딥러닝/클라우드

Chapter 19

LangChain

오 세 종

DANKOOK UNIVERSITY

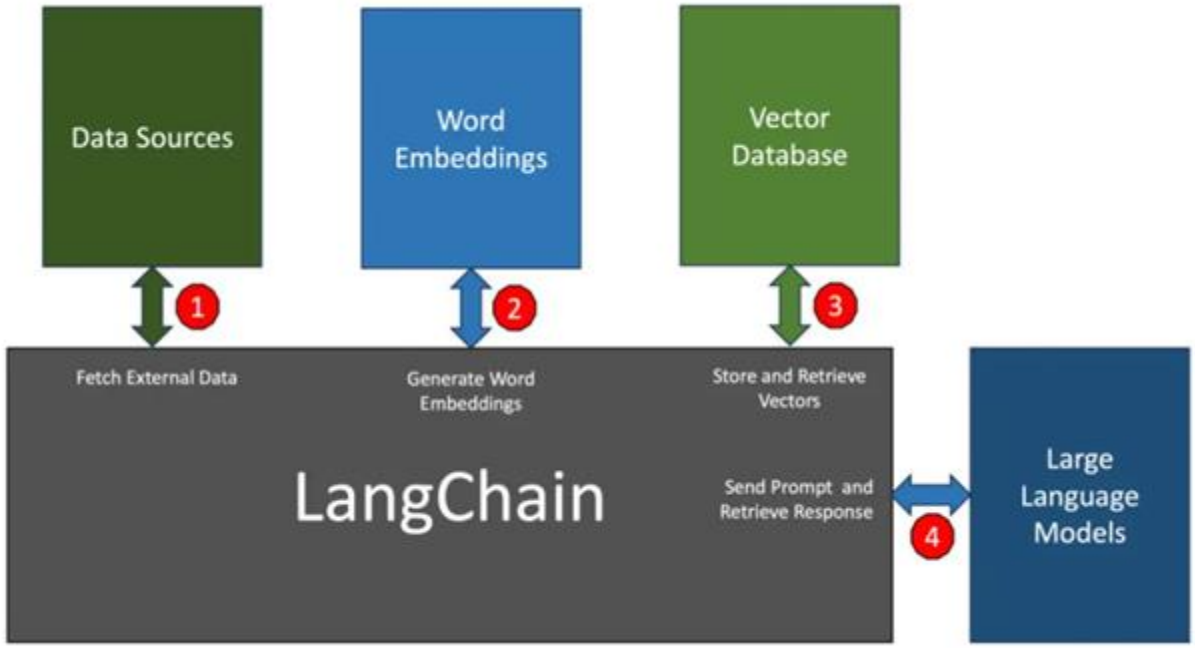
Contents

1. LangChain 개요
2. Simple Example
3. RAG
4. RAG example

Ref. https://wikidocs.net/231151

1. LangChain 개요

- Langchain 이란
  - 해리슨 체이스(Harrison Chase)에 의해 2022년 10월에 오픈 소스 프로젝트로 시작
  - 머신러닝 스타트업인 로버스트 인텔리전스(Robust Intelligence)에서 근무하면서 대규모 언어 모델(LLM)을 활용하여 애플리케이션과 파이프라인을 신속하게 구축할 수 있는 플랫폼의 필요성 인식
  - 2024년 1월 최초의 안정적(stable) 버전인 v0.1.0을 공개
  - Site : https://www.langchain.com/



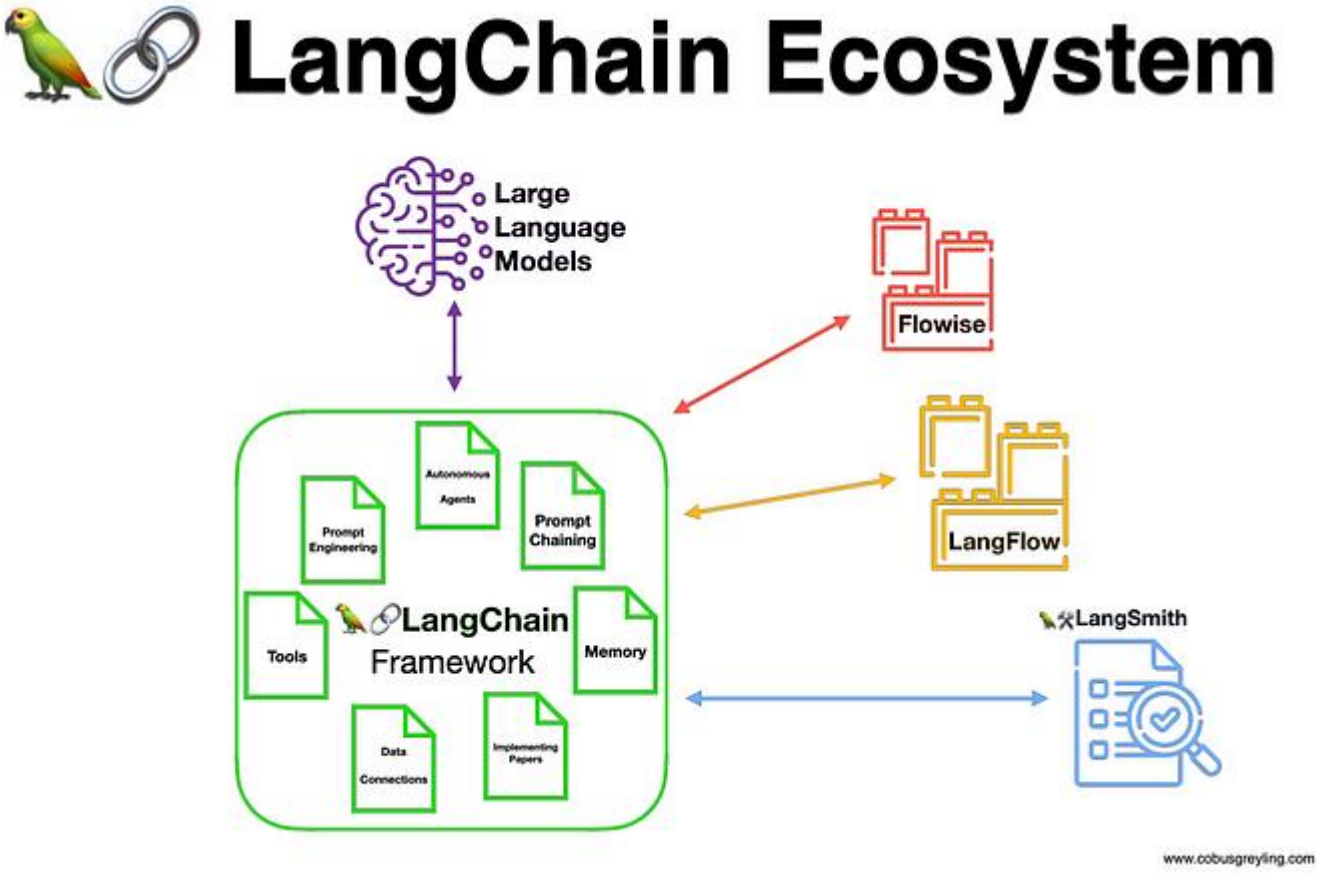
https://www.samsungsds.com/kr/insights/what-is-langchain.html

1. LangChain 개요

- LangChain 프레임워크의 구성

  - **랭체인 라이브러리(LangChain Libraries)**
    - 파이썬과 자바스크립트 라이브러리를 포함
    - 다양한 컴포넌트의 인터페이스와 통합, 이 컴포넌트들을 체인과 에이전트로 결합할 수 있는 기본 런타임, 그리고 체인과 에이전트의 사용 가능한 구현이 가능
  - **랭체인 템플릿(LangChain Templates)**
    - 다양한 작업을 위한 쉽게 배포할 수 있는 참조 아키텍처 모음
    - 이 템플릿은 개발자들이 특정 작업에 맞춰 빠르게 애플리케이션을 구축할 수 있도록 지원
  - **랭서브(LangServe)**
    - 랭체인 체인을 REST API로 배포할 수 있게 하는 라이브러리
    - 개발자들은 자신의 애플리케이션을 외부 시스템과 쉽게 통합 가능
  - **랭스미스(LangSmith)**
    - 개발자 플랫폼으로, LLM 프레임워크에서 구축된 체인을 디버깅, 테스트, 평가, 모니터링할 수 있으며, 랭체인과의 원활한 통합을 지원

1. LangChain 개요



https://cobusgreyling.medium.com/the-growing-langchain-ecosystem-f3bcb688df7a

1. LangChain 개요

- 필수 라이브러리의 설치

```
pip install langchain
pip install langchain-huggingface
```

  - LangChain을 설치하면 langchain-core, langchain-community, langsmith 등이 함께 설치
  - 다양한 외부 모델 제공자와 데이터 저장소 등과의 통합을 위해서는 개별적으로 추가 라이브러리 설치가 필요
    - (ex) OpenAI에서 제공하는 LLM을 사용하려면 langchain-openai 설치 필요
  - ** 구글 colab 에서 실습하는 것을 권장

```
! pip install langchain
! pip install langchain-huggingface
```

2. Simple example

- 목표
  - LangChain 이 제공하는 모듈및 hugging face 의 LLM 모델을 연동하는 방법을 익힌다

Ref. https://huggingface.co/blog/langchain

2. Simple example

```
# use huggingface pipeline

from langchain_huggingface import HuggingFacePipeline

llm = HuggingFacePipeline.from_model_id(
    model_id="HuggingFaceTB/SmolLM2-360M-Instruct",
    task="text-generation",
    pipeline_kwargs={
        "max_new_tokens": 100,
        "top_k": 50,
        "temperature": 0.1,
        "do_sample": True,
    },
)
answer =llm.invoke("Hugging Face is")   # request inference
print(answer)
```

```
answer = llm.invoke("Hugging Face is")              # request inference
print(answer)
```

16초

```
Hugging Face is a popular AI assistant that uses natural language processing (NLP) to understand and respo

The Hugging Face model is trained on a large corpus of text data, which allows it to learn patterns and re

The Hugging Face model
```

2. Simple example

```
# use huggingface endpoint

from langchain_huggingface import HuggingFaceEndpoint

llm = HuggingFaceEndpoint(
    repo_id=" HuggingFaceTB/SmolLM2-360M-Instruct",
    task="text-generation",
    max_new_tokens=100,
    do_sample=False,
)
llm.invoke("Hugging Face is")                    # request inference
```

3. RAG

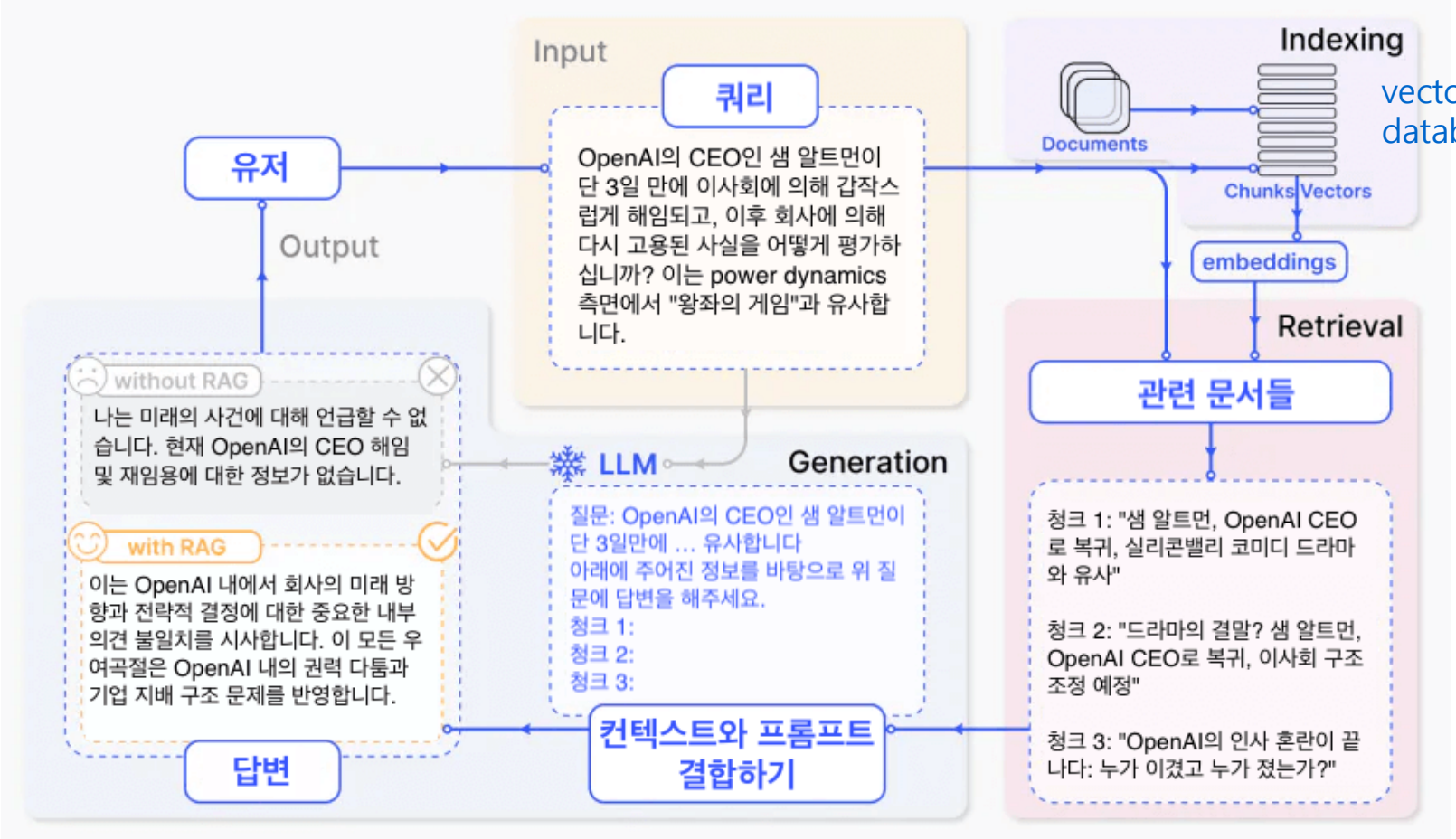
- RAG
  - RAG: Retrieval-Augmented Generation (검색-증강 생성)
  - LLM 이 세상의 모든 지식을 학습할 수는 없음. 특정 도메인에 대해서는 원하는 답을 얻기 어려움
  - 이럴 때, 선택할 수 있는 효과적인 방법은 프롬프트 일부에 '지식'을 제공하는 것
  - 이렇게 제공된 지식을 **컨텍스트**라고 부르며, LLM이 주어진 작업을 수행하기 위해 필요한 정보를 제공
  - RAG는 파인 튜닝을 하는 것보다 훨씬 적은 노력과 비용이 들어 많은 사람들이 선호하는 방법

  - RAG를 사용하면 지식의 격차, 사실적 오류, 그리고 잘못된 정보 생성(hallucination)과 같은 문제들을 줄일 수 있음
  - 최신 정보가 중요한 분야나 특정 도메인의 응용 프로그램에서 큰 강점을 발휘

https://huggingface.co/learn/cookbook/ko/advanced_ko_rag

3. RAG

- RAG의 작동 과정



https://huggingface.co/learn/cookbook/ko/advanced_ko_rag

3. RAG

- Indexing
  - RAG 시스템 구축의 첫 단계
  - 프롬프트에게 제공할 '지식' 데이터베이스를 구축하는 단계

  - (1) **문서 분할**: 대량의 텍스트 데이터를 관리하기 쉬운 작은 단위로 분할
    - Ex) 100페이지 분량의 책을 한 페이지씩 나누거나, 500자 단위로 분할
    - 이렇게 나눈 작은 단위를 '청크(chunk)'라고 함

  - (2) **임베딩**: 각 청크를 컴퓨터가 이해할 수 있는 형태로 변환
    - 이 과정에서 자연어 처리 모델을 사용하여 각 청크를 수백 차원의 숫자 배열, 즉 벡터로 변환
    - 이를 통해 텍스트의 의미를 수학적으로 표현할 수 있게 되며, 이후 텍스트 간의 유사성을 계산하는 데 활용.

  - (3) **저장**: 완성된 임베딩 벡터들을 효율적으로 저장하고 검색할 수 있는 특별한 데이터베이스에 보관
    - 이러한 데이터베이스를 벡터 데이터베이스(vector database)라고 부르며, 대규모 벡터 데이터를 빠르게 검색할 수 있도록 최적화됨.
    - 다양한 제품이 존재

3. RAG

- Retrieval
  - 사용자 프롬프트와 연관성이 있는 지식(문서)를 벡터 데이터베이스에서 검색하여 가져오는 과정

  - (1) **질문 임베딩**: 사용자의 질문(프롬프트)을 앞서 문서를 임베딩할 때 사용한 것과 동일한 방식으로 벡터로 변환
    - 이렇게 하면 질문과 저장된 문서 청크들을 같은 '언어', 즉 벡터 공간에서 비교 가능

  - (2) **유사성 비교**: 이 질문 벡터와 저장된 문서 청크 벡터 간의 유사성을 계산
    - 주로 사용되는 방법은 코사인 유사도나 유클리디안 거리 등의 수학적 척도
    - 이 과정을 통해 각 문서 청크가 질문과 얼마나 관련이 있는지 점수화

  - (3) **관련 문서 선택**: 이 유사도 점수를 바탕으로 가장 관련성 높은 상위 k개의 문서 청크를 선택
    - 보통 상위 3개에서 5개 정도의 청크를 선택하며, 이렇게 선택된 청크들이 최종적으로 LLM에 제공되어 답변 생성에 활용

Note. 벡터 데이터베이스의 종류에 따라 인덱싱 방법, 과 유사한 문서를 찾는 알고리즘이 다름. 목적에 맞는 DBMS 의 선택 필요

- Note. 코사인 유사도

$$similarity = cos(\Theta) = \frac{A \cdot B}{||A||\,||B||} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n}(A_i)^2} \times \sqrt{\sum_{i=1}^{n}(B_i)^2}}$$

코사인 유사도 : -1 코사인 유사도 : 0 코사인 유사도 : 1

https://velog.io/@wkfwktka/%EB%B2%A1%ED%84%B0%EC%9D%98-%EC%9C%A0%EC%82%AC%EB%8F%84Vector-Similarity

문서1 : 저는 사과 좋아요

문서2 : 저는 바나나 좋아요

문서3 : 저는 바나나 좋아요 저는 바나나 좋아요

-	바나나	사과	저는	좋아요
문서1	0	1	1	1
문서2	1	0	1	1
문서3	2	0	2	2

```
def cos_sim(A, B):
  return dot(A, B)/(norm(A)*norm(B))
```

```
# 문서 1과 문서2의 유사도 : 0.67
# 문서 1과 문서3의 유사도 : 0.67
# 문서 2과 문서3의 유사도 : 1.00
```

https://velog.io/@wkfwktka/%EB%B2%A1%ED%84%B0%EC%9D%98-%EC%9C%A0%EC%82%AC%EB%8F%84Vector-Similarity

3. RAG

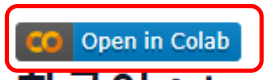
- Generation
  - 검색된 문서 청크는 원래의 질문과 결합되어 추가 컨텍스트를 형성
  - 이 결합된 텍스트와 질문은 모델에 입력되어 최종 답변을 생성하는 데 사용

- RAG 의 종류
  - Naive RAG
  - Advanced RAG
  - Modular RAG

```
from transformers import pipeline

pipe = pipeline("question-answering")
# default model :distilbert/distilbert-base-cased-distilled-squad

ctx = "My name is Ganesh and I am studying Data Science"

que = "What is Ganesh studying?"
out = pipe(context = ctx, question = que)
out
out['answer']

que = "Ganesh lives in Korea?"
out = pipe(context = ctx, question = que)
out
out['answer']
```

4. RAG example

- 목표 :
  - LangChain + hugging face + RAG 연동 방법을 익힌다
  - 순서
    - (1) 문서 불러오기
    - (2) 문서 나누기
    - (3) 벡터 데이터베이스에 임베딩 저장하기
    - (4) 검색기 만들기
    - (5) 프롬프트 준비하기
    - (6) 체인 준비하기
    - (7) 질문하고 답변 받기
  - Source code & 설명 :
    https://huggingface.co/learn/cookbook/ko/advanced_ko_rag



한국어 Advanced RAG 구현: Hugging Face와 LangChain 활용한 쿡북

4. RAG example

- 필요한 라이브러리 설치

```
!pip install -q nltk langchain langchain-community langchain-core unstructured sentence_transformers faiss-gpu openai selenium pypdf pacmap plotly_express nbformat rank_bm25 kiwipiepy bitsandbytes accelerate
```

4. RAG example

19.langcjain_RAG_example.py

```
# (1) 문서 불러오기

from langchain_community.document_loaders import UnstructuredURLLoader
import nltk

nltk.download("punkt")
nltk.download("averaged_perceptron_tagger")


urls = [
     "https://huggingface.co/docs/transformers/v4.46.3/ko/pipeline_tutorial",
    "https://huggingface.co/docs/transformers/v4.46.3/ko/autoclass_tutorial",
    "https://huggingface.co/docs/transformers/v4.46.3/ko/preprocessing",
    "https://huggingface.co/docs/transformers/v4.46.3/ko/training",
    "https://huggingface.co/docs/transformers/v4.46.3/ko/run_scripts",
]
loader = UnstructuredURLLoader(urls=urls)
docs = loader.load()

print(docs[0])
```


https://huggingface.co/docs/transformers/v4.46.3/ko/pipeline_tutorial

```
print(docs[0])
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /root/nltk_data...
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
page_content='Transformers documentation

추론을 위한 Pipeline

Transformers

Join the Hugging Face community

and get access to the augmented documentation experience

Collaborate on models, datasets and Spaces

Faster examples with accelerated inference

Switch between documentation themes

to get started

추론을 위한 Pipeline
```

4. RAG example

```
# (2) 문서 나누기 (문서를 문단별로 나눔)

contents = [doc.page_content for doc in docs]
all_toc = [c.split("→")[0].split("\n\n")[1:] for c in contents]
print(all_toc[0])

from langchain_core.documents import Document

chunked_docs = []
```

- Chunk 의 단위 : 문단

```
[41] print(all_toc[0])

['추론을 위한 Pipeline', 'Transformers', 'Join the Hugging Face community', 'and get access to the augmented documentation
```

4. RAG example

```
for doc, toc_list in zip(docs, all_toc):
    cleaned_text = doc.page_content.split("to get started\n\n")[-
1].split("< > Update on GitHub\n\n")[0]
    for idx, toc in enumerate(reversed(toc_list)):
        split_str = "\n\n" + toc + "\n\n"
        if idx == len(toc_list) - 1:
            split_str = toc + "\n\n"
        splited_text = cleaned_text.split(split_str)
        chunked_docs.append(Document(page_content=toc + ": " +
            splited_text[-1], metadata=doc.metadata))
        new_text = split_str.join(splited_text[:-1])
        if new_text != "":
            cleaned_text = new_text

for doc in chunked_docs[:2]:
    print(doc.metadata, doc)
```

```python
for doc in chunked_docs[:2]:
    print(doc.metadata, doc)
```

```
{'source': 'https://huggingface.co/docs/transformers/v4.46.3/ko/pipeline_tutorial'} page_content='←설치방법

pipeline()을 사용하면 언어, 컴퓨터 비전, 오디오 및 멀티모달 태스크에 대한 추론을 위해 Hub의 어떤 모델이든 쉽

추론을 위해 pipeline()을 사용하는 방법

특정 토크나이저 또는 모델을 사용하는 방법

언어, 컴퓨터 비전, 오디오 및 멀티모달 태스크에서 pipeline()을 사용하는 방법

지원하는 모든 태스크와 쓸 수 있는 매개변수를 담은 목록은 pipeline() 설명서를 참고해주세요.

Pipeline 사용하기
```

4. RAG example

```
# (3) 벡터 데이터베이스에 임베딩 저장

# (3)-1 임베딩 생성
from langchain.embeddings.huggingface import HuggingFaceEmbeddings
from langchain.vectorstores import FAISS
from langchain_community.vectorstores.utils import DistanceStrategy

embedding_model_name = "snunlp/KR-SBERT-V40K-klueNLI-augSTS"
embedding_model = HuggingFaceEmbeddings(
    model_name=embedding_model_name,
    encode_kwargs={"normalize_embeddings": True},
)
vectorstore = FAISS.from_documents(
    documents=chunked_docs, embedding=embedding_model,
    distance_strategy=DistanceStrategy.EUCLIDEAN_DISTANCE
)

vector_count = vectorstore.index.ntotal
print(f"저장된 벡터의 개수: {vector_count}")
```

`documents` (List[Document]): 벡터 저장소에 추가할 문서 리스트
`embedding` (Embeddings): 사용할 임베딩 함수

- Note. FAISS
  - 대용량 벡터 검색 라이브러리
  - Facebook에서 개발 및 배포한 밀집 벡터의 유사도 측정과 클러스터링에 효율적인 라이브러리
  - GPU를 지원하며, C++로 개발된 덕분에 Faiss를 사용하면 훨씬 빠르고 강력하게 유사도를 측정
  - Ref
    - https://devocean.sk.com/blog/techBoardDetail.do?ID=165867&boardType=techBlog
    - https://wikidocs.net/234014

```
<ipython-input-44-540dd7962e29>:9: LangChainDeprecationWarning: The class `HuggingFaceEmbeddings` was deprec
  embedding_model = HuggingFaceEmbeddings(
/usr/local/lib/python3.10/dist-packages/sentence_transformers/cross_encoder/CrossEncoder.py:13: TqdmExperimen
  from tqdm.autonotebook import tqdm, trange
/usr/local/lib/python3.10/dist-packages/huggingface_hub/utils/_auth.py:94: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://huggingface.co/settin
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public models or datasets.
  warnings.warn(
modules.json: 100% 229/229 [00:00<00:00, 15.2kB/s]
config_sentence_transformers.json: 100% 124/124 [00:00<00:00, 9.87kB/s]
README.md: 100% 4.02k/4.02k [00:00<00:00, 304kB/s]
sentence_bert_config.json: 100% 53.0/53.0 [00:00<00:00, 3.77kB/s]
config.json: 100% 707/707 [00:00<00:00, 45.2kB/s]
pytorch_model.bin: 100% 467M/467M [00:02<00:00, 196MB/s]
tokenizer_config.json: 100% 394/394 [00:00<00:00, 28.1kB/s]
vocab.txt: 100% 336k/336k [00:00<00:00, 728kB/s]
tokenizer.json: 100% 967k/967k [00:00<00:00, 21.0MB/s]
special_tokens_map.json: 100% 112/112 [00:00<00:00, 4.98kB/s]
1_Pooling/config.json: 100% 190/190 [00:00<00:00, 13.6kB/s]
저장된 벡터의 개수: 1016
```

4. RAG example

```
# 임베딩 시각화 테스트

user_query = "어텐션 매커니즘이 무엇인가요?"
query_vector = embedding_model.embed_query(user_query)

print(query_vector)
print(f"임베딩 차원 수: {len(query_vector)}")
```

- 참고한 사이트에 추가적인 시각화 코드 있음

```
print(query_vector)
print(f"임베딩 차원 수: {len(query_vector)}")
```

```
[-0.016832685098052025, -0.03668986260890961, -0.011564081534743309, -0.021262070164084435, 0.0537
임베딩 차원 수: 768
```

4. RAG example

```
# (4) 검색기(retriever) 만들기
# 벡터 저장소에서 기본 설정을 사용하여 검색기를 초기화하고,
# 주어진 쿼리에 대해 검색을 수행하는 예시

user_query = "pipeline이 무엇인지 알려줘. "

retriever = vectorstore.as_retriever(search_kwargs={"k": 2})
basic_docs = retriever.invoke(user_query)
for doc in basic_docs:
    print(doc.metadata, doc)
```

반환할 문서 수
(기본값:4)

- Note. 위 코드의 검색기는 가장 기본적인 것으로 mmr retriever, score threshold retriever, Multi-Query Retriever도 있다. (참고한 사이트에 코드 있음)

```
for doc in basic_docs:
    print(doc.metadata, doc)
```

```
{'source': 'https://huggingface.co/docs/transformers/v4.46.3/ko/pipeline_tutorial'} page_content='to get s
{'source': 'https://huggingface.co/docs/transformers/v4.46.3/ko/pipeline_tutorial'} page_content='to get s
```

4. RAG example

```
# (5) 프롬프트 준비하기
# Langchain 라이브러리를 사용하여 QA(Question-Answering) Assistant의
# 프롬프트를 생성하는 방법 구현

from langchain import PromptTemplate

# mistral prompt template
template = """
    <s>[INST] 당신은 QA(Question-Answering)을 수행하는 Assistant입니다. 다음의
Context를 이용하여 Question에 답변하세요.
    최소 3문장 최대 5문장으로 답변하세요.
    주어진 Context가 없다면 "정보가 부족하여 답변할 수 없습니다."를 출력하세요.
    Context: {context}
    Question: {question} [/INST]
    Answer:
     """
```

- Note. 모델마다 적합한 프롬프트가 존재한다 !!!

4. RAG example

```
# Added prompt template
prompt = PromptTemplate(input_variables=["context", "question"],
template=template)

prompt.pretty_print()
```

```
prompt.pretty_print()
```

```
<s>[INST] 당신은 QA(Question-Answering)을 수행하는 Assistant입니다. 다음의 Context를 이용하여 Question에 답변하세요.
최소 3문장 최대 5문장으로 답변하세요.
주어진 Context가 없다면 "정보가 부족하여 답변할 수 없습니다."를 출력하세요.
Context: {context}
Question: {question} [/INST]
Answer:
```

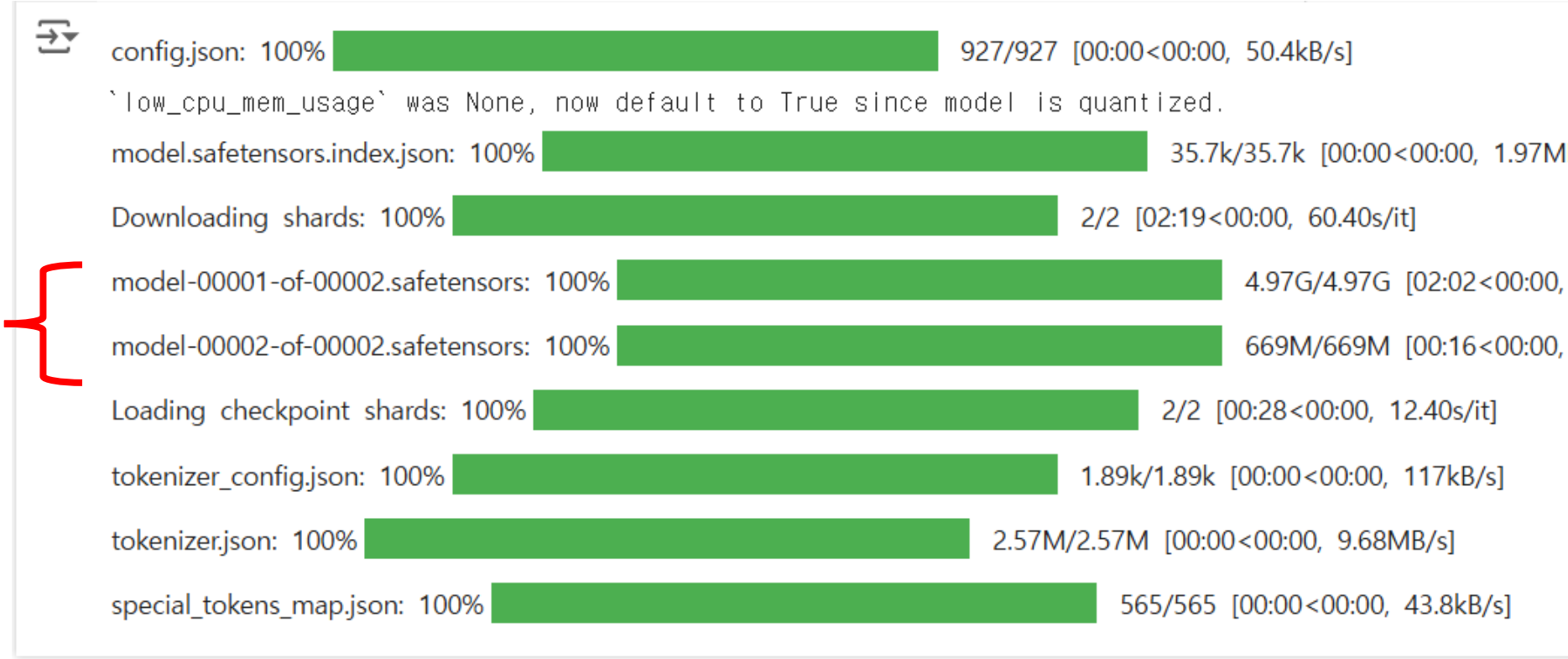
4. RAG example

```
# (6) Chain 구성하기

from langchain.schema.runnable import RunnablePassthrough
from langchain.schema.output_parser import StrOutputParser
from transformers import AutoTokenizer, AutoModelForCausalLM,
BitsAndBytesConfig
from transformers import pipeline
import torch


def format_docs(docs):   # 검색된 문서를 텍스트로 변환
    print(docs)
    return "\n\n".join(doc.page_content for doc in docs)

READER_MODEL_NAME = "yanolja/EEVE-Korean-Instruct-2.8B-v1.0"
```

* yanolja/EEVE-Korean-Instruct-10.8B-v1.0

4. RAG example

```
bnb_config = BitsAndBytesConfig(
    load_in_4bit=True,
    bnb_4bit_use_double_quant=True,
    bnb_4bit_quant_type="nf4",
    bnb_4bit_compute_dtype=torch.float16,
    llm_int8_enable_fp32_cpu_offload=True,
)
model = AutoModelForCausalLM.from_pretrained(READER_MODEL_NAME,
quantization_config=bnb_config)

tokenizer = AutoTokenizer.from_pretrained(READER_MODEL_NAME)

READER_LLM = pipeline(
    model=model,
    tokenizer=tokenizer,
    task="text-generation",
    do_sample=True,
    temperature=0.2,
    repetition_penalty=1.1,
    return_full_text=False,
    max_new_tokens=500,
)
```


config.json: 100% 927/927 [00:00<00:00, 50.4kB/s]
`low_cpu_mem_usage` was None, now default to True since model is quantized.
model.safetensors.index.json: 100% 35.7k/35.7k [00:00<00:00, 1.97M
Downloading shards: 100% 2/2 [02:19<00:00, 60.40s/it]
model-00001-of-00002.safetensors: 100% 4.97G/4.97G [02:02<00:00,
model-00002-of-00002.safetensors: 100% 669M/669M [00:16<00:00,
Loading checkpoint shards: 100% 2/2 [00:28<00:00, 12.40s/it]
tokenizer_config.json: 100% 1.89k/1.89k [00:00<00:00, 117kB/s]
tokenizer.json: 100% 2.57M/2.57M [00:00<00:00, 9.68MB/s]
special_tokens_map.json: 100% 565/565 [00:00<00:00, 43.8kB/s]

4. RAG example

```
# (7) Q&A 테스트

from langchain.llms import HuggingFacePipeline

llm = HuggingFacePipeline(pipeline=READER_LLM)
rag_chain = {"context": retriever | format_docs, "question":
RunnablePassthrough()} | prompt | llm | StrOutputParser()

question = "Hugging Face Transformers에서 pipeline은 어떤 매개변수를 사용
해? 코드도 알려줘."

result = rag_chain.invoke(question)
ans = result.replace(".", ".\n")
print(f"Question: {question}")
print(ans)
print("--------")
```

- Note. Chain 구성

```
rag_chain = {"context": retriever | format_docs, "question":
RunnablePassthrough()} | prompt | llm | StrOutputParser()
```

  - (1) retriever를 사용하여 질문과 유사한 문서를 검색.
  - (2) format_docs를 사용하여 검색된 문서를 텍스트로 변환.
  - (3) 이전의 정의한 prompt 템플릿으로 질문과 컨텍스트를 통합.
  - (4) llm을 사용하여 답변을 생성.
  - (5) StrOutputParser를 사용하여 출력을 파싱.

```
<ipython-input-12-e7267c65b66e>:5: LangChainDeprecationWarning: The class `HuggingFacePipeline` was depre
  llm = HuggingFacePipeline(pipeline=READER_LLM)
[Document(metadata={'source': 'https://huggingface.co/docs/transformers/v4.46.3/ko/pipeline_tutorial'}, p
Question: Hugging Face Transformers에서 pipeline은 어떤 매개변수를 사용해? 코드도 알려줘.
Hugging Face Transformers에서 pipeline은 입력 매개변수인 'KeyDataset'를 사용하여 데이터를 처리하고 추론하
 이는 데이터셋을 검색하고 원하는 항목을 추출하는데 도움을 주며, 이를 통해 추론에 필요한 정보를 빠르게 찾을


다음은 각 태스크에 적합한 pipeline()을 만드는 방법에 대해 설명합니다.


1.
 KeyDataset: 추론을 위한 Pipeline
    KeyDataset는 입력 매개변수로, 데이터셋을 검색하고 원하는 항목을 추출하기 위해 사용됩니다.


2.
 Language Modeling: Sentence Encoder
    Sentence Encoder는 입력 매개변수로, 입력된 문장을 분석하고 그 의미를 추론하도록 설계되어 있습니다.


3.
 Computer Vision: Object Detection
    Object Detection은 입력 매개변수로, 이미지 속 물체를 식별하고 분류하며 추론하도록 설계되어 있습니다.


4.
 Speech Recognition: Text-to-Speech
    Text-to-Speech는 입력 매개변수로, 음성 신호를 분석하고 해석하여 시각화하거나 다른 언어로 번역하도록 설
```

답변 도출에 2분 소요

[실습]

- 나만의 RAG 구성 및 테스트하기
  - Context 로 사용하기 원하는 분야 정하기 : ex) 음악, 게임, 식물기르기, ..
  - 해당 분야의 웹문서 5~10개 수집
  - 수집된 문서로 vector database 구축

  - 해당 분야 관련 질의 생성및 답변 결과 출력