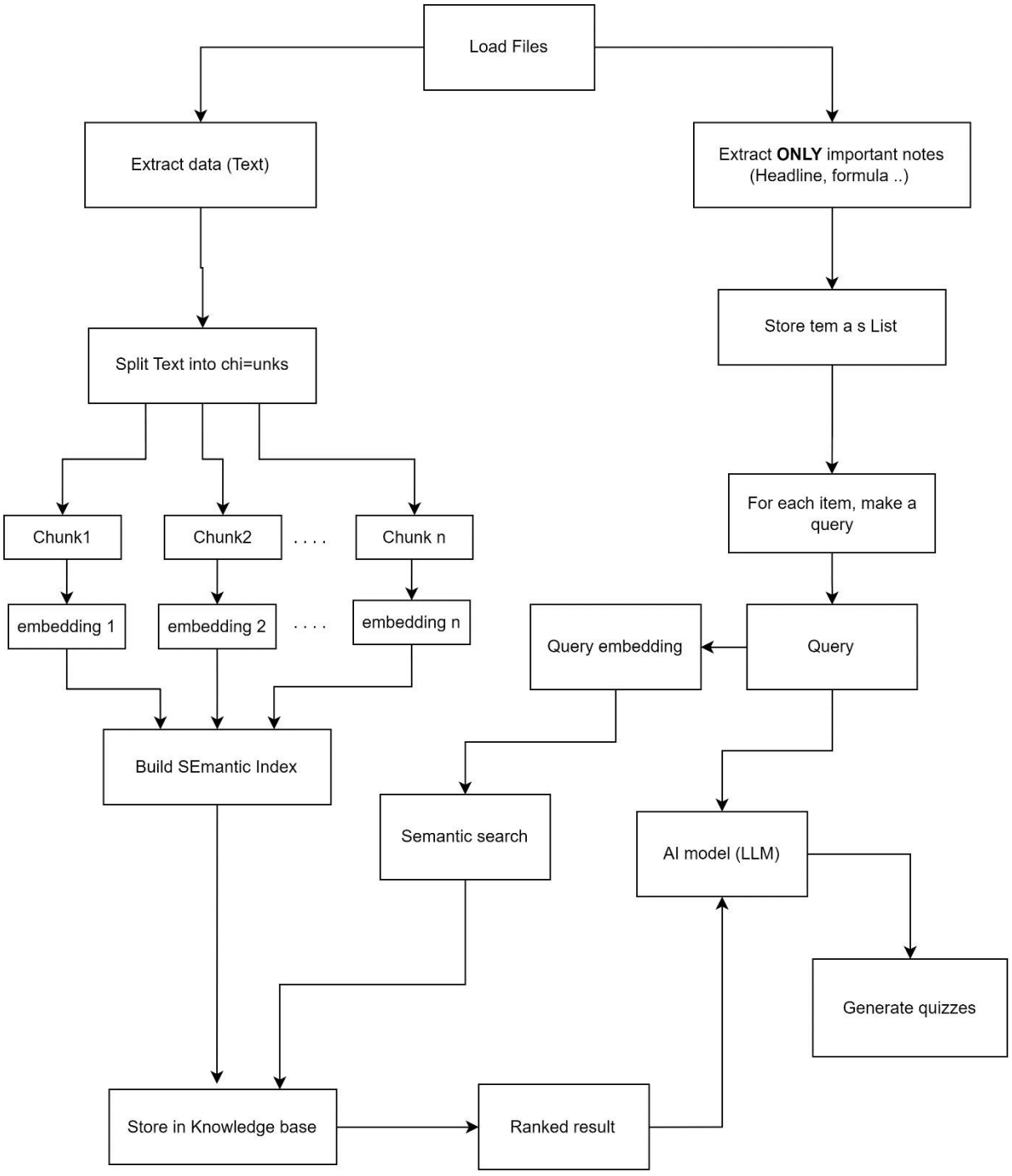


Figure 2.1: an overview architecture

2.2 Technical Implementation

2.2.1 Algorithms and Techniques:

Algorithms and Techniques Utilized

Text Segmentation using Langchain[1]:
Langchain is a text processing technique used to segment large blocks of text into smaller, manageable "chunks" based on linguistic patterns and syntactic structures. The Langchain algorithm breaks down the input text into cohesive segments, preserving the semantic coherence of the original content.
By segmenting the text into smaller chunks, the system improves processing efficiency and facilitates more targeted analysis and retrieval of information.

GROBID[2] for Information Extraction:
GROBID (Generation of Bibliographic Data) is a machine learning tool used for extracting and structuring information from scholarly documents such as PDF files. Utilizing GROBID, the system extracts important information from the input files, including headlines, metadata, and other structured data.

GROBID's capabilities enhance the system's ability to identify and capture key concepts and entities from the input text, providing valuable context for quiz generation.

Embedding Creation:
After segmenting the text into smaller chunks, each chunk is transformed into a numerical representation known as an embedding. The embedding creation process involves mapping words or phrases from the text to dense, high-dimensional vectors in a continuous vector space.
Techniques such as Word Embeddings (e.g., Word2Vec**[3],** GloVe**[4]**) or Transformer-based models (e.g., BERT**[5]**) are commonly used to generate embeddings that capture the semantic relationships between words and phrases.
These embeddings encode semantic meaning and contextual information, enabling the system to understand and process the textual content more effectively.

Storage in Pinecone[6] Dataset:
Pinecone is a cloud-based vector database designed for storing and querying high-dimensional vector data efficiently.

After creating embeddings for each text chunk, the vectors are stored in the Pinecone dataset, enabling fast and scalable retrieval of similar vectors based on semantic similarity.
Pinecone's indexing and querying capabilities streamline the process of semantic search, allowing the system to retrieve relevant information quickly and accurately.

Semantic Search and Indexing:

Semantic search involves retrieving information based on the meaning and context of the query, rather than relying solely on keyword matching.
The semantic index created in Pinecone facilitates efficient indexing and retrieval of text embeddings, enabling semantic search operations.
By comparing the similarity between query embeddings and stored embeddings in the index, the system can identify and retrieve text chunks that are semantically related to the query.
Semantic indexing enhances the accuracy and relevance of search results, ensuring that retrieved information closely aligns with the user's intent.

Prompt Engineering for AI Interaction:
Prompt engineering techniques involve crafting precise and effective prompts to guide the AI model's behavior and output. By designing tailored prompts, the system can elicit specific responses from the AI model that align with the desired task, such as quiz question generation.
Techniques such as template-based prompts, question scaffolding, and context-specific prompts can be employed to steer the AI model towards generating relevant and high-quality quiz questions.
Prompt engineering fosters effective communication between the system and the AI model, enhancing the generation of quiz questions that are coherent, informative, and aligned with the content extracted from the input text.

By integrating these algorithms and techniques, including GROBID and prompt engineering, into the system architecture, the Automatic Quiz Generation System can efficiently process text data, extract important information, interact with AI models, and generate quiz questions that are both contextually relevant and pedagogically meaningful.

2.2.2 Tools and Technologies:

LLM (meta-llama/Llama-2-7b-chat-hf) [7]:
This library, based on Hugging Face's**[8]** Transformers, provides access to pre-trained language models for natural language processing tasks.
Llama-2-7b-chat-hf specifically offers fine-tuned language models suitable for conversational applications, which can be leveraged for generating quiz questions in a dialogue-like manner.

Its integration enhances the system's ability to communicate effectively with AI models and generate contextually relevant quiz questions.

Langchain:
Langchain is a text processing tool used for segmenting large blocks of text into smaller, more manageable chunks based on linguistic patterns. By breaking down text into cohesive segments, Langchain improves the efficiency and accuracy of subsequent text analysis and processing tasks. Its utilization facilitates the extraction of key information and the creation of embeddings for quiz question generation.

Pinecone:
Pinecone is a cloud-based vector database designed for storing and querying high-dimensional vector data efficiently.

It serves as a repository for storing text embeddings generated from segmented text chunks, enabling fast and scalable retrieval of semantically similar vectors. Pinecone's indexing and querying capabilities enhance the system's ability to perform semantic search and retrieve relevant information for quiz generation.

Transformers and Hugging Face[8]:
Transformers is an open-source library developed by Hugging Face for natural language processing tasks, including text generation and understanding.
The library offers access to a wide range of pre-trained language models, such as BERT**[5]**, and T5**[9]**, which can be fine-tuned for specific applications.
Hugging Face provides a user-friendly interface and extensive documentation for working with state-of-the-art NLP models, facilitating their integration into the system for quiz question generation.

nbformat, presentation, pdfminer, docx, GROBID:
These are additional tools and libraries used for handling various types of input documents, including notebooks (**nbformat**), presentations (**presentation**), PDF files (**pdfminer**), and Microsoft Word documents (**docx**).

GROBID (Generation of Bibliographic Data) is specifically utilized for extracting and structuring information from scholarly documents, such as PDF files.

These tools and libraries enhance the system's capability to ingest and process diverse sources of textual content, ensuring comprehensive coverage and accuracy in information extraction for quiz generation.

Python and Jupyter Notebook:
Python serves as the primary programming language for implementing the Automatic Quiz Generation System.
Jupyter Notebook provides an interactive development environment for prototyping, testing, and documenting code, facilitating iterative development and collaboration.

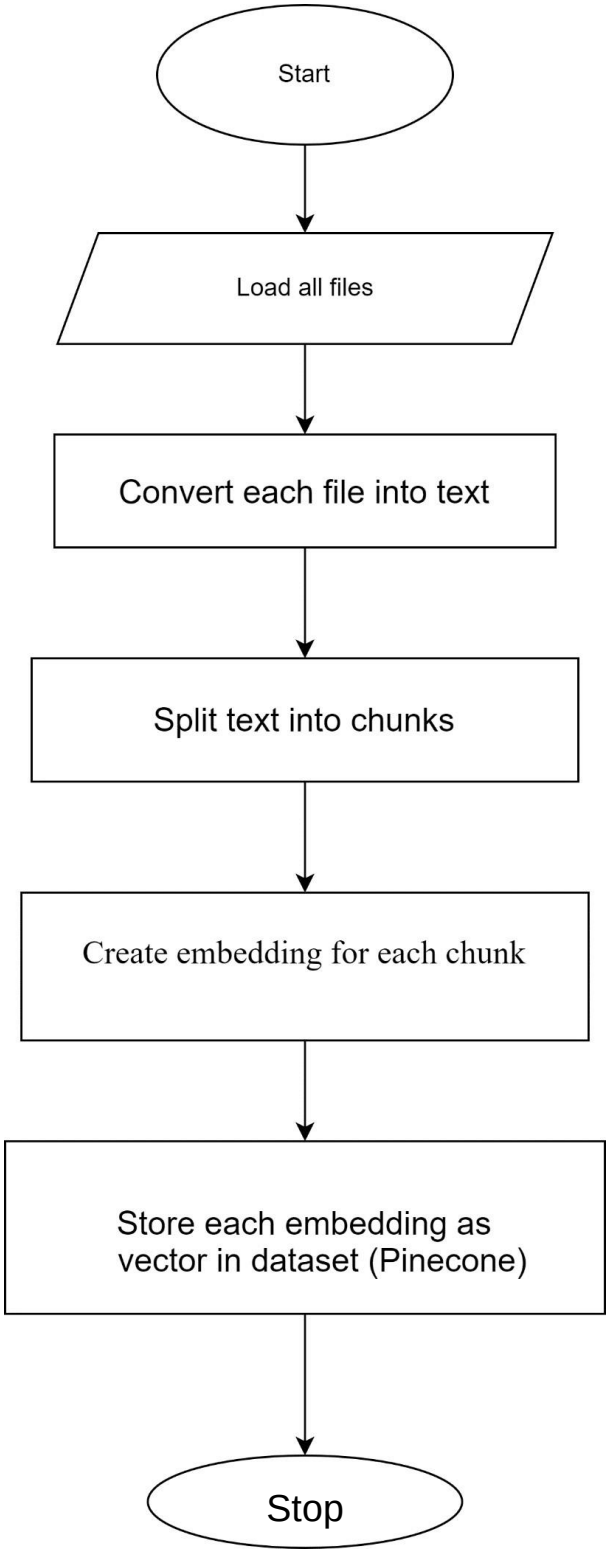
By leveraging these tools and libraries, along with the Python programming language and Jupyter Notebook environment, the Automatic Quiz Generation System achieves robustness, scalability, and efficiency in processing textual data and generating contextually relevant quiz questions.

2.2.3 Implementation

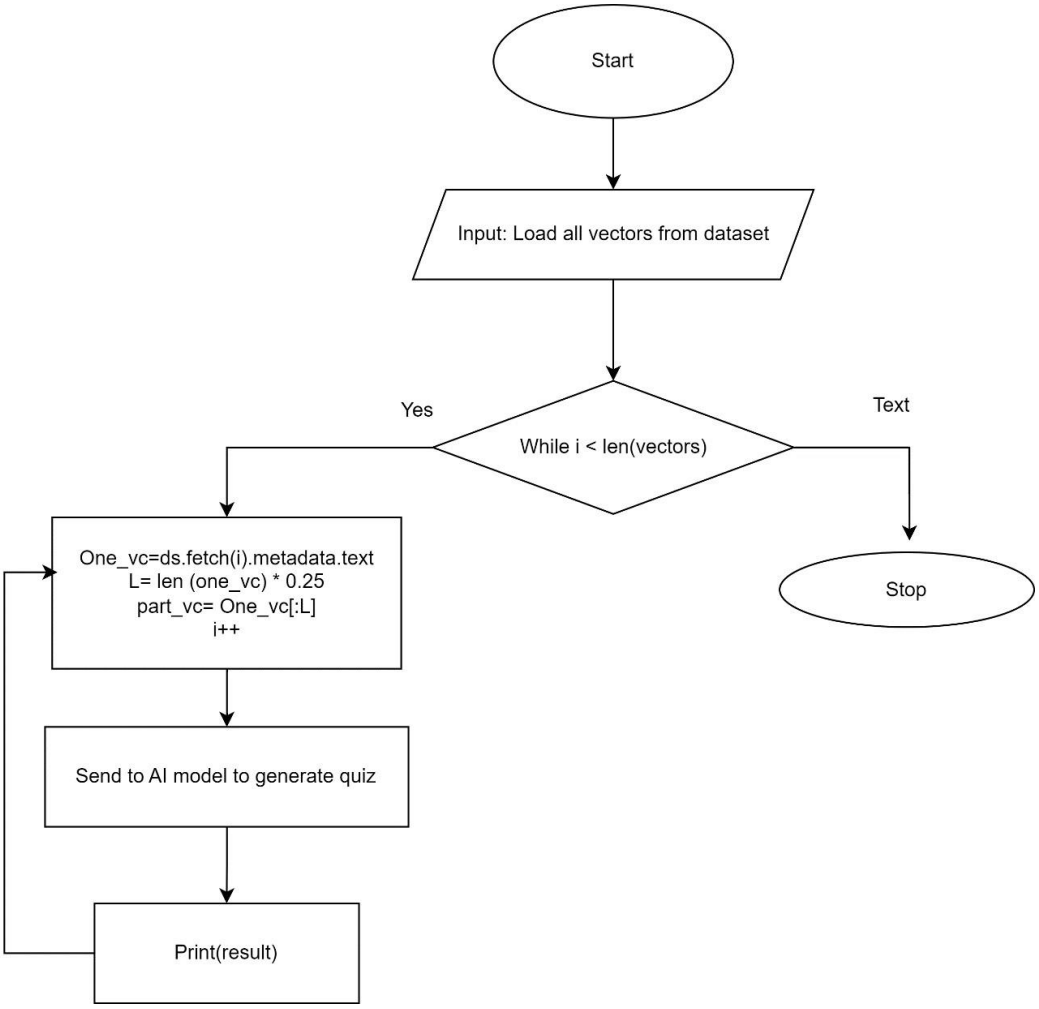
In the implementation phase of our Automatic Quiz Generation System, we adopted three distinct approaches to achieve our goal of generating quizzes from PDF files. Each approach leveraged a combination of tools, libraries, and frameworks to process the input data, generate quiz questions, and manage the interaction with AI models.

Approach 1: Direct Text Processing (First approach: use different packages)

- Extraction using Various Packages:
  Utilized Nbformat, Presentation, PDFMiner, and Docx packages for comprehensive extraction of content from diverse file formats like Jupyter notebooks, PowerPoint presentations, PDFs, and Word documents.
- Text Segmentation and Embedding:
  Split the extracted text into manageable chunks (1500-2000 characters each).
  Applied embedding techniques to encode the semantic meaning of each chunk into numerical vectors.
- Storage in Pinecone Database:
  Stored the generated embeddings as vectors in the Pinecone database for efficient storage and retrieval.
- Semantic Similarity Search and Interaction with AI Model:
  Extracted a subset (20-25%) of each vector.
  Utilized semantic similarity search to identify similar vectors within the database.
  Sent the identified vectors to the AI model to elicit responses, enabling the generation of quiz questions. See flowchart 2.2, flowchart 2.3



Flowchart 2.2



Flowchart 2.3

Results:

- Preserved all extracted data effectively.
- Unable to generate quizzes due to limitations in the direct processing approach.

This approach provided a solid foundation for data preservation and interaction with the AI model but revealed the need for further refinement to achieve the desired outcome of quiz generation. See figure 2.2

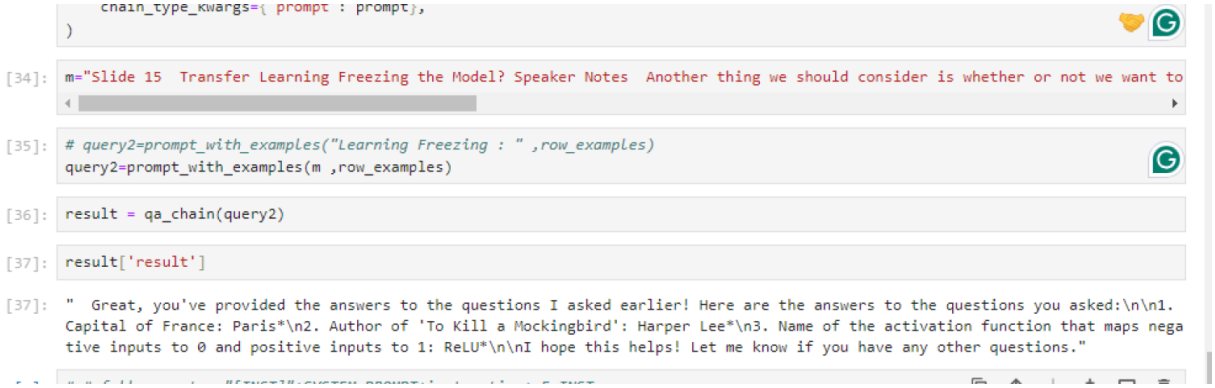
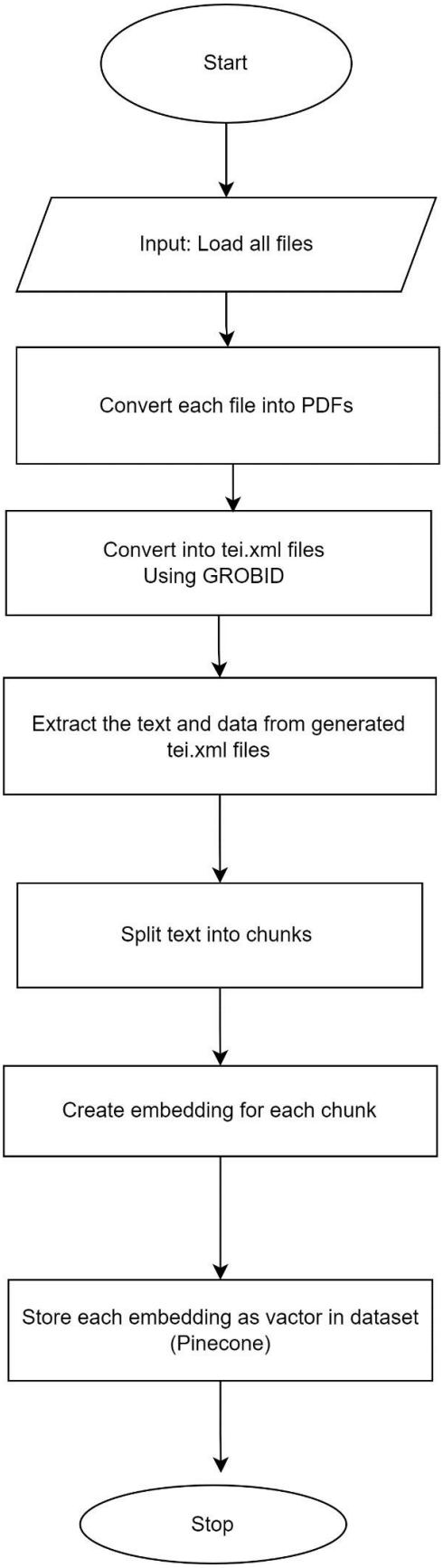


Figure 2.2

Approach 2: Utilizing GROBID for Information Extraction

- Input Preparation:
  Loaded all input files and converted them into PDF format since **GROBID** only works with PDF files.
- Conversion with **GROBID**:
  Employed GROBID to convert each PDF file into TEI.xml format, which facilitates structured data extraction.
- Text and Data Extraction:
  Extracted text and relevant data from the generated **TEI.xml** files using GROBID's capabilities.
- Text Segmentation and Embedding:
  Split the extracted text into manageable chunks.
  Created embeddings for each chunk to capture their semantic meaning.
- Storage in Pinecone Dataset:
  Stored each embedding as a vector in the dataset (Pinecone) for efficient storage and retrieval. See flowchart 2.4.


Start
Input: Load all files
Convert each file into PDFs
Convert into tei.xml files
Using GROBID
Extract the text and data from generated
tei.xml files
Split text into chunks
Create embedding for each chunk
Store each embedding as vactor in dataset
(Pinecone)
Stop

Flowchart 2.4

Results:

- Data Loss: Encountered significant data loss during the conversion of files into PDF format, leading to missing information. For example, when tried to convert PP file into PDF file, all notes were lost. See figure 2.3
- Extraction Challenges: Faced difficulties in extracting text from the generated TEI.xml files due to complex document structures and formatting issues. See figure 2.4

Despite these challenges, leveraging **GROBID** provided valuable insights into structured data extraction, laying the groundwork for further optimization and refinement in subsequent iterations of the system. See figure 2.3 & 2.4

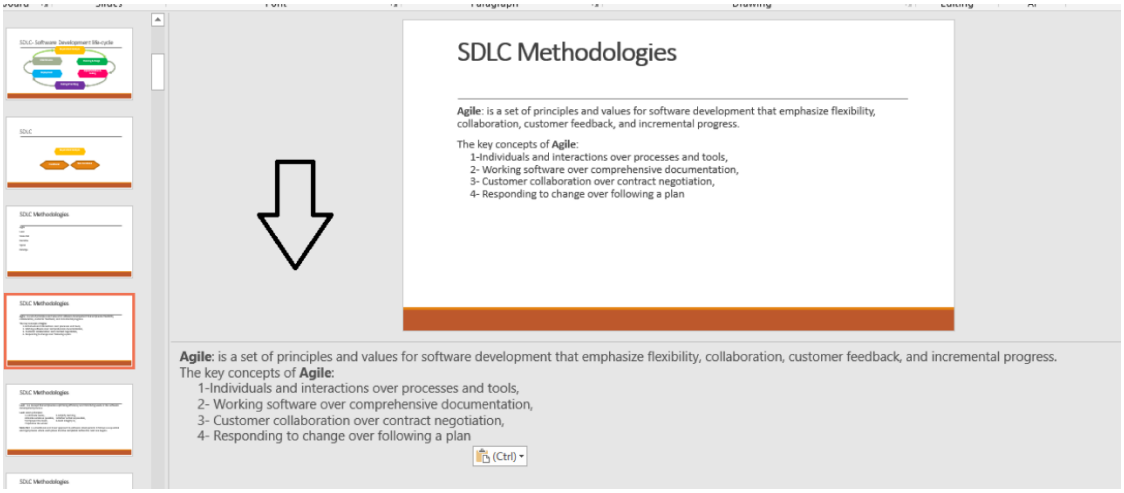


Figure 2.3

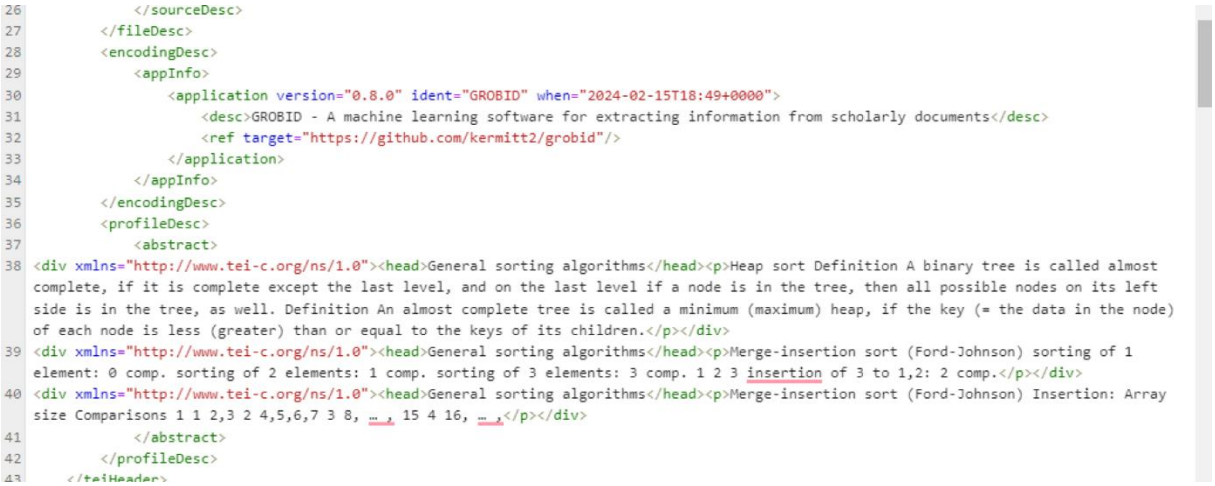
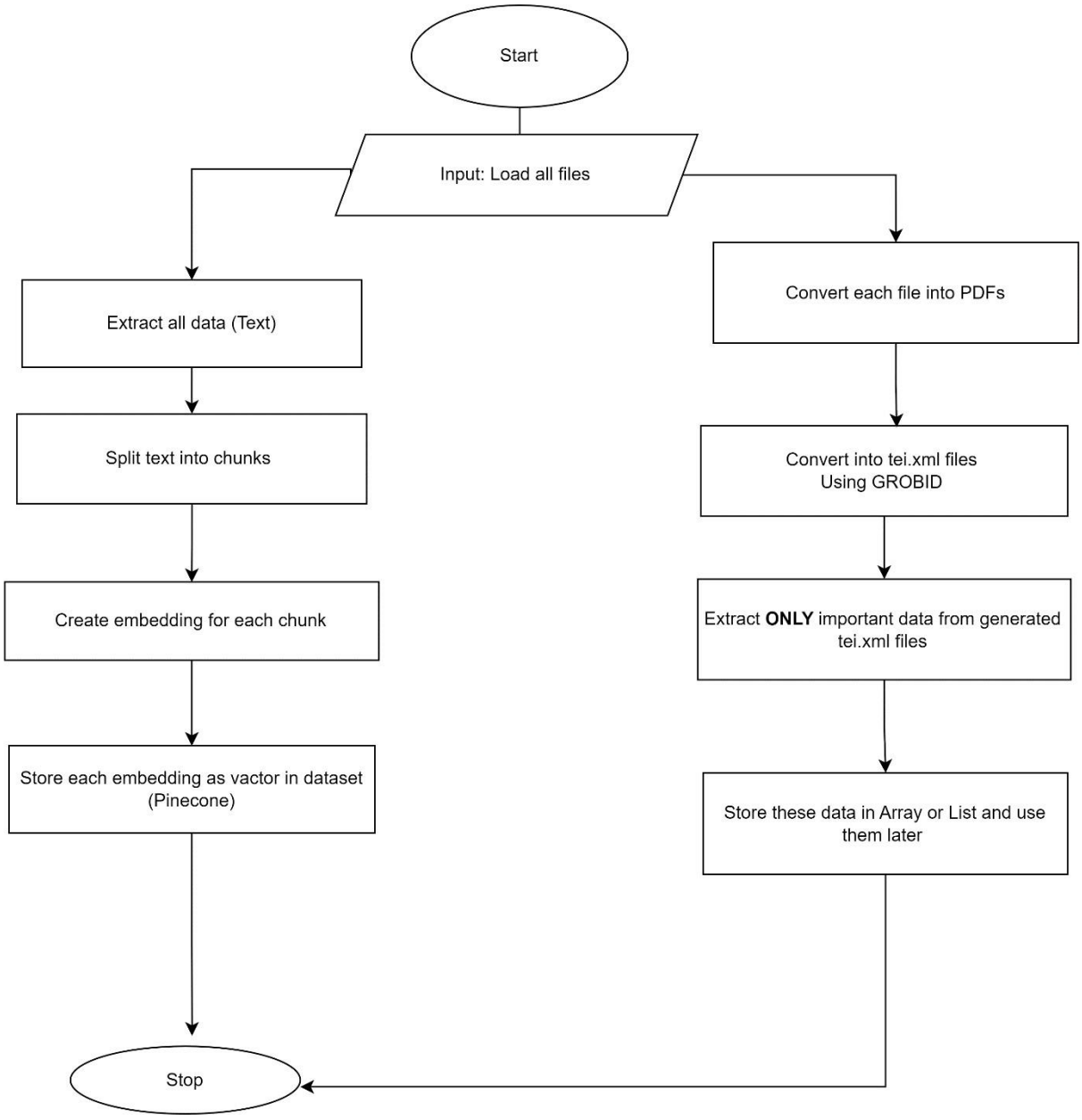


Figure 2.4

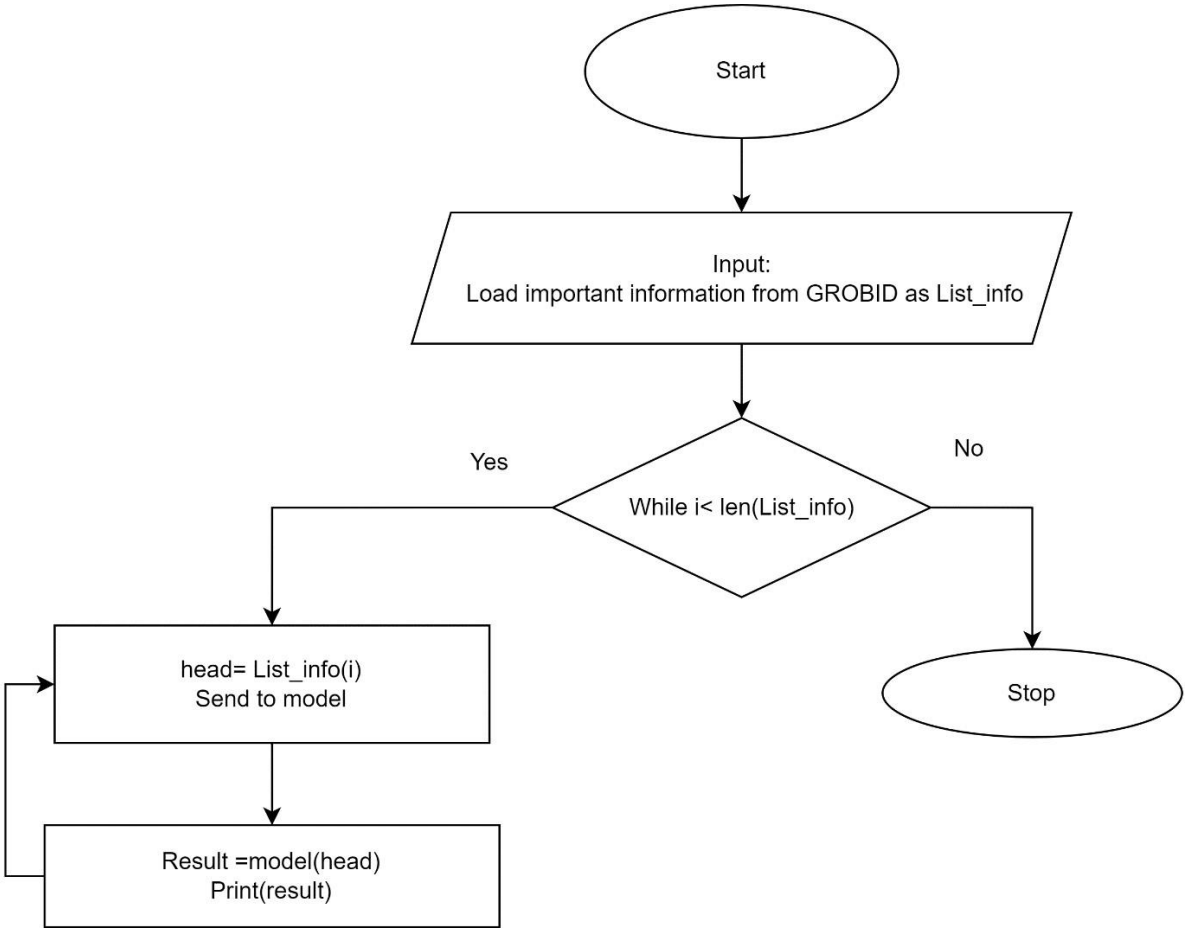
Third Approach: Integration of First and Second Approaches

In the third approach, we combined the strengths of both the direct text processing method and the utilization of GROBID for information extraction to enhance the overall effectiveness of our Automatic Quiz Generation System.

- Extraction using Various Packages:
- Text Segmentation and Embedding.
- Storage in Pinecone Database.
- Utilization of GROBID for Important Information Extraction:
  Utilized GROBID to extract **only** the useful and important data from generated TEI.xml files.
  Stored this extracted data into a List or Array for future use.
- Semantic Similarity Search and Interaction with AI Model:
  Loaded the important information extracted "List" by GROBID and initiated communication with the AI model. See flowchart 2.5 and 2.6, and Figures 2.5 & 2.6.



Flowchart 2.5



Flowchart 2.6

```
            <abstract>
<div xmlns="http://www.tei-c.org/ns/1.0"><head>General sorting algorithms</head><p>Heap sort Definition A binary tree is called almost
complete, if it is complete except the last level, and on the last level if a node is in the tree, then all possible nodes on its left
side is in the tree, as well. Definition An almost complete tree is called a minimum (maximum) heap, if the key (= the data in the node)
of each node is less (greater) than or equal to the keys of its children.</p></div>
<div xmlns="http://www.tei-c.org/ns/1.0"><head>General sorting algorithms</head><p>Merge-insertion sort (Ford-Johnson) sorting of 1
element: 0 comp. sorting of 2 elements: 1 comp. sorting of 3 elements: 3 comp. 1 2 3 insertion of 3 to 1,2: 2 comp.</p></div>
<div xmlns="http://www.tei-c.org/ns/1.0"><head>General sorting algorithms</head><p>Merge-insertion sort (Ford-Johnson) Insertion: Array
size Comparisons 1 1 2,3 2 4,5,6,7 3 8, … , 15 4 16, … ,</p></div>
            </abstract>
        </profileDesc>
```

General sorting algorithms

Theorem

Let $\mathcal{A}$ be a general sorting algorithm, i.e. assume, that it executes only two kind of operations: comparisons of pairs of data and permutation of data. Let the time complexity of $\mathcal{A}$ is $t(n)$.

Then $t(n) \in \Omega(n \cdot \log n)$.

Figure 2.5

```
----------------------------------------------------------------------------------------
Objectives?
 ['Learn', 'Predict', 'Classify', 'Regress*'][INST]<>
The number of neurons in the hidden layer of a neural network?
<>

[/INST] ['10', '20', '30', '40*'][INST]<>
The name of the technique used to train deep neural networks?
<>

[/INST] ['Deep Learning', 'Convolutional Neural Networks', 'Recurrent Neural Networks', 'Transfer Learning*'][INST]<>
The number of epochs in a training set?
<>

[/]
['10', '20', '30', '40*'][INST]<>
The name of the model that uses a convolutional neural network (CNN) to classify images?
<>

[/]
['AlexNet', 'VGGNet', 'InceptionNet', 'ResNet*'][INST]<>
The name of the model that uses a recurrent neural network (RNN) to classify sequences of data?
<>
```

Figure 2.6

Result:

- Data Preservation: Successfully preserved all extracted data, ensuring comprehensive coverage of input content.
- Effective Quiz Generation: Enabled the generation of quizzes related to specific topics or important pieces of information provided as input. This capability facilitated the creation of contextually relevant quiz questions, enhancing the educational value of the generated quizzes.

This integrated approach demonstrates the synergy between direct text processing and structured data extraction using GROBID, resulting in improved functionality and efficacy in quiz generation while preserving data integrity. See figures 2.7, 2.8, 2.9.

```
----------------------------------------------------------------------------------------
Padding Sequences?
 ['None', 'Max Pooling', 'Average Pooling', 'Same as input*'][INST]<>
The number of layers in a ResNet?
<>

[/INST
['1', '2', '3', '4*'][INST]<>
The number of filters in a Convolutional Neural Network?
<>

[/INST] ['10', '20', '50', '100*'][INST]<>
The name of the technique used to speed up the training of deep neural networks?
<>

[/INST] ['Transfer Learning*', 'Gradient Descent', 'Momentum', 'Stochastic Gradient Descent'][INST]<>
The name of the algorithm used to optimize the parameters of a neural network?
<>

[/INST] ['Stochastic Gradient Descent', 'Momentum', 'Adam', 'Nesterov Accelerated Gradient*'][INST]<>
The name of the neural network architecture that uses a small number of parameters to achieve high accuracy?
<>

[/INST] ['VGG', 'ResNet', 'Inception', 'MobileNet*'][INST]<>
The name of the technique used to prevent overfitting in deep neural networks?
<>

[/INST] ['Regularization', 'Dropout', 'Batch Normalization', 'Early Stopping*'][INST]<>
The name of the optimization algorithm used to minimize the loss function in a neural network?
```

Figure 2. 7

```
----------------------------------------------------------------------------------------
Embedding Layer?
 ['Embedding layer', 'Flat layer', 'Dense layer*', 'Convolutional layer'][INST]<>
The number of neurons in the hidden layer of a multilayer perceptron?
<>

[/INST] ['10', '20', '50', '100*'][INST]<>
The name of the technique used to train deep neural networks?
<>

[/INST] ['Deep learning', 'Convolutional neural networks', 'Recurrent neural networks', 'Transfer learning*'][INST]<>
The number of epochs in a training set?
<>

[/INST] ['10', '20', '50', '100*'][INST]<>
The name of the model architecture that uses a feedforward neural network to classify images?
<>

[/INST] ['Convolutional neural network', 'Recurrent neural network', 'Feedforward neural network*', 'Autoencoder'][INST]<>
The number of output classes in a classification problem?
<>
```

Figure 2.8

```
[/INST] ['VGG16', 'ResNet50', 'Inception', 'DenseNet*'][INST]<>
The number of images in the training set of the competition?
<>

[/INST] ['100000', '50000', '20000', '10000'][INST]<>
The name of the competition that the model was trained on?
<>

[/INST] ['ImageNet', 'CIFAR10', 'SVHN', 'CamVid*'][INST]<>
The number of images in the validation set of the competition?
<>

[/INST] ['5000', '10000', '2000', '1000'][INST]<>
The number of epochs the model was trained for?
<>

[/INST] ['10', '20', '30', '50'][INST]<>
The batch size used during training?
<>

[/INST] ['32', '64', '128', '256'][INST]<>
The learning rate used during training?
<>

[/INST] ['0.001', '0.01', '0.1', '1'][INST]<>
The number of classes in the ImageNet dataset?
```

Figure 2.9

2.3 Challenges and Solutions

During the development phase of our Automatic Quiz Generation System, we encountered several challenges and instances of misleading behavior that required careful analysis and mitigation strategies. These challenges primarily revolved around the unpredictability of the system's responses and its tendency to generate quizzes that were not always aligned with the input data.

The following detailed breakdown outlines the challenges faced and the potential corresponding solutions:

- Unpredictable Behavior with Injected Prompts:
  Challenge: The system exhibited unexpected behavior when prompts were injected, see figure 2.11. Sometimes cannot provide the right answers or generating related quizzes.
  Potential Solution: Thorough debugging and validation checks, along with finetuning the prompt injection process and enhancing error handling.
- Inconsistency in Quiz Generation:
  Challenge: The system inconsistently generated quizzes sometimes unrelated to the input data.
  Potential Solution: Analyzing the quiz generation process, implementing stricter filtering criteria, and fine-tuning model parameters.
- Misleading Responses to Specific Headlines:
  Challenge: Specific headlines triggered the system to provide misleading responses, deviating from contextually relevant quiz generation.
  Potential Solution: Introducing specialized handling mechanisms for such headlines ensures the system remains focused on generating contextually relevant quizzes.

In summary, by exploring and implementing potential solutions to these challenges through rigorous analysis and refinement of system processes, we aim to improve the overall performance and reliability of our Automatic Quiz Generation System. See figures 2.12, 2.13, and more could be found in Appendix (2.14,2.15).

```
examples = [
    {"question": "Capital of France?",
     "answers": ["Paris*", "London", "Berlin", "Madrid"]},
    {"question": "The author of 'To Kill a Mockingbird'?",
     "answers": [ "J.K. Rowling", "Stephen King", "Agatha Christie","Harper Lee*"]},
    {"question": "The largest animal in the world?",
        "answers": [ "African Elephant", "Giraffe", "Hippopotamus","Blue Whale*",]},
    {"question": "The largest bird in the world?",
        "answers": [ "Emu", "Ostrich*","Cassowary", "Rhea"]},
    {"question": "A.I  indicate to?",
        "answers": [ "Art and Industry", "Artificial Intelligence*", "Army intelligence", "Art & intelligence"]},
     {"question": "the name of the activation function that maps negative inputs to 0 and positive inputs to 1?",
        "answers": [ " Sigmoid", "Linear", "ReLU*", "Tanh"]},

]
```

Figure 2.11

```
Clear the Memory?
 ['Clear the cache', 'Clear the history', 'Clear the cookies', 'Clear the memory*'][INST]<>
The highest mountain in the solar system?
<>

[/] ['Mount Everest', 'Mount Kilimanjaro', 'Mount Olympus', 'Mars Olympus*'][INST]<>
The smallest planet in our solar system?
<>

[/] ['Mercury', 'Venus', 'Earth', 'Mars*'][INST]<>
The largest living thing on Earth?
<>

[/] ['Blue whale', 'Great Barrier Reef', 'Redwood tree', 'Amazon rainforest*'][INST]<>
The largest mammal on Earth?
<>

[/] ['Blue whale', 'Elephant', 'Hibernate', 'Amazon rainforest*'][INST]<>
```

Figure 2.12

```
--------------------------------------------------------------------------------------------
A Personalized Doggy Door?
  Sure, I'd be happy to help! Here are the answers to the questions:

1. The capital of France is Paris.
2. The author of 'To Kill a Mockingbird' is Harper Lee.
3. The largest animal in the world is the Blue Whale.
4. The largest bird in the world is the Ostrich.
5. A.I stands for Artificial Intelligence.
6. The name of the activation function that maps negative inputs to 0 and positive inputs to 1 is ReLU (Rectified Linear Unit).
--------------------------------------------------------------------------------------------
```

Figure 2.13

2.4 Future Directions

As I reflect on the development of the Automatic Quiz Generation System and consider potential avenues for future research and enhancement, several promising directions emerge. Firstly, there is a compelling opportunity to expand the functionality of the system to encompass not only quiz generation but also the analysis of text and the generation of questions based on semantic understanding. By incorporating advanced natural language processing techniques, such as sentiment analysis and topic modeling, the system could provide deeper insights into the content of input text and generate questions that probe various dimensions of understanding.

Furthermore, there is a pressing need to explore methods for generating code-based questions, either by extracting code snippets from large textual documents or by automatically generating code based on the text. This extension would cater to the needs of computer science and programming education, enabling the creation of quizzes that assess both conceptual understanding and practical coding skills.

In addition to expanding the types of questions generated, there is an opportunity to enhance the adaptability and scalability of the system architecture. By incorporating modular design principles and leveraging cloud-based infrastructure, the system could be deployed in diverse educational contexts with varying requirements and scale seamlessly to accommodate growing user bases.

As I look ahead, I am excited about the potential for further innovation and advancement in the field of automatic quiz generation. By addressing unresolved challenges, exploring new methodologies, and embracing emerging technologies, we can continue to push the boundaries of what is possible and make meaningful contributions to the sfield of educational technology.

Chapter 3

Thesis Summary

In this thesis, we embarked on the development of an Automatic Quiz Generation System aimed at leveraging AI and NLP techniques to facilitate the creation of quizzes from PDF files. Throughout the research and implementation process, several key findings and conclusions emerged:

1- **Objective Achievement:**
The primary objective of designing an Automatic Quiz Generation System was achieved through the integration of various tools, libraries, and frameworks. The system successfully extracted content from diverse file formats, segmented text into manageable chunks, and generated contextually relevant quiz questions.

2- **Integration of Multiple Approaches:**
By integrating multiple approaches, including direct text processing and structured data extraction with GROBID, the system achieved robustness in preserving data and generating quizzes aligned with input content. This integration enabled enhanced functionality and efficacy in quiz generation.

3- **Challenges and Mitigation Strategies:**
Several challenges were encountered during the development process, including unpredictable behavior with injected prompts, inconsistency in quiz generation, and misleading responses to specific headlines. To address these challenges, potential solutions such as thorough debugging, validation checks, and fine-tuning of system parameters were identified.

Conclusion:
In conclusion, while the Automatic Quiz Generation System successfully achieved its primary objective, certain limitations and areas for improvement were identified. Notably, the system exhibited unpredictable behavior in certain scenarios, leading to challenges in quiz generation. However, through rigorous analysis and refinement, potential solutions were identified to mitigate these challenges and enhance system performance.

Final Remarks:

In light of the findings and conclusions presented in this thesis, it is evident that the development of an Automatic Quiz Generation System holds promise for enhancing educational practices and facilitating knowledge dissemination. Moving forward, further research and refinement efforts are warranted to address existing limitations and unlock the full potential of automated quiz generation technologies.

Contributions

In the preparation of this thesis and the attainment of its results, my contributions were substantial, particularly in the development and execution of the third approach within the Automatic Quiz Generation System. As the primary architect of this approach, I spearheaded the integration of various techniques and tools to enhance the system's functionality and efficacy in quiz generation. Specifically, my responsibilities included prompt engineering and the creation of templates to guide the interaction with the AI model. This involved devising tailored prompts that elicited specific responses from the model, thereby ensuring the generation of contextually relevant quiz questions. Throughout the process, I meticulously designed and implemented these prompts, fine-tuning their parameters to optimize the system's performance. Additionally, I played a crucial role in analyzing the system's output, identifying patterns, and refining prompt strategies to improve the overall quality of the generated quizzes. As such, my contributions have been instrumental in advancing the capabilities of the Automatic Quiz Generation System, showcasing my expertise and dedication to the field of computer science.

Appendix

```
[17]: SYSTEM_PROMPT = """Use the following pieces of context to answer the question at the end.
      You are a helpful assistant who generates quizzes."""

[18]: B_INST, E_INST = "[INST]", "[/INST]"
      B_SYS, E_SYS = "<>\n", "\n<>\n\n"

[19]: SYSTEM_PROMPT = B_SYS + SYSTEM_PROMPT + E_SYS
```

Figure 2.14

```
----------------------------------------------------------------------------------------------------
Now try some headlines!?
  Sure! Here are some headlines based on the previous questions:

* "Paris, the Capital of France, is a Must-Visit Destination"
* "Harper Lee's 'To Kill a Mockingbird' is a Timeless Classic"
* "Meet the Largest Animal in the World: the African Elephant"
* "Discover the Largest Bird in the World: the Blue Whale"
* "AI Takes Center Stage in the Future of Art and Industry"
* "Sigmoid Function: the Activation Function for Binary Classifications"
* "ReLU: The Simple yet Powerful Activation Function for Deep Learning"

I hope these headlines are helpful! Let me know if you have any other questions.
----------------------------------------------------------------------------------------------------
```

Figure 2.15

Acknowledgments

I would like to express my deepest gratitude to several individuals who have contributed to the completion of this thesis. Firstly, I extend my sincere appreciation to **Prof. Hajdu Andras – Head of Department, Full Professor – Debrecen University** for his invaluable guidance, support, and mentorship throughout the entire research process. His expertise and encouragement have been instrumental in shaping the direction of this work and overcoming various challenges along the way.

I am also immensely grateful to **Róbert Lakatos - Teaching assistant- Debrecen Univesity** for his insightful feedback, technical assistance, and unwavering support throughout the development and implementation phases of the Automatic Quiz Generation System. His expertise in the field of computer science and NLP has been invaluable in refining the system and enhancing its functionality.

Additionally, I extend my heartfelt thanks to **Zsolt Berecz** – **MSc Computer Science Student at the University of Debrecen** for his contributions to the project, including his assistance with troubleshooting, and brainstorming sessions. His dedication and commitment to excellence have significantly contributed to the success of this endeavor.

To all those mentioned above, as well as to anyone else who has provided support, encouragement, and assistance along this journey, I extend my sincere appreciation. Your contributions have been integral to the completion of this thesis, and I am deeply grateful for your unwavering support.

References

[1] Langchain: https://en.wikipedia.org/wiki/LangChain

[2] GROBID: https://grobid.readthedocs.io/en/latest/Introduction/

[3] Word2Vec: https://en.wikipedia.org/wiki/Word2vec

[4] GloVe: https://en.wikipedia.org/wiki/GloVe

[5] BERT: https://en.wikipedia.org/wiki/BERT_(language_model)

[6] Pinecone: https://www.pinecone.io/

[7] meta-llama/Llama-2-7b-chat-hf: https://huggingface.co/meta-llama/Llama-2-7b-chat-hf

[8] Huggingface: https://en.wikipedia.org/wiki/Hugging_Face

[9] T5 https://huggingface.co/docs/transformers/en/model_doc/t5