T.R.

GEBZE TECHNICAL UNIVERSITY

FACULTY OF ENGINEERING

DEPARTMENT OF COMPUTER ENGINEERING

MULTIPLE SUBJECT DBQA WITH DOCUMENT ELIMINATION AND COMPARATIVE ANSWER UNIFICATION USING LLM

BURAK KOCAUSTA

SUPERVISOR
PROF. DR. YUSUF SINAN AKGÜL

GEBZE
2024

T.R.

GEBZE TECHNICAL UNIVERSITY

FACULTY OF ENGINEERING

COMPUTER ENGINEERING DEPARTMENT

MULTIPLE SUBJECT DBQA WITH DOCUMENT ELIMINATION AND COMPARATIVE ANSWER UNIFICATION USING LLM

BURAK KOCAUSTA

SUPERVISOR
PROF. DR. YUSUF SINAN AKGÜL

2024
GEBZE

GRADUATION PROJECT
JURY APPROVAL FORM

This study has been accepted as an Undergraduate Graduation Project in the Department of Computer Engineering on 25/01/2024 by the following jury.

JURY

Member
(Supervisor)   :   Prof. Dr. Yusuf Sinan Akgül

Member   :   Dr. Yakup Genç

ABSTRACT

Question Answering is one of the major tasks of Natural Language Processing. Generating answers for given document is one of the type which is called Document Based Question Answering System. With the improvement of LLM those systems become more advanced and useful. Most of the current systems uses LLM with Retrieval Augmented Generation, and they are answering single document. A single question may be asked to different documents and a combined answer that is short and informative for more than one document may be needed. The university senate, which tries to make decisions based on University Regulations, is an example of this situation.

In this project, current DBQA implementations improved with features like Document Classification, and Multiple Subject Question Answering. Retrieving related documents with given topic is the job of Document Classification step. Asking same question to different documents and generating a comparative, unified answer are the Multiple Subject Question Answering features. Those features integrated with current implementations using LLM. These features are integrated with existing approaches using the LLM and verified with the University Regulations example.

Keywords: Document Based Question Answering, Document Classification, Large Language Models, Retrieval Augmented Generation.

ÖZET

Soru Cevaplama Doğal Dil İşlemenin temel görevlerinden biridir. Verilen dokümana cevap üretmek, Doküman Tabanlı Soru Cevap Sistemi olarak adlandırılan türlerden biridir. Geniş Dil Modeli'nin gelişmesiyle birlikte bu sistemler daha gelişmiş ve kullanışlı hale gelmiştir. Mevcut sistemlerin çoğu, Retrieval Augmented Generation ile LLM'yi kullanarak tek bir belge üzerinden soru cevap işlemini gerçekleştirmektedirler. Tek bir sorunun farklı belgelere sorularak kısa ve birden fazla belge için bilgilendirici olabilecek birleşik bir cevaba ihtiyaç duyulabilir. Üniversite Yönetmelikleri üzerinden karar almaya çalışan üniversite senatosu bu duruma örnektir.

Bu projede, Doküman Sınıflandırma, Çoklu Özne Soru Cevaplama gibi özelliklerle mevcut Doküman Soru Cevap uygulamaları kullanılarak gelişmiş bir Soru Cevap Uygulaması geliştirilmesi hedeflendi. Verilen konuyla ilgili dokümanların ayrıştırılması, Doküman Sınıflandırma adımının işidir. Aynı soruyu farklı belgelere sormak ve karşılaştırmalı, birleşik bir yanıt oluşturmak Çoklu Özne Soru Cevaplama kısmının özelliğidir. Bu özellikler Geniş Dil Modeli kullanılarak mevcut uygulamalarla entegre edilip ve Üniversite Yönetmelikleri örneği ile test edilmiştir.

Anahtar Kelimeler: Doküman Tabanlı Soru Cevaplama, Doküman Sınıflandırma, Geniş Dil Modeli, Erişim Artırılmış Üretim.

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to my supervisor, Prof. Dr. Yusuf Sinan Akgül, for his guidance, support, and feedback throughout this project. His input has played a crucial role in shaping the progress of my project.

Burak Kocausta

LIST OF SYMBOLS AND ABBREVIATIONS

Symbol or Abbreviation	:	Explanation
DBQA	:	Document Based Question Answering System
LLM	:	Large Language Models
GPT	:	Generative Pre-trained Transformer
RAG	:	Retrieval Augmented Generation
NLP	:	Natural Language Processing
PDF	:	Portable Document Format
GUI	:	Graphical User Interface
CV	:	Coefficient of Variation
API	:	Application Programming Interface
REST	:	Representational State Transfer
JSON	:	JavaScript Object Notation

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

DBQA systems are greatly improved since the development of Large Language Models. Nowadays, there are many implementations of that task of NLP, most of them works accurately, and they are using same approach which is called Retrieval Augmented Generation.

Generating answer for multiple subjects is not a very common approach of question answering. Each document represents one subject. This subject could be institute, university, company, etc. Our aim is, generating a informative unified answer which compares all subjects. With this way, information from multiple documents (subjects) can be achieved for given topic.

Another addition to current DBQA systems is, Document Classification. Since we are generating answer for multiple documents, each document needs to be tagged with a subject name, and related documents with the topic must be retrieved. This step works before question answering loop starts. Documents are classified and tagged, then questions can be asked to those documents (subjects).

1.1. Project Description

In this project, Document Classification, Multiple Subject Question Answering, Single Document Question Answering parts are implemented, and integrated. Document format is determined as PDF. Document Classification part is performed before question answering starts, it classifies documents in given directory path whose format is PDF. Then those files are saved to database, and question answering can be made with those documents. A simple GUI is provided to use this implementation. University Student Regulation is used as main topic for the tests. A general scheme of the project is given with Figure 1.1.
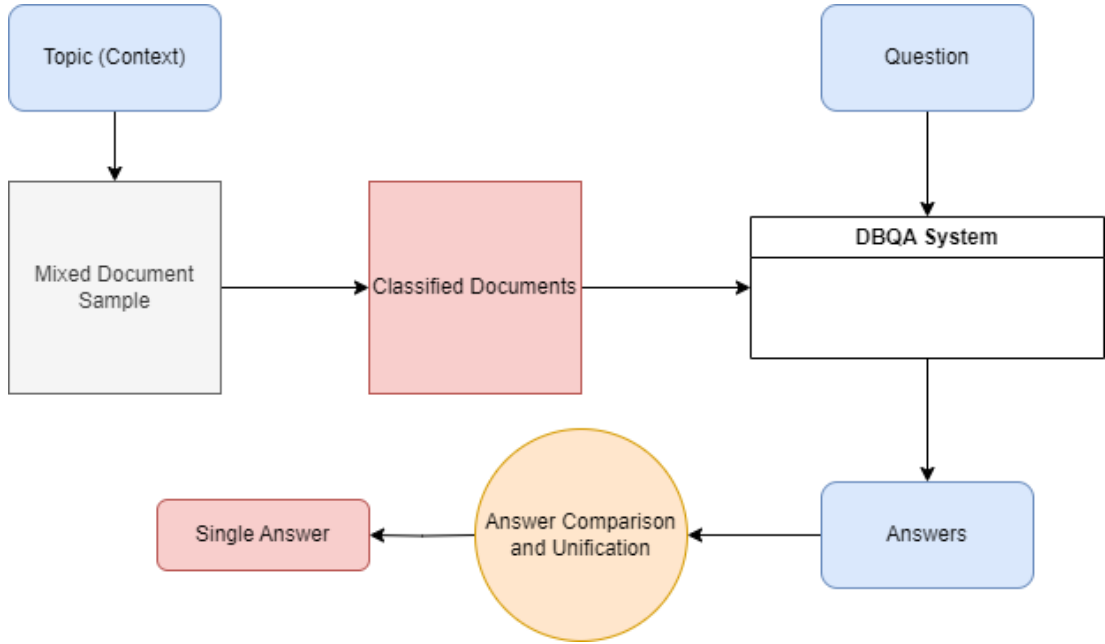
Figure 1.1: General Scheme of the Project.

1.2. Success Criteria and Objectives

There are 4 success criteria that must be satisfied in order to success this project. Also, simple GUI must be implemented to make a demonstration of the implementation.

1.2.1. Success Criteria

1.2.1.1. Response Time

During question answering loop, each the final unified answer must be generated below 45 seconds. Required time will be increase according to the number of documents, so it must satisfy this condition for at least 8 documents.

1.2.1.2. Linearly Scalable Cost

For one question answering loop, answer unification cost (LLM API cost ($)) must increase linearly with the number of documents. Average cost for each number of document must be increase as a line. Exponential growth is a failure as a scalability. Linear growth is expected.

1.2.1.3. Document Classification Success

From a given sample, documents must be successfully classified at least 80% success rate. This success rate should be calculated with geometric mean of how much of the target is found, and how much of the false documents eliminated.

$$S = \frac{\text{found}}{\text{target\_sample}} \times 100$$

S: Selection Percentage

$$E = \frac{\text{eliminated}}{\text{false\_sample}} \times 100$$

E: Elimination Percentage

$$\text{Accuracy} = \sqrt{S \times E}$$

1.2.1.4. Document Elimination Coefficient of Variation

Classification success from different tests shouldn't vary too much. Consistent response is needed. This consistency is measured with coefficient of variation. It must below 2%.

$$\text{Coefficient of Variation} = \frac{\text{standard deviation}(\text{success\_rates})}{\text{mean}(\text{success\_rates})} \times 100$$

1.2.2. Graphical User Interface

There must be a GUI for visualizing how the implementation works. It must be desktop application, and Document Classification, Multiple Document Question Answering, Single Document Question Answering features must be visualized with this interface. User can enter its question, select documents, enter topic and see retrieved documents.

2. LITERATURE REVIEW

2.1. DBQA Systems

In general, DBQA Systems works like, Semantic understanding of written sources with NLP, feature extraction, then answer is generated with Neural Network Models using dataset. With the analysis of questions, accurate queries are used for generating answers. Texts are generally analyzed with using word embeddings, then answers can be generated with using semantic embedding space. [1], [2]

With the improvement of LLM models like GPT, those models started to being used more for DBQA Systems.

2.2. Retrieval Augmented Generation

This is the most popular approach for generating an answer to a document. User makes a query, and documents, text data is stored in a vector database. Some of the data is retrieved with similarity search through embeddings, then those are used as prompt to LLM, and it generates the output. This is the basic (Naive) RAG. [3]–[5]

2.3. LLM Prompting

Prompting is important in order to LLM generate a better output. For example context must be described clearly, expected answer example can be provided in order to get the answer in accurate format. Any past data can be provided to make model more informed. [6], [7]

3. METHOD

3 part of the project implemented separately, then they are integrated. Document Classification, Single Subject DBQA, Multiple Subject DBQA are the 3 main steps of the project. As LLM, OpenAI API is used for both embedding, and text generation. In this chapter, those 3 method's architecture, and implementation details are described.

Final integration of 3 modules, and Graphical User Interface details are part of this chapter.

Table 3.1: Technologies Used

Category	Technology
Programming Language	Python
LLM Model	OpenAI gpt-3.5-turbo-1106 [8]
Embedding Model	OpenAI Text-embedding-ada-002-v2 [9]
Libraries	Langchain [4], PyQT5
Database	ChromaDB [10]

3.1. Document Classification

3.1.1. Approach and System Architecture

Aim of the classification algorithm is creating a set whose elements consist of filename, and tagged subject name who are separated from documents unrelated with the requested topic.

Algorithm assumes that heading of the document have information about its content. Most of the documents have short descriptive information about itself at their first characters. So, the classification, tagging can be done using fixed size of characters of each document. After extracting the text, the text can be send to LLM and an fixed output format can be requested. However, those extracted text might not be pre-processed well. So, in order to increase accuracy, extracted text is send to LLM separately with the purpose of acquiring well structured summary. With using those separate summaries, LLM can distinguish if document is related with given topic (context).

For the communication with LLM, different window sizes are used. Window

size means, at most how many document send for classification at each request. If more documents are sent with less requests, communication cost will decrease, and this could affect the accuracy of the output which is tested at the Experiments chapter. A general scheme of classification is provided with Figure 3.1.
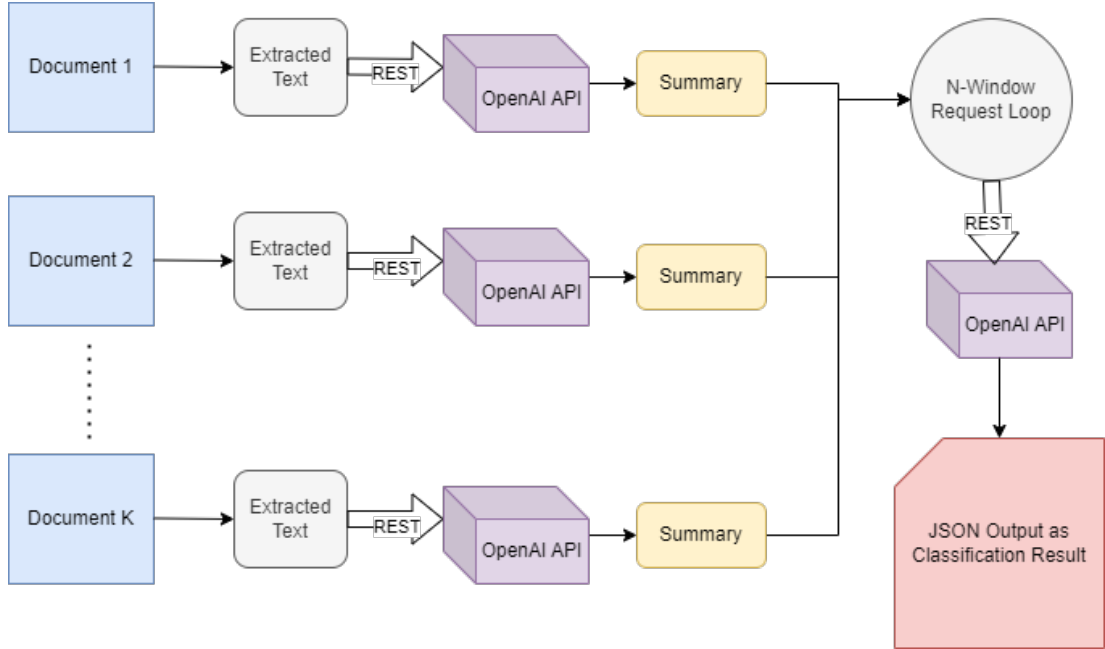
Figure 3.1: Classification Algorithm and Communication Scheme.

For the optimization, Extracting summaries part and generating classification result steps can be done concurrently. With a fixed size, summaries can be extracted from each API in parallel. Generating classification result step can also be done in parallel. They are completely separate jobs, therefore parallelization affects the performance.

3.1.2. Implementation Details

3.1.2.1. Implementation

Summaries are extracted concurrently, at most 5 thread runs in parallel. Each extracted text is provided summary function as an input. 3.2

```
with ThreadPoolExecutor(max_workers=CONC_REQUEST_MAX) as executor:
    futures = {executor.submit(process_pdf, os.path.join(path, file_name)):
                file_name for file_name in os.listdir(path) if file_name.endswith(".pdf")}

    for future in as_completed(futures):
        file_name = futures[future]
        try:
            s_text = future.result()
            files[file_name] = s_text
        except Exception as e:
            print(f"Error processing {file_name}: {e}")
```

Figure 3.2: Getting Summaries of Each Document.

Using those summaries, this time OpenAI API is used for getting classification result. Summary is given to the LLM and it generates output in JSON format. System prompt is important in this step (Figure 3.3). This step is also done concurrently. 5 document is tagged in parallel (Figure 3.4).

```
- You are a professional document classifier. You will classify similar documents
with looking small information about it for the given request. Input will be many
documents. Each consist of document name, and some part of its text. Analyze
documents 2 by 2.

Here are the important steps:
- Your job is detecting if document is related with given "request", and producing a
JSON format as an output.
- If the document is related to the "request", you provide the name of the
institution.
- If the document is not related with the request, your respond for it is "no".
- Your output will include all documents with their result.
- Output format is crucial; provide only the name or say "no."
- Your respond must be only JSON with given format. Do not ever make explanations.

Output format:
{
"DOC_NAME" : "YOUR_ANSWER"
}
```

Figure 3.3: Classification System Prompt.

```
with ThreadPoolExecutor(max_workers=CONC_REQUEST_MAX) as executor:
    futures = []

    for i in range(0, len(files), N):
        # concatenate all texts with file names
        content = ""
        for file_name, text in list(files.items())[i:i + N]:
            content += "\n" + file_name + ":\n" + text + "\n"

        # Submit the task to the thread pool
        future = executor.submit(ask_gpt_single, client, system_message, content)
        futures.append(future)

    for future in futures:
        try:
            content_result, prompt_tokens, completion_tokens = future.result()
            prompt_token += prompt_tokens
            completion_token += completion_tokens
            result.append(content_result)
        except Exception as e:
            print(f"Error processing task: {e}")
```

Figure 3.4: Getting Classification Result of Each Document.

3.1.2.2. Output

At the end, filenames and tags are mapped in a set. This set is the return of this classification algorithm. (Figure 3.5)

```
Context: "lisans eğitim öğretim yönetmeliği"

{'doc_04.pdf': 'GEBZE TEKNİK ÜNİVERSİTESİ', 'doc_06.pdf': 'Orta
Doğu Teknik Üniversitesi', 'doc_05.pdf': 'İZMİR YÜKSEK TEKNOLOJİ
ENSTİTÜSÜ', 'doc_09.pdf': 'İstanbul Teknik Üniversitesi',
'doc_10.pdf': 'KOÇ ÜNİVERSİTESİ', 'doc_13.pdf': 'Sabancı
Üniversitesi', 'doc_14.pdf': 'YILDIZ TEKNİK ÜNİVERSİTESİ',
'doc_01.pdf': 'BOĞAZİÇİ ÜNİVERSİTESİ'}
```

Figure 3.5: Example Result of Document Classification.

3.2. Single DBQA

3.2.1. Approach

Single DBQA part is implemented as mentioned in the Backround chapter. Retrieval Augmented Generation approach is used. Inorder to do this, each document must be stored in vector database. This is done using OpenAI Embeddings API. Semantic similarity search is done to the database, and related documents are retrieved whose number is set by the programmer. Those documents retrieved according to the cosine similarity between question and text chunks. Chunk size is 1000. Retrieved document number is 6. Then those documents, and the question are sent to the LLM to get the answer.

3.2.2. Implementation Details

3.2.2.1. Implementation

Documents are stored on ChromaDB vector database as 1000 chunks. (Figure 3.6)

With the question related documents retrieved and stuffed. Then answer is generated with RAG approach. (Figure 3.7)

```
# Load with UnstructuredPDFLoader
loader = UnstructuredPDFLoader(path)
documents = loader.load()
text_splitter = RecursiveCharacterTextSplitter(chunk_size=1000, chunk_overlap=0)
splits = text_splitter.split_documents(documents)

# Create Chroma database
Chroma.from_documents(
    documents=splits,
    embedding=OpenAIEmbeddings(api_key=api_key),
    persist_directory=chroma_db_path,
)
```

Figure 3.6: Database Creation.

```
# Load the existing Chroma database
vectorstore = Chroma(persist_directory=full_path,
                     embedding_function=embedding_function)
retriever = vectorstore.as_retriever(search_type="similarity", search_kwargs={"k": 6})

llm = ChatOpenAI(model_name="gpt-3.5-turbo-1106", api_key=api_key, temperature=0)

qa = RetrievalQA.from_chain_type(llm=llm,
                                  chain_type="stuff", retriever=retriever, return_source_documents=True)
answer = qa({"query" : question})

# get 'result' from answer json
result = answer["result"]
```

Figure 3.7: Generating Single Answer to Single Question.

3.2.2.2. Output

Some of the retrieved document parts for the question (Figure 3.8, and generated answer (Figure 3.9) as follows.

```
filename: gtu_regulation.pdf
Enter the question: onur öğrencisi olarak mezun olabilmek için not ortalaması kaç
olmalıdır? şartları nelerdir?


(2) Ağırlıklı genel not ortalaması, öğrencinin almış olduğu tüm derslerin
kredilerinin 4'lük sistemdeki notları ile çarpılması sonucu elde edilecek sayılar
toplamının, başarı notu EX, NC, S, U olan dersler hariç krediler toplamına
bölünmesi ile bulunacak değerdir. Bölme sonucu, virgülden sonra iki hane yürütülüp
yuvarlatılarak verilir. Çift ana dal veya yan dal programından çıkarılan
öğrencilerin başarılı olduğu ve ana dal programına sayılmayan dersleri, genel not
ortalamasına dâhil edilmeksizin transkript ve diploma ekinde gösterilir.


Başarı durumu

MADDE 31- (1) Mezuniyet sırasında genel not ortalaması 3,00-3,49 arasında olan
öğrenciler onur öğrencisi, 3,50 ve üstü olanlar ise yüksek onur öğrencisi olarak
mezun olurlar ve bu durum transkriptlerinde belirtilir.

(2) Bir yarıyıl sonunda, ağırlıklı genel not ortalaması 2,00 veya üstü olan
öğrenciler başarılı olarak değerlendirilir.
-------------------------------------------------
Mazeret sınavı

MADDE 29- (1) Haklı ve geçerli nedenlerle sınavlarına giremeyen öğrenciler, ilgili
yönetim kurulu kararlarıyla aynı yarıyıl içinde dersi veren öğretim elemanının
belirleyeceği gün ve saatte kullanılmak üzere mazeret sınav hakkı verilir.
Öğrencilerin mazeret sınavına girebilmeleri için sınav tarihinden itibaren 5 iş
günü içerisinde mazeret dilekçelerini kayıt oldukları bölümün başkanlığına teslim
etmeleri gerekir.


Not ortalamaları
```

Figure 3.8: Example Retrieved Document Parts.

```
Onur öğrencisi olarak mezun olabilmek için genel not ortalamasının 3,00-3,49
arasında olması gerekmektedir. Ayrıca, mezuniyet şartlarını tamamlamak ve kayıtlı
olduğu programdaki toplam kredi yükünün en az yarısını GTÜ'de almış olmak da
gereklidir.
```

Figure 3.9: Example Generated Answer.

3.3. Generating Comparative and Unified Answer from Multiple Answers

3.3.1. Approach and System Architecture

Aim is asking single question to multiple subjects, and after retrieving the answers, processing those in order to generate unified, comparative final answer.

Single subject DBQA is needed for each document. Each one generates an answer for their respective document. Those answers are provided to the OpenAI, and single answer request is prompted to it as a system message. It generates the final answer according the question, and generated answers. (Figure 3.10)

To improve performance, parallelization is needed. Asking Single DBQA jobs are completely separate from each other, therefore they can be parallelized in order to improve performance of question answering loop.
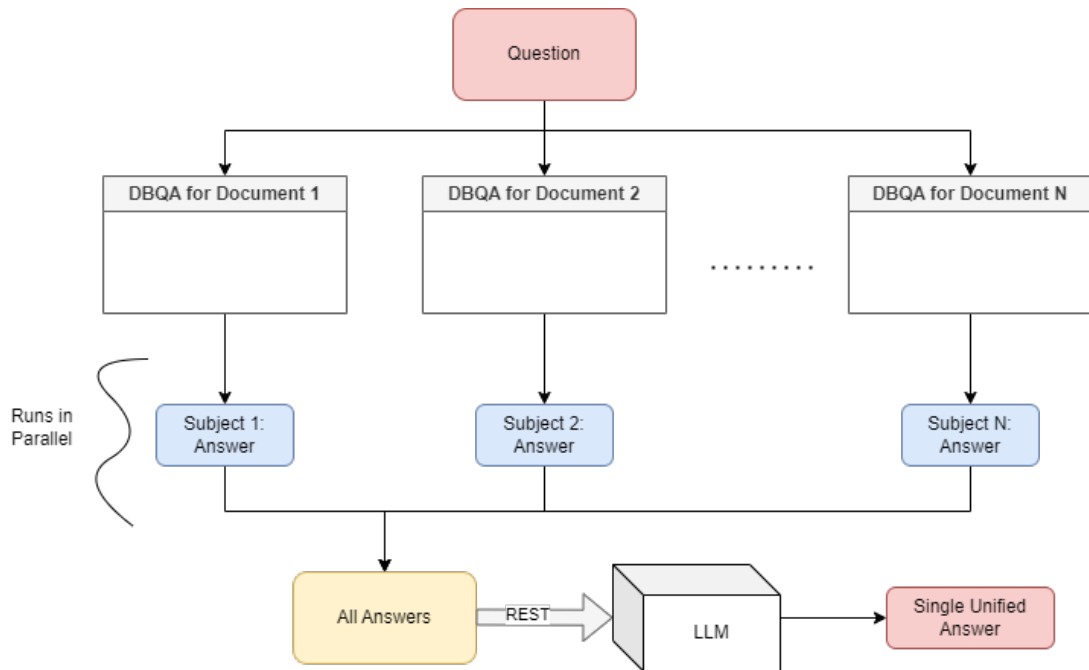
Figure 3.10: Generating Single Answer from Multiple Answers Scheme.

3.3.2. Implementation Details

3.3.2.1. Implementation

Single DBQA Module is used in order to implement this step. Every answer generated in thread pool. (Figure 3.11)

In order to make LLM to generate an answer same format, prompting is needed. With prompt, subject names, and answers it generates the final answer. (Figure 3.12)

```
with concurrent.futures.ThreadPoolExecutor(max_workers=CONC_MAX) as executor:
    future_to_subject = {executor.submit(worker_thread, file_name, question, subject):
                        subject for file_name, subject in subject_source.items()}
    for future in concurrent.futures.as_completed(future_to_subject):
        answers.append(future.result())

return answers
```

Figure 3.11: Implementation of Getting answers from Single DBQA modules.

```
Your job is generating a unified answer from different answers which were given to
the same questions from different subjects. Your aim is generating simplified and
categorized answer. Your answer shouldn't be long.

Here are the steps:
- You will be provided with answers, each answer starts with its subject.
- Clearly mark similarities and differences with a focus on specific topics or
entities.
- Provide very brief explanations for each identified topic difference.
- For the comparison step, give subject name when pointing differences and
similarities.
- Keep explanations concise, emphasizing key distinctions.
- Conclude with a concise summary.

Stick to the given format. Here is the example:

Similarity:
[YOUR_ANSWER]

Differences:
[YOUR_ANSWER]

Summary:
[YOUR ANSWER]
```

Figure 3.12: Prompt for LLM to generate Unified Answer.

3.3.2.2. Output

An example question is asked to 4 documents, and answers are generated. (Figure 3.13)

With those answers, a unified answer is generated. (Figure 3.14)

```
Question: Yabancı dil yeterliliği nasıl test edilir?

Subject: BOĞAZİÇİ ÜNİVERSİTESİ
Yabancı dil yeterliliği, Boğaziçi Üniversitesi'nde Yabancı Diller Yüksekokulu
tarafından önerilerek Senato tarafından kabul edilen bir dil sınavı ile belgelenir.
Bu sınavı başarıyla tamamlayan öğrenciler ilgili programa kayıt yaptırabilirler.


Subject: Istanbul Technical University
Yabancı dil yeterliliği, Üniversite tarafından düzenlenen ve tarihi ilan edilen
İngilizce yeterlik sınavıyla test edilir. Ayrıca, Senato tarafından eşdeğerliği
kabul edilen ulusal ve uluslararası sınavlarda belirlenen düzeyde başarılı olanlar
da yeterliliklerini belgeleyebilirler. Başvuran öğrencilerin durumlarına göre farklı
test ve belgelendirme yöntemleri bulunmaktadır.


Subject: Middle East Technical University
Yabancı dil yeterliliği, Orta Doğu Teknik Üniversitesi'nde lisans programlarına
yerleştirilen, kabul edilen veya kurumlar arası yatay geçiş yapan öğrenciler için
TİB Hazırlık Sınıfında öğretim yürürlükteki "Orta Doğu Teknik Üniversitesi Yabancı
Diller Yüksekokulu Temel İngilizce Bölümü Hazırlık Sınıfı Yönetmeliği" hükümlerine
göre belirlenir. Bu yönetmelik kapsamında belirlenen sınavlar ve koşullar ile
yabancı dil yeterliliği test edilir.


Subject: Gebze Technical University
Yabancı dil yeterliliği, Üniversite tarafından düzenlenen ve tarihi ilan edilen
İngilizce seviye tespit sınavıyla test edilir. Ayrıca, Senato tarafından eş
değerliği kabul edilen ulusal ve uluslararası sınavlarda başarılı olan öğrenciler,
ayrı bir İngilizce seviye tespit sınavına girmeden ön lisans veya lisans öğrencisi
olabilirler. Bu sınavlarda başarısız olan veya bu sınavlara katılmayan öğrenciler
ise, GTÜ Yabancı Diller Bölümünde İngilizce hazırlık öğretimine devam ederler.
```

Figure 3.13: Answers from different subjects.

```
Unified answer:

Benzerlik:
Öğrenciler yabancı dil yeterliliğini belgelendirmek için üniversitelerinde
düzenlenen sınavlar veya ulusal/uluslararası sınavlara katılarak yeterliliklerini
kanıtlamaktadırlar.

Farklılıklar:
- Boğaziçi Üniversitesi: Yabancı dil yeterliliği için Senato tarafından belirlenen
bir dil sınavı ile test edilir, başarılı olan öğrenciler ilgili programa
kaydolabilir.
- Istanbul Technical University: İngilizce yeterlik sınavı veya eşdeğer ulusal/
uluslararası sınavlarda başarılı olanlar yeterliliklerini belgeleyebilirler.
- Middle East Technical University: TİB Hazırlık Sınıfında belirlenen sınavlar ve
koşullar ile yabancı dil yeterliliği test edilir.
- Gebze Technical University: Üniversite tarafından düzenlenen İngilizce seviye
tespit sınavı veya ulusal/uluslararası sınavlarda başarılı olmak yeterli olabilir.

Özet:
Yabancı dil yeterliliği, üniversitelerde belirlenen dil sınavları veya ulusal/
uluslararası sınavlar aracılığıyla test edilmektedir. Her üniversite kendi
belirlediği yöntemle yeterliliği değerlendirmektedir.
```

Figure 3.14: Example Unified Answer.

3.4. Integration of Modules

So far 3 different modules are implemented. Document Classification, Single DBQA, Multiple DBQA. At the end, those modules are integrated, and GUI is built on top of this module. Single DBQA modules database creation part is used by Document Classification Module, other part is used by Multiple Answer Generation Module. (Figure 3.15)
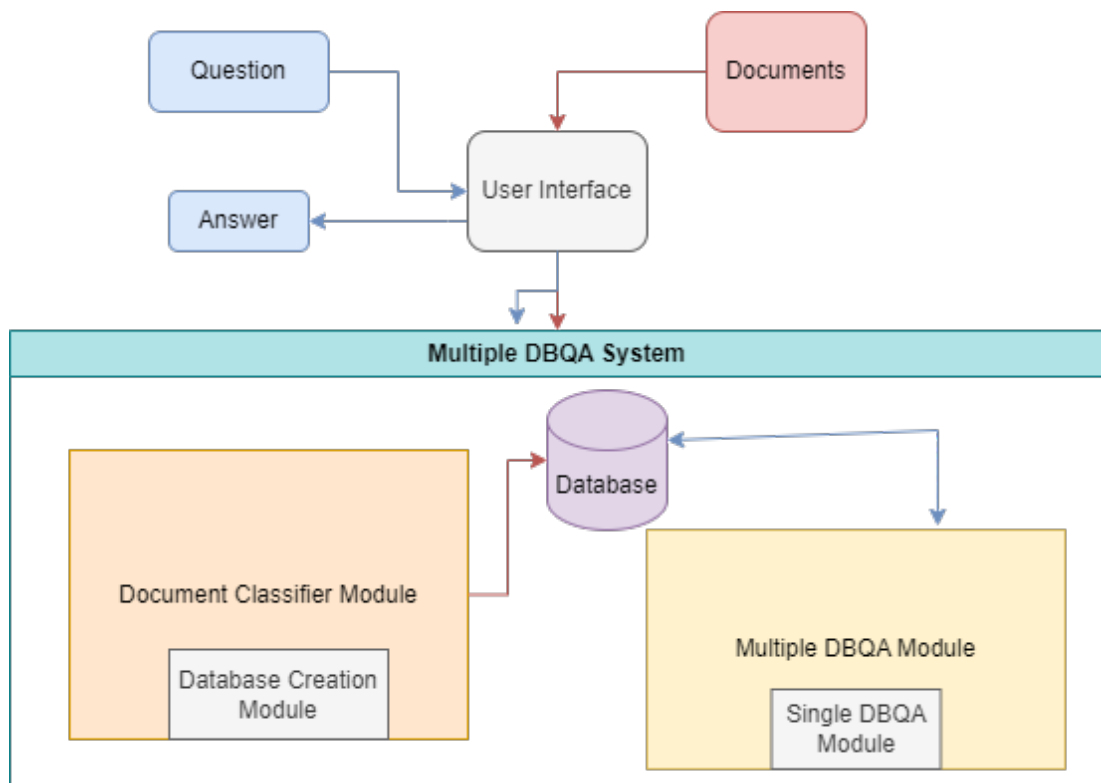

Figure 3.15: Module Hierarchy and Access Patterns via User Interface.

3.5. Graphical User Interface

GUI is implemented with PyQT5. There are 3 pages, main screen (3.16), classification screen(3.17), question answering screen(3.18).

Directory path is selected from browse button, classification is made after topic is entered(3.19).

Before starting asking question, topic and subject must be selected (3.20). It can be made with single subject (3.21). It can be made with multiple subject after selecting, "All Subjects".(3.22).
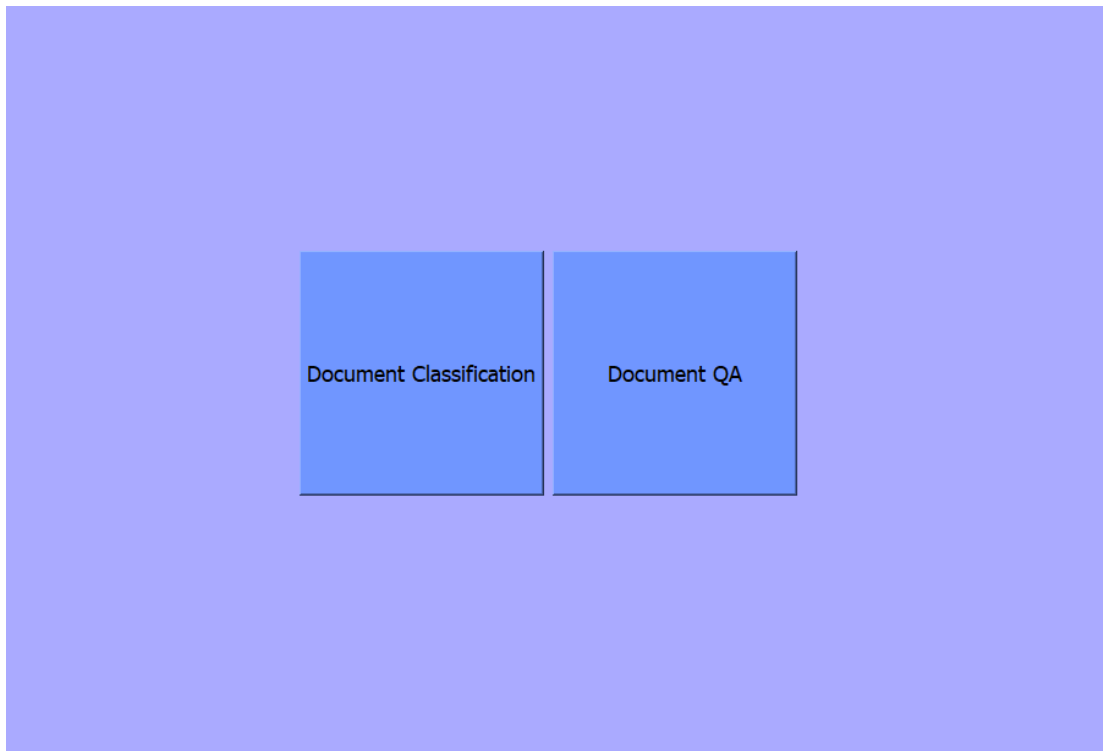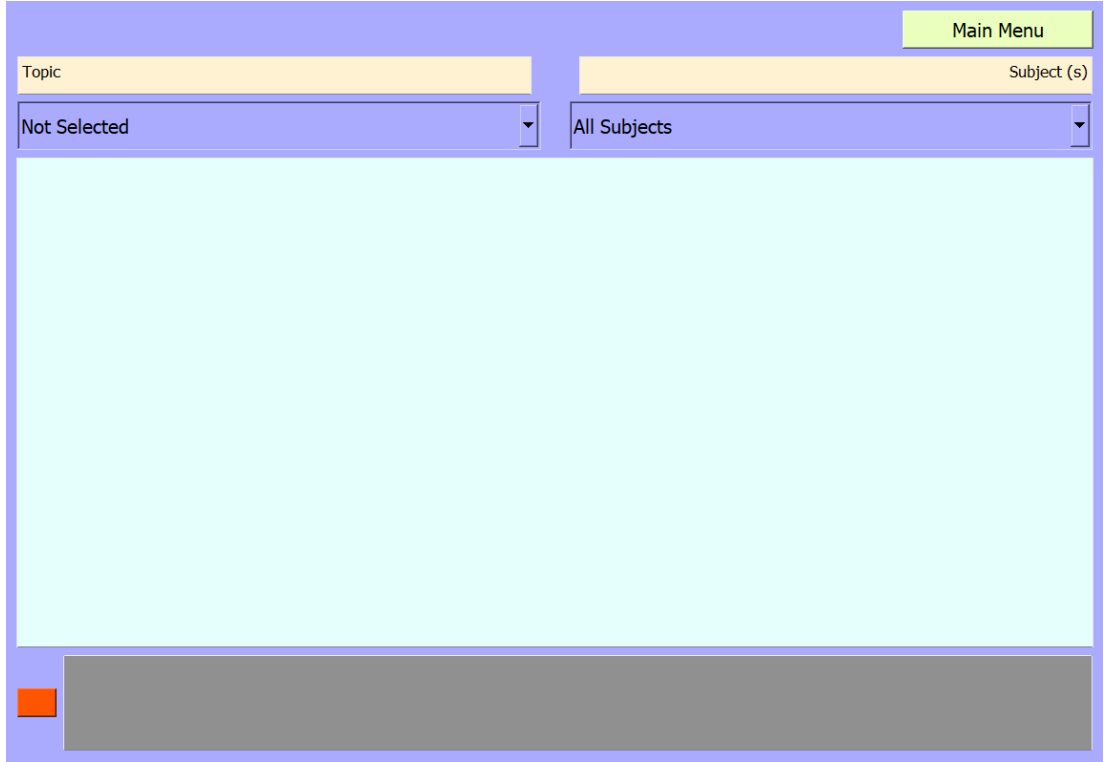
Figure 3.16: Main Screen
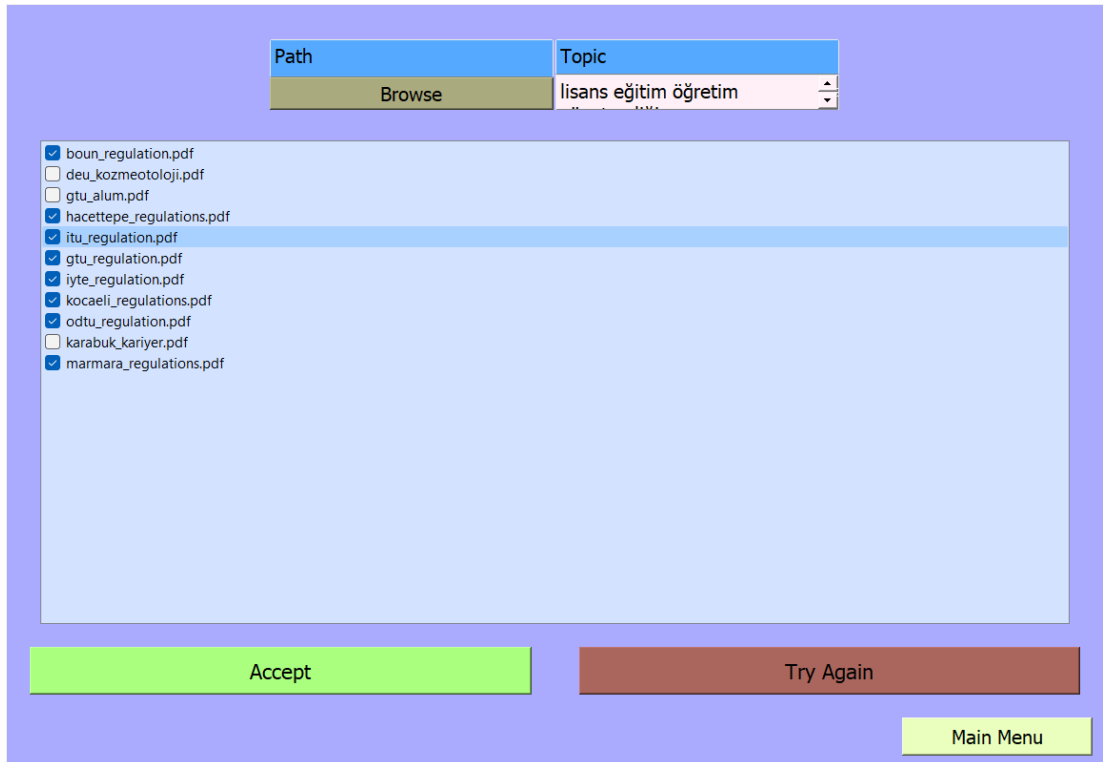
Figure 3.17: Classification Screen

Figure 3.18: Question Answering Screen

Figure 3.19: Example Classification Result

Figure 3.20: Selecting Subject and Topic

Figure 3.21: Single DBQA

Figure 3.22: Unified Answer

4. EXPERIMENTS

There are 4 success criteria. How they are evaluated, and results of each are described. Also, additional examples with GUI are given.

4.1. Evaluation of Success Criteria

4.1.1. Response Time

Response time of question answering loop must below 45 seconds.

4.1.1.1. Evaluation Method

It is tested with 2, 3, 4, 5, 6, 7, 8, 12, 16 different document packets. 15 different question is prepared (Figure 4.1), for each document packets, 15 question is asked to each packet. Each 15 question is asked 2 times. Therefore, 1890 single answers are generated. 270 unified answers are generated as a final answer. (Figure 4.2)


```
Mezuniyet süreci nasıldır?
Not ortalaması nasıl hesaplanır?
Onur öğrencisi olarak mezun olabilmek için ne yapmak gerekir?
Yabancı dil yeterliliği nasıl test edilir?
Staj gerekliliği nedir?
Ders seçimindeki kısıtlamalar nelerdir ve öğrenciler nasıl ders değişikliği yapabilir?
Akademik başarı ödülleri ve burslar için başvuru süreci nedir?
Tek ders sınavına girebilmenin şartları nelerdir?
Akademik ihlaller ve etik kurallara uymayan davranışların sonuçları nelerdir?
Ders seçimi süreci nasıl gerçekleşir?
Mezuniyet törenine katılım için gerekli olan prosedürler nelerdir?
Sınavlara itiraz için ne yapmak gerekir?
Değişim programı süreci nasıldır?
Burs imkanı var mıdır?
Okul barınma imkanı sağlar mı?
```

Figure 4.1: Question List

```
for question in questions:
    start_time = time.time()
    answer, cost = qa_api.ask_question_test(question, topic, "All Subjects")
    end_time = time.time()
    times.append(end_time - start_time)
    costs.append(cost)
```

Figure 4.2: Test Code for Question Answering Loop

4.1.1.2. Evaluation Results

All document packets response time with 16 document and upper bound. (Figure 4.3)

Averages of each document with upper bound. (Figure 4.4)

Some of the evaluations exceed the 45 seconds limit as it can be seen on table (Figure 4.3), but most of them is lower than the 45 seconds bound. For averages, none of the averages exceed the limit, but 16 document is quite close to the limit. After 16 document, it is expected that response time will become more than the limit. It satisfies the criteria with at most 16 documents.
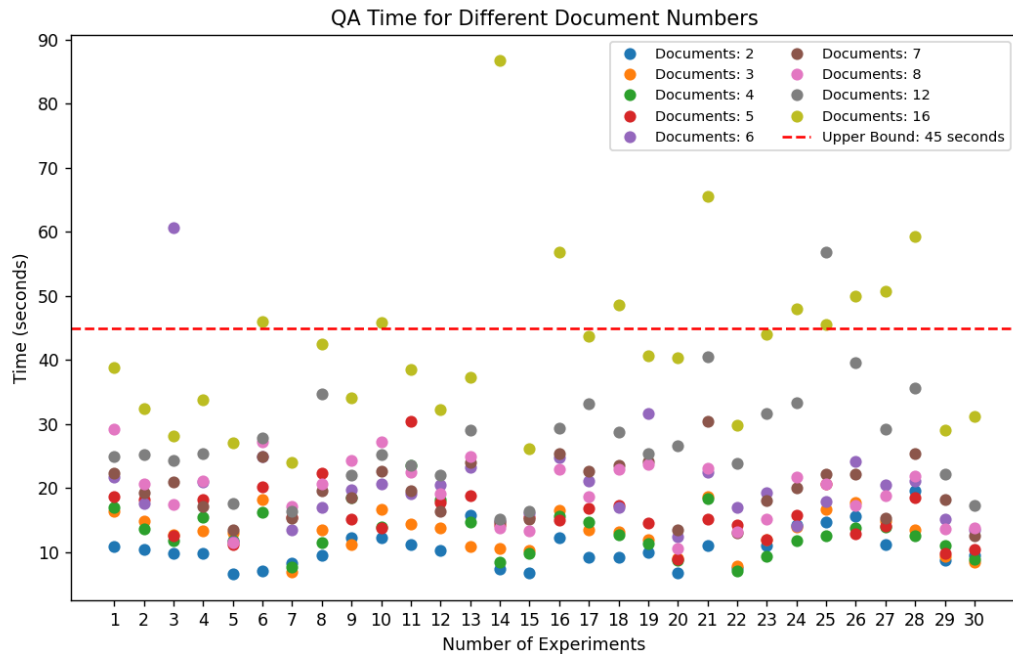
Figure 4.3: Response Time of All Questions

Table 4.1: Average Times for Each Number of Documents

Number of Documents	Average Time (seconds)
2	10.65
3	13.14
4	13.19
5	15.99
6	20.49
7	19.71
8	19.63
12	27.45
16	41.90
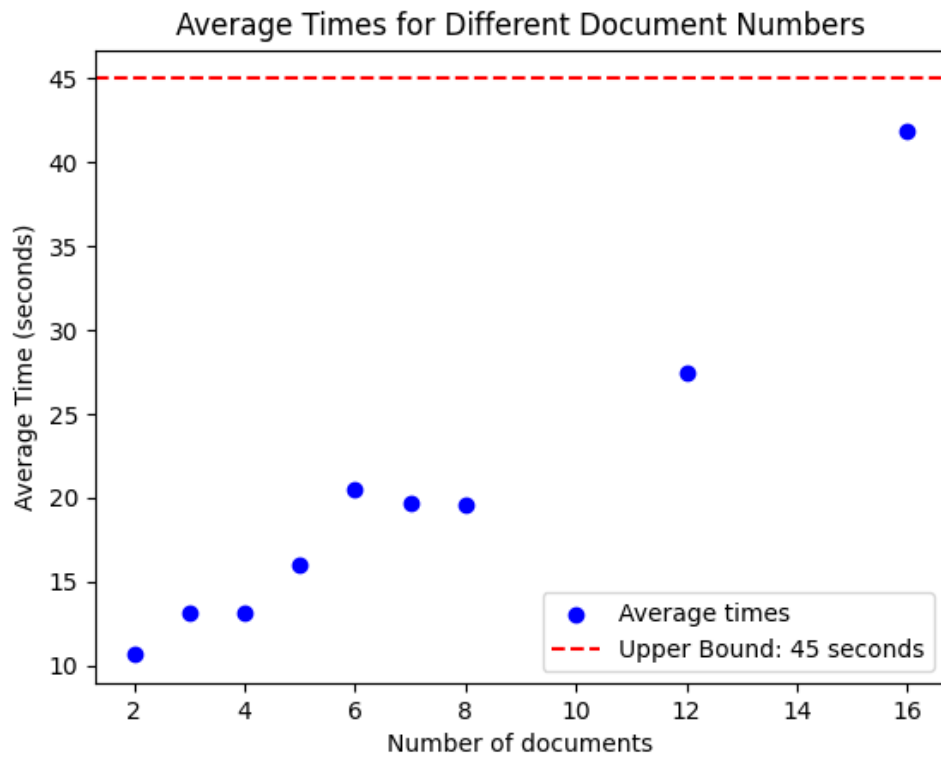
Figure 4.4: Average Response Times

4.1.2. Linearly Scalable Cost

For answer unification step, cost must increase linearly with number of documents. There shouldn't be exponential increase. It is indicated with a line.

4.1.2.1. Evaluation Method

Scalability is evaluated with the same data as the previous test. (4.1, 4.2) Cost of answer unification part is measured.

4.1.2.2. Evaluation Results

Figure 4.5 shows that, for each document there could be different cost results. But all of them increase linearly as if it is a line.
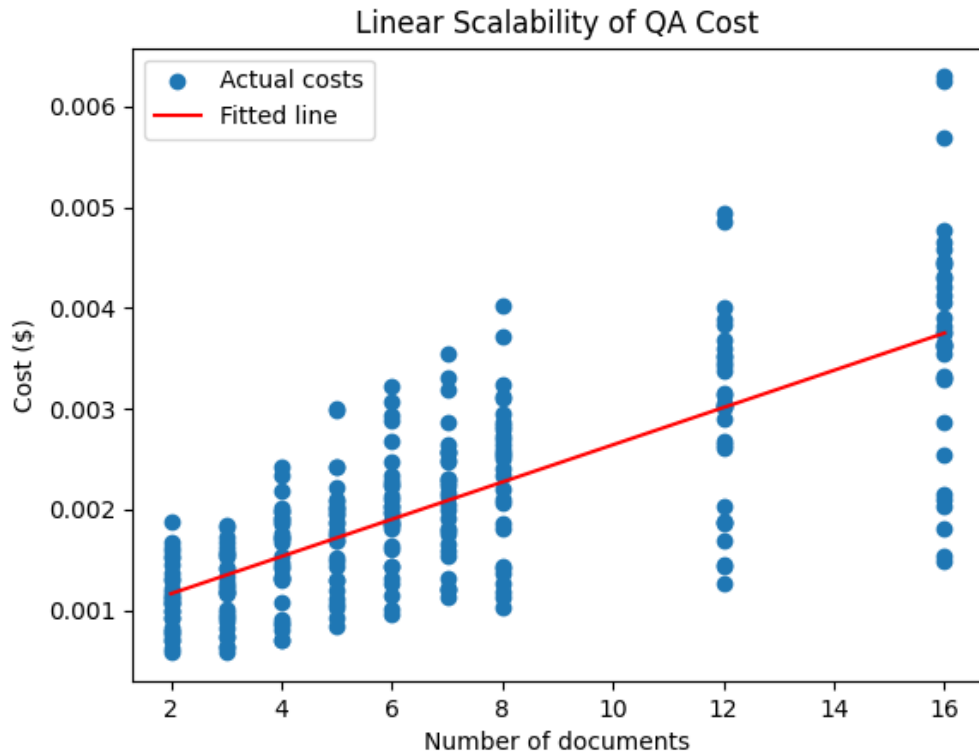
Figure 4.5: Cost of Each Question

Table 4.2: Number of Documents and Average Costs

Number of Documents	Average Cost ($)
2	0.0011
3	0.0012
4	0.0015
5	0.0018
6	0.0020
7	0.0021
8	0.0023
12	0.0030
16	0.0037

Figure 4.6 shows that average results are quite close to linear increase. All dots are around the fitted line.
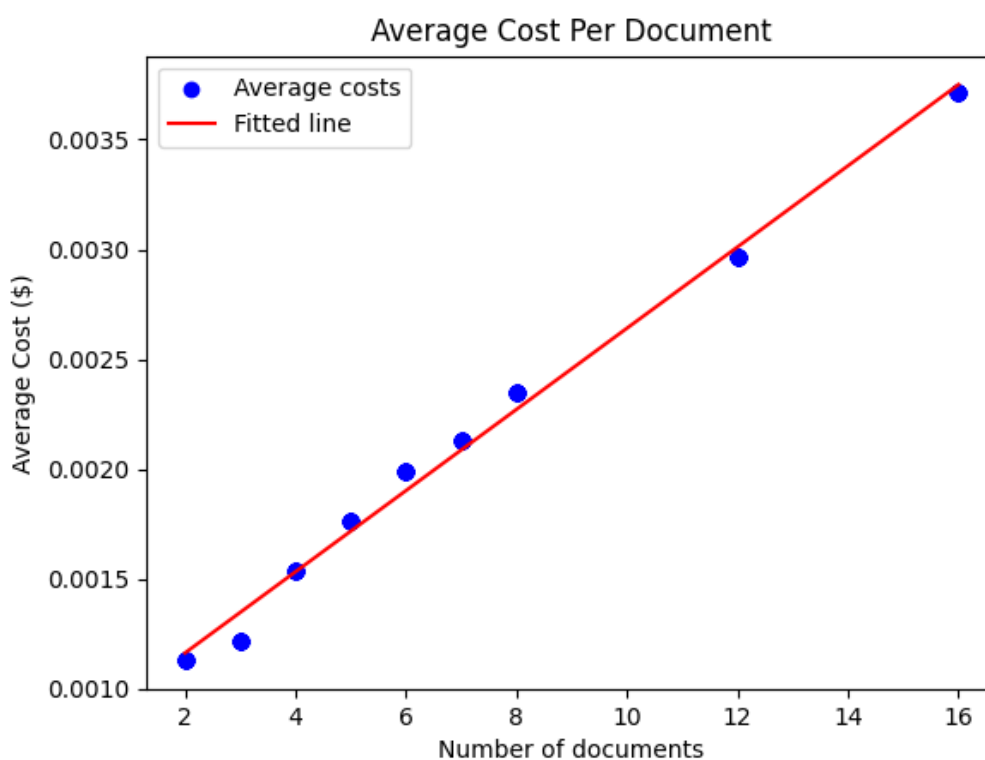

Figure 4.6: Average Costs

4.1.3. Document Classification Success

Document Classification Success means, with given document sample successful classification percentage must be at least 80%. It is tested with different window sizes. Explanation of how success rate calculated is explained at **Introduction Chapter**.

Document Classification Success is tested with different window sizes. Window size means, how many summarized document information are asked to LLM in single request. Those values are 1, 2, 4. Higher window means, we are sending less request, so cost is decreasing. So higher window is preferable.

This sample is classified 10 times, for each window size. Classification test is made 30 times on this sample. Documents are shown at Figure 4.7.

Code that runs classification is Figure 4.8, and expected output from given sample is given at Figure 4.9.

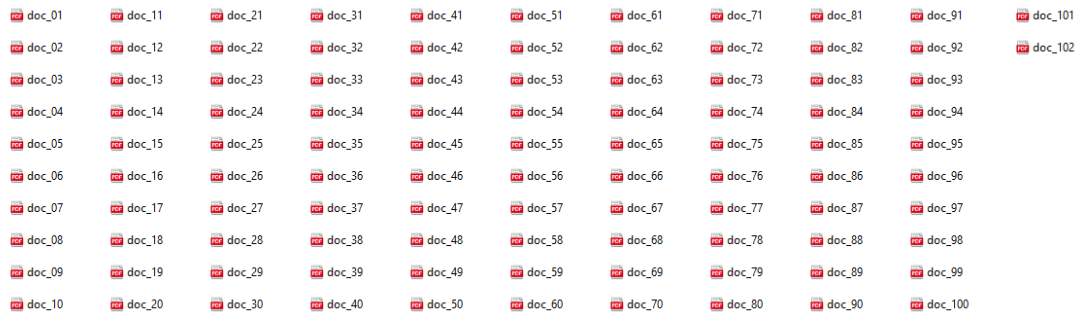|										
doc_01	doc_11	doc_21	doc_31	doc_41	doc_51	doc_61	doc_71	doc_81	doc_91	doc_101
doc_02	doc_12	doc_22	doc_32	doc_42	doc_52	doc_62	doc_72	doc_82	doc_92	doc_102
doc_03	doc_13	doc_23	doc_33	doc_43	doc_53	doc_63	doc_73	doc_83	doc_93	
doc_04	doc_14	doc_24	doc_34	doc_44	doc_54	doc_64	doc_74	doc_84	doc_94	
doc_05	doc_15	doc_25	doc_35	doc_45	doc_55	doc_65	doc_75	doc_85	doc_95	
doc_06	doc_16	doc_26	doc_36	doc_46	doc_56	doc_66	doc_76	doc_86	doc_96	
doc_07	doc_17	doc_27	doc_37	doc_47	doc_57	doc_67	doc_77	doc_87	doc_97	
doc_08	doc_18	doc_28	doc_38	doc_48	doc_58	doc_68	doc_78	doc_88	doc_98	
doc_09	doc_19	doc_29	doc_39	doc_49	doc_59	doc_69	doc_79	doc_89	doc_99	
doc_10	doc_20	doc_30	doc_40	doc_50	doc_60	doc_70	doc_80	doc_90	doc_100	

Figure 4.7: Document Sample

```
for window_size in window_sizes:
    # test for each window size
    files = classification_api.generate_documents_test(topic, path, window_size)

    # compare the files with expected results
    found = 0
    subject_sample = len(expected_results)
    eliminated = 0
    false_sample = NUM_OF_FILES - subject_sample
    false_found = 0

    for file in files:
        if file in expected_results:
            found += 1
        else:
            false_found += 1

    eliminated = false_sample - false_found

    # calculate the selection_accuracy
    selection_accuracy = found / subject_sample

    # calculate the elimination_accuracy
    elimination_accuracy = eliminated / false_sample

    # calculate the accuracy, geometric mean of selection_accuracy and elimination_accuracy
    accuracy = (selection_accuracy * elimination_accuracy) ** (1/2)
```

Figure 4.8: Test Code for Classification

```
expected_results: ['doc_01.pdf', 'doc_15.pdf', 'doc_25.pdf', 'doc_31.pdf',
'doc_36.pdf', 'doc_50.pdf', 'doc_51.pdf', 'doc_57.pdf', 'doc_59.pdf', 'doc_65.pdf',
'doc_66.pdf', 'doc_71.pdf', 'doc_76.pdf', 'doc_81.pdf', 'doc_89.pdf', 'doc_102.pdf']
```

Figure 4.9: Documents that are expected to be classified.

4.1.3.1. Evaluation Method

In the sample there are 102 documents. Most of them are related with University. Some of them are not. Classification is done with the topic as "lisans eğitim öğretim yönetmeliği". So, it must classify documents that are related with this topic. Documents are not chosen as completely unrelated from each other, because it also tests how implementation can make distinction between not so unrelated documents.

4.1.3.2. Evaluation Results

First Figure 4.10 shows that, Classification performs better when window size is 1. For windows sizes 2, and 4, it also come close to lower bound, but cannot reach it.
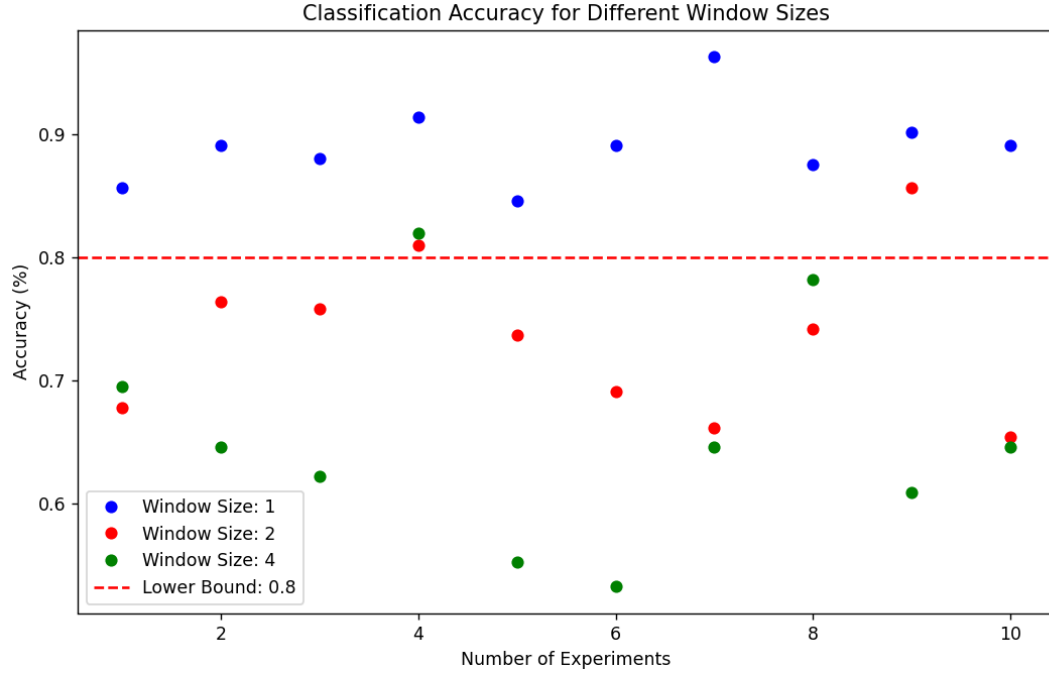

Figure 4.10: Success Rates for Each Window Size

Second Figure 4.11 shows that on average Window Size 1, passed the test. Window 2 is quite close, but Window 4 is not enough for successful classification. Window 1 is preferred according to these results.

Table 4.3: Window Size and Average Classification Accuracy

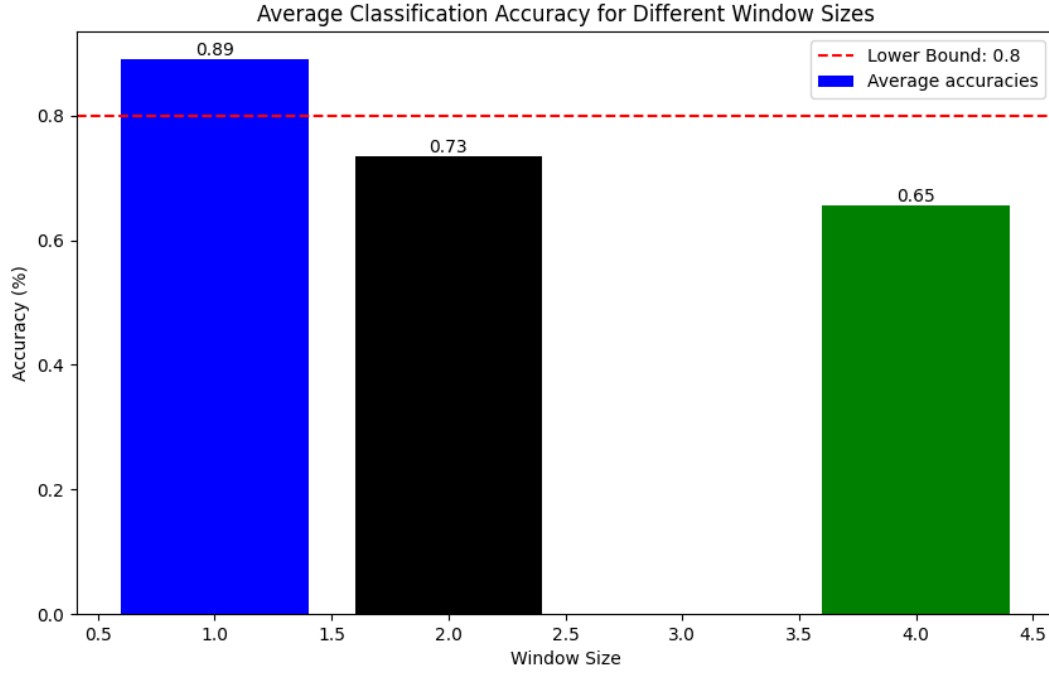| Window Size	Average Classification Accuracy
1	0.89
2	0.73
4	0.65

Figure 4.11: Average Success Rate for Each Classification and Lower Bound.

4.1.4. Document Elimination Coefficient of Variation (Consistency)

There are 10 different success rate for each Window Size. This success criteria is for how those success rates are consistent. It is measured with Coefficient of Variation, calculation is explained at Introduction Chapter. Percentage must below 2%.

4.1.4.1. Evaluation Method

The results of each Classification Accuracy Test is saved in a file, and CV is calculated for each of them with their results using accuracy's for different window sizes.

4.1.4.2. Evaluation Results

None of the Window sizes satisfy consistency success criteria (Figure 4.12). Window 1 is close to the target, but still it is not consistent as wanted. At most 3.44% can be achieved with Window 1. So, classification is not deterministic as wanted, but close.

Table 4.4: Window Size and Coefficient of Variation Percentage

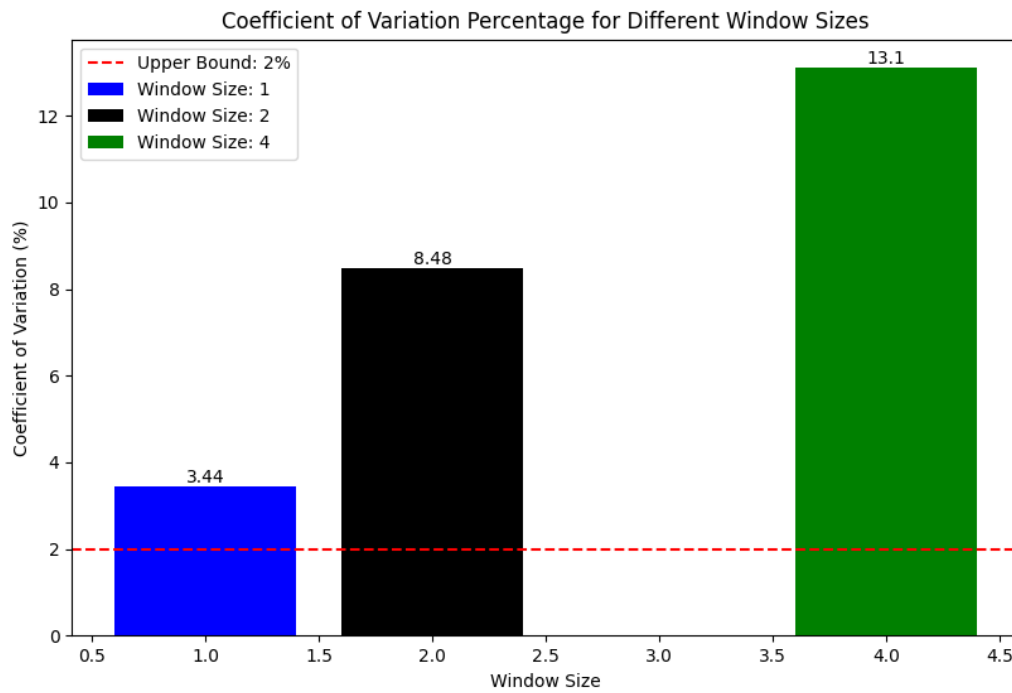| Window Size	Coefficient of Variation (%)
1	3.44
2	8.48
4	13.1

Figure 4.12: Coefficient Variation for Each Window Size

4.2. More Examples with GUI

Example classification screen (Figure 4.13), and asking same question to 8 documents (Figure 4.14), to 4 documents (Figure 4.15).

Figure 4.13: Example classification result.

Figure 4.14: Question to 8 documents.

Figure 4.15: Question to 4 documents.

Irrelevant topic for classification (Figure 4.16). No file related with this topic in given directory.

Figure 4.16: Irrelevant topic for classification.

Irrelevant question (Figure 4.17).

Topic	Subject (s)
eğitim öğretim yönetmeliği ▾	All Subjects ▾

Similarity:
None of the universities can provide information about the weather forecast for tomorrow.

Summary:
Hiçbiri hava durumu tahmini konusunda yardımcı olamıyor.

Time: 4.917078733444214

Yarınki hava durumu nedir?

Figure 4.17: Asking Irrelevant question.

Benzerlik:
Üniversiteler yabancı dil yeterliliğini belirli sınavlara veya üniversite tarafından düzenlenen sınavlara girerek sağlamaktadır.

Farklar:
- Izmir Yüksek Teknoloji Enstitusu ve Istanbul Teknik Üniversitesi, yabancı dil yeterliliğini belirli sınavlara girerek veya üniversite tarafından düzenlenen sınavlara girerek sağlanabileceğini ifade etmişlerdir.
- Orta Doğu Teknik Üniversitesi, yabancı dil yeterliliğini TİB Hazırlık Sınıfında öğretim görmek suretiyle ve belirlenen yönetmeliklere göre sağlamaktadır.
- Gebze Teknik Üniversitesi, yabancı dil yeterliliğini üniversiteye giriş sınavlarında başarılı olmak ve gerekli dil yeterlilik sınavlarına girerek sağlamaktadır.

Özet:
Farklı üniversiteler, yabancı dil yeterliliğini farklı yöntemlerle sağlamaktadır. Bazıları belirlenmiş sınavlara girerken, bazıları üniversite içi sınavlar veya hazırlık programları aracılığıyla öğrencilerin yeterliliğini değerlendirmektedir.

Time: 14.47998309135437

yabancı dil yeterliliği nasıl sağlanır?

Figure 4.18: Another Question to 4 documents.

5. CONCLUSION

Document Based Question Answering using LLM idea is taken further with adding features like Document Classification and Multiple Subject DBQA. Documents are classified according to the given topic with using LLM as preprocessor for each extracted text, and using outputs of this preprocessing step as they are summaries of each. Documents related with topic are separated and tagged after giving summaries of each document as an input to LLM again. Then each selected document is stored on Vector Database. Single DBQA modules are as sub-module for Multiple DBQA system, with retrieving answer from each subject, those answers are given to the LLM again, and it produced the final answer. All of the modules are integrated, and provided a GUI in order to access the features of the system.

Project had 4 success criteria, 2 of them for question answering loop, 2 of them for document classification. Details of these are in the Experiments chapter. **Response time criteria** is succeeded with at most 16 documents. On average, they didn't hit the upper bound. **Cost of answer unification** is increased linearly with number of documents. With using their average, they almost draw a line. So, systems cost for each answer unification increases linearly with number of documents. This linear increasing makes the system scalable. **Classification success** is 89% for window 1 case. System can successfully, classify document above 80%. **Coefficient of Variation** result is 3.44%, which is above the upper bound %2. System is close to the target, but it is not enough to say that classification results are consistent.

BIBLIOGRAPHY

[1] M. A. C. Soares and F. S. Parreiras, "A literature review on question answering techniques, paradigms and systems," *Journal of King Saud University*, vol. 32, no. 6, pp. 635–646, 2020.

[2] M.-C. Yang, D.-G. Lee, S.-Y. Park, and H.-C. Rim, "Knowledge-based question answering using the semantic embedding space," *Journal of King Saud University*, vol. 42, no. 23, pp. 9086–9104, 2015.

[3] P. Lewis, E. Perez, A. Piktus, *et al.*, "Retrieval-augmented generation for knowledge-intensive nlp tasks," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., vol. 33, Curran Associates, Inc., 2020, pp. 9459–9474. [Online]. Available: `https://proceedings.neurips.cc/paper_files/paper/2020/file/6b493230205f780e1bc26945df7481e5-Paper.pdf`.

[4] Langchain, Inc. "Question and answering with rag." (2024), [Online]. Available: `https://python.langchain.com/docs/use_cases/question_answering/` (visited on 2024).

[5] Y. Gao, Y. Xiong, X. Gao, *et al.*, *Retrieval-augmented generation for large language models: A survey*, 2024. arXiv: `2312.10997` [`cs.CL`].

[6] Lydia Husser. "Prompt engineering for generative ai." (2024), [Online]. Available: `https://www.wevolver.com/article/prompt-engineering-for-generative-ai` (visited on 2023).

[7] OpenAI. "Prompt engineering." (2024), [Online]. Available: `https://platform.openai.com/docs/guides/prompt-engineering/strategy-write-clear-instructions` (visited on 2024).

[8] OpenAI. "Api reference." (2024), [Online]. Available: `https://platform.openai.com/docs/api-reference/introduction` (visited on 2024).

[9] OpenAI. "Embeddings." (2024), [Online]. Available: `https://platform.openai.com/docs/guides/embeddings/what-are-embeddings` (visited on 2024).

[10] ChromaDB. "Chroma database." (2024), [Online]. Available: `https://docs.trychroma.com/` (visited on 2024).

CV

Personal Information

Name: Burak Kocausta
Email: bkocausta@gmail.com
Phone: +905077349202
Address: Darıca, Kocaeli
Linkedin, GitHub

Education

Bachelor of Science in Computer Engineering
Gebze Technical University, Gebze, Kocaeli, 2019-2024
GPA: 3.55

Skills

Programming Languages: Java, C, C++, Kotlin, Python, Verilog HDL, SQL, Common Lisp, Prolog, MIPS Assembly
Technologies: CUDA, QT, Android, Spring, MPI, Pthread
Tools: Git/GitHub, Linux, Maven, Jira, Postman
Concepts: OOP, Data Structures, Algorithm Analysis and Design, Agile Development, Computer Organization, Programming Languages/Paradigms, Operating Systems, Systems Programming, POSIX, Natural Language Processing, GPU Programming, Parallel Computation

Experience

Software Engineering Internship
ASELSAN, Ankara, July 2023 - August 2023
Description: Translation application with 3 different modules were developed using ML-KIT, Google Cloud, and Custom version with Glossary feature using SQLite.

APPENDICES

Appendix 1: Promotional Video

Youtube

Appendix 2: Source Code

GitHub