Search Engine Report

224491 JI Yeonwoo

Dept. Aritificial Intelligence

Chonnam National University

1. Introduction

1.1. Project Purpose and Background

: This project was undertaken to apply the knowledge learned in the seven weeks. Especially, the goal is to analyze Python code and get used to making it a function.

1.2. Goal

: To implement a basic search engine that retrieves sentences similar to the user's query.

2. Requirements

2.1. User requirements

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

2.2. Functional Requirements

- Preprocess sentences within the search target and store them in a list.
- Receive an input English string (query) from the user and preprocess it.
- Calculate the similarity between the query and sentences within the search target.
- Rank the sentences based on similarity.

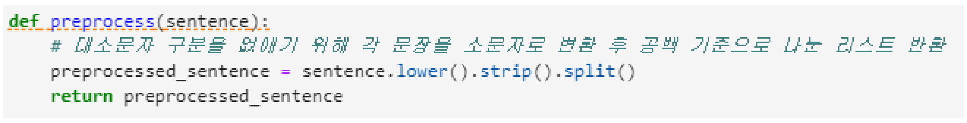
- Output the top 10 ranked sentences to the user from the ranked sentences.

3. Design and Implementation

3.1. Implementation Details

3.1.1. Preprocessing function

3.1.1.1. Screen shot



```python
def preprocess(sentence):
    # 대소문자 구분을 없애기 위해 각 문장을 소문자로 변환 후 공백 기준으로 나눈 리스트 반환
    preprocessed_sentence = sentence.lower().strip().split()
    return preprocessed_sentence
```

3.1.1.2. Input

- sentence: String of sentences to be preprocessed

3.1.1.3. Return

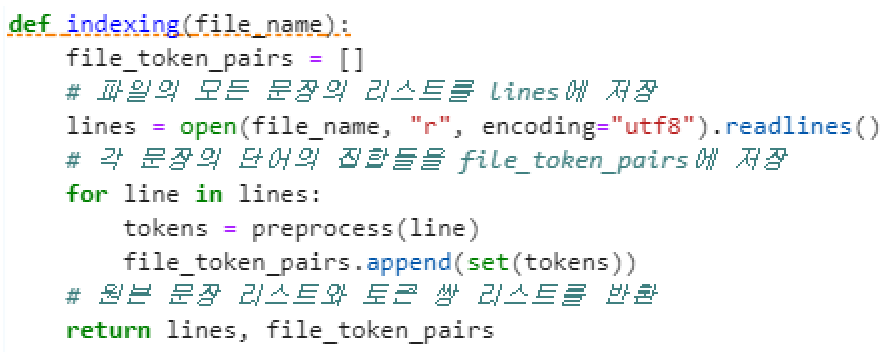
- preprocessed_sentence: A list divieded by space

3.1.1.4. Explanation

- To eliminate case-sensitive, convert each sentence to lowercase and return a list divided by space

3.1.2. Indexing function

3.1.2.1. Screen shot



```python
def indexing(file_name):
    file_token_pairs = []
    # 파일의 모든 문장의 리스트를 lines에 저장
    lines = open(file_name, "r", encoding="utf8").readlines()
    # 각 문장의 단어의 집합들을 file_token_pairs에 저장
    for line in lines:
        tokens = preprocess(line)
        file_token_pairs.append(set(tokens))
    # 원본 문장 리스트와 토큰 쌍 리스트를 반환
    return lines, file_token_pairs
```

3.1.2.2. Input

- file_name: Name of the file containing the search target sentences.

3.1.2.3. Return

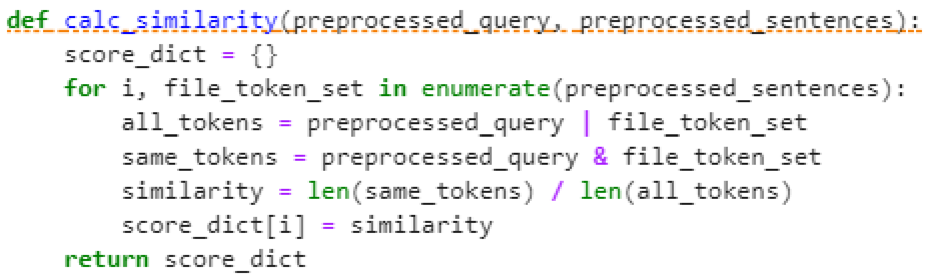
- lines: Original Sentence List
- file_token_pairs: a set of words in each sentence

3.1.2.4. Explanation

- Read the file and preprocess each sentence using the preprocess function, and create a set of preprocessed words and save them to the list

3.1.3. calc_similarity function

3.1.3.1. Screen shot



```
def calc_similarity(preprocessed_query, preprocessed_sentences):
    score_dict = {}
    for i, file_token_set in enumerate(preprocessed_sentences):
        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
```

3.1.3.2. Input

- preprocessed_query: A set of words in query sentence
- preprocessed_sentences: A list of sets of words in each sentences

3.1.3.3. Return

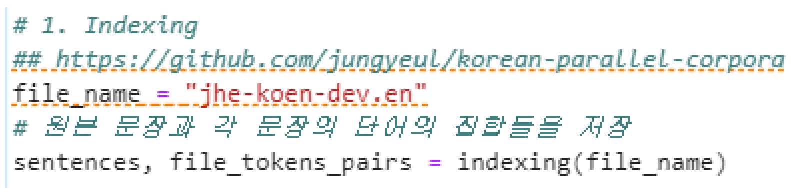
- score_dict: Dictionary with the index of the sentence in the file as the key and the similarity score with the query as the value

3.1.3.4. Explanation

- Calculate the similarity as a union divided intersection of query and file’s sentences.
- Store the index of each sentence in the dictionary as a key and the similarity as a value.

3.1.4. Indexing

3.1.4.1. Screen shot



```
# 1. Indexing
## https://github.com/jungyeul/korean-parallel-corpora
file_name = "jhe-koen-dev.en"
# 원본 문장과 각 문장의 단어의 집합들을 저장
sentences, file_tokens_pairs = indexing(file_name)
```

3.1.4.2. Input

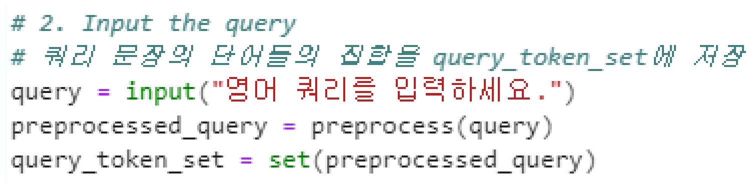
- file_name: Name of the file containing the search target sentences.

3.1.4.3. Explanation

- Store the original sentences in sentences and a list of the set of words in each sentence in file_token_pairs

3.1.5. Input the query

3.1.5.1. Screen shot



```
# 2. Input the query
# 쿼리 문장의 단어들의 집합을 query_token_set에 저장
query = input("영어 쿼리를 입력하세요.")
preprocessed_query = preprocess(query)
query_token_set = set(preprocessed_query)
```

3.1.5.2. Input

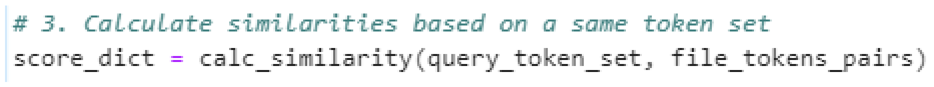
- query: Quert sentence

3.1.5.3. Explanation

- Preprocess a query sentence and store a set of words in query_token_set

3.1.6. Calculate similarities based on a same token set

3.1.6.1. Screen shot



```
# 3. Calculate similarities based on a same token set
score_dict = calc_similarity(query_token_set, file_tokens_pairs)
```

3.1.6.2. Input

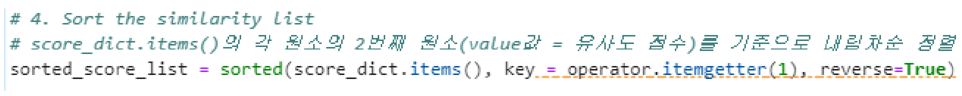
- query_token_set: A set of words of query sentence
- file_token_pairs: A list of the set of words in each sentence in file

3.1.6.3. Explantion

- Calculate similarites based on a same token set and store it in score_dict.

3.1.7. Sort the similarity list

3.1.7.1. Screen shot



```
# 4. Sort the similarity list
# score_dict.items()의 각 원소의 2번째 원소(value값 = 유사도 점수)를 기준으로 내림차순 정렬
sorted_score_list = sorted(score_dict.items(), key = operator.itemgetter(1), reverse=True)
```

3.1.7.2. Input

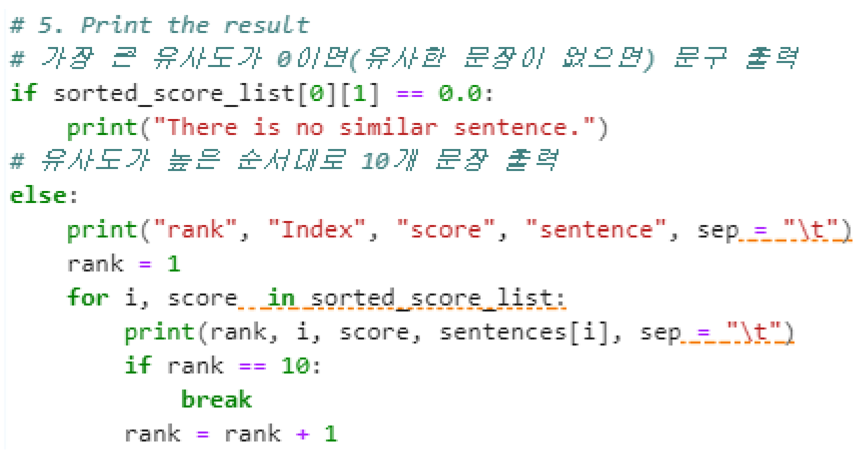
- score_dict: Dictionary that it's keys are index of sentences and values are similarity of sentences and query

3.1.7.3. Explanation

- Sort score_dict.items() in descending order by similarity and save it to the list

3.1.8. Print the result

3.1.8.1. Screen shot



```
# 5. Print the result
# 가장 큰 유사도가 0이면(유사한 문장이 없으면) 문구 출력
if sorted_score_list[0][1] == 0.0:
    print("There is no similar sentence.")
# 유사도가 높은 순서대로 10개 문장 출력
else:
    print("rank", "Index", "score", "sentence", sep = "\t")
    rank = 1
    for i, score  in sorted_score_list:
        print(rank, i, score, sentences[i], sep = "\t")
        if rank == 10:
            break
        rank = rank + 1
```

3.1.8.2. Input

- sorted_score_list: A list of score_dict.items() sorted by similarity

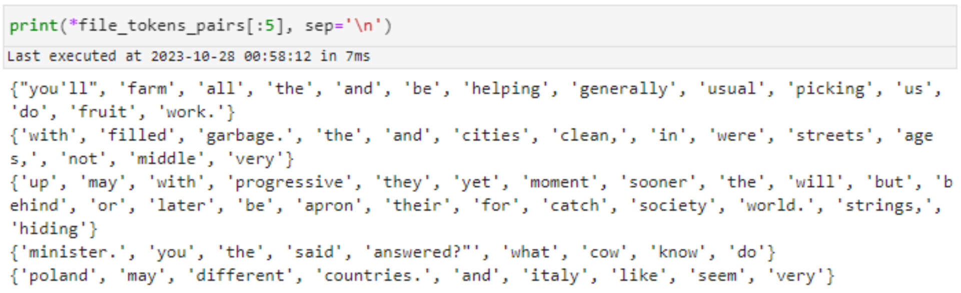
3.1.8.3. Explanation

- Print 10 sentences similar to the query in order of similarity

4. Testing

4.1. Test Results for Each Functionality

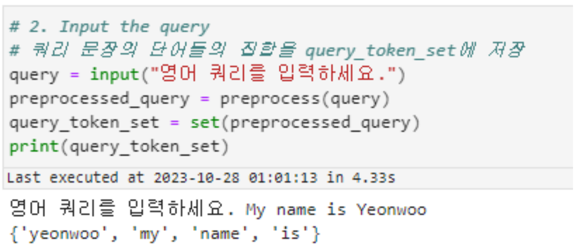
4.1.1. Preprocess sentences within the search target and store them in a list



```
print(*file_tokens_pairs[:5], sep='\n')
Last executed at 2023-10-28 00:58:12 in 7ms
```

```
{"you'll", 'farm', 'all', 'the', 'and', 'be', 'helping', 'generally', 'usual', 'picking', 'us',
'do', 'fruit', 'work.'}
{'with', 'filled', 'garbage.', 'the', 'and', 'cities', 'clean,', 'in', 'were', 'streets', 'age
s,', 'not', 'middle', 'very'}
{'up', 'may', 'with', 'progressive', 'they', 'yet', 'moment', 'sooner', 'the', 'will', 'but', 'b
ehind', 'or', 'later', 'be', 'apron', 'their', 'for', 'catch', 'society', 'world.', 'strings,',
'hiding'}
{'minister.', 'you', 'the', 'said', 'answered?"', 'what', 'cow', 'know', 'do'}
{'poland', 'may', 'different', 'countries.', 'and', 'italy', 'like', 'seem', 'very'}
```

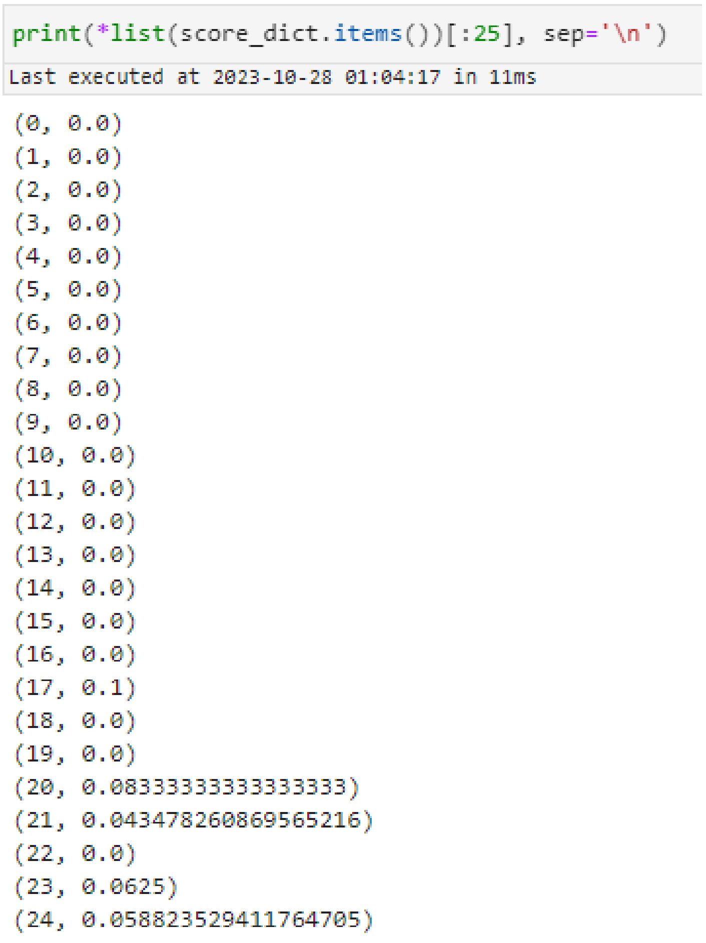
4.1.2. Receive an input English string (query) from the user and preprocess it



```
# 2. Input the query
# 쿼리 문장의 단어들의 집합을 query_token_set에 저장
query = input("영어 쿼리를 입력하세요.")
preprocessed_query = preprocess(query)
query_token_set = set(preprocessed_query)
print(query_token_set)
Last executed at 2023-10-28 01:01:13 in 4.33s
```

```
영어 쿼리를 입력하세요. My name is Yeonwoo
{'yeonwoo', 'my', 'name', 'is'}
```

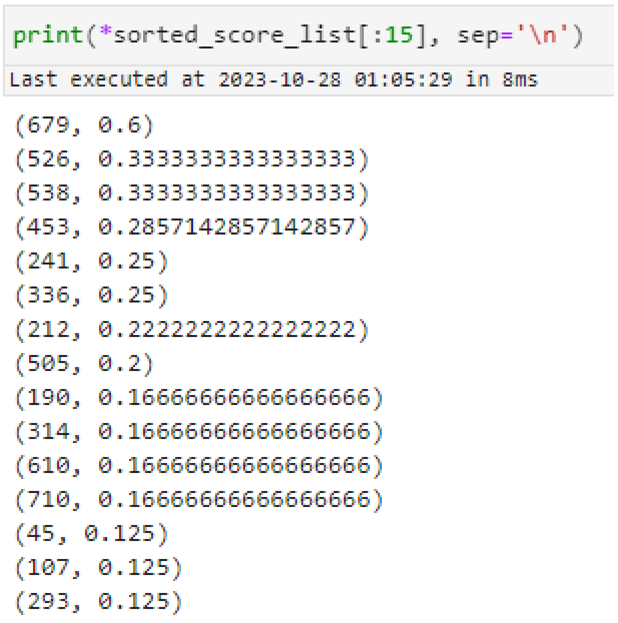
4.1.3. Calculate the similarity



```
print(*list(score_dict.items())[:25], sep='\n')
Last executed at 2023-10-28 01:04:17 in 11ms
```

```
(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, 0.1)
(18, 0.0)
(19, 0.0)
(20, 0.08333333333333333)
(21, 0.043478260869565216)
(22, 0.0)
(23, 0.0625)
(24, 0.058823529411764705)
```

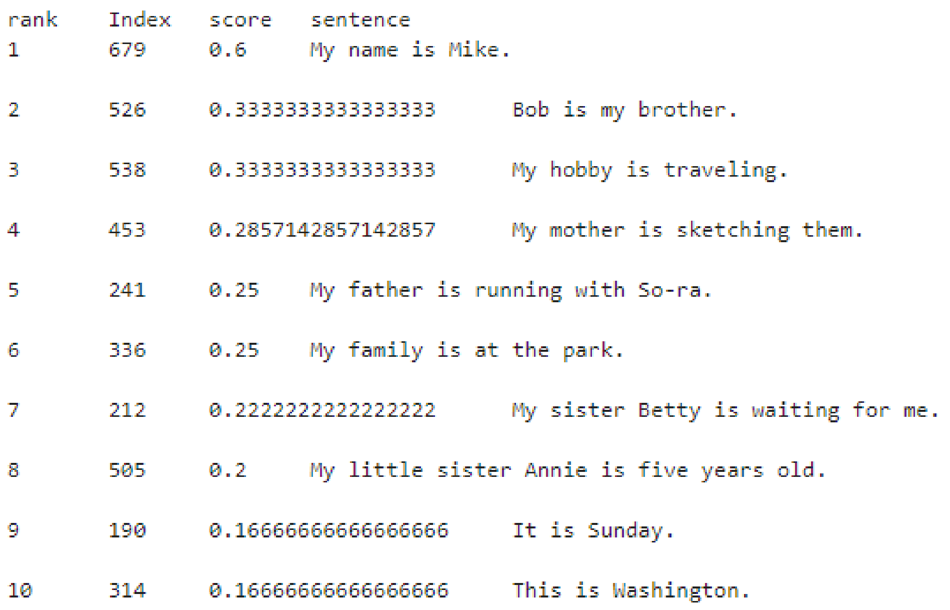
4.1.4. Rank the sentences based on similarity



```
print(*sorted_score_list[:15], sep='\n')
Last executed at 2023-10-28 01:05:29 in 8ms
```

```
(679, 0.6)
(526, 0.3333333333333333)
(538, 0.3333333333333333)
(453, 0.2857142857142857)
(241, 0.25)
(336, 0.25)
(212, 0.2222222222222222)
(505, 0.2)
(190, 0.16666666666666666)
(314, 0.16666666666666666)
(610, 0.16666666666666666)
(710, 0.16666666666666666)
(45, 0.125)
(107, 0.125)
(293, 0.125)
```

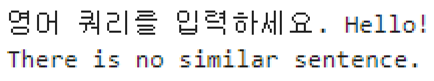
4.1.5. Output the top 10 ranked sentences



```
rank    Index   score   sentence
1       679     0.6     My name is Mike.

2       526     0.3333333333333333      Bob is my brother.

3       538     0.3333333333333333      My hobby is traveling.

4       453     0.2857142857142857      My mother is sketching them.

5       241     0.25    My father is running with So-ra.

6       336     0.25    My family is at the park.

7       212     0.2222222222222222      My sister Betty is waiting for me.

8       505     0.2     My little sister Annie is five years old.

9       190     0.16666666666666666     It is Sunday.

10      314     0.16666666666666666     This is Washington.
```

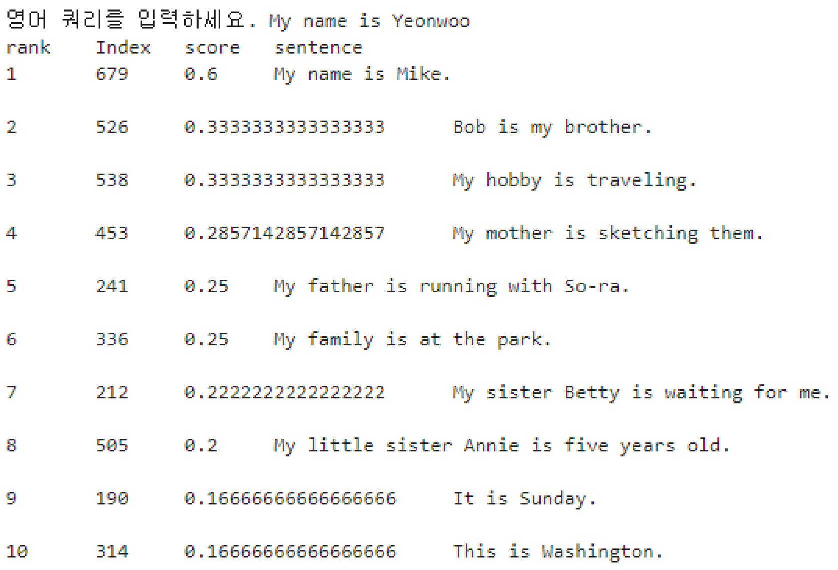
4.2. Final Test Screenshot

4.2.1. If there is no similar sentence



```
영어 쿼리를 입력하세요. Hello!
There is no similar sentence.
```

4.2.2. If there are similar sentences



```
영어 쿼리를 입력하세요. My name is Yeonwoo
rank    Index   score   sentence
1       679     0.6     My name is Mike.

2       526     0.3333333333333333      Bob is my brother.

3       538     0.3333333333333333      My hobby is traveling.

4       453     0.2857142857142857      My mother is sketching them.

5       241     0.25    My father is running with So-ra.

6       336     0.25    My family is at the park.

7       212     0.2222222222222222      My sister Betty is waiting for me.

8       505     0.2     My little sister Annie is five years old.

9       190     0.16666666666666666     It is Sunday.

10      314     0.16666666666666666     This is Washington.
```

5. Results and Conclusion

5.1. Result

: I made each code into functions and improved the program to be case-insensitive.

5.2. Conclusion

: I found it harder to write a report than to write a code. Writing a report in English takes a lot longer than I think, so the report to be written later will start in advance before the deadline.