1. Introduction

1-1. Project Purpose and Background: Practice what I have learned until week 7.

1-2. Goal: Create a simple Search Engine that retrieves sentences similar to the user's query.

2. Requirements

2-1. User requirements: The system that searches for sentences similar to the query when a user enters a query

2-2. Functional Requirements:

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

2. Receive an input English 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.

5. Output the top 10 ranked sentences to the user from the ranked sentences.

6. Calculate similarity without considering case sensitivity

3. Design and Implementation

3-1. implementation Details:

1-1.

```
# 입력 받은 sentence를 띄어쓰기 기준으로 리스트 생성
def preprocess(sentence):
    preprocessed_sentence = sentence.strip().split(" ")
    return preprocessed_sentence
```

2. Input

Preprocessed_sentence : The list based on the spacing of the received sentence

3. Result

Return : Preprocessed_sentence

Create the list based on the spacing of the received sentence.

4. Explanation : Make a list by separating the words based on the spacing of the English queries received.

2-1.

```
# 파일을 열어 문장들을 하나씩 file_tokens_pairs 리스트에 저장
def indexing(file_name):
    file_tokens_pairs = []
    lines = open(file_name, "r", encoding="utf8").readlines()
    for line in lines:
        tokens = preprocess(line)
        file_tokens_pairs.append(tokens)
    return file_tokens_pairs
```

2. Input

file_tokens_pairs : List that stores the sentences in the file one by one

3. Result : file_tokens_pairs

Make a list that stores the sentences in the file one by one.

4. Explanation : Read the sentences in the file one by one and save them in turn in the blank list.

3-1.

```python
# 앞서 만든 두 리스트의 유사성 비교
def calc_similarity(preprocessed_query, preprocessed_sentences):
    score_dict = {}
    for i in range(len(preprocessed_sentences)):
        # 대소문자 구분 없는 토큰 셋 만들기
        sentence = preprocessed_sentences[i]
        query_str = ' '.join(preprocessed_query).lower()
        sentence_str = ' '.join(sentence).lower() # 문장 전부 소문자로
        preprocessed_query = set(preprocess(query_str))
        preprocessed_sentence = preprocess(sentence_str)
        file_token_set = set(preprocessed_sentence)
        all_tokens = preprocessed_query | file_token_set
        same_tokens = preprocessed_query & file_token_set
        similarity = len(same_tokens) / len(all_tokens)
        score_dict[i] = similarity
    return score_dict
```

2. Input

sentence :

query_str : Change capital letters to lowercase letters

sentence_str : Change capital letters to lowercase letters

preprocessed_query : Create the set based on the spacing of the query_str

preprocessed_sentence : Create the list based on the spacing of the sentence_str

file_token_set : Create the set about preprocessed_sentence

all_tokens : Total words count

same_tokens : Overlapping words count

similarity : Dividing the number of overlapping words by the total number of words

score_dict : Empty dictionary to store similarity

3. Result : score_dict

Make similarity score for each sentence by dividing the number of overlapping words by the total number of words and save it to the list.

4. Explanation : Make similarity score for each sentence and save it to the list.

4-1.

```
# 유사성 점수를 정렬
sorted_score_list = sorted(score_dict.items(), key = operator.itemgetter(1), reverse=True)
```

2. Input

Sorted_score_list : Sort score_dict

3. Result : None

Sort score_dict.

4. Explanation : Sort score_dict.

5-1.

```
# 결과 출력
if sorted_score_list[0][0] == 0.0:
    print("There is no similar sentence.")
else:
    print("rank", "Index", "score", "sentence", sep = "₩t")
    rank = 1
    for i, score  in sorted_score_list:
        print(rank, i, score, ' '.join(file_tokens_pairs[i]), sep = "₩t")
        if rank == 10:
            break
        rank = rank + 1
```

2. Input

Sorted_score_list : List saved with similarity scores sorted by sentence

3. Result : None

Print out sentences and scores with similarity scores up to 10th place.

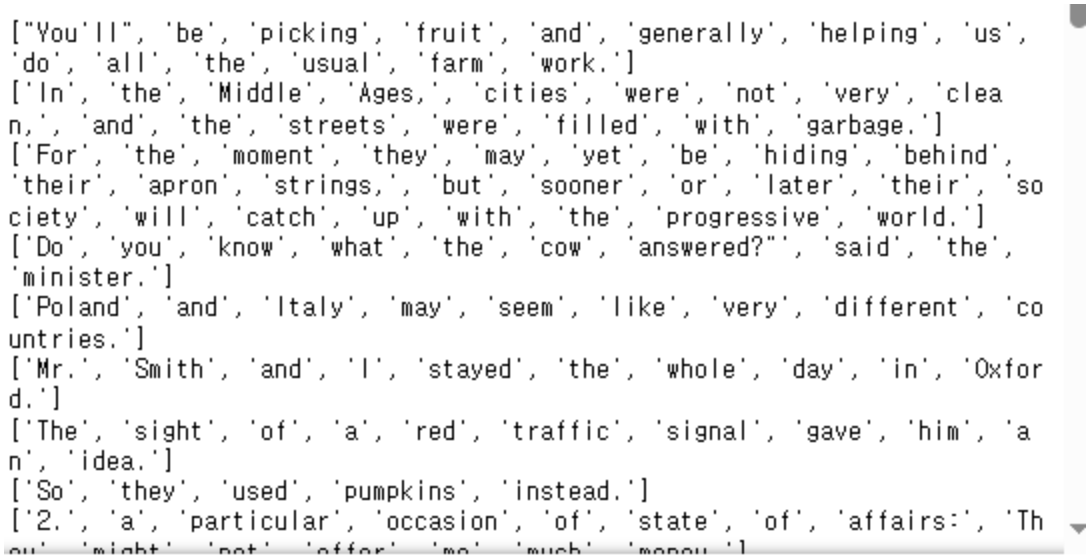
If there is no similar sentence, print that there is no similar sentence.

4. Explanation : If the score of the first item in the sorted list is 0, it is determined that there is no similar sentence and print message.

Otherwise, Print out a sentence and sentence number from 1st to 10th place.

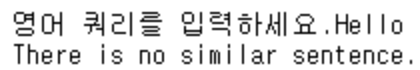
4. Testing

1. Test Results for Each Functionality



2. Final Test Screenshot

1) If there is no similar sentence



2) If there are similar sentences

```
영어 쿼리를 입력하세요.Hello My name is Sungmin Joo
rank    Index   score   sentence
1       679     0.42857142857142855     My name is Mike.
2       526     0.25    Bob is my brother.
3       538     0.25    My hobby is traveling.
4       453     0.2222222222222222      My mother is sketching them.
5       241     0.2     My father is running with So-ra.
6       336     0.2     My family is at the park.
7       212     0.18181818181818182     My sister Betty is waiting for me.
8       505     0.16666666666666666     My little sister Annie is five years
old.
9       610     0.14285714285714285     I would raise my voice and yell, "LU
NCH IS READY!"
10      190     0.125   It is Sunday.
```

5. Results and Conclusion

5-1. Result: Created the search engine to compare sentence similarities.

5-2. Conclusion: It was difficult to understand because there were many functions that I didn't know in the example given by the professor.