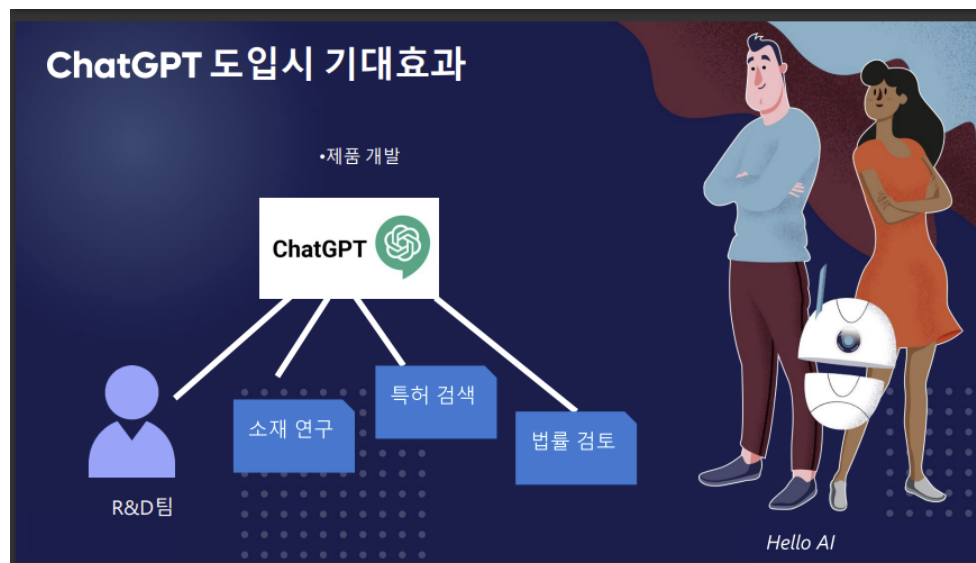
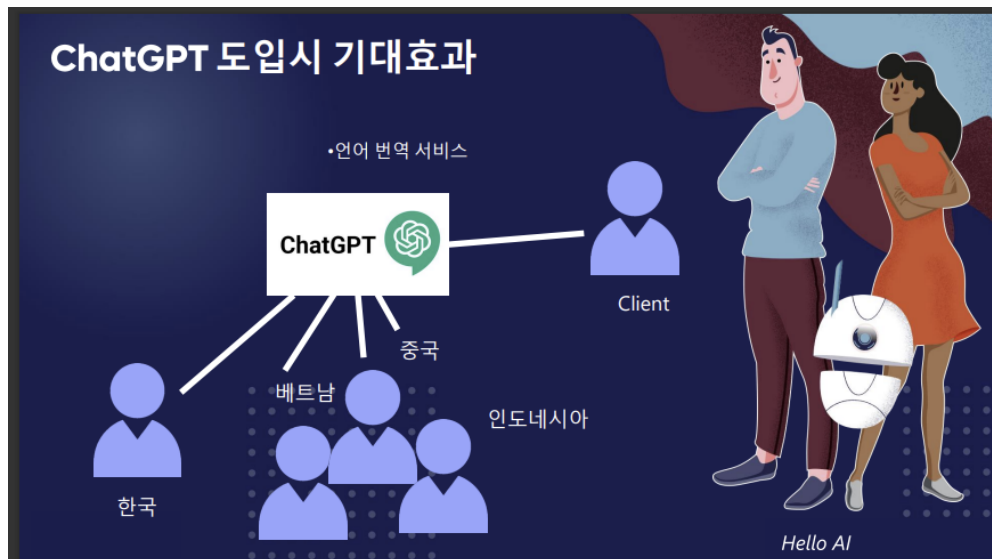
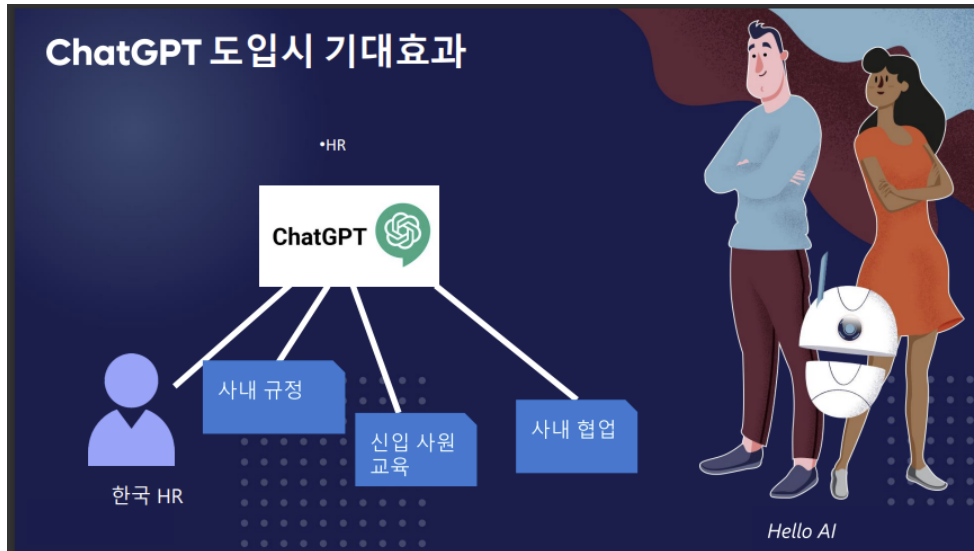


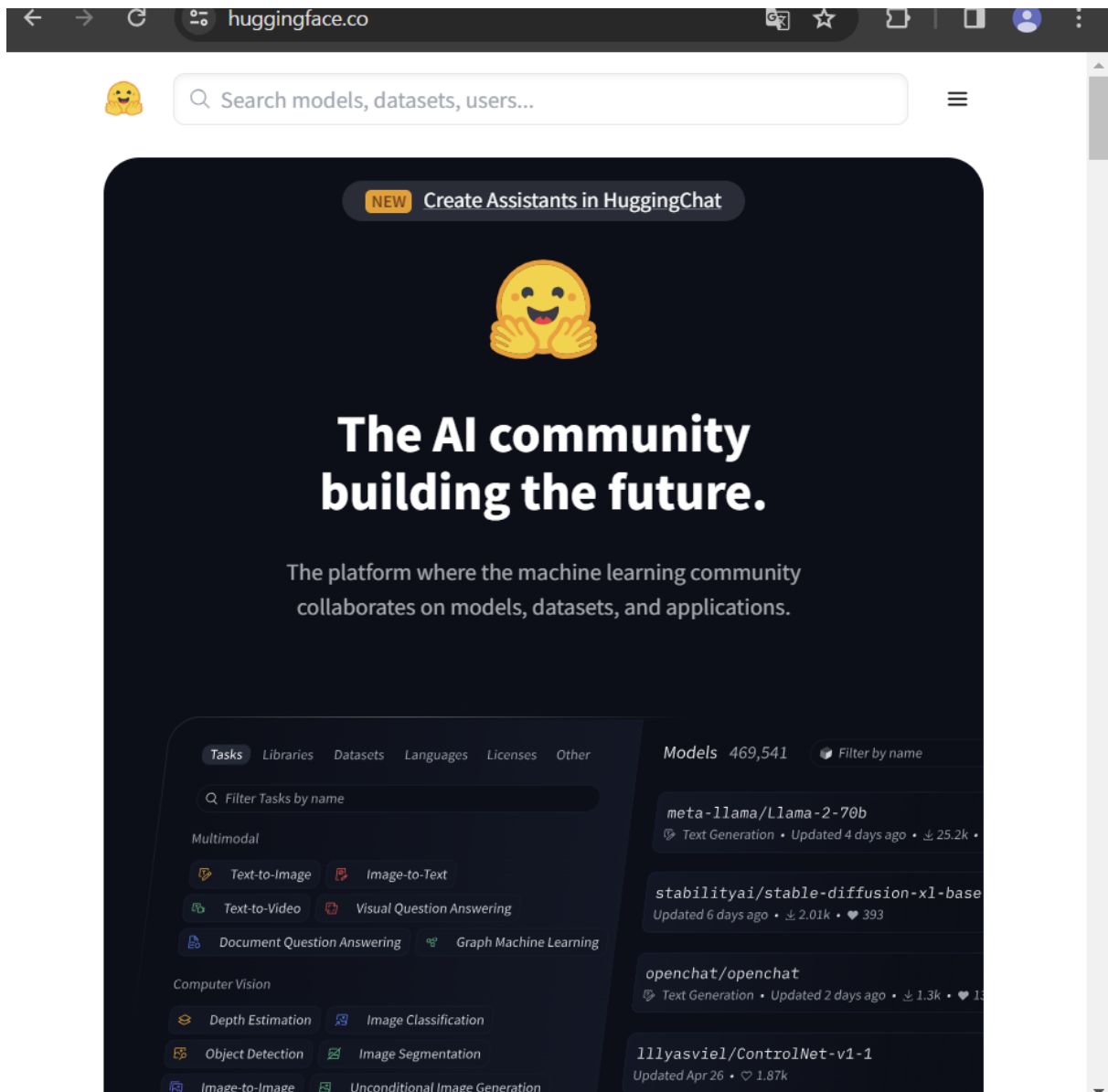
Azure 워크샵 (5일차-1)

오늘은 Langchain을 통해서 문서 분할, 임베딩, — 할 것

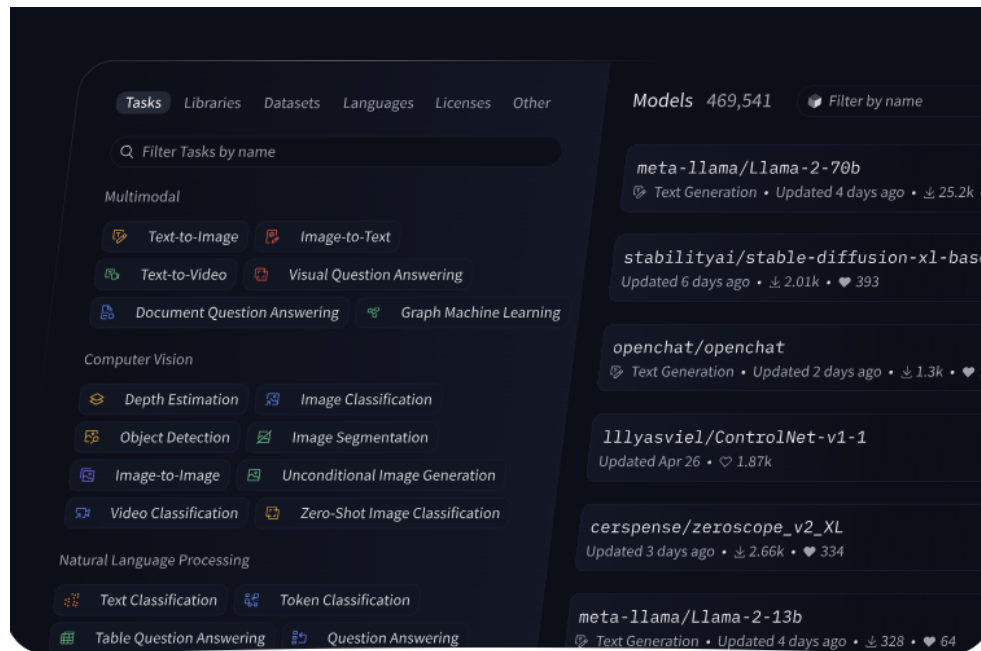




<Hugging face>



- 언어 모델계의 깃허브
새로운 모델이 나오면 여기 다 올라옴
- 언어 모델 뿐만 아니라 AI 모델도 많이 올라와 있음



- Multimodal: 두 가지 이상의 모델을 함께 사용하는 것. ex) GPT
- Meta-llama-2-70b: 70 billion의 파라미터
- 암튼 이렇게 공개된 모델을 다운받아 튜닝할 수도 있음
- 오픈소스 모델인데도 한국어 지원됨
- LangCon...

<LangChain>

<https://colab.research.google.com/drive/1Zk4c4Yomu86KgUEla1J1NOF6J0wr6jyW?usp=sharing>

pdf 파일을 준비하자

[https://onedrive.live.com/redir?
resid=E04DC3BEC3161F8B!443813&authkey=!AKnTCh0F5UEh-
og&page=View&wd=target\(Class Room.one|50f7877f-d674-46e1-a7e2-
25db0cd73579%2F00. 시작하기|1e48b674-7832-c845-8992-
5cb3b21bb7bb%2F\)&wdorigin=NavigationUrl](https://onedrive.live.com/redir?resid=E04DC3BEC3161F8B!443813&authkey=!AKnTCh0F5UEh-og&page=View&wd=target(Class%20Room.one|50f7877f-d674-46e1-a7e2-25db0cd73579%2F00.%20시작하기|1e48b674-7832-c845-8992-5cb3b21bb7bb%2F)&wdorigin=NavigationUrl)

<대한상공회의소_탄소중립_미래전략_보고서_Part1>

그리고 새로운 코랩 노트 열자

```
!pip install openai
!pip install langchain

Collecting openai
  Downloading openai-1.14.3-py3-none-any.whl (262 kB)
Requirement already satisfied: anyio<5,>=3.5.0 in /usr/local/lib/python3.10/dist-packages (from openai) (3.7.1)
Requirement already satisfied: distro<2,>=1.7.0 in /usr/lib/python3/dist-packages (from openai) (1.7.0)
Collecting httpx<1,>=0.23.0 (from openai)
  Downloading httpx-0.27.0-py3-none-any.whl (75 kB)
Requirement already satisfied: pydantic<3,>=1.9.0 in /usr/local/lib/python3.10/dist-packages (from openai) (2.6.4)
Requirement already satisfied: sniffio in /usr/local/lib/python3.10/dist-packages (from openai) (1.3.1)
Requirement already satisfied: tqdm>4 in /usr/local/lib/python3.10/dist-packages (from openai) (4.66.2)
Requirement already satisfied: typing-extensions<5,>=4.7 in /usr/local/lib/python3.10/dist-packages (from openai) (4.10.0)
Requirement already satisfied: idna>=2.8 in /usr/local/lib/python3.10/dist-packages (from anyio<5,>=3.5.0->openai) (3.6)
Requirement already satisfied: exceptiongroup in /usr/local/lib/python3.10/dist-packages (from anyio<5,>=3.5.0->openai) (1.2.0)
Requirement already satisfied: certifi in /usr/local/lib/python3.10/dist-packages (from httpx<1,>=0.23.0->openai) (2024.2.2)
Collecting httpcore==1.* (from httpx<1,>=0.23.0->openai)
  Downloading httpcore-1.0.5-py3-none-any.whl (77 kB)
Collecting h11<0.15,>=0.13 (from httpcore==1.*->httpx<1,>=0.23.0->openai)
  Downloading h11-0.14.0-py3-none-any.whl (58 kB)
Requirement already satisfied: annotated-types>=0.4.0 in /usr/local/lib/python3.10/dist-packages (from pydantic<3,>=1.9.0->openai) (0.6.0)
Requirement already satisfied: pydantic-core==2.16.3 in /usr/local/lib/python3.10/dist-packages (from pydantic<3,>=1.9.0->openai) (2.16.3)
Installing collected packages: h11, httpcore, httpx, openai
Successfully installed h11-0.14.0 httpcore-1.0.5 httpx-0.27.0 openai-1.14.3
Collecting langchain
  Downloading langchain-0.1.13-py3-none-any.whl (810 kB)
Requirement already satisfied: PyYAML>=5.3 in /usr/local/lib/python3.10/dist-packages (from langchain) (6.0.1)
Requirement already satisfied: SQLAlchemy<3,>=1.4 in /usr/local/lib/python3.10/dist-packages (from langchain) (2.0.29)
Requirement already satisfied: aiohttp<4.0.0,>=3.8.3 in /usr/local/lib/python3.10/dist-packages (from langchain) (3.9.3)
Requirement already satisfied: async-timeout<5.0.0,>=4.0.0 in /usr/local/lib/python3.10/dist-packages (from langchain) (4.0.3)
Collecting dataclasses-json<0.7,>=0.5.7 (from langchain)
  Downloading dataclasses_json-0.6.4-py3-none-any.whl (28 kB)
```

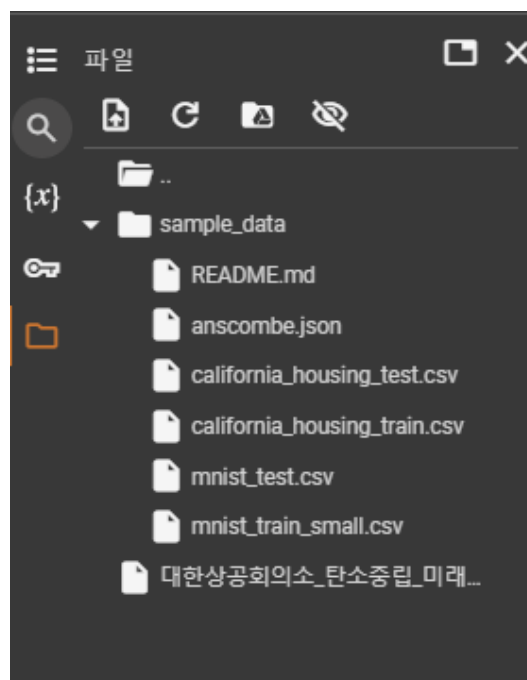
```
[3] from langchain.document_loaders import WebBaseLoader

loader = WebBaseLoader('https://www.naver.com')
data = loader.load()

print(data)

[Document(page_content=' NNAVER', metadata={'source': 'https://www.naver.com'})]
```

pdf 파일을 업로드해서 테스트해보자



```
# PDF Loader
from langchain.document_loaders import PyPDFLoader # PyPDFLoader 이용하면 PDF를 빨리 읽을 수 있음

loader = PyPDFLoader('/content/대한상공회의소_탄소중립_미래전략_보고서_Part1.pdf') # Loader를 랩핑하는 이유는 다양한 형식의 데이터를 같은 방식으로 사용하기 위해서임. 괄호 안에 pdf 경로 써주기 (경로 복사)
pages = loader.load_and_split()

ModuleNotFoundError: Traceback (most recent call last)
/usr/local/lib/python3.10/dist-packages/langchain_community/document_loaders/pdf.py in __init__(self, file_path, password, headers, extract_images)
    176     try:
--> 177         import pypdf # noqa:F401
    178     except ImportError:

ModuleNotFoundError: No module named 'pypdf'

During handling of the above exception, another exception occurred:

ImportError: Traceback (most recent call last)
/usr/local/lib/python3.10/dist-packages/langchain_community/document_loaders/pdf.py in __init__(self, file_path, password, headers, extract_images)
    177         import pypdf # noqa:F401
    178     except ImportError:
--> 179         raise ImportError(
    180             "pypdf package not found, please install it with ``pip install pypdf``"
    181         )

ImportError: pypdf package not found, please install it with 'pip install pypdf'

NOTE: If your import is failing due to a missing package, you can
manually install dependencies using either pip or apt.

To view examples of installing some common dependencies, click the
"Open Examples" button below.
```

- 오류가 나는 이유는 install pypdf가 없어서

```
!pip install openai
!pip install langchain
!pip install pypdf

Requirement already satisfied: openai in /usr/local/lib/python3.10/dist-packages (1.10.0)
Requirement already satisfied: anyio<5,>=3.5.0 in /usr/local/lib/python3.10/dist-packages (3.5.0)
Requirement already satisfied: distro<2,>=1.7.0 in /usr/local/lib/python3.10/dist-packages (1.7.0)
Requirement already satisfied: httpx<1,>=0.23.0 in /usr/local/lib/python3.10/dist-packages (0.23.0)
Requirement already satisfied: pydantic<3,>=1.9.0 in /usr/local/lib/python3.10/dist-packages (1.10.12)
Requirement already satisfied: sniffio in /usr/local/lib/python3.10/dist-packages (1.3.0)
Requirement already satisfied: tqdm>4 in /usr/local/lib/python3.10/dist-packages (4.66.1)
Requirement already satisfied: typing-extensions<4.11.0,>=4.5.0 in /usr/local/lib/python3.10/dist-packages (4.5.0)
Requirement already satisfied: idna>=2.8 in /usr/local/lib/python3.10/dist-packages (3.4)
Requirement already satisfied: exceptiongroup in /usr/local/lib/python3.10/dist-packages (1.1.2)
```

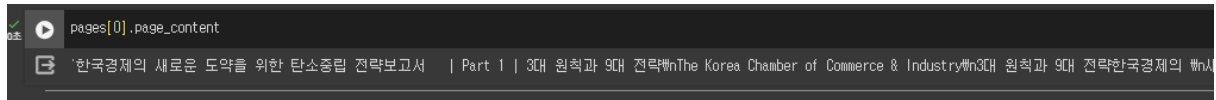
```
[11] # PDF Loader
from langchain.document_loaders import PyPDFLoader # PyPDFLoader 이용하면 PDF를 빨리 읽을 수 있음

loader = PyPDFLoader('/content/대한상공회의소_탄소중립_미래전략_보고서_Part1.pdf') # Loader를 랩핑하는
pages = loader.load_and_split() # 여기서의 split은 page를 split하는 것
# pages는 하나의 배열로 되어 있음

pages[0]

Document(page_content='한국경제의 새로운 도약을 위한 탄소중립 전략보고서 | Part 1 | 3대 원칙과 9대 전', metadata={'source': '/content/대한상공회의소_탄소중립_미래전략_보고서_Part1.pdf', 'page': 0})
```

- 이제 잘 됨 & pages[0]하면 첫 번째 페이지가 출력됨
- return되는 결과가 document. 실제 내용은 page_content에 있음
- 그 안의 내용을 가져오고 싶으면 .page_content



- 이렇게 내용이 나옴

<두 번째... >

Max Token = 글자 수와 상관 없이 토큰 단위로 한계가 설정되어 있음

페이지 단위로 자르는데 어떤 페이지는 1500자, 어떤 페이지는 2000자

근데 그게 중요한 게 아님. 토큰 수가 중요! 토큰 수가 Max Token 넘어가면 잘라줘야 함

한 페이지이지만 내부적으로는 두 개의 블록이 될 수도 있다는 것

1. 토큰 단위로 페이지 세기
2. 토큰 단위로 페이지 쪼개기

를 해보자

<쪼개자>

내용을 자를 땐 두 가지 splitter. CharacterTextSplitter와 RecursiveCharacterTextSplitter

- 첫 번째 CharacterTextSplitter: 주어진 단위대로 자르고 넘치면 버림
- 두 번째 RecursiveCharacterTextSplitter: 한 페이지가 넘치면 다시 돌아와 남은 부분을 다시 쪼갬. 그래서 필요한 단위로 쪼개질 때까지 재귀호출... 그래서 이걸 많이 사용함

```
[16] # split
from langchain_text_splitters import CharacterTextSplitter
from langchain_text_splitters import RecursiveCharacterTextSplitter
# 내용을 자를 땐 두 가지 splitter. CharacterTextSplitter와 RecursiveCharacterTextSplitter

text_splitter = CharacterTextSplitter(
    separator = '\n',
    chunk_size = 100, # 100자 단위로 잘라서 페이지를 쪼갬
    chunk_overlap = 10, # 10자씩 겹침(?)
    length_function = len
)

[17] texts = text_splitter.split_text(pages[45].page_content)
```

```
[18] print(texts[0]) # 0번 문단 print
print('-'*100)
print(texts[1]) # 1번 문단 print
print('-'*100)
print(texts[2]) # 2번 문단 print
```

46 | 47
The Korea Chamber of Commerce & Industry제고될 것으로 기대된다. 또한 재생에너지 확대에 따른 전력 계통의 안정적
운영에도 긍정적 영향을 미칠 것으로 예상된다. 우선 장단기 계약에서 실시간
시장에 이르기까지 다양한 전력시장의 구축으로 물량과 가격의 위험성을
완화할 수 있다. 이와 함께 재생에너지의 변동성을 반영하는 실시간 가격기능의
작동으로 수용성이 커질 뿐만 아니라 공급여력이 부족할 때 시장가치를

- 쪼갤 때 사용 가능한 속성들
 - splitter: 자르는 기준. 뭘 기준으로 자를 건지.
 - chunk: 자르는 단위(의 사이즈)
 - chunk_overlap: 자를 때 칼같이 자르면 연결고리가 없어지니까 살짝 overlap 시켜놓는 것

각 문단의 size를 보기 위해 list를 만들어보자

```
# split
from langchain_text_splitters import CharacterTextSplitter
from langchain_text_splitters import RecursiveCharacterTextSplitter
# 내용을 자를 땐 두 가지 splitter, CharacterTextSplitter와 RecursiveCharacterTextSplitter

text_splitter = CharacterTextSplitter(
    separator = '\n',
    chunk_size = 50, # 100자 단위로 잘라서 페이지를 쪼갬
    chunk_overlap = 5, # 10자씩 겹침(?)
    length_function = len
)
```

일단 숫자 좀 바꾸고

```
[22] texts = text_splitter.split_text(pages[45].page_content)

WARNING:langchain_text_splitters.base:Created a chunk of size 81, which is longer than the specified 50

print(texts[0]) # 0번 문단 print
print('-'*100)
print(texts[1]) # 1번 문단 print
print('-'*100)
print(texts[2]) # 2번 문단 print

46 | 47

-----
The Korea Chamber of Commerce & Industry제고될 것으로 기대된다. 또한 재생에너지 확대에 따른 전력 계통의 안정적
운영에도 긍정적 영향을 미칠 것으로 예상된다. 우선 장단기 계약에서 실시간
```

글자가 넘쳐서 잘려나감 & page 수 늘어남 앵 난 안 늘어난 거 아닌가 뭐 잘못된 듯

암튼. splitter2를 만들어보자 이번엔 recursive로!!


```
0초 # split
from langchain_text_splitters import CharacterTextSplitter
from langchain_text_splitters import RecursiveCharacterTextSplitter
# 내용을 자를 땐 두 가지 splitter, CharacterTextSplitter와 RecursiveCharacterTextSplitter

text_splitter = CharacterTextSplitter(
    separator = '\n',
    chunk_size = 50, # 100자 단위로 잘라서 페이지를 쪼갬
    chunk_overlap = 5, # 10자씩 겹침(?)
    length_function = len
)

text_splitter2 = RecursiveCharacterTextSplitter(
    separators = '\n',
    chunk_size = 50,
    chunk_overlap = 5,
    length_function = len
)
```

```
0초 texts = text_splitter.split_text(pages[45].page_content)
    texts2 = text_splitter2.split_text(pages[45].page_content)

WARNING:langchain_text_splitters.base:Created a chunk of size 81, which is longer than the specified 50
```

- recursive로 하니까 warning이 안 뜸

```
0초 char_list = []
for i in range(len(texts)):
    char_list.append(len(texts[i]))

print(char_list)

char_list = []
for i in range(len(texts2)):
    char_list.append(len(texts2[i]))

print(char_list)

[8, 80, 41, 37, 42, 37, 41, 40, 43, 39, 12, 40, 39, 42, 36, 40, 40, 39, 36, 38, 38, 38, 39, 40, 39, 49, 47]
[8, 82, 41, 37, 42, 37, 41, 40, 43, 39, 12, 40, 39, 42, 36, 40, 40, 39, 36, 38, 38, 38, 39, 40, 39, 34, 40, 20]
```

- 하지만 이건 별로임. 토큰 단위로 자르자

```
0초 !pip install openai
!pip install langchain
!pip install pypdf
!pip install tiktoken

Requirement already satisfied: http
Requirement already satisfied: pyda
Requirement already satisfied: sniff
Requirement already satisfied: tod
```

```
import tiktoken
tokenizer = tiktoken.get_encoding('cl100k_base')
```

- "tokenizer": 토큰 단위로 잘라주는 객체
- 토크나이저 자르는 방식도 여러 가지. gpt4 방식이랑 gpt3 방식이 다름
- 토큰 세는 방식은 encoder에 의해 결정됨
- gpt4는 cl100k 써야함. 이게 표준처럼 쓰이는 중.

토큰의 개수를 셀 수 있는 함수를 만들어보자

```
[36] # 토큰의 개수를 세는 함수
def tiktoken_len(text):
    tokens = tokenizer.encode(text)
    print(tokens)
    return len(tokens)

tiktoken_len('I love you')

[40, 3021, 499]
3
```

- 토큰의 개수 3개 출력
- 다른 걸 길게 넣어도 잘
- 글자 수 \geq 토큰 수

```
tiktoken_len('동해물과 백두산이')

[58189, 34983, 167, 105, 120, 54780, 22817, 109, 167, 80010, 86157, 13094]
12
```

- 한글은 무조건 토큰의 수가 많아짐 \Rightarrow 한글을 쓰면 토큰 소모가 심함
- 한글은 unicode를 쓰니까...

Tokenizer를 적용해서 split해보자

```
[39] # 토크나이저를 적용해서 스플릿

text_splitter2 = RecursiveCharacterTextSplitter(
    separators = '\n',
    chunk_size = 50,
    chunk_overlap = 5,
    length_function = tiktoken_len # 이제부터 글자 단위가 아니라 토큰 단위로 쪼갬
)
```

```
[41] texts2 = text_splitter2.split_text(pages[45].page_content)

[2790, 765, 220, 220, 2618]
[198, 791, 12126, 32479, 315, 31480, 612, 24780, 38187, 35495, 33943, 254, 72208, 43139, 55216, 67945, 53400, 13447, 13, 5251,
[198, 94772, 36092, 223, 19954, 49085, 41871, 235, 30381, 82068, 39623, 223, 169, 244, 98, 18359, 5251, 107, 116, 49072, 254,
[198, 30426, 41953, 19954, 23955, 16306, 112, 21121, 84291, 234, 22035, 50467, 28498, 239, 24486, 57519, 29854, 30426, 41953,
[198, 59399, 226, 57390, 48936, 29833, 91786, 13, 23955, 81673, 52072, 166, 119, 246, 16633, 105, 77535, 19954, 76242, 230, 22
[198, 68611, 58189, 43139, 29833, 27797, 33931, 13094, 90195, 97, 17164, 230, 5251, 123, 238, 73653, 49508, 84136, 51440, 4623
[198, 39277, 246, 36092, 223, 44005, 62060, 50643, 26799, 55421, 21028, 66610, 9019, 105, 19954, 49085, 55216, 58126, 48936, 7
[198, 13447, 24140, 33229, 13879, 227, 26799, 21028, 45618, 41953, 86351, 44966, 34804, 91586, 76242, 230, 22035, 55216, 168,
[198, 71682, 96064, 230, 84136, 25941, 41847, 101, 69697, 116, 21028, 47932, 250, 33931, 57390, 19954, 49085, 46413, 108, 6590
[198, 21121, 168, 230, 254, 82068, 27797, 43139, 3396, 225, 230, 17835, 94772, 90960, 71682, 25941, 18918, 80402, 112, 167, 24
[198, 168, 95, 234, 41381, 48936, 72208, 43139, 64432, 32428, 13447, 13, 220]
[198, 167, 246, 238, 24486, 57519, 29854, 46204, 238, 40011, 97, 30426, 41953, 21028, 74623, 39277, 102, 34804, 46204, 238, 40
[198, 66965, 29854, 30426, 41953, 21028, 10997, 248, 101, 27433, 101, 33931, 3396, 99, 251, 86351, 78102, 28867, 116, 36609, 2
[198, 28740, 225, 13094, 13447, 13, 48555, 80979, 62398, 66965, 13094, 65905, 227, 14806, 238, 16582, 35495, 65621, 46204, 238
[198, 24140, 59269, 222, 13094, 22035, 73653, 11, 46204, 238, 40011, 97, 60861, 39277, 102, 43139, 59777, 24486, 50467, 28498,
[198, 34693, 243, 67945, 48936, 29833, 36439, 18359, 72208, 13094, 13447, 13, 62398, 99105, 56154, 21028, 62398, 36609, 22035,
[198, 56154, 13879, 227, 26799, 21028, 38164, 113, 98934, 7459, 102, 33229, 13879, 227, 13094, 23955, 167, 223, 234, 32179, 96
[198, 39277, 242, 20565, 64038, 72208, 43139, 96717, 168, 116, 94, 53400, 13447, 13, 5251, 246, 238, 24486, 46204, 238, 40011,
[198, 167, 239, 242, 50467, 28498, 239, 24486, 90960, 71682, 25941, 18918, 63171, 79225, 79053, 43139, 168, 235, 101, 78696, 2
[198, 42771, 32428, 13447, 13, 33229, 13879, 227, 26799, 65950, 34804, 78696, 234, 71682, 26799, 19954, 58901, 38164, 113, 989
[198, 167, 246, 238, 16969, 36609, 28740, 102, 66406, 168, 253, 223, 29854, 57575, 66822, 108, 82001, 18918, 36609, 22035, 354
[198, 40011, 230, 22035, 40011, 231, 43139, 57519, 29854, 46204, 238, 40011, 97, 30426, 41953, 21028, 74623, 39277, 102, 19954
[198, 64189, 58126, 16969, 46204, 238, 40011, 97, 56154, 13879, 227, 26799, 21028, 75086, 27797, 14806, 230, 14705, 238, 38164
[198, 167, 48478, 52688, 19954, 10997, 248, 101, 54780, 82068, 33177, 72208, 43139, 96717, 57002, 53400, 13447, 13, 99901, 820
[198, 30426, 41953, 60861, 39277, 102, 18359, 23955, 169, 52375, 79053, 43139, 168, 235, 101, 75461, 226, 44690, 59269, 239, 2
[198, 28740, 225, 43139, 55216, 67945, 53400, 13447, 13, 57519, 29854, 46204, 238, 40011, 97, 30426, 41953, 21028, 74623, 3927
[198, 67218, 238, 40011, 97, 30426, 41953, 21028, 86503, 20565, 20565, 60798, 220]
[198, 34693, 243, 67945, 11, 78696, 234, 71682, 26799, 95415, 77535, 96064, 251, 67945, 11, 220]
[198, 66965, 29854, 30426, 41953, 21028, 10997, 248, 101, 27433, 101, 33931, 3396, 99, 251, 86351, 220]
[198, 21819, 109, 28867, 116, 36609, 22035, 3396, 116, 94, 33390, 57575, 220]
[198, 21121, 67945, 169, 248, 101, 54780, 18918, 38164, 254, 39277, 250, 48936, 29833, 220]
[198, 13467, 230, 18359, 72208, 13094, 13447, 13]
[]
[2790, 765, 220, 220, 2618]
[198, 791, 12126, 32479, 315, 31480, 612, 24780, 38187, 35495, 33943, 254, 72208, 43139, 55216, 67945, 53400, 13447, 13, 5251,
[2790, 765, 220, 220, 2618]
[198, 94772, 36092, 223, 19954, 49085, 41871, 235, 30381, 82068, 39623, 223, 169, 244, 98, 18359, 5251, 107, 116, 49072, 254,
[198, 12126, 32479, 315, 31480, 612, 24780, 38187, 35495, 33943, 254, 72208, 43139, 55216, 67945, 53400, 13447, 13, 5251,
[198, 30426, 41953, 19954, 23955, 16306, 112, 21121, 84291, 234, 22035, 50467, 28498, 239, 24486, 57519, 29854, 30426, 41953,
[198, 94772, 36092, 223, 19954, 49085, 41871, 235, 30381, 82068, 39623, 223, 169, 244, 98, 18359, 5251, 107, 116, 49072, 254,
```

```
char_list = []
for i in range(len(texts2)):
    char_list.append(len(texts2[i]))

print(char_list)

[8, 80, 41, 37, 42, 37, 41, 40, 43, 39, 12, 40, 39, 42, 36, 40, 40, 39, 36, 38, 38, 38, 39, 40, 39, 34, 40, 20]
```

<https://colab.research.google.com/drive/1Zk4c4Yomu86KgUEla1J1NOF6J0wr6jyW?usp=sharing>

<https://colab.research.google.com/drive/1Zk4c4Yomu86KgUEla1J1NOF6J0wr6jyW?usp=sharing>

<generative ai.txt 파일 다운로드 받자>

```
# Text file을 열어서 내용을 가져온다.
with open('/content/generative_ai.txt') as f:
    text_gen_ai = f.read()

[ ] !pip install langchain pypdf

Collecting langchain
  Downloading langchain-0.1.13-py3-none-any.whl (810 kB)
      810.5/810.5 kB 6.9 MB/s eta 0:00:00
Collecting pypdf
  Downloading pypdf-4.1.0-py3-none-any.whl (286 kB)
      286.1/286.1 kB 9.5 MB/s eta 0:00:00
Requirement already satisfied: PyYAML>=5.3 in /usr/local/lib/python3.10/dist-packages (from langchain) (6.0.1)
Requirement already satisfied: SQLAlchemy<3, >=1.4 in /usr/local/lib/python3.10/dist-packages (from langchain) (2.0.29)
```

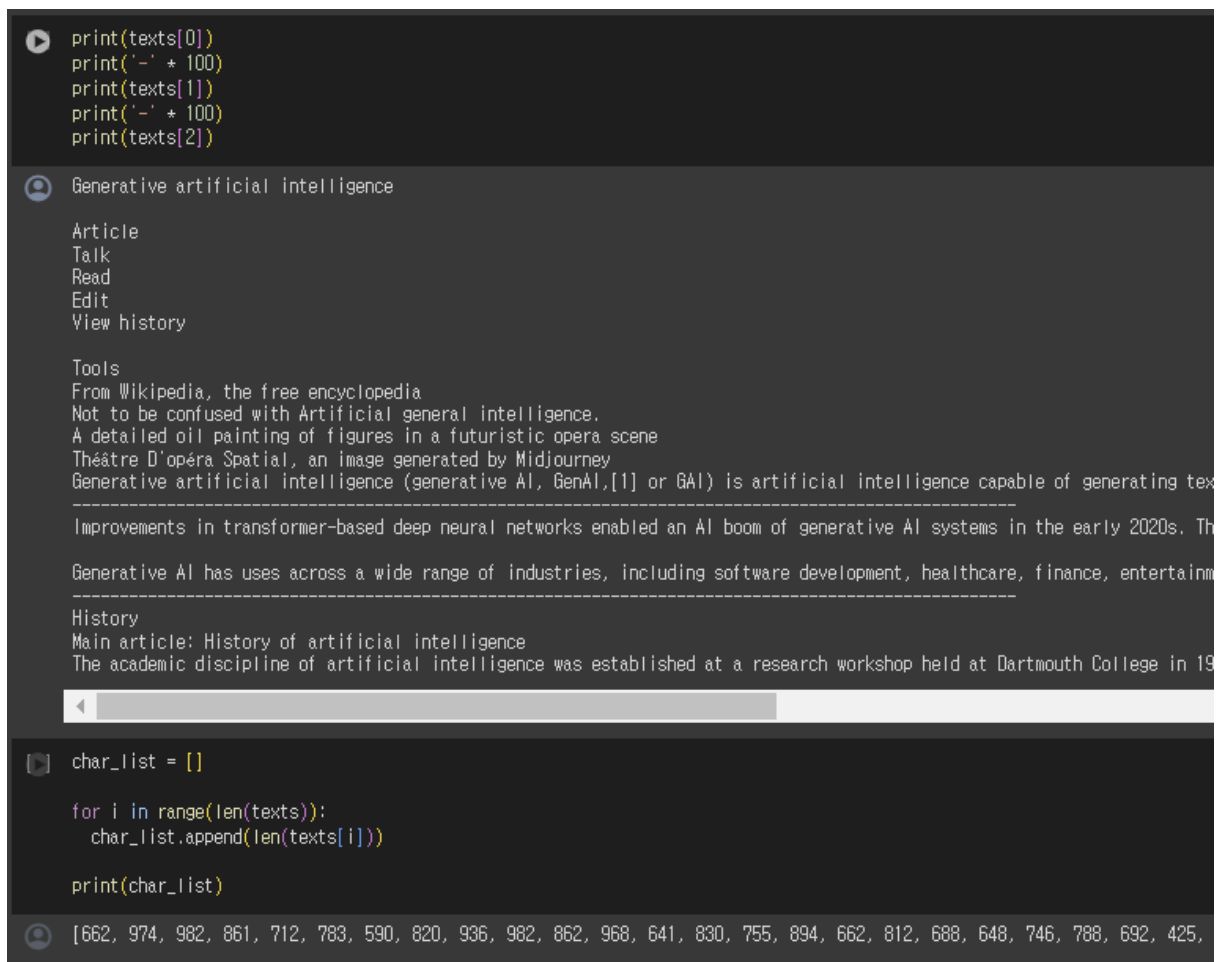
- 그냥 python 텍스트 코드 파일 & langchain과 pypdf 읽어서 가져오는 코드

```
[ ] from langchain.text_splitter import CharacterTextSplitter
    text_splitter = CharacterTextSplitter(
        separator='\\n\\n',
        chunk_size=1000,
        chunk_overlap=100,
        length_function=len
    )

[ ] texts = text_splitter.split_text(text_gen_ai)
    len(texts)

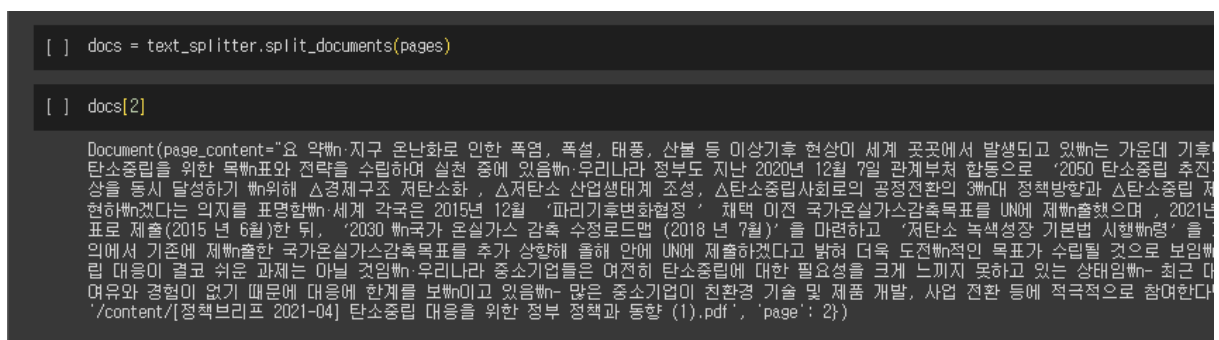
WARNING:langchain_text_splitters.base:Created a chunk of size 1130, which is longer than the specified 1000
35
```

- 문단 단위로 쪼개겠다 & \n:
- splitter를 만든 다음 text 파일 만들어서 잘라보겠다~ 했는데 하나가 오버된 걸 확인함



- // 텍스트 자르고 오버된 걸 확인

split document로 문서 객체를 만들어서 출력해보자



- 일부러 문서 객체를 만드는 이유는 문서 객체를 만들 때 메타 데이터가 들어가기 때문
- 그래서 그냥 자르지 않고 굳이 문서 객체를 만들어서 자르는 것

```
[ ] # PDF 파일을 Split한다.

from langchain.document_loaders import PyPDFLoader

loader = PyPDFLoader('/content/대한상공회의소_탄소중립_미래전략_보고서_Part1 (1).pdf')
pages = loader.load_and_split()

[ ] # 총 페이지 수
print(len(pages))
print(type(pages))
print(type(pages[0]))
print(pages[0].metadata)
print(pages[0].metadata['source'])
print(pages[0].metadata['page'])

19
<class 'list'>
<class 'langchain_core.documents.base.Document'>
{'source': '/content/[정책브리프 2021-04] 탄소중립 대응을 위한 정부 정책과 동향 (1).pdf', 'page': 0}
/content/[정책브리프 2021-04] 탄소중립 대응을 위한 정부 정책과 동향 (1).pdf
0
```

- pdf 파일을 읽어들이었는데 메타 데이터가 들어있음 (source: pdf 파일의 이름, page: 이 pdf 의 몇 번째 페이지인지)
- pdf 파일로 정보를 만들어 return했을 때 원본 데이터가 뭔지 알고 싶을 때 메타 데이터를 이 용 가능

```
[ ] docs = text_splitter.split_documents(pages)

[ ] docs[2]

Document(page_content="요약\n지구 온난화로 인한 폭염, 폭설, 태풍, 산불 등 이상기후 현상이 세계 곳곳에서 발생되고 있\n탄소중립을 위한 목표와 전략을 수립하여 실천 중에 있을\n우리나라 정부도 지난 2020년 12월 7일 관계부처 합동으로 '2050 탄소중립 추진전략\n상을 동시 달성하기\n경제구조 저탄소화 , 스마트 산업생태계 조성, 스마트중립사회로의 공정한 전환의 3\n정책방향과 스마트중립 제도\n현하\n것이다는 의지를 표명\n한 뒤, '2030\n'파리기후변화협정' 채택 이전 국가온실가스감축목표를 UN에 제출\n했으며, 2021년 10월 제출(2015년 6월)한 뒤, '2030\n온실가스 감축 수준으로만 (2018년 7월)'을 마련하고 '저탄소 녹색성장 기본법 시행령'을 개정\n의에서 기존에 제\n출한 국가온실가스감축목표를 추가 상향해 올해 안에 UN에 제출하겠다고 밝힌 더욱 도전적\n인 목표가 수립될 것으로 보\n인\n대응이 결코 쉬운 과제는 아닐 것\n우리나라 중소기업들은 여전히 탄소중립에 대한 필요성을 크게 느끼지 못하고 있는 상태\n최근 대가\n여유와 결핍이 없기 때문에 대응에 한계를 보\n있음\n많은 중소기업이 친환경 기술 및 제품 개발, 사업 전환 등에 적극적으로 참여한다면\n/content/11월12일-2021-04-14 탄소중립 대응을 위한 정보 정책과 동향 (1).pdf', 'page': 2})
```

- 위에서 가져온 pdf 파일을 다시 한번 split해서 세 번째 페이지 확인한 것

```
[ ] char_list = []
    for i in range(len(docs)):
        char_list.append(len(docs[i].page_content))

    print(char_list)

[175, 239, 1200, 1324, 1231, 1210, 1258, 1093, 1023, 840, 962, 823, 686, 855, 818, 823, 751, 366, 781]
```

- 여기서도 chunk size넘어가는 애들이 있음을 확인

recursive ~로 다시 한번 잘라보자

```
[ ] from langchain.text_splitter import RecursiveCharacterTextSplitter
text_splitter = RecursiveCharacterTextSplitter(
    chunk_size=1000,
    chunk_overlap=100,
    length_function=len
)

[ ] docs = text_splitter.create_documents([text_gen_ai])

[ ] print(len(docs))
print(len(docs[0].page_content))

27
673

[ ] print(docs[0].page_content)
print('-' * 100)
print(docs[1].page_content)
print('-' * 100)
print(docs[2].page_content)

What is generative AI?
Generative AI can learn from existing artifacts to generate new, realistic artifacts (at scale) that reflect the characteristics of the training data. Generative AI uses a number of techniques that continue to evolve. Foremost are AI foundation models, which are trained on a broad set of data.
-----
Today, generative AI most commonly creates content in response to natural language requests - it doesn't require knowledge of or entering code.
-----
What's behind the sudden hype about generative AI?
Gartner has tracked generative AI on its Hype Cycle™ for Artificial Intelligence since 2020 (also, generative AI was among our Top Strategic Technologies for 2023). ChatGPT, launched by OpenAI, became wildly popular overnight and galvanized public attention. (OpenAI's DALL·E 2 tool similarly generates images from text prompts.)
```

```
[ ] char_list = []
for i in range(len(docs)):
    char_list.append(len(docs[i].page_content))

print(char_list)

[673, 328, 659, 672, 904, 989, 925, 915, 927, 936, 370, 754, 915, 984, 858, 907, 356, 995, 374, 767, 851, 878, 791, 868, 934, 777, 669]
```

- 결과가 다르게 나오는 것을 확인. 1000자 넘는 게 없음
- ⇒ 우리가 원하는 사이즈대로 자르려면 recursive~을 이용하자!

<토큰 단위 텍스트 분할기>

✓ 토큰 단위 텍스트 분할기

```
[ ] !pip install tiktoken
```

```
Collecting tiktoken
  Downloading tiktoken-0.5.2-cp310-cp310-manylinux_2_17_x86_64.manylinux...

Requirement already satisfied: regex>=2022.1.18 in /usr/local/lib/python...
Requirement already satisfied: requests>=2.26.0 in /usr/local/lib/python...
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/li...
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10...
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/pyth...
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/pyth...
Installing collected packages: tiktoken
ERROR: pip's dependency resolver does not currently take into account all...
llm× 0.0.15a0 requires cohere, which is not installed.
llm× 0.0.15a0 requires openai, which is not installed.
Successfully installed tiktoken-0.5.2
```

```
[ ] import tiktoken
tokenizer = tiktoken.get_encoding('cl100k_base')
```

```
# 토큰 수를 세는 함수
def tiktoken_len(text):
    tokens = tokenizer.encode(text)
    return len(tokens)
```

```
[ ] # 내용의 토큰 수를 세어 본다.
print('Total chars: ' + str(len(docs[0].page_content)))
print('Total tokens: ' + str(tiktoken_len(docs[0].page_content)))
```

```
Total chars: 673
Total tokens: 137
```

- 영어가 많이 들어간 페이지: 글자 수 > 토큰 수 (?)
- 비용은 토큰 단위로 차감됨

```
[ ] # PDF의 내용을 Token 단위로 잘라 본다.
text_splitter = RecursiveCharacterTextSplitter(
    chunk_size=500,
    chunk_overlap=50,
    length_function=tiktoken_len
)

docs = text_splitter.split_documents(pages)
```

```
[ ] docs[10].page_content
```

'- 민간부문에서도 글로벌 기업-금융사의 RE100+ 참여 및 ESG+ 투자 확대, 환경-친화적 기업 투자 제한한 등 환경을 고려한 경영 활동이 확산되고 있는 추세임
는 자발적 캠페인임'++ESG : 기업의 비재무적 요소인 환경(Environment)· 사회(Social)· 지배구조 (Governance) 를 뜻하는 말로 기업 활동에 친환경 , 사회적

```
[ ] token_list = []
for i in range(len(docs)):
    token_list.append(tiktoken_len(docs[i].page_content))

print(token_list)
```

[168, 244, 486, 470, 292, 479, 489, 234, 476, 463, 265, 492, 479, 278, 472, 473, 408, 448, 437, 275, 463, 436, 140, 498, 374, 454, 475, 102, 474, 334, 4

- 11페이지 출력. 한글이니까 토큰으로 보면 좀 안 좋겠지ㅠ

- 학습시킬 땐 영어로 하자!

<이번엔 임베딩을 해보자>

<https://colab.research.google.com/drive/1SuJuwEIOOHBqVJz3UH9J7cnpNpKXInY0?usp=sharing>

필요한 라이브러리 설치하고 시작

```
!pip install openai langchain pypdf tiktoken

Requirement already satisfied: openai in /usr/local/lib/python3.10/dist-packages (1.10.0)
Requirement already satisfied: langchain in /usr/local/lib/python3.10/dist-packages (0.1.5)
Requirement already satisfied: pypdf in /usr/local/lib/python3.10/dist-packages (4.0.1)
Requirement already satisfied: tiktoken in /usr/local/lib/python3.10/dist-packages (0.5.2)
Requirement already satisfied: anyio<5,>=3.5.0 in /usr/local/lib/python3.10/dist-packages (from openai) (3.7.1)
Requirement already satisfied: distro<2,>=1.7.0 in /usr/local/lib/python3.10/dist-packages (from openai) (1.7.0)
Requirement already satisfied: httpx<1,>=0.23.0 in /usr/local/lib/python3.10/dist-packages (from openai) (0.26.0)
Requirement already satisfied: pydantic<3,>=1.9.0 in /usr/local/lib/python3.10/dist-packages (from openai) (1.10.14)
Requirement already satisfied: sniffio in /usr/local/lib/python3.10/dist-packages (from openai) (1.3.0)
Requirement already satisfied: tqdm>4 in /usr/local/lib/python3.10/dist-packages (from openai) (4.66.1)
Requirement already satisfied: typing-extensions<5,>=4.7 in /usr/local/lib/python3.10/dist-packages (from openai) (4.9.0)
Requirement already satisfied: PyYAML>=5.3 in /usr/local/lib/python3.10/dist-packages (from langchain) (6.0.1)
Requirement already satisfied: SQLAlchemy<3,>=1.4 in /usr/local/lib/python3.10/dist-packages (from langchain) (2.0.24)
Requirement already satisfied: aiohttp<4.0.0,>=3.8.3 in /usr/local/lib/python3.10/dist-packages (from langchain) (3.9.3)
Requirement already satisfied: async-timeout<5.0.0,>=4.0.0 in /usr/local/lib/python3.10/dist-packages (from langchain) (4.0.3)
Requirement already satisfied: dataclasses-json<0.7,>=0.5.7 in /usr/local/lib/python3.10/dist-packages (from langchain) (0.6.4)
Requirement already satisfied: jsonpatch<2.0,>=1.33 in /usr/local/lib/python3.10/dist-packages (from langchain) (1.33)
Requirement already satisfied: langchain-community<0.1,>=0.0.17 in /usr/local/lib/python3.10/dist-packages (from langchain) (0.0.34)
Requirement already satisfied: langchain-core<0.2,>=0.1.16 in /usr/local/lib/python3.10/dist-packages (from langchain) (0.1.32)
Requirement already satisfied: langsmith<0.1,>=0.0.83 in /usr/local/lib/python3.10/dist-packages (from langchain) (0.0.85)
Requirement already satisfied: numpy<2,>=1 in /usr/local/lib/python3.10/dist-packages (from langchain) (1.23.5)
Requirement already satisfied: requests<3,>=2 in /usr/local/lib/python3.10/dist-packages (from langchain) (2.31.0)
Requirement already satisfied: tenacity<9.0.0,>=8.1.0 in /usr/local/lib/python3.10/dist-packages (from langchain) (8.2.3)
Requirement already satisfied: regex<=2022.1.18 in /usr/local/lib/python3.10/dist-packages (from tiktoken) (2023.6.3)
Requirement already satisfied: aiosignal<=1.1.2 in /usr/local/lib/python3.10/dist-packages (from aiohttp<4.0.0,>=3.8.3->langchain) (1.1.2)
Requirement already satisfied: attrs<=17.3.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp<4.0.0,>=3.8.3->langchain) (23.1.0)
Requirement already satisfied: frozenlist<=1.1.1 in /usr/local/lib/python3.10/dist-packages (from aiohttp<4.0.0,>=3.8.3->langchain) (1.1.1)
Requirement already satisfied: multidict<7.0,>=4.5 in /usr/local/lib/python3.10/dist-packages (from aiohttp<4.0.0,>=3.8.3->langchain) (6.0.4)
Requirement already satisfied: yarl<2.0,>=1.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp<4.0.0,>=3.8.3->langchain) (1.9.4)
Requirement already satisfied: idna<=2.8 in /usr/local/lib/python3.10/dist-packages (from anyio<5,>=3.5.0->openai) (3.6)
Requirement already satisfied: exceptiongroup in /usr/local/lib/python3.10/dist-packages (from anyio<5,>=3.5.0->openai) (1.2.0)
Requirement already satisfied: marshmallow<4.0.0,>=3.18.0 in /usr/local/lib/python3.10/dist-packages (from dataclasses-json->langchain) (3.21.0)
Requirement already satisfied: typing-inspect<1,>=0.4.0 in /usr/local/lib/python3.10/dist-packages (from dataclasses-json->langchain) (0.9.0)
Requirement already satisfied: certifi in /usr/local/lib/python3.10/dist-packages (from httpx<1,>=0.23.0->openai) (2023.7.22)
```

```
[ ] import os

os.environ['OPENAI_API_KEY'] = 'bb6860ae08e147eeae418efdc560e4e'
os.environ['AZURE_OPENAI_ENDPOINT'] = 'https://helloai-openai-025.openai.azure.com/'
os.environ['OPENAI_API_TYPE'] = 'azure'
os.environ['OPENAI_API_VERSION'] = '2023-05-15'
```

임베딩 = 데이터를 데이터 체계로 바꾼다

일반적으로 수치로 바꾼다.. 언어 모델에서의 임베딩... 단어들 사이의 좌표 관계

사람과 바나나가 가까운지 원숭이와 바나나가 가까운지

언어마다 거리감이 달라서 임베딩하는 모델이 다 다른 한국어, 영어 등등 다 다른

작은 걸 쓸 땐 임베딩이 안 맞음

openAI처럼 돈 많은 곳 전용

암튼 임베딩-ada-002 배포

일반적인 language 모델에 비해 임베딩 모델은 상대적으로 난이도가 많이 떨어짐
일단 이걸로 하겠지만 오픈소스 기반의 LLM 사용하는 것도 이따가 볼 것

```
from langchain.embeddings import AzureOpenAIEmbeddings

embeddings_model = AzureOpenAIEmbeddings(
    model = 'dev-text-embedding-ada-002'
)

/usr/local/lib/python3.10/dist-packages/langchain_core/_api/deprecation.py:117: LangchainDeprecationWarning: The `warn_deprecated` method is deprecated. Use the `warn` method instead.
```

- Azure에서 내부적으로 임베딩 모델 랩핑해놓은 애를 가져옴

이제 단어 말고 문장 위주로 보자

```
[ ] embedded_query_q = embeddings_model.embed_query('이 대화에서 언급된 이름은 무엇입니까?')
    embedded_query_a = embeddings_model.embed_query('이 대화에서 언급된 이름은 홍길동 입니다.')

WARNING:langchain_community.embeddings.openai:Warning: model not found. Using cl100k_base encoding.
WARNING:langchain_community.embeddings.openai:Warning: model not found. Using cl100k_base encoding.

[ ] len(embedded_query_q)

1536
```

- 다섯 개가 임베딩돼서 들어감
- 0번에 1536차원, 1번에 1536차원, ... : 각각의 문장을 임베딩할 때 각각의 데이터가 1536차원으로 임베딩된다는 것. 임베딩 모델이 클수록 차원 수가 많아지고, 차원 수가 많아지면 더 정확하게 분류 가능.
- 대신 차원 수를 늘릴 땐 데이터 수도 함께 늘려줘야 한다는 것. GPU 비용도 고려해야 함.

```
embedding_result = embeddings_model.embed_documents(
    [
        '안녕하세요',
        '제 이름은 홍길동입니다.',
        '이름은 무엇인가?',
        '행체인은 유용합니다.',
        'Hello World'
    ]
)

WARNING:langchain_community.embeddings.openai:Warning: model not found. Using cl100k_base encoding.

[ ] # 결과는 1536 차원으로 벡터화 된다.
    len(embedding_result[0])

1536
```

- cl100k_base: GPT3.5 이상에서 쓰고 있는 encoding 방법
- 암튼 이렇게 encoding해도 각각의 문장은 1536차원으로 인코딩되어 있다는 것

```
#벡터의 거리를 측정하는 cos_sim()를 만든다.
from numpy import dot
from numpy.linalg import norm
import numpy as np

def cos_sim(A,B):
    distance = dot(A,B) / (norm(A)+norm(B))
    return distance

print(cos_sim(embedded_query_q, embedded_query_a))
print(cos_sim(embedded_query_q, embedding_result[1]))
print(cos_sim(embedded_query_q, embedding_result[3]))

0.9070222021812047
0.7762442819388633
0.7316183027788279
```

- 2차원 좌표 거리 측정... 두 단어의 유사도를 측정하기 위해 거리를 측정하는 함수를 만들기
- dot 연산을 하자! cos simulator 기법으로. 가장 기본적인 방법.
- 각 좌표 사이의 거리로 유사도를 알 수 있음
- 1에서 멀어지면 멀어질수록 유사도가 낮아짐
- // 이걸 왜 이렇게 할까?
 - 페이지 수 엄청 많은 파일에서 뭔가를 찾으려고 할 때
 - 차원으로 바꾼 다음 벡터 기반으로 검색하면
 - 성능에 비해 비용이 거의 안 듦
 - 그래서 이걸 벡터 DB화 시키는 것

<HuggingFace> LLM계의 오픈소스

- huggingface에 있는 모델을 여기서 바로 다운 받고 사용 가능
- 모델명만 알면 됨

▼ Huggingface Embedding

```
!pip install sentence_transformers
```

```
Requirement already satisfied: sentence_transformers in /usr/local/lib/python3.10/dist-packages (2.3.1)
Requirement already satisfied: transformers<5.0.0,>=4.32.0 in /usr/local/lib/python3.10/dist-packages (from sentence_transformers) (4.66.2)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from sentence_transformers) (4.66.2)
Requirement already satisfied: torch>=1.11.0 in /usr/local/lib/python3.10/dist-packages (from sentence_transformers) (2.0.1)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (from sentence_transformers) (1.23.5)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.10/dist-packages (from sentence_transformers) (1.2.2)
Requirement already satisfied: scipy in /usr/local/lib/python3.10/dist-packages (from sentence_transformers) (1.11.0)
Requirement already satisfied: nltk in /usr/local/lib/python3.10/dist-packages (from sentence_transformers) (3.8.1)
Requirement already satisfied: sentencepiece in /usr/local/lib/python3.10/dist-packages (from sentence_transformers) (0.1.99)
Requirement already satisfied: huggingface-hub>=0.15.1 in /usr/local/lib/python3.10/dist-packages (from sentence_transformers) (0.16.4)
Requirement already satisfied: Pillow in /usr/local/lib/python3.10/dist-packages (from sentence_transformers) (9.4.0)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from huggingface-hub>=0.15.1->sentence_transformers) (3.12.2)
Requirement already satisfied: fsspec>=2023.5.0 in /usr/local/lib/python3.10/dist-packages (from huggingface-hub>=0.15.1->sentence_transformers) (2023.6.0)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from huggingface-hub>=0.15.1->sentence_transformers) (2.31.0)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.10/dist-packages (from huggingface-hub>=0.15.1->sentence_transformers) (6.0.1)
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.10/dist-packages (from huggingface-hub>=0.15.1->sentence_transformers) (4.5.0)
Requirement already satisfied: packaging>=20.9 in /usr/local/lib/python3.10/dist-packages (from huggingface-hub>=0.15.1->sentence_transformers) (23.1)
Requirement already satisfied: sympy in /usr/local/lib/python3.10/dist-packages (from torch>=1.11.0->sentence_transformers) (1.11.0)
Requirement already satisfied: networkx in /usr/local/lib/python3.10/dist-packages (from torch>=1.11.0->sentence_transformers) (3.1)
Requirement already satisfied: Jinja2 in /usr/local/lib/python3.10/dist-packages (from torch>=1.11.0->sentence_transformers) (3.1.2)
Requirement already satisfied: triton==2.1.0 in /usr/local/lib/python3.10/dist-packages (from torch>=1.11.0->sentence_transformers) (2.1.0)
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.10/dist-packages (from transformers<5.0.0,>=4.32.0->sentence_transformers) (2023.6.3)
Requirement already satisfied: tokenizers<0.19,>=0.14 in /usr/local/lib/python3.10/dist-packages (from transformers<5.0.0,>=4.32.0->sentence_transformers) (0.15.1)
Requirement already satisfied: safetensors>=0.3.1 in /usr/local/lib/python3.10/dist-packages (from transformers<5.0.0,>=4.32.0->sentence_transformers) (0.4.0)
Requirement already satisfied: click in /usr/local/lib/python3.10/dist-packages (from nltk->sentence_transformers) (8.1.6)
Requirement already satisfied: joblib in /usr/local/lib/python3.10/dist-packages (from nltk->sentence_transformers) (1.3.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn->sentence_transformers) (3.1.0)
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-packages (from Jinja2->torch>=1.11.0->sentence_transformers) (2.1.3)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests->huggingface-hub>=0.15.1->sentence_transformers) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests->huggingface-hub>=0.15.1->sentence_transformers) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests->huggingface-hub>=0.15.1->sentence_transformers) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests->huggingface-hub>=0.15.1->sentence_transformers) (2023.7.22)
```

- 이것도 써봤다고 하는 게 좋은~
- sentence_transformers: huggingface에 있는 모델을 사용할 때 쓰는 패키지명
- 참고로... language 모델은 운영할 때도 gpu가 많이 듬
- huggingface에 있는 모델들은 잘만 튜닝하면 cpu만으로도 실행 가능하다는 장점...

```
from langchain.embeddings import HuggingFaceBgeEmbeddings
```

```
model_name = 'BAAI/bge-small-en'
model_kwargs = {'device':'cpu'}
encode_kwargs = {'normalize_embeddings':True}
```

```
hf = HuggingFaceBgeEmbeddings(
    model_name = model_name,
    model_kwargs = model_kwargs,
    encode_kwargs = encode_kwargs
)
```

```
/usr/local/lib/python3.10/dist-packages/huggingface_hub/utils/_token.py:88: 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/settings/tokens)
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(
```

- 모델명, 활동명, 영어만 가능한 모델..
- 모델에 대한 설정.. GPU/CPU에서 돌아가게끔
- HuggingFaceEmbeddings: huggingFace에서 실제로 모델을 가져오는 명령

```

from langchain.embeddings import HuggingFaceBgeEmbeddings

model_name = 'BAAI/bge-small-en'
model_kwargs = {'device':'cpu'}
encode_kwargs = {'normalize_embeddings':True}

hf = HuggingFaceBgeEmbeddings(
    model_name = model_name,
    model_kwargs = model_kwargs,
    encode_kwargs = encode_kwargs
)

```

/usr/local/lib/python3.10/dist-packages/huggingface_hub/utils/_token.py:88: 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/settings/tokens>), set it
 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%	349/349	[00:00<00:00, 13.8kB/s]
config_sentence_transformers.json: 100%	124/124	[00:00<00:00, 5.37kB/s]
README.md: 100%	90.8k/90.8k	[00:00<00:00, 1.73MB/s]
sentence_bert_config.json: 100%	52.0/52.0	[00:00<00:00, 3.11kB/s]
config.json: 100%	684/684	[00:00<00:00, 38.2kB/s]
model.safetensors: 100%	133M/133M	[00:01<00:00, 96.6MB/s]
tokenizer_config.json: 100%	366/366	[00:00<00:00, 14.5kB/s]
vocab.txt: 100%	232k/232k	[00:00<00:00, 3.55MB/s]
tokenizer.json: 100%	711k/711k	[00:00<00:00, 10.6MB/s]
special_tokens_map.json: 100%	125/125	[00:00<00:00, 4.55kB/s]
1_Pooling/config.json: 100%	190/190	[00:00<00:00, 8.43kB/s]

- 이런 식으로 출력됨

```

embedding_result = hf.embed_documents(
    [
        'today is monday',
        'weather is nice today',
        'what is the problem',
        'langchain is useful',
        'Hello World',
        'my name is morris'
    ]
)

len(embedding_result[1])

```

384

- 흠... 뭐라 하셨는지 까먹음

```
[ ] BGE_query_q = hf.embed_query('Hello? who is this?')
    BGE_query_a = hf.embed_query('hi this is harrison?')

[ ] print(cos_sim(BGE_query_q, BGE_query_a))
    print(cos_sim(BGE_query_q, embedding_result[1]))
    print(cos_sim(BGE_query_q, embedding_result[5]))

0.8585610628040985
0.7469068511786857
0.792870425931663
```


- 아까랑 똑같이 두 개를 만들어 비교해보기
- 유사도 85%... 1번 5번 둘 다 판소리니까 0.78 정도에서 그침
- 보통 80% 넘어가면 어느 정도 맞다고 봄


한글 무료로 쓰려면...


<ko-sbert-nli> 한글로 튜닝해서 만들어 놓은 것


▼ ko-sbert-nli


```
model_name = 'jhgan/ko-sbert-nli'
model_kwargs = {'device': 'cpu'}
encode_kwargs = {'normalize_embeddings': True}
hf = HuggingFaceBgeEmbeddings(
    model_name = model_name,
    model_kwargs = model_kwargs,
    encode_kwargs = encode_kwargs
)
```


modules.json: 100%  229/229 [00:00<00:00, 3.93kB/s]


config_sentence_transformers.json: 100%  123/123 [00:00<00:00, 3.45kB/s]


README.md: 100%  4.46k/4.46k [00:00<00:00, 288kB/s]


sentence_bert_config.json: 100%  53.0/53.0 [00:00<00:00, 3.02kB/s]


config.json: 100%  620/620 [00:00<00:00, 34.8kB/s]


pytorch_model.bin: 100%  443M/443M [00:05<00:00, 121MB/s]

tokenizer_config.json: 100%  538/538 [00:00<00:00, 18.0kB/s]

vocab.txt: 100%  248k/248k [00:00<00:00, 4.08MB/s]

tokenizer.json: 100%  495k/495k [00:00<00:00, 8.86MB/s]

special_tokens_map.json: 100%  112/112 [00:00<00:00, 7.32kB/s]

1_Pooling/config.json: 100%  190/190 [00:00<00:00, 8.15kB/s]

```

sentence = [
    '안녕하세요',
    '제 이름은 홍길동 입니다.',
    '이름이 무엇인가요?',
    '행체인은 유용합니다.',
    '홍길동 아버지의 이름은 홍상직 입니다.'
]

embedding_result = hf.embed_documents(sentence)

[ ] q = '홍길동은 아버지를 아버지라 부르지 못했었습니다. 홍길동 아버지의 이름은 무엇입니까?'
    a = '홍길동의 아버지는 엄했습니다.'

    ko_query_q = hf.embed_query(q)
    ko_query_a = hf.embed_query(a)

[ ] print(cos_sim(ko_query_q, ko_query_a))
    print(cos_sim(ko_query_q, embedding_result[1]))
    print(cos_sim(ko_query_q, embedding_result[3]))
    print(cos_sim(ko_query_q, embedding_result[4]))

0.8412900341366036
0.5585852341123605
0.24146539720456686
0.626779213871763

```

- 각각 비교했더니 유사도가 위와 같이 나옴
- langchain 부분은 유사도가 0.24로 확실히 낮음
- 암튼 이렇게 무료로 쓸 수 있는 임베딩 모델이 있다는 것...
- 임베딩할 땐 무료 모델을 쓰고 질문할 땐 chat gpt를 쓰면 비용 줄일 수 있음

```

len(embedding_result[0])

768

```

- 768차원...
- 나중에 이걸 써보기 대신 ada를 사용하면 좀 더 결과 잘 나오긴 함