Individual Assignment 2

리뷰데이터 Topic Modeling을 통한

어플리케이션 내 데이터 제공 방안

담당교수님: 윤장혁 교수님

과목명: Data Analytics

이름: 박민성

학번: 201611145

전공: 산업공학과

제출일: 2020.06.12

1. 주제선정

주제 선정 초기

우선 주제를 선정하기에 앞서 나에게 주어진 데이터 'data_review.csv'에 대해서 파악해 보았다. 내가 흔히 알고 있는 배달 앱에 대한 리뷰였다. Csv 파일 안에는 별점, 업체명, 주문한 음식, 리뷰, 날짜에 대한 나에게 익숙한 데이터들이 존재했다.

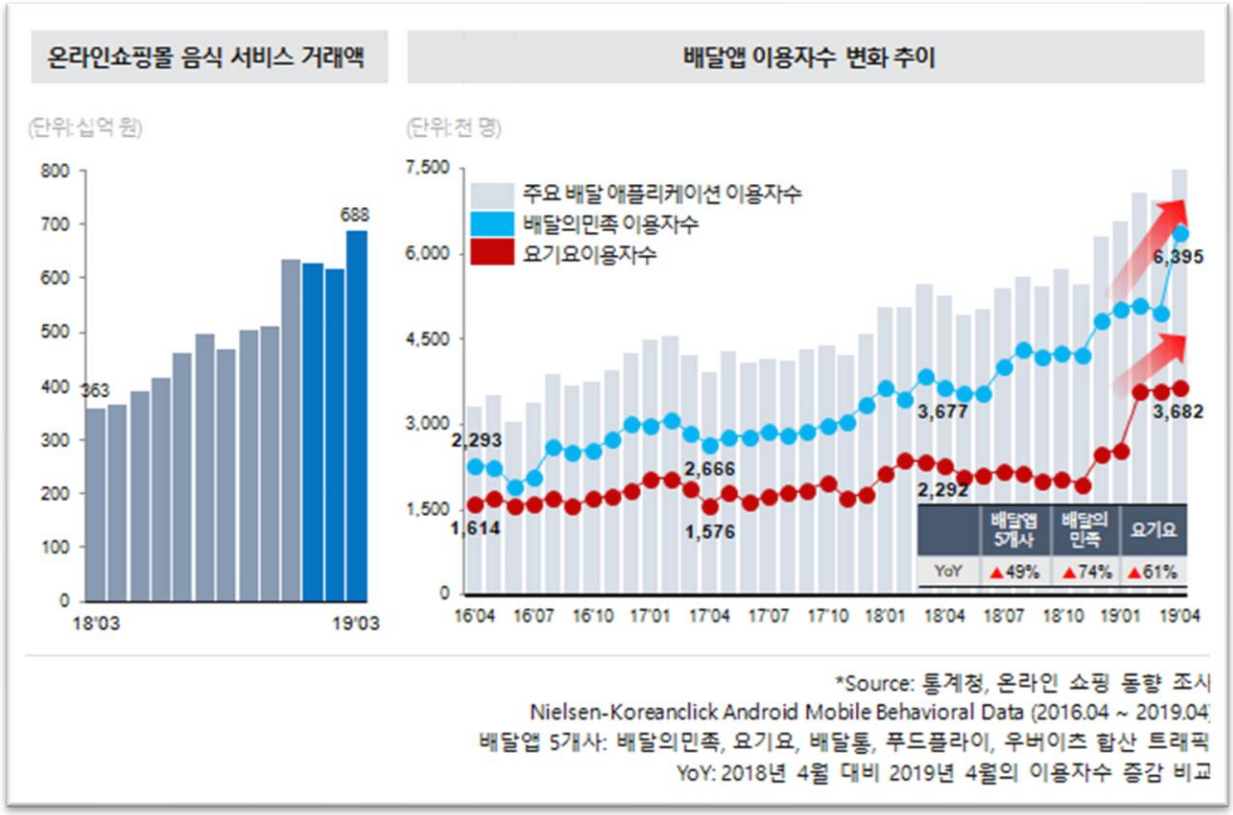
앱 내에서 평소에 내가 주로 이용하는 기능은 원하는 종류의 음식을 선택하고, 특정 업체를 클릭하여서 리뷰, 별점 등을 보고 업체, 음식을 선정하는 것이다. 하지만 어느 지점의 별점이 높은지, 리뷰가 가장 많은지 등의 정보를 이용하는 것은 SQL로도 충분히 해결 가능하고 DA와는 맞지 않다고 생각했다. 다음으로는 생각해 본 것은 음식이나 업체의 Recommendation에 관한 것인데 이 또한 User의 데이터가 없고 내가 가진 데이터와는 부합하지 않다고 생각했다.

결론적으로 '내 자료와 적합하고 DA를 통해서만 할 수 있는 주제, 목표를 설정하는 것이 좋을 것 같다.'라는 생각을 하게 되었고 잠재적인 결과를 도출할 수 있는 것이 무엇이 있을지 생각해 보던 와중에 '지금 배달 앱에 없는 것이 무엇인가?'라는 생각을 하게 되었다. 현재 배달 어플은 물론 사업주에게 매출증가라는 기능을 하고 있지만 어플 자체에서 사업주에게 정보의 전달은 거의 이루어지지 않고 있다. 사업주 혹은 사업을 준비하고 있는 사람들에게 도움이 될 수 있는 정보를 제공하고 싶다는 생각을 했다.

또한 현재 1인가구 증가, 배달 어플의 엄청난 성장이 이루어지고 있는데 리뷰 데이터가 지금처럼 눈으로 하나하나 볼 수 있는 수준이 아니라 엄청난 데이터의 양이 존재한다고 했을 때 '소비자에게 한 줄로 이 업체에 대한 평가가 이루어 지면 좋겠다'라는 생각을 했다.

주제선정과정과 이유

단지 단어의 출현 빈도수로만 네트워크 분석을 진행하게 된다면 가장 빈출하는 단어를 한눈에 볼 수있고 단어의 동시 출현을 볼 수 있지만 단어 사이의 잠재적인 의미는 도출하지 못한다고 생각했다. 따라서 VOC, 즉 소비자들이 남긴 리뷰를 업종/업체에 따라 Topic Modeling을 통해 소비자들에겐 '한 줄 평가 제공'하고 사업자/사업을 준비하는 사람에게는 '업체에 대한 문제점', '이 업종의 전체적인 문제점 제시'라는 주제를 선정하게 되었다.



<배달 앱 이용자 수 증가 추세>

1인 가구가 늘어가고, 배달 앱의 이용자수가 해가 갈 수록 늘어가고 있다. 현재와 같은 리뷰 시스템은 소비자가 일일히 확인을 해야된다는 번거로움이 있다. 지금은 데이터가 많지 않아서 확인하고 업체별로 불만 사항을 확인할 수 있지만 앞으로 데이터가 많아지게 된다면 리뷰를 읽는 시간도 많이 걸릴 것이고 User 입장에서 혼란이 있을 수도 있다. 따라서, 업체별로 고객들의 불만사항, 업체의 좋은 점을 등을 Topic Modeling을 통해서 자동화하여 한 줄로 제시한다.

지금의 배달 앱에서는 사업주를 대상으로 한 정보 제공이 이루어지지 않고 있다. 리뷰 분석을 통한 정보 제공으로 기업 입장에서는 현재 주 소비자의 불만이 무엇인지 파악할 수 있다.

또한, 지금 사업을 운영하고 있지 않지만 잠재적으로 가게를 오픈하거나 관심이 있는 사람들에게 동종업계의 주된 불만 사항이 무엇인지 좋은 점이 무엇인지 알 수 있다.

이러한 기능의 구현을 통해서 소비자는 가게의 품질개선으로 인한 잠재적인 혜택까지 기대해 볼 수 있다.

최종 목표 설정

1. 리뷰와 별점이 동시에 존재하는 데이터에 대해 (나쁨,보통,좋음)을 태깅한다.
2. 내가 가진 데이터를 정제하는 과정을 거쳐서 벡터화 시킨다.
3.. 리뷰는 존재하지만 별점이 존재하지 않는 데이터를 Logistic Regression Classifier를 통해서 태깅해주는 전처리 과정을 거친다.
4. 리뷰와 별점이 원래 있던 데이터와 전처리과정을 거쳐서 태깅이 된 데이터를 합쳐서 Topic Modeling을 진행한다.
5. 선택한 업체/업종에 대한 고객 불만/만족 사항을 제시한다.

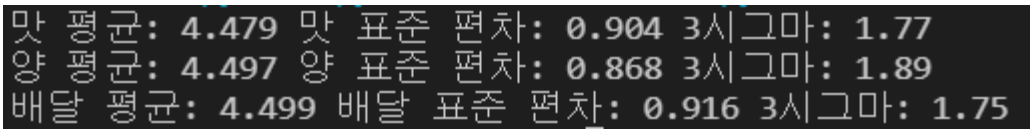
2. 목표에 따른 구체적 방법론과 과정

Tagging

표본을 분석할때 통계적으로 표본이 많을 수록 결과가 유의하기 때문에 따라서 리뷰는 있지만 별점이 없는 데이터와 둘다 있는 데이터를 모아서 분석을 진행하기로 하였다.

리뷰가 없는 데이터에 대해서 처리를 어떻게 해줘야 될지 고민을 많이 했다. 이 부분은 리뷰와 별점이 동시에 존재하는 데이터를 통해서 분류기를 학습시킨 후 별점이 없는 데이터를 가공하기로 하였다.

우선, 별점을 어느수준으로 해서 고객이 만족하는지 불만족하는지 알아보기 위해 각 리뷰의 별점 평균, 표준 편차, 3시그마 수준을 계산해보았다.



<별점 계산.py의 실행결과>

별점의 평균이 높아서 나쁨의 수준은 2로 설정하였고 좋음의 수준은 4이상으로 설정했다.

```python
import numpy as np
import konlpy
import csv
from tqdm import tqdm
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer, CountVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
import gensim
import gensim.corpora as corpora

review_O_data=[]
f = open('./data_review.csv','r',encoding='utf-8')
rdr = csv.reader(f)
review_O_data_tag = []
review_X_data =[]
f2 = open('./data_review.csv','r',encoding='utf-8')
rdr2 = csv.reader(f2)
select_category=input("야식, 족발/보쌈, 한식, 분식, 일식/돈까스, 피자, 중식, 치킨")
company_list=[]
for j in rdr2:
    if j[1] ==select_category:
        company_list.append(j[0])
print(set(company_list))
select_company = input("위의 목록 또는 전체를 입력해 주세요")
Want_comfortable=input("좋음,보통,나쁨 을 입력하세요")
```

<main.py(1)>

```python
for i in rdr:
    try:
        if i[1] ==select_category:
            if select_company =='전체':
                if i[3] == '':
                    review_X_data.append(i[6])
                else:
                    if float(i[3])<=2 and float(i[4])<=2 and float(i[5])<=2:
                        review_O_data.append(i[6])
                        review_O_data_tag.append(0) #나쁨
                    elif float(i[3])>=4 and float(i[4])>=4 and float(i[5])>=4:
                        review_O_data.append(i[6])
                        review_O_data_tag.append(1) #좋음
                    else:
                        review_O_data.append(i[6])
                        review_O_data_tag.append(2) #보통
            elif i[0] ==select_company:
                if i[3] == '':
                    review_X_data.append(i[6])
                else:
                    if float(i[3])<=2 and float(i[4])<=2 and float(i[5])<=2:
                        review_O_data.append(i[6])
                        review_O_data_tag.append(0) #나쁨
                    elif float(i[3])>=4 and float(i[4])>=4 and float(i[5])>=4:
                        review_O_data.append(i[6])
                        review_O_data_tag.append(1) #좋음
                    else:
                        review_O_data.append(i[6])
                        review_O_data_tag.append(2) #보통
    except:
        pass
```

<main.py(2)>

처음에 input 값으로 정보를 업고 싶은 업종/그 업종의 업체명을 받는다.

업체 소비자의 만족에 대한 데이터를 얻고 싶다면 '좋음'을 업체 소비자의 불만족에 대한 데이터를 얻고 싶다면 '나쁨'을 입력한다.

<이 보고서의 모든 분석의 결과는 결과의 통일성을 위해 치킨-전체-나쁨에 대해서 다룬다.>

사용자의 목적에 따라 별점 수준을 조절할 수 있다. 이 보고서에는 별점 2점이하를 나쁨으로 별점 4점이상을 좋음으로 진행했다. 또한, 리뷰가 없는데이터와 리뷰가 있는 데이터는 각각 리스트에 추가해서 분석을 진행했다.

Vectorize

```
61  reveiw_Total_data = review_O_data + review_X_data
62  for i, document in tqdm(enumerate(reveiw_Total_data)):
63      okt = konlpy.tag.Okt()
64      clean_words = []
65      for word in okt.pos(document, stem =True): #Letimazation
66          if word[1] in ['Noun', 'Verb']:
67              clean_words.append(word[0])
68      document = ' '.join(clean_words)
69      reveiw_Total_data[i] = document
70
71  vectorize = CountVectorizer(min_df=10)
72  X = vectorize.fit_transform(reveiw_Total_data)
73  features = vectorize.get_feature_names()
74  Vector_Matrix = np.array(X.toarray())
75  review_O_data_df = pd.DataFrame(Vector_Matrix)
76  review_X_data_df = review_O_data_df.iloc[len(review_O_data)+1:,:]
77  review_X_data_df.to_csv("review_x_data.csv",encoding="utf-8-sig",header = features)
78  review_O_data_df=review_O_data_df.head(len(review_O_data))
79  review_O_data_df["태그"] = review_O_data_tag
80  features.append("태그")
81  review_O_data_df.to_csv("review_o_data.csv",encoding="utf-8-sig",header = features)
82  review_O_data_df=pd.read_csv('./review_o_data.csv')
83  review_X_data_df=pd.read_csv('./review_x_data.csv')
84  Y=review_O_data_df['태그']
85  features.pop(-1)
86  X=review_O_data_df[features]
```

<main.py(3)>

분석을 진행하기 위해 전체 데이터를 벡터화시켰다. 벡터화하는 과정에서 10번이하로 출현하는 단어들에 대해서 제외해주었고 정확한 Topic Modeling을 위해 명사와 동사만을 남겼다. 또한 원소간 중복성을 최대한 줄여주기 위해 어간추출을 진행했다. 이 과정을 진행한 결과 리뷰와 별점 둘다 있는 벡터화된 행렬에 좋음/나쁨/보통이 태깅되어 추가된 하나의 행렬을 얻었다.

	단어 1	단어 2	단어 3	....	단어 x	태그
리뷰 1	1	0	3	....	0	좋음
리뷰 2	0	1	0	....	0	나쁨
....	0	0	1	....	0	보통
리뷰 n	....	....	....	....	....	...

<Vectorize한 Matrix 예시 >

Logistic Regression Model Learning and prediction

```
88  X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.3)
89  log_clf = LogisticRegression()
90  log_clf.fit(X_train,Y_train)
91  print("정확도:",round(log_clf.score(X_test, Y_test)*100,2),"%")
92  review_X_data_tag = log_clf.predict(review_X_data_df[features])
93  review_x_tag_df = pd.DataFrame(review_X_data_tag)
94  review_Total_data_tag = review_O_data_tag+list(np.array(review_x_tag_df[0].tolist()))
95  final_data = []
96  spool = []
97  for i in range(0,len(review_Total_data_tag)):
98      spool.append(reveiw_Total_data[i])
99      if review_Total_data_tag[i] == 1:
100         spool.append('좋음')
101     elif review_Total_data_tag[i] == 2:
102         spool.append("보통")
103     else:
104         spool.append("나쁨")
105     final_data.append(spool)
106     spool = []
```

<main.py(4)>

리뷰 데이터의 별점의 태그를 예측하기 위해 로지스틱 회귀 모델을 사용하였다.

리뷰는 있지만 별점이 없는 데이터에 대해 (좋음, 나쁨, 보통)을 태그해준다.

```
정확도: 81.56 %
```

<91번째줄 실행결과>

정확도는 80% 전후가 나왔다.

Topic Modeling

```
lda_list =[]
for i in final_data:
    if i[1] ==Want_comfortable:
        lda_list.append(i[0])
documents =[]
documents = [line.rstrip('\n') for line in lda_list]
stoplist=["제","걸","번","그것","요","거","온","함","분","하다","임","것","다시다","전","치","뿌","링클"]
#결과를 계속 보고 stoplist를 채워준다.
texts = [[word for word in document.lower().split() if word not in stoplist] for document in documents]
dictionary = corpora.Dictionary(texts)
corpus = [dictionary.doc2bow(text) for text in texts]
lda = gensim.models.ldamodel.LdaModel(corpus=corpus, id2word=dictionary,
                                      num_topics=6, update_every=1, chunksize=100, passes=30)
result=lda.print_topics(num_words=9)
for i in result:
    print(i)
```

이제 lda_list에는 내가 원하는 업종/업체의 원하는 태그가 들어있는 정제된 모든 리뷰가 들어 있다. 이 list를 통해 Topic Modeling을 진행한다.

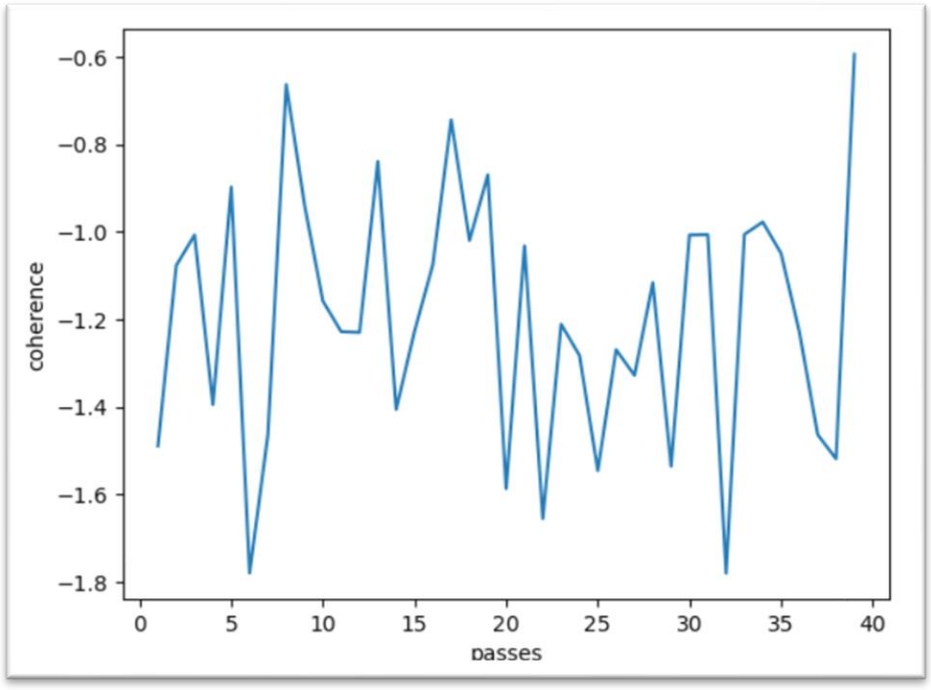
Topic Modeling 최적 K값 결정

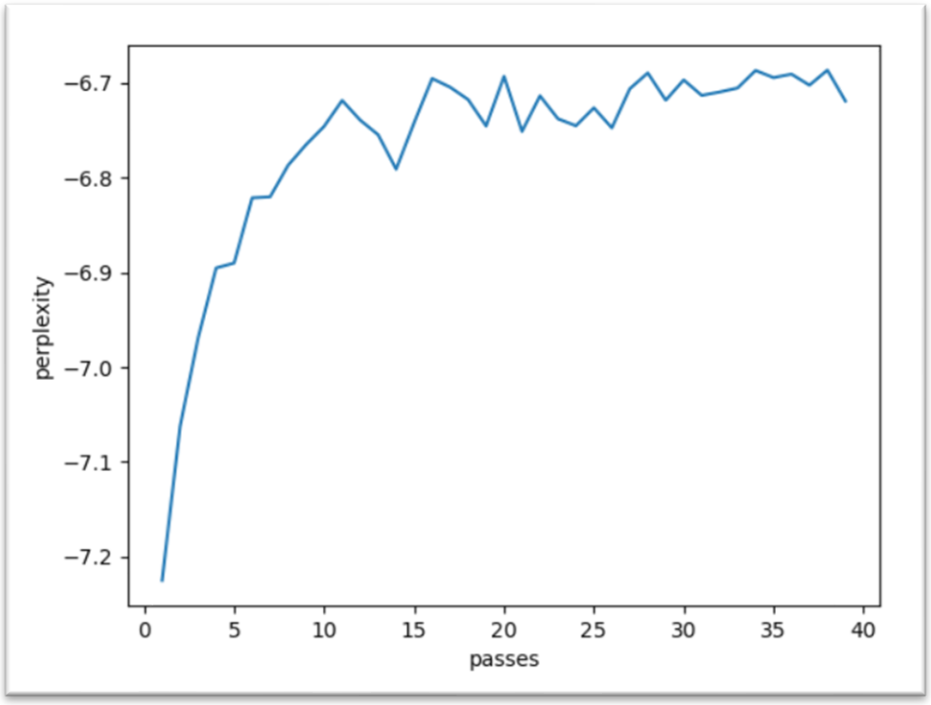
```
for i in tqdm(range(1,40)):
    lda = gensim.models.ldamodel.LdaModel(corpus=corpus, id2word=dictionary,
    num_topics=11, update_every=1, chunksize=100, passes=40)
    result=lda.print_topics(num_words=i)

    cm = CoherenceModel(model=lda, corpus=common_corpus, coherence='u_mass')
    coherence = cm.get_coherence()
    c=coherence
    p=lda.log_perplexity(corpus)
    x.append(i)
    y.append(c)
    z.append(p)

plt.plot(x,y)
plt.xlabel('num_words')
plt.ylabel('coherence')
plt.show()

plt.plot(x,z)
plt.xlabel('num_words')
plt.ylabel('perplexity')
plt.show()
```

<최적 k값.py- num_words에 대해 i로 설정>

Passes, num_topics, num_words = i로 설정, 반복 횟수에 따라 Topic Coherence, Perplexity를 관찰 후 적절한 K값을 설정한다.

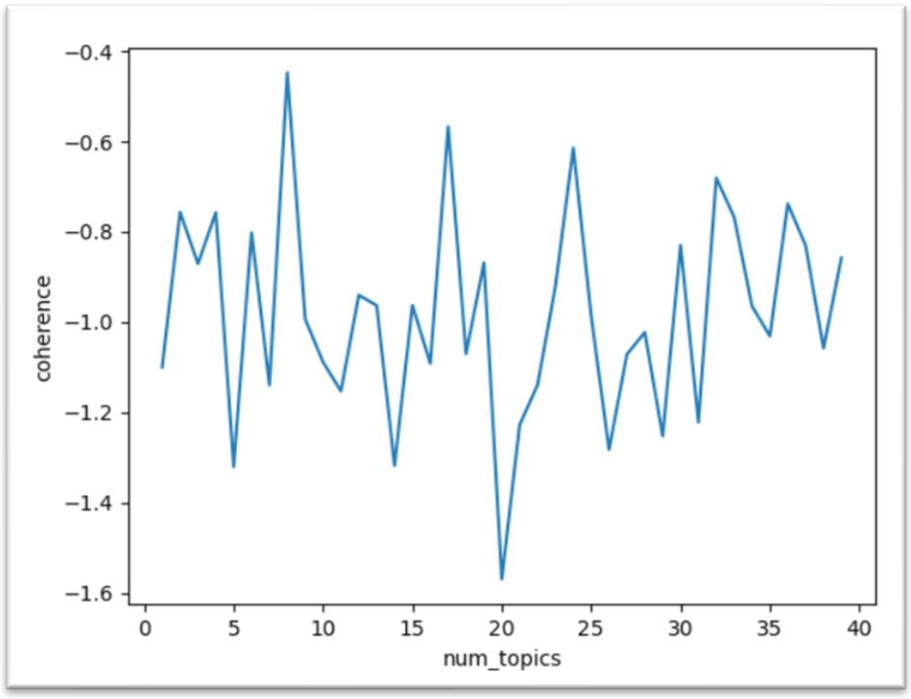
Passes 결정 – 모델 학습 횟수

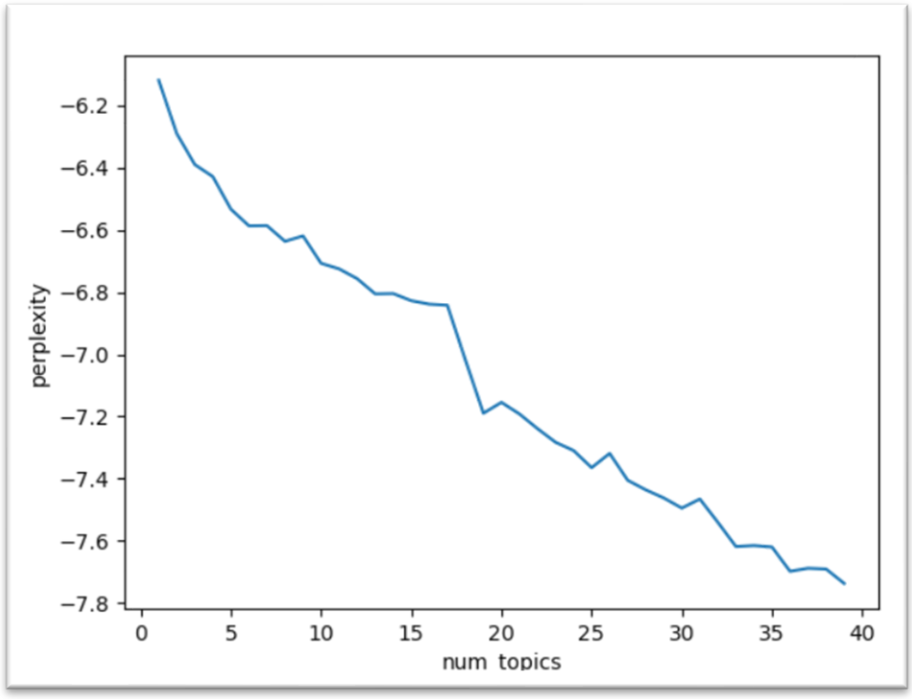




최종 Passes 설정 - 30

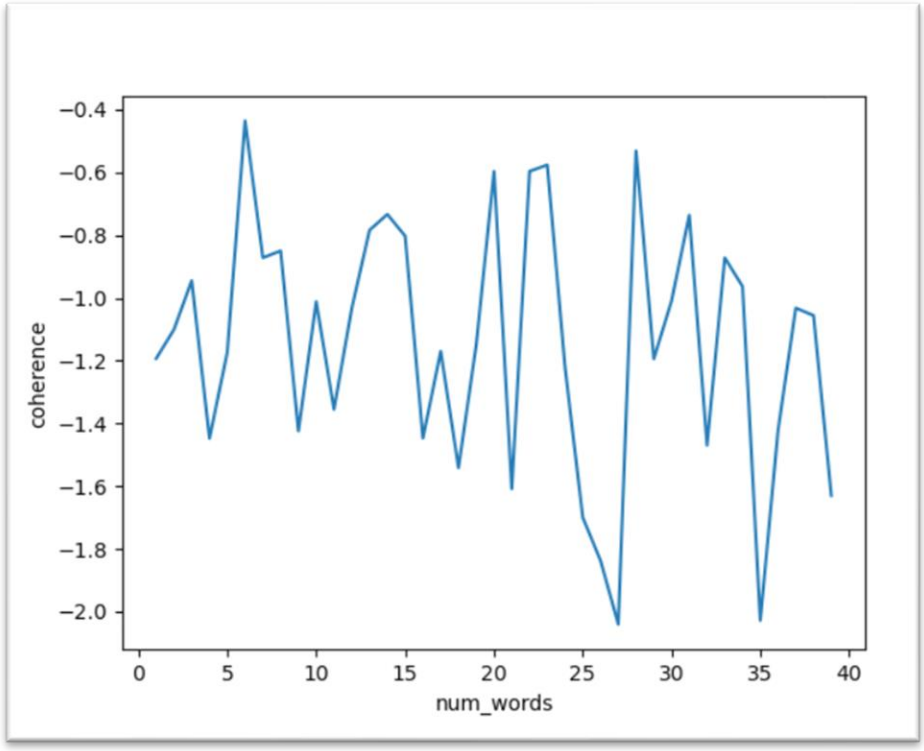
num_topics 결정 - 주제의 개수

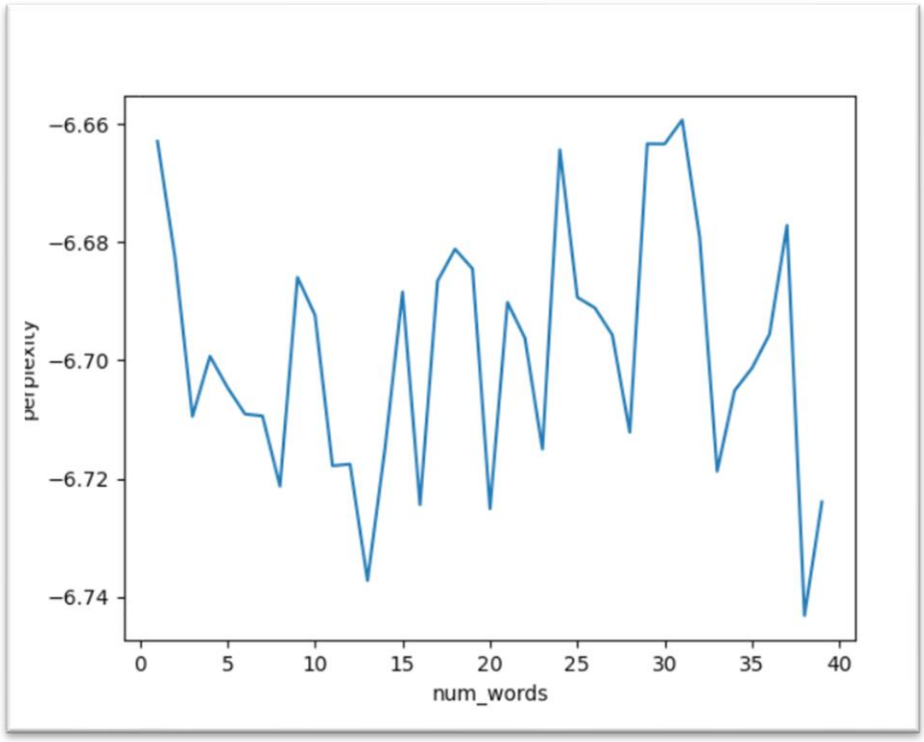




최종 num_topics 설정 - 6

num_words 결정 - 주제를 이루고 있는 단어의 개수





최종 num_words 설정 - 9

관측 및 불용어 추가 <main.py 114번째줄>

```
114    stoplist=["제","걸","번","그것","요","거","온","함","분","하다","임","것","다시다","전","치","뿌","링클"]
```

몇번의 코드 실행 후 Topic들을 관찰한다. 의미 없는 단어가 있다면

불용어를 추가해줘서 Topic Modeling의 결과가 더 정확하게 나올 수 있도록 해준다.

3. 결과 및 결과 해석

Main.py의 실행 결과 (치킨-전체-나쁨)

```
(0, '0.033*"배달" + 0.025*"치킨" + 0.025*"맛" + 0.021*"튀김" + 0.017*"옷" + 0.013*"별로" + 0.009*"닭" + 0.009*"정도" + 0.009*"실망"')
(1, '0.025*"맛" + 0.022*"치킨" + 0.016*"양념" + 0.013*"배달" + 0.013*"무슨" + 0.013*"다시는" + 0.010*"전화" + 0.010*"반" + 0.010*"시간"')
(2, '0.037*"치킨" + 0.021*"배달" + 0.012*"그냥" + 0.012*"시간" + 0.007*"주문" + 0.007*"사장" + 0.007*"처음" + 0.007*"감자" + 0.007*"출발"')
(3, '0.022*"배달" + 0.016*"주문" + 0.014*"맛" + 0.014*"해서" + 0.014*"치킨" + 0.008*"때" + 0.008*"최악" + 0.008*"식" + 0.008*"해도"')
(4, '0.019*"기름" + 0.016*"살" + 0.016*"다음" + 0.013*"좀" + 0.011*"맛" + 0.010*"튀김" + 0.010*"건지" + 0.010*"양념" + 0.010*"느낌"')
(5, '0.027*"시간" + 0.023*"양" + 0.019*"배달" + 0.019*"튀김" + 0.019*"맛" + 0.012*"살" + 0.012*"시켰는데" + 0.012*"반" + 0.008*"시켜"')
```

0: 배달 치킨 맛 튀김 옷 별로 닭 정도 실망

1: 맛 치킨 양념 배달 무슨 다시는 전화 반 시간

2: 치킨 배달 그냥 시간 주문 사장 처음 감자 출발

3: 배달 주문 맛 해서 치킨 때 최악 식 해도

4: 기름 살 다음 좀 맛 튀김 건지 양념 느낌

5: 시간 양 배달 튀김 맛 살 시켰는데 반 시켜

결과를 문장으로 구성하기엔 부자연스럽고 만족스럽지 않다고 생각했다. 따라서 전체적인 문제점을 파악한후 코드를 수정해보기로 하였다.

최적 k값을 설정할 때 Coherence, Perplexity 수렴하지 않아서 좋은 결과가 나오지 않을 거라고 예상했다. Topic들이 겹치는 느낌이 들었고 Topic안의 단어들의 연관성이 떨어지는 느낌을 받았다.

이 값들을 안정시키고 더 좋은 결과를 도출하기 위해서 두 가지 가설을 세웠다. 첫번째로는 자료의 수가 너무 적어서 좋은 결과가 나오지 않았다. 두번째로는 데이터를 Vector화 하는 과정을 더 정교하게 진행해야 한다. 우선 너무 자료수가 부족한게 아닌가 싶어 전체 자료에 대해서 분석을 진행해봤지만 역시 원하는 결과처럼 깔끔한 결과가 나오지 않았다.

따라서 가장 큰 이유는 바로 Vector화하는 과정에 있다고 생각했다. 단어의 출현 횟수로 Vector 화 한 기존 방식이 아닌 가중치를 주는 Tf-idf 방식을 적용해 봤다.

main_2.py 실행결과 (치킨-전체-나쁨)

<최적 K값은 최적 k값_2.py를 토대로 계산했다.>

<Main_2.py 실행 결과 (passes: 30, num_topics: 8, num_words: 10)>

0: 여기 처음 먹다 양념 치킨 묻다

1: 치킨 맛 그냥 배달 시키다 식다 오다

2: 치킨 시키다 맛 보다 튀김 옷 기름 버리다 남기다

3: 배달 주문 시키다 치킨 먹다 양 양념 별로 시간 넘기다

4: 배달 시간 넘다 양념 살 별로

5: 정말 소스 별로 배달 식다

6: 배달 전화 주소 치킨 남기다

7: 튀김 뒷맛 기름 색깔 감자칩 건지다

결과를 토대로 유의미 하다고 생각되는 Topic을 임의로 정제과정을 거쳐 문장으로 만들어 보았다.

최종 결과 (0,6,7 제외)

1. 치킨 맛 그냥 그렇다. 배달시켰는데 식어서 왔다.

2. 치킨 시켰는데 맛보다도 튀김 옷에 기름 때문에 다 남겨서 버렸다.

3. 주문시켰는데 치킨 양, 양념 별로고 배달 시간을 넘겼다.

4. 배달 시간 넘겼고 양념이랑 살 별로다.

5. 정말 소스 별로고 배달오는 과정에서 식어서 왔다.

비슷한 결과지만 전보다 더 좋은 결과가 도출되었다고 생각한다.

4. 결론 및 느낀점

결론 및 한계

2018년 5/9 이전 작성된 리뷰수가 500건 이상인 치킨집 중에 소비자가 '나쁨' 즉, 불만족한 데이터의 Topic Modeling을 토대로 도출한 최종 결과는 아래와 같다.

1. 치킨 맛 그냥 그렇다. 배달시켰는데 식어서 왔다.
2. 치킨 시켰는데 맛보다도 튀김 옷에 기름 때문에 다 남겨서 버렸다.
3. 주문시켰는데 치킨 양, 양념 별로고 배달 시간을 넘겼다.
4. 배달 시간 넘겼고 양념이랑 살 별로다.
5. 정말 소스 별로고 배달오는 과정에서 식어서 왔다.

이 결과 데이터를 토대로 도출한 2018년 5/9 이전 작성된 리뷰수가 500건 이상인 치킨집의 문제점은 대략 배달과정의 지연, 치킨의 양과 맛, 소스의 맛 등이라고 할 수 있다.

소비자의 입장에서 만약 위의 결과와 같이 카테고리의 전체 점포가 아닌 특정 업체를 선택한 후 이러한 데이터 얻게 되었다면 지금이 점포의 문제점에 대해서 알 수 있었을 것이다. 또한 소비자 불만이 아닌 만족을 선택했다면 이 가게의 장점에 대해 한 눈에 알 수 있었을 것이다. 앞으로 배달 산업이 더 상용화되고 커진다면 Topic Modeling을 통해서 앱 자체 내에서 프로젝트 결과와 같은 데이터를 제공한다면 효과적일 것이라고 생각된다.

창업을 준비하는 사업자의 입장에서는 치킨-전체-나쁨의 카테고리를 선택하게 되면 위와 같은 소비자 불만 데이터를 얻을 수 있고 창업할 때 어느정도 지표로 창업에 참고할 수 있는 효과를 기대할 수 있다.

현재 가게를 운영하고 있는 사업자에게는 Topic Modeling을 통해서 도출된 주제의 데이터를 제공함으로써 리뷰를 하나하나 분석하지 않아도 고객의 불만/만족의 데이터를 제공할 수 있다.

데이터를 전처리, 정제하는 과정에서 미숙한 부분이 존재하기 때문에 부자연스러운 문장이 도출된 것 같다. 이를 해결하기 위해서는 많은 모델을 구축해보고 더 정교한 어간추출, 불용처리 등의 과정이 필요할 것 같다. 또한 완성된 문장을 다시 부드럽게 변화시키기 위해 어간 추출한 것을 다시 어미를 붙여주는 과정을 거치면 좋을 것 같다. 마지막에 Topic Modeling을 통해 도출된 문장을 다시 nltk등 파이썬 module을 통해 어간들을 구분하고 구분된 어간들에 다시 어미를 붙여주는 과정을 자동화하면 좋을 것 같다.

더 많은 데이터가 있었다면 더 정교했을 것이라는 예상이 든다. 저번에 진행했던 개인과제1과 달리 이번에는 데이터가 너무 작아서 오히려 진행이 더 힘들었다는 생각이 든다.

Logistic 회귀 모델 말고 더 정확도가 높은 정교한 분류기를 사용한다면 Topic Modeling의 결과가 더 좋았을 것 같다.

느낀점 및 프로젝트 소감

결과가 잘 나와서 뿌듯했던 저번 프로젝트와 달리 이번 프로젝트에서는 데이터의 정제과정과 수정과정이 나름 즐거웠다. 나의 실생활과 밀접한 데이터를 정제하는 과정을 통해서 학문과 실생활이 융합되는 느낌이 들어서 정말로 무언가를 배운다는 느낌이 들었다.

실제로 '내 프로젝트가 실현화 되어 앱에 도입된다면 어떨까?'라는 생각을 해보게 되면서 이 프로젝트가 만약 나의 논문 프로젝트나 D.A를 수강하고 한학기 통째로 진행되는 프로젝트였으면 좋겠다라는 생각까지 했다. 여러 사람들과 의견을 나눠보고 더 정교하고 좋은 결과를 만들어보고 싶은 욕심이 났다.

프로젝트를 진행하고 나서 정말 DA라는 과목이 모든 분야에 이용되고 응용될 수 있을 것이라는 생각이 들었다. 경영자의 입장에서 매출이나 고객을 관리할때에도 사용가능 하다고 생각하고 나의 진로, 내가 하고 싶은 것들에 융합하고 실제로 응용될 수 있을 것 같은 생각이 들었고 확신이 들었다.

5학기 동안 배웠던 과목들 중에 가장 실용적이고 흥미로운 프로젝트였던 것 같다. 결과가 보이지 않는 이론적인 것들 사이에서 나의 실생활과도 관련이 있고 눈에 보이다 보니까 더 와닿고 흥미가 생겼던 것 같다.

5. 참고 문헌

Insight Korea – Insight Korea Big Data Solution 및 서비스: Insight Deep MininG i-VOC 서비스.pdf

고려대학교 산업경영공학과 김태경, 최하련, 이홍철. (2016). 토픽 모델링을 이용한 핀테크 기술 동향 분석.

통계청 - 배달앱_및_배달대행_이용현황_20200523.xlsx

Insight - 월간토픽 2019. 5 24. 제 292-2호- Android Mobile App '배달의민족', '요기요' 4월 이용 행태 분석