오픈소스전문프로젝트

-2차 데이터분석 결과보고서-

개요

1차로 제출했던 데이터 수집 및 분석 계획서의 경우 사용자를 비슷한 연령, 성별, 나이 등으로 묶어 선호 컨텐츠를 묶는 방법을 적용할 계획이었지만, 테스트 데이터를 수집하기 위한 어려움과 데이터 분석의 개념과는 조금 거리가 먼 측면이 있다고 느껴져 이번에 새로이 python의 sklearn과 gensim등의 라이브러리를 사용해서 임의의 사용자 히스토리 데이터를 기반으로 유사한 음식점을 추천하는 방법을 적용하기로 했다. Doc2Vec을 사용해 학습시킬 데이터를 임베딩 하였고, 임베딩 데이터를 cosine similarity를 사용해 유사한 데이터를 선택하는 방법과 임베딩 데이터에 딥러닝을 적용하는 방법까지 두 가지 방법을 준비했다.

Dataset

추천시스템을 작성함에 있어서 사용할 음식점 DB는 다음과 같다.

```
In [3]: data = pd.read_csv('./test_2.csv', encoding='cp949')

In [4]: data.head(5)
Out[4]:
```

	shop_id	업소명	영업상태	업종	주소	Y	X	형태소분석	feature
0	0	원가네식당	운영중	분식	경상북도 포항시 북구 흥해읍 중성로32번길 23-2, 1층	129.347187	36.106447	['원가', '식당']	원가네식당 분식 원가 식당
1	1	파티하하	운영중	경양식	경기도 파주시 금빛로 24, 4층 404호 (금촌동, 우령프라자)	126.766882	37.751989	['파티']	파티하하 경양식 파티
2	2	미(味)	운영중	일식	경기도 남양주시 두물로27번길 30, 1층 우측(102)호 (별내동)	127.126366	37.658216	['미']	미(味) 일식 미
3	3	소담왕족발	운영중	식육(숯불구이)	충청북도 청주시 상당구 낙영로24번길 11 (용암동,(1층))	127.509187	36.617471	['소', '소담왕족', '담', '왕족']	소담왕족발 식육(숯불구이) 소 소담왕족 담 왕족
4	4	뉴샤방샤방	운영중	호프/통닭	서울특별시 송파구 새말로 143, 지상1층 (문정동)	127.129575	37.483968	['뉴', '뉴샤방샤방', '샤방샤', '방']	뉴샤방샤방 호프/통닭 뉴 뉴샤방샤방 샤방샤 방

```
In [5]: data.shape
Out[5]: (90662, 9)
```

전국에 분포한 음식점 중에서 주소정보로 경위도 좌표를 수집할 수 있었던

90662개의 음식점의 업종, 이름에 대한 데이터로 아쉽게도 메뉴나 고객 리뷰와 같은 데이터를 확보할 수는 없었다.

데이터셋은 음식점 별 고유한 id값인 'shop_id', 업소명과 업종의 텍스트 데이터를 형태소 분석 과정을 거쳐서 만든 텍스트 정보인 'feature'가 분석 데이터로 사용된다. Feature행에 업소명이 들어가는 이유로는 많은 가게들이 가게의 이름에 대표적인 메뉴, 프랜차이즈 가게의 특징 등을 담고 있기 때문에 '버거킹' 이라는 패스트푸드점과 유사한 업소를 찾을 때 '버거킹'이라는 단어 혹은 '버거'라는 단어가 그 유사성을 나타낼 수 있기 때문에 업소명에 대한 정보도 feature로 활용하기로 했다.

Recommender System_1

먼저 Doc2Vec이란 문서를 지정한 차원의 벡터로 표현하는 임베딩 방법이다. 위에서 데이터셋의 feature행은 형태소 분석 과정을 거친 텍스트로 예를 들어 '신하수제숯불갈비' 라는 가게의 feature행의 정보는 [신하수제숯불갈비 식육(숯불구이) 신하 신하수제숯불갈비 수제 숯불 갈비] 가 된다. 이러한 feature 행의 텍스트를 적절한 처리과정을 거쳐 Doc2Vec 모델에 데이터를 입력하면 모델은 각 문서를 통해 단어들의 연관성을 학습하고 비슷한 단어는 비슷한 벡터를 가지도록하고, 그 결과를 토대로 문서의 벡터가 나타나는데, 다른 단어로 이루어진 문서라도 단어들의 벡터가 유사하다면 문서의 벡터도 유사하게 나타나는 등의 결과를 얻을 수 있다. 이번 보고서에선 각 feature를 100차원의 임베딩 벡터로 만들었다.

```
def make_doc2vec_models(doc_data, vector_size=100, window=2, epochs=40, min_count = 0, workers = 4):
    model = Doc2Vec(doc_data, vector_size=vector_size, window=window, epochs=epochs, min_count=min_count, workers=workers)
    model.save(f'./datas/model.doc2vec')
```

```
def make_doc2vec_data(data, column):
    data_doc = []
    for tag, doc in zip(data.index, data[column]):
        doc = doc.split(" ")
        data_doc.append(([tag], doc))
    data = [TaggedDocument(words=text, tags=tag) for tag, text in data_doc]
    return data
```

```
data_doc_title_content = make_doc2vec_data(data, 'feature')
make_doc2vec_models(data_doc_title_content)
```

Make_doc2vec_data는 feature 데이터와 각 문서를 인덱스 번호와 함께 태그로 묶어서 모델에 데이터를 넣기 전에 일종의 처리과정을 위한 함수인데, 결과는 다

음과 같다.

```
data_doc_title_content
```

```
[TaggedDocument(words=['원가네식당', '분식', '원가', '식당'], tags=[0]),
 TaggedDocument(words=['파티하하', '경양식', '파티'], tags=[1]),
 TaggedDocument(words=['미(味)', '일식', '미'], tags=[2]),
 TaggedDocument(words=['소담왕족발', '식육(숯불구이)', '소', '소담왕족', '담', '왕족'], tags=[3]),
 TaggedDocument(words=['뉴사방사방', '호프/통닭', '뉴', '뉴사방사방', '사방사', '방'], tags=[4]),
 TaggedDocument(words=['코바코(장기점)', '일식', '코', '코바코', '바', '장', '장기점', '기점'], tags
=[5]),
 TaggedDocument(words=['의령소바', '까페', '의', '소', '소바', '바'], tags=[6]),
 TaggedDocument(words=['도니버거', '대구서문시장점', '패스트푸드', '버거', '대구', '대구서문시장',
'서문', '시장'], tags=[7]),
 TaggedDocument(words=['반포식스', '일식', '반', '반포식', '포식'], tags=[8]),
 TaggedDocument(words=['김밥나라칠전점', '김밥(도시락)', '김밥', '김밥나라', '나라', '전점'], tags=
[9]),
 TaggedDocument(words=['의령', '일식', '의'], tags=[10]),
 TaggedDocument(words=['교토플러스', '호프(소주방)', '교', '교토플러스', '토', '플러스'], tags=[1
1]),
 TaggedDocument(words=['오동동부엉이집', '일식', '오동', '오동동', '오동동부엉이집', '동', '부엉이',
'집'], tags=[12]),
 TaggedDocument(words=['넙죽이족발', '식육취급', '족발'], tags=[13]),
 TaggedDocument(words=['사바루횟집', '일식', '사바', '횟집'], tags=[14]),
```

이제 위의 데이터를 make_doc2vec_model 함수에 입력해서 학습 모델을 만들고 0번 인덱스에 있던 '원가네식당'의 데이터를 출력하면 다음과 같다.

```
model_title_content = Doc2Vec.load('./datas/model/doc2vec')
np.save('./datas/dataset.npy', model_title_content.wv.vectors)
```

```
model_title_content[0]
```

```
array([-0.03917537,  0.10594855,  0.10233715,  0.14252079,  0.02598151,
        0.04260526, -0.02632715,  0.0855971 , -0.14473706,  0.01543656,
       -0.01641685, -0.08261634,  0.05622625, -0.06977849,  0.02035714,
       -0.00701176,  0.16771846,  0.093465  ,  0.12025268, -0.1396037 ,
        0.10957785,  0.04916864,  0.00774175,  0.12012922,  0.04323043,
       -0.13733995,  0.0358238 ,  0.03042283,  0.15451412, -0.11642642,
       -0.16696267,  0.01758105, -0.0946425 ,  0.05311266,  0.14573146,
        0.07317635, -0.15047391, -0.01905679,  0.12747353,  0.13634197,
       -0.07460263,  0.13623299,  0.11613204,  0.12793155,  0.03663867,
        0.0668996 , -0.0347681 ,  0.03853299, -0.06756265,  0.11693189,
       -0.05292255,  0.06427879,  0.04867439,  0.08822783,  0.04672954,
       -0.01989117,  0.02258473,  0.04630484,  0.01331522,  0.14216097,
        0.17179753, -0.10450093, -0.00092074,  0.01024409,  0.08671535,
        0.05224331,  0.0328327 , -0.14112736, -0.18017727, -0.14782548,
       -0.16998562, -0.01919007, -0.05519172,  0.03229369,  0.25814515,
       -0.01537122,  0.09029192, -0.04205963,  0.03411492, -0.08963239,
       -0.11395849, -0.05380706,  0.10595147,  0.10932746,  0.10243347,
        0.05868003,  0.01911516,  0.10018782, -0.08915866, -0.02588548,
       -0.09305958, -0.1490807 , -0.07941093, -0.07859728,  0.01467021,
        0.14630428,  0.11885983, -0.18986966, -0.19702983, -0.07588882],
      dtype=float32)
```

추천 시스템의 목적은 위와 같은 임베딩 벡터와 유사한 음식점을 찾아내는 것이다. Gensim에서 제공하는 Doc2Vec의 내장함수에서 most_similarity 메소드를 사용해 원가네식당과 가장 유사한 문서가 무엇인지 찾아본 결과이다.

```
model_title_content.wv.most_similar('원가네식당')
```

```
[('또아식당', 0.949655294418335),
 ('왔다식당', 0.9483097791671753),
 ('통큰식당', 0.9429266452789307),
 ('다올식당', 0.9349324107170105),
 ('또또와식당', 0.9309077858924866),
 ('일봉', 0.9198901653289795),
 ('둥글레식당', 0.9138662219047546),
 ('경인밀면', 0.9130666851997375),
 ('정김밥&누들마리', 0.9101676940917969),
 ('신세대식당', 0.9094788432121277)]
```

위의 결과를 보면 업소명에 '식당'이라는 단어를 포함하는 업종이 '분식'으로 분류된 음식점들이 유사도가 높다고 나온 것을 볼 수 있다. 업종과 업소명의 텍스트 데이터에 의존해 유사도를 구하는 방법이라 근본적으로 위의 음식점들이 원가네식당과 유사한 식당이라고 말할 수는 없다.

이제 테스트를 위해서 임의의 사용자가 방문한 15개의 식당의 히스토리 데이터를 통해 각 식당의 임베딩 벡터를 평균화하고 해당 벡터와 유사한 식당을 15개 출력하는 과정이다.

```
]: def make_user_embedding(index_list, data_doc, model):
    user = []
    user_embedding = []
    for i in index_list:
        user.append(data_doc[i][1][0])
    for i in user:
        user_embedding.append(model.docvecs[i])
    user_embedding = np.array(user_embedding)
    user = np.mean(user_embedding, axis=0)
    return user
```

```
]: def get_recommened_contents(user, data_doc, model):
    scores=[]

    for text, tags in data_doc:
        trained_doc_vec = model.docvecs[tags[0]]
        scores.append(cosine_similarity(user.reshape(-1, 100), trained_doc_vec.reshape(-1, 100)))

    scores = np.array(scores).reshape(-1)
    scores = np.argsort(-scores)[:15]

    return data.loc[scores, :]
```

함수는 사용자가 방문한 히스토리 데이터를 임베딩하고 이를 평균화하는 작업을 make_user_embedding이 수행하고 get_recommende_contents는 그 결과값 벡터에 코사인 유사도를 적용해 추천리스트를 만든다.

Test data

```
user_1=make_user_embedding(user,index.values.tolist(), data_doc_title_content, model_title_content)
user
```

	shop_id	업소명	영업상태	업종	주소	Y	X	형태소분석	feature
0	1	원가네식당	운영중	분식	경상북도 포항시 북구 흥해읍 중성로32번길 23-2, 1층	129.347187	36.106447	['원가', '식당']	원가네식당 분식 원가 식당
1	2	파티하하	운영중	경양식	경기도 파주시 금빛로 24, 4층 404호 (금촌동, 우평프라자)	126.766882	37.751989	['파티']	파티하하 경양식 파티
2	3	미(味)	운영중	일식	경기도 남양주시 두물로27번길 30, 1층 우측(102)호 (별내동)	127.126366	37.658216	['미']	미(味) 일식 미
3	4	소담왕족발	운영중	식육(숯불구이)	충청북도 청주시 상당구 낙영로24번길 11 (용암동,(1층))	127.509187	36.617471	['소', '소담왕족', '담', '왕족']	소담왕족발 식육(숯불구이) 소 소담왕족 담 왕족
4	6	코바코(장기점)	운영중	일식	경기도 김포시 김포한강1로51번길 46, 102호 (장기동,나래프라자)	126.666114	37.644548	['코', '코바코', '바', '장', '장기점', '기점']	코바코(장기점) 일식 코 코바코 바 장 장기점 기점
5	7	의령소바	운영중	까페	경기도 평택시 평택5로34번길 40, 1층 (합정동)	127.114104	36.989206	['의', '소', '소바', '바']	의령소바 까페 의 소 소바 바
6	9	반포식스	운영중	일식	서울특별시 서초구 신반포로 257, 지하2층 (잠원동, 신반포11차상가)	127.011906	37.508596	['반', '반포식', '포식']	반포식스 일식 반 반포식 포식
7	11	의령	운영중	일식	부산광역시 부산진구 동평로 416 (양정동)	129.070112	35.173082	['의']	의령 일식 의
8	12	교토플러스	운영중	호프(소주방)	전라남도 목포시 평화로73번길 25 (상동,,2층)	126.431205	34.798439	['교', '교토플러스', '토', '플러스']	교토플러스 호프(소주방) 교 교토플러스 토 플러스
9	13	오동동부엉이집	운영중	일식	경상남도 창원시 성산구 마디미로3번길 8 (상남동,청송빌딩 201호)	128.686215	35.223225	['오동', '오동동', '오동동부엉이집', '동', '부엉이', '집']	오동동부엉이집 일식 오동 오동동 오동동부엉이집 동 부엉이 집
10	14	넙죽이족발	운영중	식육취급	경기도 안산시 상록구 본오로 56, 1층 (본오동)	126.866523	37.290002	['족발']	넙죽이족발 식육취급 족발
11	15	사바루횟집	운영중	일식	서울특별시 은평구 불광로 60, 3층 (불광동)	126.932157	37.612242	['사바', '횟집']	사바루횟집 일식 사바 횟집
12	16	허생	운영중	일식	서울특별시 동대문구 천호대로 15 (신설동)	127.025238	37.574880	['허']	허생 일식 허
13	17	엔야	운영중	일식	서울특별시 은평구 응암로11길 10, 1층 (응암동)	126.915805	37.588594	['엔', '엔야', '야']	엔야 일식 엔 엔야 야
14	36	스시카츠	운영중	일식	경상남도 통영시 광도면 죽림4로 23-72, 1층	128.418410	34.885476	['스시', '스시카츠', '카', '츠']	스시카츠 일식 스시 스시카츠 카츠

Result

```
result =get_recommened_contents(user_1, data_doc_title_content, model_title_content)
```

```
pd.DataFrame(result)
```

	업소명	업종	X	Y	feature
45707	만석궁	일식	37.563877	126.962741	만석궁 일식 석궁
56803	고쌈촌	일식	37.568886	126.801818	고쌈촌 일식 쌈촌
30851	위아래복집	일식	36.345921	127.395135	위아래복집 일식 위아래 위아래복집 복 집
51621	젠슈야	일식	37.569065	126.986328	젠슈야 일식 젠슈
56166	호래오육점	일식	35.896142	128.612939	호래오육점 일식 호래 호래오육 오육
70593	해남수산	일식	37.492620	127.111698	해남수산 일식 남수
1368	아중이조참치	일식	35.828988	127.172502	아중이조참치 일식 중이 조참
74825	로바다야 카덴	일식	37.553711	126.918314	로바다야 카덴 일식 로바다
7011	지구당(地球堂)	일식	37.478262	126.952767	지구당(地球堂) 일식 지구당
73267	마고꼬로	일식	37.526013	127.051420	마고꼬로 일식 꼬로
83725	로기(rrokii)	일식	37.511603	127.032283	로기(rrokii) 일식 로기
69557	구화산	일식	37.468360	127.044270	구화산 일식 구화
76827	한야(HANYA)	일식	35.155475	126.848994	한야(HANYA) 일식 한야
7482	항도스시	일식	37.731086	127.057852	항도스시 일식 항도
30404	죠문	일식	37.531071	127.080286	죠문 일식 죠문

● 평가

임의로 작성한 사용자의 15개 음식점 방문기록은 일식집을 주로 방문한 데이터로 이러한 데이터들의 임베딩 벡터의 평균 또한 일식이라는 단어에 의해 추천 음식점이 업종이 일식인 음식점들이 추천되었다. 굉장히 단순한 아이디어와 부족한 데이터셋으로 신뢰도가 낮은 추천 시스템이며 지정한 좌표로부터 떨어진 거리가 추천 요소에 영향을 미치지 못하기 때문에 약속장소를 지정하고 그 주변의 음식점을 추천하는 시스템이 되기엔 기능적으로 부족하다. 추가적으로 거리에 대한 부분을 고려하여 추천할 수 있다면, 혹은 사용자의 리뷰 데이터를 수집해서 feature를 좀 더 다양하게 만들 수 있다면 추천시스템의 성능을 높일 수 있을 것이라고 생각한다.

Recommender System_2

이번에는 딥러닝을 활용한 방법이다. 앞서 활용한 90662개의 음식점 DB와 Recommender System_1에서 학습시킨 Doc2Vec 모델도 load하여 모델을 테스트 한다.

먼저 모델의 학습과 테스트를 위해 충분한 양의 방문기록 데이터를 만들어 준다. 100000개의 데이터 중에서 2할은 무작위로 선택한 일식집이고 나머지는 업종까지 무작위인 음식점이다. 딥러닝 모델의 X_train은 약 100000개의 데이터에서 음식점 DB의 shop_id와 대응하는 임베딩 벡터 값이며 y_train은 방문을 했는지, 하지 않았는지에 대한 binary label이다. 방문 여부는 방문하지 않은 경우가 7할 방문한 경우가 3할로 설정하였다. 이렇게 설정한 이유는 약 3할의 방문데이터에서 2할은 일식집이고 1할은 업종 무작위 음식점으로 3회의 외식에서 2회는 일식집에 방문하는 사용자의 히스토리를 임의로 만든 것이다.

```
user_history=pd.DataFrame(index=range(0,100000), columns=['user_id', 'shop_id'])
cnt=0
for i in range(1000):
    for j in range(100):
        new_user='user_'+str(i)
        if(j>=80):
            new_data=data.loc[random.sample(data.loc[data['업종']=='일식', :].index.values.tolist(), 1), :]
            new_data={'user_id':new_user, 'shop_id':new_data.index.values[0]}
        else:
            new_data = {'user_id':new_user, 'shop_id':random.randrange(0,90662)}
        user_history.iloc[cnt]=new_data
        cnt=cnt+1
```

```
visit=[]
for i in range(1000):
    for k in range(70):
        visit.append(0)
    for k in range(30):
        visit.append(1)
```

```
user_history['visit']=visit
```

```
user_history.tail(35)
```

	user_id	shop_id	visit
99965	user_999	19832	0
99966	user_999	66040	0
99967	user_999	26591	0
99968	user_999	6600	0
99969	user_999	90074	0
99970	user_999	28185	1
99971	user_999	6745	1
99972	user_999	17462	1
99973	user_999	82836	1
99974	user_999	46886	1
99975	user_999	51292	1
99976	user_999	70713	1

앞서 설명했듯이 X_train은 위의 user_history의 shop_id와 대응하는 Doc2Vec 임베딩 벡터가 되고, visit 행은 y_train이 된다.

```
In [43]: history_emb=[]
         for i in range(100000):
             history_emb.append(model_title_content[user_history.iloc[i]['shop_id']])

In [44]: history_emb=np.array(history_emb)

In [45]: history_emb.shape
Out[45]: (100000, 100)

In [46]: history_visit=user_history['visit']
         history_visit.shape
Out[46]: (100000,)
```

```
[47]: def keras_model():
          user_vector_input = Input(shape=(100, ))

          dense_u_v = Dense(100, activation = 'relu')(user_vector_input)
          dense_u_v = Dropout(0.3)(dense_u_v)
          dense_u_v = Dense(50, activation = 'relu')(dense_u_v)
          dense_u_v = Dropout(0.3)(dense_u_v)
          dense_u_v = Dense(25, activation = 'relu')(dense_u_v)
          dense_u_v = Dropout(0.3)(dense_u_v)
          dense_u_v = Dense(15, activation = 'relu')(dense_u_v)
          dense_u_v = Dropout(0.3)(dense_u_v)
          dense_u_v = Dense(5, activation = 'relu')(dense_u_v)
          dense_u_v = Dense(1, activation = 'sigmoid')(dense_u_v)

          model = Model(inputs=user_vector_input, outputs=dense_u_v)
          model.compile(optimizer = 'Adam', loss='binary_crossentropy', metrics=['acc'])
          return model
```

모델은 단순히 Dense layer만 쌓은 형태로 처음에는 100개의 input을 받고 점점 작아져서 마지막은 sigmoid로 0, 1을 판단하도록 했다.

- **Model Test**

```
4]: model = keras_model()

5]: X_train, X_test, y_train, y_test = train_test_split(history_emb, history_visit, test_size = 0.2, random_state=1)

6]: modelpath = './datas/recommender.model'
checkpointer = ModelCheckpoint(filepath = modelpath, monitor='val_loss', verbose=1, save_best_only=True)
early_stop = EarlyStopping(monitor='val_loss', patience=3)

7]: hist=model.fit(X_train, y_train, validation_split =0.2, epochs=50, batch_size=25, callbacks=[early_stop, checkpointer])
Train on 64000 samples, validate on 16000 samples
Epoch 1/50
64000/64000 [==============================] - 4s 64us/step - loss: 0.3210 - acc: 0.8543 - val_loss: 0.2795 - val_acc: 0.8742

Epoch 00001: val_loss improved from inf to 0.27951, saving model to ./datas/recommender.model
Epoch 2/50
64000/64000 [==============================] - 4s 57us/step - loss: 0.2836 - acc: 0.8735 - val_loss: 0.2687 - val_acc: 0.8772

Epoch 00002: val_loss improved from 0.27951 to 0.26865, saving model to ./datas/recommender.model
Epoch 3/50
64000/64000 [==============================] - 4s 68us/step - loss: 0.2747 - acc: 0.8760 - val_loss: 0.2643 - val_acc: 0.8780

Epoch 00003: val_loss improved from 0.26865 to 0.26427, saving model to ./datas/recommender.model
Epoch 4/50
64000/64000 [==============================] - 4s 56us/step - loss: 0.2681 - acc: 0.8784 - val_loss: 0.2603 - val_acc: 0.8806

Epoch 00004: val_loss improved from 0.26427 to 0.26031, saving model to ./datas/recommender.model
Epoch 5/50
64000/64000 [==============================] - 4s 64us/step - loss: 0.2657 - acc: 0.8788 - val_loss: 0.2587 - val_acc: 0.8808

Epoch 00005: val_loss improved from 0.26031 to 0.25873, saving model to ./datas/recommender.model
Epoch 6/50
64000/64000 [==============================] - 4s 68us/step - loss: 0.2627 - acc: 0.8801 - val_loss: 0.2555 - val_acc: 0.8827

Epoch 00006: val_loss improved from 0.25873 to 0.25548, saving model to ./datas/recommender.model
Epoch 7/50
64000/64000 [==============================] - 4s 63us/step - loss: 0.2601 - acc: 0.8817 - val_loss: 0.2576 - val_acc: 0.8819

Epoch 00007: val_loss did not improve from 0.25548
Epoch 8/50
64000/64000 [==============================] - 4s 60us/step - loss: 0.2602 - acc: 0.8814 - val_loss: 0.2561 - val_acc: 0.8824

Epoch 00008: val_loss did not improve from 0.25548
Epoch 9/50
64000/64000 [==============================] - 4s 63us/step - loss: 0.2579 - acc: 0.8817 - val_loss: 0.2557 - val_acc: 0.8821

Epoch 00009: val_loss did not improve from 0.25548
```

테스트 결과는 다음과 같은데, epochs를 50회로 설정했지만 10회 반복 이전에 더 이상 loss가 떨어지지 않아 학습을 종료한 것을 볼 수 있다. 모델의 acc는 약 88퍼센트로 임의로 만든 dataset인데도 불구하고 loss가 어느정도 떨어지는 것을 볼 수 있었다.

- **평가**

위의 테스트 결과는 역시 임의로 만든 데이터이고 모델에 user의 profile 등의 정보가 input으로 주어지지 않았다는 점, 모델이 단순하고 Recommender System_1에서와 마찬가지로 거리 등의 요건도 input에 고려되지 않았다는 점 등이 문제로 보인다. 개선을 위해선 모델에 input 하는 데이터를 단순히 업소명, 업종의 임베딩 벡터만이 아니라 추가적인 사용자의 특징, 장소 등을 고려할 수 있도록 해야 할 것으로 보인다.