



포비카의 이야기 보따리

빨간망토와 아기돼지 삼형제

♡ 정수경
🐷 김민재
🐷 유창호
🐷 최창훈



INDEX

1. 프로젝트 배경
2. 팀 구성 및 역할
3. 프로젝트 수행 절차 및 결과
4. 프로젝트 후기



1. 프로젝트 배경

- 1.1 프로젝트 개요
- 1.2 프로젝트 과정

1. 프로젝트 배경

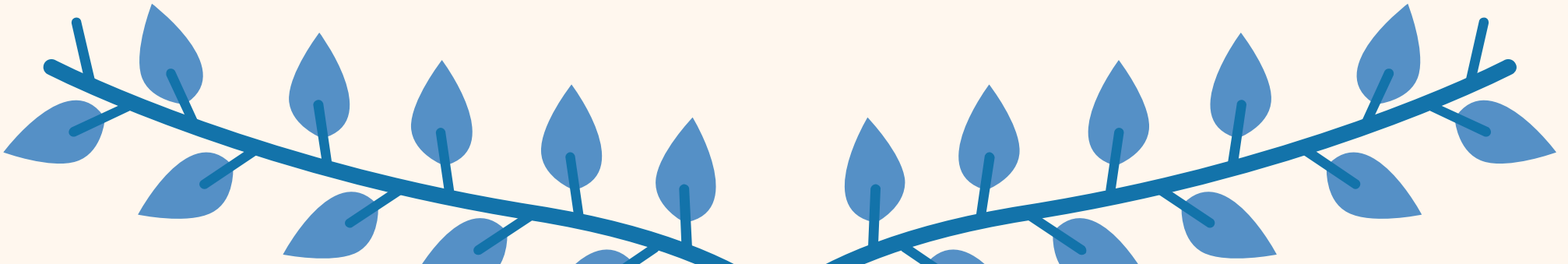
1.1 프로젝트 개요

A. 문제인식

맞벌이, 재택근무
부모님으로 인해 혼자
있어야하는 아이들을
위해 해줄 수 있는
일이 있을까?

부모님이 읽어주듯
감정을 넣어 동화책을
읽어 주는 서비스를
만들면 어떨까?

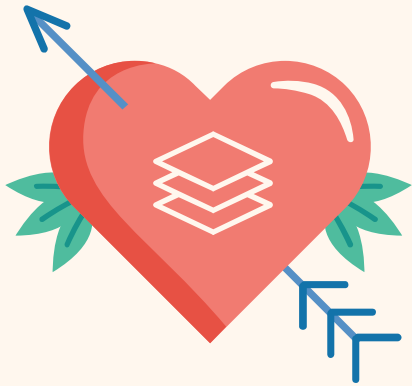
시력 저하 등으로
어려움을 겪는 분에게
책을 대신 읽어 주는
서비스를 만들면
어떨까?



1. 프로젝트 배경

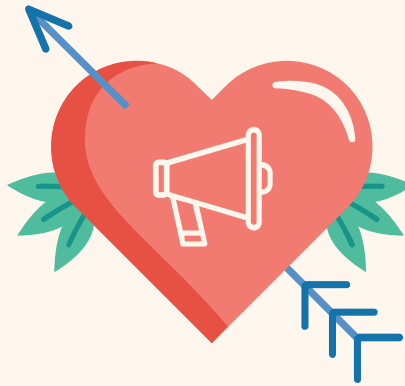
1.1 프로젝트 개요

B. Mission



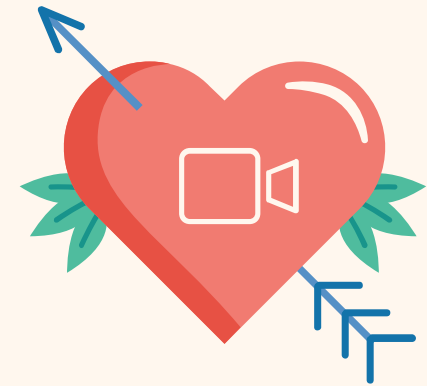
Mission 1

동화책 내용을 넣으면
텍스트를 인식하고
감정을 분석한다



Mission 2

분석된 감정에 맞는
배경음악을 동화책과
함께 출력한다



Mission 3

조금 더 쉽고,
편안하게, 아이들이
즐겁게 읽을 수
있도록 서비스를
만든다

1. 프로젝트 배경

1.1 프로젝트 개요

C. Solutions

Step 1

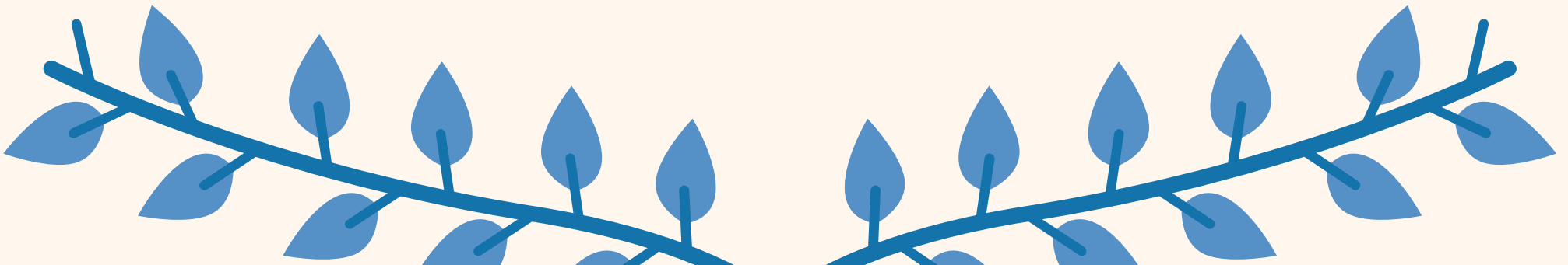
자연어 처리를 통한
감성분석 딥러닝 모델
구현

Step 2

음성인식 API 를
활용하여 텍스트를
읽어주고, 분석된 감성에
맞는 배경음악 송출

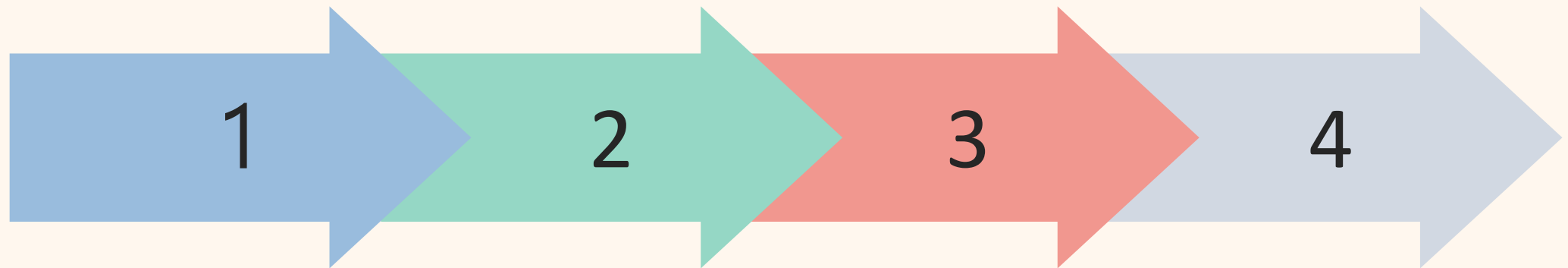
Step 3

Django 를 이용하여
웹서비스 구현



1. 프로젝트 배경

1.2 프로젝트 과정



기획

- 1.1 문제인식
- 1.2 프로젝트 주제선정
- 1.3 기획안 작성
- 1.4 WBS

데이터

- 2.1 데이터 수집
- 2.2 탐색적 데이터 분석 (EDA)
- 2.3 데이터 전처리 (라벨링)

딥러닝 모델

- 3.1 RNN
- 3.2 LSTM
- 3.3 BERT
- 3.4 KoBERT

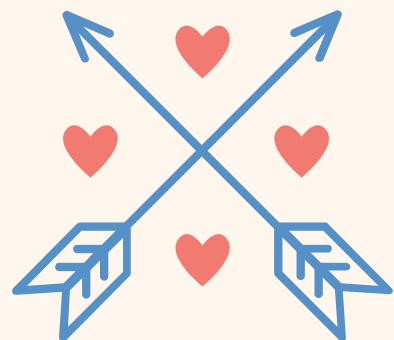
웹 서비스 개발

- 3.1 MySQL
- 3.2 Django
- 3.3 API



2. 팀 구성 및 역할

이름	역할
 정수경 (팀 리더)	<ul style="list-style-type: none"> • Deep learning model (RNN, LSTM) • Web service 개발 (Django, MySQL)
 김민재 (팀원)	<ul style="list-style-type: none"> • Deep learning model (RNN, LSTM) • Web service 개발 (Django, MySQL)
 유창호 (팀원)	<ul style="list-style-type: none"> • Deep learning model (RNN, LSTM, BERT) • PPT 제작
 최창훈 (팀원)	<ul style="list-style-type: none"> • Deep learning model (RNN, LSTM, BERT, KoBERT) • Web service 개발 (Django) • 피드백 총괄



Hello!

We are 포비카!





3. 프로젝트 수행 절차 및 결과

- 3.1 PDCA
- 3.2 개발일정
- 3.2 Deep learning
- 3.2 Web service

3.1 PDCA

1. Plan

- 1.1 프로젝트 주제 선정
- 1.2 프로젝트 대상 파악
- 1.3 시스템 구성도 설계
- 1.4 ERD 설계
- 1.5 개발 기획서 작성
- 1.6 WBS 작성

4. Action

- 4.1 개선 활동



2. Do

- 1.1 Deep learning
- 1.2 Web service

3. Check

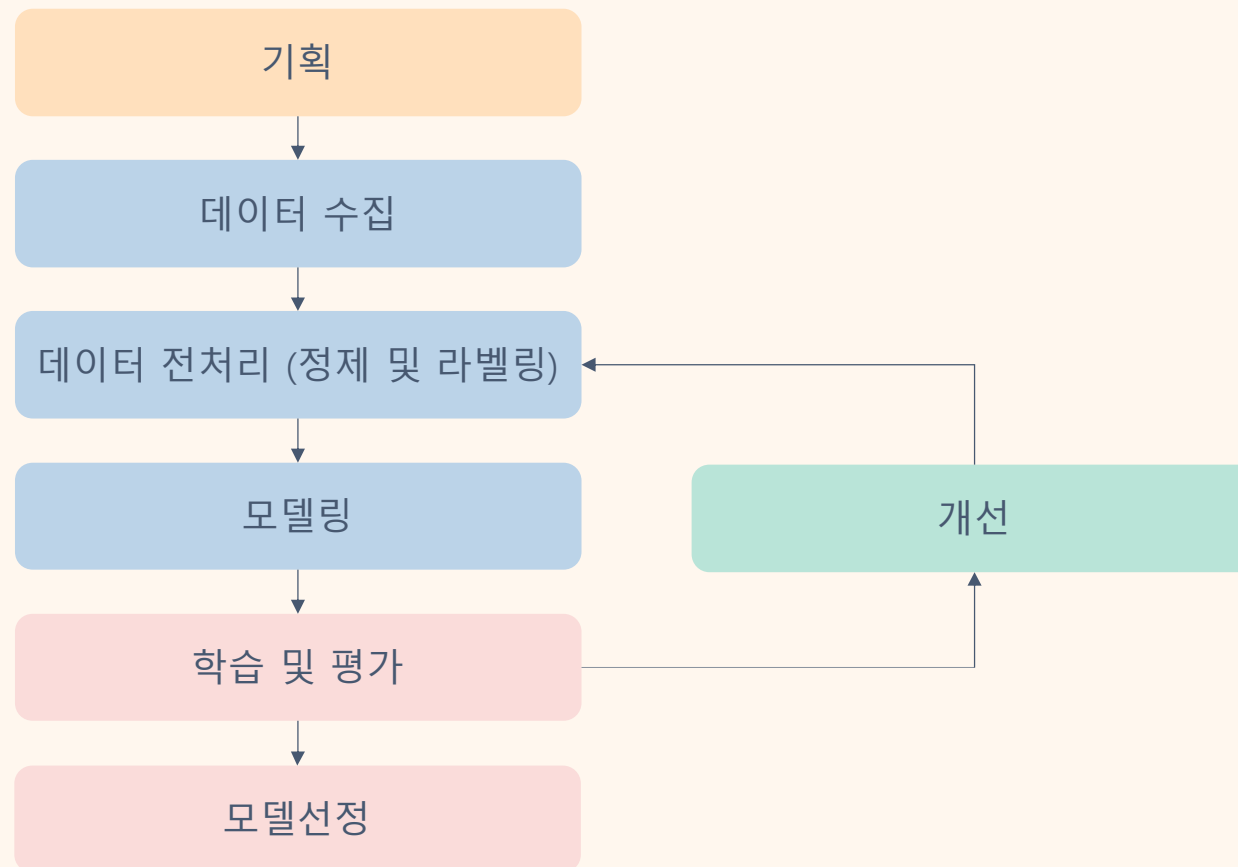
- 3.1 검토
- 3.2 평가

3.2 개발일정

구분	기간	활동	비고
기획 및 데이터	• 4/5 (월) ~ 4/11(일)	<ul style="list-style-type: none"> 프로젝트 기획 및 WBS 작성 데이터 수집 및 전처리 	공통
개발	• 4/12(월) ~ 4/18(일)	<ul style="list-style-type: none"> 감성 학습모델 구축 및 평가 (RNN & LSTM) 	공통
	• 4/19(월) ~ 4/22(월)	<ul style="list-style-type: none"> Web service 개발 Deep learning model 성능개선 	<ul style="list-style-type: none"> Web service 2명 Deep model 2명
수정/보완	• 4/23(금) ~ 4/26(금)	<ul style="list-style-type: none"> 피드백 의견 반영하여 기능 보완 버그 수정 	
총 개발기간	• 4/5 (월) ~ 4/26(월)		

3.3 Deep learning

Deep learning



3.3 Deep learning

3.3.1 데이터 수집

- 청와대 전래동화 100권

놀이마당

[우리나라 여행기](#) [즐거운 전통놀이](#) [재미있는 전래동화](#) [함께 보는 책](#)



별난 재주 삼형제

옛날 시골마을에 별난 재주 세 형제가 살았습니다. 눈이 밝은 맏이는 별별 것을 다 봅니다. 둘째는 천하장사 바둑들도 번쩍번쩍. 개구쟁이 막내는 회한한 재주인데 매 맞는 재주입니다. 회초리로 때리면 간지럽다고 깔깔대고 절구공이 내리치면 시원하다 깔깔댑니다. 그러던 어느 날 마을에 ..



밥만 먹는 밥벌레 장군

옛날 어느 마을에 한 총각이 살았습니다. 사람들이 그 총각을 밥벌레 장군이라고 놀렸습니다. 왜냐하면, 매일 밥만 먹고 빈둥빈둥 놀기 때문입니다. 밥도 한두 그릇 먹는 게 아니라 가마솥 통째로 먹어 치웁니다. 그래서 몸집도 엄청 큼니다. 그런데 힘은 어찌나 없는지 밥그릇 하나도 들지 ..

- 네이버 블로그 동화책

영어동화 (해석)

그림 형제 동화집 전체 (해석)(pdf, txt텍스트, 한글, 워드 파일 다운 가능)(2018년 수정본)



마음
2018. 1. 7. 7:52

+ 이웃추가

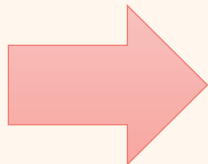
그림 형제 동화집 전체 (해석)(pdf, txt텍스트, 한글, 워드 파일 다운 가능)(2018년 수정본)

3.3 Deep learning

3.3.2 데이터 전처리

A. 라벨링 – How to 감정분류

무감정
기쁨
긴장감
놀람
분노
속임수
긴박
슬픔
위로
잔잔
평화



0 의미없는 문장

ex) 이 말을 들은 그녀들이 그에게 말했어요

1 평화/기쁨

Ex) 그러자 신부는 너무너무 기뻐 그에게 키스를 하고 진실로 사랑했어요

2 긴장/속임수

ex) 뭔가 공공이 속이 있는 게 분명해

3 최고조/분노

ex) 우리 남편이 죽을 뻔 했어! 그 놈이 내 남편을 계단 아래로 집어 던져 남편의 다리가 부서졌어

4 슬픔

ex) 누가 널 괴롭히니, 공주야? 네 모습을 보고 돌도 눈물을 똑똑 흘리겠다

5 잔잔

ex) 아이들은 서로서로 손을 맞잡고서 그 연못을 돌면서 술래잡기를 했습니다

3.3 Deep learning

3.3.2 데이터 전처리

A. 라벨링 – 감정분류 Rule

1. 문맥을 생각하고 라벨링 실시
Ex) 공주는 눈물을 흘렸어요 (기쁨의 눈물 / 슬픔의 눈물) 구별
2. 1인칭 주인공시점이 아닌 3인칭 전지적작가 시점 감정 평가
3. 알고있는 동화가 나온 경우 배경지식에 감정 휩쓸리지 않기
4. 동화책이니 좋게좋게 생각하기
5. 누구누구가 말했어요 (의미없는 문장)으로 평가
6. 애매한 감정은 (잔잔)으로 평가

3.3 Deep learning

3.3.2 데이터 전처리

A. 라벨링 – 감정라벨링 결과

- 동화책 124권 (문장 13,059개 / 감정분류 6가지)

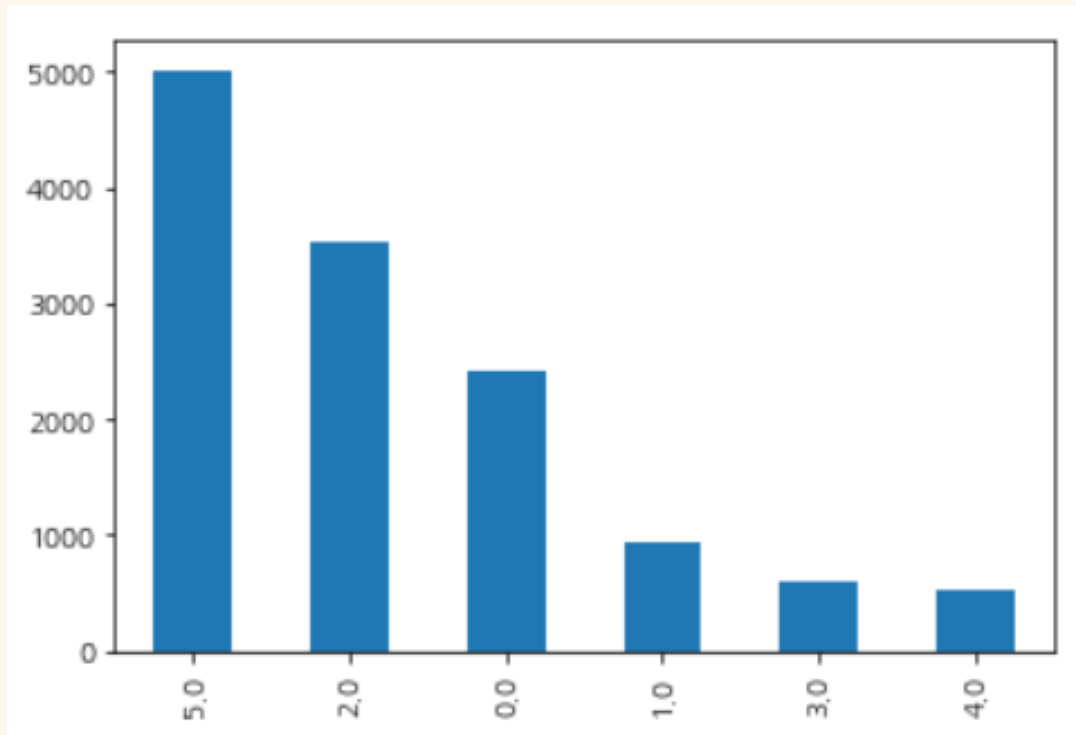
1	Text	Feeling	Talking	book_id	sentence_id
2	'인도'가 적에게 포위를 당했는데, 적들이 은화 6백 개를 내놓을 때까지	2	0	5001	1
3	그래서 주민들이 북을 치며 선포하길,	2	0	5001	2
4	*누구든 그 돈을 마련하는 사람을 시장으로 삼겠다*	2	1	5001	3
13046	아가씨는 전날 밤 뒤뜰에서 호랑이를 만났던 이야기를 했습니다	5	0	7009	21
13047	나무꾼은 그동안 있었던 이야기를 해주었습니다	5	0	7009	22
13048	아가씨는 나무꾼의 마음씨에 감동해 나무꾼과 부부가 되었습니다	5	0	7009	23
13049	*당신 집에 인사를 드리러 가야 하는데 가져갈 선물이 없구려*	5	1	7009	24
13050	다음 날 아침에 마당에는 많은 선물과 소, 당나귀가 있었습니다	5	0	7009	25
13051	*이번에도 호랑이가 도와줬구나, 고맙다 호랑이야*	1	1	7009	26
13052	덕분에 아가씨네 가족들도 나무꾼을 마음에 들어 했고 행복한 생활	1	0	7009	27
13053	시간이 한참 흘러 호랑이를 잡으면 큰 선물을 내리겠다는 임금님의	5	0	7009	28
13054	저기 멀리서 호랑이 한 마리가 보여 활을 쏘아 쓰러뜨렸습니다	5	0	7009	29
13055	그런데 그 호랑이는 바로 옛날에 나무꾼이 도와주었던 호랑이였습니	4	0	7009	30
13056	*저는 나이가 들어 어차피 죽을 목숨이니 저를 임금에게 제물로 바치	4	1	7009	31
13057	그러곤 호랑이는 숨을 거두었습니다	4	0	7009	32
13058	나무꾼은 숨을 거둔 호랑이를 차마 제물로 바치지 못하고 양지바른	4	0	7009	33
13059	목에 걸린 뼈를 나무꾼이 빼준 이후로 죽을 때까지 나무꾼에게 은혜	4	0	7009	34

3.3 Deep learning

3.3.2 데이터 전처리

A. 라벨링 – 감정분류 분포

- 잔잔 (5) > 긴장 (2) > 무의미 (0) > 기쁨/평화 (1) > 최고조 (3) > 슬픔(4)



3.3 Deep learning

3.3.3 RNN & LSTM

A. 불용어 제거 및 Tokenizer

불용어 제거

```
file = open('../data/stopwords2.txt', 'r')
words = file.readlines()
stopwords = [i.replace('\n', '') for i in words]
stopwords = [i.replace('\t', '') for i in stopwords]
print(stopwords)
len(stopwords)
```

```
['이', '있', '하', '것', '들', '그', '되', '수', '이', '보', '알', '없', '나', '사람', '주', '아니', '등', '같', '우리', '때', '일', '년', '가', '한', '지', '대하', '오', '말', '일', '그럴', '위하', '때문', '그것', '두', '말하', '알', '그러나', '밤', '못하', '일', '그런', '오', '문제', '더', '사회', '말', '그리고', '출', '크', '따르', '중', '나오', '가지', '씨', '시키', '만들', '지금', '생각', '하', '그러', '속', '하나', '집', '살', '모르', '적', '월', '데', '자신', '안', '어떤', '내', '내', '경우', '명', '생각', '시간', '그녀', '다시', '이런', '앞', '보이', '번', '나', '다른', '여덟', '여자', '개', '전', '들', '사실', '이월', '점', '싶', '말', '정', '도', '좁', '원', '잘', '통하', '소리', '놀']
```

TOKENIZER

```
threshold = 3
total_cnt = len(tokenizer.word_index) # 단어의 수
rare_cnt = 0 # 등장 빈도수가 threshold보다 작은 단어의 개수를 카운트
total_freq = 0 # 훈련 데이터의 전체 단어 빈도수 총 합
rare_freq = 0 # 등장 빈도수가 threshold보다 작은 단어의 등장 빈도수의 총 합

# 단어와 빈도수의 쌍(pair)을 key와 value로 받는다.
for key, value in tokenizer.word_counts.items():
    total_freq = total_freq + value

    # 단어의 등장 빈도수가 threshold보다 작으면
    if(value < threshold):
        rare_cnt = rare_cnt + 1
        rare_freq = rare_freq + value

print('단어 집합(vocabulary)의 크기 : ', total_cnt)
print('등장 빈도가 %s번 이하인 희귀 단어의 수: %s'%(threshold - 1, rare_cnt))
print("단어 집합에서 희귀 단어의 비율:", (rare_cnt / total_cnt)*100)
print("전체 등장 빈도에서 희귀 단어 등장 빈도 비율:", (rare_freq / total_freq)*100)
```

```
단어 집합(vocabulary)의 크기 : 29390
등장 빈도가 2번 이하인 희귀 단어의 수: 23372
단어 집합에서 희귀 단어의 비율: 79.52364749914938
전체 등장 빈도에서 희귀 단어 등장 빈도 비율: 26.205062221539404
```

0 '인도' 가 적에게 포위를 당했는데, 적들이 은화 6백 개를 내놓을 때까지 물러나지 ...
1 그래서 주민들이 북을 치며 선포하길,
2 *누구든 그 돈을 마련하는 사람을 시장으로 삼겠다*
3 라고 했습니다
4 때마침 가난한 어부가 자신의 아들과 호수에서 물고기를 잡아 생활하다, 그 적들이 들...



```
[716, 622, 175, 1978, 3],
[9, 1266],
[499, 1, 564, 421],
[42, 623],
[288, 422, 1553, 12, 1738, 655, 1, 4272, 423, 455, 716, 622, 1072, 812, 361],
[9, 219, 121, 1, 564],
[132, 4273, 592],
[107],
[499, 1148],
[1554, 2280, 2281],
[565, 151],
[624, 33, 865, 48, 8, 866, 356, 813],
[84, 865, 1979, 717, 683, 625, 401, 1267],
```

3.3 Deep learning

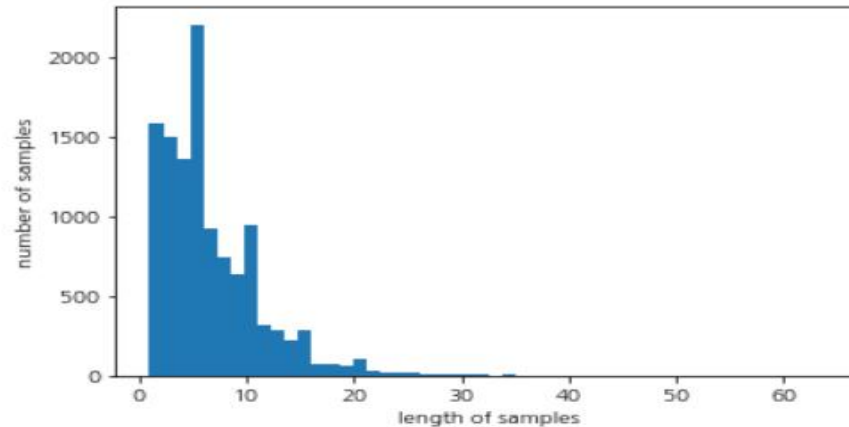
3.3.3 RNN & LSTM

B. PADDING

PADDING

```
print('문장의 최대 길이 :', max(len(l) for l in X_train))  
print('문장의 평균 길이 :', sum(map(len, X_train))/len(X_train))  
plt.hist([len(s) for s in X_train], bins=50)  
plt.xlabel('length of samples')  
plt.ylabel('number of samples')  
plt.show()
```

문장의 최대 길이 : 64
문장의 평균 길이 : 6.731453096259963



```
array([ 0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  
       0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  
       0,  0,  0, 716, 622, 175, 1978, 3], dtype=int32)
```

```
max_len = 30  
below_threshold_len(max_len, X_train)
```

전체 샘플 중 길이가 30 이하인 샘플의 비율: 99.71095734431111

3.3 Deep learning

3.3.3 RNN & LSTM

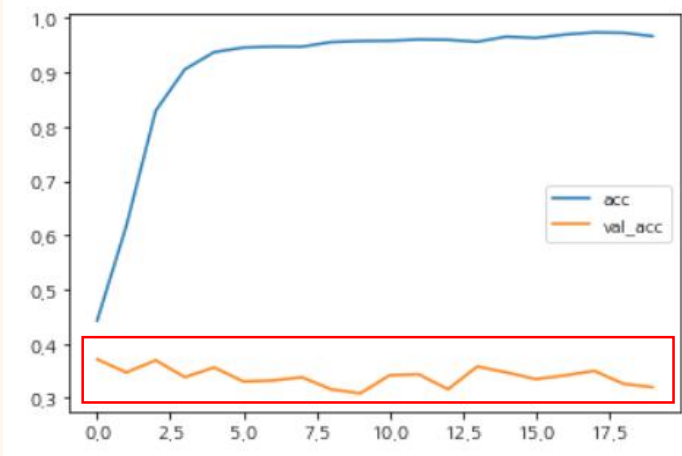
B. 학습 및 평가

학습

```
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['acc'])  
history = model.fit(X_train, y_train, epochs=20, callbacks=[], batch_size=20, validation_split=0.2, shuffle = True)
```

```
Epoch 15/20  
457/457 [=====] - 7s 15ms/step - loss: 0.0887 - acc: 0.9690 - val_loss: 4.3013 - val_acc: 0.3472  
Epoch 16/20  
457/457 [=====] - 7s 15ms/step - loss: 0.0831 - acc: 0.9662 - val_loss: 4.0543 - val_acc: 0.3349  
Epoch 17/20  
457/457 [=====] - 7s 15ms/step - loss: 0.0728 - acc: 0.9742 - val_loss: 4.7668 - val_acc: 0.3415  
Epoch 18/20  
457/457 [=====] - 7s 15ms/step - loss: 0.0620 - acc: 0.9767 - val_loss: 4.8707 - val_acc: 0.3503  
Epoch 19/20  
457/457 [=====] - 7s 15ms/step - loss: 0.0626 - acc: 0.9770 - val_loss: 4.7006 - val_acc: 0.3262  
Epoch 20/20  
457/457 [=====] - 7s 15ms/step - loss: 0.0781 - acc: 0.9718 - val_loss: 4.5337 - val_acc: 0.3201
```

RNN
정확도 0.35



3.3 Deep learning

3.3.3 RNN & LSTM

C. 원인분석 1 – 전처리 과정에 대한 문제 접근

EDA 기반 제 3 사분위 길이로 재선정

```
print('문장 길이 최대 값: {}'.format(np.max(train_length)))
print('문장 길이 최소 값: {}'.format(np.min(train_length)))
print('문장 길이 평균 값: {:.2f}'.format(np.mean(train_length)))
print('문장 길이 표준편차: {:.2f}'.format(np.std(train_length)))
print('문장 길이 중간 값: {}'.format(np.median(train_length)))
# 사분위의 대한 경우는 0~100 스케일로 되어있음
print('문장 길이 제 1 사분위: {}'.format(np.percentile(train_length, 25)))
print('문장 길이 제 3 사분위: {}'.format(np.percentile(train_length, 75)))
```

```
문장 길이 최대 값: 259
문장 길이 최소 값: 1
문장 길이 평균 값: 33.54
문장 길이 표준편차: 23.22
문장 길이 중간 값: 29.0
문장 길이 제 1 사분위: 16.0
문장 길이 제 3 사분위: 45.0
```

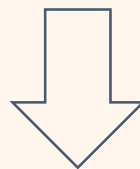
문장 맞춤법 교정 모델 사용

```
# 띄어 쓰기 및 맞춤법 보정
from hanspell import spell_checker
spelled_sent = spell_checker.check("외안되는거야 나는 된다고생각했어").checked
spelled_sent
```

'왜 안되는 거야 나는 된다고 생각했어'

Tokenizer 변경

Mecab, okt, Khaiii



전처리 변경 후 모델 성능 개선 x

3.3 Deep learning

3.3.3 RNN & LSTM

D. 원인분석 2 – 딥러닝 깊이의 대한 문제 접근

RNN + MLP
정확도 0.38

```
model = Sequential()
model.add(Embedding(vocab_size, 256))
model.add(SimpleRNN(256))
model.add(Dense(64, activation = 'relu'))
model.add(Dense(6, activation = 'softmax'))

es = EarlyStopping(monitor = 'val_loss', mode = 'min', verbose = 1, patience = 4)
mc = ModelCheckpoint('best_model.h5', monitor = 'val_acc', mode = 'max', verbose = 1, save_best_only = True)
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['acc'])
history = model.fit(X_train, y_train, epochs=20, callbacks=[es], batch_size=20, validation_split=0.2, shuffle = True)

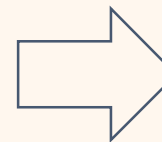
Epoch 1/20
457/457 [=====] - 8s 16ms/step - loss: 1.4988 - acc: 0.4018 - val_loss: 1.4278 - val_acc: 0.4151
Epoch 2/20
457/457 [=====] - 7s 15ms/step - loss: 1.2245 - acc: 0.5470 - val_loss: 1.5801 - val_acc: 0.3958
Epoch 3/20
457/457 [=====] - 7s 15ms/step - loss: 0.7065 - acc: 0.7594 - val_loss: 2.0432 - val_acc: 0.3380
Epoch 4/20
457/457 [=====] - 7s 15ms/step - loss: 0.3793 - acc: 0.8765 - val_loss: 2.3495 - val_acc: 0.3227
Epoch 5/20
457/457 [=====] - 7s 15ms/step - loss: 0.2597 - acc: 0.9152 - val_loss: 3.2189 - val_acc: 0.3183
Epoch 00005: early stopping
```

LSTM + LSTM
정확도 0.40

```
model = Sequential()
model.add(Embedding(vocab_size, 256))
model.add(Bidirectional(LSTM(256, return_sequences = True)))
model.add(Bidirectional(LSTM(128)))
model.add(Dense(6, activation = 'softmax'))

es = EarlyStopping(monitor = 'val_loss', mode = 'min', verbose = 1, patience = 4)
mc = ModelCheckpoint('best_model.h5', monitor = 'val_acc', mode = 'max', verbose = 1, save_best_only = True)
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['acc'])
history = model.fit(X_train, y_train, epochs=20, callbacks=[es], batch_size=20, validation_split=0.2, shuffle = True)

Epoch 1/20
457/457 [=====] - 43s 86ms/step - loss: 1.4565 - acc: 0.4184 - val_loss: 1.4201 - val_acc: 0.4553
Epoch 2/20
457/457 [=====] - 38s 84ms/step - loss: 1.1139 - acc: 0.5869 - val_loss: 1.6376 - val_acc: 0.4159
Epoch 3/20
457/457 [=====] - 38s 84ms/step - loss: 0.7413 - acc: 0.7338 - val_loss: 1.8791 - val_acc: 0.4015
Epoch 4/20
457/457 [=====] - 38s 84ms/step - loss: 0.5008 - acc: 0.8275 - val_loss: 2.3962 - val_acc: 0.3888
Epoch 5/20
457/457 [=====] - 38s 84ms/step - loss: 0.3493 - acc: 0.8787 - val_loss: 2.8055 - val_acc: 0.3778
```



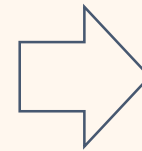
모델의 층을 여러 개 만들어도
모델 성능 개선 x

3.3 Deep learning

3.3.3 RNN & LSTM

E. 원인분석 결과

1) 딥러닝 모델에 평균적으로 필요한 데이터가 10만개 필요
> 현재 Train data 13,000개 (추가적인 라벨링 불가능)



1) 문제점을 해결하기 위해
BERT모델 사용



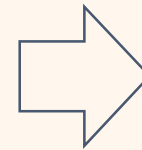
2) 문맥에 따른 감정평가를 하였음

ex) 그녀는 눈물을 흘렸어요.(기쁨/슬픔 문장단위시 판단불가)

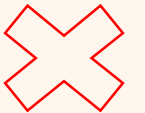
> 앞뒤 문장을 기반으로 판단할 수 있게

문장 순서를 변수로 두었지만,

현재 딥러닝 모델은 문장순서를 변수로 넣지 못함



2) 문제점은 해결 불가능

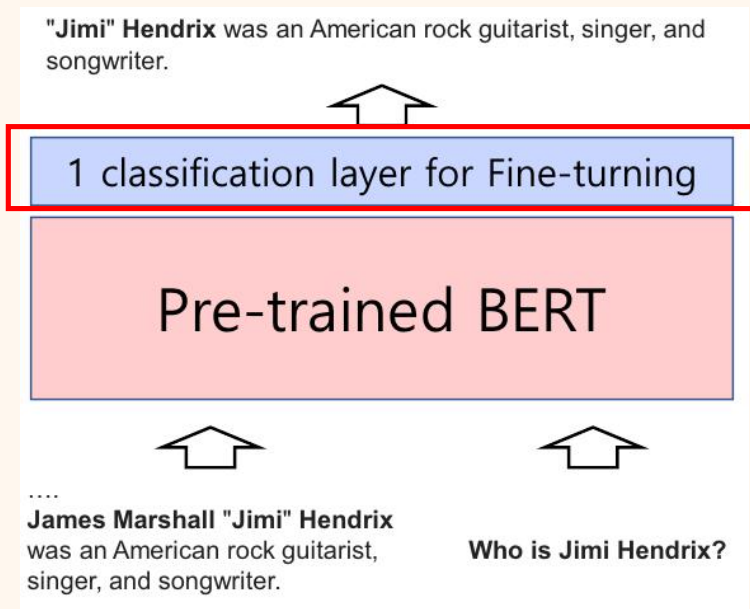


3.3 Deep learning

3.3.4 BERT (Bi-directional Encoder Representations from Transformers)

A. 소개

- BERT 는 2018년도 구글에서 개발한 NLP 사전훈련 기술이며, 특정 분야에 국한 된 기술이 아니라 모든 자연어 처리 분야에서 좋은 성능을 내는 범용 Language Model 입니다.
- Pretrained 된 언어모델 위에 1개의 Classification layer만 부착하여 다양한 NLP Task를 수행 합니다.



```
def create_sentiment_bert():
    model = TFBertModel.from_pretrained("monologg/kobert", from_pt=True)
    token_inputs = tf.keras.layers.Input((SEQ_LEN,), dtype=tf.int32, name='input_word_ids')
    mask_inputs = tf.keras.layers.Input((SEQ_LEN,), dtype=tf.int32, name='input_masks')
    segment_inputs = tf.keras.layers.Input((SEQ_LEN,), dtype=tf.int32, name='input_segment')
    bert_outputs = model([token_inputs, mask_inputs, segment_inputs])
    bert_outputs = bert_outputs[1]
    sentiment_first = tf.keras.layers.Dense(6, activation='softmax', kernel_initializer=tf.keras.initializers.TruncatedNormal(0.02))
    sentiment_model = tf.keras.Model([token_inputs, mask_inputs, segment_inputs], sentiment_first)
    sentiment_model.compile(optimizer=tf.keras.optimizers.Adam(lr=0.00001), loss=tf.keras.losses.SparseCategoricalCrossentropy(from_
metrics=['sparse_categorical_accuracy']))
    return sentiment_model
```

3.3 Deep learning

3.3.4 BERT (Bi-directional Encoder Representations from Transformers)

B. 학습 코퍼스 데이터

- Book Corpus (800M words)
- English Wikipedia (2,500M words without lists, tables and headers)
- 30,000 Token vocabulary

Input-Token (InputLayer)	(None, 128)	0	
Input-Segment (InputLayer)	(None, 128)	0	
Embedding-Token (TokenEmbedding (None, 128, 768), (91812088			Input-Token[0][0]
Embedding-Segment (Embedding) (None, 128, 768)	1536		Input-Segment[0][0]
Embedding-Token-Segment (Add) (None, 128, 768)	0		Embedding-Token[0][0] Embedding-Segment[0][0]
Embedding-Position (PositionEmb (None, 128, 768)	98304		Embedding-Token-Segment[0][0]
Embedding-Dropout (Dropout) (None, 128, 768)	0		Embedding-Position[0][0]
Embedding-Norm (LayerNormalizat (None, 128, 768)	1536		Embedding-Dropout[0][0]
Encoder-1-MultiHeadSelfAttention (None, 128, 768)	2862368		Embedding-Norm[0][0]
Encoder-1-MultiHeadSelfAttention (None, 128, 768)	0		Encoder-1-MultiHeadSelfAttention[
Encoder-1-MultiHeadSelfAttention (None, 128, 768)	0		Embedding-Norm[0][0] Encoder-1-MultiHeadSelfAttention[
Encoder-1-MultiHeadSelfAttention (None, 128, 768)	1536		Encoder-1-MultiHeadSelfAttention[
Encoder-1-FeedForward (FeedForw (None, 128, 768)	4722432		Encoder-1-MultiHeadSelfAttention[
Encoder-1-FeedForward-Dropout ((None, 128, 768)	0		Encoder-1-FeedForward[0][0]
Encoder-1-FeedForward-Add (Add) (None, 128, 768)	0		Encoder-1-MultiHeadSelfAttention- Encoder-1-FeedForward-Dropout[0][
Encoder-1-FeedForward-Norm (Lay (None, 128, 768)	1536		Encoder-1-FeedForward-Add[0][0]
Encoder-2-MultiHeadSelfAttention (None, 128, 768)	2862368		Encoder-1-FeedForward-Norm[0][0]
Encoder-2-MultiHeadSelfAttention (None, 128, 768)	0		Encoder-2-MultiHeadSelfAttention[
Encoder-2-MultiHeadSelfAttention (None, 128, 768)	0		Encoder-1-FeedForward-Norm[0][0] Encoder-2-MultiHeadSelfAttention[
Encoder-2-MultiHeadSelfAttention (None, 128, 768)	1536		Encoder-2-MultiHeadSelfAttention[
Encoder-2-FeedForward (FeedForw (None, 128, 768)	4722432		Encoder-2-MultiHeadSelfAttention[
Encoder-2-FeedForward-Dropout ((None, 128, 768)	0		Encoder-2-FeedForward[0][0]
Encoder-2-FeedForward-Add (Add) (None, 128, 768)	0		Encoder-2-MultiHeadSelfAttention- Encoder-2-FeedForward-Dropout[0][
Encoder-2-FeedForward-Norm (Lay (None, 128, 768)	1536		Encoder-2-FeedForward-Add[0][0]
Encoder-3-MultiHeadSelfAttention (None, 128, 768)	2862368		Encoder-2-FeedForward-Norm[0][0]
Encoder-3-MultiHeadSelfAttention (None, 128, 768)	0		Encoder-3-MultiHeadSelfAttention[
Encoder-3-MultiHeadSelfAttention (None, 128, 768)	0		Encoder-2-FeedForward-Norm[0][0] Encoder-3-MultiHeadSelfAttention[
Encoder-3-MultiHeadSelfAttention (None, 128, 768)	1536		Encoder-3-MultiHeadSelfAttention[
Encoder-3-FeedForward (FeedForw (None, 128, 768)	4722432		Encoder-3-MultiHeadSelfAttention[
Encoder-3-FeedForward-Dropout ((None, 128, 768)	0		Encoder-3-FeedForward[0][0]
Encoder-3-FeedForward-Add (Add) (None, 128, 768)	0		Encoder-3-MultiHeadSelfAttention- Encoder-3-FeedForward-Dropout[0][
Encoder-3-FeedForward-Norm (Lay (None, 128, 768)	1536		Encoder-3-FeedForward-Add[0][0]
Encoder-4-MultiHeadSelfAttention (None, 128, 768)	2862368		Encoder-3-FeedForward-Norm[0][0]
Encoder-4-MultiHeadSelfAttention (None, 128, 768)	0		Encoder-4-MultiHeadSelfAttention[

3.3 Deep learning

3.3.5 KoBERT

A. 소개

- KoBERT는 SKT의 T-Brain 에서 기존 BERT의 한국어 성능 한계를 극복하기 위해 개발 된 모델이며, 위키피디아 뉴스 등에서 수집한 한국어 문장으로 학습한 한국형 모델이다.
- 영어는 띄어쓰기를 기준으로 Tokenizing이 가능하지만 한국어는 형태소 기준으로 Tokenizing이 이루어져야한다.
- KoBERT SKT Brain 제공 기준 : 한국어 위키 (문장5M, 단어54M) 기반 KOBERT 구축

B. Tokenizing 비교 (BERT vs KoBERT)

ex) 보는 내내 그대로 들어 맞는 예측 카리스마 없는 악역

BERT

['보', '##는', '##내', '##내', '그대로', '들어', '##맞', '##는', '예', '##측', '카', '##리스', '##마', '없는', '악', '##역']

KoBERT

['_보는', '내', '내', '_그대로', '_들어', '맞', '는', '_예측', '_카리스마', '_없는', '_악', '역']

3.3 Deep learning

3.3.5 KoBERT

C. HuggingFace

- HuggingFace는 자연어 처리 인공지능 모델에서, BERT 모델 같은 트랜스포머 모델들을 쉽게 다룰 수 있게 해주는 패키지입니다.

D. Pettrained 모델

- Monologg / KoBERT (HuggingFace KoBERT 모델 중에 성능 1위)

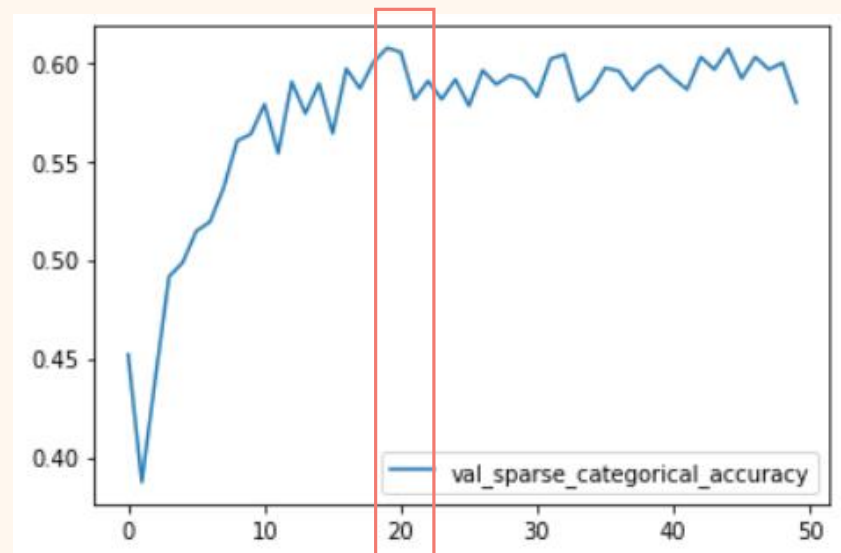
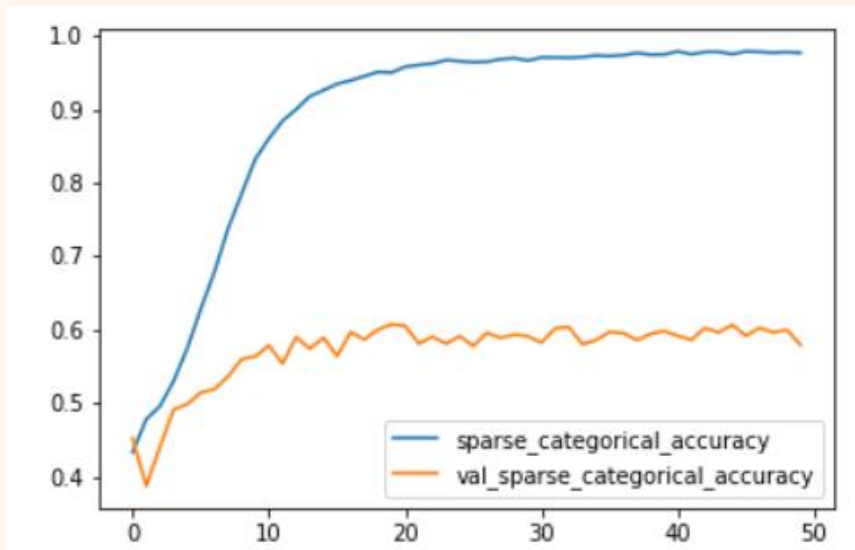
```
def create_sentiment_bert():  
    model = TFBertModel.from_pretrained("monologg/kobert", from_pt=True)  
    token_inputs = tf.keras.layers.Input((SEQ_LEN,), dtype=tf.int32, name='input_word_ids')  
    mask_inputs = tf.keras.layers.Input((SEQ_LEN,), dtype=tf.int32, name='input_masks')  
    segment_inputs = tf.keras.layers.Input((SEQ_LEN,), dtype=tf.int32, name='input_segment')  
    bert_outputs = model([token_inputs, mask_inputs, segment_inputs])  
    bert_outputs = bert_outputs[1]  
    sentiment_first = tf.keras.layers.Dense(6, activation='softmax', kernel_initializer=tf.keras.initializers.TruncatedNormal(0.02))  
    sentiment_model = tf.keras.Model([token_inputs, mask_inputs, segment_inputs], sentiment_first)  
    sentiment_model.compile(optimizer=tf.keras.optimizers.Adam(lr=0.00001), loss=tf.keras.losses.SparseCategoricalCrossentropy(from_=  
        metrics=['sparse_categorical_accuracy']))  
    return sentiment_model
```

3.3 Deep learning

3.3.5 KoBERT

E. 모델선택

- 50epoch 결과 20epoch부터 과적합이 발생하는 것으로 판단
- 20epoch 모델을 선정하여 서비스에 사용



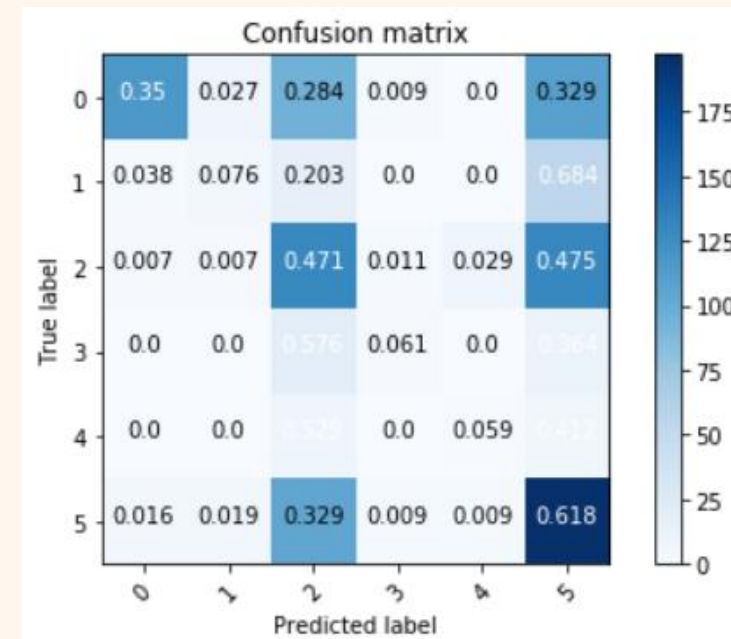
3.3 Deep learning

3.3.5 KoBERT

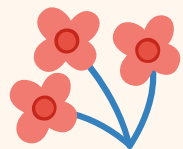
F. 모델특징

- 2(긴장), 3(최고조), 4(슬픔) 감정이 혼동되는 경향이 있음

회초리로 때리면 간지럽다고 깔깔대고 절긱공이 내리치면 시원하다 깔깔댁니다
5 0.8603309
그러던 어느 날 마을에 흉년이 들어 세 형제는 쫄쫄 굶고 나무뿌리 캐 먹으러 산 위로 올라갔습니다
2 0.9418866
“사람들 어찌 사나 어디 한 번 둘러볼까?” 눈 밝은 망이가 어허 쫓쫓 허를 참니다
5 0.9921685
논바닥이 찌억 갈라지고 나무껍질이 흘러당 벗겨져 있었어요
5 0.9389751
어른들은 굶어서 눈이 움푹 들어가고 아이들은 배고파 울고불고 소리칩니다
4 0.54572535
“으앙 배고파~” 그중에서 어디서 밥을 짓나 하얀 연기가 모락모락 피어 오르고 있었어요
5 0.9945128
“마을 사또 사는 곳에 잔치가 열렸구나” 곡간에는 쌀가마니가 산처럼 쌓여 있고 대청마루 상 위에는
5 0.99326754
맘 흘러 거둔 곡식을 사또 혼자 차지한 것입니다
5 0.98731226
세 형제는 그 모습을 보고 화가 나서 이리해 볼까, 저리해 볼까 고민을 했습니다
2 0.8925357
밤은 깊어 가는데 둘째가 쌀가마니 이고 갑니다
5 0.9154151
이 집 저 집 다니며 배불리 먹으라고 세 형제는 밤새도록 흥길동 노릇을 합니다
5 0.69529355
“야단났네! 야단났어! 곡간이 텅 비었네!” 자신의 곡간이 도둑이 든 것을 안 사또는 소리칩니다
2 0.93586856
대신 집집마다 밥을 짓는 연기로 하얀 연기가 나오고 있었습니다
2 0.7532437
욕심 많은 사또는 화가 났습니다
2 0.95675033

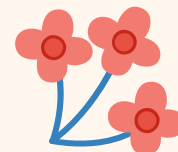


Fun Facts



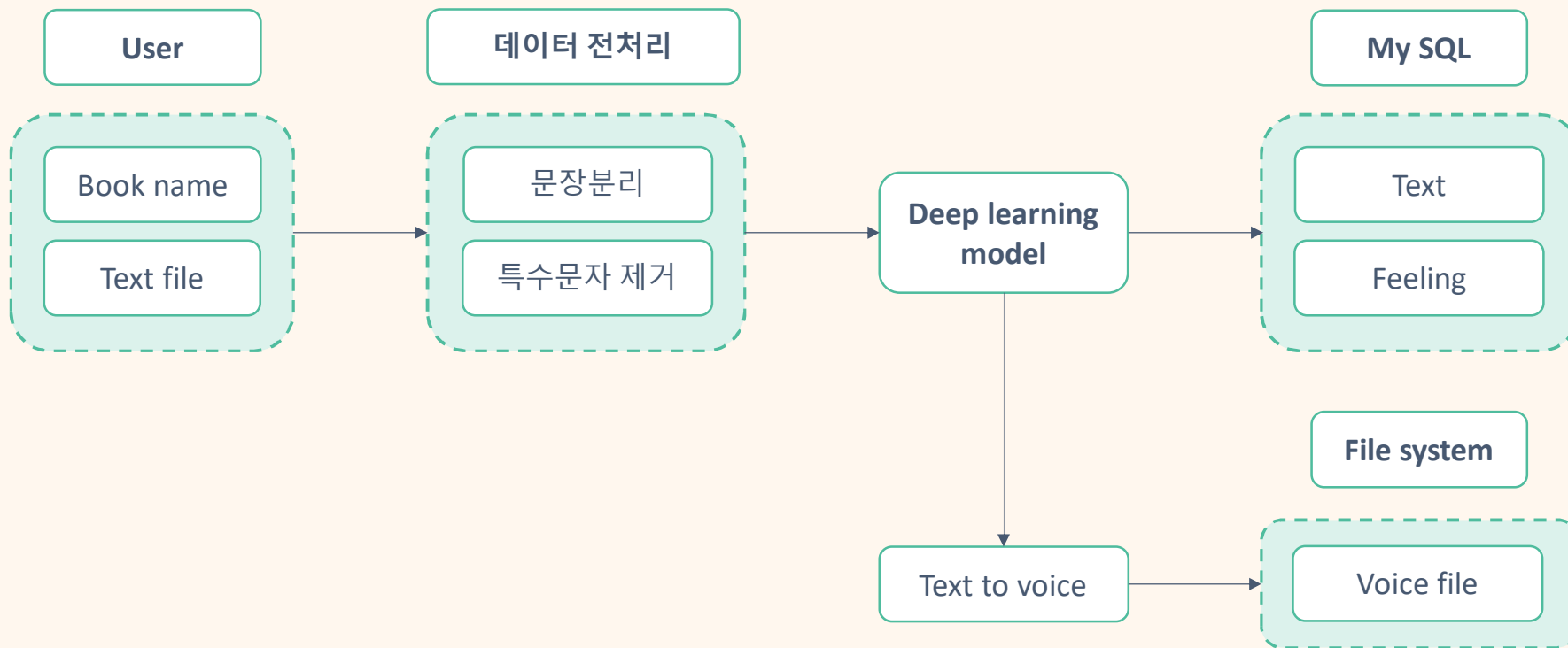
13,059
(학습 데이터 개수)

0.5946%
Accuracy (KoBERT)



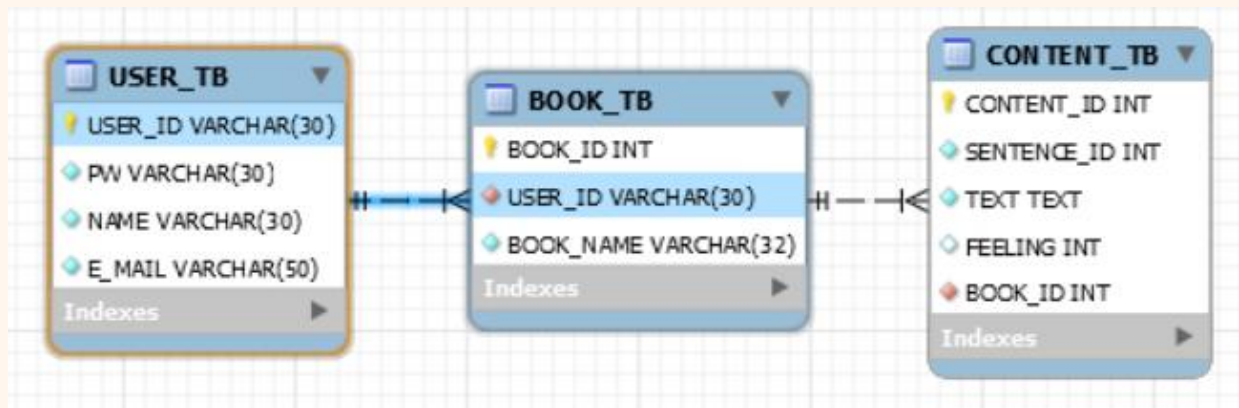
3.4 Web service

3.4.1 시스템 구성도



3.4 Web service

3.4.2 ERD (Entity Relationship Diagram)



3.4 Web service

TTS

카카오 TTS 서비스 활용

음성 합성 Text-to-Speech system

텍스트를 음성으로 변환하는 기술입니다.

음성 안내, 책 읽기 서비스 등 다양한 분야에서 활용할 수 있습니다.

여성/남성 밝은 대화체 선정

Attribute	Possible Value
name	<ul style="list-style-type: none">• WOMAN_READ_CALM : 여성 차분한 낭독체 (default)• MAN_READ_CALM : 남성 차분한 낭독체• WOMAN_DIALOG_BRIGHT : 여성 밝은 대화체• MAN_DIALOG_BRIGHT : 남성 밝은 대화체

3.4 Web service

TTS

<SSML 태그>

<speaking>

<voice name="WOMAN_DIALOG_BRIGHT">

<prosody rate="slow" volume="loud"><audio src="http://116.44.136.77/5janjan.mp3" clipBegin="3s" clipEnd="6s"/>

옛날 시골마을에 형제들이 살았어요

</prosody>

</voice>

</speaking>

3.4 Web service

```
def get_tts_text(text_feeling_lists):
    book_text = "<speak>"
    pre = 9999
    b1,b2,b3,b4,b5 = 3,3,3,3,3

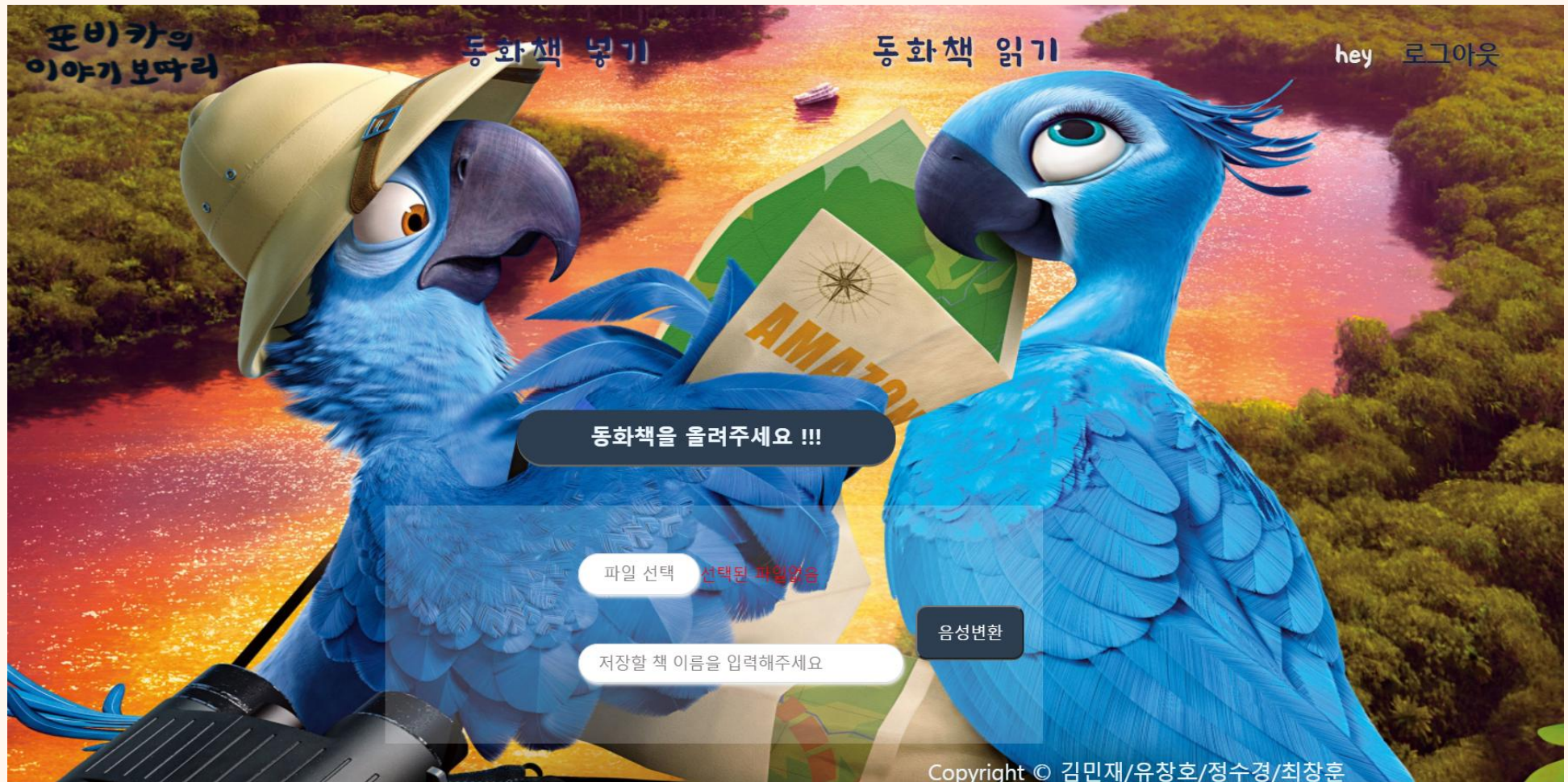
    for text_feeling_list in text_feeling_lists:
        text = text_feeling_list[0]
        sentiment = text_feeling_list[1]
        if sentiment == 0:
            bgm = ''
        if sentiment == 1:
            bgm = '<audio src="http://116.44.136.77/1peace.mp3" clipBegin="{s}" clipEnd="{s}">'.format(str(b1),
                                                                                               str(b1 + 3))
            b1 += 3
        if sentiment == 2:
            bgm = '<audio src="http://116.44.136.77/2angry.mp3" clipBegin="{s}" clipEnd="{s}">'.format(str(b2),
                                                                                               str(b2 + 3))
            b2 += 3
        if sentiment == 3:
            bgm = '<audio src="http://116.44.136.77/3veryangry.mp3" clipBegin="{s}" clipEnd="{s}">'.format(str(b3),
                                                                                               str(
                                                                                                   b3 + 3))
            b3 += 3
        if sentiment == 4:
            bgm = '<audio src="http://116.44.136.77/4sad.mp3" clipBegin="{s}" clipEnd="{s}">'.format(str(b4),
                                                                                               str(b4 + 3))
```

Ssml 태그 생성

```
<speak><voice name="WOMAN_DIALOG_BRIGHT"><prosody rate="slow" volume="loud"><audio src="http://116.44.136.77/5janjan.mp3" clipBegin="3s" clipEnd="6s">옛날 서울마을에 별난 제주 세 형제
가 살았습니다</prosody></voice><voice name="WOMAN_DIALOG_BRIGHT"><prosody rate="slow" volume="loud"><audio src="http://116.44.136.77/5janjan.mp3" clipBegin="8s" clipEnd="11s"> 눈이 밝
은 만이이 별별 것을 다 봅니다</prosody></voice><voice name="WOMAN_DIALOG_BRIGHT"><prosody rate="slow" volume="loud"><audio src="http://116.44.136.77/5janjan.mp3" clipBegin="13s" clipEn
d="16s"> 동해는 천하장사 바닷물도 번쩍번쩍</prosody></voice><voice name="WOMAN_DIALOG_BRIGHT"><prosody rate="slow" volume="loud"><audio src="http://116.44.136.77/5janjan.mp3" clipBegin
n="18s" clipEnd="21s"> 개구쟁이 막내는 희한한 제주인데 매 맞는 제주입니다</prosody></voice><voice name="WOMAN_DIALOG_BRIGHT"><prosody rate="slow" volume="loud"><audio src="http://116.
44.136.77/5janjan.mp3" clipBegin="23s" clipEnd="26s"> 회초리로 때리면 잔치럽다고 깔깔대고 절굿공이 내리치면 시원하다 깔깔댁니다</prosody></voice><voice name="WOMAN_DIALOG_BRIGHT"><pro
sody rate="slow" volume="loud"><audio src="http://116.44.136.77/2angry.mp3" clipBegin="3s" clipEnd="6s"> 그러던 어느 날 마을에 흉년이 들어 세 형제는 썰뽕 굶고 나무뿌리 캐 먹으러 산 위
로 올라갔습니다</prosody></voice><voice name="WOMAN_DIALOG_BRIGHT"><prosody rate="slow" volume="loud"><audio src="http://116.44.136.77/5janjan.mp3" clipBegin="28s" clipEnd="31s"> "사
람들 어찌 사나 어디 한 번 둘러볼까?" 는 밝은 만이가 어허 쫓듯 허를 참니다</prosody></voice><voice name="WOMAN_DIALOG_BRIGHT"><prosody rate="slow" volume="loud"><audio src="http://116.4
4.136.77/5janjan.mp3" clipBegin="33s" clipEnd="36s"> 논밭이 썩어 갈라지고 나무껍질이 흘러당 벗겨져 있었어요</prosody></voice><voice name="WOMAN_DIALOG_BRIGHT"><prosody rate="slow" v
olume="loud"><audio src="http://116.44.136.77/4sad.mp3" clipBegin="3s" clipEnd="6s"> 어른들은 굶어서 눈이 움푹 들어가고 아이들은 배고파 울고불고 소리칩니다</prosody></voice><voice nam
e="WOMAN_DIALOG_BRIGHT"><prosody rate="slow" volume="loud"><audio src="http://116.44.136.77/5janjan.mp3" clipBegin="38s" clipEnd="41s"> "으랏 배고파~" 그중에서 어디서 밥을 찾아 하얀 연
기가 모락모락 피어 오르고 있었어요</prosody></voice><voice name="WOMAN_DIALOG_BRIGHT"><prosody rate="slow" volume="loud"><audio src="http://116.44.136.77/5janjan.mp3" clipBegin="43s" c
lipEnd="46s"> "마을 사또 사는 곳에 잔치가 열렸구나" 곡간에는 쌀가마니가 산처럼 쌓여 있고 대청마루 상 위에는 온갖 음식이 가득히 차려져 있었어요</prosody></voice><voice name="WOMAN_DIAL
OG_BRIGHT"><prosody rate="slow" volume="loud"><audio src="http://116.44.136.77/5janjan.mp3" clipBegin="48s" clipEnd="51s"> 땀 흘려 거둔 곡식을 사또 혼자 차지한 것입니다</prosody></voi
ce><voice name="WOMAN_DIALOG_BRIGHT"><prosody rate="slow" volume="loud"><audio src="http://116.44.136.77/2angry.mp3" clipBegin="6s" clipEnd="9s"> 세 형제는 그 모습을 보고 화가 나서 이
리해 볼까, 저리해 볼까 고민을 했습니다</prosody></voice><voice name="WOMAN_DIALOG_BRIGHT"><prosody rate="slow" volume="loud"><audio src="http://116.44.136.77/5janjan.mp3" clipBegin="53
s" clipEnd="56s"> 밝은 길어 가는데 동해가 쌀가마를 이고 갑니다</prosody></voice><voice name="WOMAN_DIALOG_BRIGHT"><prosody rate="slow" volume="loud"><audio src="http://116.44.136.77/5
janjan.mp3" clipBegin="58s" clipEnd="61s"> 이 집 저 집 다니며 배불리 먹으라고 세 형제는 밤새도록 흥얼흥 노래를 합니다</prosody></voice><voice name="WOMAN_DIALOG_BRIGHT"><prosody rate=
"slow" volume="loud"><audio src="http://116.44.136.77/2angry.mp3" clipBegin="9s" clipEnd="12s"> "야단났네! 야단났네! 곡간이 텅 비었네!" 자신의 곡간이 도둑이 든 것을 안 사또는 소리칩니
다</prosody></voice><voice name="WOMAN_DIALOG_BRIGHT"><prosody rate="slow" volume="loud"><audio src="http://116.44.136.77/2angry.mp3" clipBegin="12s" clipEnd="15s"> 대신 집집마다 밥을
있는 연기로 하얀 연기가 나오고 있었습니다</prosody></voice><voice name="WOMAN_DIALOG_BRIGHT"><prosody rate="slow" volume="loud"><audio src="http://116.44.136.77/2angry.mp3" clipBegin=
"15s" clipEnd="18s"> 육식 많은 사또는 화가 났습니다</prosody></voice><voice name="WOMAN_DIALOG_BRIGHT"><prosody rate="slow" volume="loud"><audio src="http://116.44.136.77/2angry.mp3"
```

Ssml 태그 기반 voice 파일 생성

3.4 Web service



3.4 Web service

포비카의 이야기 보따리

동화책 넣기

동화책 읽기

포비카 로그아웃

0:00 / 1:33

남자목소리

여자목소리

나의 동화책 모음

별난재주삼형제

밥만 먹는 밥벌레 장군

씨 뿌리는 강아지

옛날 시골마을에 별난 재주 세 형제가 살았습니다 눈이 밝은 만이는 별별 것을 다 봅니다 둘째는 천하장사 바윗돌도 번쩍번쩍 개구쟁이 막내는 희한한 재주인데 매 맞는 재주입니다 회초리로 때리면 간지럽다고 깔깔대고 절굿공이 내리치면 시원하다 깔깔댁니다 그러던 어느 날 마을에 흉년이 들어 세 형제는 쪼쪼 굶고 나무뿌리 캐 먹으러 산 위로 올라갔습니다 "사람들 어찌 사나 어디 한 번 돌려볼까?" 눈 밝은 만이가 어허 쫓쫓 허를 찔니다 논바닥이 찌억 갈라지고 나무껍질이 홀라당 벗겨져 있었어요 어른들은 굶어서 눈이 움푹 들어가고 아이들은 배고파 울고울고 소리칩니다 "으앙 배고파~" 그중에서 어디선가 밥을 짓는 하얀 연기가 모락모락 피어 오르고 있었어요 "마을 사또 사는 곳에 잔치가 열렸구나" 곡간에는 쌀가마니가 산처럼 쌓여 있고 대청마루 상 위에는 온갖 음식이 가득히 차려져 있었어요 땀 흘려 거둔 곡식을 사또 혼자 차지한 것입니다 세 형제는 그 모습을 보고 화가 나서 이리해 볼까, 저리해 볼까 고민을 했습니다 밤은 깊어 가는데 둘째가 쌀가마니를 이고 갑니다 이 집저 집 다니며 배불리 먹으라고 세 형제는 밤새도록 흥길동 노릇을 합니다 "야 단났네! 야 단났어! 곡간이 텅 비었네!" 자신의 곡간이 도둑이 든 것을 안 사또는 소리칩니다 대신 집집마다 밥을 짓는 연기로 하얀 연기가 나오고 있었습

Copyright © 김민재/유



4. 프로젝트 후기

1.1 번외편

1.2 프로젝트후기

4.1 번외편

4.1.1 BERT + Saltlux API (감성분류 라벨링)


A. 문제접근

- 데이터 라벨링 시간과 일관성 한계가 있음.
- 감성분류 API 제공해주는 모델을 통한 데이터 라벨링 작업을 하면 BERT 의 성능은 개선 될 것인가?

B. 실험과정

- 데이터 라벨링 API 작업 (JSON / 감성분류 8가지 / 문장 개수 : 15,914 개)

감성 지능

 텍스트 감정 분석

주어진 한국어 문장, 문서에 대한 감성과 다중 감정을 추천합니다.

[업그레이드 서비스 신청](#)  

특징	활용 사례/예시	개발 가이드	테스트
<div>서비스 개요</div> <p>감정분석은 문장에서 표현될 수 있는 발화자의 8가지 감정(기쁨, 신뢰, 공포, 놀라움, 슬픔, 혐오, 분노, 기대)을 다중으로 분석합니다. 다중 감정 분석을 위해 BI-LSTM 기반 Attention Network 모델을 채택하고 있습니다.</p> <p>감성분석은 문장에서 표현될 수 있는 발화자의 극성(긍정, 부정, 중립)을 분석합니다. GPU 오버헤드를 줄이고 추론속도 향상을 위해 Kernel Fusion과 모델 압축(양자화)이 적용된 BERT 모델을 채택하고 있습니다.</p>			

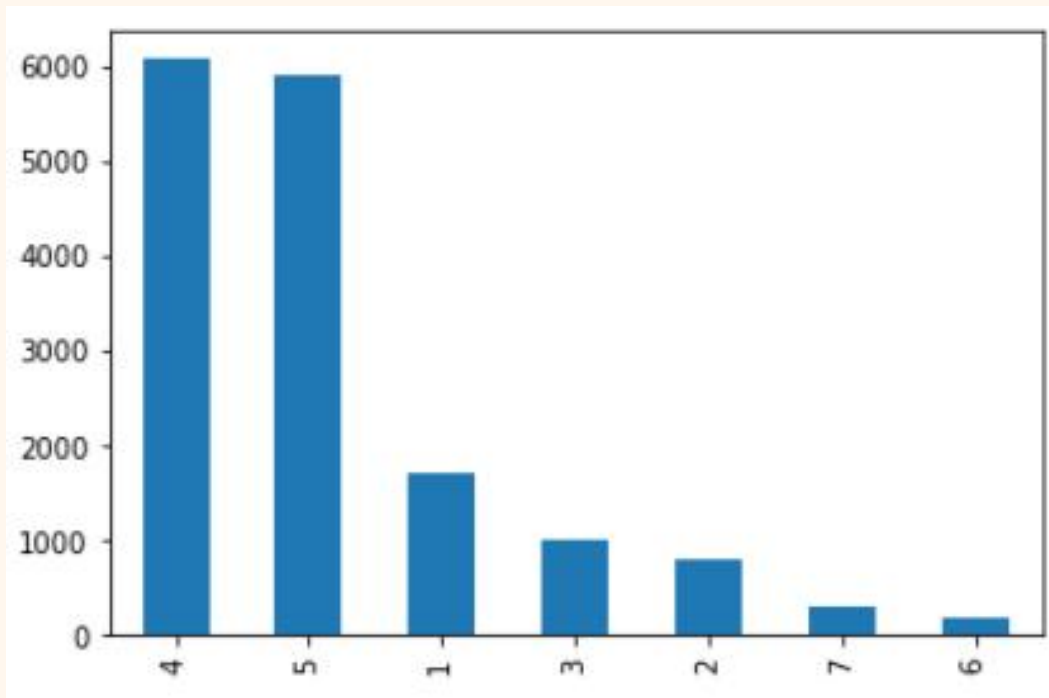
```
{
  "query": "바퀴벌레는 무섭습니다.",
  "type": "감정분석",
  "Result": [
    [
      0.7217257022857666,
      "혐오"
    ]
  ]
}
```

4.1 번외편

4.1.1 BERT + Saltlux API (감성분류 라벨링)

c. 데이터 분포

- 신뢰 (4) > 혐오 (5) > 분노 (1) > 기대 (3) > 공포 (2) > 슬픔 (7) > 기쁨 (6)



4.1 번외편

4.1.1 BERT + Saltlux API (감성분류 라벨링)

C. 모델 학습

BERT
정확도 0.65

```
model_name = "tf2_bert_naver_movie"

# overfitting을 막기 위한 earllystop 추가
earlystop_callback = EarlyStopping(monitor='val_accuracy', min_delta=0.0001, patience=4)

# min_delta: the threshold that triggers the termination (acc should at least improve 0.0001)
# patience: no improvment epochs (patience = 1, 1번 이상 상승이 없으면 종료)

checkpoint_path = os.path.join(DATA_OUT_PATH, model_name, 'weights.h5')
checkpoint_dir = os.path.dirname(checkpoint_path)

# Create path if exists
if os.path.exists(checkpoint_dir):
    print("{} -- Folder already exists {}".format(checkpoint_dir))
else:
    os.makedirs(checkpoint_dir, exist_ok=True)
    print("{} -- Folder create complete {}".format(checkpoint_dir))

cp_callback = ModelCheckpoint(
    checkpoint_path, monitor='val_accuracy', verbose=2, save_best_only=True, save_weights_only=True)

# 학습과 eval 시작
history = cls_model.fit(x_train,
                        y_train,
                        epochs=NUM_EPOCHS,
                        shuffle = True,
                        batch_size=BATCH_SIZE,
                        validation_data=(x_val, y_val),
                        callbacks=[earlystop_callback, cp_callback])

print(history.history)
```

```
results = cls_model.evaluate(test_movie_inputs, test_data_labels, batch_size=16)
print("test loss, test acc: ", results)
```

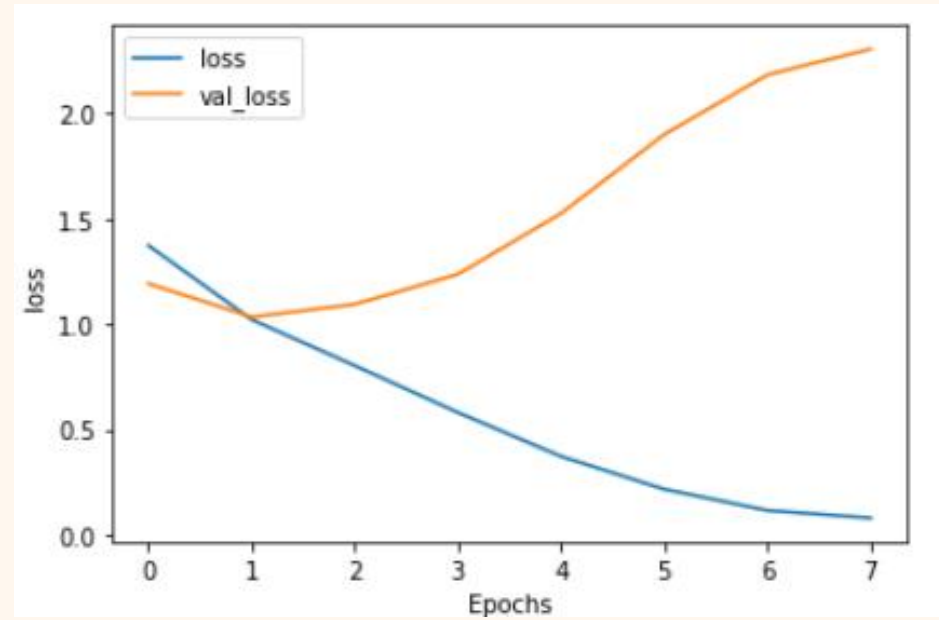
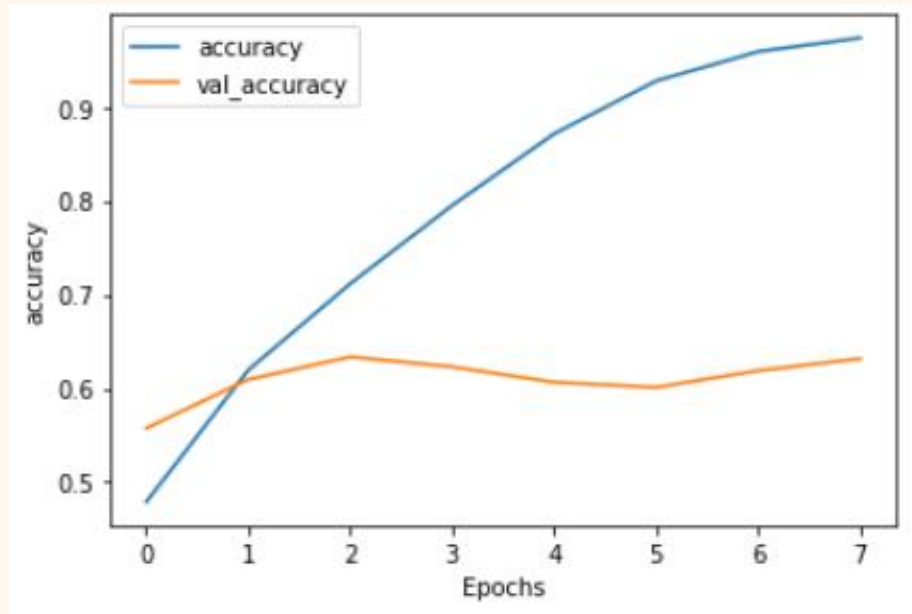
```
125/125 [=====] - 8s 67ms/step - loss: 2.1596 - accuracy: 0.6500
test loss, test acc: [2.1595633029937744, 0.6499999761581421]
```

4.1 번외편

4.1.1 BERT + Saltlux API (감성분류 라벨링)

D. 평가

- 과적합 발생

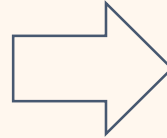
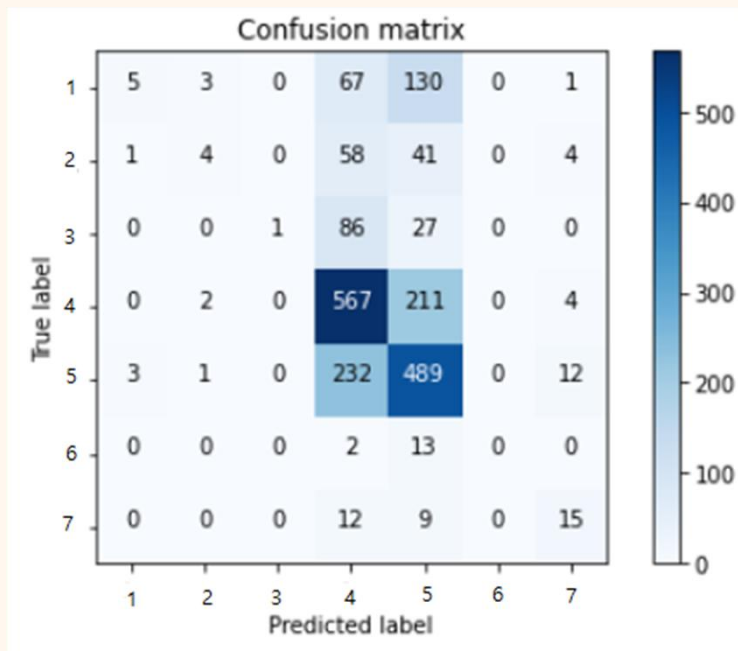


4.1 번외편

4.1.1 BERT + Saltlux API (감성분류 라벨링)

D. 평가

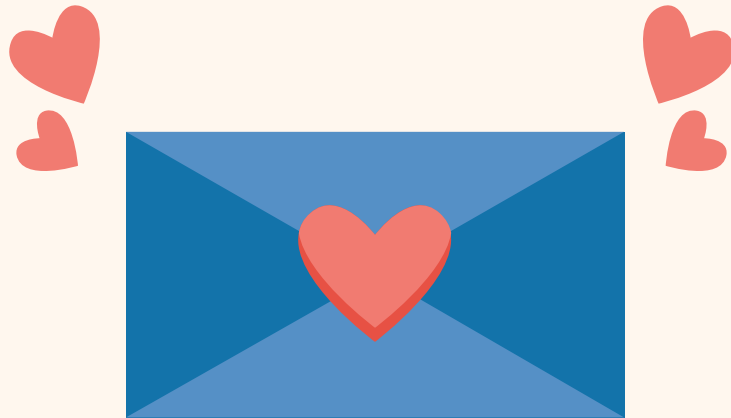
- 데이터의 불균형 발생

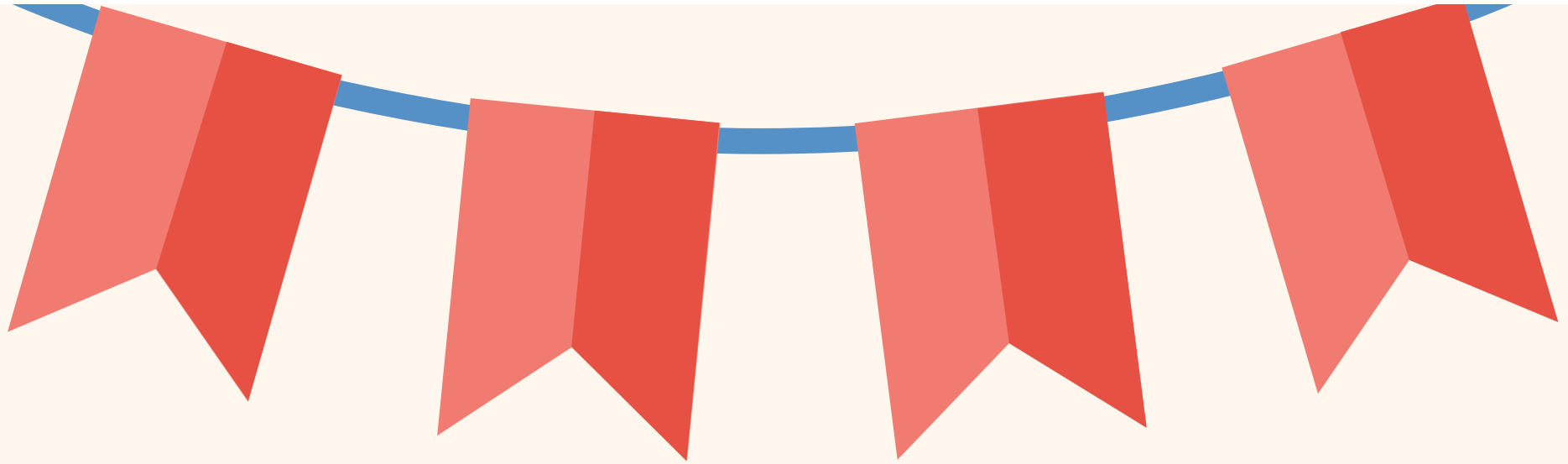


데이터의 분포가 특정감정에 몰려 있어
모델로 적합하지 않음

4.2 프로젝트 후기

“사랑스러운 팀원들과 누구 하나 기술적인 공백 없이 함께 성장할 수 있는 프로젝트였습니다.”





Thanks!

Any questions?

Reference

◆ BGM

- 평화 / Love Story / Artist: Andrew_G

Link: https://www.youtube.com/watch?v=pBT5Rxt3gss&ab_channel=AndrewGIBackgroundMusicforVideos

- 긴장 / Dream Of You / 김성원

Link: <https://gongu.copyright.or.kr/gongu/wrt/wrt/view.do?wrtSn=13073697&menuNo=200020>

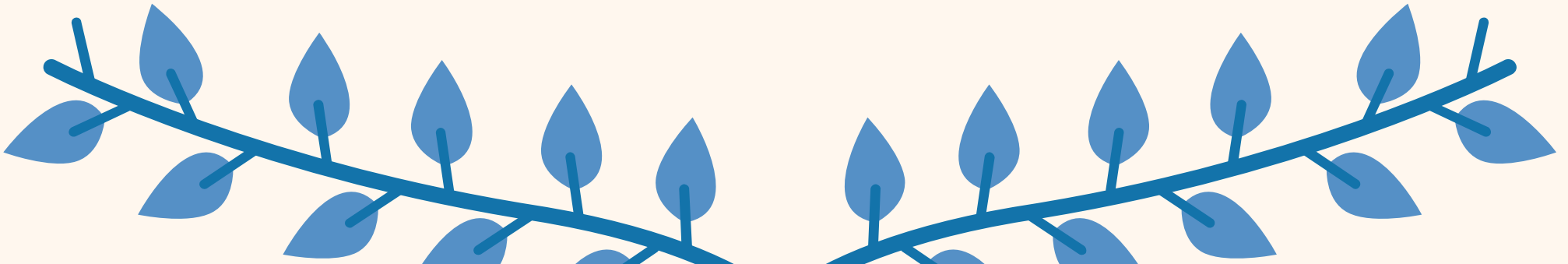
- 분노 / Mist / 이어브로

Link: <http://earbro.com/goods/view?no=1271&setMode=pc>

- 슬픔 / Epic, - Background, piano, Sad) theme - Film scores

Link: https://www.youtube.com/watch?v=o_2XYtojmbI&ab_channel=FesliyanStudiosBackground-MusicFesliyanStudiosBackgroundMusic

- 잔잔 / Ending / SellBuyMusic / Link: <https://www.sellbuymusic.com/musicDetail/9257>



Reference

◆ Labeling (감정분석)

- Saltlux 감정분석 API / https://www.saltlux.ai/portal/api_detail?id=category14

◆ 음성합성 API

- 카카오 Speech API (TTS) / <https://speech-api.kakao.com>

◆ 학습 데이터

- 청와대 전래동화 / 크롤링 / http://18children.president.pa.go.kr/mobile/our_space/fairy_tales.php
- 동화전집 / 다운로드 / <https://m.blog.naver.com/PostList.nhn?blogId=osy2201>

