











(주)선도소프트 윤훈주, 오영권 멘토와 함께하는

빅데이터 기반 한글 언어처리

검색 솔루션 구현

By 권상지, 김기범, 김연준, 손수한, 현재은

중앙정보처리학원

제작 환경		version
프로그래밍 언어	 python™	3.7.4
개발 툴	 CODE Visual Studio Code	1.81.0
	 colab	pro
	 ANACONDA	4.13.0
분석툴	 PyTorch	1.6.0
웹개발	 django	3.2.20
자연어처리	 fastText	0.9.2
	 gensim	3.8.3
	 KoNLPy	0.5.1
가상환경	 docker	6.1.3
DB	 SQLite	v0.14.1

목 차

- 프로젝트 소개
- 개발 환경 with 'Docker'
- 데이터 수집
- DB구축 with 'SQLite'
- TASK별 모델링
- 웹 구현 과정 with 'Django'
- AI Search 웹 시현
- 결론

프로젝트 소개

검색어 솔루션 기능 제공

‘선도 소프트’의 멘토링
한국어 텍스트 데이터를 활용한
검색 솔루션 제공

주제 선정 배경

‘정보의 홍수’시대
기존 검색포털사이트와 차별화된 서비스 제공
검색어에 다양한 기능 추가

[Google Search](#)[I'm Feeling Lucky](#)

< 기존 검색포털 >

AI Search

< 검색어 솔루션 >

개발 환경 with 'Docker'



docker?

앱 구축, 테스트 및 배포 가능 플랫폼.

컨테이너 환경(프로그램 버전, 라이브러리, 코드 및 기타 세팅 포함)에서

앱 실행이 가능하여

모든 사용자가 동일한 환경에서 작업 수행 가능

개발 환경 with 'Docker'

docker image 생성 및 배포

동일 작업 환경 구현 목적

docker 컨테이너 생성

Task에 필요한 라이브러리 등 설치


컨테이너를 docker 이미지로 생성

docker Hub에 배포(업로드)

docker image 다운로드 및 실행

- docker hub 또는 prompt에서 docker image 다운로드
- 다운받은 image파일을 컨테이너로 불러오기
- 컨테이너로 docker image파일 실행

Explore [bianne1004/my-ubuntu](#)





 **bianne1004/my-ubuntu** ☆ ↓ Pulls 14

By [bianne1004](#) • Updated 2 hours ago

Image

Overview **Tags**

Sort by Newest Filter Tags 🔍

TAG v1.4 Last pushed 2 hours ago by bianne1004	DIGEST e6baf1fa34e6	OS/ARCH linux/amd64	COMPRESSED SIZE ⓘ 9.45 GB	<code>docker pull bianne1004/my-ubuntu</code> 
TAG v1.3 Last pushed a day ago by bianne1004	DIGEST 81d294cc6cb4	OS/ARCH linux/amd64	COMPRESSED SIZE ⓘ 8.72 GB	<code>docker pull bianne1004/my-ubuntu</code> 
TAG v1.2 Last pushed a day ago by bianne1004	DIGEST d44f66ce8153	OS/ARCH linux/amd64	COMPRESSED SIZE ⓘ 7.99 GB	<code>docker pull bianne1004/my-ubuntu</code> 
TAG v1.0 Last pushed 9 days ago by bianne1004				<code>docker pull bianne1004/my-ubuntu</code> 

데이터 수집

NAVER API 크롤링

네이버 API 이용 웹크롤링 진행

뉴스기사의 제목, 본문, 링크, 발행일 수집

수집된 데이터 (.json) DataFrame 변환 후

CSV 파일 저장

```
# 검색어에 대한 뉴스 1000개 json으로 저장
# [CODE 1]
def getRequestUrl(url):
    req = urllib.request.Request(url)
    req.add_header("X-Naver-Client-Id", Client_id)
    req.add_header("X-Naver-Client-Secret", Client_pw)

    try:
        response = urllib.request.urlopen(req)
        if response.getcode() == 200:
            print("[%s] Url Request Success" % datetime.datetime.now())
            return response.read().decode('utf-8')
    except Exception as e:
        print(e)
        print("[%s] Error for URL : %s" % (datetime.datetime.now(), url))
        return None

# [CODE 2]
# api를 이용한
def getNaverSearch(node, srcText, start, display):
    base = "https://openapi.naver.com/v1/search"
    node = "/%s.json" % node
    parameters = "?query=%s&start=%s&display=%s" % (urllib.parse.quote(srcText), start, display)

    url = base + node + parameters
    responseDecode = getRequestUrl(url) # [CODE 1]

    if (responseDecode == None):
        return None
    else:
        return json.loads(responseDecode)
```

< 필요항목 네이버에서 json파일로 받아오기 >

	title_f	pDate	content	categories	likes
0	제2회 삼수 매봉산 바람개비축제 개최	2023-07-13 14:35:00	【태백】삼수 매봉산 바람개비축제가 오는 8월 4일부터 6일까지 바람의언덕 밀자락에서...	사회	0
1	속조시, 강원세계산림엑스포 앞두고 '안심먹거리' 만전	2023-07-13 14:35:00	핵심요약\n개최지 주변 대형-유명음식점 900여 곳 점검\n강원 속조시청. 전영래 ...	사회	0
2	관광공사, 아동권리보장원과 자립준비청년 취업지원 업무협약	2023-07-13 14:34:00	아코르 엠버서더 코리아 빈센트 릴레이 부사장(왼쪽부터), 한국관광공사 이재환 부사장...	사회	0
3	진도군 4개 해수욕장, 오는 14일(금) 일제히 개장	2023-07-13 14:34:00	가계, 금값, 신전, 관매도 해수욕장, 8월 15일까지 운영\n진도군이 14일 관내...	사회	0
4	"만화책에 들어간 느낌"...4m 높이 책처럼 꾸민 웰튼특별관(종합)	2023-07-13 14:34:00	캐릭터 라이선싱 페어 개막...애니.완구.이모티콘 캐릭터 한자리에\n\n\n(서울=연합...	생활	1
...
1195	[프레스룸LIVE] "제일 좋은 건 사표내고 나가는 것" MZ세대와 소통한 홍준표	2023-07-11 16:56:00	<진행>\n김태일 앵커\n<출연>\n김용태 / 전 국민의힘 최고위원\n이승훈 /...	정치	2
1196	20년도 더 된 추억의 IP '콩야'...어떻게 MZ세대 사로잡았나	2023-07-11 16:50:00	직장인 된 MZ세대에게 마음 얻은 '양파콩야'\n넷마블 '추억의 IP에 캐릭터성 부...	IT	2
1197	기발한 아이디어...차세대 팝아트 주자 모인다	2023-07-11 16:43:00	'어반브레이크' 13일 개막\nSA+, 김준식-김대운 등 전시\n\n\n김준식 'Mic...	생활	0
1198	GS, 인터리커 와인 판매	2023-07-11 16:43:00	드링크 인터내셔널의 자회사인 인터리커는 GS25와 함께 전 세계 판매 1위 보르도 ...	경제	1
1199	패션쇼에 콘서트까지...HD현대 사옥, 매달 '변신'	2023-07-11 16:39:00	정기선 체제 새 기업문화 확산\n'일하고 싶은 회사'로 인재 유치\n\n올해 5월 경기...	경제	0

1200 rows × 5 columns

< 생성된 DataFrame >

Data 전처리

Data 전처리

한글 텍스트 데이터 활용

파이썬 내장함수 ▶ 불용어처리

TEANAPS 전처리 활용 ▶ 반복어 처리

DataFrame 칼럼 정의

```
# 뽑은 text 전처리
def processing_text(text):
    content_text = BeautifulSoup(text, "html5lib").get_text() # HTML 태그 제거
    content_text = re.sub("\n", " ", content_text)
    return content_text
```

```
text1 = processing_text(similar_sentences[0][0])
text2 = processing_text(similar_sentences[1][0])
```

```
# 데이터 전처리
df['Content'] = df['Content'].apply(lambda x: x.strip())
df['Title'] = df['Title'].apply(lambda x: x.strip())
df['Content'] = df['Content'].apply(lambda x: x.lstrip())
df['Title'] = df['Title'].apply(lambda x: x.lstrip())
df['Content'] = df['Content'].apply(lambda x: x.rstrip())
df['Title'] = df['Title'].apply(lambda x: x.rstrip())

# 반복 문자열 제거
df['Title'] = df['Title'].astype(str).apply(pro.iteration_remover, replace_char=".")
df['Content'] = df['Content'].astype(str).apply(pro.iteration_remover, replace_char=".")
```

< 불용어 / 반복어 처리 >

데이터 수집 - 이슈사항

웹 크롤링 이슈

4개 플랫폼(네이버, 다음, 줌, 네이버 블로그) 크롤링 진행.
네이버 블로그 - 많은 광고로 인해 정보 전달 부적격 판단
포털 3사 뉴스 크롤링 진행했으나 중복 기사 다량 발견
네이버 뉴스만 사용

API 이슈

이코리아 | 2023.07.03.

트위터·레딧 '데이터 유료화'에 대체 SNS 찾아 나선 이용자들

지난 2월에는 개발자들이 이용하는 트위터 API를 유료화했다. 또 대표적인 AI 개발사인 마이크로소프트에 대해 "마이크로소프트가 트위터의 데이터를 불법적으로..."

ZDNet Korea PICK | 2023.07.02. | 네이버뉴스

월 5천만원 트위터 API, 최악 품질로 불만 고조

트위터 API 유료 모델이 가짜 크롤링 서비스 부마음 제기하 차고도 제때는 자도

정책 이슈

Google

크롤링 정책

뉴데일리 경제

"초거대 AI, 데이터 60~70% '무단' 학습"... '사각지대'

초거대 인공지능(AI) 경쟁이 격화되면서 '데이터 소유권'이 핵심 법적 쟁점으로 떠오르고 있다. 온라인에 공개된 무료 데이터로 초거대 AI를 학습시킬...

1개월 전

DB구축 with 'SQLite'

전처리 된 Data 빠른 웹 서비스를 위해 DB에 구현
django에서 사용 가능하도록 SQLite3를 이용하여 DB구축

SQLite3 작업

	Title	Date	Content
	Search column...	Search column...	Search column...
1	매매도 전세도 수도...	2023-07-13 14:02:00	전국 아파트값 3주 연...
2	'호반써밋파크에디션'...	2023-07-13 14:01:00	인천 연희공원 내 지...
3	전국 아파트값 3주 연...	2023-07-13 14:00:00	서울은 강남, 강북 모...
4	'호우주의보' 인천 도...	2023-07-13 13:55:00	시, 지대분 1단계 운...
5	송도 72.5mm...'호우주...	2023-07-13 13:46:00	13일 낮 12시10분께 ...
6	인천 '호반써밋 파크...	2023-07-13 13:21:00	인천 서구 연희동에...
7	호바거석 인천 '호바	2023-07-13 13:19:00	호바거석은 오는 18

DB에서 데이터프레임으로 불러오기

```
# SQLite 데이터베이스에 연결
conn = sqlite3.connect('인천서구.db')

# 테이블 조회 쿼리
query = "SELECT * FROM Table_"

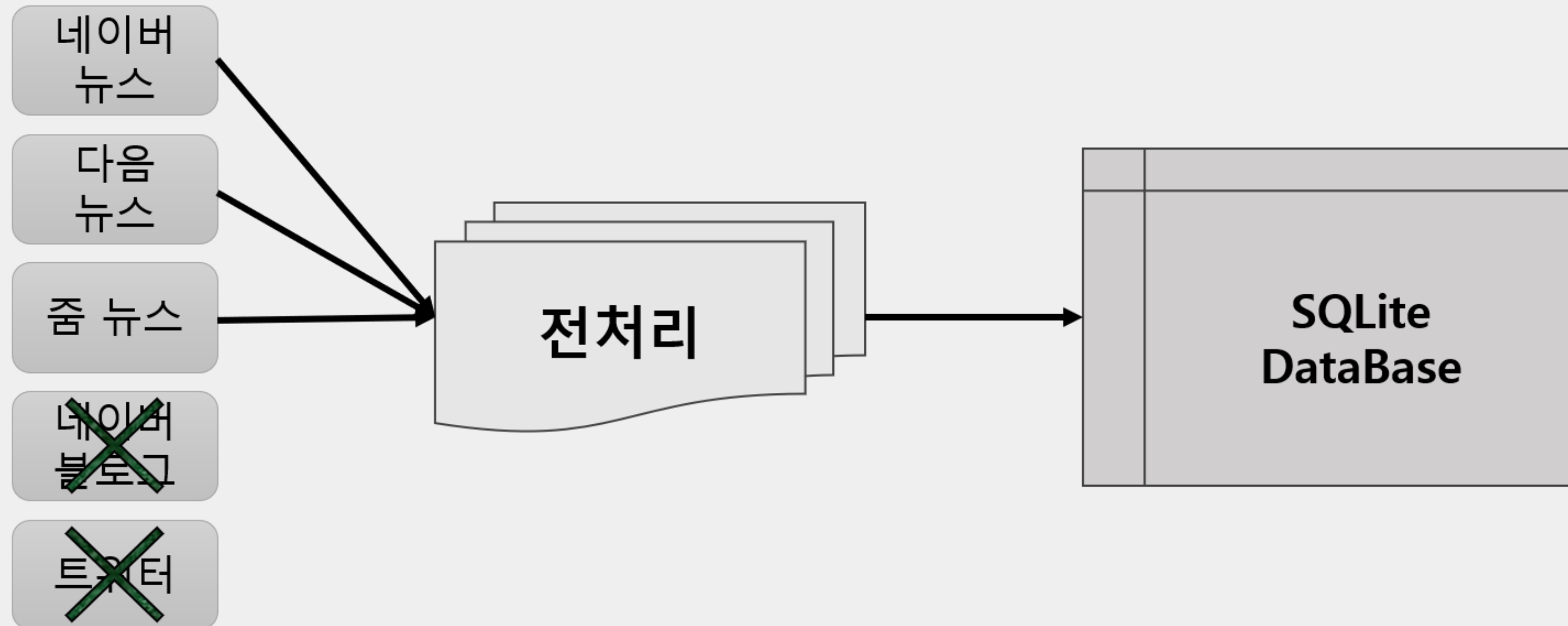
# 쿼리 실행하여 데이터프레임으로 저장
df = pd.read_sql_query(query, conn)

# 연결 종료
conn.close()
```

DB 구축 과정

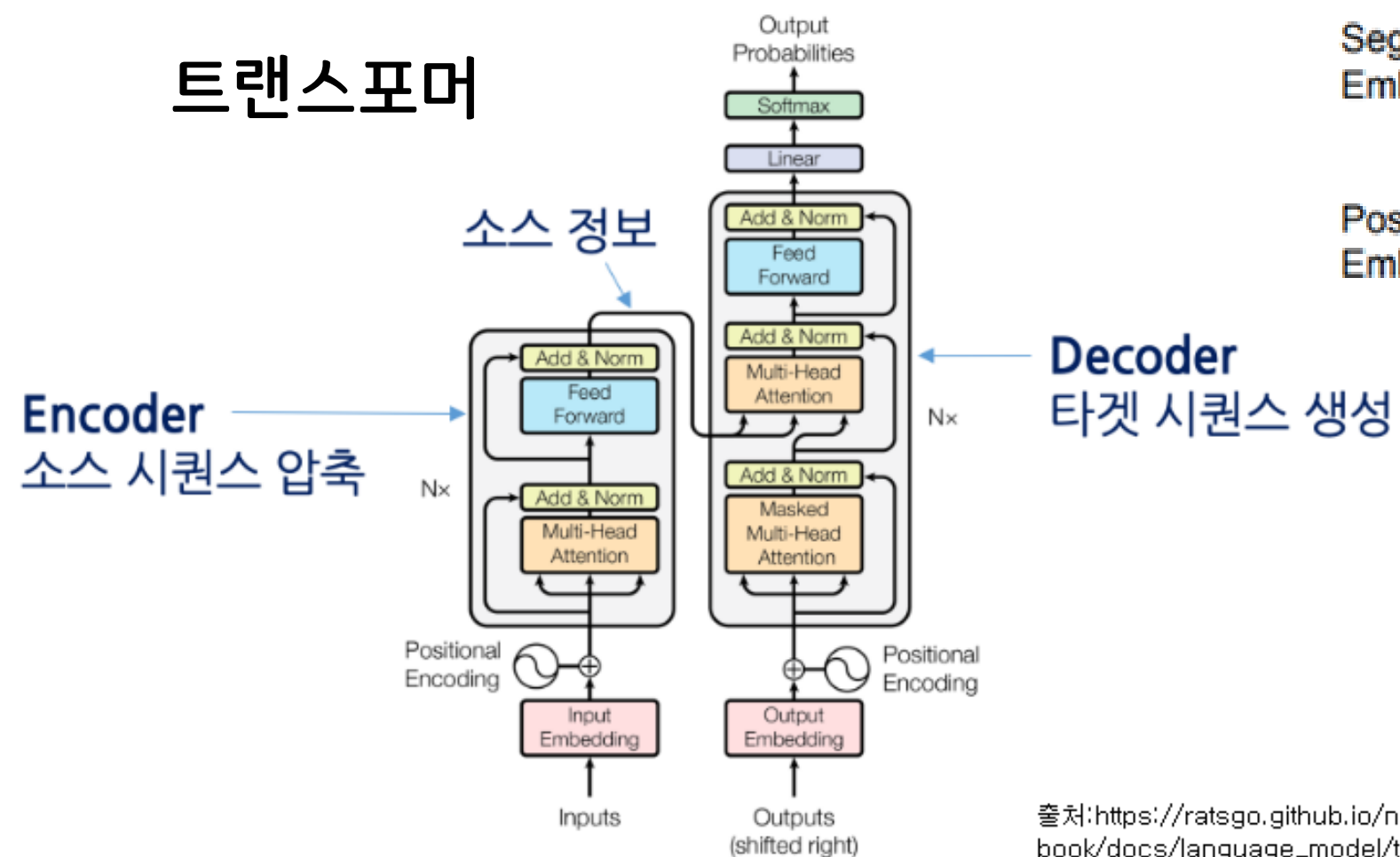
API 를 활용한 크롤링

SQL을 통한 DB 구축



모델 정리

BERT



Input	[CLS]	my	dog	is	cute	[SEP]	he	likes	play	##ing	[SEP]
Token Embeddings	$E_{[CLS]}$	E_{my}	E_{dog}	E_{is}	E_{cute}	$E_{[SEP]}$	E_{he}	E_{likes}	E_{play}	$E_{##ing}$	$E_{[SEP]}$
+	+	+	+	+	+	+	+	+	+	+	+
Segment Embeddings	E_A	E_A	E_A	E_A	E_A	E_A	E_B	E_B	E_B	E_B	E_B
+	+	+	+	+	+	+	+	+	+	+	+
Position Embeddings	E_0	E_1	E_2	E_3	E_4	E_5	E_6	E_7	E_8	E_9	E_{10}

Task별 사용 모델

Task별 최적의 결과 값을 찾기 위해
다양한 라이브러리의 모델 사용



Hugging Face

Task #1 문서 유사도 - Word2Vec -> fastText

각 Task별 적용 목적
텍스트(기사 본문)데이터
벡터화 작업

Word2Vec

사전 미등록 단어 처리 불가능



FastText

사전 미등록 단어도
subword 이용 임베딩 생성 가능

OOV(Out-of-Vocabulary)
단어 처리 효과적으로 수행

Task #1 문서 유사도 - fastText 구현 과정

fastText

'gensim' 자연어처리 파이썬 라이브러리
임베딩된 데이터를 DataFrame 열 추가

```
# 한글 텍스트로 Word2Vec 모델 학습
fasttext = FastText.load('fasttext_model.gensim')

# 텍스트를 임베딩하여 벡터로 변환하는 함수
def embed_text(text):
    words = text.split()
    vectors = [fasttext.wv[word] for word in words if word in fasttext.wv]
    if vectors:
        return sum(vectors) / len(vectors)
    else:
        return None

# DataFrame의 '한글텍스트' 열을 임베딩하여 'Embedding' 열 생성
df['Embedding'] = df['Content'].apply(embed_text)

# 검색어 임베딩 구하기
target_vector = embed_text(search_word).astype(float)
```

코사인 유사도

“검색어”와 “기사 본문”의
코사인 유사도 계산
높은 값을 가진 2개 본문 사용

```
# 코사인 유사도 계산 함수
def cosine_similarity(vector1, vector2):
    dot_product = np.dot(vector1, vector2)
    norm_vector1 = np.linalg.norm(vector1)
    norm_vector2 = np.linalg.norm(vector2)
    return dot_product / (norm_vector1 * norm_vector2)

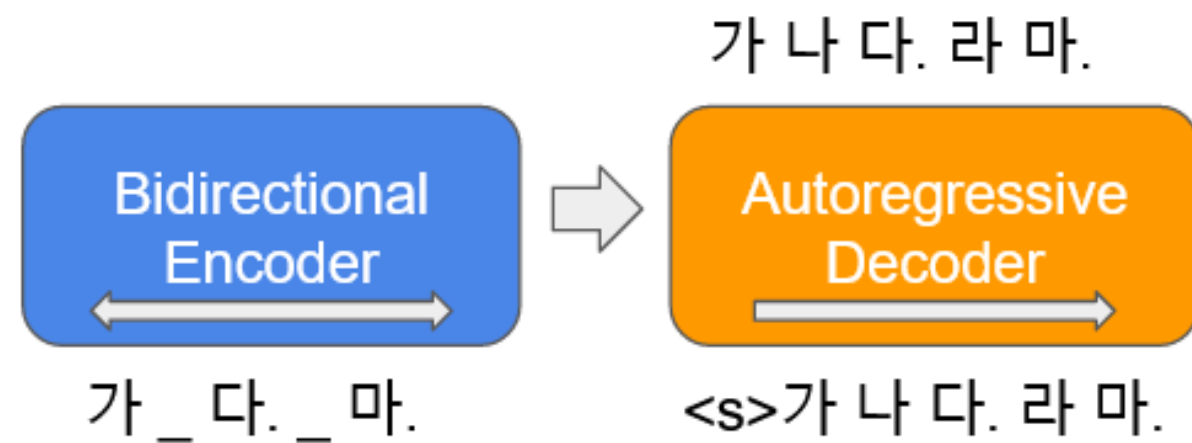
# 코사인 유사도 계산
similar_sentences = []
for index, row in df.iterrows():
    embedding = row['Embedding']
    similarity = cosine_similarity(target_vector, embedding)
    similar_sentences.append([row['Title'], row['Date'], row['Content'], similarity, row['Link']])

# 유사도가 높은 순서대로 정렬
similar_sentences.sort(key=lambda x: x[3], reverse=True)
```

Task #2 본문요약 – KoBART

KoBART

한국어 문장에 대해 양방향 인코딩을 수행하는
BART 아키텍처의 한국어 버전



Hugging Face

[ainize/kobart-news](#) 📄

Task #2 본문요약 – KoBART 구현과정

```
# 먼저 huggingface의 transformers 라이브러리 설치
!pip install transformers
```

```
# 모델과 문장 분리를 위한 토큰나이저 불러오기
from transformers import PreTrainedTokenizerFast, BartForConditionalGeneration
tokenizer = PreTrainedTokenizerFast.from_pretrained("ainize/kobart-news")
model = BartForConditionalGeneration.from_pretrained("ainize/kobart-news")
```

```
# 토큰나이저를 사용해 뉴스기사 원문을 모델이 인식할 수 있는 토큰형태로 바꿔주기
input_ids = tokenizer.encode(news_text, return_tensors="pt", max_length=1024)
```

```
# input_ids를 확인해보면 아래와 같이 텍스트가 토큰 단위로 쪼개어진 뒤 모두 id형태의 숫자로 바뀐 것을 확인할 수 있음.
# "0" : 문장 시작을 나타내는 토큰 id, "1" : 문장 끝을 나타내는 토큰 id
print(input_ids)
```

```
tensor([[ 0, 20873, 14091, 13831, 17822, 14067, 16185, 249, 12855, 20974,
         12332, 11224, 25121, 12855, 230, 25254, 255, 251, 325, 22341,
         17867, 14281, 14780, 18302, 17804, 14883, 230, 13699, 10773, 26632,
         15058, 14091, 13831, 17822, 18707, 27707, 14063, 13699, 10773, 11527,
         10761, 14240, 13090, 11786, 20569, 16036, 14834, 16841, 14035, 22853,
         14249, 15307, 20632, 230, 1542, 13699, 10773, 11527, 10761, 17674,
         11786, 20569, 238, 14077, 23348, 245, 239, 21673, 260, 13699,
         10773, 15719, 15110, 230, 230, 13699, 10773, 11527, 10761, 14240,
         13090, 11786, 20569, 12005, 16185, 249, 12855, 20974, 12332, 11224,
         25121, 12855, 243, 21670, 14114, 243, 17341, 23202, 325, 22341,
         17867, 14281, 14780, 18302, 8981, 16108, 14029, 17481, 15422, 10338,
         15401, 9908, 14130, 17341, 20087, 10872, 19468, 16228, 14948, 23202,
         17867, 264, 16457, 254, 17804, 14948, 23202, 17867, 265, 22034,
         17804, 14948, 23202, 17867, 266, 15431, 255, 17804, 14948, 22341,
         17867, 20017, 250, 17804, 10338, 14038, 14039, 15934, 14845, 16544,
         15787, 14904, 22537, 22646, 29939, 230, 20449, 12041, 15499, 16826,
         14998, 9115, 19396, 18285, 16841, 14035, 11374, 19683, 16790, 14043,
         22853, 14249, 15307, 14434, 21158, 22690, 14259, 16748, 14615, 20502,
         15818, 12005, 15639, 254, 15243, 14239, 15364, 16667, 8985, 19662,
         230, 13699, 10773, 11527, 10761, 14240, 13090, 11786, 20569, 12005,
         15935, 17822, 14162, 10329, 27675, 15390, 14343, 17181, 14130, 15935,
         17822, 14162, 10329, 24957, 15081, 17822, 20127, 14059, 15357, 12332,
         20260, 9698, 14061, 15624, 15401, 13607, 17963, 14357, 9507, 14058,
         20380, 9698, 15935, 14379, 14401, 15422, 15270, 14646, 14049, 15281,
```

input_ids : **토큰 형태**로 바꾼 뉴스 기사 원문

input_ids를 model.generate에 넣고 **요약문 생성**
결과값을 토큰나이저를 통해 **다시 텍스트 형태로 변경**

```
# 숫자로 변환된 뉴스 기사를 모델의 input으로 사용하여 생성된 요약문 확인
summary_text_ids = model.generate(
    input_ids=input_ids,
    bos_token_id = model.config.bos_token_id,
    eos_token_id = model.config.eos_token_id,
    length_penalty=2.0,
    max_length=1000,
    min_length=32,
    num_beams=4,
)
```

```
# 요약된 결과를 확인해보면 모두 id 형태(숫자)로 이루어져 있음
summary_text_ids
```

```
tensor([[ 2, 0, 13699, 11806, 26632, 15058, 14091, 13831, 17822, 18707,
         27707, 14063, 13699, 10773, 11527, 10761, 14240, 13090, 11786, 20569,
         16036, 14834, 16841, 14035, 22853, 14249, 15307, 14555, 18126, 14130,
         15058, 14249, 10213, 16631, 15429, 14742, 20069, 17715, 14025, 14187,
         14336, 14077, 10512, 12005, 14812, 25746, 9120, 27573, 21013, 14048,
         19041, 23289, 17999, 27992, 25689, 14032, 14082, 14186, 14377, 14417,
         27531, 14130, 1]])
```

```
# 토큰나이저를 사용해 이를 텍스트 형태로 바꾸기
print(tokenizer.decode(summary_text_ids[0], skip_special_tokens=True))
```

호연건설이 인천 연희공원 내에 공급하는 '호반써밋 파크에디션'이 오는 18일 1순위 청약을 접수한다. 인천 청라국제도시 바로 옆에 위치한 이 단지는 조망은 물론 산책과 휴식을 즐기는 등 쾌적한 주거환경을 누릴 수 있는 것이 가장 큰 장점이다.

Task #3 감성분석 - TEANAPS (Ko-BERT)

감성 분석

감성 분석

```
from teanaps.text_analysis import SentimentAnalysis
```

```
senti = SentimentAnalysis()  
sentence = "그의 말년에 특히 해파이스티온의 죽음 이후 알렉산더는 과대망상과 편집증 증세를 보이기 시작했다."  
result = senti.tag(sentence, neutral_th=0.3)  
result
```

[1] ✓ 4.9s

((0.8289, 0.2047), 'negative')

뉴스 본문내용 감성 분석
긍정, 부정 표현

teanaps 라이브러리의 'SentimentAnalysis' 사용
입력 문장에 대한 긍정, 부정값이 튜플 형태로 표현
긍, 부정 값이 50%를 넘기게 되면 결과값 도출

```
class SentimentAnalysis():  
    def __init__(self, model_path=con.SENTIMENT_MODEL_PATH, kobert_path=con.SENTIMENT_UTIL_PATH["kobert"]):  
        self.ctx = mx.cpu()  
        self.kobert_path = kobert_path  
        bert_base, vocab = self.__get_kobert_model()  
        self.model = BERTClassifier(bert_base, num_classes=2, dropout=0.1)  
        self.model.load_parameters(model_path, ctx=self.ctx)  
        tokenizer_path = con.SENTIMENT_UTIL_PATH["tokenizer"]  
        self.tok = nlp.data.BERTSPTokenizer(tokenizer_path, vocab, lower=False)  
        self.sa = SyntaxAnalyzer()  
  
    def tag(self, sentence, neutral_th=0.3):  
        sentence_list = [[sentence, '0']]  
        bert_sentence_list = BERTDataset(sentence_list, 0, 1, self.tok,  
                                         con.SENTIMENT_MODEL_CONFIG["max_len"], True, False)  
        gluon_sentence_list = mx.gluon.data.DataLoader(bert_sentence_list, num_workers=0,  
                                                       batch_size=int(con.SENTIMENT_MODEL_CONFIG["batch_size"]/2))  
  
        for t, v, s, label in gluon_sentence_list:  
            token_ids = t.as_in_context(self.ctx)  
            valid_length = v.as_in_context(self.ctx)  
            segment_ids = s.as_in_context(self.ctx)  
            label = label.as_in_context(self.ctx)  
            _, output = self.model(token_ids, segment_ids, valid_length.astype("float32"))  
            for r in output:  
                neg = 1/(1 + np.exp(-r[0].asnumpy()))[0]  
                pos = 1/(1 + np.exp(-r[1].asnumpy()))[0]  
                sentiment_label = "positive" if neg < pos else "negative"  
                if abs(neg - pos) < neutral_th:  
                    sentiment_label = "neutral"  
            return ((round(neg, 4), round(pos, 4)), sentiment_label)
```

Task #4 연관검색어 - TF-IDF

TF-IDF의 입력값 : 문서의 집합(뉴스 본문 데이터 전체)

≠

웹페이지 구현에 필요한 입력값 : 검색어



FastText로 모델 변경

TF-IDF: 자연어 처리에서 많이 사용되는 텍스트 마이닝 기법 중 하나
'특정 단어'의 문서 내 빈도와 문서 집합의 빈도로 가중치를 평가하는 방법

이를 통해 각 문서의 특성을 나타내는 벡터를 생성하여
문서 간 유사성이나 중요한 단어를 찾는 등
다양한 자연어 처리 작업 가능

Task #4 연관검색어 - TF-IDF코드

```
[ ] import joblib
    from sklearn.feature_extraction.text import TfidfVectorizer
```

```
[13] import pandas as pd
    from sklearn.feature_extraction.text import TfidfVectorizer
```

```
# 뉴스 본문 데이터 컬럼값을 리스트로 변환하여 documents에 할당
documents = df['content'].tolist()
```

```
# TF-IDF 벡터화
tfidf_vectorizer = TfidfVectorizer()
tfidf_matrix = tfidf_vectorizer.fit_transform(documents)
```

```
# 단어 목록
feature_names = tfidf_vectorizer.get_feature_names_out()
```

```
# TF-IDF 행렬 정보 출력
print(tfidf_matrix.toarray()) # 실제 행렬 데이터
print(feature_names) # 단어 목록
```

```
[[0.05096686 0.         ... 0.         0.02947294 0.         ]
 [0.         0.         0.         ... 0.         0.         0.         ]
 [0.04435441 0.         0.         ... 0.         0.         0.         ]
 ...
 [0.         0.         0.         ... 0.         0.         0.         ]
 [0.         0.         0.         ... 0.         0.         0.         ]
 [0.         0.         0.         ... 0.         0.         0.         ]]
['00' '000가구' '000만' ... '힘이' '힘입어' '힘찬']
```

```
# 모든 문서에 대한 TF-IDF 값의 평균 계산
avg_tfidf = tfidf_matrix.mean(axis=0).A1
```

```
# 상위 연관어 3개 추출
top_related_indices = avg_tfidf.argsort()[-3:][::-1]
top_related_words = [(feature_names[i], avg_tfidf[i]) for i in top_related_indices]
```

```
# 상위 연관어 출력
print("상위 연관어:")
for word, score in top_related_words:
    print(f"{word}: {score:.4f}")
```

```
상위 연관어 :
아파트: 0.0381
있다: 0.0328
인천: 0.0299
```

scikit-learn 라이브러리의 'TfidfVectorizer' 사용

텍스트 데이터를 TF-IDF 행렬로 변환

Task #4 연관검색어 – FastText

```
# Mecab 형태소 분석기 이용한 전처리 과정
import MeCab
from gensim.models import FastText
from gensim.models.keyedvectors import KeyedVectors
from collections import defaultdict
from operator import itemgetter
import pandas as pd

# Mecab 형태소 분석기 초기화
m = MeCab.Tagger()

# 특정 컬럼의 텍스트 데이터 가져오기 (실제 컬럼 이름으로 대체)
text_data = df['Content'].tolist()

# 명사 추출을 위한 함수
def extract_nouns(text):
    parsed_result = m.parse(text)
    lines = parsed_result.split('\n')
    nouns = []
    for line in lines:
        if '\t' in line:
            word, info = line.split('\t')
            pos_info = info.split(',')[0]
            if pos_info == 'NNG' and len(word) > 1: # 한 글자 명사 제거
                nouns.append(word)
    return nouns

# 각 문장에 대해 명사만 추출하여 리스트로 저장
nouns_list = [extract_nouns(sentence) for sentence in text_data]
```

✓ 1.8s

```
# FastText 모델 학습
model = FastText(sentences=nouns_list, size=100, window=5, min_count=1, sg=1)

# 학습된 모델 저장
model.save('fasttext_model.bin')
```

```
from gensim.models import FastText
```

```
# FastText 모델 로드 (unsupervised)
model_path = 'fasttext_model.bin' # 사전학습된 FastText 모델 파일 경로
fasttext_model = FastText.load(model_path)
```

```
related_words = fasttext_model.wv.most_similar('인천 서구', topn=3)
```

```
related_words
```

```
[('서구', 0.9950172901153564),
 ('종구', 0.802208423614502),
 ('강화군', 0.7843339443206787)]
```

형태소 분석기 Mecab 사용

명사인 품사만 추출하여 학습 데이터 구축

FastText 모델로 연관어 추출

속도 개선 작업! 저장 모델 불러오기

모델 저장

```
# 모델 저장 변수 이름.save_pretrained(원하는 디렉토리) 형태
tokenizer.save_pretrained('tokenizer.pt') # 파이토치 기반 모델
model.save_pretrained('model.pt') # 파이토치 기반 모델
```

모델 저장시 Hugging Face Transformers 라이브러리의 'save_pretrained' 함수 이용

모델 불러오기

```
# 모델 클래스 이름.from_pretrained(저장된 디렉토리) 형태
from transformers import PreTrainedTokenizerFast, BartForConditionalGeneration
tokenizer = PreTrainedTokenizerFast.from_pretrained("tokenizer.pt")
model = BartForConditionalGeneration.from_pretrained("model.pt")
```

모델 사용시 Hugging Face Transformers 라이브러리의 'from_pretrained' 함수 이용

웹 구현 과정 with 'django'



django



html

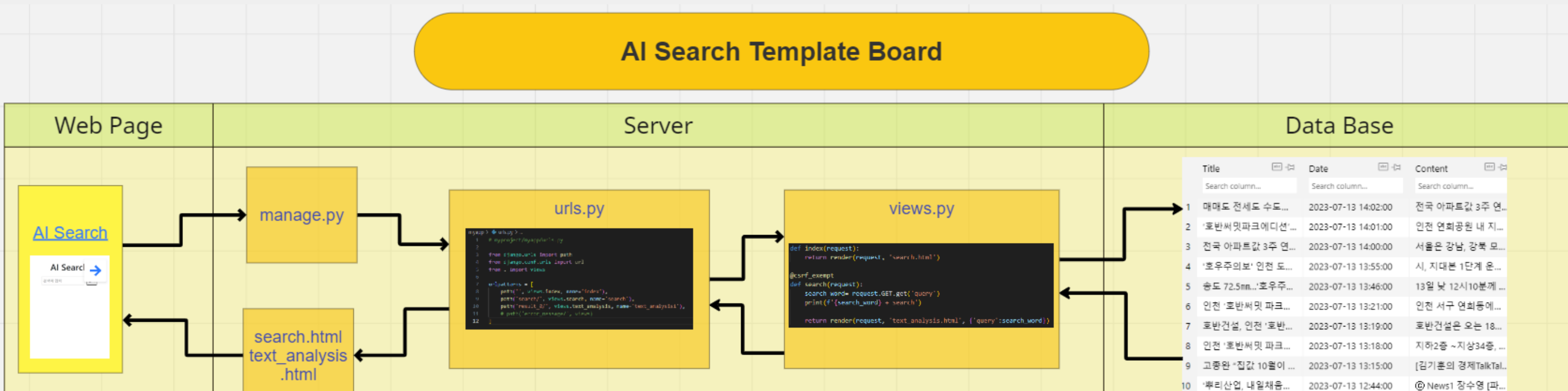


bootstrap

django?

파이썬 기반 웹 애플리케이션 프레임워크 오픈 소스 기술
개발 과정을 단축시키고, 부트스트랩(bootstrap)을 이용하여
다양한 추가 기능 구현 가능

웹 구현 및 시현 with 'django'



웹 구현 과정 with 'django'

< urls.py >

myapp >  urls.py > ...

```
1  # myproject/myapp/urls.py
2
3  from django.urls import path
4  from django.conf.urls import url
5  from . import views
6
7  urlpatterns = [
8      path('', views.index, name='index'),
9      path('search/', views.search, name='search'),
10     path('result_0/', views.text_analysis, name='text_analysis1'),
11     # path('error_message/', views)
12 ]
```

```
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('myapp.urls')),
    # path('', include('myapp.urls')),
]
```

manage.py 에서 runserver를 실행시
기본 제공되는 서버 활성화

웹 구현 과정 with 'django'

< views.py >

```
context = {
    # 검색어
    'search_word' : search_word,

    # 유사도 1등 기사
    'title1' : f"{df_search[0][0]}",
    'date1' : f"{df_search[0][1]}",
    'result_sen1': f"{result_sen1}",
    'result_sum1': result_sum1,
    'link1' : f"{df_search[0][4]}",

    # 유사도 2등 기사
    'title2' : f"{df_search[1][0]}",
    'date2' : f"{df_search[1][1]}",
    'result_sen2': f"{result_sen2}",
    'result_sum2': result_sum2,
    'link2' : f"{df_search[1][4]}",

    # 연관어 리스트
    'related_word_list': related_word_list
}

return JsonResponse(context)

return render(request, 'text_analysis.html')
```

```
# 감성분석
senti = SentimentAnalysis()
result_sen1 = senti.tag(text1, neutral_th=0.5)
result_sen2 = senti.tag(text2, neutral_th=0.5)
print('senti')

# 본문 요약
tokenizer = PreTrainedTokenizerFast.from_pretrained("tokenizer.pt")
model = BartForConditionalGeneration.from_pretrained("model.pt")

input_ids1 = tokenizer.encode(text1, return_tensors="pt", max_length=1024)
input_ids2 = tokenizer.encode(text2, return_tensors="pt", max_length=1024)

summary_text_ids1 = model.generate(
    input_ids=input_ids1,
    bos_token_id = model.config.bos_token_id,
    eos_token_id = model.config.eos_token_id,
    length_penalty=2.0,
    max_length=1000,
    min_length=32,
    num_beams=4,
)

summary_text_ids2 = model.generate(
    input_ids=input_ids2,
    bos_token_id = model.config.bos_token_id,
    eos_token_id = model.config.eos_token_id,
    length_penalty=2.0,
    max_length=1000,
    min_length=32,
    num_beams=4,
)

result_sum1 = tokenizer.decode(summary_text_ids1[0], skip_special_tokens=True)
result_sum2 = tokenizer.decode(summary_text_ids2[0], skip_special_tokens=True)
```

```
def text_analysis(request):
    if request.method == 'POST':

        # 텍스트 추출
        search_word = request.POST.get('query')
        print(f'{search_word} + 텍스트 추출')

        # DB를 통해서 df 불러오기
        df_search = load_text(search_word)
        print('DB_loaded')

        # 본문 유사도가 낮을 경우 에러메세지 출력
        # if df_search[0][3] < 0.99 or df_search[1][3] < 0.99:
        #     error_message = '검색결과없음'
        #     return render(request, 'error_page.html', {'error_message': error_message})

        # 연관어추출 FastText 모델 로드 (unsupervised)
        model_path = '/home/myproject/fasttext_model.bin'
        fasttext_model = FastText.load(model_path)
        print('모델 로드됨')

        related_words = fasttext_model.wv.most_similar(search_word, topn=3)
        print('연관어 추출됨')

        related_word_list = [word for word, _ in related_words]
        print('리스트로 묶음')

        # 본문기사 변수로 지정
        text1 = df_search[0][2]
        text2 = df_search[1][2]
```

웹 구현 과정 with 'django'

< views.py >

```
# DataFrame의 '한글텍스트' 열을 임베딩하여 'Embedding' 열 생성
df['Embedding'] = df['Content'].apply(embed_text)

# 검색어 임베딩 구하기
target_vector = embed_text(search_word).astype(float)

# 코사인 유사도 계산 함수
def cosine_similarity(vector1, vector2):
    dot_product = np.dot(vector1, vector2)
    norm_vector1 = np.linalg.norm(vector1)
    norm_vector2 = np.linalg.norm(vector2)
    return dot_product / (norm_vector1 * norm_vector2)

# 코사인 유사도 계산
similar_sentences = []
for index, row in df.iterrows():
    embedding = row['Embedding']
    similarity = cosine_similarity(target_vector, embedding)
    similar_sentences.append([row['Title'], row['Date'], row['Content'], similarity, row['Link']])

# 유사도가 높은 순서대로 정렬
similar_sentences.sort(key=lambda x: x[3], reverse=True)

# 짧은 text 전처리 함수
def processing_text(text):
    content_text = BeautifulSoup(text, "html5lib").get_text() # HTML 태그 제거
    content_text = re.sub("\n", " ", content_text)
    return content_text
```

```
# 1번째와 2번째 본문에 대한 전처리
similar_sentences[0][2] = processing_text(similar_sentences[0][2])
similar_sentences[1][2] = processing_text(similar_sentences[1][2])

return similar_sentences[:2] # 첫번째, 두번째 기사정보 list 형태

def load_text(search_word):
    print(search_word)

    # SQLite 데이터베이스에 연결
    # conn = sqlite3.connect(f'{search_word}')
    conn = sqlite3.connect('인천서구.db')

    # 테이블 조회 쿼리
    query = "SELECT * FROM Table_"

    # 쿼리 실행하여 데이터프레임으로 저장
    df = pd.read_sql_query(query, conn)

    # 연결 종료
    conn.close()

    # 한글 텍스트로 fasttext 모델
    fasttext = FastText.load('fasttext_model.gensim')

    # 텍스트를 임베딩하여 벡터로 변환하는 함수
    def embed_text(text):
        words = text.split()
        vectors = [fasttext.wv[word] for word in words if word in fasttext.wv]
        if vectors:
            return sum(vectors) / len(vectors)
        else:
            return None
```

웹 구현 과정 with 'django'

< search.html >

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>search</title>
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@4.4.1/dist/css/bootstrap.min.css" rel="stylesheet"
  <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
  <style>
    body, html {
      height: 80%;
      margin: 0;
      display: flex;
      justify-content: center;
      align-items: center;
    }
    .container {
      text-align: center;
    }
    #searchForm {
      display: flex; /* Flexbox를 사용하여 요소를 나란히 배치 */
      align-items: center; /* 요소를 수직 정렬 */
      margin-top: 20px;
    }
    #query {
      padding: 5px;
      border: 1px solid #ccc;
      margin-right: 10px; /*오른쪽 마진 추가*/
    }
    button {
      padding: 5px 5px;
      white-space: nowrap; /* 텍스트 줄바꿈 방지 */
    }
  </style>
```

```
<script>
  $(document).ready(function() {
    $("#searchForm").submit(function(event) {
      event.preventDefault(); // 폼 제출 막기
      var query = $("#query").val(); // 입력한 검색어 가져오기
      window.location.href = "{% url 'search' %}?query=" + query;
    });
  });
</script>

</head>
<body>
  <div class="container my-3">
    <div class="row">
      <div class="col-md-8 col-lg-9">
        <h1 class="text-center">AI Search</h1>
        <form id="searchForm">
          {% csrf_token %}
          <input type="text" id="query" placeholder="검색어 입력">
          <button type="submit">검색</button>
        </form>
      </div>
    </div>
  </div>
</body>
```

웹 구현 과정 with 'django'

< text_analysis.html > 중 일부

< 검색 창>

```
<form method="post" class="analyze-form">
  {% csrf_token %}
  <textarea name="query" id="query" rows="2" cols="80">{{ query }}</textarea><br>
  <button type="button" id="newSearchButton">검색</button><br>
  <input type="submit" id="submitButton" value="분석" style="display: none;"><br>
</form>
```

< 연관검색어>

```
<div class="col-lg-6">
  <a href="#" id="related_word1"></a>
  <a href="#" id="related_word2"></a>
  <a href="#" id="related_word3"></a>
</div>
```

< 자바스크립트 코드> 일부

```
// 검색버튼 눌렀을 때, 웹페이지 찾아가기
$("#newSearchButton").click(function() {
  var query = $("#query").val(); // 입력한 검색어 가져오기
  window.location.href = "{% url 'search' %}?query=" + query;
});

document.getElementById("related_word1").addEventListener("click", function() {
  event.preventDefault(); // 기본 동작(링크 이동) 방지
  const relatedWord = this.textContent; // 연관검색어 텍스트
  const newURL = "/search/?query=" + encodeURIComponent(relatedWord); // 새로운 URL 생성
  window.location.href = newURL; // 페이지 리다이렉트
});

document.getElementById("related_word2").addEventListener("click", function() {
  event.preventDefault(); // 기본 동작(링크 이동) 방지
  const relatedWord = this.textContent;
  const newURL = "/search/?query=" + encodeURIComponent(relatedWord);
  window.location.href = newURL;
});

document.getElementById("related_word3").addEventListener("click", function() {
  event.preventDefault(); // 기본 동작(링크 이동) 방지
  const relatedWord = this.textContent;
  const newURL = "/search/?query=" + encodeURIComponent(relatedWord);
  window.location.href = newURL;
});
```

```
$.ajax({
  type: "POST",
  url: "{% url 'text_analysis1' %}",
  data: formData,
  beforeSend: function () {
  },
  success: function (response) {

    // 검색버튼 활성화
    $("#newSearchButton").prop("disabled", false); // 버튼 활성화
    $("#articleLink1").removeClass('disabled');
    $("#articleLink2").removeClass("disabled");

    // 기본 정보

    $("#date1").html(response.date1);
    $("#date2").html(response.date2);

    $("#articleLink1").html(response.title1);
    $("#articleLink1").attr("href", response.link1);
    $("#articleLink2").html(response.title2);
    $("#articleLink2").attr("href", response.link2);

    $(".result-sum1").html(response.result_sum1);
    $(".result-sum2").html(response.result_sum2);

    // 연관어 리스트
    $("#related_word1").html(response.related_word_list[0]);
    $("#related_word2").html(response.related_word_list[1]);
    $("#related_word3").html(response.related_word_list[2]);
  }
});
```

```
// // 비동기 함수가 실행되면 원그리기 함수 호출
var result_sen1_neg = response.result_sen1_neg;
var result_sen1_pos = response.result_sen1_pos;
var result_sen1_text = response.result_sen1_text;
var result_sen2_neg = response.result_sen2_neg;
var result_sen2_pos = response.result_sen2_pos;
var result_sen2_text = response.result_sen2_text

executeAsyncTasks('Chart1', result_sen1_neg, result_sen1_pos);
executeAsyncTasks('Chart2', result_sen2_neg, result_sen2_pos);
},
});
```

웹 구현 및 시현 with 'django'

AI Search

검색어 입력

검색

웹페이지 첫 화면

AI Search

인천서구

검색

웹페이지 검색어 입력

웹 구현 및 시현 with 'django'

AI Search

Search (추천 검색어: 인천서구)

인천서구

검색

연관 검색어

기사 제목 #1

발행일 #1

본문 요약 중...

감성분석

기사 제목 #2

발행일 #2

본문 요약 중...

감성분석

검색 진행 중

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  2

인천서구
DB연결
모델로드
임베딩
유사도 끝
DB_loaded
모델 로드됨
연관어 추출됨
리스트로 묶음
senti
You passed along `num_labels=3` with an incompatible id
'1': 'POSITIVE'}. The number of labels will be overwritt
Truncation was not explicitly activated but `max_length`
please use `truncation=True` to explicitly truncate exam
to 'longest_first' truncation strategy. If you encode pa
ith the tokenizer you can select this strategy more prec
strategy to `truncation`.
[]

Container bianne1004/my-ubuntu:v1.5 (sun...  0 0  3  UTF
```

django server 가동 과정

웹 구현 및 시현 with 'django'

연관 검색어

서구 중구 연합

인천서 모하비 차량 3m 아래 놀이터 추락...보조석 60대 여성 사망

2023-07-10 18:37:00

인천시 서구의 한 아파트 놀이터에서 모하비 차량이 추락해 보조석에 타고 있던 60대 여성이 숨졌고, 운전자 B씨(70대)가 경상을 입고 병원에서 치료를 받고 있으며 모하비 차량 보조석에 타고 있던 A씨(60대·여)가 숨졌고, 운전자 B씨(70대)가 경상을 입고 병원에서 치료를 받고 있고 모하비 차량 보조석에 타고 있던 A씨(60대·여)가 숨졌고, 운전자 B씨(70대)가 경상을 입고 병원에서 치료를 받고 있다.

감성분석



인천 아파트서 20~30m 아래로 SUV 추락...1명 사망·1명 경상

2023-07-10 18:35:00

인천소방본부 등에 따르면 10일 인천소방본부 등에 따르면 이날 오후 4시52분 인천서구 가정동 한 아파트에서 스포츠유틸리티차량(SUV)이 20~30m 아래인 옆 단지 놀이터로 추락해 동승자 B(63·여)씨가 심정지 상태로 병원으로 옮겨졌으나 끝내 숨졌으며, 운전자 A씨는 열상 등 경상을 입었다.

감성분석



연관 검색어 출력 (3개)

유사도 순 기사 2개 출력

우측 감성분석 결과 출력
(기사의 대한 긍/부정)

결론

감성분석, 본문요약, 연관어 추천 기술적 구현

제공하는 기능을 활용한 새로운 정보 추출 가능

시간적, 자원적 한계 / 한국어 처리 모델 한계

조원 자체 평가 및 소감



팀장 김기범

검색 솔루션이라는 최종 목표를 구현하기 위해 배워보지 않았던 장고, html, 도커 등의 여러 플랫폼을 다뤄보며 팀원들과 역할을 분담하여 최종적으로 구현하고자 했던 기능들을 구현할 수 있었습니다.

Teanaps 모듈이 도커에서만 가능했기 때문에 처음 다뤄보는 vscode의 extension을 활용해서 도커를 연결하는데 성공했고 이미지를 도커허브에 올림으로써 환경을 공유하는 방법에 대해 알 수 있었습니다.

장고의 구조를 파악하기 위해 그림을 그려가며 장고의 각 변수나 객체들이 어떻게 assign 되는지 분석할 수 있었습니다. 또한 팀원들과 상의하며 서로 어려운 점을 공유하고 해결해 나가며 팀워크에 대한 중요성과 팀장으로서의 해야 할 역할에 대해 알 수 있었습니다.

웹서버에서 받은 input을 알고리즘 모델로 가져와 output을 도출하고 로컬 웹 서버에 구현까지 성공하면서, 하나의 서버 구현에 필요한 전반적인 개발과정을 경험하고, 파악할 수 있었던 좋은 경험이었습니다.



김연준

프로젝트를 진행하면서 지난 6개월간 학습했던 내용뿐만 아니라 SQL, 장고, 도커 등 여러 툴을 다루게 되어서 새롭게 느껴졌습니다.

또한, 자연어 처리를 위해 KoBART, Teanaps 등 여러 모듈의 다양한 모델을 사용해 보는 경험을 할 수 있었습니다.

저희 팀의 최종 결과물에 만족하지만, 부족한 점도 많습니다. 마감이 다가 올 수록 더 구현하고 싶은 과제도 많기도 했지만, 물리적 한계로 어쩔 수 없이 타협해야할 부분도 있었습니다.

하지만, 이번 프로젝트를 무사히 마쳤다는 부분에 자신감을 갖게 되었고, 지속적인 학습과 개발을 통해 더 나은 모습을 갖고자하는 계기가 되었습니다.

또한, 지난 5주간 프로젝트를 진행하면서, 파이썬의 신규 버전이나, 자연어 처리 관련 모델들이 계속 나오는것들을 보면서, 뒤쳐지지 않으려면 계속 노력하고 공부해야겠다는 생각을 하게 되었습니다.

조원 자체 평가 및 소감



손수한

처음해보는 기업프로젝트이고, 처음해보는 자연어처리 모델링이기에 긴장을 하며 프로젝트를 시작하였습니다. 멘토님은 하나의 기능을 구현하더라도 여러가지 방법과 경험치를 쌓는것을 중요시 하셨습니다. 덕분에 저희의 데이터 수집 과정은 다양한 방법과 다양한 사이트의 크롤링을 경험하며 진행하고 모델링또한 다양한 방법과 경험을 하도록 지도하셨습니다. 비록 속도는 더디였지만, 다양한 경험과 노력을 하게되었습니다. 가상환경을 다루고 DB를 다루고 Django를 다루었습니다. BERT와 트랜스포머에 대해 공부하게 되었습니다.

무엇보다 성공적으로 프로젝트를 구현하게 된것은 조원들의 도움이 컸습니다. 가장 크게 깨닫은 점은 혼자서는 프로젝트를 진행할 수 없다는것과 팀원과의 원활한 의사소통, 포기하지 않는 마음가짐이 정말 중요하다는것을 깨달았습니다. 마지막 프로젝트 답게 여러가지 기능 구현과, 여러가지 경험을 하였고 데이터 프로젝트와 직무에 실제로 중요한것이 무엇인지 깨닫게 해주는 좋은 경험이었습니다



현재은

이번 프로젝트를 통해 장고, 도커 등 새로운 툴들을 다뤄볼 수 있었고 TF-IDF, FastText, KoBART 등 여러 자연어 처리 관련 모델을 접해볼 수 있어서 좋았습니다.

처음에는 저희의 주제가 명확하게 느껴지지 않고 뜬구름 잡는 느낌이 들었지만 멘토님과의 면담, 조원들과의 회의를 통해 주제가 점점 명확해졌습니다.

진행 도중에 도커 세팅, 모델 다운받기 등의 과정에서 문제를 겪기도 하였고 처음에 목적으로 주제보다는 부족한 결과물이 나왔지만 구현하는 과정을 통해 얻은 것이 많기에 만족스럽게 생각합니다.

또한 이를 바탕으로 저희의 결과물을 더 발전시킬 수 있을것같아 기대가 됩니다.

THANK YOU