RAG (Retrieval Augmented Generation)

1. LLM 모델의 한계

- Data Privacy 문제
- 데이터 최신성의 문제
- 환각 현상 및 데이터 정확성의 문제

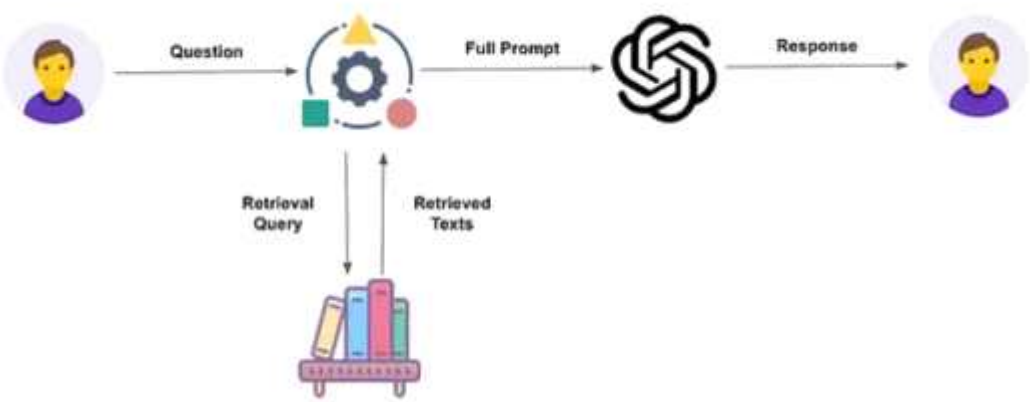
◉ 해결책 : (1)Fine-Tuning, (2)Rag 1.0, (3) Rag 2.0, (4)Agents

◉ Fine-Tuning의 문제점

- 비용이 많이 든다
- Pre-Training처럼 permanent하다 (유연성 부족)
- Pre-training에 의해 만들어진 학습 결과에 (악)영향을 미칠 수 있다

2. RAG의 개념

- 논문 "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks" (2020)
  (by Facebook AI Research, University College London, and New York Univ.)
- 외부 지식으로 LLM의 생성 능력을 증강시키는 것
- Semi-parametric 접근법



(https://www.linkedin.com/pulse/what-retrieval-augmented-generation-grow-right/)

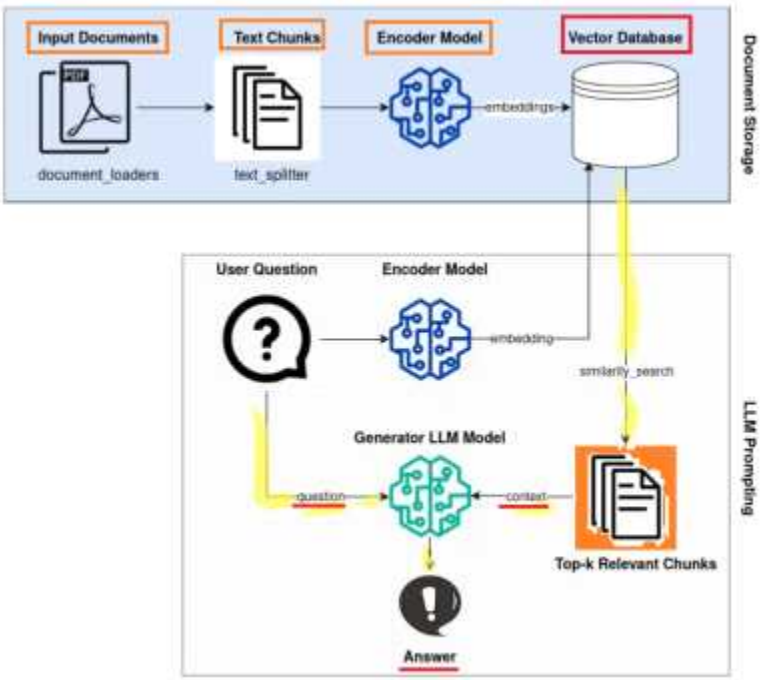
◉ 장점

(1) LLM 제작시 학습하지 않았던 지식을 추가할 수 있음 (**Flexibility**)
(2) 대량의 데이터를 재학습할 필요가 없음 (**Efficiency**)
(3) LLM 모델의 재학습 없이 지식을 현행화 할 수 있음 (**Updatability**)

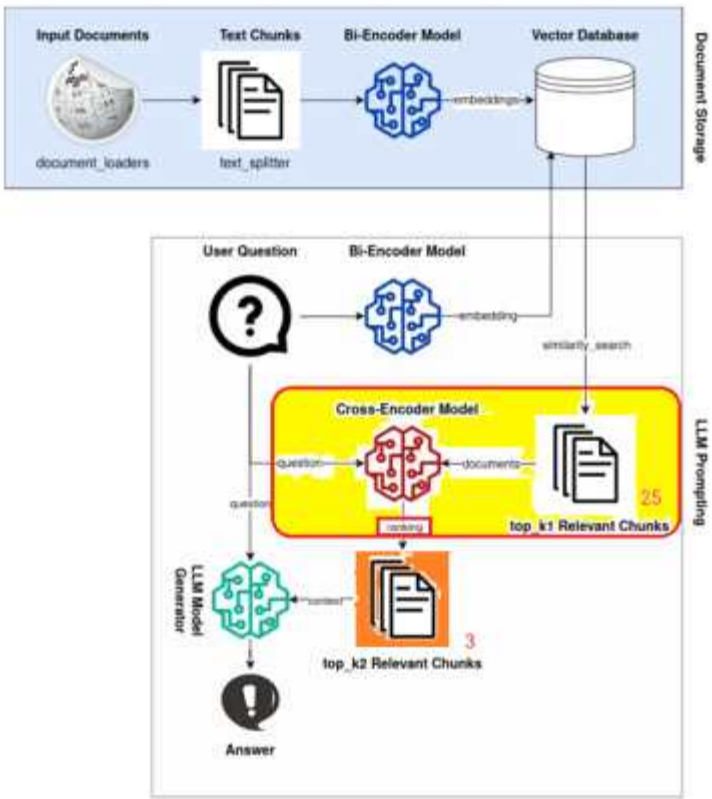
◉ 적용분야 : Question-Answering, Content generation

◉ RAG 적용시 주요 이슈

(1) 외부지식이 클 경우 응답지연이 생길 수 있음 (**Latency**)
(2) Rag을 통해 추출된 정보가 Query에 얼마나 관련이 있는가 (**Relevance**)
(3) Rag Dataset에 의한 답변의 Quality (**Response Quality**)



기본 RAG 파이프라인



Two-Step RAG 파이프라인

(https://medium.com/towards-data-science/how-to-use-re-ranking-for-better-llm-rag-retrieval-243f89414266)

3. RAG를 이용한 LLM 이용의 2단계 (using Langchain)

(https://python.langchain.com/docs/use_cases/question_answering/)

3.1 (1단계) Indexing (offline에서 미리 적용)

(1) **Load** : 외부 Data Loading (DocumentLoaders 사용)

(2) **Split** : 큰 Document data를 작은 chink들로 나눔 (Text Splitters 사용)
Data indexing이나 Prompt length가 작은 llm에 도움

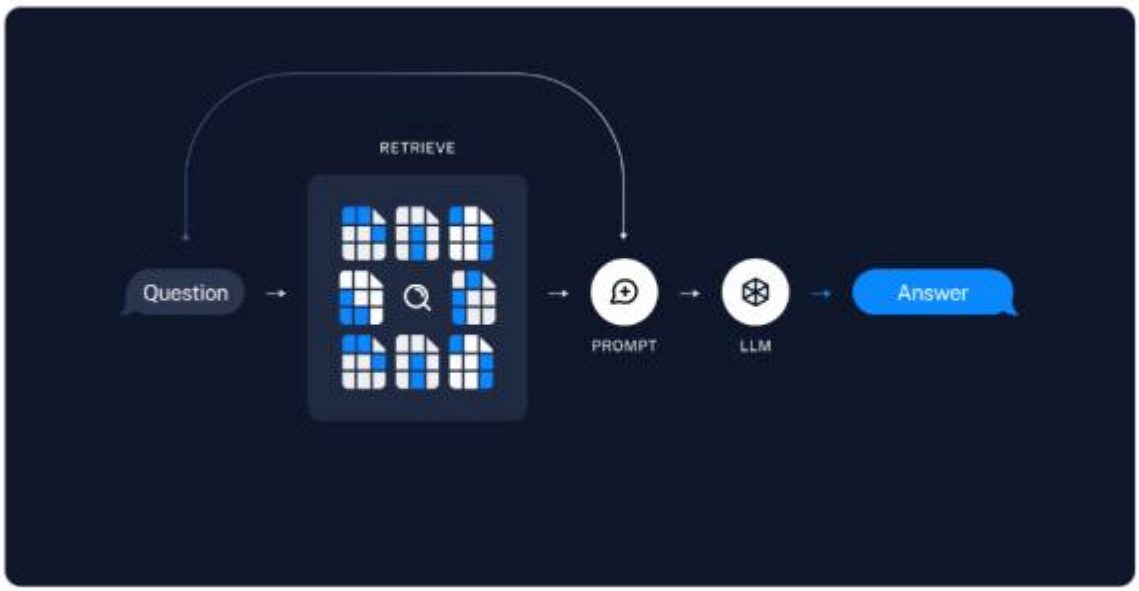
(3) **Embedding** : (Embedding 모델 사용)

(4) **Store** : online실행시 검색될 수 있도록 vector DB에 저장 (Vector Store 사용)

3.2 (2단계) Retrieval and Generation

(1) **Retrieve** : 사용자 입력에 대하여 Storage로부터 적절한 Split들을 추출 (Retrieval 사용)

(2) **Generate** : (question+retrieved data)를 이용하여 LLM으로부터 출력 생성 (LLM 사용)



3.3 Document Loader 선택시 고려 사항

- 한글 인코딩의 적절성과 특수문자의 처리
- 메타데이터의 제공 정도

  .page_content, page_no, title, summary, table, image, ...
- 데이터 로딩 속도

※ 대표적 Loader : **pdfPlumber**, pyPDFLoader, unstructuredPDFLoader, fitz,

3.4 TextSplitter 선택시 고려 사항

- 소제목이 잘리지 않게
- 한글이 잘 Split이 되는가
- **RecursiveCharacterTextSplitter**와 Huggingface에서 제공하는 Splitter들이 가장 유용함

3.5 Embedding 선택시 고려 사항

- Document와 Query에 동일 임베딩 사용
- OpenAIEmbedding(유료, CacheBackedEmbeddings 사용): 한글처리 우수
- **Huggingface가 제공하는 Embedding**
- 실제 임베딩 성능(결과)를 보고 선택할 것

3.6 Vector Store(Vector DB) 선택시 고려 사항

- local LLM 시용시, FAISS 이 유리.

  Opensource이고 local 버전이고 GPU 사용가능
- OpenAIEmbedding(유료, CacheBackedEmbeddings 사용): 한글처리 우수

3.6 전처리시 유의사항 일반

- 페이지 번호가 포함되도록 페이지 단위로 분할 : 자료의 출처 확인에 유리
- 가능하면 Modified Date도 있으면 유리 (버전이 있는 자료의 경우)
- 문서에 반복적으로 사용되는 무의미한 정보(예:페이지 하단의 기관이름)는 제거하는게 유리
- 다단으로 구성된 pdf 문서는 PDFminer 같은 툴로 문단의 순서 정리할 것
- 표가 있을 경우, ocr툴로 표만 따로 추출하는 것이 유리
- chunk overlap 값은 비교적 여유있게 지정할 것
- LLM을 사용하여 다수의 Query를 생성한 후 Multu-Query Retriever를 사용
- Ensemble Retriever (Sparse Retriever(키워드가 중요) + Dense Retriever)를 사용

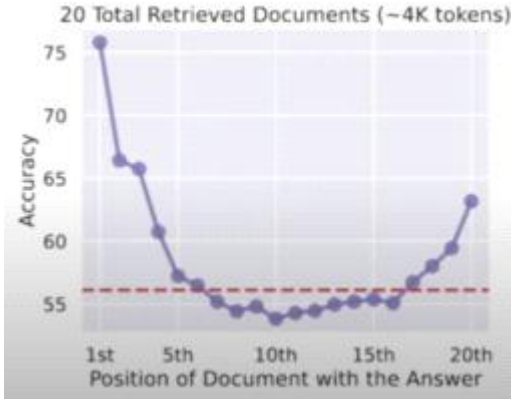
```
bm25retriever = BM25Retriever_from_texts(doc_list)
faiss_retriever = faiss_vectorsotre.as_retriever(search_kwarfs={“k”:2})
ensemble_retriever = EnsembleRetriever( retrievers =
                        [bm25_retriever, faiss_retriever], weights=[0.4,0.6])
```

- Long-context Reorder : 검색된 문서를 10개 이상 포함시에 LLM이 성능저하를 발생한다

  따라서, 긴 컨텍스트의 중간에 상대적으로 덜 중요한 문서를 위치 시킨다

  (langchain의 관련 함수 사용)

```
reordering = LongContextReorder()
reorderd_docs = reordering.transform_documents(docs)
```



3.7 Prompt 작성시 유의사항

- LLM이 FewShotPrompt에 잘 반응하므로, FewShotPromptTemplate를 사용하는 것이 좋음
  langchain사이트의 langSmith에 있는 예제 Prompt를 사용하는 것이 도움

4. RAG Trend

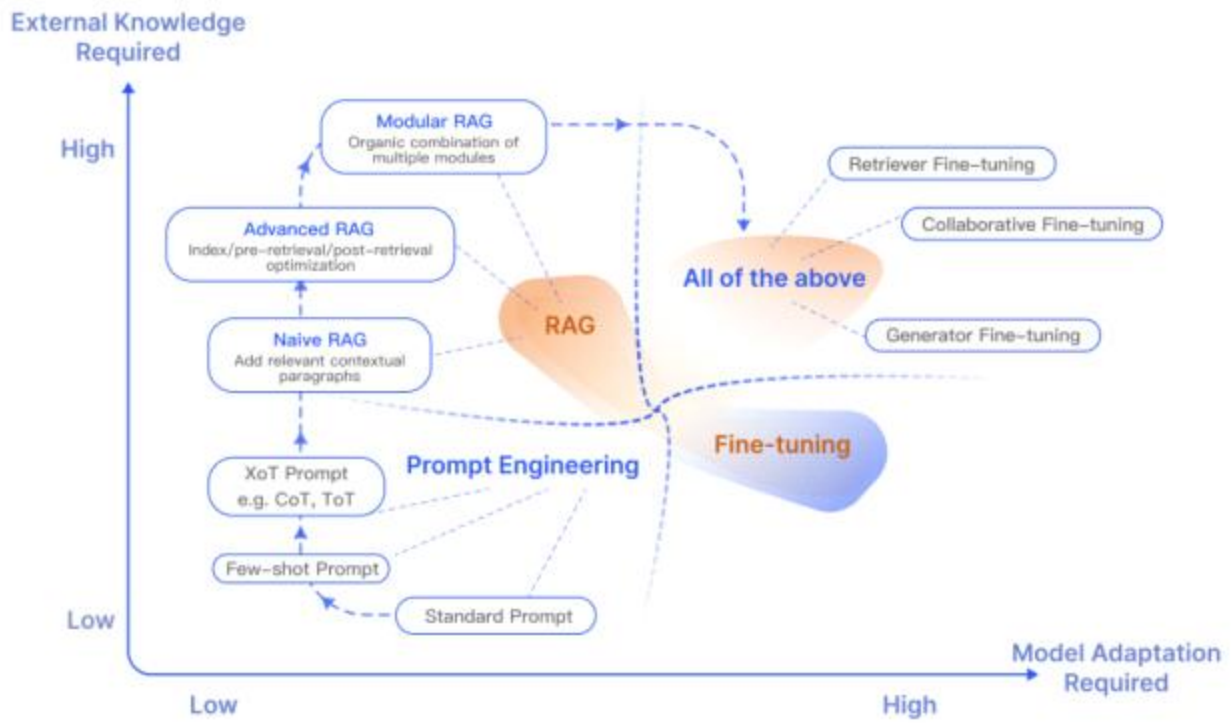


Fig. 4. RAG compared with other model optimization methods in the aspects of "External Knowledge Required" and "Model Adaption Required". Prompt Engineering requires low modifications to the model and external knowledge, focusing on harnessing the capabilities of LLMs themselves. Fine-tuning, on the other hand, involves further training the model. In the early stages of RAG (Naive RAG), there is a low demand for model modifications. As research progresses, Modular RAG has become more integrated with fine-tuning techniques.

(https://arxiv.org/pdf/2312.10997)

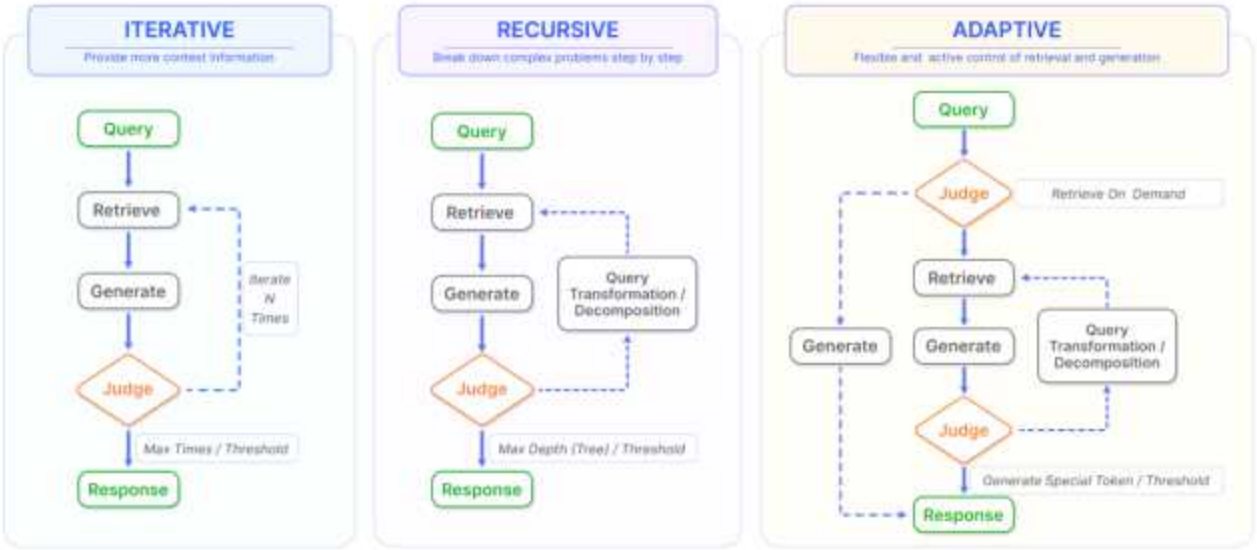


Fig. 5. In addition to the most common once retrieval, RAG also includes three types of retrieval augmentation processes. (left) Iterative retrieval involves alternating between retrieval and generation, allowing for richer and more targeted context from the knowledge base at each step. (Middle) Recursive retrieval involves gradually refining the user query and breaking down the problem into sub-problems, then continuously solving complex problems through retrieval and generation. (Right) Adaptive retrieval focuses on enabling the RAG system to autonomously determine whether external knowledge retrieval is necessary and when to stop retrieval and generation, often utilizing LLM-generated special tokens for control.

(https://arxiv.org/pdf/2312.10997)

5. RAG using Wikipedia and PDFs 실습

5.1 Source Code (RAG using Wikipedia)

```
# conda create -n rag transformers
# conda activate rag
# conda install pytorch torchvision torchaudio pytorch-cuda=11.8 -c pytorch -c nvidia
# conda install -c conda-forge faiss-gpu
# pip install langchain pdfplumber datasets
# pip install wikipedia sentence-transformers tiktoken
# pip install --upgrade huggingface_hub
# pip install --upgrade openai
import numpy as np
fromtransformers importAutoTokenizer
fromlangchain_community.document_loaders importWikipediaLoader
fromlangchain.text_splitter importRecursiveCharacterTextSplitter
fromlangchain_community.embeddings importHuggingFaceEmbeddings
fromlangchain_community.vectorstores import FAISS
fromlangchain_community.vectorstores.utils importDistanceStrategy
fromsentence_transformers importCrossEncoder
import os,openai
fromopenai importOpenAI
SHOW_INSIDE =True
# [1] harrypotter 키워드에 대한 wikipedia page 3개 읽어오기
pages =WikipediaLoader(query="Harry Potter",load_max_docs=3,lang="en").load()
# [2.1] text splitter 정의
text_splitter =RecursiveCharacterTextSplitter.from_huggingface_tokenizer(
    tokenizer=AutoTokenizer.from_pretrained("sentence-transformers/all-MiniLM-L12-v2"),
    chunk_size=256,     # 256 chars / chunk
    chunk_overlap=16,   # 이웃하는 chunk사이 16 chars overlap
    strip_whitespace=True,  # chunk 생성시에 공백 제거
)
# [2.2]불러온 페이지에 대하여 text_splitting하여 chunks 구하기
chunks =text_splitter.split_documents(pages)
if SHOW_INSIDE :
    print ('\nsplit된 chunk의 갯수: ',len(chunks))
    print('\nchunk[0]의 전체내용: \n',chunks[0])
    print('\nchunk[0]의 page_content: \n',chunks[0].page_content)
    print('\nchunk[0]의 metadata: \n)',chunks[0].metadata)
    print('\nchunk[0]의 metadata title: \n',chunks[0].metadata['title'])
    print('\nchunk[0]의 metadta summary: \n',chunks[0].metadata['summary'])
    print('\nchunk[0]의 metadata source: \n',chunks[0].metadata['source'])
# [3] bidirectional encpoder 모델을 이용한 Embedding 정의
bi_encoder =HuggingFaceEmbeddings(
    #model_name="sentence-transformers/all-MiniLM-L12-v2", model_kwargs={"device": "cpu"}
    model_name="sentence-transformers/all-MiniLM-L12-v2",model_kwargs={"device":"cuda"}
)
# question과 첫번째 chunk의 내용 embedding 해보기
'''
embedding1 = bi_encoder.embed_query("Harry Potter")
embedding2 = bi_encoder.embed_query(chunks[0].page_content)
'''
# [4] chunks들로부터 bi_encoder를 이용한 임베딩을 사용하고, dot-product거리를 이용하여 faiss index
생성.
# FAISS (Facebook AI Similarity Search)
faiss_db =FAISS.from_documents(
    chunks,bi_encoder,distance_strategy=DistanceStrategy.DOT_PRODUCT
)
# [5] cross encoder 정의
```

```
cross_encoder =CrossEncoder(
    "cross-encoder/ms-marco-TinyBERT-L-2-v2",max_length=512,device="cpu"
)
'''
cross_encoder.rank(
query="Do I like cats?",
documents=["I like cats", "I like horses", "I like dogs"],
return_documents=True,
)
'''
client =OpenAI(
    #api_key=os.environ.get("OPENAI_API_KEY"),
    api_key ="sk-proj-P1dJkZ2685**********************tKn0x7vYAeqXrL18AR"
)
#=============================================================================================
default_question ="On what date was the first book in the Harry Potter series originally published?"
while True :
    question =input("\nType 'q' to quit: ")

    if question =='q':break
    elif question =='':
        question =default_question

    # [5] question과 유사도가 가장 가까운 상위 25의 chunk를 추출
    retrieved_chunks =faiss_db.similarity_search(question,k=25)
    reranked_chunks =cross_encoder.rank(
        (question),
        [doc.page_content for doc in retrieved_chunks],
        top_k=3,
        return_documents=True,
    )
    context ="".join(doc["text"]+"\n----------\n"for doc in reranked_chunks)
    print("\n\n***최종 상위 답변 3개 통합문장 : \n",context)
    if SHOW_INSIDE :
          print(f"\n\n*** RAG에 의한 연관된 상위 {len(reranked_chunks)}개 chunks의 제목과 출처 (URL)")
        for doc in reranked_chunks:
            corpus_id =doc["corpus_id"]
            print(
                    f"Chunk {corpus_id}from '{retrieved_chunks[corpus_id].metadata['title']}' \n\t(URL: {retrieved_chunks[corpus_id].metadata['source']})"
            )

    #-----------------------------------------------------------------------------------------

    # Use OpenAI's LLM to generate a response based on the context
    prompt=f"You are a helpful assistant. Based on the following details about Harry Potter, {context}Answer the question.",
    try :
        response =client.chat.completions.create(
            messages=[
                    {
                        "role":"system","content":prompt,
                        "role":"user","content":question,
                    }
            ],
                            model="gpt-3.5-turbo-instruct",     #    or     "text-davinci-002"  "gpt-3.5-turbo-instruct"
            max_tokens=150,
            temperature=0.5,
```

```
            stop=["\n"]
            )
        print("\nOpenAI LLM의 답변:\n",response.choices[0].text.strip())
    except Exception as e:
        print("Error : ",e)
```

5.2 실행결과 (RAG using Wikipedia)

```
(rag) D:\myprojects\nlp\rag>python rag_wiki.py
D:\program\Anaconda30\envs\rag\Lib\site-packages\huggingface_hub\file_download.py:1132: FutureWarning: `resume_download` is deprecated and will be removed in version 1.0.0. Downloads always resume when possible. If you want to force a new download, use `force_download=True`.
  warnings.warn(
Token indices sequence length is longer than the specified maximum sequence length for this model (549 > 512). Running this sequence through the model will result in indexing errors

split된 chunk의 갯수:  18

chunk[0]의 전체내용:
 page_content="Harry Potter is a series of seven fantasy novels written by British author J. K. Rowling. The novels chronicle the lives of a young wizard, Harry Potter, and his friends Hermione Granger and Ron Weasley, all of whom are students at Hogwarts School of Witchcraft and Wizardry. The main story arc concerns Harry's conflict with Lord Voldemort, a dark wizard who intends to become immortal, overthrow the wizard governing body known as the Ministry of Magic, and subjugate all wizards and Muggles (non-magical people).\nThe series was originally published in English by Bloomsbury in the United Kingdom and Scholastic Press in the United States.  A series of many genres, including fantasy, drama, coming-of-age fiction, and the British school story (which includes elements of mystery, thriller, adventure, horror, and romance), the world of Harry Potter explores numerous themes and includes many cultural meanings and references. Major themes in the series include prejudice, corruption, madness, and death." metadata={'title': 'Harry Potter', 'summary': "Harry Potter is a series of seven fantasy novels written by British author J. K. Rowling. The novels chronicle the lives of a young wizard, Harry Potter, and his friends Hermione Granger and Ron Weasley, all of whom are students at Hogwarts School of Witchcraft and Wizardry. The main story arc concerns Harry's conflict with Lord Voldemort, a dark wizard who intends to become immortal, overthrow the wizard governing body known as the Ministry of Magic, and subjugate all wizards and Muggles (non-magical people).\nThe series was originally published in English by Bloomsbury in the United Kingdom and Scholastic Press in the United States.  A series of many genres, including fantasy, drama, coming-of-age fiction, and the British school story (which includes elements of mystery, thriller, adventure, horror, and romance), the world of Harry Potter explores numerous themes and includes many cultural meanings and references. Major themes in the series include prejudice, corruption, madness, and death.\nSince the release of the first novel, Harry Potter and the Philosopher's Stone, on 26 June 1997, the books have found immense popularity, positive reviews, and commercial success worldwide. They have attracted a wide adult audience as well as younger readers, and are widely considered cornerstones of modern literature. As of February 2023, the books have sold more than 600 million copies worldwide, making them the best-selling book series in history, and have been available in 85 languages. The last four books consecutively set records as the fastest-selling books in history, with the final instalment selling roughly 2.7 million copies in the United Kingdom and 8.3 million copies in the United States within twenty-four hours of its release.\nThe original seven books were adapted into an eight-part namesake film series by Warner Bros. Pictures. In 2016, the total value of the Harry Potter franchise was estimated at $25 billion, making Harry Potter one of the highest-grossing media franchises of all time. Harry Potter and the Cursed Child is a play based on a story co-written by Rowling.\nThe success of the books and films has allowed the Harry Potter franchise to expand with numerous derivative works, a travelling exhibition that premiered in Chicago in 2009, a studio tour in London that opened in 2012, a digital platform on which J. K. Rowling updates the series with new information and insight, and a trilogy of spin-off films premiering in November 2016 with Fantastic Beasts and Where to Find Them, among many other developments. Themed attractions, collectively known as The Wizarding World of Harry Potter, have been built at several Universal Destinations & Experiences amusement parks around the world.", 'source': 'https://en.wikipedia.org/wiki/Harry_Potter'}

chunk[0]의 page_content:
 Harry Potter is a series of seven fantasy novels written by British author J. K. Rowling. The novels chronicle the lives of a young wizard, Harry Potter, and his friends Hermione Granger and Ron Weasley, all of whom are students at Hogwarts School of Witchcraft and Wizardry. The main story arc concerns Harry's conflict with Lord Voldemort, a dark wizard who intends to become immortal, overthrow the wizard governing body known as the Ministry of Magic, and subjugate all wizards and Muggles (non-magical people).
The series was originally published in English by Bloomsbury in the United Kingdom and Scholastic Press in the United States.  A series of many genres, including fantasy, drama, coming-of-age fiction, and the British school story (which includes elements of mystery, thriller, adventure, horror,
```

and romance), the world of Harry Potter explores numerous themes and includes many cultural meanings and references. Major themes in the series include prejudice, corruption, madness, and death.

chunk[0]의 metadata:
) {'title': 'Harry Potter', 'summary': "Harry Potter is a series of seven fantasy novels written by British author J. K. Rowling. The novels chronicle the lives of a young wizard, Harry Potter, and his friends Hermione Granger and Ron Weasley, all of whom are students at Hogwarts School of Witchcraft and Wizardry. The main story arc concerns Harry's conflict with Lord Voldemort, a dark wizard who intends to become immortal, overthrow the wizard governing body known as the Ministry of Magic, and subjugate all wizards and Muggles (non-magical people).\nThe series was originally published in English by Bloomsbury in the United Kingdom and Scholastic Press in the United States. A series of many genres, including fantasy, drama, coming-of-age fiction, and the British school story (which includes elements of mystery, thriller, adventure, horror, and romance), the world of Harry Potter explores numerous themes and includes many cultural meanings and references. Major themes in the series include prejudice, corruption, madness, and death.\nSince the release of the first novel, Harry Potter and the Philosopher's Stone, on 26 June 1997, the books have found immense popularity, positive reviews, and commercial success worldwide. They have attracted a wide adult audience as well as younger readers, and are widely considered cornerstones of modern literature. As of February 2023, the books have sold more than 600 million copies worldwide, making them the best-selling book series in history, and have been available in 85 languages. The last four books consecutively set records as the fastest-selling books in history, with the final instalment selling roughly 2.7 million copies in the United Kingdom and 8.3 million copies in the United States within twenty-four hours of its release.\nThe original seven books were adapted into an eight-part namesake film series by Warner Bros. Pictures. In 2016, the total value of the Harry Potter franchise was estimated at $25 billion, making Harry Potter one of the highest-grossing media franchises of all time. Harry Potter and the Cursed Child is a play based on a story co-written by Rowling.\nThe success of the books and films has allowed the Harry Potter franchise to expand with numerous derivative works, a travelling exhibition that premiered in Chicago in 2009, a studio tour in London that opened in 2012, a digital platform on which J. K. Rowling updates the series with new information and insight, and a trilogy of spin-off films premiering in November 2016 with Fantastic Beasts and Where to Find Them, among many other developments. Themed attractions, collectively known as The Wizarding World of Harry Potter, have been built at several Universal Destinations & Experiences amusement parks around the world.", 'source': 'https://en.wikipedia.org/wiki/Harry_Potter'}

chunk[0]의 metadata title:
Harry Potter

chunk[0]의 metadta summary:
Harry Potter is a series of seven fantasy novels written by British author J. K. Rowling. The novels chronicle the lives of a young wizard, Harry Potter, and his friends Hermione Granger and Ron Weasley, all of whom are students at Hogwarts School of Witchcraft and Wizardry. The main story arc concerns Harry's conflict with Lord Voldemort, a dark wizard who intends to become immortal, overthrow the wizard governing body known as the Ministry of Magic, and subjugate all wizards and Muggles (non-magical people).
The series was originally published in English by Bloomsbury in the United Kingdom and Scholastic Press in the United States. A series of many genres, including fantasy, drama, coming-of-age fiction, and the British school story (which includes elements of mystery, thriller, adventure, horror, and romance), the world of Harry Potter explores numerous themes and includes many cultural meanings and references. Major themes in the series include prejudice, corruption, madness, and death.
Since the release of the first novel, Harry Potter and the Philosopher's Stone, on 26 June 1997, the books have found immense popularity, positive reviews, and commercial success worldwide. They have attracted a wide adult audience as well as younger readers, and are widely considered cornerstones of modern literature. As of February 2023, the books have sold more than 600 million copies worldwide, making them the best-selling book series in history, and have been available in 85 languages. The last four books consecutively set records as the fastest-selling books in history, with the final instalment selling roughly 2.7 million copies in the United Kingdom and 8.3 million copies in the United States within twenty-four hours of its release.
The original seven books were adapted into an eight-part namesake film series by Warner Bros. Pictures. In 2016, the total value of the Harry Potter franchise was estimated at $25 billion, making Harry Potter one of the highest-grossing media franchises of all time. Harry Potter and the Cursed Child is a play based on a story co-written by Rowling.
The success of the books and films has allowed the Harry Potter franchise to expand with numerous derivative works, a travelling exhibition that premiered in Chicago in 2009, a studio tour in London

that opened in 2012, a digital platform on which J. K. Rowling updates the series with new information and insight, and a trilogy of spin-off films premiering in November 2016 with Fantastic Beasts and Where to Find Them, among many other developments. Themed attractions, collectively known as The Wizarding World of Harry Potter, have been built at several Universal Destinations & Experiences amusement parks around the world.

chunk[0]의 metadata source:
https://en.wikipedia.org/wiki/Harry_Potter
D:\program\Anaconda30\envs\rag\Lib\site-packages\huggingface_hub\file_download.py:1132: FutureWarning: `resume_download` is deprecated and will be removed in version 1.0.0. Downloads always resume when possible. If you want to force a new download, use `force_download=True`.
warnings.warn(

Type 'q' to quit: On what date was the first book in the Harry Potter series originally published?

최종 상위 답변 3개 통합문장 :
Harry Potter and the Philosopher's Stone is a fantasy novel written by British author J. K. Rowling. The first novel in the Harry Potter series and Rowling's debut novel, it follows Harry Potter, a young wizard who discovers his magical heritage on his eleventh birthday, when he receives a letter of acceptance to Hogwarts School of Witchcraft and Wizardry. Harry makes close friends and a few enemies during his first year at the school and with the help of his friends, Ron Weasley and Hermione Granger, he faces an attempted comeback by the dark wizard Lord Voldemort, who killed Harry's parents, but failed to kill Harry when he was just 15 months old.

The book was first published in the United Kingdom on 26 June 1997 by Bloomsbury. It was published in the United States the following year by Scholastic Corporation under the title Harry Potter and the Sorcerer's Stone. It won most of the British book awards that were judged by children and other awards in the US. The book reached the top of the New York Times list of best-selling fiction in August 1999, and stayed near the top of that list for much of 1999 and 2000. It has been translated into at least 73 other languages, and has been made into a feature-length film of the same name, as have all six of its sequels. The novel has sold in excess of 120 million copies, making it the fourth best-selling book of all time.

Since the release of the first novel, Harry Potter and the Philosopher's Stone, on 26 June 1997, the books have found immense popularity, positive reviews, and commercial success worldwide. They have attracted a wide adult audience as well as younger readers, and are widely considered cornerstones of modern literature. As of February 2023, the books have sold more than 600 million copies worldwide, making them the best-selling book series in history, and have been available in 85 languages. The last four books consecutively set records as the fastest-selling books in history, with the final instalment selling roughly 2.7 million copies in the United Kingdom and 8.3 million copies in the United States within twenty-four hours of its release.
The original seven books were adapted into an eight-part namesake film series by Warner Bros. Pictures. In 2016, the total value of the Harry Potter franchise was estimated at $25 billion, making Harry Potter one of the highest-grossing media franchises of all time. Harry Potter and the Cursed Child is a play based on a story co-written by Rowling.

RAG에 의한 연관된 상위 3 개 chunks의 제목과 출처(URL)
Chunk 4 from 'Harry Potter and the Philosopher's Stone'
(URL: https://en.wikipedia.org/wiki/Harry_Potter_and_the_Philosopher%27s_Stone)
Chunk 3 from 'Harry Potter and the Philosopher's Stone'
(URL: https://en.wikipedia.org/wiki/Harry_Potter_and_the_Philosopher%27s_Stone)
Chunk 0 from 'Harry Potter'
(URL: https://en.wikipedia.org/wiki/Harry_Potter)
Error : Error code: 429 - {'error': {'message': 'You exceeded your current quota, please check your plan and billing details. For more information on this error, read the docs: https://platform.openai.com/docs/guides/error-codes/api-errors.', 'type': 'insufficient_quota', 'param': None, 'code': 'insufficient_quota'}}

Type 'q' to quit: Who is author of "Harry Potter"?

최종 상위 답변 3개 통합문장 :
Harry Potter and the Philosopher's Stone is a fantasy novel written by British author J. K. Rowling. The first novel in the Harry Potter series and Rowling's debut novel, it follows Harry Potter, a young

wizard who discovers his magical heritage on his eleventh birthday, when he receives a letter of acceptance to Hogwarts School of Witchcraft and Wizardry. Harry makes close friends and a few enemies during his first year at the school and with the help of his friends, Ron Weasley and Hermione Granger, he faces an attempted comeback by the dark wizard Lord Voldemort, who killed Harry's parents, but failed to kill Harry when he was just 15 months old.

Harry Potter is a series of seven fantasy novels written by British author J. K. Rowling. The novels chronicle the lives of a young wizard, Harry Potter, and his friends Hermione Granger and Ron Weasley, all of whom are students at Hogwarts School of Witchcraft and Wizardry. The main story arc concerns Harry's conflict with Lord Voldemort, a dark wizard who intends to become immortal, overthrow the wizard governing body known as the Ministry of Magic, and subjugate all wizards and Muggles (non-magical people).
The series was originally published in English by Bloomsbury in the United Kingdom and Scholastic Press in the United States. A series of many genres, including fantasy, drama, coming-of-age fiction, and the British school story (which includes elements of mystery, thriller, adventure, horror, and romance), the world of Harry Potter explores numerous themes and includes many cultural meanings and references. Major themes in the series include prejudice, corruption, madness, and death.

Harry Potter is a film series based on the eponymous novels by British author J. K. Rowling. The series is produced and distributed by Warner Bros. Pictures and consists of eight fantasy films, beginning with Harry Potter and the Philosopher's Stone (2001) and culminating with Harry Potter and the Deathly Hallows – Part 2 (2011). A spin-off prequel series started with Fantastic Beasts and Where to Find Them (2016), marking the beginning of the Wizarding World shared media franchise.
The series was mainly produced by David Heyman, and stars Daniel Radcliffe, Rupert Grint, and Emma Watson as the three leading characters: Harry Potter, Ron Weasley, and Hermione Granger. Four directors worked on the series: Chris Columbus, Alfonso Cuarón, Mike Newell, and David Yates. Michael Goldenberg wrote the screenplay for Harry Potter and the Order of the Phoenix (2007), while the remaining films' screenplays were written by Steve Kloves. Production took place over ten years, with the main story arc following Harry's quest to overcome his arch-enemy Lord Voldemort.

RAG에 의한 연관된 상위 3 개 chunks의 제목과 출처(URL)
Chunk 2 from 'Harry Potter and the Philosopher's Stone'
(URL: https://en.wikipedia.org/wiki/Harry_Potter_and_the_Philosopher%27s_Stone)
Chunk 0 from 'Harry Potter'
(URL: https://en.wikipedia.org/wiki/Harry_Potter)
Chunk 1 from 'Harry Potter (film series)'
(URL: https://en.wikipedia.org/wiki/Harry_Potter_(film_series))
Error : Error code: 429 - {'error': {'message': 'You exceeded your current quota, please check your plan and billing details. For more information on this error, read the docs: https://platform.openai.com/docs/guides/error-codes/api-errors.', 'type': 'insufficient_quota', 'param': None, 'code': 'insufficient_quota'}}

Type 'q' to quit:

5.3 Source Code (RAG using PDFs)

```
# conda create -n rag transformers
# conda activate rag
# conda install pytorch torchvision torchaudio pytorch-cuda=11.8 -c pytorch -c nvidia
# conda install -c conda-forge faiss-gpu
# pip install langchain pdfplumber datasets
# pip install wikipedia sentence-transformers tiktoken
# pip install --upgrade huggingface_hub
# pip install --upgrade openai
import numpy as np
fromtransformers importAutoTokenizer
fromlangchain_community.document_loaders importWikipediaLoader
fromlangchain.text_splitter importRecursiveCharacterTextSplitter
fromlangchain_community.embeddings importHuggingFaceEmbeddings
fromlangchain_community.vectorstores import FAISS
fromlangchain_community.vectorstores.utils importDistanceStrategy
fromsentence_transformers importCrossEncoder
import os,openai
fromopenai importOpenAI
importpdfplumber
SHOW_INSIDE =True
'''
# [1] harrypotter 키워드에 대한 wikipedia page 3개 읽어오기
pages = WikipediaLoader(query="Harry Potter", load_max_docs=3, lang="en").load()
'''
# [1] Load PDF documents
class Document:
    def __init__(self,page_content,metadata=None):
        self.page_content =page_content
        self.metadata =metadata or {}      # text_splitter가 metadata를 기본 필요로 함

pdf_paths     =['example1.pdf','example2.pdf','glocal1.pdf','glocal2.pdf','glocal3.pdf']      #
Specify your PDF file paths here
pages_text =[]
for pdf_path in pdf_paths:
    with pdfplumber.open(pdf_path)as pdf:
        for page in pdf.pages:
            page_text =page.extract_text()
            if page_text:  # Ensure there is text on the page
                doc =Document(page_text)
                pages_text.append(doc)

# [2.1] text splitter 정의
text_splitter =RecursiveCharacterTextSplitter.from_huggingface_tokenizer(
    tokenizer=AutoTokenizer.from_pretrained("sentence-transformers/all-MiniLM-L12-v2"),
    chunk_size=256,     # 256 chars / chunk
    chunk_overlap=16,   # 이웃하는 chunk사이 16 chars overlap
    strip_whitespace=True,  # chunk 생성시에 공백 제거
)
# [2.2]불러온 페이지에 대하여 text_splitting하여 chunks 구하기
#chunks = text_splitter.split_documents(pages)
chunks =text_splitter.split_documents(pages_text)
if SHOW_INSIDE :
    print ('\nsplit된 chunk의 갯수: ',len(chunks))
    print('\nchunk[0]의 전체내용: \n',chunks[0])
    print('\nchunk[0]의 page_content: \n',chunks[0].page_content)
# [3] bidirectional encpoder 모델을 이용한 Embedding 정의
bi_encoder =HuggingFaceEmbeddings(
    #model_name="sentence-transformers/all-MiniLM-L12-v2", model_kwargs={"device": "cpu"}
```

```
    model_name="sentence-transformers/all-MiniLM-L12-v2",model_kwargs={"device":"cuda"}
)
# question과 첫번째 chunk의 내용 embedding 해보기
'''
embedding1 = bi_encoder.embed_query("글로컬대학")
embedding2 = bi_encoder.embed_query(chunks[0].page_content)
'''
# [4] chunks들로부터 bi_encoder를 이용한 임베딩을 사용하고, dot-product거리를 이용하여 faiss index 생성.
# FAISS (Facebook AI Similarity Search)
faiss_db =FAISS.from_documents(
    chunks,bi_encoder,distance_strategy=DistanceStrategy.DOT_PRODUCT
)
# [5] cross encoder 정의
cross_encoder =CrossEncoder(
    "cross-encoder/ms-marco-TinyBERT-L-2-v2",max_length=768,device="cuda"
)
client =OpenAI(
    #api_key=os.environ.get("OPENAI_API_KEY"),
    api_key ="sk-proj-*****************************************AR"
)
#=============================================================================================
default_question ="글로컬대학 30 예비지정대학으로 선정된 대학은?"
while True :
    question =input("\nType 'q' to quit: ")

    if question =='q':break
    elif question =='':
        question =default_question

    # [5] question과 유사도가 가장 가까운 상위 25의 chunk를 추출
    retrieved_chunks =faiss_db.similarity_search(question,k=25)
    reranked_chunks =cross_encoder.rank(
        (question),
        [doc.page_content for doc in retrieved_chunks],
        top_k=3,
        return_documents=True,
    )
    context ="".join(doc["text"]+"\n----------\n"for doc in reranked_chunks)
    print("\n\n*** 최종 상위 답변 3개 통합문장 : \n",context)
    if SHOW_INSIDE :
        print(f"\n*** RAG에 의한 연관된 상위 {len(reranked_chunks)}개 chunks")
        for doc in reranked_chunks:
            corpus_id =doc["corpus_id"]
            print(
                f"Chunk {corpus_id}"
            )

    #---------------------------------------------------------------------------------------

    # Use OpenAI's LLM to generate a response based on the context
     prompt=f"You are a helpful assistant. Based on the following details about Harry Potter, {context}Answer the question.",
    try :
        response =client.chat.completions.create(
            messages=[
                    {
                        "role":"system","content":prompt,
                        "role":"user","content":question,
                    }
            ],
```

```
                            model="gpt-3.5-turbo-instruct",     #    or     "text-davinci-002"
"gpt-3.5-turbo-instruct"
          max_tokens=150,
          temperature=0.5,
          stop=["\n"]
          )
      print("\nOpenAI LLM의 답변:\n",response.choices[0].text.strip())
   except Exception as e:
      print("Error : ",e)
```

5.4 실행결과 (RAG using PDFs)

```
(rag) D:\myprojects\nlp\rag>python rag_pdf.py
D:\program\Anaconda30\envs\rag\Lib\site-packages\huggingface_hub\file_download.py:1132: FutureWarning: `resume_download` is deprecated and will be removed in version 1.0.0. Downloads always resume when possible. If you want to force a new download, use `force_download=True`.
  warnings.warn(

split된 chunk의 갯수:  186

chunk[0]의 전체내용:
 page_content='Understanding of Technological Evolution and\nNew-age Education\nAI Professor in Silla University, S.Korea\nCEO of IACST\npkkim95@gmail.com pkkim95'

chunk[0]의 page_content:
 Understanding of Technological Evolution and
New-age Education
AI Professor in Silla University, S.Korea
CEO of IACST
pkkim95@gmail.com pkkim95
D:\program\Anaconda30\envs\rag\Lib\site-packages\huggingface_hub\file_download.py:1132: FutureWarning: `resume_download` is deprecated and will be removed in version 1.0.0. Downloads always resume when possible. If you want to force a new download, use `force_download=True`.
  warnings.warn(

Type 'q' to quit: 글로컬대학 30 예비지정대학으로 선정된 대학은?


*** 최종 상위 답변 3개 통합문장 :
 정이며, 본지정 평가위원회의 평가 및 글로컬대학위원회의 최종 심
의를 거쳐 오는 8월 말 최종 글로컬대학을 지정해 발표할 예정이다.
----------
학의 혁신기획서를 높게 평가하였습니다.
글로컬대학 예비지정 15개를 가, 나, 다, 라 순서로 발표하겠습니다.
강원대학교·강릉원주대학교 공동, 경상국립대학교, 부산대학교·부산교육대학교 공동, 순천대학
----------
교육부 2024 글로컬대학 신라대-동명대 연합대학 예비지정 선정... 최종선정에
총력
조회742
2024-04-16 00:00:00
첨부파일
• 1. 신라대학교 전경 사진.jpg (2.45 MB)
교육부 2024 글로컬대학 신라대-
동명대 연합대학
예비지정 선정... 최종선정에 총력
----------


*** RAG에 의한 연관된 상위 3 개 chunks
Chunk 13
Chunk 5
Chunk 15
Error :  Error code: 429 - {'error': {'message': 'You exceeded your current quota, please check your plan and billing details. For more information on this error, read the docs: https://platform.openai.com/docs/guides/error-codes/api-errors.', 'type': 'insufficient_quota', 'param': None, 'code': 'insufficient_quota'}}

Type 'q' to quit: 글로컬대학 예비대학에 신라대학교가 선정 되었나?


*** 최종 상위 답변 3개 통합문장 :
 교, 순천향대학교, 안동대학교·경북도립대학교 공동, 연세대학교 미래캠퍼스, 울산대학교, 인제
대학교, 전남대학교, 전북대학교, 충북대학교·한국교통대학교 공동, 포항공과대학교, 한동대학
교, 한림대학교입니다.
```

```
----------
대규모 AI 일상화 시대, 교육에의 시사점
• 기존 가치체계와 판단방법에서 탈피하라 (직업의 세계, 필요역량, 가치, …)
• 내가 가진(가질) 능력(전문성)이 AI보다 우월한가? (문제해결 능력보다 문제 발견 능력)
• AI와 사람간의 융합 능력을 길러라 (AI 활용 능력)
----------
이 될 것입니다.
글로컬대학위원회와 교육부는 글로컬대학으로 지정되는 대학뿐 아니라 모든 대학들이 혁신의
비전과 과제들을 차근차근 구현해나갈 수 있도록 다각적으로 지원하겠습니다.
다시 한번 글로컬대학 프로젝트에 참여해주신 모든 대학에 감사의 마음을 전합니다. 감사합니
다.
[질문·답변]
----------


*** RAG에 의한 연관된 상위 3 개 chunks
Chunk 22
Chunk 23
Chunk 1
Error :  Error code: 429 - {'error': {'message': 'You exceeded your current quota, please check your
plan and billing details. For more information on this error, read the docs:
https://platform.openai.com/docs/guides/error-codes/api-errors.', 'type': 'insufficient_quota', 'param':
None, 'code': 'insufficient_quota'}}

Type 'q' to quit: q

(rag) D:\myprojects\nlp\rag>
```

5.5 프로그램 소스 (RAG using PDFs and local LLM)

```python
# Ollama다운로드 및 설치하기  (https://ollama.com/download)
# conda create -n rag transformers
# conda activate rag
# conda install pytorch torchvision torchaudio pytorch-cuda=11.8 -c pytorch -c nvidia
# conda install -c conda-forge faiss-gpu
# pip install langchain pdfplumber datasets
# pip install wikipedia sentence-transformers tiktoken
# pip install --upgrade huggingface_hub
# pip install --upgrade openai
# ollama pull llama3:text          (4.7 Giga bytes)
import numpy as np
fromtransformers importAutoTokenizer
#from langchain_community.document_loaders import WikipediaLoader
fromlangchain.text_splitter importRecursiveCharacterTextSplitter
fromlangchain_community.embeddings importHuggingFaceEmbeddings
fromlangchain_community.vectorstores import FAISS
fromlangchain_community.vectorstores.utils importDistanceStrategy
fromsentence_transformers importCrossEncoder
fromlangchain_community.llms importOllama
#from langchain_community.chat_models import ChatOllama
fromlangchain_core.messages importHumanMessage,AIMessage
fromlangchain_core.prompts importChatPromptTemplate,MessagesPlaceholder
importpdfplumber,  warnings
from queue import Queue
#from openai import OpenAI
def get_all_queue_element(que):
    result_str =""
    while not que.empty():
        result_str +=(que.get()+"\n")
    return result_str
#=================================================================================================
warnings.simplefilter(action='ignore',category=FutureWarning)# warning message 안나오게
SHOW_INSIDE =False
```

```
# [1] Load PDF documents ----------------------------------------
class Document:
    def __init__(self,page_content,metadata=None):
        self.page_content =page_content
        self.metadata =metadata or {}      # text_splitter가 metadata를 기본 필요로 함

pdf_paths =['Advanced English Conversations.pdf']
pages_text =[]
for pdf_path in pdf_paths:
    with pdfplumber.open(pdf_path)as pdf:
        for page in pdf.pages:
            page_text =page.extract_text()
            if page_text:  # Ensure there is text on the page
                doc =Document(page_text)
                pages_text.append(doc)

# [2.1] text splitter 정의 ---------------------------------------
text_splitter =RecursiveCharacterTextSplitter.from_huggingface_tokenizer(
    tokenizer=AutoTokenizer.from_pretrained("sentence-transformers/all-MiniLM-L12-v2"),
    chunk_size=256,     # 256 chars / chunk
    chunk_overlap=16,   # 이웃하는 chunk사이 16 chars overlap
    strip_whitespace=True,  # chunk 생성시에 공백 제거
)
# [2.2]불러온 페이지에 대하여 text_splitting하여 chunks 구하기
chunks =text_splitter.split_documents(pages_text)
if SHOW_INSIDE :
    print ('\nsplit된 chunk의 갯수: ',len(chunks))
    print('\nchunk[0]의 전체내용: \n',chunks[0])
    print('\nchunk[0]의 page_content: \n',chunks[0].page_content)
# [3] bidirectional encpoder 모델을 이용한 Embedding 정의 ----------------
bi_encoder =HuggingFaceEmbeddings(
    #model_name="sentence-transformers/all-MiniLM-L12-v2", model_kwargs={"device": "cpu"}
    model_name="sentence-transformers/all-MiniLM-L12-v2",model_kwargs={"device":"cuda"}
)
# [4] chunks들로부터 bi_encoder를 이용한 임베딩을 사용하고, dot-product거리를 이용하여 faiss index
생성.
# FAISS (Facebook AI Similarity Search)
faiss_db =FAISS.from_documents(
    chunks,bi_encoder,distance_strategy=DistanceStrategy.DOT_PRODUCT
)
# [5] cross encoder 정의 -----------------------------------------------
cross_encoder =CrossEncoder(
    "cross-encoder/ms-marco-TinyBERT-L-2-v2",max_length=768,device="cuda"
)
prompt0 ="You are an AI helper to help me practice English"
prompt2  ='''You  will  play  a  rile  as  a  native  English  speaker  for  practicing  English
conversation from a textbook.
Your name in the conversation textbook will be one of the two Names in the conversation.
Your name can be changed depending on the conversation sentence set.
In the given conversation texts, a person's name is followded by a colon and the follwong
sentences of the person.
Return the appropriate person's first sentences from the given conversation sets,
which are located just after the user's each new input sentence.
Don't return user's sentences and any extra AI-generated sentences except the your first
answer'''
prompt2  ='''You  will  play  a  rile  as  a  native  English  speaker  for  practicing  English
conversation from a textbook.
Your name in the conversation textbook is 'Charley', and the user's name is Diana.
In the given conversation texts, a person's name is followded by a colon and the follwong
sentences of the person.
Return the appropriate Charley's first sentences from the given conversation sets,
which are located just after the user's each new input sentence (Diana's sentence).
Don't return Diqanan's sentences and any extra AI-generated sentences except the Charley's
first answer'''
prompt_template =ChatPromptTemplate.from_messages (
    [
```

```
        ("system","{system_prompt}"),
        ("human","{input}"),
        #MessagesPlaceholder(variable_name='chat_history'),
    ]
)
    # chat 기록 저장용 리스트
chat_history =[]
    # llm 설정
llm =Ollama(model="llama3",temperature=0.0)       # llama3:70b fine-tuned chat/dialogue model
#llm = ChatOllama(model="llama3:Text",temperature=0.0)     # pre-trained base model
    # chain 설정
chain =prompt_template |llm
TOP_CHUNK_NO =1
ANSWER_SCORE_THRESH =2
chat_queue =Queue(maxsize =4)
#=================================================================================================
while True :

    print("-"*80)
    question =input("User ['q' to quit] : ")

    q =question[:]
    if q.strip().lower()=='q':break
    elif  q =='':continue

    user_prompt =get_all_queue_element(chat_queue)+question
    print("user prompt : ",user_prompt)

    # [5] question과 유사도가 가장 가까운 상위 5개의 chunk를 추출
    retrieved_chunks =faiss_db.similarity_search(user_prompt,k=5)
    reranked_chunks =cross_encoder.rank(
        (user_prompt),
        [doc.page_content for doc in retrieved_chunks],
        top_k=TOP_CHUNK_NO,
        return_documents=True,
    )
    answer_score =reranked_chunks[0]['score']
    print(f"\n\n*** 최종 상위 답변 {TOP_CHUNK_NO}개 문장의 corpus_id, score  : \n")
    for doc in reranked_chunks:
        print('CORPUS_ID: ',doc['corpus_id'],'SCORE: ',doc['score'])
    print("-"*80)
    context ="".join(doc["text"]+"\n\n"for doc in reranked_chunks)
    print("\n*** 최종 상위 답변 문장의 TEXT  : \n",reranked_chunks[0]['text'])
    if SHOW_INSIDE :
        print(f"\n*** RAG에 의한 연관된 상위 {len(reranked_chunks)}개 chunks")
        for doc in reranked_chunks:
            corpus_id =doc["corpus_id"]
            print(f"Chunk {corpus_id}")

    #-------------------------------------------------------------------------------------------

    try :

        if answer_score <=ANSWER_SCORE_THRESH :    # vector store의 유사도 검색결과값이 낮은 경우
            # 새로 대화 시작                              # LLM이 대답하도록 함
            print("n**** New Conversation Starting ***\n")
            with chat_queue.mutex:
                chat_queue.queue.clear()
            chat_history =[]
                                                                                response
=chain.invoke({"input":question,"chat_history":chat_history,"system_prompt":prompt0})
            # 주어진 vector store의 데이터로 대답을 하도록 함
        else :
                                response    =chain.invoke({"input":"user's    sentence:"+question
+"\ntextbook_sentences: "+context,\
                "chat_history":chat_history,"system_prompt":prompt2})
```

```
                    #response   =   chain.invoke({"input":"user's   sentence:"+   question   +
"\ntextbook_sentences: "+context, \
            #         "system_prompt":prompt2})

        print("-"*80,'\n')
        print("*** user_prompt: ",user_prompt,'\n')
           #print("***  AI:  ",response.content,'\n')      #  in  case  of  ChatOllama:  content,
response_metadata, id
        print("*** AI: ",response,'\n')

        chat_history.append(HumanMessage(content=question))
        chat_history.append(AIMessage(content=response))

        if (chat_queue.empty()):
            chat_queue.put(question)
        elif (chat_queue.full()):
            chat_queue.get()

        if answer_score <=ANSWER_SCORE_THRESH :
            pass
        else :
            if '"'and '\n'in response :
                answer =response[response.find('"'):response.rfind('"')+1]
            else :
                answer =response[:]
            print("answer: ",answer)
            chat_queue.put(answer)
            #chat_queue.put(response.content)   # ChatOllama

    except Exception as e:
        print("Error : ",e)
```

※ 질문의 출처:

Set (39) - I'm not that strong-willed!

Dialogue

Diana : This smell **stinks** ! Oops! I've forgot to put the food in the fridge. It's **rotten** .
Charley : **Good for you** ! Just **fix** anything. I've **lost my appetite** , anyway.
Diana : Oh dear; I'll make it up for you. I promise.
Charley : Alright, let's **eat out on second thought** .
Diana : But I'm **on a diet** . I'm trying to **lose weight** and I can be easily tempted. You know I'm not that **strong-willed** !
Charley : **Enough already** ! My **head is spinning.** I just need to **grab a bite** .

Vocabulary

Stink: to have a strong unpleasant smell.
Rotten : (adj) decomposing or decaying; putrid; tainted, foul, or bad-smelling.
Good for you ! Well done (sarcastic meaning; the speaker is not impressed)
Lose one's appetite : to no longer feel hungry.
Eat out : to eat in a restaurant.
On second thought : resulting from a revised opinion or change of mind.
On a diet : following a specific nutritional plan.
Lose weight : to become thinner.
Strong-willed : (adj) determined to do as one wants.
Enough already : used to indicate unwillingness to tolerate any more of something undesirable.
Somebody's head is spinning : to feel as if they might faint.
Grab a bite : To get something to eat.

5.6 실행결과 (RAG using PDFs and local LLM)

기본 llama3 모델(chatiin에 fine tuning된 모델) 사용시 :

ollama pull llama3

또는, ollama3:text 모델 사용시에는 아래처럼 실행


```
(rag) D:\myprojects\nlp\rag>python rag_pdf_ollama.py
--------------------------------------------------------------------------------
User ['q' to quit] : This smell stinks ! Oops! I've forgot to put the food in the fridge.
user prompt :  This smell stinks ! Oops! I've forgot to put the food in the fridge.


*** 최종 상위 답변 1개 문장의 corpus_id, score  :

CORPUS_ID:  4  SCORE:  7.905603
--------------------------------------------------------------------------------

*** 최종 상위 답변 문장의 TEXT  :
 Set (39) - I'm not that strong-willed!
Dialogue
Diana : This smell stinks ! Oops! I've forgot to put the food in the fridge. It's
rotten .
Charley : Good for you ! Just fix anything. I've lost my appetite , anyway.
Diana : Oh dear; I'll make it up for you. I promise.
Charley : Alright, let's eat out on second thought .
Diana : But I'm on a diet . I'm trying to lose weight and I can be easily
tempted. You know I'm not that strong-willed !
Charley : Enough already ! My head is spinning. I just need to grab a bite .
Vocabulary
Stink: to have a strong unpleasant smell.
Rotten : (adj) decomposing or decaying; putrid; tainted, foul, or bad-smelling.
Good for you ! Well done (sarcastic meaning; the speaker is not impressed)
--------------------------------------------------------------------------------

*** user_prompt:  This smell stinks ! Oops! I've forgot to put the food in the fridge.

*** AI:  I'm not that strong-willed!

answer:  I'm not that strong-willed!
--------------------------------------------------------------------------------
User ['q' to quit] : Oh dear; I'll make it up for you. I promise.
user prompt :  This smell stinks ! Oops! I've forgot to put the food in the fridge.
I'm not that strong-willed!
Oh dear; I'll make it up for you. I promise.


*** 최종 상위 답변 1개 문장의 corpus_id, score  :

CORPUS_ID:  0  SCORE:  8.979391
--------------------------------------------------------------------------------

*** 최종 상위 답변 문장의 TEXT  :
 Set (39) - I'm not that strong-willed!
Dialogue
Diana : This smell stinks ! Oops! I've forgot to put the food in the fridge. It's
rotten .
Charley : Good for you ! Just fix anything. I've lost my appetite , anyway.
Diana : Oh dear; I'll make it up for you. I promise.
Charley : Alright, let's eat out on second thought .
```

Diana : But I'm on a diet . I'm trying to lose weight and I can be easily
tempted. You know I'm not that strong-willed !
Charley : Enough already ! My head is spinning. I just need to grab a bite .
Vocabulary
Stink: to have a strong unpleasant smell.
Rotten : (adj) decomposing or decaying; putrid; tainted, foul, or bad-smelling.
Good for you ! Well done (sarcastic meaning; the speaker is not impressed)

*** user_prompt: This smell stinks ! Oops! I've forgot to put the food in the fridge.
I'm not that strong-willed!
Oh dear; I'll make it up for you. I promise.

*** AI: Here's my response as Charley:

Alright, let's eat out on second thought.

answer:

User ['q' to quit] : But I'm on a diet
user prompt : Oh dear; I'll make it up for you. I promise.

But I'm on a diet

*** 최종 상위 답변 1개 문장의 corpus_id, score :

CORPUS_ID: 0 SCORE: 4.685632

*** 최종 상위 답변 문장의 TEXT :
 Set (39) - I'm not that strong-willed!
Dialogue
Diana : This smell stinks ! Oops! I've forgot to put the food in the fridge. It's
rotten .
Charley : Good for you ! Just fix anything. I've lost my appetite , anyway.
Diana : Oh dear; I'll make it up for you. I promise.
Charley : Alright, let's eat out on second thought .
Diana : But I'm on a diet . I'm trying to lose weight and I can be easily
tempted. You know I'm not that strong-willed !
Charley : Enough already ! My head is spinning. I just need to grab a bite .
Vocabulary
Stink: to have a strong unpleasant smell.
Rotten : (adj) decomposing or decaying; putrid; tainted, foul, or bad-smelling.
Good for you ! Well done (sarcastic meaning; the speaker is not impressed)

*** user_prompt: Oh dear; I'll make it up for you. I promise.

But I'm on a diet

*** AI: Enough already !

answer: Enough already !

User ['q' to quit] :

※ [Practical Consideration]

RAG를 잘 만들기 위해서는 다음 요소가 잘 고려되어야 한다

1. RAG 기능(FewShotPrompting)을 잘 지원하는 LLM의 선정
2. 해결하고자 하는 TASK에 맞는 좋은 성능을 낼 수 있는 RAG용 Prompt의 작성
3. 좋은 품질의 RAG용 원천 **외부 데이터**의 선정
4. 해당 TASK에 잘 동작하도록 **외부데이터의 정제/가공**

※ [Questions]

1. RAG와 Fine-tuning의 차이점을 생각해 보고, 이 두 가지 방법론을 조합할 수 있을지도 생각해보라

2. RAG 의 Indexing의 네 가지 단계가 무엇인지 설명하고, 각 단계에서 유의해야 할 것들이 어떤 것이 있을지 생각해보라. 특히 영어가 아닌 한국어와 중국어가 처리 대상인 경우에 대하여 생각해 보라

3. naive RAG 의 단점에 대하여 논하라

4. RAG의 개선방향에 대하여 논하라

※ [실습 문제]

5. Ollama툴을 이용하여 로컬 머신에 local sLLM(예: llama3)을 다운받고, local llm을 사용하여 Rag를 구현하라.

6. 특정 영어회화 책(.pdf)을 내용으로 하는 영어회화 연습기능을 RAG로 구현해 보라.
Open Source STT(Speech-to-Text)와 TTS(Text-to-Speech)를 이용하라