Muchine learning Project
by Team 먼지점프

KoNLPy로
영화 리뷰 평점 예측하기

팀장 유민석
팀원 구은혜 박지홍 엄예찬 임예지 최서연

개발일정

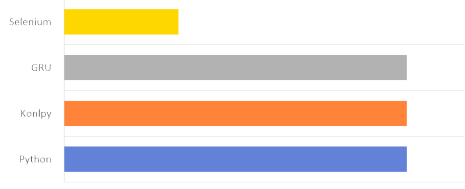
2022-01-10 : 데이터 수집 및 전처리
2022-01-10 ~ 2022-01-11 : 모델 구현 및 학습
2022-01-12 ~ 2022-01-14 : 모델 평가 및 최적화
2022-01-14 : 결과 시각화 및 PPT 작성
2022-01-15 : 최종 점검 및 결과물 제출

Work dataset

네이버영화 리뷰데이터

다양한 형태소 분석기를 사용하여 영화 리뷰데이터를 학습시켜
리뷰의 평점을 예측해보자

리뷰의 평점을 1~10점까지 예측 가능하도록 설계



CONTENTS

01 기획

- 목표
- 프로젝트 일정
- 개발 도구/라이브러리
- 역할 분담

02 개발 및 구현

- 데이터 수집
- 데이터 전처리
- 모델 구현
- 모델 학습

03 성능 테스트

- 모델 평가/개선
- 결과 분석
- 결과 시각화

04 후기 및 아쉬운 점

- 아쉬운 점
- 느낀 점
- Q&A

[기 획]

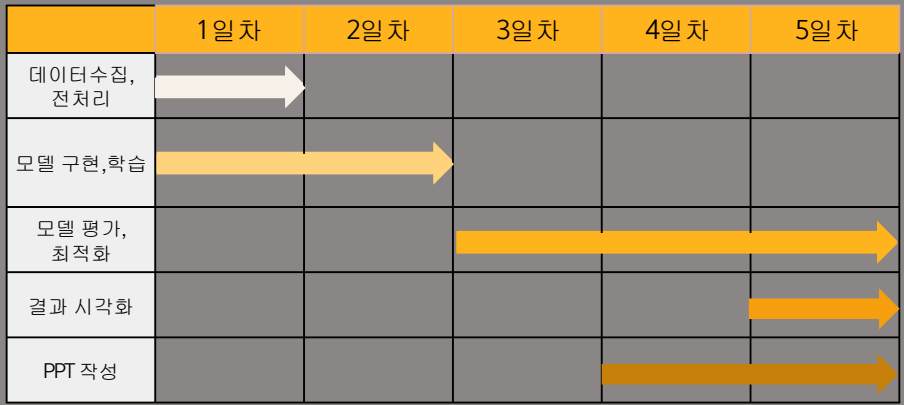
01

목표

박스오피스 TOP100 영화 리뷰 평점 예측하기

02

프로젝트 일정



03

개발 도구 및 라이브러리

Python

Tensor Flow

komoran

04
역할 분담

유민석

- 데이터 수집
- 데이터 전처리
- 모델 구현
- 모델 성능 개선

구은혜

- 데이터 전처리
- 모델 구현
- 모델 성능 개선

박지홍

- 자료 조사
- 데이터 수집
- 모델 구현
- 모델 성능 개선
- PPT 제작

엄예찬

- 데이터 전처리
- 모델 구현
- 모델 성능 개선
- 발표

임예지

- 데이터 전처리
- 모델 구현
- 모델 성능 개선

최서연

- 자료 조사
- 데이터 전처리
- 모델 구현
- 모델 성능 개선

[개발 및 구현]

개발 및 구현

01. 데이터 수집

- 댓글 수집
- 한국어 불용어 수집

02. 데이터 전처리

- 불용어 제거
- 정규식을 이용하여 특수문자 제거
- 중복/공백 댓글, Nan값 삭제
- 토큰 추출하여 인덱싱
- 댓글 길이 패딩

03. 모델 구현

- 데이터 분리
- 각 알고리즘 별 모델

04. 모델 학습

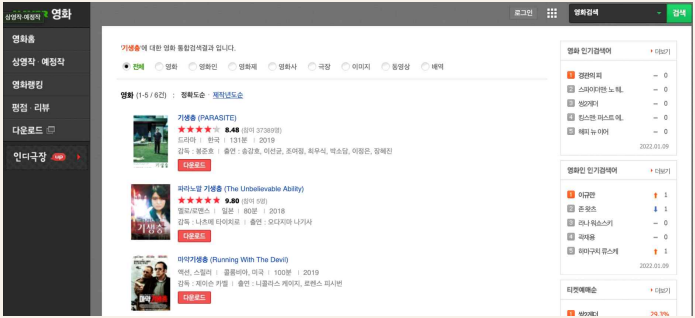
- 각 모델 별 학습 결과 시각화

데이터 수집

- 댓글 수집
- 한국어 불용어 수집

01
데이터 수집

댓글 수집
(selenium을 이용하여 댓글 수집)



네이버 영화에서 리뷰데이터를 크롤링

댓글 수집

(selenium을 이용하여 댓글 수집)

- 박스오피스 TOP100 영화 제목 추출

	A	B	C	D
1	순위	영화명	개봉일	매출액
2	1	명량	2014.7.30	135,748,398,910
3	2	극한직업	2019.1.23	139,647,979,516
4	3	신과함께-죄와 벌	2017.12.20	115,698,654,137
5	4	국제시장	2014.12.17	110,913,469,630
6	5	어벤져스: 엔드게임	2019.4.24	122,182,694,160
7	6	겨울왕국 2	2019.11.21	114,810,421,450
8	7	아바타	2009.12.17	128,447,097,523
9	8	베테랑	2015.8.5	105,168,155,250
10	9	괴물	2006.7.27	0
11	10	도둑들	2012.7.25	93,665,568,500
12	11	7번방의 선물	2013.1.23	91,431,914,670
13	12	암살	2015.7.22	98,463,132,781
14	13	알라딘	2019.5.23	106,983,620,359
15	14	광해, 왕이 된 남자	2012.9.13	88,900,208,769
16	15	왕의 남자	2005.12.29	0
17	16	신과함께-인과 연	2018.8.1	102,666,146,909
18	17	택시운전사	2017.8.2	95,855,737,149
19	18	태극기 휘날리며	2004.2.5	0
20	19	부산행	2016.7.20	93,178,283,048
21	20	해운대	2009.7.22	81,934,638,201

```
xlsx = pd.read_excel('KOBIS_역대_박스오피스_내역.xlsx',engine="openpyxl")

movie = []

# 박스오피스 영화 순위 100
for x in xlsx["영화명"][0:100]:
    movie.append(x)

print(movie)
```

댓글 수집

(selenium을 이용하여 댓글 수집)

- 영화 코드 추출

영화코드2.txt

```
39378
188472
179705
103760
65540
37919
174004
68196
203383
```

```
# 영화 목록

movie_list = open("영화코드.txt","w")

for x in movie:
    driver.get("https://movie.naver.com/movie/search/result.naver?section=movie&quer
    time.sleep(1)
    q = driver.find_element(By.XPATH, "/html/body/div/div[4]/div/div/div/div/div[1]/
    w = q.find_element(By.TAG_NAME,"a")
    e = w.get_attribute("href")
    data2.append(e[53:])
    movie_list.write(str(e[53:])+"₩n") # 문자 있는 텍스트임
    #print(a[53:])

movie_list.close()
```

01
데이터 수집

댓글 수집

(selenium을 이용하여 댓글 수집)

- 셀레니움 이용하여 각 영화별 댓글 100개 추출
- 총 댓글 10,000개의 데이터 파일 생성

```
## 영화 관련 정보 추출
driver = webdriver.Chrome()
data = []
# 내가 읽고 싶은 페이지 수
page = 20

for z in data2:
    for a in range(1,page+1): # 페이지1장 당 20개의 리뷰
        #다음 장 넘기기
        driver.get("https://movie.naver.com/movie/point/af/list.naver?st=mcode&sword
        time.sleep(2)
        # 영화 이름 20개 이름, 별점, 리뷰내용 추출
        b = driver.find_elements(By.CLASS_NAME,"title")
        for x in range(10):
            # 영화 이름
            name = b[x].text.split("₩n")[0]

            # 영화 평점 별점
            num = b[x].text.split("₩n")[2]

            # 영화 평점 내용
            review = b[x].text.split("₩n")[3][:-3]

            data.append(np.asarray((name,num,review)))

data_df = pd.DataFrame(data,columns=['name','num','review'])
```

불용어 수집

- 불용어 TXT 파일 다운
- TXT파일을 리스트로 변환

한국어불용어100.txt

단어	특징	확률
이	VCP	0.018279601
있	VA	0.011699048
하	VV	0.009773658
것	NNB	0.00973315
들	XSN	0.00689824
그	MM	0.005327252
되	VV	0.00361335
수	NNB	0.003473622
이	NP	0.003361203
보	VX	0.003310379
않	VX	0.0029757

```
# 불용어 파일
stop = pd.read_csv("./한국어불용어100.txt",sep="\t",encoding="utf-8")

# 불용어 리스트 생성
stopwords = stop["불용어"].tolist()
```

공통
데이터 전처리

- 불용어 제거
- 정규식 사용해 특수문자 제거
- 중복/공백 댓글, Nan값 삭제
- 토큰 추출하여 인덱싱
- 댓글 길이 패딩

불필요한 데이터 제거

- 형태소 분석기 이용하여 불용어 제거
- 정규식으로 특수문자 제거
- whitespace는 Nan값으로 처리
- Nan 삭제

```
xlsx4 = []
for x in xlsx3["review"]:
    temp = []
    temp = m.morphs(x)

    token = []
    for y in temp:
        if not y in stopwords:
            token.append(y)
    xlsx4.append(token)

## xlsx4에 모든 리뷰들의 정제된단어들이 들어있어
```

```
# 특수문자 제거
df['review']=df['review'].str.replace(r'[^ ㄱ-ㅣ가-힣]+', " ")
df['review'].head()
```

```
# 공백만 남은 데이터 nan으로 처리
df['review'] = df['review'].str.replace('^ +','')
df['review'].replace('',np.nan, inplace = True)
df.isnull().sum()
```

```
# nan 삭제
df = df.dropna(how='any')
len(df)
```

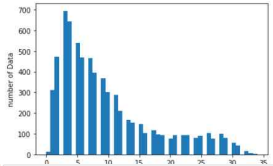
토큰화, 패딩

- 토큰 추출
- 정수 인덱싱
- 리뷰 길이가 서로 다르므로 패딩

```
from keras.preprocessing.text import Tokenizer

tokenizer = Tokenizer()
tokenizer.fit_on_texts(data_X)
sequences = tokenizer.texts_to_sequences(data_X)

word_index = tokenizer.word_index
word_index
```

```
print('리뷰의 최대 길이 : {}'.format(max_len))
print('리뷰의 평균 길이 : {}'.format(sum(map(len, sequences)) / len(sequences)))

plt.hist([len(s) for s in sequences], bins = 50)
plt.xlabel('length of Data')
plt.ylabel('number of Data')
plt.show()
```

```
리뷰의 최대 길이 : 34
리뷰의 평균 길이 : 9.800452872912539
```




```
from keras.datasets import reuters
from keras.models import Sequential
from keras.layers import Dense, LSTM, Embedding
from keras.preprocessing import sequence
from keras.preprocessing.sequence import pad_sequences
from keras.utils import np_utils

X = pad_sequences(sequences, maxlen = max_len)
y = np_utils.to_categorical(data_Y)
```

형태소 분석기

- okt
- Kkma
- komoran
- Mecab

03
okt

1. 명사 추출

```
word_list = []
for i in x_stopword:
    word = []
    for j in i:
        # 명사만 추출
        for x in okt.pos(j,norm= True):
            if(x[1] == 'Noun'):
                word.append((x[0]))
    word_list.append(word)
word_list
```

2. FreqDict 라이브러리 – 빈도수 20회 이하 단어 삭제

```
from nltk import FreqDist
```

```
fd_names = FreqDist(word_freq)
dict_fd = fd_names.items()
```

```
# 빈도수 제거
for dict in dict_fd:
    key = dict[0] # 단어
    val = dict[1] # 횟수
    if val<20:
        for w in word_list:
            if key in w:
                w.remove(key)
print(word_list)
```

3. 모델 구성

```
# GRU 모델 - 유민석님 공유
import re
from tensorflow.keras.datasets import imdb
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, GRU, Embedding


hidden_units = 512


model = Sequential()
model.add(Embedding(vocab_size +1, textlen))
model.add(GRU(hidden_units))
model.add(Dense(10, activation='softmax'))

model.compile(optimizer='rmsprop', loss='categorical_crossentropy', metrics
history = model.fit(train_x, train_y, epochs=15,
                    validation_data=(validation_x,validation_y))
```

03
Kkma

1. 불용어 제거 – Kkma는 시간/메모리 소요가 커 슬라이싱으로 실행

```
x_stopl = []
for sentece in x_data[0:1000]:
    tokenized_sentece = kkma.morphs(sentece)
    stopwords_remove_sentece = [word for word in tokenized_sentece if not word in stopwords]

    x_stopl.append(stopwords_remove_sentece)
x_stopl
```

```
'그럼',
'과',
'뭔',
'해',
'정하',
'나',
'최종',
'보스'],
```

java.lang.OutOfMemoryError: Java heap space 에러로 okt로 진행

2. kkma로 품사 태깅으로 단어 리스트 생성

```
word_list = []
for i in x_stopword:
    word = []
    for j in i:
        for x in kkma.pos(j):
            word.append((x[0]))
    word_list.append(word)
word_list
```

3. 모델 구성

```
# GRU 모델
import re
from tensorflow.keras.datasets import imdb
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, GRU, Embedding

hidden_units = 512

model = Sequential()
model.add(Embedding(vocab_size +1, textlen))
model.add(GRU(hidden_units))
model.add(Dense(10, activation='softmax'))

model.compile(optimizer='rmsprop', loss='sparse_categorical_crossentropy
history = model.fit(train_x, train_y, epochs=15,
                    validation_data=(validation_x,validation_y))
```

03
Mecab

1. 단순 문장 분석 예시

```
from konlpy.tag import Mecab

m = Mecab(dicpath="/usr/local/lib/mecab/dic/mecab-ko-dic")
# 사전 저장 경로에 자신이 mecab-ko-dic를 저장한 위치를 적는다. (default: "/usr
```

executed in 28ms, finished 11:25:35 2022-01-13

```
#m.pos("안녕하세요") # 형태소 분석
#m.morphs("안녕하세요") # 분리
print(m.morphs("안녕하세요"))
print(m.nouns("안녕하세요"))
```

executed in 5ms, finished 11:52:08 2022-01-13

```
['안녕', '하', '세요']
['안녕']
```

2. 댓글 불용어 제거

```
xlsx4 = []
for x in xlsx3["review"]:
    temp = []
    temp = m.nouns(x)

    token = []
    for y in temp:
        if not y in stopwords:
            token.append(y)
    xlsx4.append(token)

## xlsx4에 모든 리뷰들의 정제된단어들이 들어있어
```

3. 모델 구성

```
import re
from tensorflow.keras.datasets import imdb
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, GRU, Embedding

vocab_size = 500
embedding_dim = 30
hidden_units = 512


model = Sequential()
model.add(Embedding(vocab_size+1, embedding_dim))
model.add(GRU(hidden_units))
model.add(Dense(10, activation='softmax'))

model.compile(optimizer='rmsprop', loss='sparse_categorical_crosse
history = model.fit(train_x, train_y, epochs=7,
                    validation_data=(test_x,test_y))
```

03
komoran

1. 객체 생성 & 불용어 리스트 로드

```
from konlpy.tag import Komoran

# Komoran 객체 생성하기
komoran = Komoran()

# 불용어 파일
stop = pd.read_csv('./한국어불용어100.txt', sep = '₩t', encoding = 'utf-8')

# 불용어 리스트 생성
stopwords = stop['불용어'].tolist()
```

2. 댓글 불용어 제거

```
# 명사만 추출하기
k_pos = []

for data in data_X:
    k_pos_data = komoran.nouns(data)

    # 불용어 제거
    result_list = [word for word in k_pos_data if not word in stopwords]

    k_pos.append(result_list)
```

3. pos-tagging 결과(nouns)

역대 대명작이라 자부합니다
진짜 이건 우리나라 영화계에 한 획을 그을 정도에 명작입니다 우리가 역사를 배워야 ...
국뽕 범벅 거품 스크린독점 관객동원
초등 고학년 아들과 진도에 다녀온후 같이 찾는데 정말 감사하고 감동이며 죄송스러운 ...
꼽게 숨을 들이쉬고 후하고 내뱉은 뒤 이 영화를 보아라 다보고 나서 이를 뒤에 ...

```
['역대', '명작', '자부']
['이건', '우리나라', '영화', '획', '명작', '역사', '이유', '이유', '영화']
['국', '뽕', '범벅', '거품', '스크린', '독점', '관객', '동원']
['초등', '학년', '아들', '진도', '후', '감사', '감동', '미음']
['숨', '후', '뒤', '영화', '이를', '뒤', '보', '동안', '조금', '정국장', '뒤', '영화', '일주일', '뒤', '모두', '잠', '새벽', '보', '영화', '이영화', '한처의', '좀', '완벽', '이순신', '장군', '영웅', '하전', '연출', '영화', '감독', '존경']
```

4. 모델 구성

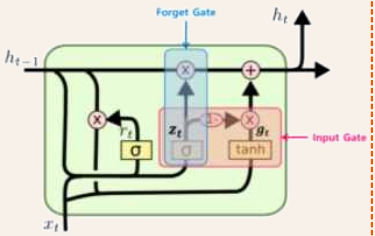
```
model = Sequential()
model.add(Embedding(vocab_size + 1, embedding_dim))
model.add(GRU(hidden_units))
model.add(Dense(10, activation = 'softmax'))

model.compile(optimizer = 'rmsprop', loss = 'categorical_crossentropy', metrics = ['accuracy'])
```

모델

- Naive Bayes
- LSTM
- GRU

Naive Bayes

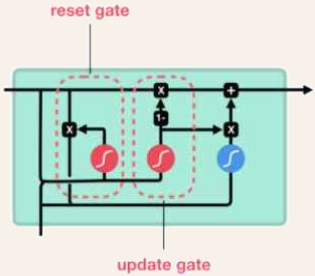
- 데이터가 각 클래스에 속할 확률을 계산하는 조건부 확률 기반의 분류 방법
- 특징이 서로 독립이라는 조건이 필요
- 간단하고 빠른, 효율적인 알고리즘으로, 훈련 시 데이터 크기에 관계 없이 잘 동작함
- 이중 분류(스팸 필터링, 질병 진단), 다중 분류(문서 분류) 등에 쓰임
- 라플라스 스무딩 : 특징의 출현 횟수 초기값을 1부터 시작 → 0을 곱하여 발생하는 문제 해결



LSTM

(Long-Short-Term-Memory)

- RNN과 달리 장기기억이 가능함
- 3개의 gate로 구성되어 있으며 sigmoid함수를 사용함
  forget gate :과거 정보를 잊기 위한 게이트
  input gate : 현재 정보를 기억하기 위한 게이트
  output gate : 최종 결과를 내보내기 위한 게이트
- sigmoid : 0 → 넘기지 않음 / 1 → 모든 것을 넘김



GRU

(Long-Short-Term-Memory)

- LSTM과 비슷한 이유로 만들어졌음
- LSTM보다 간단하게 구성되어 비교적 데이터 처리가 빠름
- LSTM은 Input, Forget, Output 3개의 gate인 반면, GRU는 Update, Reset 2개의 gate로 이루어져 있음
- Update gate에서 LSTM의 input, forget gate를 제어
- 성능 면에서는 두 알고리즘이 비슷하나 데이터 양에 따라 많으면 LSTM, 적으면 GRU를 쓰는 편

[성능 테스트]

03
GRU-okt(1)

1. 감성분석을 위해 10점 : 긍정 / 그외 : 부정 으로 임의 설정

```
data['label'] = np.select([point > 9], [1], default=0)
data[:5]
```

	num	review	label
0	10	역대 대명작이라 자부합니다^^	1
1	10	진짜 이건 우리나라 영화계에 한 획을 그을 정도에 명작입니다 우리가 역사를 배워야 ...	1
2	3	국뽕 범벅, 거품, 스크린독점 관객동원	0
3	10	초등 고학년 아들과 진도에 다녀온후 같이 봤는데 정말 감사하고 감동이며 죄송스러운 ...	1
4	10	깊게 숨을 들이쉬고 후하고 내뱉은 뒤, 이 영화를 보아라. 다보고 나서 이틀 뒤에 ...	1

2. okt로 명사 추출(단어 빈도수로 긍정/ 부정 확률 추출)

```
train_data['tokenized'] = train_data['review'].apply(okt.nouns)
```

```
train_data['tokenized'].head()
```

```
8725                   [기, 영화, 감독, 욕, 바리, 영화, 현실]
1432                                       [최고, 영화]
2052    [영화, 다운, 영화로, 영화, 국밥, 재회씬, 번, 번, 울, 밥, 그게, 여태,...
4082                                      [마블, 시리즈]
5147                                  [기억, 역사, 영화]
Name: tokenized, dtype: object
```

```
test_data['tokenized'] = test_data['review'].apply(okt.nouns)
```

```
negative_words = np.hstack(train_data[train_data.label == 0]['tokenized'].values)
positive_words = np.hstack(train_data[train_data.label == 1]['tokenized'].values)
```

```
from collections import Counter
```

```
negative_word_count = Counter(negative_words)
print(negative_word_count.most_common(50))
```

```
[('영화', 1212), ('스토리', 255), ('이', 235), ('것', 235), ('연기', 220), ('생각', 188), ('진짜', 188), ('그', 167), ('사람', 157), ('배
우', 151), ('왜', 150), ('점', 141), ('더', 138), ('좀', 137), ('액션', 135), ('그냥', 129), ('수', 126), ('감독', 120), ('볼', 120),
```

3. 모델 구성

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

```
전체 샘플 중 길이가 30 이하인 샘플의 비율: 96.70787809085681
```

```
X_train = pad_sequences(X_train, maxlen=max_len)
X_test = pad_sequences(X_test, maxlen=max_len)
```

```
from tensorflow.keras.layers import Embedding, Dense, GRU
from tensorflow.keras.models import Sequential
from tensorflow.keras.models import load_model
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint

embedding_dim = 100
hidden_units = 128

model = Sequential()
model.add(Embedding(vocab_size, embedding_dim))
model.add(GRU(hidden_units))
model.add(Dense(1, activation='sigmoid'))

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='rmsprop', loss='binary_crossentropy', metrics=['acc'])
history = model.fit(X_train, y_train,
                    epochs=50, callbacks=[es, mc], batch_size=64, validation_split=0.2)
```

긍정/부정의 이진 분류이므로 sigmoid 함수 사용

03
SK-ok

```
from sklearn.naive_bayes import MultinomialNB, BernoulliNB # 다항분포 나이브 베이즈 모델
from sklearn.metrics import accuracy_score #정확도 계산
```

```
from sklearn.metrics import classification_report
```

```
y_label = np.array(data.num.values)
print(y_label)
```

```
[10 10  3 ... 10 10 10]
```

```
Y_train, Y_test = train_test_split(y_label,
                                   test_size = 0.2,
                                   random_state = 88)
```

```
mod = MultinomialNB()
mod.fit(X_train, Y_train)
```

```
MultinomialNB()
```

```
MultinomialNB(alpha=1.0, class_prior=None, fit_prior=True)
```

```
MultinomialNB()
```

```
predicted = mod.predict(X_test) #테스트 데이터에 대한 예측
print("정확도:", accuracy_score(Y_test, predicted)) #예측값과 실제값 비교
```

```
정확도: 0.3195402298850575
```

```
print(classification_report(Y_test, predicted))
```

```
              precision    recall  f1-score   support

           1       0.10      0.01      0.01       132
           2       0.09      0.11      0.10        70
           3       0.01      0.07      0.01        14
           4       0.00      0.00      0.00        49
           5       0.03      0.12      0.04        26
           6       0.06      0.14      0.08        70
           7       0.07      0.16      0.10        69
           8       0.11      0.09      0.10       181
           9       0.03      0.01      0.01        99
          10       0.64      0.49      0.55      1030

    accuracy                           0.32      1740
   macro avg       0.11      0.12      0.10      1740
weighted avg       0.41      0.32      0.35      1740
```

- 각 라벨 별 데이터 양이 불균형할 뿐더러 데이터의 품질이 좋지 않아 정확도가 매우 낮음

03
GRU-okt

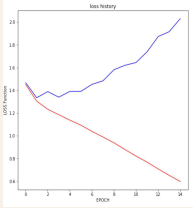
모델 구성

```
# GRU 모델 - 유민석님 공유
import re
from tensorflow.keras.datasets import imdb
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, GRU, Embedding


hidden_units = 512


model = Sequential()
model.add(Embedding(vocab_size +1, textlen))
model.add(GRU(hidden_units))
model.add(Dense(10, activation='softmax'))

model.compile(optimizer='rmsprop', loss='categorical_crossentropy', metrics
history = model.fit(train_x, train_y, epochs=15,
                    validation_data=(validation_x,validation_y))
```

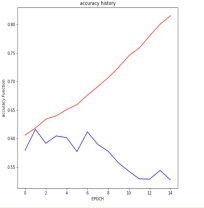
epochs = 30 실행

```
Epoch 13/30
6087/6087 [==============================] - 151s 25ms/sample - loss: 0.9
177 - accuracy: 0.7163 - val_loss: 1.5216 - val_accuracy: 0.5899
Epoch 14/30
6087/6087 [==============================] - 115s 19ms/sample - loss: 0.8
828 - accuracy: 0.7227 - val_loss: 1.5952 - val_accuracy: 0.5588
Epoch 15/30
6087/6087 [==============================] - 137s 22ms/sample - loss: 0.8
512 - accuracy: 0.7365 - val_loss: 1.5544 - val_accuracy: 0.5680
Epoch 16/30
6087/6087 [==============================] - 145s 24ms/sample - loss: 0.8
148 - accuracy: 0.7465 - val_loss: 1.6327 - val_accuracy: 0.5638
```

loss



accuracy



red- train / blue -validation

03
GRU-Kkma

모델 구성

```
# GRU 모델
import re
from tensorflow.keras.datasets import imdb
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, GRU, Embedding

hidden_units = 512

model = Sequential()
model.add(Embedding(vocab_size +1, textlen))
model.add(GRU(hidden_units))
model.add(Dense(10, activation='softmax'))

model.compile(optimizer='rmsprop', loss='sparse_categorical_crossentropy
history = model.fit(train_x, train_y, epochs=15,
                    validation_data=(validation_x,validation_y))
```

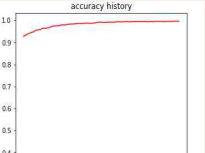
```
early_stop = keras.callbacks.EarlyStopping(monitor='val_loss', patience=50)
history = model.fit(train_x, train_y, epochs=50,
                    validation_data=(validation_x, validation_y),
                    callbacks=[early_stop])
```

epochs = 50 실행

```
Epoch 6/50
186/186 [==============================] - 48s 260ms/step - loss: 0.1407 - accuracy:
0.9575 - val_loss: 4.8151 - val_accuracy: 0.2592
Epoch 7/50
186/186 [==============================] - 48s 257ms/step - loss: 0.1248 - accuracy:
0.9635 - val_loss: 4.8598 - val_accuracy: 0.2804
Epoch 8/50
186/186 [==============================] - 46s 248ms/step - loss: 0.1161 - accuracy:
0.9638 - val_loss: 5.1137 - val_accuracy: 0.2675
Epoch 9/50
186/186 [==============================] - 47s 256ms/step - loss: 0.1052 - accuracy:
0.9670 - val_loss: 4.8999 - val_accuracy: 0.2745
```

loss

accuracy



red- train / blue -validation

03
GRU-Mecab

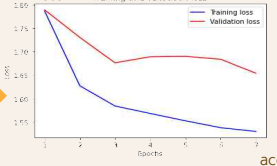
모델 구성

```
import re
from tensorflow.keras.datasets import imdb
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, GRU, Embedding

vocab_size = 500
embedding_dim = 30
hidden_units = 512


model = Sequential()
model.add(Embedding(vocab_size+1, embedding_dim))
model.add(GRU(hidden_units))
model.add(Dense(10, activation='softmax'))

model.compile(optimizer='rmsprop', loss='sparse_categorical_crosse
history = model.fit(train_x, train_y, epochs=7,
                    validation_data=(test_x,test_y))
```

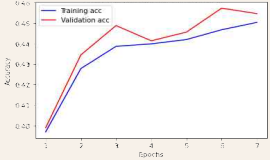
epochs = 7 실행

```
Epoch 7/7
237/237 [==============================] - 8s 33ms/step - loss: 1
.5304 - acc: 0.4503 - val_loss: 1.6543 - val_acc: 0.4545
```

loss



accuracy



red- train / blue -validation

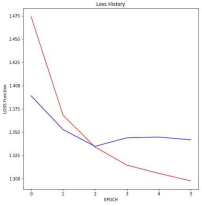
03
GRU-komoran

모델 구성

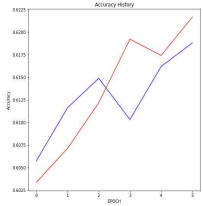
```
model = Sequential()
model.add(Embedding(vocab_size + 1, embedding_dim))
model.add(GRU(hidden_units))
model.add(Dense(10, activation = 'softmax'))
```

```
model.compile(optimizer = 'rmsprop', loss = 'categorical_crossentropy', metrics = ['accuracy'])
```

loss



accuracy



red- train / blue -validation

평점 예측

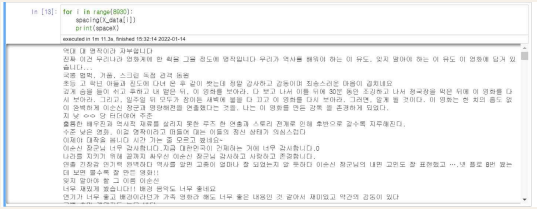
```
==========
1 번째 리뷰

한 마디로 '처묵' 의 과정
평점 : ★★★★★★★★★★ 10
==========
2 번째 리뷰

감동입니다..너무 재미있어용
평점 : ★★★★★★★★★★ 10
==========
3 번째 리뷰

최악 !! 다신 안본다.
평점 : ★☆☆☆☆☆☆☆☆☆ 1
==========
4 번째 리뷰

처음엔 좋았으나 갈수록 보기거북함
평점 : ★★★★★★★★★★ 10
==========
5 번째 리뷰

사형수는 개뿔, 죄없는 애들 속여서 끌고간거라고! 유족들은 억울할도 못풀었는데... 이런식으로 팔아먹었어야 했나? 감독 진짜 양심 터졌구나.
평점 : ★☆☆☆☆☆☆☆☆☆ 1
==========
```

4번째 리뷰에 '거북' 이라는 단어가 전체 데이터에서 20회 미만으로 나와 정확도가 떨어짐

[후기 및 아쉬운 점]

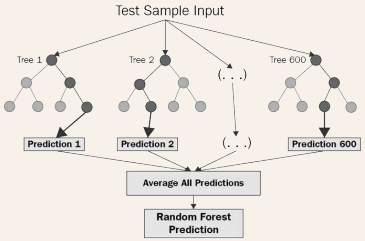
04
구현하지 못 한 내용

pykospacing



Pykospacing 이용 시
Tensorflow 강제 다운그레이드
문제로 전체적으로 사용을
못함

04
구현하지 못 한 내용



Random Forest

- 앙상블 머신러닝 모델
  ※ 좋은 성능을 위해 다수의 학습 알고리즘을 사용하는 것을 앙상블 학습법이라 함
- 다수의 의사결정 트리의 분류를 집계하여 최종적으로 분류
- 오버피팅이 일어나도 다수의 트리를 기반으로 하여 오버피팅의 영향력은 줄어들음
- 중복 허용하여 랜덤으로 학습 데이터셋을 추출하여 트리 생성
- 약 분류기(결정트리)가 모여 강 분류기(랜덤 포레스트) 생성됨

```
from sklearn.ensemble import RandomForestClassifier

clf = RandomForestClassifier()
clf.fit(X_train, y_train)
```

```
RandomForestClassifier()
```

```
from sklearn.model_selection import cross_val_score

cross_val_score(clf, X_train, y_train, cv = 7, scoring = 'accuracy')
```

```
array([0.42857143, 0.46971429, 0.464     , 0.44571429, 0.47085714,
       0.42971429, 0.42105263])
```

- 정확도가 낮아 쓰지 않음

```
from sklearn.naive_bayes import MultinomialNB, BernoulliNB # 다항분포 나이브 베이즈 모델
from sklearn.metrics import accuracy_score #정확도 계산
```

```
from sklearn.metrics import classification_report
```

```
y_label = np.array(data.num.values)
print(y_label)
```

```
[10 10  3 ... 10 10 10]
```

```
Y_train, Y_test = train_test_split(y_label,
                                   test_size = 0.2,
                                   random_state = 88)
```

```
mod = MultinomialNB()
mod.fit(X_train, Y_train)
```

```
MultinomialNB()
```

```
MultinomialNB(alpha=1.0, class_prior=None, fit_prior=True)
```

```
MultinomialNB()
```

```
predicted = mod.predict(X_test) #테스트 데이터에 대한 예측
print("정확도:", accuracy_score(Y_test, predicted)) #예측값과 실제값 비교
```

```
정확도: 0.31954022988505175
```

```
print(classification_report(Y_test, predicted))
```

```
              precision    recall  f1-score   support

           1       0.10      0.01      0.01       132
           2       0.09      0.11      0.10        70
           3       0.01      0.07      0.01        14
           4       0.00      0.00      0.00        49
           5       0.03      0.12      0.04        26
           6       0.06      0.14      0.08        70
           7       0.07      0.16      0.10        69
           8       0.11      0.09      0.10       181
           9       0.03      0.01      0.01        99
          10       0.64      0.49      0.55      1030

    accuracy                           0.32      1740
   macro avg       0.11      0.12      0.10      1740
weighted avg       0.41      0.32      0.35      1740
```

- 감성 분석 결과로 각 댓글의 긍정/ 부정 확률을 얻으며, 이를 단어 빈도수 기준으로 1부터 10까지 나누어 나이브 베이즈에 대입
- 각 라벨 별 데이터 양이 불균형할 뿐더러 데이터의 품질이 좋지 않아 정확도가 매우 낮음

04
느낀점

유민석

- 프로젝트의 주제를 명확히 선정해야 방향성을 잃지 않는다는 것을 깨달음
- 편향된 데이터로는 여러 방면으로 전처리를 해도 모델 성능 개선에 한계가 있음을 느낌
- 차후에는 이러한 점들을 토대로 보완할 것임

구은혜

- 자연어 전처리를 해보며 그간 학습한 내용을 활용해볼 수 있었음
- 모델 성능 개선에 아쉬움이 있지만, 팀원들과의 지식 공유가 많은 도움이 됨

박지홍

- 직접 데이터를 수집하고 전처리하는 과정을 통해 좋은 데이터란 무엇인지 생각해볼 수 있었음
- 팀원들과의 교류로 부족한 점을 인지하고 복습 방향을 잡는 계기가 됨

엄예찬

- 인간의 표현이 얼마나 간단하면서도 복잡한지 생각해볼 수 있었음
- 구글 어시스턴트는 자연어처리의 신이다

임예지

- 전처리 과정에 어려움이 있어 생각보다 시간 투자를 많이 하며 그 중요성을 느낌
- 팀원들과 함께 연구하고 시도하는 과정에서 얻는 것이 많았음

최서연

- 유효한 데이터가 아니더라도 학습이 가능하다는 사실을 배움
- 좋은 성능으로 구현하기엔 어려움이 있어 학습 데이터 제작 사업 성장의 이유를 알 수 있었음


QA

감사합니다