실제 음식점에서의 질의를와 claude가 생성한 답변의 유사도를 word2vec모델을 학습하여 각 답변의 벡터를 구하고, cosine유사도로 결과를 정량적으로 비교한다.

데이터 처리 및 claude 사용 준비

ai hub의 소상공인 고객응대 데이터셋(https://aihub.or.kr/aihubdata/data/view.do?currMenu=115&topMenu=100&aihubDataSe=data&dataSetSn=102)의 validation_음식점.csv를 이용하여 음식점에서의 질의를 담은 데이터를 정리하고,

claude sonnet의 사용을 준비한다.

[ ] ↳ *숨겨진 셀 14개*

데이터 load

사용한 소상공인 고객응대 데이터셋와 claude를 거쳐서 나온 결과를 저장한 데이터를 load

대상파일명 : data.pkl , model_result.pkl

```
import pickle

data = pickle.load(open('/content/data.pkl','rb'))
model_result = pickle.load(open('/content/model_result.pkl','rb'))
```

word2vec 한국어

위의 데이터들(claude의 답변과 사용한 데이터의 질문과 답변)을 이용해서 mecab으로 토크나이징 후 word2vec모델에 학습하여 이용예정

저장할 모델명 : kor_w2v

```
!pip install konlpy
!pip install python-mecab-ko
```

```
Requirement already satisfied: konlpy in /usr/local/lib/python3.10/dist-packages (0.6.0)
Requirement already satisfied: JPype1>=0.7.0 in /usr/local/lib/python3.10/dist-packages (from konlpy) (1.5.0)
Requirement already satisfied: lxml>=4.1.0 in /usr/local/lib/python3.10/dist-packages (from konlpy) (4.9.4)
Requirement already satisfied: numpy>=1.6 in /usr/local/lib/python3.10/dist-packages (from konlpy) (1.25.2)
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-packages (from JPype1>=0.7.0->konlpy) (24.0)
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the system package manager. It is recommended to use a virtual environment instead: http
Requirement already satisfied: python-mecab-ko in /usr/local/lib/python3.10/dist-packages (1.3.5)
Requirement already satisfied: python-mecab-ko-dic in /usr/local/lib/python3.10/dist-packages (from python-mecab-ko) (2.1.1.post2)
```

```
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the system package manager. It is recommended to use a virtual environment instead: http
```

```
from mecab import MeCab as mecab

# Mecab 로드 (또는 다른 형태소 분석기 사용 가능)
mecab = mecab()
word2vecTrain =  [mecab.morphs(sentence) for sentence in data['raw']]+[mecab.morphs(sentence) for sentence in data['label']]+[mecab.morphs(sentence) for sentence in model_result]

stop_words = set([
    "이", "는", "을", "있", "것", "있다", "수", "등", "의", "이다", "들", "그", "에서", "하다"
])

word2vecTrain =[[word for word in words if word not in stop_words] for words in word2vecTrain  ]


len(word2vecTrain)
```

```
21322
```

```
!pip install gensim
```

```
Requirement already satisfied: gensim in /usr/local/lib/python3.10/dist-packages (4.3.2)
Requirement already satisfied: numpy>=1.18.5 in /usr/local/lib/python3.10/dist-packages (from gensim) (1.25.2)
Requirement already satisfied: scipy>=1.7.0 in /usr/local/lib/python3.10/dist-packages (from gensim) (1.11.4)
Requirement already satisfied: smart-open>=1.8.1 in /usr/local/lib/python3.10/dist-packages (from gensim) (6.4.0)
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the system package manager. It is recommended to use a virtual environment instead: http
```

```
word2vecTrain[:3]
```

```
[['카드', '번호', '부를까요', '?'],
 ['참이슬', '도', '한', '병', '부탁', '드립니다', '.'],
 ['기본', '세팅', '과자', '안', '주', '셔도', '돼요', '.']]
```

```
from gensim.models import Word2Vec

model = Word2Vec(sentences=word2vecTrain, vector_size=100, window=5, min_count=1, workers=4)

from gensim.models import KeyedVectors

model.wv.save_word2vec_format('kor_w2v') # 모델 저장
```

비교

mecab을 통해 토크나이징한 문장 실제 대답데이터와 claude가 생성한 대답데이터를 위에서 학습한 모델로 벡터화하여 cosine 유사도를 구한다.

```
import numpy as np
from gensim.models import KeyedVectors
from sklearn.metrics.pairwise import cosine_similarity
from gensim.test.utils import datapath
from mecab import MeCab as mecab

model = KeyedVectors.load_word2vec_format("kor_w2v") # 모델 로드

# Mecab 로드 (또는 다른 형태소 분석기 사용 가능)
mecab = mecab()

# 문장 토큰화 및 임베딩 벡터 변환 함수
def sentence_to_vec(sentence, model):
    words = mecab.morphs(sentence)  # 형태소 분석기로 토큰화
    word_vecs = []

    for word in words:
        if word in model:
            word_vecs.append(model[word])

    if len(word_vecs) == 0:  # 만약 문장의 모든 단어가 모델에 없는 경우
        return np.zeros(model.vector_size)

    # 문장의 임베딩 벡터는 단어 벡터의 평균으로 계산
    sentence_vec = np.mean(word_vecs, axis=0)
    return sentence_vec

# cosine 유사도 계산 함수
def cosine_similarity_vec(vec1, vec2):
    vec1 = vec1.reshape(1, -1)
    vec2 = vec2.reshape(1, -1)
    return cosine_similarity(vec1, vec2)[0][0]


# test
# 두 문장 예제
sentence1 = "오늘 날씨가 좋다."
sentence2 = "날씨가 매우 맑다."

# 문장을 임베딩 벡터로 변환
vec1 = sentence_to_vec(sentence1, model)
vec2 = sentence_to_vec(sentence2, model)

# cosine 유사도 계산
similarity = cosine_similarity_vec(vec1, vec2)
print(f"Cosine 유사도: {similarity}")
```

```
Cosine 유사도: 0.9566093683242798
```

```
resultGraph = [0] * 101
print(type(resultGraph))
```

```
<class 'list'>
```

```
sum = 0
for num in range(len(model_result)):
  vec1 = sentence_to_vec(data['label'][num], model)
  vec2 = sentence_to_vec(model_result[num], model)

  similarity = cosine_similarity_vec(vec1, vec2)
  print(f"Cosine 유사도: {similarity}")

  sum += similarity

  resultGraph[int(similarity//0.01)] += 1
```

```
Cosine 유사도: 0.8692265748977661
Cosine 유사도: 0.8320180177688599
Cosine 유사도: 0.7581821084022522
Cosine 유사도: 0.7125447392463684
Cosine 유사도: 0.7878491878509521
Cosine 유사도: 0.8831192255020142
Cosine 유사도: 0.8819239735603333
Cosine 유사도: 0.7941224575042725
Cosine 유사도: 0.9184372425079346
Cosine 유사도: 0.9043839573860168
Cosine 유사도: 0.8607904314994812
Cosine 유사도: 0.9622786641120911
Cosine 유사도: 0.9312378764152527
Cosine 유사도: 0.8521316647529602
Cosine 유사도: 0.8846523761749268
Cosine 유사도: 0.8090696930885315
Cosine 유사도: 0.943999707698822
Cosine 유사도: 0.9781274795532227
Cosine 유사도: 0.8030695915222168
Cosine 유사도: 0.9449055194854736
Cosine 유사도: 0.7387276887893677
Cosine 유사도: 0.8968445658683777
Cosine 유사도: 0.9332382678985596
Cosine 유사도: 0.5034820437431335
Cosine 유사도: 0.6654075980186462
Cosine 유사도: 0.505500316619873
Cosine 유사도: 0.9338523149490356
Cosine 유사도: 0.7176447510719299
Cosine 유사도: 0.8310052752494812
Cosine 유사도: 0.7056047320365906
Cosine 유사도: 0.7369682192802429
Cosine 유사도: 0.8920966386795044
Cosine 유사도: 0.767967164516449
Cosine 유사도: 0.8929330110549927
Cosine 유사도: 0.8319000601768494
```

```
Cosine 유사도: 0.7230215668678284
Cosine 유사도: 0.8233462572097778
Cosine 유사도: 0.8329710364341736
Cosine 유사도: 0.9407063722610474
Cosine 유사도: 0.9346274137496948
Cosine 유사도: 0.8550809025764465
Cosine 유사도: 0.7914286255836487
Cosine 유사도: 0.9457863569259644
Cosine 유사도: 0.8601680994033813
Cosine 유사도: 0.7767636775970459
Cosine 유사도: 0.7685064077377319
Cosine 유사도: 0.9222552180290222
Cosine 유사도: 0.7920057773590088
Cosine 유사도: 0.9682343006134033
Cosine 유사도: 0.9665189385414124
Cosine 유사도: 0.7098554372787476
Cosine 유사도: 0.35647091269493103
Cosine 유사도: 0.939434289932251
Cosine 유사도: 0.8346763253211975
Cosine 유사도: 0.49137431383132935
Cosine 유사도: 0.7389897704124451
Cosine 유사도: 0.7975737452507019
Cosine 유사도: 0.7436763644218445
```

```
print('평균 유사도:',sum/len(model_result))
```

```
평균 유사도: 0.8312343618174394
```

```
import pickle

with open('result.pkl', 'wb') as f:
    pickle.dump(resultGraph, f)
```

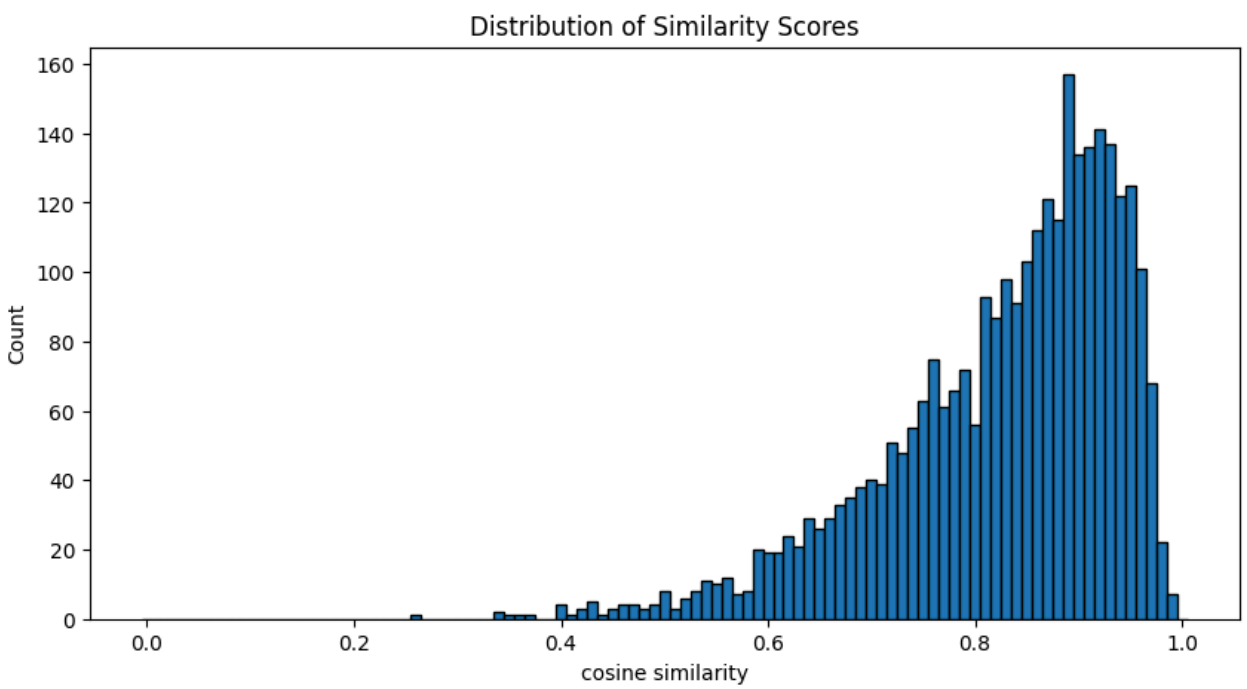
시각화

실제 라벨과 claude를 통해 생성한 답변의 유사도를 시각화 하였다.

코사인 유사도의 범위에 따라 정리한 데이터량 : result.pkl

```
import pickle

resultGraph = pickle.load(open('result.pkl','rb'))
```

```
import matplotlib.pyplot as plt
import numpy as np

plt.figure(figsize=(10, 5))

# 그래프 그리기
plt.bar(np.arange(0,1.01,0.01), resultGraph, width=0.01, edgecolor='black')



# 그래프 제목 및 축 레이블 추가
plt.title('Distribution of Similarity Scores')
plt.xlabel('cosine similarity')
plt.ylabel('Count')

# 그래프 표시
plt.show()
```



위 결과를 통해 대부분의 대답들이 실제 답변과 매우 높은 유사성을 보여준다.