

일기 감정 기반 영화 추천 시스템

-Diary Emotion-based Movie Recommendation System-

2022.12.19.

시스템생명공학과 201814246 김예지
컴퓨터공학부 201512265 박인우
응용통계학과 201711649 손소영
수학과 201710304 정창해

지도교수 김 학 수 교수님

Summary

This project is an iOS application that recommends movies based on the emotion of the user's diary. It is for people like us, living in a rapidly changing world, to take care and reflect their emotions. Being able to express emotions is important to live in a mentally healthy society. The easiest way for us to express our feelings is to write a diary. Another way is to enjoy cultural content such as movies. Nowadays, when we watch a movie in our spare time on the OTT platforms, we select one of the numerous contents to watch. The emotion at the time affects the choice.

Considering this, we've planned a service where the user writes a diary and can get movie recommendations based on the emotion of the diary we analyze. Through this service, the user can reflect their emotions and get movie recommendations, which we expect to have a beneficial effect on their mental health.

First of all, the diary emotion analysis model uses ELECTRA, which is a model derived from BERT. ELECTRA consists of a generator and a discriminator. We used KcELECTRA, a pre-trained model with vocabularies from user generated content such as online comments and blogs. It's better suited for our use case than KoELECTRA, the basic Korean ELECTRA model, which is trained with vocabularies from news articles that tend to have more formal tones. Then we added LSTM layer to track the flow of emotions in the paragraph. Also, we changed the scheduler from linear LR scheduler to cosine annealing LR scheduler to solve local minimum problem and the performance improved from 0.54 to 0.62. Additionally, we trained the model with addition of three losses from CLS token and first and last SEP token. To use the loss from SEP tokens, we split the data into sentences using KSS(Korean sentence split) library, then put them in KOTE model and make the silver label. We tried to make the gold label of each sentence by editing silver labels by ourselves.

Secondly, the recommendation system model outputs probability values for each emotion label when text is input. We compute cosine similarities between the vector output of the diary and the movie plot to recommend a movie.

Finally, the application is an iOS app and can be used in the iOS environment.

목차

1. 배경 및 필요성.....	4
1.1 프로젝트 동기	4
1.2 핵심 시나리오	4
2. 요구사항 정의.....	5
2.1 기능적 요구사항	5
2.2 비기능적 요구사항	5
3. 시스템 설계.....	6
3.1 시스템 구조도	6
3.2 컴포넌트 다이어그램	7
3.3 시퀀스 다이어그램	11
3.4 핵심 알고리즘	12
4. 시스템 구현.....	15
4.1 핵심코드 상세설명	15
4.2 구현시 문제점과 해결방법	17
5. 시스템 실행 결과.....	19
6. 시스템 시험 결과.....	23
7. 팀원 및 역할 분담.....	24
8. 향후 졸업작품의 활용 방안.....	25
9. 맺음말.....	26
깃허브 활동 내역	27
시연영상 링크	28
참고자료	28

1. 배경 및 필요성

1.1 프로젝트 동기

현대인들은 복잡하고 빠르게 변화하는 사회 속에서 방대한 양의 감정을 소비하며 살아가고 있다. 소비하는 감정이 많음에도 불구하고 그들은 사회적 지위를 위해, 혹은 자신이 더 낫다고 생각하는 가치를 위해 본인의 감정을 억압하며 살아간다. 감정의 억압은 결국 정신 질환을 야기하고, 정신 질환은 더 심각한 사회적 문제를 초래할 수 있다.

감정을 표출할 줄 아는 것은 정신적으로 건강한 사회를 살아가는 데 있어서 중요한 요소이다. 우리가 가장 쉽게 감정을 표출할 수 있는 방법은 일기를 작성하는 것이다. 또한 영화와 같은 문화 콘텐츠를 시청하는 것도 하나의 방법이다.

“라이프로깅(Lifelogging)”이 트렌드가 되면서 많은 사람들이 자신의 일상에서 일어나는 일을 기록하고 데이터화하면서 라이프로그에 초점을 맞춘 다양한 서비스가 등장하고 있다. 또한 OTT 플랫폼이 활성화된 오늘날 영화 시청에 대한 시공간적 제약이 사라졌고, 영화의 선택지가 다양해지면서 퇴근 후 하루를 마무리하며, 또는 주말에 집에서 영화를 시청하는 사람이 늘어났다. 그리고 시청할 영화를 선택할 때 선택 당시의 감정이 영화 선정에 영향을 준다. 이러한 점을 고려하여 본 팀은 일기 감정 기반 영화 추천 시스템을 주제로 선정하게 되었다.

1.2 핵심 시나리오

사용자가 일기를 작성하면 일기에 내포된 그날의 감정의 분포를 보여주고, 그 감정과 유사한 감정을 가진 영화를 추천해준다. 이 서비스를 통해 사용자는 자신의 감정을 표출하고, 부정적인 감정은 해소하며 정신적으로 건강한 삶을 영위하는 데 도움을 받을 수 있을 것이라고 예상한다.

2. 요구사항 정의

2.1 기능적 요구사항

2.1.1 일기 입력

1. User가 System에 일기를 입력한다.
2. 일기 데이터를 저장하고 일기감정분석으로 넘어간다.

2.1.2 일기 감정 분석

1. 저장된 일기 데이터를 감정 분석 모델에 insert한다.
2. 일기 데이터의 일기 감정 벡터를 출력한다.
3. User에게 일기 감정 벡터를 기반으로 감정을 보여준다.

2.1.3 감정 기반 영화 추천

1. 영화 메타데이터를 이용해 영화 감정 벡터를 생성한다.
2. 일기 감정 벡터와 영화 감정 벡터를 추천시스템에 insert한다.
3. 일기 감정 벡터와 영화 감정 벡터의 유사도를 기반으로 영화를 추천한다.
4. User가 추천된 영화를 확인한다.

2.2 비기능적 요구사항

2.2.1 사용 편리성

- 사용자가 쉽게 앱을 이용할 수 있도록 직관적이고 깔끔한 사용자 경험을 제공한다.
- 사용자가 원하는 영화가 어떤 ott에 있는지 확인할 수 있어서 시청에 도움을 준다.
- 일기에 대한 감정을 차트 형태로 제공하여 사용자가 한 눈에 현재 감정 상태를 확인할 수 있도록 한다.
- 영화에 대한 기본정보(개봉연도, 출연진, 상영시간 등)을 제공한다.

2.2.2 신뢰성

- 모델 별로 감정 분석 신뢰도를 분석하여 보다 명확한 감정상태를 파악할 수 있는 모델을 선

정한다.

- 졸업 전시회 기간 동안 사용자의 만족도를 수집하여 추천시스템의 신뢰도를 파악한다.

2.2.3 성능

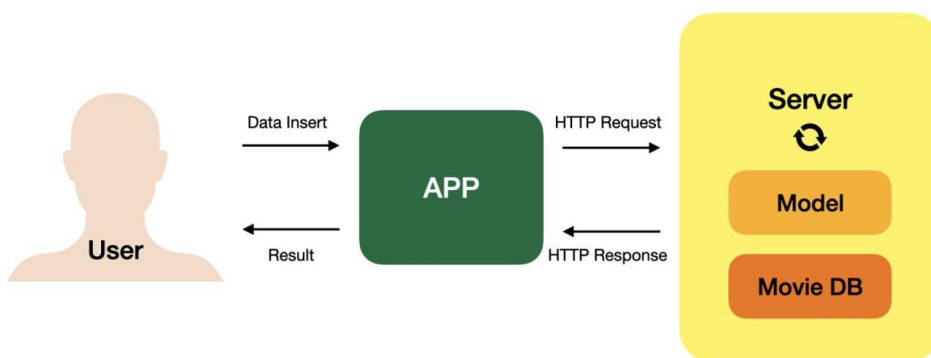
- 감정 분석 모델의 성능은 70%를 목표로 한다.
- 일기를 입력하고 서버에서 데이터를 받아올 때 3초 이내로 결과가 출력된다.

2.2.4 이식성

- iOS를 사용하는 플랫폼에서 사용할 수 있다.
- View의 Layout이 기기마다 유동적으로 변경되도록 하여 모든 iOS 기기에서 유사한 UI 배치를 확인할 수 있다.
- 웹 및 안드로이드 플랫폼에서는 사용할 수 없다.

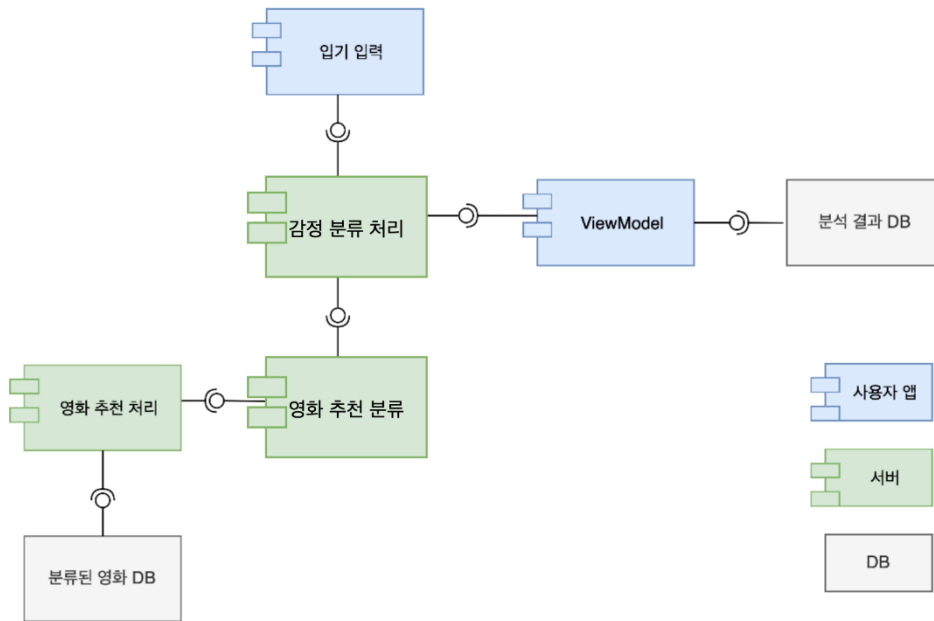
3. 시스템 설계

3.1 시스템 구조도



- 앱에서 사용자가 일기를 입력하면 서버에서 사전에 학습된 모델을 이용해 일기의 감정 벡터 반환한다.
- 이를 미리 모델에 돌려서 db에 저장한 영화 벡터와 유사도를 계산하여 감정 확률과 영화 추천 결과를 반환한다.

3.2 컴포넌트 다이어그램



컴포넌트 명	설명
View Model	분석된 결과와 이전의 결과들을 보여줄 수 있는 UI 담당한다.
일기 입력	사용자가 일기를 입력한다.
감정 분류 처리	사용자의 일기 Text를 받아서 감정 정보를 분류한다.
영화 추천 요청	영화 추천 모델에 사용자의 감정정보를 입력하며, 추천을 요청한다.
영화 추천 처리	분류된 영화 data와 유사도 검사를 통해 사용자의 감정상태와 유사한 영화를 추천한다.
분류된 영화 DB	줄거리를 기반으로 감정 분류된 영화 Data
분석 결과 DB	감정과 연결된 영화 매칭 정보 저장

Component Definition

컴포넌트명	일기입력
개요	사용자가 일기를 입력한다.
내부 클래스	
클래스명	비고
InputText	사용자가 입력한 Text 정보를 받아서 처리한다. Text 글자 수 정보 판단 감정 분류처리에 해당 Text 정보를 전달
인터페이스	
인터페이스명	비고
일기 Text 전달	사용자가 입력한 일기 Text를 감정분류 모델에 전달하여, 일기의 감정 분석을 시작한다.

컴포넌트명	감정 분류 처리
개요	사용자의 일기 Text를 받아서 감정 정보를 분류한다.
내부 클래스	
클래스명	비고
EmotionClassification	Text를 기반으로 감정 정보를 분류한다. 일기 Text는 URL Session을 기반으로 REST API 데이터 요청을 구성해 전달한다.
인터페이스	
인터페이스명	비고
감정분류 결과 저장	감정 분류 결과를 View Model에 전달하여 화면에 보여준다.
감정정보 전달	일기로 분석한 감정 정보를 기반으로 영화 추천을 요청한다.

컴포넌트명	영화 추천 요청
개요	영화 추천 모델에 사용자의 감정정보를 입력하며, 추천을 요청한다.
내부 클래스	
클래스명	비고
Fetch Recommendation	감정정보를 기반으로 서버에 영화 추천을 요청한다. 리턴 값은 JSON 형태의 영화 추천 데이터이다.
인터페이스	
인터페이스명	비고

감정정보 IO	감정 정보를 서버에 전달한다.
추천 결과 저장	추천된 영화를 View Model에 전달해 화면에 출력한다.

컴포넌트명	영화 추천 처리
개요	분류된 영화 data와 유사도 검사를 통해 사용자의 감정상태와 유사한 영화를 추천한다.
내부 클래스	
클래스명	비고
RecommendProcess	기 분류된 영화 데이터에 존재하는 감정 정보와 사용자의 감정 정보를 비교해 우선순위를 도출한다 그 중 상위 10개를 판별한다. 처리된 결과를 다시 Client에게 전달한다.
인터페이스	
인터페이스명	비고
영화 정보 I/O	분류된 영화 DB에서 영화 데이터를 가져온다.

Component Interface

인터페이스 명	일기 Text 전달
인터페이스 개요	사용자가 입력한 일기 Text를 감정분류 모델에 전달하여, 일기의 감정 분석을 시작한다.
파라미터	일기 Text
반환 값	없음

인터페이스 명	감정 정보 전달
인터페이스 개요	일기로 분석한 감정 정보를 기반으로 영화 추천을 요청한다.
파라미터	감정 정보 Data
반환 값	없음

인터페이스 명	감정 분류 결과 저장
인터페이스 개요	감정 분류 결과를 View Model에 전달하여 화면에 보여준다.
파라미터	감정 정보 Data
반환 값	없음

인터페이스 명	감정 정보 I/O
인터페이스 개요	감정 정보를 서버에 전달한다.
파라미터	감정 정보 Data
반환 값	영화 추천 Data

인터페이스 명	영화 정보 I/O
인터페이스 개요	분류된 영화 DB에서 영화 데이터를 가져온다
파라미터	없음
반환 값	없음

인터페이스 명	추천 결과 저장
인터페이스 개요	추천된 영화를 View Model에 전달해 화면에 출력한다.
파라미터	영화 추천 Data
반환 값	없음

인터페이스 명	분석 결과 저장
인터페이스 개요	출력된 정보와 사용자의 영화 평가를 분석 결과 DB에 저장한다.
파라미터	감정 분석 Data, 영화 추천 Data, 사용자 평가
반환 값	없음

Component Algorithm

1. 감성분석 알고리즘

- 훈련 완료된 모델을 불러온다
- 모델의 input으로 일기 데이터를 넣고 감정 벡터를 output으로 반환한다

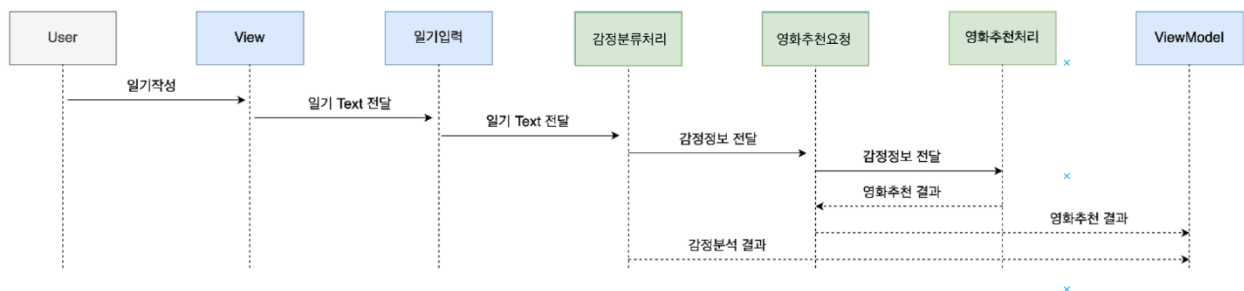
2. 추천 알고리즘

- 감정 분석 모델이 반환한 감정 벡터를 가져온다
- 사전에 저장해 놓은 영화 감정 벡터를 불러온다
- 벡터 유사도를 계산해 추천된 영화를 반환한다

3. 감성분석 모델 훈련 알고리즘

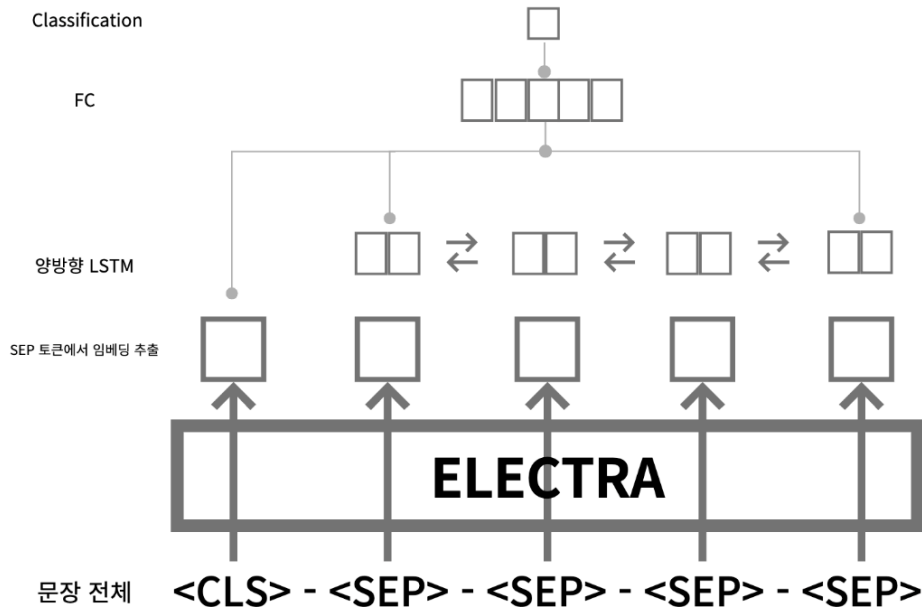
- 데이터셋을 불러온다.
- 전처리 진행 (결측치 및 중복 값 제거)
- 데이터를 split한다.
- 데이터 토큰화/벡터화 한다.
- Pre-trained 모델을 불러온다.
- 감정 정보 데이터를 통해 Train한다.
- 테스트 데이터를 통해 Test한다.

3.3 시퀀스 다이어그램

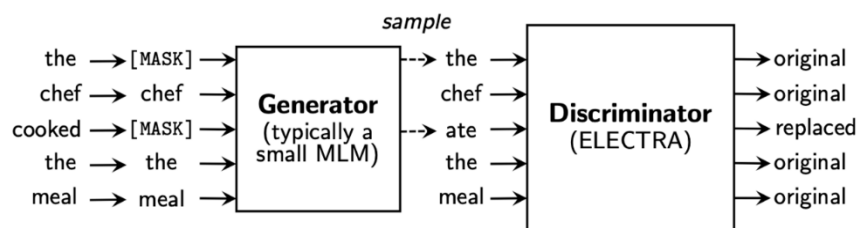


3.4 핵심 알고리즘

Model



1. ELECTRA



ELECTRA는 BERT의 파생모델 중 하나로, Generator와 Discriminator를 사용하여 사전학습을 진행한다. 이 모델은 BERT의 많은 계산을 요구한다는 단점을 해결하기 위해 RTD(Replaced Token Detection) 기법을 사용하였다. 여기서 RTD 기법이란 실제 입력의 일부 토큰을 그럴싸한 가짜 토큰으로 바꾸고, 진위 여부를 맞추게 하는 기법으로, 이는 모든 토큰에 대해 학습하여 효율적이면서 효과적이다.

Generator에서는 MLM 태스크를 수행한다. 즉, 15%의 확률로 토큰을 마스크 된 토큰([MASK])으로 교체하여 Generator에 입력하고, 마스크 된 토큰이 vocabulary 안에 있는 단어 중 어느 것에 속할 것인지에 대한 확률 분포를 출력한다.

그리고 Discriminator에서 토큰이 대체되었는지 여부를 판단한다. 우선 Generator를 이용하여

마스킹 된 입력 토큰들을 예측하고, generator에서 마스킹 할 위치의 집합에 해당하는 위치의 토큰을 generator의 softmax 분포에 대해 샘플링한 토큰으로 치환한다. 그 다음 치환된 입력에 대해 각 토큰이 원래 입력과 동일한지 여부를 예측한다.

일기의 문장의 흐름을 반영하기 위해 bi-LSTM 레이어를 추가하였다.

KSS(Korean Sentence Split)를 사용하여 문장 사이에<SEP> 토큰 추가하고, 첫번째 문장과 마지막 문장의<SEP> 토큰에서 추출한 임베딩들을 합쳐 bi-LSTM에 넣는다.

<CLS> 토큰과 첫 문장과 마지막 문장의 <SEP> 토큰에서 계산된 loss를 합쳐서 학습을 진행한다.

2. Classification

토큰화된 각 문장을 인코딩한 후 각 문장 사이의 [SEP] 임베딩을 bi-LSTM을 이용하여 감정의 흐름을 파악한 후, 첫 문장과 마지막 문장의 값과 [CLS] 임베딩을 Fully Connected 시키고 Sigmoid를 통해 각 감정에 대한 확률 값을반환한다.

3. Cosine Annealing Learning Rate

$$\eta_t = \eta_{\min} + \frac{1}{2}(\eta_{\max} - \eta_{\min}) \left(1 + \cos\left(\frac{T_{\text{cur}}}{T_{\text{max}}} \pi\right) \right)$$

Local Minimum에 빠지는 문제를 해결하기 위해서 기존의 선형 방식의 스케줄러에서 Learning Rate를 좀 더 효율적으로 변경하기 위해 여러가지 방법을 고민했다. 그 중 Learning Rate의 변화가 진동하는 방식을 채택하면 Local Minimum에서 벗어날 수 있다고 판단했다. 그래서 Cosine Annealing으로 변경했다.

4. Recommendation System

텍스트를 감정 분석 모델에 적용시키면 44개의 감정을 각각에 대한 확률 벡터가 반환된다. 이 중 상위 k개만 추출한 벡터를 사용한다. 일기에 대한 감정 벡터와 영화 줄거리에 대한 감정

벡터의 유사도를 산출한다. 산출된 유사도를 기반으로 영화가 추천된다.

타 모듈(KSS, Transformer(KcELECTRA))

KSS (Korean Sentence Splitter)

일기 특성상 여러 개의 문장으로 구성되어 있다. 본 프로젝트에서는 한 문장의 감정을 파악하고 그 감정들의 흐름을 파악하고자 KSS를 사용하여 문장 단위로 분리를 시켜주었다.

Transformers (KcELECTRA)

자연어 처리 모델의 성능을 위해서는 많은 양의 단어 사전으로 학습을 시켜야 한다. 그리고 앱을 이용해 일기를 입력한다는 점을 고려하여, 인터넷에서 많이 사용되는 단어들로 학습을 해야 한다. 그러한 단어들로 미리 학습이 된 모델을 이용할 수 있는 Transformer를 이용했고, 그 가운데 인터넷 댓글들을 이용해 학습시킨 KcELECTRA를 사용했다.

4. 시스템 구현

4.1 핵심코드 상세설명

1) Model

- model class 내 forward 함수

```
def forward(self, input_ids=None, attention_mask=None, sep_idx=None):

    electra_output = self.electra(input_ids, attention_mask)[0]

    cls = electra_output[:, 0, :] # <CLS> embeddings
    # sep 토큰 가져오기
    sep_idx_x = sep_idx[0]
    sep_idx_y = sep_idx[1]

    idx = 0
    cnt = 0
    longest = torch.where(sep_idx_x==torch.mode(sep_idx_x).values)[0].size()[0]
    # 초기화
    sep_embeddings = torch.zeros(cls.size(0), longest, self.embedding_size).to(self.device)

    # embedding 값 집어넣어주기
    for x, y in zip(sep_idx_x, sep_idx_y):
        if idx == x:
            sep_embeddings[x, cnt, :] += electra_output[x, y, :]
            cnt += 1
        else:
            idx += 1
            cnt = 0
            sep_embeddings[x, cnt, :] += electra_output[x, y, :]
```

```

# lstm 실행
lstm_output, (h, c) = self.lstm(sep_embeddings) # (batch_size, seq_length, embedding_size)

# lstm 처음과 끝 가져오기
sep_first = lstm_output[:, 0, :]
sep_last = lstm_output[:, -1, :]

# lstm 결과와 cls 토큰 합치기
concats = torch.cat((cls, sep_first, sep_last), dim=1)
# fc 레이어에 넣고 44개 output
x = self.gelu(concats)
output = self.fc1(x)

first_output = self.fc2(sep_first)
last_output = self.fc2(sep_last)

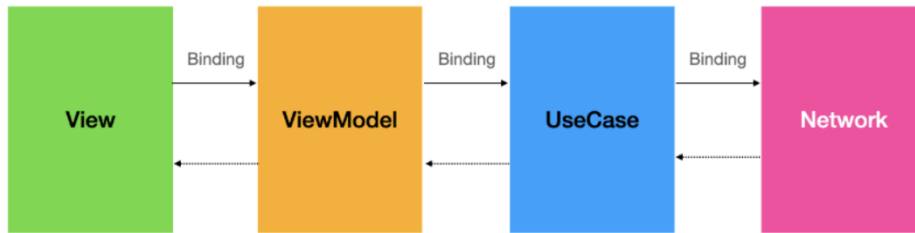
return output, first_output, last_output

```

설계해놓은 모델 구조를 코드로 구현한 부분이다. 우선 HuggingFace에서 불러온 kcELECTRA pre-trained 모델에 임베딩된 문장을 input으로 넣어 받은 output에서 CLS 토큰에 해당하는 임베딩을 가져온다. 그리고 batch 내 문장 수가 제일 많은 데이터를 기준으로 초기값의 shape을 설정하고 SEP 토큰에 해당하는 인덱스를 통해 SEP 토큰에 해당하는 임베딩들을 가져온다. 이렇게 함으로써 padding을 따로 진행하지 않아도 되며 데이터마다 문장 수가 달라서 생기는 오류를 해결할 수 있다.

다음으로 SEP 토큰 임베딩들을 bi-lstm에 넣고, 여기서 나온 output에서 첫 문장과 마지막문장에 해당하는 임베딩을 가져와서 기존에 가져왔던 CLS 토큰의 임베딩과 합쳐준다. 이를 gelu 함수에 넣은 후 fc레이어를 거쳐 44개 라벨에 대응하는 값을 반환한다. bi-lstm에서 나온 output들 또한 fc레이어에 넣어서 44개의 라벨에 대응하는 값을 반환하며, 이들은 각각 전체에 대한 loss, 첫번째 문장에 대한 loss, 마지막 문장에 대한 loss를 계산할 때 사용되고, 모델 학습 시 각 loss 값들을 더한 값으로 학습을 진행한다.

2) App



MVVM 아키텍처를 기반으로 시스템을 구성했다. 중간에는 UseCase 계층을 두어서 동일한 작업에 대한 중복코드를 줄이도록 처리했다.

```
private func bind() {
    viewModel.emotionListSubject
        .bind(to: emotionCollectionView.rx.items(cellIdentifier: EmotionCollectionViewCell.identifier, cellType:
EmotionCollectionViewCell.self)) { (row, element, cell) in
        let color = self.colorList[row % 5]
        cell.configure(name: element.emotion.rawValue, color: color, percentage: element.percentage)
    }
    .disposed(by: disposeBag)
}
```

Combine을 통해 데이터를 Binding하여 받아오는 형태이며, Network 요청 역시 마찬가지로 구성된다.

3) Server

```
@api.route('/emotion', methods=['POST'])
class Emotion(Resource):
    def post(self):
        ...
```

Flask를 통해 HTTP 통신을 구현해 POST 요청을 받을 수 있도록 구현했다. 로컬에 구성된 후 ngrok를 통해 외부에 서버를 열어 앱에서도 사용이 가능하도록 처리했다.

4.2 구현시 문제점과 해결방법

- 감정의 흐름을 반영하기 위한 문제

학습에 사용한 KOTE 데이터는 하나 이상의 문장 전체에 대한 감정이 라벨링되어 있다. 이를 가지고 학습을 시킨 모델 또한 여러 문장을 입력하면 그에 대한 감정이 출력된다. 일기 문장들의 감정의 흐름을 반영하려면 모델과 데이터를 수정해야 했다.

그래서 문장의 감정의 흐름을 파악할 수 있게 bi-LSTM층을 추가하고, 학습을 위해 기존의 데이터를 KSS를 이용하여 문장 단위로 나누어 준 뒤 각 문장들의 감정과 전체의 감정으로 새로 라벨링하는 작업을 진행했다.

- 영화 추천의 문제

처음 영화 데이터는 KMDB의 API를 이용했다. 하지만 실제로 서비스되고 있는 영화의 수는 적었고, 단편 영화의 개수가 많았다. 이는 실제로 사용자에게 영화를 추천해도 의미가 없음을 의미했고, 다른 영화 데이터를 찾아야했다.

그래서 사람들이 자주 이용하는 OTT 플랫폼(왓챠, 넷플릭스 등 총 6개)에 있는 영화를 정보 제공 사이트에서 크롤링하여 데이터를 구축하였다. 사용자들이 영화에 접근하기 편하도록 영화를 볼 수 있는 OTT와 상영시간, 주연 배우 등 영화 시청에 도움이 되는 데이터를 추가로 구축했다.

그럼에도 영화를 추천하는 데 있어 데이터의 제약으로 단순히 유사도만 사용할 수밖에 없었다. 사용자의 감정에 따른 영화 시청 내역 데이터가 전혀 존재하지 않았기 때문이다. 하지만 단순히 유사도를 사용하였을 때도 테스터에게 괜찮은 평가를 받았다는 점에서 감정의 유사도는 영화를 추천할 때 유용한 feature임을 알 수 있었다.


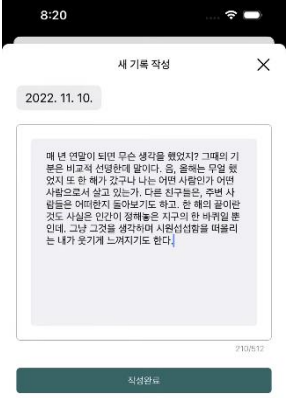
- 전체적인 성능 문제


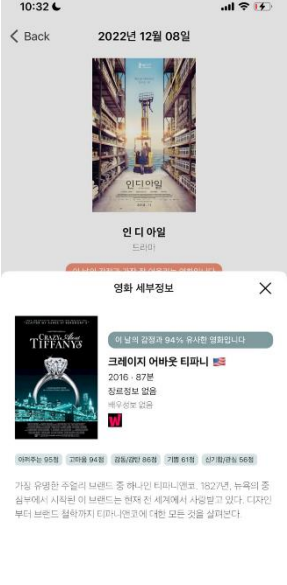
모델의 성능을 높이기 위해서 데이터 라벨링, 모델 레이어 추가, 스케줄러 변경의 시도를 통해 성능을 올렸다. 마지막으로 모델의 하이퍼파라미터를 튜닝하여 성능을 높여보고자 시도했다.

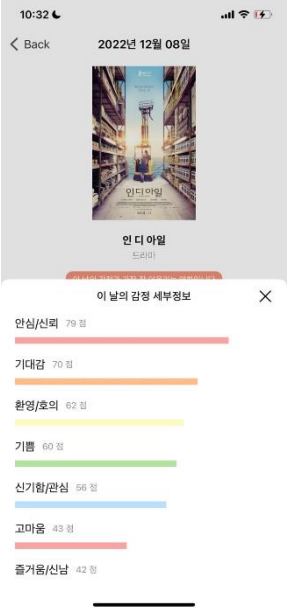

우선 직접 하이퍼파라미터를 변경하며 성능을 확인하였다. ELECTRA의 activation 함수, lstm 레이어 수 변경, threshold 변경 등을 시도했지만 직접 하이퍼파라미터를 변경하면서 성능을 확인하는 것은 비효율적이라고 생각하여 파라미터 서치 방법 중 하나인 RAY를 사용하기로 했다.


그래서 RAY를 이용해 모델의 하이퍼 파라미터를 주어진 범위로 검색했다. 하지만, 모델의 성능이 더 올라가지 않았고, 초기 설정 값을 그대로 이용하기로 했다.

5. 시스템 실행 결과

	<p>[메인 화면]</p> <ul style="list-style-type: none"> - 앱의 가장 메인인 되는 화면입니다. - 현재까지 작성한 일기들이 모여서 보이며, 검색 또한 가능합니다. - 지난 일기에 대한 삭제 기능 또한 포함되어 있습니다. - 오른쪽 상단에 위치한 버튼을 눌러서 일기 입력 화면으로 이동할 수 있습니다.
	<p>[일기 입력 화면]</p> <ul style="list-style-type: none"> - 일기를 입력하는 화면입니다 - 원하는 내용을 입력하고 완료버튼을 클릭하면 메인 화면에 결과가 나타납니다.

	<p>[일기 결과 화면]</p> <ul style="list-style-type: none"> - 작성한 일기의 결과를 터치하여 이동합니다. - 이 날의 감정과 가장 잘 어울리는 영화를 중심으로 이 날의 감정, 다른 추천 영화 등을 요약하여 확인할 수 있습니다.
	<p>[영화 상세 화면]</p> <ul style="list-style-type: none"> - 추천된 영화의 상세 정보를 볼 수 있습니다. - 제목, 국가, 개봉연도, 배우정보, OTT 정보, 줄거리 등의 정보를 파악할 수 있습니다.

	<p>[감정 상세 화면]</p> <ul style="list-style-type: none"> - 이 날 감정의 자세한 정보를 볼 수 있습니다. - 44개의 감정 중 일정수치를 넘어선 경우에만 표에 나타납니다.
	<p>[감정 총괄 화면]</p> <ul style="list-style-type: none"> - 지금까지 입력된 감정들의 평균치를 표시합니다.

	<p>[설정 화면]</p> <p>- 기록된 일기를 한 번에 지우거나 개발자 정보를 확인하는 작업을 수행할 수 있습니다</p>
---	---

6. 시스템 시험 결과

감정 분석 모델

모델	microF1	차이
kcELECTRA	0.57	
KcELECTRA + bi-LSTM	0.62	+0.05
KcELECTRA + bi-LSTM + CosineAnnealingLR Scheduler	0.65	+0.03

- KcELECTRA + bi-LSTM 아키텍처를 적용했을 때 kcELECTRA 단독 모델에 비해 성능이 5% 향상되었다.

- 여기에 스케줄러를 Cosine Annealing LR 스케줄러로 변경한 결과 성능이 추가로 3% 향상되어 총 8%가 향상되었다.

추천 시스템

졸업작품 전시회 기간 중 15명에게 받은 종합적인 평가이다.

대부분의 이용자에게 좋은 평가를 받았고 평가를 통해서 부족한 점을 알게되었다.

좋은 점	부족한 점
"정말 신기해요. 제 일기와 맞는 영화가 추천되네요."	"슬픈 감정일 때, 슬픈 영화만 추천받아서 아쉬워요 행복한 영화도 추천받고 싶어요"
"그날의 감정을 알 수 있고 몰랐던 영화들도 알 수 있어서 좋은 것 같아요"	"영화 결과에 따른 만족과 불만족 평가하고 싶어요"
"매일 매일 일기를 작성하면서 제 감정을 체크 할 수 있는 점이 좋은 것 같아요"	"제가 좋아하는 장르를 추천받고 싶어요"

7. 팀원 및 역할 분담

김예지: 모델 설계 및 구현하였습니다. 또한 영화 데이터 크롤링 및 데이터 라벨링을 하였습니다.

손소영: 모델을 설계하고 구현하였습니다. 데이터를 임베딩하고, 필요한 정보를 가져와 batch로 묶는 dataset 클래스, ELECTRA에 bi-LSTM을 적용한 모델, trainer 함수, 구현한 모듈을 사용하여 모델을 학습시키는 main.py를 구현하였습니다. 또한 영화 데이터 크롤링을 보조하였습니다. 그리고 학습된 모델을 불러와 영화추천을 하는 프로세스가 서버에서 보다 빠르게 잘 작동할 수 있도록 코드를 수정하였습니다.

정창해 : 모델을 설계하고 구현했습니다. 학습에 필요한 데이터와 모듈들을 살펴보고, 데이터를 모델에 사용하기 쉽게 전처리하는 과정을 담당했습니다. 또한, 모델을 앱에 적용시키기 위한 코드들을 구현했습니다.

박인우: Client 역할을 해주는 iOS 앱을 설계하고 구현했습니다. 또한 모델을 돌리고 결과를 반환해주는 Server도 함께 설계하고 구현했습니다.

8. 향후 졸업작품의 활용 방안

크게 두 가지 측면으로 졸업작품을 활용할 수 있습니다.

첫째, 앱을 서비스로 출시할 수 있습니다. 하지만 서비스로 출시하기 위해서는 개선 사항이 몇 가지 있습니다. 우선 추천시스템의 고도화가 필요합니다. 현재 추천시스템은 일기, 영화 줄거리에 대한 44개 감정에 대한 확률 벡터를 코사인 유사도로 계산한 굉장히 단순한 시스템입니다. 하지만 보다 많은 사용자들에게 만족을 주기 위해서는 사용자의 어떤 감정일 때 어떤 영화를 시청하였는지에 대한 데이터를 구축하고, 이를 활용하여 추천시스템이 감정 정보 이외의 다른 정보도 사용할 수 있도록 해야 합니다.

하지만 앱을 출시하게 된다면 서비스적인 측면에서 확장성이 용이할 것으로 생각됩니다. 우선 일기로부터 도출된 감정을 기반으로 정신과 상담 연계를 진행할 수 있습니다. 그리고 영화 뿐만 아니라 드라마, 도서 등 다른 콘텐츠를 추천하는 방향으로 확장이 가능합니다.

둘째, 논문을 작성할 수 있습니다. 시간적인 제약으로 인해 많이 진행되지 않았던 라벨링을 추가로 진행하여 성능을 확인하고, 모델 하이퍼파라미터 튜닝을 진행하고, 다른 구조에 대한 다양한 실험을 통하여 성능을 높인 후 이 방법에 대하여 논문을 작성할 수 있을 것 같습니다.

9. 맺음말

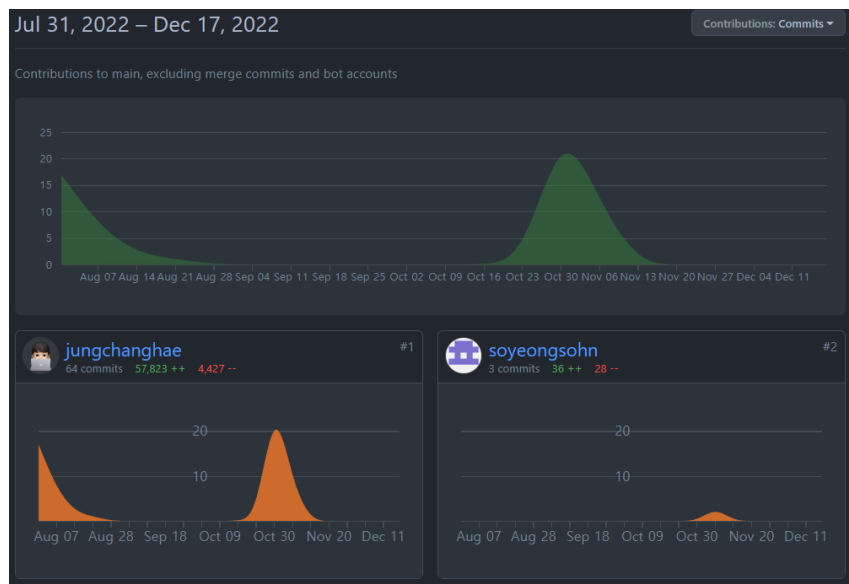
김예지: bert모델을 시작으로 다양한 nlp모델들을 공부하면서 모델들의 구조와 특징에 대해 공부할 수 있는 시간이었습니다. 최적의 모델을 찾기 위해서 다양한 모델들을 구현해보았고, 이 과정에서 어떤 모델이 저희 프로젝트의 태스크에 적합한지 찾을 수 있었습니다. 그리고 모델의 정확도를 높이기 위하여 데이터 라벨링을 직접 진행해 보면서, 데이터 정확성의 중요도를 알게 되었습니다.

손소영: 단순히 이론만 알고 있었던 대형 언어 모델과 딥러닝을 프로젝트를 통해 직접 다뤄볼 수 있었습니다. 데이터셋 준비부터 모델 학습 및 저장까지 모든 단계를 직접 구현해봄으로써 모델 구현 실력을 향상시킬 수 있었던 좋은 기회였습니다. 설계한 모델을 직접 구현하는 과정에 많은 오류를 맞닥뜨려 어려움이 있었지만, 이를 해결하며 pytorch와 numpy를 더 능숙하게 다룰 수 있게 되었습니다. 비록 설정했던 목표 성능은 달성하지 못했지만, 이 프로젝트를 통해 다양한 실험을 해보며 연구에 대한 흥미가 생겼고, 향후 연구를 하는 데 좋은 밑거름이 될 수 있을 것 같습니다.

정창해: 자연어 처리 분야를 전혀 모르는 상태로 이 프로젝트를 시작했었는데, 두 학기 동안 공부하고 진행하면서 어떻게 구현하면 좋을지 막막했던 것이, 이제 다른 모델에 대해서 더 실험해보지 못한 아쉬움이 남는 것을 보고 많은 성장을 했다고 생각합니다. 특히 대학원에 진학 후 이 프로젝트의 성능을 더 높여보고자 하는 목표가 생기고, 자연어 처리 분야에 대한 흥미가 본격적으로 생긴 것 같아서 저에게 큰 의미를 준 두 학기였다고 생각합니다.

박인우 : 직접적으로 모델 개발에 참여하지는 않았지만 자연어처리의 기본 이론에 대해 배울 수 있었습니다. 확장성 및 테스트 용이성을 고려해 iOS App을 개발하면서 더 확장성있는 앱 구조를 구성하는 방법을 익혔습니다. 시스템 설계서를 포함한 문서 작성을 진행하고 프로젝트를 진행하면서, 문서 작성의 중요도 또한 느낄 수 있었습니다. Flask를 기반으로 모델 서버를 제작하고 로컬 환경을 외부에 공유하여 앱에서 데이터를 조회하는 방법을 배웠습니다. 이를 통해 보다 완성도 있는 앱 서비스를 제작할 수 있는 능력을 갖추었습니다.

깃허브 활동 내역



시연영상 링크

<https://www.youtube.com/watch?v=LEfbPEpC9IY>

참고자료

User Guide for KOTE: Korean Online Comments Emotions Dataset

(<https://arxiv.org/pdf/2205.05300.pdf>)

[함영준의 마음 디톡스] (25) 마음의 소리나 감정을 잘 모른채 살아가는 현대인

(<http://www.mindgil.com/news/articleView.html?idxno=69419>)

감정 온톨로지 기반의 영화 추천 기법 (김옥섭, 이석원,

2015)(<http://koreascience.or.kr/article/JAKO201531736561252.pdf>)

Cosine Annealing Scheduler (<https://sanghyu.tistory.com/113>)

학습 데이터(<https://github.com/searle-j/KOTE>)

일기 테스트 데이터: 모두의 말뭉치 - 비출판물 말뭉치 (<https://corpus.korean.go.kr/#down>)