CNU
203734 Dong Ho Han
Colledge of Humanites
Department of English Language and Literature

SEARCH_ENGINE.PY

FILE

2023-10-23	PROJECT NAME	STUDENT
Chonnam National University	search_engine.cpp	Dong ho Han

INTRODUCTION

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 covered

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

REQUIREMENTS

1. User requirements

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

2. 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

④ Rank the sentences based on similarity.

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

1. preprocess(sentence)

- **input: a query or each sentence of search targets**
- **return: preprocessed query or sentence (=a set of tokens)**

```
def preprocess(sentence):
    preprocessed_sentence = 
sentence.strip().split(" ")
    return preprocessed_sentence
```

2. indexing(file_name)

- **input: a file name with its path for search targets**
- **return: a set of tokens for each sentence in the file**

```
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
```

3.calc_similarity(preprocessed_query,preprocessed_sentences)

• **input: preprocessed query, preprocessed sentences (=search target)**

• **return: a dictionary containing file_id and corresponding similarity score**

```python
def calc_similarity(preprocessed_query, preprocessed_sentences):

    score_dict = {}

    for i in range(len(preprocessed_sentences)):

        # 시작: 대소문자 구분 없는 토큰 셋을 만들기 위한 코드

        sentence = preprocessed_sentences[i]

        query_str = ' '.join(preprocessed_query).lower()

        sentence_str = ' '.join(sentence).lower()

        preprocessed_query = set(preprocess(query_str))

        preprocessed_sentence = preprocess(sentence_str)

        # 끝: 대소문자 구분 없는 토큰 셋을 만들기 위한 코드


        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
```

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

```
# ①+②
# 2. Input the query
query = input("영어 쿼리를 입력하세요.")
preprocessed_query = preprocess(query)
query_token_set =
set(preprocessed_query)
```

③ Calculate the similarity between the query and sentences within the search target

```
# 3. Calculate similarities based on a same token set

score_dict = calc_similarity(query_token_set, file_tokens_pairs)
```

④ Rank the sentences based on similarity.
⑤ Output the top 10 ranked sentences to the user from the ranked sentences.

```
# ④ + ⑤

# 4. Sort the similarity list

sorted_score_list = sorted(score_dict.items(), key = operator.itemgetter(1),

reverse=True)



# 5. Print the result

if sorted_score_list[0][1] == 0.0:
```

```python
    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
```

TEST RESULTS FOR EACH FUNCTIONALITY

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
④ Rank the sentences based on similarity.
⑤ Output the top 10 ranked sentences to the user from the ranked sentences.

```
영어 쿼리를 입력하세요.we are
rank	Index	score	sentence
1	180	0.25	How are you?
2	530	0.2222222222222222	Now we are working hard for the 21st century.
3	115	0.2	The parks are beautiful.
4	143	0.2	But they are alive.
5	196	0.2	In Theater U-ju, we are playing 'Bear' and 'Star Wars'.
6	283	0.2	Humans are no exception.
7	378	0.2	We are going to sell cookies at baseball games, too.
8	93	0.16666666666666666	Our money troubles are over!
9	194	0.16666666666666666	Judy and Betty are sisters.
10	508	0.16666666666666666	So we may say that sports, like music, are an
international language.
```

FINAL TEST SCREENSHOT

```
영어 쿼리를 입력하세요.we are
rank	Index	score	sentence
1	180	0.25	How are you?
2	530	0.2222222222222222	Now we are working hard for the 21st century.
3	115	0.2	The parks are beautiful.
4	143	0.2	But they are alive.
5	196	0.2	 In Theater U-ju, we are playing 'Bear' and 'Star Wars'.
6	283	0.2	Humans are no exception.
7	378	0.2	We are going to sell cookies at baseball games, too.
8	93	0.16666666666666666	Our money troubles are over!
9	194	0.16666666666666666	Judy and Betty are sisters.
10	508	0.16666666666666666	So we may say that sports, like music, are an
international language.
```

RESULTS AND CONCLUSIOIN

Result

⇨ The development of the search engine was successfully accomplished.

Conclusion

⇨ I wanna sleep…!