뉴키주

뉴스 요약 및 키워드 추출과 긍부정 분류(Final Project)

Wrap-Up 리포트

2023.07.03 ~ 2023.07.28

NLP-4 조(데굴^2)

곽민석_T5014

이인균_T5150

임하림_T5178

최휘민_T5221

황윤기_T5229

1. 프로젝트 개요

프로젝트 주제

주식을 새로 시작하는 투자자들이 가장 힘들어하는 것은 유용한 정보를 찾고 그것을 종합하는 것이라고 생각한다. 통계에 따르면 대부분의 사람들은 언론을 통해서 투자 정보를 접하지만 포털 사이트를 들어가 보면 무엇을 읽어야할지, 중요한 내용은 무엇인지, 이 내용이 기업에 긍정적인지 부정적인지 한 눈에 볼 수 없을 정도로 많은 양의 데이터가 쏟아지듯 제공되고 있다.

따라서 **"뉴"스** 기사에서 기업에 대한 긍정적이거나 부정적인 **"키"워드**를 추출하여 사용자에게 시장 동향을 파악하고 **"주"식** 투자에 도움이 될 수 있는 서비스를 개발하였다. 이런 프로젝트 내용을 따서 **"뉴키주"**라고 이름을 지었다.

프로젝트 구현 내용, 컨셉, 교육 내용과의 관련성 등

프로젝트 주제에 맞춰 방대한 뉴스 기사로부터 키워드를 추출한다. 그 후 기사의 긍부정 여부를 판단한 뒤 이를 키워드의 긍부정 여부에 대한 대리 지표로 사용한다. 사용자가 키워드를 클릭하면 요약된 뉴스 기사가 제공될 수 있게끔 기사를 요약하여 저장한다.

서빙은 Batch-Serving 으로 계획하였기 때문에 속도보다는 정확성에 목표를 맞춰서 모델 구조 변경, 데이터 변형 등을 시도한다.

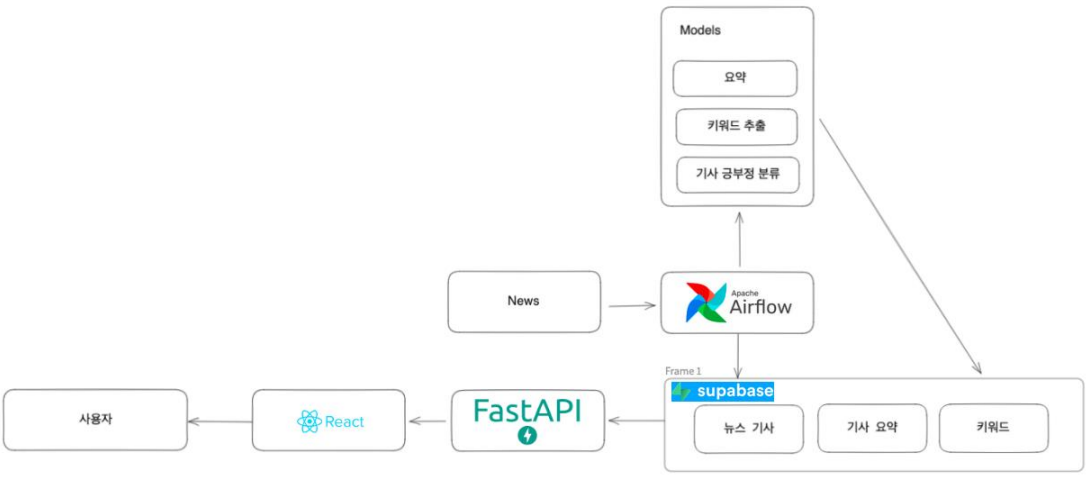
내용에 맞춰서 키워드 추출 모델, 감성 분석 모델, 자연어 요약 모델 3 가지 모델을 구현한다. End-to-End 서비스 배포를 위하여 프론트엔드와 백엔드 구현 역시 이루어진다.

활용 장비 및 재료(개발 환경, 협업 tool 등)

(팀 구성 및 컴퓨팅 환경) 5 인 1 팀으로 각자에게 지원되는 V100 서버를 활용하여 각자 VSCode 혹은 PyCharm 등 편한 IDE 를 사용한다.

(협업 환경 및 의사 소통) Notion, Github, Wandb, Slack, Zoom

프로젝트 구조



프로젝트 팀 구성 및 역할

이름	역할
곽민석	Summarization, Labeling using LLM, Frontend
이인균	Sentiment Analysis
임하림	Sentiment Analysis, Data preprocessing
최휘민	Keyword Extraction, Labeling
황윤기	Keyword Extraction, Data Crawling, AirFlow, Project Leader

2. 프로젝트 수행 절차 및 방법

전체 프로세스

데이터 크롤링

KRX300 지수 상위 30 개 기업에 대한 뉴스를 수집하였다. 프로젝트 수행 기간상 뉴스별 기업 분류 작업은 생략하기 위해, 네이버 증권 종목에 뉴스 및 공시에 기업별로 올라오는 뉴스를 수집하고, 해당 뉴스를 해당 기업으로 라벨링했다. 많은 데이터의 빠른 수집을 위해 C 언어로 제작된 HTML 렌더러를 Python 으로 wrapping 한 `Selectolax`를 사용했다. 이미 존재하는 데이터의 수집을 방지하여, 30 개 기업의 크롤링 속도를 1 분 이내로 줄일 수 있었다.

기사 전처리

수집한 뉴스에 대해서 전처리를 진행하였다. 정말 다양한 경우가 있었고, 모든 기사에 대해 확인을 할 수 없었으므로 하나씩 찾아 해결하는 string 기반의 replace 함수로는 명확한 한계가 있었다. 그래서 노이즈들의 특징들을 잡아 정규표현식 기반의 sub 함수로 전처리를 진행했다. 기사의 의미와는 관련 없는 기자 이름, 소속, 사진 등등이 있었고, UNK 토큰을 유발할 수 있는 특수기호를 처리하는 것을 목표로 했고 해당하는 10 개에 해당하는 항목을 찾아 전처리를 진행했다.

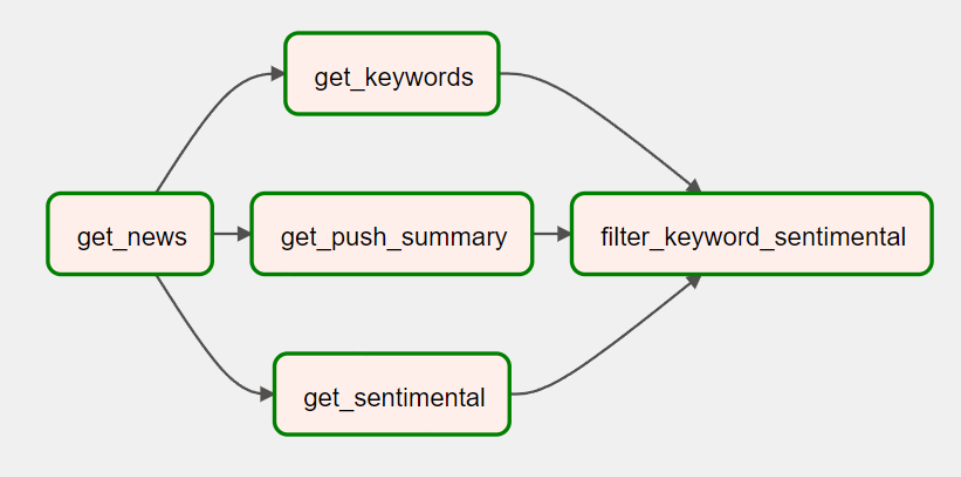
이를 통해 키워드 추출 진행시 발생했던 언론사, 기자, 이미지 등등이 추출되는 것을 방지했고, 요약 모델과 긍부정 판별은 입력의 토큰 수를 줄이는 것 뿐만 아니라 기사의 본문 내용 과는 관련 없는 내용들은 배제하고 요약과 기사 감성분석을 추출할 수 있게 되었다,

모델 적용 및 서빙

- 기사의 최신화를 위해 주기적인 수집을 Airflow 로 진행하였으며, 서빙 또한 Airflow 로 키워드의

주기적인 업데이트를 진행했다.

- 모델 서빙을 위한 Airflow DAG 내부의 Task 구조는 아래와 같이 구성되어 있다.



- `get_news` : DAG 에 넘겨지는 execution Time 이전 7 시간 동안의 뉴스를 DB 에서 받아온다.
- `get_keywords` : 앞서 넘겨진 기사의 키워드 추출을 API 를 통해 진행한다.
- `get_push_summary` : 앞서 넘겨진 기사의 요약을 API 를 통해 진행하고, DB 에 Insert 한다.
- `get_sentimental` : 앞서 넘겨진 기사의 긍부정을 API 를 통해 진행한다.
- `filter_keyword_sentilmental` : 앞서 넘겨진 키워드와 긍부정 자료들을 이용하여, 키워드의 긍부정을 분류하고 DB 에 Insert 한다.

기사 긍 / 부정 데이터셋 라벨링

현재 이용 가능한 데이터셋에 대해서 기사의 긍/부정이 분류된 데이터셋은 존재하지 않았다. 다만, 경제 기사 내의 한 문장에 대해 긍/부정이 나누어진 영문 데이터셋은 존재했는데, 이를 기반으로 기사 전문에 대해 긍/부정 판단이 가능할 것으로 판단하여 기사 전문에 대한 라벨 데이터를 만들기로 결정하였다.

처음에 사람이 읽고 라벨링을 하는 것에 대해 고려했었으나, 팀원들의 작업량, 긍/부정에 대한 기준 설립과 같은 과제가 산재하여 LLM 을 이용한 라벨링을 사용하기로 결정하였다. 후보 LLM 에는 GPT, Bard 그리고 Clova 가 있었다. Bard 의 경우 API 호출 제한으로 인하여 사용이 불가능하였고, Clova 의 경우 출력 형식(JSON)은 올바르나, 모델에서 뽑은 라벨링 기준이 상대적으로 모호하여 GPT 를 사용하였다.

GPT 모델의 라벨링 성능을 강화하기 위해 먼저, 모델의 역할을 부여하였다. 모델의 역할은 증권사 재무팀에서 근무하는 애널리스트로 기사의 긍/부정을 판단하는것이 주 업무임을 명시하였다. 두 번째로 선택가능한 라벨들과 출력 규정과 같은 생성시 지켜야 하는 규칙을 명시하였다. 또한, 출력시 어떠한 이유로 해당 라벨을 선택하였는지 이유를 작성하도록 하여 유사 Chain-of-Thought 를 도입하였다. 마지막으로 출력 형식은 바로 사용이 가능하도록 JSON 형식을 예시로 주었다.

이를 종합하면 아래와 같은 프롬프트가 완성된다.

```
### 역할:
너는 어느 증권사 재무팀에서 일하고 있는 애널리스트야. 너의 주 업무는 해당 기사가
"긍정" 혹은 "부정"인지 라벨링 하는 것이야. 이때 아래의 규칙을 따라야해.

### 기사:
{news}
```

규칙:
1. 주어진 기사에 대해 하나의 ["긍정", "부정"]만을 선택하여 라벨링한다.
2. 기사는 "### 기사:" 부분부터 시작된다.
3. 출력은 "### 출력"와 같이 json 형식으로 출력한다.
4. 라벨링은 기사 전문에 대해서 라벨링한다.
5. 중립은 선택지에 없다.
6. 출력은 형식외에 아무것도 작성하지 않는다.

출력
{{"label": "긍정", "reason": "새로운 투자 유치로 인해 해당 기업의 전망이 밝을것으로 예측된다.
"}}

하지만, 해당 프롬프트를 보게 된다면 긍/부정의 주체가 명확하지 않다. 그래서 기존 데이터셋에 "company" column 을 추가하여 어느 기업 입장에서 평가를 해야하는 지 명확하게 적어주었다. 그래서 최종적으로 적용한 프롬프트는 아래와 같다.

'### 역할:
너는 어느 증권사 재무팀에서 일하고 있는 애널리스트야. 너의 주 업무는 뉴스 기사를 보고 {company} 입장에서 해당 기사가 "긍정" 혹은 "부정"인지 라벨링 하는 것이야. 이때 아래의 규칙을 따라야해.

... 이하 생략 ...

결과적으로 위의 최종 프롬프트를 이용하여 데이터셋을 생성하여 모델에 적용하였다.

키워드 추출

본 프로젝트에선 특정 기간 내의 뉴스 기사들의 주요 키워드를 나열해 보여준다. 따라서 주기적으로 각각의 새로운 뉴스 기사에서 키워드를 추출할 필요가 있다. 본 프로젝트는 키워드를 추출하는 방법론으로 비지도 방식의 키워드 추출 방법인 KeyBERT 기반의 모델을 구현하고, 또한 유사한 구조의 자체 모델을 구현하였다. 두 가지 모델을 사용한 이유는 한국어 데이터의 키워드 추출이 데이터셋이 존재하지 않았기 때문이다. 따라서 앞선 두 가지 비지도 방법론의 사용으로 학습데이터가 없는 문제를 해결하면서 다양한 관점, 결과의 신뢰성, 성능 비교를 통한 정확한 결과를 얻고자 하였다. 또한 정교한 평가를 위해 50 개의 기사로부터 키워드를 직접 추출하여 데이터셋을 제작하였다.

두 방법론 모두 위의 프로세스의 구조로 구현되었다

KeyBERT 기반 모델

주어진 문서에서 Vectorizer 를 이용해 단어별 가중치를 얻는다. 해당 가중치를 키워드 후보 단어로 하여, 후보 단어와 문서를 `Sentence-Transformer`모델로 임베딩을 진행한다. (1). 키워드 후보 임베딩과 문서 임베딩을 코사인 유사도로 점수를 일차적으로 내고, (2). 키워드 후보 임베딩 자기 자신과의 코사인 유사도 점수를 냄으로써, 단어 간의 유사도 점수를 얻는다.

(1). 을 통해 계산된 가장 높은 유사도 점수를 가지는 키워드가 일차적으로 추출되는 키워드가 되고, 키워드 후보에서는 해당 키워드를 제거한다. 일차적으로 추출된 키워드의 경향성을 유지하기 위해 MMR(Maximal Marginal Relevance) Score 를 계산하는데, 계산식은 아래와 같다.

$$MMR\ Score\ =\ (1-diversity)\ \cdot\ candidate\ sim\ -\ diversity\ \cdot\ t\arg e\,t\ sim$$

(1). 에서 키워드 후보의 점수(candidate sim)와, 이미 추출된 키워드의 점수(target sim)를 얻는다. 그것을 통해 diversity 를 이용해 원래 추출된 키워드와의 연관성을 고려하도록 점수를 낸다. 그 후 가장 높은 점수를 가지는 키워드를 추출한다.

기본적인 KeyBERT 모델은 위와 같이 구성되어 있고, 키워드 추출의 성능 향상을 위해서 몇가지 변형을 도입했다. 단어별 가중치를 이용할 때 모든 단어들을 사용하게 될 때, 중요하지 않은 단어들이 유사도 계산에 이용되어 정확하지 않은 키워드들이 추출됐다. 그래서 단어별 가중치의 상위 일정 비율만을 이용하도록 구성하였다.

또한 추출된 키워드들이 토크나이징이 제대로 진행되지 않는 문제점이 있어서, 단어 단위로 추출되지 않고 문자일부가 추출되는 경우가 비일비재했다. 그래서 키워드의 특성을 띄도록 명사형을 최대한 남기기 위한 두 가지 실험을 진행했다. 첫 번째는 Konlpy 형태소 분석을 문서의 내용 중 명사형만 남겨 문서를 가공하는 것이었다. 두 번째는 기사를 그냥 대입하여 키워드 추출을 진행하고, 후처리로 추출된 키워드를 오른쪽 경계에서부터 명사가 아닌 단어들을 제거하여 한국어의 형태에 맞춰서 명사 형태처럼 가공해 주는 것이었다.

자체 모델

자체 모델은 KeyBERT 와 유사하게 키워드 후보군과 뉴스 기사와의 코사인 유사도를 이용하여 Top-K 의 키워드를 추출한다. 차이점은 KeyBERT 는 키워드 후보군 단어와 그 단어의 가중치를 TF-IDF 로 이용하지만, 자체 모델은 형태소 분석기를 통하여 키워드 후보군 단어를 추출하고 해당 단어의 빈도수 Log 값을 단어별 가중치로 이용하였다. 로그를 사용하여 빈도수에 따른 영향력을 높이면서도 과하게 작용하지 않는 효과를 볼 수 있었다. 또한 KeyBERT 의 MMR 방식의 점수 계산과 다르게, 뉴스 기사 문장과 키워드 후보군의 코사인 유사도와 단어별 가중치의 곱으로 점수를 계산하여 Top-K 를 추출하였다.

$$Score = \log_{100}(n+1)\cdot\sum_{i=1}^{m} W_i \cdot \cos \quad sim_i$$

위의 수식은 최종적으로 사용한 자체 모델의 Score 수식이다. 뉴스 기사 내의 각각의 문장마다 다양한 가중치를 적용하여 뉴스 기사에서 적합한 키워드를 추출할 수 있는지, 그리고 어느 문장이 키워드 추출에 도움이 되는지를 실험하였다.

기사 감성분석

추출된 키워드를 종목에 끼치는 영향을 파악하기 쉽게 보여주기 위하여 감성 분석을 통하여 긍정 혹은 부정인지 분류한다. 방식은 키워드가 기사의 중심 내용을 담고 있다고 가정해서 기사의 긍부정 여부와 키워드의 긍부정 여부가 일치한다고 판단하여 기사 자체의 긍부정을 가리는 것을 목표로 했다.

데이터셋으로는 상술하였듯이 네이버 뉴스 기사를 크롤링한 후 챗 GPT 를 통해 라벨링을 진행한 4 개 기업의 3 천개의 기사를 사용하였고, Baseline 으로는 일반적인 BERT 계열의 RoBERTa 로 truncation 된 맨 앞 512 토큰만을 사용하는 모델을 구성하였다. 하지만 baseline 에서는 최대 1900 토큰에 달하는 기사를 입력으로 받지 못하는 문제가 있었기에 이를 해결하기 위해 다음 3 가지 방안을 시도하였다.

방법 1. 긴 입력을 받을 수 있는 모델(deBERTa, BigBird)

deBERTa 등의 모델은 512 토큰 이상의 수천, 수만 토큰을 입력으로 받을 수 있다. 길이를 길게 입력받을수록 놓치는 문장이 없기 때문에 더 높은 성능을 보여줄 것이라고 기대하였다. 32, 64, 128, 256, 512, 1024, 2048 토큰을 입력으로 사용하는 실험을 진행하였다. BigBird 모델의 경우는 초기 베이스라인 실험에서 deBERTa 모델보다 안 좋은 성능을 보여줬기 때문에 제한된 시간의 한계상 실험에서는 제외하였다.

방법 2. 기사의 전문을 나누어 학습하는 모델(RoBERTa)

BERT 계열 중, 가장 좋은 성능을 내는 RoBERTa 모델은 512 토큰 이내로만 받을 수 있기 때문에 기사 전문을 512 토큰씩 나누어 긍부정을 학습하는 방법을 시도하였다. 시도한 2 가지 방법 중 pooling 방법의 경우 기사 원문을 모두 512 토큰의 길이로 나눈 뒤 나눠진 부분마다 모두 라벨을 대응시켜 학습시키는 방식이고 BELT(https://github.com/mim-solutions/bert_for_longer_texts) 방법의 경우 나눠진 부분에서 나온 여러 예측값을 평균 내어 다시 합친 뒤 라벨을 1 개만 대응시켜 학습시키는 방식이다. 나눠진 부분들 사이에서 겹치는 토큰의 길이를 의미하는 stride 인자와 나눠진 부분이 일정 토큰 미만일 경우 학습에 사용하지 않는 min 인자를 다양하게 설정해 실험하였다.

방법 3. corpus 를 축약 후 입력하는 모델(RoBERTa)

뉴스에서 중요한 정보는 제목, 두괄식으로 작성된 첫 문장들, 미괄식으로 작성된 마지막 문장들에 담겨있다고 생각하여 해당 토큰만을 활용하는 방식으로 요약하였다. 제목의 사용 여부, 첫 문장들과 마지막 문장들의 사용 비율(1 대 3, 2 대 2, 3 대 1), 요약된 결과물의 길이(256 토큰, 512 토큰) 조절해 가며 다양한 방법을 실험하였다.

일반화 성능

지금까지는 기업 4 개의 뉴스의 데이터로 validation 을 한 결과였다. 이를 30 개의 다양한 기업에 적용시키기 위해서는 일반화 성능을 확인해야 한다. 따라서 나머지 26 개 기업으로부터 적은 양의 20 개 기사만 뽑아서 학습하지 못했던 기업과 뉴스에 대해서도 여전히 좋은 성능을 보이는지 실험하였다.

요약

요약 방식에는 크게 Extractive 방식 Abstractive 방식 두 가지가 있다. 전자의 방식은 입력 텍스트의 문장을 추출하여 요약문을 구성한다. 이러한 방식은 문장을 발췌하여 서로 붙이는 방식이기 때문에, 문맥상 사용자가 독해 시 어색함을 느낄 수 있다. 후자의 경우 모델이 전체 문맥을 읽은 후 새로 문장을 만들어 내 조금 더 매끄러운 요약문을 만들어 내는 장점이 있다. 하지만 생성형 모델이 가지고있는 문제점인 hallucination 문제가 존재한다.

해당 서비스에서는 이런 두 방식의 장단점을 고려하여 Abstractive 방식을 채택하였다. 사용자가 요약문을 직접 읽고 접하는 것이 서비스의 주 기능 중 하나기에 사용자의 가독성을 고려하여 결정하였다.

논문 요약 데이터셋과 IT / 경제 뉴스 요약 데이터셋을 이용한 T5 모델 학습 실험

본 실험에서는 두 데이터셋을 이용하여 학습한 모델이 어느 정도의 성능을 내는지 알아보기 위한 실험이다.

먼저 논문 요약 데이터셋에 대한 특성이다. 해당 데이터셋은 AIHub 에 "논문자료 요약"으로 올라온 데이터셋이며, 학술논문 18 만 건에서 전체(생성) 요약 18 만 건, 섹션별 (생성)요약 18 만 건, 특허명세서 전체 (생성)요약 17 만 건, 섹션별 (생성) 요약 17 만 건 등 총 70 만 건의 요약문으로 구성되어 있다.

IT / 경제 뉴스 요약 데이터셋의 경우 huggingface 의 "daekeun-ml/naver-news-summarization-ko" 데이터셋을 이용하였다. 본 데이터셋은 2022 년 7 월 1 일부터 동년 7 월 10 일까지 네이버 뉴스의 IT, 경제 분야에 대한 뉴스 요약본이다.

모델의 파라미터는 다음과 같은 변수로 고정한 후 실험을 진행하였다.

```
learning_rate=2e-5,
per_device_train_batch_size=8,
per_device_eval_batch_size=8,
weight_decay=0.01,
predict_with_generate=True,
fp16=True
```

T5 와 ko-GPT 를 이용한 요약 성능 비교 실험

위의 실험에서 IT / 경제 분야 뉴스로 학습된 T5 모델과 "kakaobrain/kogpt"의 프롬프트 방식을 이용한 결과 비교 실험도 진행하였다.

T5 모델의 경우 실험 1 의 결과물로 나온 모델을 이용하였다. "kakaobrain/kogpt" 모델의 경우 fine-tuning 시 OOM(Out of Memory) 문제로 인하여 모델에 기본적으로 제공되는 프롬프트를 이용한 요약을 시도해 보았다.

ko-GPT 의 경우 Zero-shot, One-shot 그리고 Few-shot 으로 실험하였다. 프롬프트는 아래와 같다.

아래의 문장을 요약해줘.

{기사}

요약:

T5 와 polyglot-ko 를 이용한 성능 비교 실험

T5 모델의 경우 실험 1 의 결과물로 나온 모델을 이용하였다. polyglot-ko 모델은 1.3b, 3.8b 그리고 5.8b 크기의 모델을 시험해 보았다. 해당 모델은 fine-tuning 시 OOM(Out of Memory) 문제로 인하여 peft 를 이용한 LoRA 방식의 fine-tuning 을 진행하였다.

데이터셋은 실험 1 에 기반하여 뉴스 요약 데이터셋을 이용하였다. 모델 크기 이외의 파라미터는 아래와 같은 동일한 변수를 이용하여 실험을 진행하였다. 먼저 LoRA 의 설정은 다음과 같다.

```
r=8
lora_alpha=16
target_modules=["query_key_value"]
lora_dropout=0.05
bias="none"
task_type="CAUSAL_LM"
```

Trainer 의 설정값은 다음과 같다.

```
per_device_train_batch_size=4
gradient_accumulation_steps=32
learning_rate=2e-4
fp16=True
logging_steps=10
save_total_limit=3
output_dir="outputs"
optim="paged_adamw_8bit"
```

모델 학습 시 사용된 프롬프트 형식은 다음과 같다.

질문: 아래의 문서를 요약해줘.

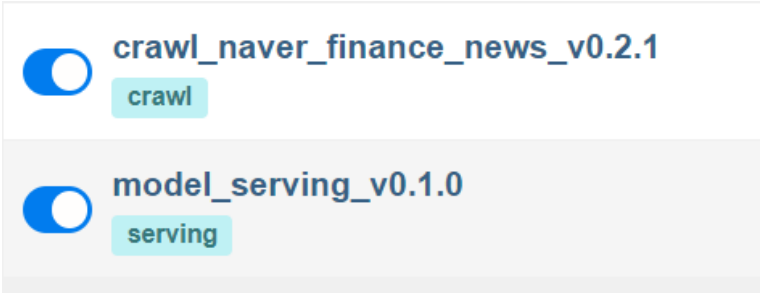
###맥락: {NEWS}

답변: {SUMMARY}<|endoftext|>

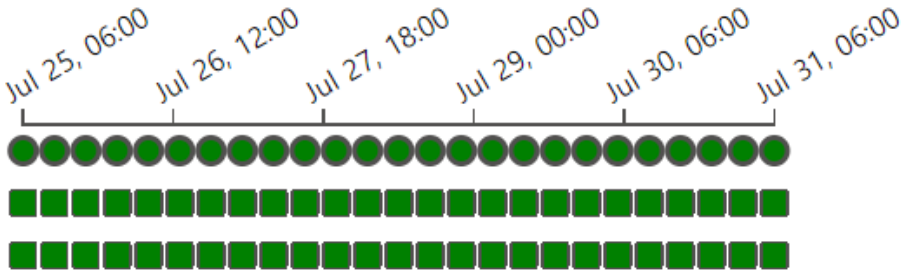
3. 프로젝트 수행 결과

전체 프로세스

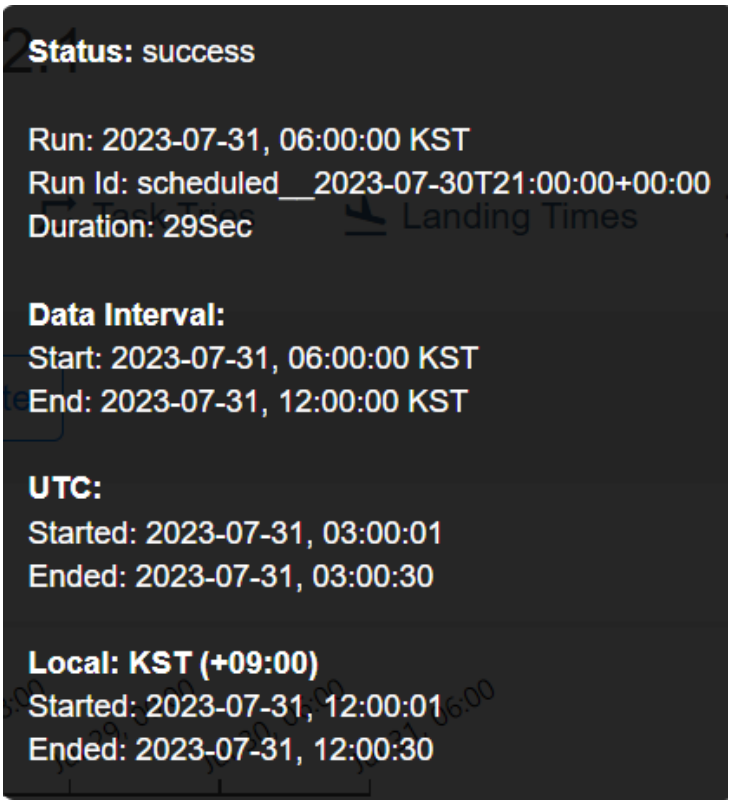
아래와 같이 Airflow 에 2 개의 DAG(Crawling, Model Serving)이 구성되어 있다.



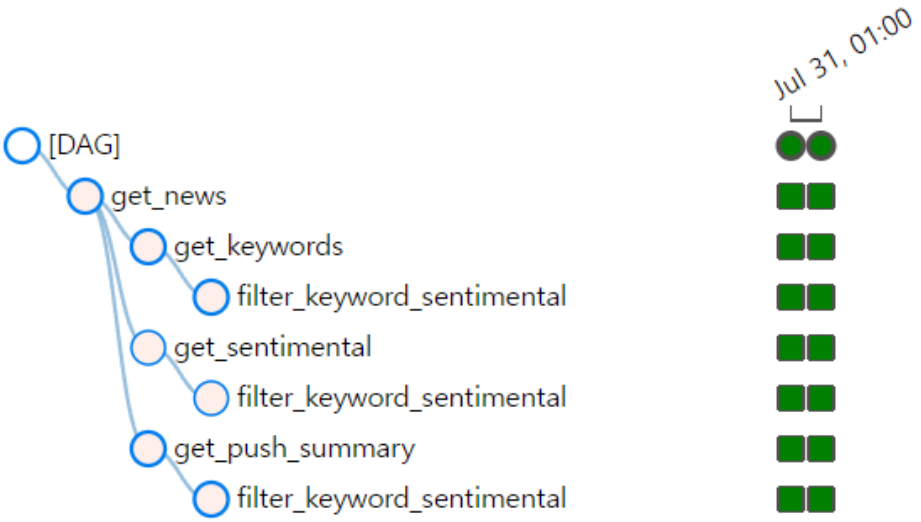
스케쥴링을 통해 주기적인 뉴스 기사 수집을 진행하고 있다.



또한 한번 Task 를 진행할 때 걸리는 시간도 1 분 미만으로 최적화를 진행하였다.

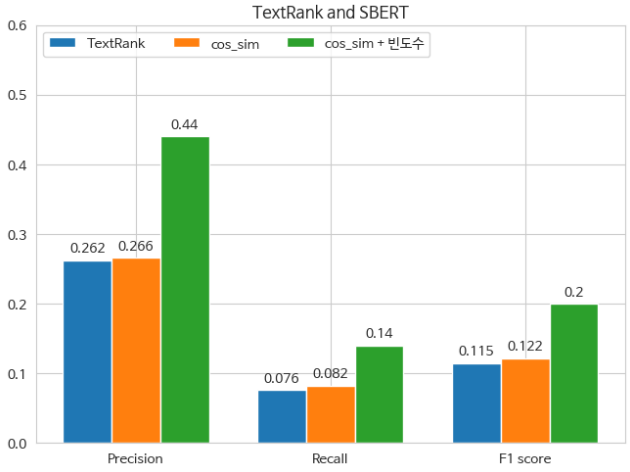


모델 서빙은 뉴스를 가져오는 작업은 공통적으로 수행되고, 나머지 3 개의 작업(기사 긍부정 분류, 키워드 추출, 요약)은 병렬적으로 수행되고, 마지막에 합쳐져서 긍부정 분류가 된 키워드를 지정하여 DB 에 삽입하도록 수행했다. 또한 해당 작업도 뉴스 기사 수집이 끝나고 한 시간 뒤에 스케쥴링을 진행해서 꾸준히 최신화된 키워드 업데이트를 진행했다.



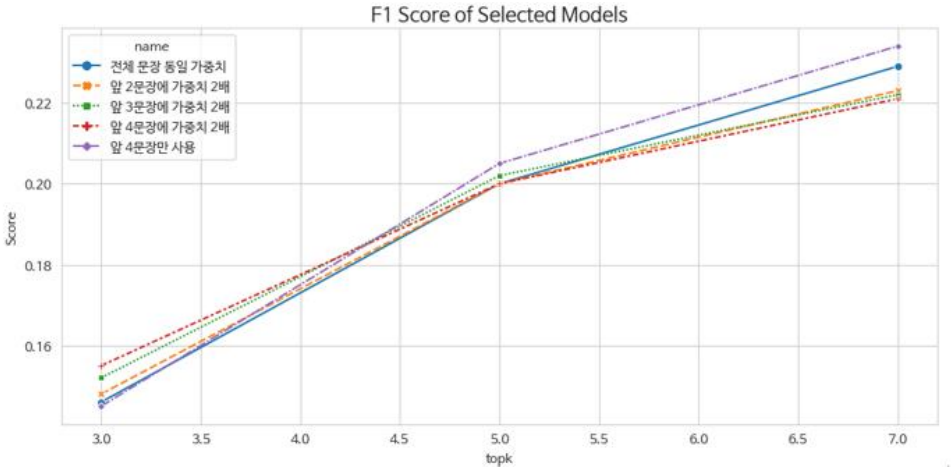
키워드 추출

키워드 추출의 성능 평가로 뉴스기사에서 자체 라벨링 한 50 개의 키워드 데이터셋을 사용하였다. 우선적으로 자체 모델과 고전적인 키워드 방법론인 TextRank 와 비교하였다. 그리고 자체 모델의 단어별 가중치를 적용한 것과 하지 않은 것을 비교하였다.



위의 이미지에서 Precision, Recall, F1 score 를 확인할 수 있다. 최종적으로 TextRank 에 비해 자체 모델의 성능이 뛰어나고 자체 모델의 단어별 가중치를 적용하였을 때 성능이 좋았던 것을 확인할 수 있다.

다음은 자체 모델의 문장별 가중치를 문장의 위치에 따라서 준 경우의 차이이다.



위의 이미지에서 문장의 가중치에 따른 F1 score 를 확인할 수 있다. 전체 문장에 동일한 가중치를 둔 경우와 비교해서 기사 앞쪽의 문장에 2 배의 가중치를 주었을 때 Top-K 3 의 성능은 올라가고 Top-K 7 의 성능은 떨어지는 경향을 확인할 수 있다. 특이한 것은 오히려 전체 문장을 사용한 경우보다 앞의 문장만을 사용하는 것이 Top-K 가 올라갈수록 높은 성능을 나타내는 것을 확인할 수 있다. 본 프로젝트에서는 뉴스 기사마다 Top-K 5 의 키워드만 사용하였기 때문에 최종적으로 연산을 줄이면서 앞 4 문장만 사용하는 것과 성능이 동일했던 뉴스 기사의 초반 128 토큰의 임베딩을 사용하였다.

위치에 따른 가중치 말고도 다른 방법론에 따른 문장별 가중치도 적용해 보았다. 첫 번째는 TextRank 의 문장 Ranking 방식을 응용한 방법이다. 두 번째 방법과 세 번째 방법은 KeyBERT 에서도 쓰이는 MSS 과 MRR 방법으로 중요 문장을 선택하는 방식을 실험하였다. 순서대로 F1 score 가 위의 방법 대비 0.01, 0.11, 0.15 가 하락하여 사용하지 않았다.

임베딩을 위한 모델들을 탐색했을 때, Hugging Face 의 한국어 NER 모델, NLI 모델, STS 모델 등 여러 모델을 비교했을 때 성능이 Sentence Transformer 기반 모델들이 더 중요한 정보는 코사인 유사도가 높고

노이즈는 낮게 나오는 경향을 확인했다. 최종적으로 Sentence Transformer 의 sroberta 모델과 SBERT 모델을 임베딩 모델로 선정하였다.

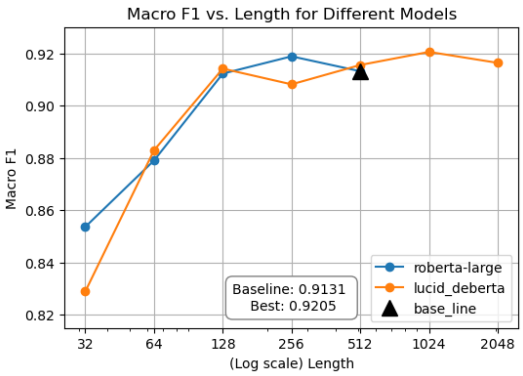
최종적으로는 TF-IDF 를 사용하는 KeyBERT 기반의 모델을 최종 모델로 선택하였는데 이는 서비스 TF-IDF 의 입력이 많아질수록 정확해지는 특성을 생각하였기 때문이다. 우리의 서비스는 주기적인 크롤링으로 한 번에 많은 뉴스 기사를 수집하여 TF-IDF 로 더욱 강점을 가질 수 있다고 판단했다.

기사 감성분석

수집된 데이터셋은 긍정 70%, 부정 30%로 라벨의 불균형을 보인다. 하지만 기사 데이터셋이 현실에서 가져온 것이기 때문에 데이터셋의 기사 긍부정 비율을 인위적으로 조정하는 것보다는 평가 방식을 잘 설정해야겠다고 판단하였다. 그래서 precision 과 recall 모두를 고려하는 F1 점수 중에서 라벨별로 단순 평균을 취하여 불균형에 큰 영향을 받지 않는 Macro F1 을 평가지표로 채택하였다. 그리고 모델실험시 사용했던 hyperparameter 들이 많았지만 grid search 를 통해 진행한 결과, 모델별 learning rate 를 제외하고 가장 영향을 많이 주었던 모델별 입력 토큰 수에 집중해서 생각해보려고 한다.

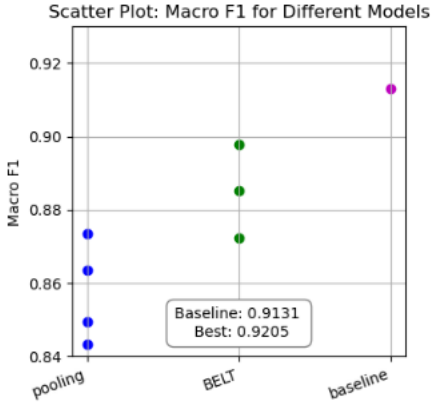
방법 1. 긴 입력을 받을 수 있는 모델(deBERTa, BigBird)

아래의 그림에 나오듯이 결과는 Macro F1 0.9205 로 다소 개선되었지만 큰 차이는 없었다. 기사의 제목을 포함시켜 학습한 결과는 오히려 성능이 약간 하락하였다. 128 토큰 이후부터는 토큰의 길이가 늘어나더라도 정확도가 크게 증가하지는 않았고, 이를 통해 전체적인 흐름을 학습하기 위해 긴 입력을 그대로 넣는 것만으로는 충분한 성능 향상이 없을 것이라고 생각하였다.



방법 2. 기사의 전문을 나누어 학습하는 모델(RoBERTa)

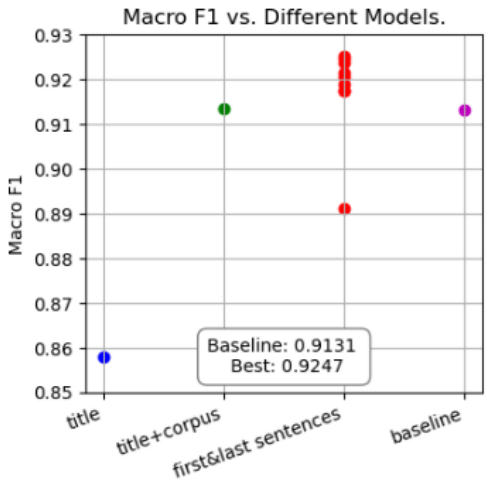
아래의 그림처럼 기사의 전문을 나누어 평균을 내는 방법과 BELT 방법 모두 Macro F1 이 baseline 보다 크게 뒤떨어졌다. 그 중 가장 좋았던 방법은 512 토큰 미만의 chunk 는 학습에 이용하지 않으면서 stride 는 0 으로 chunk 간의 겹치는 부분이 없었던 BELT 모델이었다.



방법 1 과 방법 2 는 기사 전문을 사용하면 성능이 늘어날 것이라는 가설에 시작한 방법이었다. 하지만 모두 기존 baseline 에 비해 큰 성능 향상이 없었다. 그래서 오히려 긴 입력은 노이즈로 작용한다고 생각하여 기사 내에서 핵심 내용을 축약해 긍/부정을 평가하면 성능이 좋아질 것이라는 가설을 세웠고 방법 3 을 실험하였다.

방법 3. corpus 를 축약 후 입력하는 모델(RoBERTa)

아래의 그림과 같이 성능이 크게 개선되는 것을 볼 수 있었다. 그중에서도 "제목은 사용하지 않고 기사의 첫 128 토큰, 마지막 384 토큰을 사용"하는 것이 성능이 가장 좋게 나왔다. Macro F1 0.9247 을 기록하였으며 최종적으로 이 방식을 배포에 사용하기로 선택하였다. 구현 방식은 논문 (https://arxiv.org/abs/1905.05583)을 참고하였으며 논문과 동일한 결과가 나타났다.



일반화 성능

이전에 학습 시켰던 데이터셋은 삼성전자, 네이버, 현대차, 하나금융지주 이렇게 4 개에 해당하는 데이터셋이다. 그래서 4 개 기업을 제외한 다른 기업에 대한 뉴스에 대해 추가적인 훈련을 진행하지 않을 경우, 30 개에 해당하는 데이터셋을 test set 으로 할 경우 아래의 표와 같이 0.9247 에서 0.8678 로 떨어지는 모습을 보였다. 하지만 26 개의 기업에 해당하는 260 개의 뉴스를 추가적으로 학습시키면 0.8923 으로 상당히 개선되는 모습을 나타냈다. 따라서 4 개 기업에 대한 뉴스 3000 개와 26 개 기업에 대한 520 개의 데이터를 전부 학습시킨 모델은 더욱 개선된 성능을 보여줄 것이라고 생각하였고, 해당 모델을 최종적으로 배포에 사용하기로 선택하였다.

훈련 데이터 개수	3800 개	3800 개+260 개
Macro F1	0.8678	0.8923

요약

고정된 모델, 고정된 파라미터를 이용하여 데이터셋을 실험해본 결과 Validation dataset 을 뉴스로 주었을 때 뉴스의 Rouge 1 socre 가 약 8%p 높게 나온것을 보여주었다. 이는 논문 데이터셋의 크기가 약 2 배 많음에도 불구하고 더 적은 데이터셋을 가진 뉴스 데이터가 좋은 성능을 보여주었다. 이러한 이유로

인하여 모든 모델은 뉴스 데이터셋을 이용하여 학습을 진행하였다.

모델 선정에 앞서 먼저 기준은 다음과 같다.

1. 뉴스 요약 과정에서 거짓된 사실이 없어야 한다.
2. 요약하는데 있어서 동작시간이 짧아야(15 초 미만)한다.
3. Rouge1 score 가 높은 모델을 우선적으로 택한다.

각 모델에 대한 총평은 다음과 같다. 서술한 모델의 순서는 가장 결과가 좋은 모델 순으로 정렬하였다.

T5

해당 모델을 뉴스 기사를 이용하여 학습 시켰을 때 다음과 같은 결과가 나왔다.

STEPS	TRAINING LOSS	VALIDATION LOSS	ROUGE1	ROUGE2	ROUGEL	ROUGELSUM	GEN LEN
124000	0.204800	0.208545	0.105900	0.028400	0.105800	0.106200	19.000000
126000	0.195700	0.209584	0.106300	0.028400	0.106000	0.106900	19.000000
128000	0.190200	0.206511	0.107700	0.028400	0.107400	0.107900	19.000000

초기 기본 파라미터를 이용하여 생성하였을 때 불필요한 문장 생성 혹은 비교적 적은 확률로 hallutinatio 이 발생하였다. 이러한 이유로 인해 해당 모델에 대해 생성시 파라미터를 조정하여 생성하였을 때 조금 더 나은 결과를 보여주었다.

모델의 특징은 다음과 같다. 먼저 polyglot-ko 와 비교하였을 때 hallutination 이 상대적으로 아주 낮은 확률로 발현하였다. 대부분의 기사에서 거짓된 내용 혹은 수치(금액, 길이와 같은) 변동이 없었다. 두 번째로 기사를 한 문장으로 요약하려하는 특성이 있다. 해당 문제는 데이터셋 특성으로 인하여 발생한 문제로 추청된다. 마지막으로 비교적 짧은 문장을 생성해주었다. 비교적 짧은 추론 시간이였지만 아쉬운 점으로 남는다.

polyglot-ko

해당 모델을 이용하여 1.3b, 3.8b 그리고 5.8b 크기의 모델을 시험해보았다. 여기서 5.8b 보다 1.3b, 3.8b 의 결과가 조금 더 나았으며, 결정적으로 추론시 걸리는 시간을 줄이기 위해 1.3b 크기의 모델을 비교군으로 선정하였다.

모델의 특징은 다음과 같다. 먼저 모델을 이용하여 요약시 상당히 높은 확률로 hallutination 이 발생하였다. 예를 들어 노사간의 갈등에 대한 기사에서 회의를 진행한 내용이 없었으나, 모델이 해당 사실에 대해 지어내 적는 등 기사에 거짓된 내용을 첨부하였다. 두 번째로 특정 프롬프트를 주었을 때 요약 후 요약의 추가 설명으로 추정되는 문장을 작성해주었다. 이 프롬프트는 학습 때 사용된 프롬프트가 아닌 다른 프롬프트를 주었을 때 발생하였고, 추가 설명은 "###"으로 시작하여 설명을 이어 나갔다. 마지막으로 T5 에 비해 상당히 긴 추론시간이 필요했다.

ko-GPT

해당 모델은 따로 학습 과정없이 프롬프트를 이용하여 요약을 시행해보았다.

모델의 특징은 다음과 같다. 먼저 해당 모델은 Zero-shot 에서 Few-shot 으로 갈수록 문장의 길이가 짧아졌었다. 또한 Few-shot 부터는 기사에 해당 내용이 있으나, 중요하지 않은 내용을 첨부하는 등 요약의 질이 저하됐었다. 두 번째로 요약시 hallutination 이 발생하였다. 예를 들어 차량 출시에 대한 기사에서 차량의 토크값이 기사 내에 존재하지 않음에도, 모델이 사실을 지어내어 작성하였다. 마지막으로 두 모델과 비교하였을 때 상당히 긴 추론시간이 필요했다.

*** 추가 작업**

T5 모델 특성 중 한 문장으로 요약하는 기능이 뛰어나고, polyglot-ko 의 경우 여러 문장을 길게 만드는 기능이 뛰어나 이를 조합한 방식도 구현해 보았다. 상단에 T5 모델을 이용하여 헤드라인을 제공한 후 자세한 내용을 polyglot-ko 로 요약하게 만들어, 사용자에게 실제 기사를 제공하는것과 같은 효과를 내도록 구현해 보았다. 하지만, 해당 방식은 두 모델을 직렬로 작동하게 하여 상당히 긴 추론시간이 필요하여 사용하지 않았다.

종합

위의 사항을 고려하여 다음과 같은 결정을 내렸다.

1. 다른 모델에 비해 ko-GPT 은 중요하지 않은 사실 추가와 hallutination 문제로 인하여 선택지에서 배제하였다.
2. polyglot-ko 는 T5 와 비교하였을 때 생성된 문장의 길이는 더 많이 생성하여, 독자에게 더 나은 가독성을 제공하였지만, hallutination 문제가 완전히 제거되지 않아 사용하지 않았다.
    - 2.1 생성시 사용되는 파라미터 튜닝을 이용하여 hallutination 문제가 어느정도 해결은 되었으나, 완전히 해결하지 못하였다.
    - 2.2 서비스에 있어서 기사의 사실성이 매우 중요하기 때문에 가독성보다 사실성에 더 중점을 두었다.
3. 파라미터 튜닝을 통해 T5 의 문장 생성 능력을 끌어올려 문장 완결성, 필요 내용 적시 등이 보완되었다.

이러한 점을 고려하여, T5 를 이용한 모델을 최종 선택하였다.

4. 자체 평가 의견

● **잘한 점들**

- LLM 을 이용하여 라벨링을 시도하여 시간을 절약하고, 유의미한 성능을 도출하였다.
- 생각했던 것 이상의 최종 서비스 퀄리티를 만들어낼 수 있어서 만족스럽다.
- 요약 모델 부분에서, 겉으로만 알던 프롬프트 엔지니어링에 대해 실제로 작업해보는 경험이 있어서 좋았다.
- Airflow 를 이용한 데이터 최신화와 키워드 업데이트를 통해 서비스의 완성도를 높일 수 있었다.
- 키워드 추출과정에서 다른 사람들이 사용했던 일반적인 방법이 아닌, 후처리 과정을 도입하여 추출된 키워드를 확인해봤을 때 정성적인 성능이 올라갔다고 생각한다.

- 기사 감성 분석 과정에 참고할만한 데이터셋은 영어로 구성된 짧은 문장이 있었고 난이도가 비교적 쉬운 문제(Baseline F1 0.88, SOTA 0.97)였다. 이를 감안하면 우리의 어려운 문제인 한국어로 구성된 긴 문장의 감성 분류가 만족할만한 결과(F1 0.9247)를 냈다고 생각한다.

● **아쉬웠던 점들과 개선사항**

- 시간 관리에 아쉬움이 있었다. 프로젝트 초기에 주제를 가다듬고 데이터를 수집하는 데에 계획 이상의 시간이 소요되었다. 주제가 다소 바뀌더라도 유지되는 공통 사항과 데이터가 없더라도 할 수 있는 일을 병행해서 한다면 개선할 수 있다.

- 요약 모델에서 사용하지 못한 논문 데이터셋이 대해 커리큘럼 러닝을 적용해보지 못한 것에 대해 아쉬움이 남는다.

- 요약 모델에서 긴 기사도 요약이 가능하도록 BigBird 와 같은 긴 입력의 처리가 가능한 모델에 대해 실험해보지 못하였다.

- 키워드 추출과정에서 다양한 Score 를 활용해보지 못한 것이 아쉽다.

- 키워드 추출과정에서 이용할 수 있는 한국어 데이터셋이 있다면, Automatic 키워드 추출도 시도해보고싶다.

- 기사 감성 분석 모델에서 입력의 길이 외에 다른 인자에 대해서는 시간상 변화를 주지 못했던 점이 아쉽다.

- 30 개의 기업을 목표로 했지만 그 외에 다른 기업들에 대해서는 어떻게 작동할 지 미지수이고, 전체적으로 Chat GPT 라벨링 검수과정이 한번은 있어야 했는데 없어서 아쉬웠다.

- 서비스에 사용자 피드백을 넣지 못하여 아쉬웠다.

● **프로젝트를 통해 배운 점 또는 시사점**

- 데이터셋 선정 시 동일한 task 를 해결하기 위한 데이터셋이라도, 처리할 도메인에 적합한 데이터셋을 선정하는것이 중요함을 깨달았다.

- 모델 학습시 나오는 loss 와 score 도 중요한 수치지만, 생성형 모델에 있어서 실제 결과를 받아보는 사람의 정성적 평가도 함께 동반되어야 함을 느꼈다. 동시에 정성적 평가를 설정하는 것이 얼마나 어려운지도 느꼈다.

5. 개인 회고

곽민석

1. 이번 프로젝트에서 얻은 것

- 이번 프로젝트를 통해 처음으로 생성형 모델을 다루어보고, 생성형 모델을 위한 데이터셋을 구성해보았다.
- 데이터셋 선정에서 Task 의 특성을 파악한 후 데이터셋을 선정하는것이 매우 중요함을 깨달았다.
- 표면적으로 알고있던 프롬프트 엔지니어링에 대해 학습한 모델에 실제 적용해보며, 해당 작업이 무엇을 위해 하는지, 어떠한 어려움이 있는지 알게 되었다.
- 모델의 크기와 실제 스코어는 상관관계가 있을지라도, 둘과 사용자의 만족도는 완전히 일치하지 않을 수 있음을 알게 되었다.
    - 기사 내용을 사용자가 실제로 읽을 것을 생각하여 모델이 만들어낸 기사를 정성적 평가도 고려하였었는데, 이때 들었던 생각이다.
    - 이전 마스터님이 말씀하신, 모델의 loss 와 score 뿐만 아니라 실제 결과물 또한 유심히 봐야한다는 말씀이 생각나는 대목이다.
- 긍부정 데이터셋 라벨링에서 LLM 을 이용한 데이터 라벨링을 처음 시도해보고 좋은 결과를 받아보았다.
    - 이전부터 생각만 해왔던 부분이였는데, 이번 결과를 통해 이것이 가능함을 확인하였다.
    - 해당 방식을 이용하여 다른 문제(Cold start 등)에 적용 가능할것으로 생각한다.
- 이제 복잡한 Git 사용이 가능하다.
    - 몇번의 프로젝트와 이번 최종 프로젝트를 통하여 이제 어느정도의 github 의 기능은 사용할수 있는것 같다.

2. 이번 프로젝트에서 아쉬웠던 것

- 데이터셋 선정 시 뉴스 요약에. 있어서 낮은 성능을 내어 배제하였던 논문 데이터셋에 대해 활용방안을 찾아볼 수 있지 않았을까에 대한 아쉬움이 남는다.
- 데이터셋 선정 시 뉴스 요약에 있어서 낮은 성능을 내어 배제하였던 논문 데이터셋에 대해 활용방안을 찾아볼수 있지 않았을까에 대한 아쉬움이 남는다.
    - 해당 데이터셋을 이용한 커리큘럼 러닝을 도입하려 했으나 다른 작업(프론트엔드)로 인하여 실험만 해보고 실제 적용은 해보지 못하였다.
    - 위 실험에서 약 3%p 의 성능 향상을 확인하여, 발전 가능성이 있는것으로 보인다.
- 서빙 시 최초 계획은 BentoML 을 이용한 서빙을 생각했으나, 시간상의 문제로 FastAPI 를 이용하여 작업하였다.
    - 프론트엔드 작업을 주로 하게되면서 모델 서빙 부분을 약간 소홀한 경향이 있었는데, 실제 엔터프라이즈에서 어떤 방식으로 서빙하게 되는지 알아보고싶었다.

- LLaMA 2 모델이 작업 중 나오게 되어 적용해보지 못하였다. 물론 한국어 학습이 잘 되어있지 않아

시도하지 않은 것도 있지만, 조금 더 일찍 나왔었다면 polyglot 과 같이 실험을 해보지 않았을까 한다.

3. 총평

이전부터 관심이 있었던 요약 모델에 대해 어떤식으로 작동하고 학습시켜야 하는지 알아보아 좋은 경험이였다. 프로젝트 초반에는 어떤 방향으로 진행해야 하는지 감이 잘 오지 않았었는데, 관심분야를 맡게되어 조금 더 빨리 알아보게 된것같다.

위의 1 번에서 언급한 얻은 것 외에 추가로 한가지가 더 있다. 바로 오랜만에 애정이 가는 내가만든 프로젝트이다. 터미널에서 각자의 화면으로 모델의 결과를 보았을 때는 어쩌면 볼품없어 보였었지만, 실제 서비스화를 위해 하나로 모였을 때 진정한 모습을 보여준것 같다. 아마 처음으로 모델을 만들어 서비스화 시킴으로 인해 애정이 가는것 같다.

이인균

1. 이번 프로젝트에서 얻은 것

- 최종적으로는 사용하지 않았지만, 프로젝트 초안에서는 직접 데이터셋을 수집하였고 직접 라벨링도 해보았다. 라벨링 과정에서 일관성을 유지하는 일의 어려움과 라벨링에 드는 시간을 절약하는 것에 어려움을 깨달았다.
- 감성 분석에서는 길이가 긴 지문을 다루는 일을 중점적으로 하였는데 작업 레퍼런스를 찾는 것에 어려움을 겪었었다. deBERTa, BigBird, BELT, arxiv.org/abs/1905.05583 을 참고할 수 있었지만 더 나은 참고가 있을 것이라고 생각하고 앞으로 꾸준히 업계의 연구를 따라가는 것이 중요하다고 생각한다. 이 과정을 통해서 얻었던 긍정적인 요소는 찾았던 레퍼런스를 바탕으로 수십 개의 실험을 시행하였고 매우 긴 지문을 다루는 방법들에 대해서 익힐 수 있었던 점이다.
- 문제 정의의 중요성을 깨달았다. 긍부정 분류는 시각적으로 매우 중요하다고 생각하였으나 피드백 과정에서는 긍부정 분류 모델의 존재 의의를 잘 공감을 하지 못했다는 피드백이 많았다. 자세한 설명이나 그림을 제시했으면 조금 더 전달이 쉬웠을 듯하다.
- 이번 프로젝트에서는 기사의 긍부정을 키워드의 긍부정으로 사용했는데 이 점에서 질문이 많이 들어왔다. 프로젝트 과정에서 논리적인 연결성을 앞으로는 조금 더 신경써야 할 것 같다. 이것 역시 실제로 좋았던 예시를 보여줬으면 전달이 쉬웠을 듯하다.

2. 이번 프로젝트에서 아쉬웠던 것

- 팀원과 협업 과정에서 명확한 설명과 명확한 문서화가 중요하다는 것을 다시금 깨달았고 이 점에 대해 어려움을 겪었다. 이번 최종 프로젝트에서는 대면 작업이 길었는데 오히려 아침에 하던 작업물 공유가 안 됐었던 것 같다. 조금 더 빨리 이 점을 개선하자고 말했으면 좋았을 듯하다.
- 작업을 병렬적으로 하는 것에 어려움이 있었다. 선행 작업이 필요한 작업이 있고 아닌 작업이 있는데 이 둘을 구별하지 못하고 선행 작업을 기다리느라 작업 속도가 늦추어졌다. 예를 들어 데이터 수집과 라벨링 과정 동안 조금 더 페이퍼 리서치에 신경을 썼으면 좋았을 듯하다.
- 모델 3 개를 활용하였고 이전보다 브랜치를 조금 더 잘게 쪼개려고 계획하다 보니 깃허브 사용에 어려움이 있었다. 프로젝트 중간에 분리된 브랜치가 지나치게 많다는 지적을 멘토님으로부터 받은 뒤로는 조금 더 깔끔하고 이해하기 쉽도록 사용할 수 있었다.

3. 총평

LLM 과 PLM 의 장점은 기학습된 모델을 이용하여 약간의 파인튜닝만 적용하면 기업이 가지고 있는 데이터를 활용할 수 있다는 점이다. 이때 기업이 갖고 있는 데이터가 512 토큰 이상의 경우 BERT 모델을 곧이 곧대로 사용할 수는 없기 때문에 장점이 퇴색되어버린다. 이 점을 문제로 정의하고 해결하는 데에 집중하였다. 그동안에는 큰 문제 정의가 기본적으로 주어지기 때문에 작은 문제만 직접 정의하였는데 이번에는 큰 문제도 직접 정의 후 해결하는 과정을 경험할 수 있어서 좋았다.

SQL 문을 활용해 view 를 1 개 생성했던 것 빼고는 프론트엔드와 백엔드 과정에서 도움이 되지 못했다.

앞으로도 프론트엔드와 백엔드를 깊게 학습할 계획은 없지만 대화 내용을 이해하는 정도로는 학습하면 좋을 것이라는 생각이 들었다.

임하림

수행한 활동

- 기사 데이터셋 구성을 위한 전처리

- chat gpt 를 통한 긍부정 라벨링

- 기사 감성분석 모델 개발 및 API 설계

활동 세부 내용

- 3000 개에 해당하는 기사에 대해 전처리를 진행했다. 기사에서 감성 분석에 방해가 되는 부분, 예를 들어 기자 이름, 언론사, 이메일, 사진 정보 등등을 비롯한 10 개에 대해서 각각의 특징에 따라 분류하여 전처리를 진행 했다.
- 이전에는 데이터셋이 주어진채로 모델의 성능만을 올리는 것을 진행했지만, 이번에는 모델링에 쓸 feature 를 먼저 정하는 과정을 통해 데이터셋을 구성했고, 필요에 따라 feature 를 추가하며 모델의 성능을 더 유동적으로 올릴 수 있었다.
- 기존에는 데이터셋을 구성함에 있어서 human labeling 이 압도적이었지만, chat gpt 를 통해 라벨링을 진행해본 결과 데이터셋의 성능이 꽤 좋다는 것을 알게 되었다. 그리고 chat gpt 에 들어가는 프롬프트에서도 명확한 feature 를 잡아 진행했다.
- 기사 감성분석 모델에서 입력의 수를 늘리는 것보다 줄이는 것에 주목하여 입력을 축약하는 모델을 구성했다.
- FAST API 를 통해 각각의 입력과 출력을 설계해서 감성분석 모델을 직접 사용할 수 있는 API 를 구성했다.

아쉬웠던 점

- Llama 2 가 프로젝트 진행 도중에 공개되어서 요약 테스크를 잘 수행하는 지 프롬프트를 통해 확인해보는 과정중, 한국말에 대해서는 성능이 좋지 않은 것을 알았다. 시간 문제 때문에 시도를 못 했지만 PEFT 를 통해 시도라도 해봤으면 어땠을까 라는 아쉬움이 있다. 그래서 프로젝트가 끝났지만 GPU 는 남아있어서 한번 시도해보려고 한다.
- 이전에는 데이터셋이 고정되어 있어서 feature 를 추가하는 방법론에 대해서도 접근하지 못 했고, chat gpt 를 통해서 진행할때 feature 를 명확하게 잡아서 진행했다면 다른 시도들을 더 할 수 있었을까 라는 아쉬움이 있다.
- chat gpt 를 통한 데이터셋의 검수가 한번은 제대로 진행 했어야 했는데 아쉽다.

잘 했던 점

- 감성분석 모델에서 입력을 줄이는 것과 모델의 성능에 대한 타당한 근거를 가지고 근거와 가설을 통해 성능을 올렸다.
- 이전에는 못 했던 여러 개의 Task 가 병렬적으로 이루어지는 프로젝트를 통해, 협업할때 github 에서

기본적인 사항은 쓸 수 있다.

- 이전에는 백엔드와는 거리가 멀었지만 FAST API 를 통해 API 구성을 해봐서 좋았고, 추가적으로 더 배워보고 싶다는 생각이 들었다.

최휘민

나의 활동

- 키워드 추출 평가 데이터 제작
- 키워드 추출 자체 모델 설계 및 실험
- 키워드 추출 API 설계

목표를 위한 세부내용

- 한국어 데이터에 관한 키워드 추출 모델과 데이터셋이 존재하지 않았기에 기존의 키워드 추출 방법인 TextRank 와 KeyBERT 를 참고하여 학습이 필요하지 않는 비지도 방법의 모델들이 적합하다고 생각하였다. 하지만 실험의 성능 평가 지표는 필요하다고 생각하여 평가 데이터셋의 직접 제작하였다.
- 뉴스 기사마다 중요 키워드의 기준이 모호했기에 LLM 을 이용하여 키워드 후보를 반복하여 추출하고 피드백하는 방식으로 키워드를 선별하여 총 50 개의 평가 데이터셋을 제작하였습니다.
- 자체 모델은 KeyBERT 처럼 문장 임베딩과 코사인 유사도를 활용하여 키워드를 추출하는 것을 생각하였고 가장큰 차이점은 KeyBERT 는 TF-IDF 로 여러 뉴스 기사를 참조하나 자체 설계한 모델은 단일 뉴스 만으로 키워드를 추출한다. TF-IDF 의 단점인 vocabulary size 에 따른 문제에서 벗어나기 위함이었다.
- 단어별 가중치를 적용하기 위해 빈도수에 Log 값을 사용하였다. Log 를 적용하여 기사에 등장하는 단어 빈도수의 영향력을 반영하면서 과하지 않게 작동하도록 하였다.
- 뉴스 기사의 문장별로 가중치를 주는 실험을 했다. 뉴스 기사는 중요한 정보를 제목과 앞의 문장에 일반적으로 사용한다고 생각하였기 때문이다. 실험결과 Top-K 5 와 7 에서는 전체 문장보다 앞의 문장 4 개만을 사용한 경우가 성능이 더 좋았다.
- 단순히 위치에 따른 가중치 보다 문장의 중요도에 따라서 가중치를 적용하는 실험도 진행하였다. TextRank 방법을 응용하여 문장 임베딩의 코사인 유사도로 문장의 가중치를 적용한 방법과 KeyBERT 의 MSS, MMR 을 사용하였지만 성능은 떨어졌다.
- FastAPI 로 모델의 입력과 결과를 API 로 이용 가능하게 설계하였다.

자체 평가 의견

- 가장 좋았던 점은 데이터가 없고 정해진 모델이 없는 상태로 최종적으로 모델을 만들고 결과가 잘 나오며 가설에 따른 평가가 가능하도록 지표가 어느정도 잘 완성되었다는 점이다. 앞으로 데이터가 하나도 주어지지 않은 새로운 Task 를 시도해도 자신감을 얻고 도전해 볼 수 있을 것이다.
- 결과 기록을 잘 해두어서 좋았다고 생각한다. 특히나 이번의 여러 시도들은 어떠한 결과가 나오고 이것으로 어떤 결과를 내릴 수 있을지 막막했었다. 예를 들어 문장 위치에 따른 가중치를 실험하면서 가중치를 주지 않은 경우와 비교를 원했지만 눈에 보이는 결론이 딱히 보이지 않았다. 하지만 조건을 변경해 가면서 실험을 하여 가중치를 주는 문장의 수에 따라 Top-K 에 따른 성능 차이와 변화율이 달라진다는 결론을 확인할 수 있었다.
- FastAPI 로 모델을 서버로 작동하게 만들어 서비스에 사용할 수 있게 설계한 것이 좋았다. 서비스 론칭 경험이 없는 나에게는 값진 경험이였다.

아쉬운 점

- 이번에는 최신 방법론에 의거한 구현을 시도한 것이 아쉽다. 또한 처음의 자체모델이 끝나고 Few Shot 을 통한 학습 데이터 생성과 지도 학습 모델을 시도해 보고 싶었지만 일정 문제도 시도하지 못한 것이 아쉽다. 그래도 자체 모델의 성능이 잘 나와서 서비스 차원에서 심각한 개선이 필요하지는 않은 수준이였기 때문에 학습 차원에서는 아쉽지만 프로젝트 서비스 차원에서는 만족하였다.
- 이번에는 백엔드와 프론트엔드에 기여를 하지 못한것이 아쉽다. 이번 API 설계를 경험으로 이후에는 전체적인 프로세스를 다 경험해 보고 싶다.

황윤기

수행한 것

- 키워드 추출 모델 실험(KR-WordRank, KeyBERT) 및 후처리
- FastAPI 를 이용한 키워드 추출 모델 서빙
- MMR Score 와 MaxSum Score 실험
- AirFlow 를 이용한 스케쥴링 구성

수행하면서 깨달은 것

- Airflow 의 DAG 을 설계하면서 시간 정보를 이용할 때, Execution Time 이 UTC 시간 기준으로 적용되기에, 시간에 대한 문제를 해결하는데 많은 시간이 들었다. Test DAG 을 구성하여 여러번 실험하면서, KST 기준으로 정확한 시간에 맞춰서 작동하도록 macro 를 구성하여, exectuion time 을 KST 기준으로 변경하여 넘겨주도록 구성했다.
- Airflow 를 통해 전체적인 서비스 설계를 위해, 넘겨받는 데이터의 형태도 중요했는데, 우리의 서비스는 각각의 모듈들이 API 로 서빙하도록 구성하고, 전체 서비스 서빙을 위해서는 각각의 API 를 호출하여 가공한 뒤 연쇄하여 처리하는 방식으로 설계해두었다. 그러면서 서로의 데이터 넘겨주는 방식이 달라서 문제를 일으키기도 하고, 프론트엔드(JS), 백엔드(Python)간의 같은 데이터 형태더라도 넘겨받을 때 생기는 문제가 발생하면서 오류를 수정하는데 큰 시간을 들였다.

프로젝트를 수행하면서 아쉬웠던 점

- KeyBERT 기반 모델을 이용할 때, 임베딩 된 모델의 점수를 계산할 때 여러가지 계산식을 실험해봤으면 좋을 것 같다. 현재 프로젝트에 적용된 점수는 MMR Score 인데, 점수를 계산하는 방식이 뽑힌 키워드와의 연관성을 고려하도록 수식이 작성되어 있기에, 우리만의 Score 를 더 만들어서 뽑히는 키워드의 특성을 바꿀 수 있지 않을까라는 실험 구상을 했지만, 전체 서비스 수행을 위해선 해당 작업을 완료할 시간이 부족했다. 이것은 프로젝트가 다 마무리된 이후에 더 실험을 진행해보려고한다.
- 단어별 가중치를 계산할 때 TF-IDF Vectorizer 를 사용했는데, 이 부분에 KR-WordRank 를 이용해서 계산된 가중치를 사용해보지 못한 것도 아쉬웠다. 기본적인 TF-IDF Vectorizer 는 적게 나온 단어들과 존재하는 문서들의 갯수를 이용해서 계산하는데, KR-WordRank 를 이용한다면 더 정확한 가중치를 구할 수 있었을 것으로 예상된다.
- 데이터의 부족으로 지도 학습 기반 Automatic 키워드 추출을 수행하지 못한 것도 아쉬웠다. NER 과 같은 방식으로 키워드 추출을 진행하는 것도 고려했으나, 한국어 기반 키워드 추출 데이터셋이 존재하지 않고, 우리가 데이터를 만들어보려고 했지만, 시간부족으로 인해 학습데이터로 사용할 정도로 많은 갯수를 만들진 못하여 평가하는 데이터셋의 형태로만 사용하고, 학습에는 사용하지 못했다.

잘한 점

- 키워드 추출과정에서 KeyBERT 를 사용하는 것은 많이 보였으나, 해당하는 방법을 변형하여 사용하는

대부분의 방법은 문서내에서 명사형만 추출하여 붙이는 방법으로 문서를 재구성해서 작업을 진행했다. 하지만 실험해본 결과 해당 방법론은 키워드라고 생각될만한 내용들이 추출되지 않았으며, 문서의 주제를 잘 이해하지 못한다고 생각이 들었다. 그래서 도입한 방법이 기본적으로 문서를 전처리하여 넣어주고, 추출된 키워드를 한국어 형태로 가공한 것이었다. 새롭게 도입한 후처리 방식으로 앞서 언급한 방법 대비 문서에 대한 주요한 단어들이 추출되는 것으로 보여졌고, 다른 팀원들도 그렇게 인식된다고 말씀하여 최종적으로 해당 방법론을 도입하였다.

- Sentence-Transformer 를 이용할 시에 모델 1 개의 결과만을 가지고 임베딩을 진행하면, 각 모델의 특성에 맞게 키워드들이 편향되는 문제점이 관찰됐다. 그렇기에 여러 모델들에서 키워드를 추출하여, 해당 키워드의 점수를 합친 뒤에 가장 높은 점수를 가진 키워드들을 추출하는 방식으로 진행했을 때가 더욱 좋은 결과물을 내서, 해당 방법론을 최종 서비스에 반영하여 키워드를 추출하였다.