# Search Engine Report

230750 Seo Jun

## 1. Project Purpose and Background

This project was undertaken to apply the knowledge learned in the seven weeks.

The objective was to practice practical implementation based on the lessons learned.

## 2. Objectives

To develop a basic search engine that retrieves sentences similar to the user's query

## 3. User Requirements

The system should be capable of searching for sentences similar to the user's query.

## 4. Functional Requirements

① Preprocess sentences within the search target and store them in a list.

② Receive an input English string (query) from the user and preprocess it.

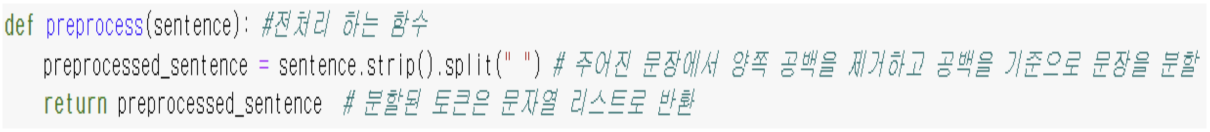
③ Calculate the similarity between the query and sentences within the search target • Similarity is based on the count of the same “word."

④ Rank the sentences based on similarity. And Output the top 10 ranked sentences to the user from the ranked sentences

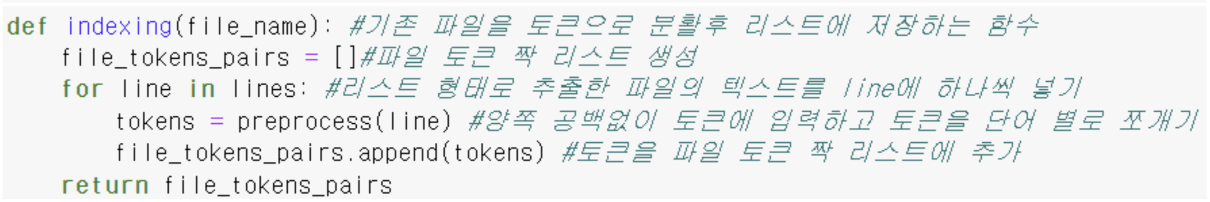
## 5. Code by Requirements

① Preprocess sentences within the search target and store them in a list.

1.



```python
def preprocess(sentence): #전처리 하는 함수
    preprocessed_sentence = sentence.strip().split(" ") # 주어진 문장에서 양쪽 공백을 제거하고 공백을 기준으로 문장을 분할
    return preprocessed_sentence  # 분할된 토큰은 문자열 리스트로 반환
```



```python
def indexing(file_name): #기존 파일을 토큰으로 분활후 리스트에 저장하는 함수
    file_tokens_pairs = []#파일 토큰 짝 리스트 생성
    for line in lines: #리스트 형태로 추출한 파일의 텍스트를 line에 하나씩 넣기
        tokens = preprocess(line) #양쪽 공백없이 토큰에 입력하고 토큰을 단어 별로 쪼개기
        file_tokens_pairs.append(tokens) #토큰을 파일 토큰 짝 리스트에 추가
    return file_tokens_pairs
```

```
# 1. Indexing
## https://github.com/jungyeul/korean-parallel-corpora
file_name = "jhe-koen-dev.en.txt" #파일 지정
file_tokens_pairs=indexing(file_name)
```

2. Input

- `sentence`: The target string to preprocess.
- `filename`: A variable representing the file name or path.
- `file_tokens_pairs`: A list to store tokens from the file.

3. Result

- `preprocess(sentence)` return value: Returns a list of split tokens as strings.
- `indexing(file_name)` return value: Returns a list created by tokenizing the target file.
- Tokenize the contents of the file named "jhe-koen-dev.en.txt" and create a list of token pairs.

4. Explanation

•`preprocess(sentence)`: This function is a preprocessing function that removes leading and trailing spaces and splits the sentence based on spaces. The split tokens are returned as strings.

•`indexing(file_name)`: This function tokenizes an existing file and stores it in a list. It calls the `preprocess(sentence)` function to add token pairs to the list.

•The file token pairs list contains preprocessed text read from the target file, with each line tokenized.

② Receive an input English string (query) from the user and preprocess it.

1.

```
# 2. Input the query
query = input("영어 쿼리를 입력하세요.").lower() #사용자가 문자 입력
preprocessed_query = query.strip().split(" ") #전처리 쿼리에 입력한 문자 공백없이 입력
query_token_set = set(preprocessed_query) #입력받은 토큰 세트에 집합 만들기
```

2. Input

•query: the string received from the user as input

3.Result

•preprocessed_query: into English is "The query string that has had leading and trailing spaces removed and has been split by spaces.

•query_token_set: A set created by preprocessing the query string and tokenizing it based on whitespace.

4. Explanation

•Receive the query input from the user.

•Remove leading and trailing spaces and create the preprocessed query string, `preprocessed_query`, split by spaces.

•Create a set from the preprocessed string to form `query_token_set`. Using a set allows for fast searching.

③ Calculate the similarity between the query and sentences within the search target • Similarity is based on the count of the same "word."

1.

```
def calc_similarity(preprocessed_query, preprocessed_sentences): #유사도를 계산하는 함수
    score_dict={} #유사도를 저장하기 위한 변수
    for i in range(len(preprocessed_sentences)):
        # 시작: 대소문자 구분 없는 토큰 셋을 만들기 위한 코드
        sentence = preprocessed_sentences[i] #문장을 가져오기
        query_str = ' '.join(preprocessed_query).lower() #전처리 된 쿼리를 소문자로 변환하고 문자열로 결합
        sentence_str = ' '.join(sentence).lower() #소문자로 변환하고 문자열로 결합
        preprocessed_query = set(preprocess(query_str)) #소문자로 변환한 쿼리를 전처리 하고 토큰 화하기
        preprocessed_sentence = preprocess(sentence_str)#소문자로 변환된 문장으로 전처리하기
        # 끝: 대소문자 구분 없는 토큰 셋을 만들기 위한 코드

        file_token_set = set(preprocessed_sentence) #토큰 집합 만들기
        all_tokens = preprocessed_query | file_token_set #쿼리와 문장의 토큰 합치기
        same_tokens = preprocessed_query & file_token_set #쿼리와 문장의 토큰 교집합 만들기
        similarity = len(same_tokens) / len(all_tokens)# 유사성 계산하기
        score_dict[i] = similarity #유사 점수를 변수에 저장
    return score_dict #유사 점수가 있는 사전 반환
```

```
# 3. Calculate similarities based on a same token set
score_dict = calc_similarity(query_token_set, file_tokens_pairs)#입력받은 문자열과 검색 대상 내 문장들 유사도 계산
```

2. Input

•preprocessed_query: Preprocessed query token set.

• preprocessed_sentences: List of preprocessed sentences.

3.Result

•Store the sentence number and the similarity between the sentence and the query in the score_dict.

4. Explanation

•Create the `score_dict` variable to store similarity scores.

•Create a case-insensitive token set and create a token set for the sentence to store in `file_token_set`.

•Calculate the similarity by dividing the intersection of all tokens and common tokens (i.e., the intersection) by the union of all tokens.

•Store the similarity of each sentence in the `score_dict` dictionary with the sentence number as the key.

•Use the `calc_similarity` function to calculate the similarity of sentences and store them in the `score_dict`.

④ Rank the sentences based on similarity. And Output the top 10 ranked sentences to the user from the ranked sentences

1.

```
# 4. Sort the similarity list
sorted_score_list = sorted(score_dict.items(), key = operator.itemgetter(1), reverse=True) #큰 순으로 정렬이 된 리스트 생성
# 5. Print the result
if sorted_score_list[0][1] == 0.0: # 만약 가장 유사한 문장의 점수가 0.0이면,
    print("There is no similar sentence.") #유사한 문장이 없다고 출력
else: #그게 아니라면
    print("rank", "Index", "score", "sentence", sep = "₩t") # ₩t간격으로 출력
    rank = 1 #랭크는 1로 초기화
    for i, score  in sorted_score_list: #i와 스코어에 튜플 형태의 아이템 가져오기
        print(rank, i, score, ' '.join(file_tokens_pairs[i]), sep = "₩t") #순위 인덱스 점수 문장 출력하기
        if rank == 10: # 상위 10개만 출력후
            break # 탈출
        rank = rank + 1 #순위를 증가 시키기
```

2. Input

•preprocessed_query: Preprocessed query token set.

• preprocessed_sentences: List of preprocessed sentences.

3.Result

•Return sorted_score_list: A list sorted in descending order based on similarity scores.

•Print the rank, index, score, and sentence for the top 10 sentences.

4. Explanation

• Preprocess the input query and sentences to create query tokens and sentence tokens.

• Calculate the similarity and store the similarity score and the index of the corresponding sentence in a dictionary.

• Sort the similarity dictionary in descending order based on scores to create sorted_score_list.

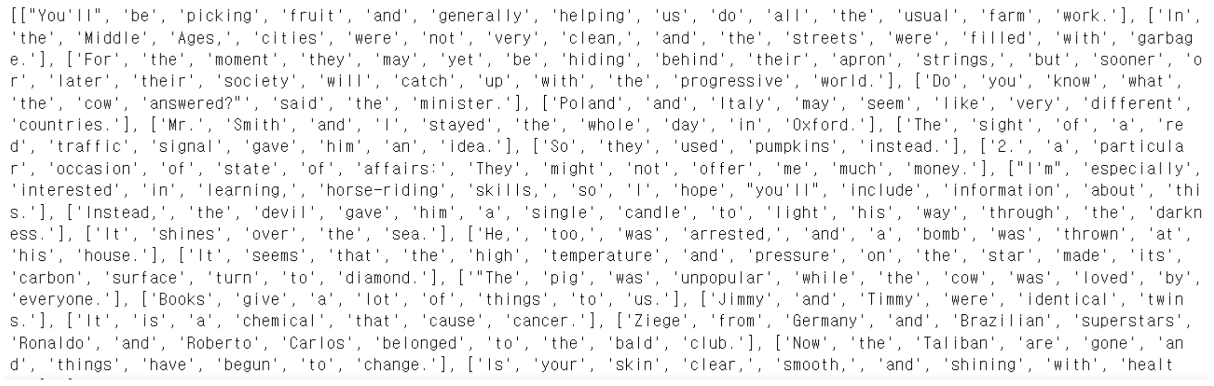
•If the score of the most similar sentence is 0.0, print the message "There is no similar sentence."

•Otherwise, print the rank, index, score, and sentence for the top 10 sentences.

6. Test

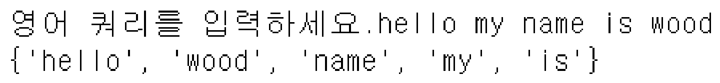
1. Test Results by Functionality

① Preprocess sentences within the search target and store them in a list.



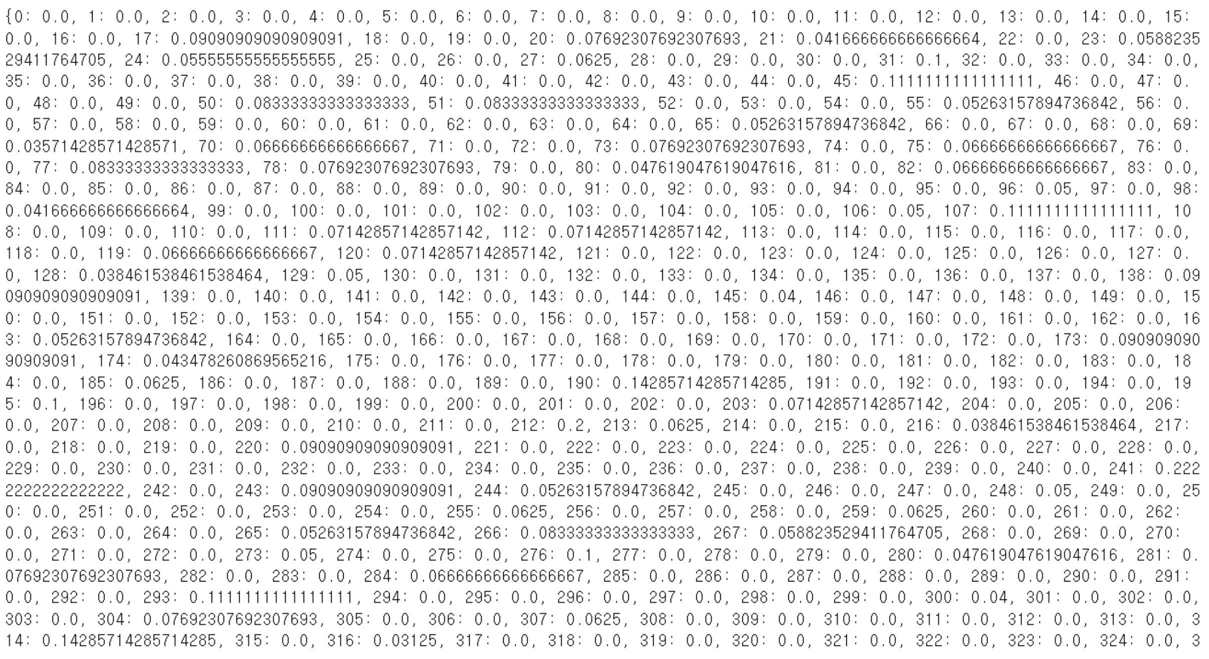
```
[["You'll", 'be', 'picking', 'fruit', 'and', 'generally', 'helping', 'us', 'do', 'all', 'the', 'usual', 'farm', 'work.'], ['In',
'the', 'Middle', 'Ages,', 'cities', 'were', 'not', 'very', 'clean,', 'and', 'the', 'streets', 'were', 'filled', 'with', 'garbag
e.'], ['For', 'the', 'moment', 'they', 'may', 'yet', 'be', 'hiding', 'behind', 'their', 'apron', 'strings,', 'but', 'sooner', 'o
r', 'later', 'their', 'society', 'will', 'catch', 'up', 'with', 'the', 'progressive', 'world.'], ['Do', 'you', 'know', 'what',
'the', 'cow', 'answered?"', 'said', 'the', 'minister.'], ['Poland', 'and', 'Italy', 'may', 'seem', 'like', 'very', 'different',
'countries.'], ['Mr.', 'Smith', 'and', 'I', 'stayed', 'the', 'whole', 'day', 'in', 'Oxford.'], ['The', 'sight', 'of', 'a', 're
d', 'traffic', 'signal', 'gave', 'him', 'an', 'idea.'], ['So', 'they', 'used', 'pumpkins', 'instead.'], ['2.', 'a', 'particula
r', 'occasion', 'of', 'state', 'of', 'affairs:', 'They', 'might', 'not', 'offer', 'me', 'much', 'money.'], ["I'm", 'especially',
'interested', 'in', 'learning,', 'horse-riding', 'skills,', 'so', 'I', 'hope', "you'll", 'include', 'information', 'about', 'thi
s.'], ['Instead,', 'the', 'devil', 'gave', 'him', 'a', 'single', 'candle', 'to', 'light', 'his', 'way', 'through', 'the', 'darkn
ess.'], ['It', 'shines', 'over', 'the', 'sea.'], ['He,', 'too,', 'was', 'arrested,', 'and', 'a', 'bomb', 'was', 'thrown', 'at',
'his', 'house.'], ['It', 'seems', 'that', 'the', 'high', 'temperature', 'and', 'pressure', 'on', 'the', 'star', 'made', 'its',
'carbon', 'surface', 'turn', 'to', 'diamond.'], ['"The', 'pig', 'was', 'unpopular', 'while', 'the', 'cow', 'was', 'loved', 'by',
'everyone.'], ['Books', 'give', 'a', 'lot', 'of', 'things', 'to', 'us.'], ['Jimmy', 'and', 'Timmy', 'were', 'identical', 'twin
s.'], ['It', 'is', 'a', 'chemical', 'that', 'cause', 'cancer.'], ['Ziege', 'from', 'Germany', 'and', 'Brazilian', 'superstars',
'Ronaldo', 'and', 'Roberto', 'Carlos', 'belonged', 'to', 'the', 'bald', 'club.'], ['Now', 'the', 'Taliban', 'are', 'gone', 'an
d', 'things', 'have', 'begun', 'to', 'change.'], ['Is', 'your', 'skin', 'clear,', 'smooth,', 'and', 'shining', 'with', 'healt
```

② Receive an input English string (query) from the user and preprocess it.



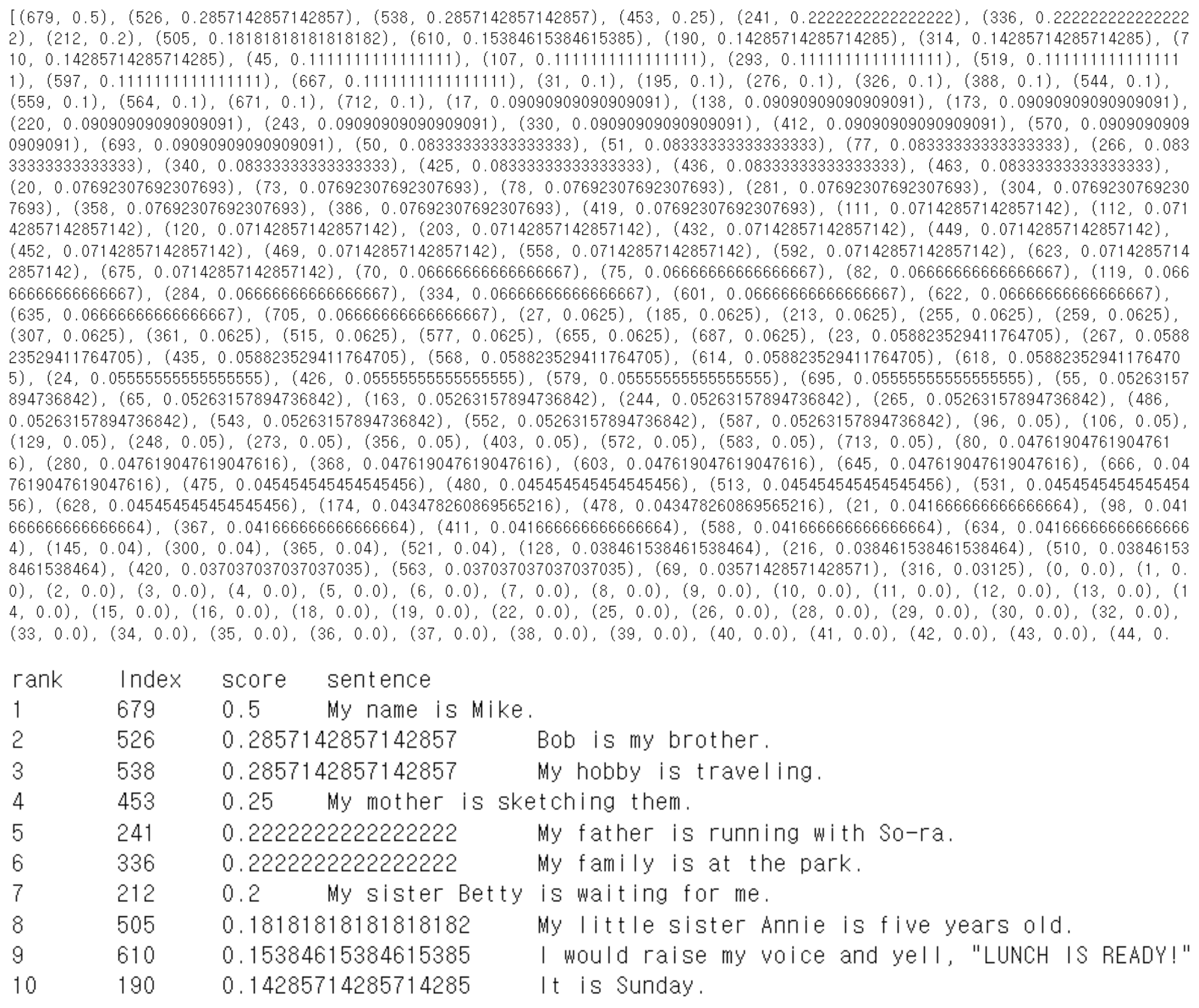
```
영어 쿼리를 입력하세요.hello my name is wood
{'hello', 'wood', 'name', 'my', 'is'}
```

③ Calculate the similarity between the query and sentences within the search target • Similarity is based on the count of the same “word."



```
{0: 0.0, 1: 0.0, 2: 0.0, 3: 0.0, 4: 0.0, 5: 0.0, 6: 0.0, 7: 0.0, 8: 0.0, 9: 0.0, 10: 0.0, 11: 0.0, 12: 0.0, 13: 0.0, 14: 0.0, 15:
0.0, 16: 0.0, 17: 0.09090909090909091, 18: 0.0, 19: 0.0, 20: 0.07692307692307693, 21: 0.041666666666666664, 22: 0.0, 23: 0.0588235
29411764705, 24: 0.05555555555555555, 25: 0.0, 26: 0.0, 27: 0.0625, 28: 0.0, 29: 0.0, 30: 0.0, 31: 0.1, 32: 0.0, 33: 0.0, 34: 0.0,
35: 0.0, 36: 0.0, 37: 0.0, 38: 0.0, 39: 0.0, 40: 0.0, 41: 0.0, 42: 0.0, 43: 0.0, 44: 0.0, 45: 0.1111111111111111, 46: 0.0, 47: 0.
0, 48: 0.0, 49: 0.0, 50: 0.08333333333333333, 51: 0.08333333333333333, 52: 0.0, 53: 0.0, 54: 0.0, 55: 0.05263157894736842, 56: 0.
0, 57: 0.0, 58: 0.0, 59: 0.0, 60: 0.0, 61: 0.0, 62: 0.0, 63: 0.0, 64: 0.0, 65: 0.05263157894736842, 66: 0.0, 67: 0.0, 68: 0.0, 69:
0.03571428571428571, 70: 0.06666666666666667, 71: 0.0, 72: 0.0, 73: 0.07692307692307693, 74: 0.0, 75: 0.06666666666666667, 76: 0.
0, 77: 0.08333333333333333, 78: 0.07692307692307693, 79: 0.0, 80: 0.047619047619047616, 81: 0.0, 82: 0.06666666666666667, 83: 0.0,
84: 0.0, 85: 0.0, 86: 0.0, 87: 0.0, 88: 0.0, 89: 0.0, 90: 0.0, 91: 0.0, 92: 0.0, 93: 0.0, 94: 0.0, 95: 0.0, 96: 0.05, 97: 0.0, 98:
0.041666666666666664, 99: 0.0, 100: 0.0, 101: 0.0, 102: 0.0, 103: 0.0, 104: 0.0, 105: 0.0, 106: 0.05, 107: 0.1111111111111111, 10
8: 0.0, 109: 0.0, 110: 0.0, 111: 0.07142857142857142, 112: 0.07142857142857142, 113: 0.0, 114: 0.0, 115: 0.0, 116: 0.0, 117: 0.0,
118: 0.0, 119: 0.06666666666666667, 120: 0.07142857142857142, 121: 0.0, 122: 0.0, 123: 0.0, 124: 0.0, 125: 0.0, 126: 0.0, 127: 0.
0, 128: 0.038461538461538464, 129: 0.05, 130: 0.0, 131: 0.0, 132: 0.0, 133: 0.0, 134: 0.0, 135: 0.0, 136: 0.0, 137: 0.0, 138: 0.09
090909090909091, 139: 0.0, 140: 0.0, 141: 0.0, 142: 0.0, 143: 0.0, 144: 0.0, 145: 0.04, 146: 0.0, 147: 0.0, 148: 0.0, 149: 0.0, 15
0: 0.0, 151: 0.0, 152: 0.0, 153: 0.0, 154: 0.0, 155: 0.0, 156: 0.0, 157: 0.0, 158: 0.0, 159: 0.0, 160: 0.0, 161: 0.0, 162: 0.0, 16
3: 0.05263157894736842, 164: 0.0, 165: 0.0, 166: 0.0, 167: 0.0, 168: 0.0, 169: 0.0, 170: 0.0, 171: 0.0, 172: 0.0, 173: 0.090909090
90909091, 174: 0.043478260869565216, 175: 0.0, 176: 0.0, 177: 0.0, 178: 0.0, 179: 0.0, 180: 0.0, 181: 0.0, 182: 0.0, 183: 0.0, 18
4: 0.0, 185: 0.0625, 186: 0.0, 187: 0.0, 188: 0.0, 189: 0.0, 190: 0.14285714285714285, 191: 0.0, 192: 0.0, 193: 0.0, 194: 0.0, 19
5: 0.1, 196: 0.0, 197: 0.0, 198: 0.0, 199: 0.0, 200: 0.0, 201: 0.0, 202: 0.0, 203: 0.07142857142857142, 204: 0.0, 205: 0.0, 206:
0.0, 207: 0.0, 208: 0.0, 209: 0.0, 210: 0.0, 211: 0.0, 212: 0.2, 213: 0.0625, 214: 0.0, 215: 0.0, 216: 0.038461538461538464, 217:
0.0, 218: 0.0, 219: 0.0, 220: 0.09090909090909091, 221: 0.0, 222: 0.0, 223: 0.0, 224: 0.0, 225: 0.0, 226: 0.0, 227: 0.0, 228: 0.0,
229: 0.0, 230: 0.0, 231: 0.0, 232: 0.0, 233: 0.0, 234: 0.0, 235: 0.0, 236: 0.0, 237: 0.0, 238: 0.0, 239: 0.0, 240: 0.0, 241: 0.222
2222222222222, 242: 0.0, 243: 0.09090909090909091, 244: 0.05263157894736842, 245: 0.0, 246: 0.0, 247: 0.0, 248: 0.05, 249: 0.0, 25
0: 0.0, 251: 0.0, 252: 0.0, 253: 0.0, 254: 0.0, 255: 0.0625, 256: 0.0, 257: 0.0, 258: 0.0, 259: 0.0625, 260: 0.0, 261: 0.0, 262:
0.0, 263: 0.0, 264: 0.0, 265: 0.05263157894736842, 266: 0.08333333333333333, 267: 0.058823529411764705, 268: 0.0, 269: 0.0, 270:
0.0, 271: 0.0, 272: 0.0, 273: 0.05, 274: 0.0, 275: 0.0, 276: 0.1, 277: 0.0, 278: 0.0, 279: 0.0, 280: 0.047619047619047616, 281: 0.
07692307692307693, 282: 0.0, 283: 0.0, 284: 0.06666666666666667, 285: 0.0, 286: 0.0, 287: 0.0, 288: 0.0, 289: 0.0, 290: 0.0, 291:
0.0, 292: 0.0, 293: 0.1111111111111111, 294: 0.0, 295: 0.0, 296: 0.0, 297: 0.0, 298: 0.0, 299: 0.0, 300: 0.04, 301: 0.0, 302: 0.0,
303: 0.0, 304: 0.07692307692307693, 305: 0.0, 306: 0.0, 307: 0.0625, 308: 0.0, 309: 0.0, 310: 0.0, 311: 0.0, 312: 0.0, 313: 0.0, 3
14: 0.14285714285714285, 315: 0.0, 316: 0.03125, 317: 0.0, 318: 0.0, 319: 0.0, 320: 0.0, 321: 0.0, 322: 0.0, 323: 0.0, 324: 0.0, 3
```

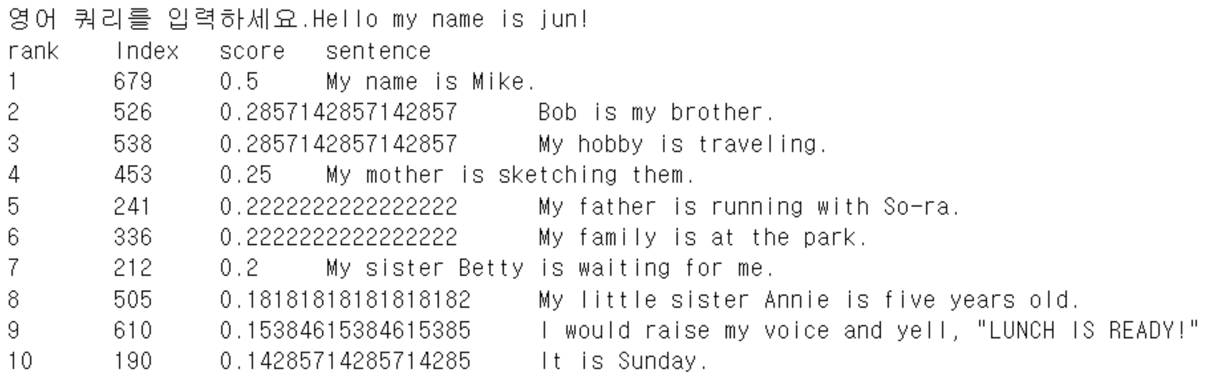
④ Rank the sentences based on similarity. And Output the top 10 ranked sentences to the user from the ranked sentences



```
[(679, 0.5), (526, 0.2857142857142857), (538, 0.2857142857142857), (453, 0.25), (241, 0.2222222222222222), (336, 0.222222222222222
2), (212, 0.2), (505, 0.18181818181818182), (610, 0.15384615384615385), (190, 0.14285714285714285), (314, 0.14285714285714285), (7
10, 0.14285714285714285), (45, 0.1111111111111111), (107, 0.1111111111111111), (293, 0.1111111111111111), (519, 0.111111111111111
1), (597, 0.1111111111111111), (667, 0.1111111111111111), (31, 0.1), (195, 0.1), (276, 0.1), (326, 0.1), (388, 0.1), (544, 0.1),
(559, 0.1), (564, 0.1), (671, 0.1), (712, 0.1), (17, 0.09090909090909091), (138, 0.09090909090909091), (173, 0.09090909090909091),
(220, 0.09090909090909091), (243, 0.09090909090909091), (330, 0.09090909090909091), (412, 0.09090909090909091), (570, 0.0909090909
0909091), (693, 0.09090909090909091), (50, 0.08333333333333333), (51, 0.08333333333333333), (77, 0.08333333333333333), (266, 0.083
33333333333333), (340, 0.08333333333333333), (425, 0.08333333333333333), (436, 0.08333333333333333), (463, 0.08333333333333333),
(20, 0.07692307692307693), (73, 0.07692307692307693), (78, 0.07692307692307693), (281, 0.07692307692307693), (304, 0.0769230769230
7693), (358, 0.07692307692307693), (386, 0.07692307692307693), (419, 0.07692307692307693), (111, 0.07142857142857142), (112, 0.071
42857142857142), (120, 0.07142857142857142), (203, 0.07142857142857142), (432, 0.07142857142857142), (449, 0.07142857142857142),
(452, 0.07142857142857142), (469, 0.07142857142857142), (558, 0.07142857142857142), (592, 0.07142857142857142), (623, 0.0714285714
2857142), (675, 0.07142857142857142), (70, 0.06666666666666667), (75, 0.06666666666666667), (82, 0.06666666666666667), (119, 0.066
66666666666667), (284, 0.06666666666666667), (334, 0.06666666666666667), (601, 0.06666666666666667), (622, 0.06666666666666667),
(635, 0.06666666666666667), (705, 0.06666666666666667), (27, 0.0625), (185, 0.0625), (213, 0.0625), (255, 0.0625), (259, 0.0625),
(307, 0.0625), (361, 0.0625), (515, 0.0625), (577, 0.0625), (655, 0.0625), (687, 0.0625), (23, 0.058823529411764705), (267, 0.0588
23529411764705), (435, 0.058823529411764705), (568, 0.058823529411764705), (614, 0.058823529411764705), (618, 0.0588235294117647O
5), (24, 0.05555555555555555), (426, 0.05555555555555555), (579, 0.05555555555555555), (695, 0.05555555555555555), (55, 0.05263157
894736842), (65, 0.05263157894736842), (163, 0.05263157894736842), (244, 0.05263157894736842), (265, 0.05263157894736842), (486,
0.05263157894736842), (543, 0.05263157894736842), (552, 0.05263157894736842), (587, 0.05263157894736842), (96, 0.05), (106, 0.05),
(129, 0.05), (248, 0.05), (273, 0.05), (356, 0.05), (403, 0.05), (572, 0.05), (583, 0.05), (713, 0.05), (80, 0.04761904761904761
6), (280, 0.047619047619047616), (368, 0.047619047619047616), (603, 0.047619047619047616), (645, 0.047619047619047616), (666, 0.04
7619047619047616), (475, 0.045454545454545456), (480, 0.045454545454545456), (513, 0.045454545454545456), (531, 0.0454545454545454
56), (628, 0.045454545454545456), (174, 0.043478260869565216), (478, 0.043478260869565216), (21, 0.041666666666666664), (98, 0.041
666666666666664), (367, 0.041666666666666664), (411, 0.041666666666666664), (588, 0.041666666666666664), (634, 0.0416666666666666
4), (145, 0.04), (300, 0.04), (365, 0.04), (521, 0.04), (128, 0.038461538461538464), (216, 0.038461538461538464), (510, 0.03846153
8461538464), (420, 0.037037037037037035), (563, 0.037037037037037035), (69, 0.03571428571428571), (316, 0.03125), (0, 0.0), (1, 0.
0), (2, 0.0), (3, 0.0), (4, 0.0), (5, 0.0), (6, 0.0), (7, 0.0), (8, 0.0), (9, 0.0), (10, 0.0), (11, 0.0), (12, 0.0), (13, 0.0), (1
4, 0.0), (15, 0.0), (16, 0.0), (18, 0.0), (19, 0.0), (22, 0.0), (25, 0.0), (26, 0.0), (28, 0.0), (29, 0.0), (30, 0.0), (32, 0.0),
(33, 0.0), (34, 0.0), (35, 0.0), (36, 0.0), (37, 0.0), (38, 0.0), (39, 0.0), (40, 0.0), (41, 0.0), (42, 0.0), (43, 0.0), (44, 0.

rank    Index   score   sentence
1       679     0.5     My name is Mike.
2       526     0.2857142857142857      Bob is my brother.
3       538     0.2857142857142857      My hobby is traveling.
4       453     0.25    My mother is sketching them.
5       241     0.2222222222222222      My father is running with So-ra.
6       336     0.2222222222222222      My family is at the park.
7       212     0.2     My sister Betty is waiting for me.
8       505     0.18181818181818182     My little sister Annie is five years old.
9       610     0.15384615384615385     I would raise my voice and yell, "LUNCH IS READY!"
10      190     0.14285714285714285     It is Sunday.
```

## 2. Final Test Screenshot



```
영어 쿼리를 입력하세요.Hello my name is jun!
rank    Index   score   sentence
1       679     0.5     My name is Mike.
2       526     0.2857142857142857      Bob is my brother.
3       538     0.2857142857142857      My hobby is traveling.
4       453     0.25    My mother is sketching them.
5       241     0.2222222222222222      My father is running with So-ra.
6       336     0.2222222222222222      My family is at the park.
7       212     0.2     My sister Betty is waiting for me.
8       505     0.18181818181818182     My little sister Annie is five years old.
9       610     0.15384615384615385     I would raise my voice and yell, "LUNCH IS READY!"
10      190     0.14285714285714285     It is Sunday.
```

## 7. Results and Conclusion

① Project Results.:

When a user enters a text, the sentences in the target file are compared to the user's input, disregarding case sensitivity. The top 10 sentences with the highest similarity will be ranked and printed along with their rank, index, score, and the sentence.

② Impressions:

Through the document search program, the core of the project was data preprocessing and refinement, and I realized the importance of this for obtaining reliable results. Handling tasks like data tokenization, creating sets, and converting to lowercase improved my technical skills. Additionally, it allowed me to enhance my user-centered design abilities. This project helped me improve both my technical skills and project management abilities, and made me aware of the importance of user-centered design and data management. Furthermore, this experience will be valuable for future natural language processing and data analysis projects.