항공 리뷰 감성 분석

1. 목표

- 항공사별 리뷰 긍·부정 비율 계산
- 항공사별 리뷰 내에 등장하는 빈도가 높은 키워드 출력
- 리뷰 기반 동적 워드클라우드 구현

2. 데이터

- 항공사 선정
  - 인천 공항 발 국제선 운영 항공사
  - 유나이티드항공, 아시아나항공, 아메리칸항공, 중국동방항공, 독일항공, 네덜란드항공, 대한항공, 진에어, 프랑스항공, 중국남방항공, 델타항공, 에미레이트항공, 티웨이항공, 제주항공, 에어서울, 카타르항공
- 리뷰 데이터
  - 제목 및 내용

3. 항공사 리뷰 긍부정 비율

3.1 환경 설정

```
// konlpy 설치

!apt-get update
!apt-get install g++ openjdk-8-jdk
!pip install konlpy
```

```
# 라이브러리 설치

import pandas as pd
import numpy as np
%matplotlib inline
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")

from matplotlib import pyplot as plt
plt.rc('font', family='NanumBarunGothic')

import re
import urllib.request
from konlpy.tag import Okt
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
```

3.2 훈련

```
urllib.request.urlretrieve("https://raw.githubusercontent.com/e9t/nsmc/master/ratings_train.txt", filename="ratings_train.txt")
urllib.request.urlretrieve("https://raw.githubusercontent.com/e9t/nsmc/master/ratings_test.txt", filename="ratings_test.txt")

train_data = pd.read_table('ratings_train.txt')
test_data = pd.read_table('ratings_test.txt')
```

```
train_data['document'].nunique(), train_data['label'].nunique()
train_data.drop_duplicates(subset=['document'], inplace=True)
train_data.loc[train_data.document.isnull()]
train_data = train_data.dropna(how = 'any')
train_data['document'] = train_data['document'].str.replace("[^ㄱ-ㅎㅏ-ㅣ가-힣 ]","")
train_data['document'] = train_data['document'].str.replace('^ +', "")
train_data['document'].replace('', np.nan, inplace=True)
train_data = train_data.dropna(how = 'any')

test_data.drop_duplicates(subset = ['document'], inplace=True)
test_data['document'] = test_data['document'].str.replace("[^ㄱ-ㅎㅏ-ㅣ가-힣 ]","")
test_data['document'] = test_data['document'].str.replace('^ +', "")
test_data['document'].replace('', np.nan, inplace=True)
test_data = test_data.dropna(how='any')

stopwords = ['의','가','이','은','들','는','좀','잘','걍','과','도','를','으로','자','에','와','한','하다']
okt = Okt()

X_train = []
for sentence in train_data['document']:
    temp_X = okt.morphs(sentence, stem=True)
    temp_X = [word for word in temp_X if not word in stopwords]
    X_train.append(temp_X)

X_test = []
for sentence in test_data['document']:
    temp_X = okt.morphs(sentence, stem=True)
    temp_X = [word for word in temp_X if not word in stopwords]
    X_test.append(temp_X)

tokenizer = Tokenizer()
tokenizer.fit_on_texts(X_train)

threshold = 3
total_cnt = len(tokenizer.word_index)
rare_cnt = 0
total_freq = 0
rare_freq = 0

for key, value in tokenizer.word_counts.items():
    total_freq = total_freq + value

    if(value < threshold):
        rare_cnt = rare_cnt + 1
        rare_freq = rare_freq + value

tokenizer = Tokenizer(vocab_size)
tokenizer.fit_on_texts(X_train)
X_train = tokenizer.texts_to_sequences(X_train)
X_test = tokenizer.texts_to_sequences(X_test)

y_train = np.array(train_data['label'])
y_test = np.array(test_data['label'])

drop_train = [index for index, sentence in enumerate(X_train) if len(sentence) < 1]

X_train = np.delete(X_train, drop_train, axis=0)
y_train = np.delete(y_train, drop_train, axis=0)

def below_threshold_len(max_len, nested_list):
  cnt = 0
  for s in nested_list:
    if(len(s) <= max_len):
        cnt = cnt + 1

max_len = 30
below_threshold_len(max_len, X_train)

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

3.3 모델 학습

```
from tensorflow.keras.layers import Embedding, Dense, LSTM
from tensorflow.keras.models import Sequential
from tensorflow.keras.models import load_model
```

```
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint

model = Sequential()
model.add(Embedding(vocab_size, 100))
model.add(LSTM(128))
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=15, callbacks=[es, mc], batch_size=60, validation_split=0.2)
loaded_model = load_model('best_model.h5')
```

3.4 항공사 리뷰 감성 분석

```
AirCanada = pd.read_csv("/content/drive/MyDrive/data/항공사리뷰/AirCanada.csv")

AirCanada['review'] = AirCanada['review'].str.replace("[^ㄱ-ㅎㅏ-ㅣ가-힣 ]","")

positive_cnt = 0
for review in AirCanada['review']:
    new_sentence = okt.morphs(review, stem=True)
    new_sentence = [word for word in new_sentence if not word in stopwords]
    encoded = tokenizer.texts_to_sequences([new_sentence])
    pad_new = pad_sequences(encoded, maxlen = max_len)
    score = float(loaded_model.predict(pad_new))
    if(score > 0.5):
      positive_cnt  += 1
positive_ratio = round(positive_cnt / len(AirCanada), 2) * 100
print(f'총 {len(AirCanada)}개의 리뷰 중 {positive_cnt}개의 긍정 리뷰입니다. 긍정리뷰 비율은 {positive_ratio} % 입니다.')
```

3.5 결과

Aa 항공사 종류	긍정 리뷰 비율
AirCanada	53%
AirFrance	53%
AirSeoul	57%
AmericanAirline	53%
Asiana	65%
ChinaEastern	37%
ChinaSouthern	39%
Delta	53%
Emirates	48%
JejuAir	48%
JinAir	57%
KLM	49%
KoreanAir	62%
LuftHansa	57%
Qatar	63%
Tway	51%
United	39%

4. 항공사 리뷰 키워드

4.1 코드

```
# 형태소 분석
# JAVA 환경에서 사용되는 konlpy를 Python의 Django 환경에서 사용하기 위해
# konlpy의 komoran을 Python에 맞게 변형시킨 pykomoran을 사용하여 한글 형태소 분석
# 새로운 리뷰 반영 가능한 동적 형태의 키워드 추출

@api_view(['GET'])
def review_keyword(request, airline_id):
    words = pd.read_csv('npl/stopwords.csv', header=None)
    stopwords = list(words[0])
    reviews = get_list_or_404(Review, airline=airline_id)
    airline_review = ""
    for review in reviews:
        airline_review += review.content.replace("[^ㄱ-ㅎㅏ-ㅣ가-힣 ]","")
        airline_review += review.title.replace("[^ㄱ-ㅎㅏ-ㅣ가-힣 ]","")

    from PyKomoran import Komoran, DEFAULT_MODEL
    komoran = Komoran(DEFAULT_MODEL['LIGHT'])
    target_tags = ['NNG']
    noun_adj_list = komoran.get_morphes_by_tags(airline_review, tag_list=target_tags)
    final_list = []
    for noun_adj in noun_adj_list:
        if noun_adj not in stopwords:
            final_list.append(noun_adj)

    count = Counter(final_list)
    words = dict(count.most_common())
    words = sorted(words.items(), key=lambda x: x[1], reverse=True)

    return HttpResponse(json.dumps(words, ensure_ascii = False), content_type = 'application/json; charset=utf8')
```

4.2 결과

5. 항공사 리뷰 클라우드

5.1 코드

```
# 형태소 분석
# JAVA 환경에서 사용되는 konlpy를 Python의 Django 환경에서 사용하기 위해
# konlpy의 komoran을 Python에 맞게 변형시킨 pykomoran을 사용하여 한글 형태소 분석
# 새로운 리뷰 반영한 동적 형태의 워드클라우드

@api_view(['GET'])
def review_wordcloud(request, airline_id):
    words = pd.read_csv('npl/stopwords.csv', header=None)
    stopwords = list(words[0])
    reviews = get_list_or_404(Review, airline=airline_id)
    airline_review = ""
    for review in reviews:
        airline_review += review.content.replace("[^ㄱ-ㅎㅏ-ㅣ가-힣 ]","")
        airline_review += review.title.replace("[^ㄱ-ㅎㅏ-ㅣ가-힣 ]","")

    from PyKomoran import Komoran, DEFAULT_MODEL
```

```
    komoran = Komoran(DEFAULT_MODEL['LIGHT'])
    target_tags = ['NNG', 'VA']
    noun_adj_list = komoran.get_morphes_by_tags(airline_review, tag_list=target_tags)
    final_list = []
    for noun_adj in noun_adj_list:
        if noun_adj not in stopwords:
            final_list.append(noun_adj)

    count = Counter(final_list)
    word_dict = []
    for word, cnt in count.most_common()[:100]:
        if len(word) >= 2:
            obj = {
                "name": word,
                "weight": cnt,
            }
            word_dict.append(obj)

    return HttpResponse(json.dumps(word_dict, ensure_ascii = False), content_type = 'application/json; charset=utf8')
```

5.2 결과