CHONNAM NATIONAL
UNIVERSITY

Search Engine

과 목 명 :	파이썬 프로그래밍 및 실습
담당교수 :	김미수 교수님
제 출 일 :	2023.10.27
학 과 :	인공지능학부
학 번 :	213407
이 름 :	왕사빈

Chonnam National University

1. Introduction

1. Project Purpose and Background

The purpose of this project is to develop a practical program using the knowledge learned in the "Python Programming and Practice" course up to week 7.

2. Goal

The final goal of the project is to calculate the similarity between the user's input query and the sentences in the file 'jhe-koen-dev.en' to output the top 10 sentences most similar to the user's input.

2. Requirements

1. User requirements

The system should be able to search and output sentences similar to these sentences when a user enters a string (query).

2. Functional Requirements

① Preprocess the sentences of search target and store them in the list.
② An English character string (query) is inputted by the user and preprocessed.
③ Calculate the similarity between the input English string and the sentences of search target.
(Here, the similarity is calculated by dividing the number of overlapping tokens by the total number of tokens based on the same number of "words".)
④ Rank the sentences based on similarity.
⑤ The top 10 of the ranked sentences are output to the user and shown.

3. Design and Implementation

1. Implementation Details

▶ lower function

```
import operator

def lower_case(early_sentence) :
    lower_sentence = ''.join(early_sentence).lower()
    return lower_sentence
```

- input : Initial search query or each sentence in the initial search target.
- output : Lowercased search query or sentence.
- explanation : Lower the input sentence and return it.

▶ preprocess function

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

- input : Lower case query or each sentence of search target
- output : Tokenized query or sentence
- explanation : Tokenize sentences by dividing them based on spaces.

▶ indexing function

```
def indexing(file_name) :
    file_tokens_pairs = []
    for line in lines : #
        lower_line = lower_case(line)
        tokens = preprocess(lower_line)
        file_tokens_pairs.append(tokens)
    return file_tokens_pairs
```

- input : Name of the file to be searched
- output : Token list for each sentence in the file
- explanation : After generating a list of tokens to return, execute a statement as many as the number of sentences in the file. For each sentence within the repeating statement, the preprocessed sentence is added to the token list by executing "lower_case" and "preprocess function".

▶ calc_similarity function

```
def calc_similarity(preprocessed_query, preprocessed_sentences) :
    score_dict = {}
    for i in range(len(file_tokens_pairs)):
        file_token_set = set(file_tokens_pairs[i])
        all_tokens = query_token_set | file_token_set
        same_tokens = query_token_set & file_token_set
        similarity = len(same_tokens) / len(all_tokens)
        score_dict[i] = similarity
    return score_dict
```

- input : Preprocessed query and preprocessed search target sentence.
- output : A dictionary with file index and similarity score for the query and sentence.
- explanation : After creating a dictionary to store similarity scores, run a loop as many times as the number of sentences in the file. Within the loop, convert the token lists for each sentence in the file into sets. Calculate the similarity by dividing the number of overlapping tokens by the total number of tokens and store the computed similarity in the dictionary.

▶ Indexing

```
# 1. Indexing
file_name = "jhe-koen-dev.en"
lines = open(file_name, "r", encoding="utf8").readlines()
file_tokens_pairs = indexing(file_name)
```

- input : Data to be searched ('jhe-koen-dev.en')
- result : Token list for each sentence in the file
- explanation : Open the data file read-only and preprocess it to the indexing function.

▶ Input the query

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

- input : English string(query) received from user
- result : Store the preprocessed query in a set format.
- explanation : Preprocess the input query through the lower_case and preprocess function.

▶ Calculate similarities

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

- input : Query set and token list for each sentence in the file.
- result : Returns the dictionary containing similarity scores.

▶ Sort the similarity list

```
# 4. Sort the similarity list
sorted_score_list = sorted(score_dict.items(), key = operator.itemgetter(1), reverse=True)
```

- input : The dictionary containing similarity scores.
- result : Returns a list sorted in sequential order of similarity scores.

▶ Result

```
# 5. Print the result
if sorted_score_list[0][1] == 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
```

- input : List sorted in sequential order of similarity scores for each sentences.
- output : If there are similar sentences, the top 10 sentences are output along with ranking, index, and similarity score. If there is no similar sentence, output "There is no similar sentence.".
- explanation : The criterion for determining that there is no similar sentence is when the score of the first item in the sorted list is zero.

4. Testing

1. Test results for each functionality

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

```
import operator

def lower_case(early_sentence) :
    lower_sentence = ''.join(early_sentence).lower()
    return lower_sentence


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


def indexing(file_name) :
    file_tokens_pairs = []
    for line in lines : #
        lower_line = lower_case(line)
        tokens = preprocess(lower_line)
        file_tokens_pairs.append(tokens)
    return file_tokens_pairs

file_name = "jhe-koen-dev.en"
lines = open(file_name, "r", encoding="utf8").readlines()
file_tokens_pairs = indexing(file_name)

print(file_tokens_pairs)
```

```
[["you'll", 'be', 'picking', 'fruit', 'and', 'generally', 'helping', 'us', 'do',
'all', 'the', 'usual', 'farm', 'work.'], ['in', 'the', 'middle', 'ages,', 'citie
s', 'were', 'not', 'very', 'clean,', 'and', 'the', 'streets', 'were', 'filled',
'with', 'garbage.'], ['for', 'the', 'moment', 'they', 'may', 'yet', 'be', 'hidin
g', 'behind', 'their', 'apron', 'strings,', 'but', 'sooner', 'or', 'later', 'thei
r', 'society', 'will', 'catch', 'up', 'with', 'the', 'progressive', 'world.'],
['do', 'you', 'know', 'what', 'the', 'cow', 'answered?"', 'said', 'the', 'ministe
r.'], ['poland', 'and', 'italy', 'may', 'seem', 'like', 'very', 'different', 'cou
ntries.'], ['mr.', 'smith', 'and', 'i', 'stayed', 'the', 'whole', 'day', 'in', 'o
xford.'], ['the', 'sight', 'of', 'a', 'red', 'traffic', 'signal', 'gave', 'him',
'an', 'idea.'], ['so', 'they', 'used', 'pumpkins', 'instead.'], ['2.', 'a', 'part
icular', 'occasion', 'of', 'state', 'of', 'affairs:', 'they', 'might', 'not', 'of
fer', 'me', 'much', 'money.'], ["i'm", 'especially', 'interested', 'in', 'learnin
g,', 'horse-riding', 'skills,', 'so', 'i', 'hope', "you'll", 'include', 'informat
ion', 'about', 'this.'], ['instead,', 'the', 'devil', 'gave', 'him', 'a', 'singl
e', 'candle', 'to', 'light', 'his', 'way', 'through', 'the', 'darkness.'], ['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', 'ca
```

② An English character string (query) is inputted by the user and preprocessed.

```
import operator

def lower_case(early_sentence) :
    lower_sentence = ''.join(early_sentence).lower()
    return lower_sentence


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


query = input("영어 쿼리를 입력하세요.")
lower_query = lower_case(query)
preprocessed_query = preprocess(lower_query)
query_token_set = set(preprocessed_query)

print(query_token_set)

```

```
영어 쿼리를 입력하세요.My name is Sabin
{'sabin', 'is', 'my', 'name'}
```

③ Calculate the similarity between the input English string and the sentences of search target.
(Here, the similarity is calculated by dividing the number of overlapping tokens by the total number of tokens based on the same number of "words".)

```
import operator

def lower_case(early_sentence) :
    lower_sentence = ''.join(early_sentence).lower()
    return lower_sentence


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


def indexing(file_name) :
    file_tokens_pairs = []
    for line in lines : #
        lower_line = lower_case(line)
        tokens = preprocess(lower_line)
        file_tokens_pairs.append(tokens)
    return file_tokens_pairs


def calc_similarity(preprocessed_query, preprocessed_sentences) :
    score_dict = {}
    for i in range(len(file_tokens_pairs)):
        file_token_set = set(file_tokens_pairs[i])
        all_tokens = query_token_set | file_token_set
        same_tokens = query_token_set & file_token_set
        similarity = len(same_tokens) / len(all_tokens)
        score_dict[i] = similarity
    return score_dict
```

```
file_name = "jhe-koen-dev.en"
lines = open(file_name, "r", encoding="utf8").readlines()
file_tokens_pairs = indexing(file_name)

query = input("영어 쿼리를 입력하세요.")
lower_query = lower_case(query)
preprocessed_query = preprocess(lower_query)
query_token_set = set(preprocessed_query)

score_dict = calc_similarity(query_token_set, file_tokens_pairs)

print(score_dict)
```

```
영어 쿼리를 입력하세요.My name is Sabin
{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, 25: 0.0, 26: 0.0, 27: 0.06666666666666667, 28: 0.0, 29:
0.0, 30: 0.0, 31: 0.1111111111111111, 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.
125, 46: 0.0, 47: 0.0, 48: 0.0, 49: 0.0, 50: 0.09090909090909091, 51: 0.090909090
90909091, 52: 0.0, 53: 0.0, 54: 0.0, 55: 0.05555555555555555, 56: 0.0, 57: 0.0, 5
8: 0.0, 59: 0.0, 60: 0.0, 61: 0.0, 62: 0.0, 63: 0.0, 64: 0.0, 65: 0.0555555555555
5555, 66: 0.0, 67: 0.0, 68: 0.0, 69: 0.037037037037037035, 70: 0.0714285714285714
2, 71: 0.0, 72: 0.0, 73: 0.08333333333333333, 74: 0.0, 75: 0.07142857142857142, 7
6: 0.0, 77: 0.09090909090909091, 78: 0.08333333333333333, 79: 0.0, 80: 0.05, 81:
0.0, 82: 0.07142857142857142, 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: 0.052631578
94736842, 97: 0.0, 98: 0.043478260869565216, 99: 0.0, 100: 0.0, 101: 0.0, 102: 0.
0, 103: 0.0, 104: 0.0, 105: 0.0, 106: 0.05263157894736842, 107: 0.125, 108: 0.0,
109: 0.0, 110: 0.0, 111: 0.07692307692307693, 112: 0.07692307692307693, 113: 0.0,
114: 0.0, 115: 0.0, 116: 0.0, 117: 0.0, 118: 0.0, 119: 0.07142857142857142, 120:
```

④ Rank the sentences based on similarity.

(Only codes to sort in order of similarity scores have been added from ③ code snippets that calculate similarity scores.)

```
44
45 sorted_score_list = sorted(score_dict.items(), key = operator.itemgetter(1), rev
46
47
48 print(sorted_score_list)
49
```

```
영어 쿼리를 입력하세요.My name is Sabin
[(679, 0.6), (526, 0.3333333333333333), (538, 0.3333333333333333), (453, 0.285714
2857142857), (241, 0.25), (336, 0.25), (212, 0.2222222222222222), (505, 0.2), (19
0, 0.16666666666666666), (314, 0.16666666666666666), (610, 0.16666666666666666),
(710, 0.16666666666666666), (45, 0.125), (107, 0.125), (293, 0.125), (519, 0.12
5), (597, 0.125), (667, 0.125), (31, 0.1111111111111111), (195, 0.111111111111111
1), (276, 0.1111111111111111), (326, 0.1111111111111111), (388, 0.111111111111111
1), (544, 0.1111111111111111), (559, 0.1111111111111111), (564, 0.111111111111111
1), (671, 0.1111111111111111), (712, 0.10526315789473684), (17, 0.1), (138, 0.1),
(173, 0.1), (220, 0.1), (243, 0.1), (330, 0.1), (412, 0.1), (570, 0.1), (693, 0.
1), (50, 0.09090909090909091), (51, 0.09090909090909091), (77, 0.0909090909090909
1), (266, 0.09090909090909091), (340, 0.09090909090909091), (425, 0.0909090909090
9091), (436, 0.09090909090909091), (463, 0.09090909090909091), (20, 0.08333333333
333333), (73, 0.08333333333333333), (78, 0.08333333333333333), (281, 0.0833333333
3333333), (304, 0.08333333333333333), (358, 0.08333333333333333), (386, 0.0833333
3333333333), (419, 0.08333333333333333), (111, 0.07692307692307693), (112, 0.0769
2307692307693), (120, 0.07692307692307693), (203, 0.07692307692307693), (432, 0.0
7692307692307693), (449, 0.07692307692307693), (452, 0.07692307692307693), (469,
0.07692307692307693), (558, 0.07692307692307693), (592, 0.07692307692307693), (62
```

⑤ The top 10 of the ranked sentences are output to the user and shown.

```
46
47 if sorted_score_list[0][1] == 0.0:
48     print("There is no similar sentence.")
49 else:
50     print("rank", "Index", "score", "sentence", sep = "₩t")
51     rank = 1
52     for i, score  in sorted_score_list:
53         print(rank, i, score, ' '.join(file_tokens_pairs[i]), sep = "₩t")
54         if rank == 10:
55             break
56         rank = rank + 1
57
58
```

```
영어 쿼리를 입력하세요.My name is Sabin
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.
```

2. Final Test Screenshot

- If there is no similar sentence

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

- If there are similar sentences

```
영어 쿼리를 입력하세요.Hello My name is Sabin Wang
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, "lunch is r
eady!"
10      190     0.125   it is sunday.
```

5. Results and Conclusion

1. Result

I have successfully developed a search engine program in accordance with the provided features.

2. Conclusion

It was my first time to load a file and implement a search engine in this way, so it took me a lot of time to understand the code presented, and it took me quite a while to write a report. But compared to that, I felt that the deadline for submitting the assignment was a little tight.

Nonetheless, I'm proud to have successfully completed the project. Additionally, I've always had some difficulty with implementing functions, so even though they were simple functions, gaining confidence in this aspect by using multiple functions and calling functions within functions makes me even prouder.