전남대학교
CHONNAM NATIONAL UNIVERSITY

[search engine development report]

■ subject : Python programming and practice

■ professor: Misso kim

■ deadline : 2023.10.27

■ department : Eldctroning engineering

■ studen number : 200244

■ name : Minho kim

Introduction

1. Project Purpose and Background

The project was carried out to practice what was learned until the 7th week.

2. Goal

Output 5 sentences similar to the user-entered string (query).

Requirements

1. User requirements

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

A. Similarity is based on the count of the same “word" without considering case sensitivity.

④ Rank the sentences based on similarity.

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

3. Function Plan

preprocess() : a function that that splits sentences into tokens
indexing() : a function that returns a list of processed sentences
lowering() : a function that converts sentence to lowered sentence
calc_similarity() : a function that calculates the similarity between the query and sentences within the search target

Design and Implementation1.

1. Implementation Details:

User defined functions

preprocess() : a function that that splits sentence into tokens

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

Input : a sentence
result : a list of tokens split in a sentence
Explanation :
After removing the spaces on both sides of the sentence, divide the sentence into tokens based on the spacing, and replace it with a list. Then put this list into 'preprocessed_sentence'

indexing() : a function that returns a list of processed sentences

```
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
```

Input : name of a file that contains sentences
result : a list of sentences divided into tokens
Explanation :
Open the file and save it to 'line', preprocess each sentence and save it as a list in 'file_tokens_pairs'.
This function will return a list of lists that consist of tokens.

lowering() : a function that converts sentence to lowered sentence

```
def lowering(sentence):
    lowered_sentence = []
    for i in range(len(sentence)):
        lowered_sentence.append(sentence[i].lower())
    return ''.join(lowered_sentence)
```

Input : a sentence
result : lowered sentence
Explanation :
Convert each letter of the sentence to lowercase and save it as a list in 'lowered_sentence'. Combine 'lowered_sentence' into a sentence and return it.

calc_similarity() : a function that calculates the similarity between the query and sentences within the search target

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

Input : preprocessed query, preprocessed sentences
result : score dictionary
Explanation :
For each sentence, convert the sentence and query to lowercase and into a set. Save all tokens of the two sentences in 'all_tokens' without duplication. And save the tokens in 'same_tokens' that the two sentences have in common. Calculate the similarity by dividing ‘same_tokens’ by ‘all_tokens’ and save it into ‘score_dict’ as a dictionary. The key of the dictionary is the index of the sentence.

Programming codes

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

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

Input : name of a file that contains search targets
result : a list of sentences divided into tokens
Explanation :
Save search targets into 'file_name', and convert it into a list of sentences divided into tokens by using function 'indexing'.

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

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

Input : a query
result : a list of preprocessed query
Explanation :
Receive a query from the user. Save the input query as a list after pre-processing.

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(preprocessed_query, file_tokens_pairs)
```

Input : a preprocessed query, preprocessed sentences
result : a dictionary of similarities based on a same token set
Explanation :
Calculate the similarities between a query and each sentence by function 'calc_similarity' and save the dictionary of the similarities into 'score_dict'.

④ Rank the sentences based on similarity.

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

Input : a preprocessed query, preprocessed sentences
result : a list of tuples of indexes and similarities
Explanation :
Sort items in descending order by similarity, and save the sorted tuples of indexes and similarities as a list in 'sorted_score_list'. The first index of this list is 'sorted rank', and the second index of this list is that, '0' means 'index', and '1'means 'similarity score'.

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

```
# 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 : a preprocessed query, preprocessed sentences
result : a list of sets of indexes and similarities
Explanation :
If the similarity of 'sorted_score_list[0][1]' with the highest similarity is zero, print 'there is no similar sentence.'
Else, Table the top 10 sentences in order of similarity.

Testing

1. Test Results for each functionality

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

```
file_name = "jhe-koen-dev.en"
indexing(file_name)
```

```
Out[11]: [["You'll",
           'be',
           'picking',
           'fruit',
           'and',
           'generally',
           'helping',
           'us',
           'do',
           'all',
           'the',
           'usual',
           'farm',
           'work.'],
          ['In',
           'the',
           'Middle',
           'Ages,',
           'cities',
```

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

```
query = input("영어 쿼리를 입력하세요.")
preprocessed_query = preprocess(query)
```

```
영어 쿼리를 입력하세요.ThIs Is teSt SeTeNce
```

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

```
print(lowering("Hello My name is Misoo"))
```

```
hello my name is misoo
```

```
calc_similarity(preprocessed_query, file_tokens_pairs)
```

```
영어 쿼리를 입력하세요."ThIs Is teSt SeTeNce"

Out[14]: {0: 0.2916666666666667,
          1: 0.3181818181818182,
          2: 0.2916666666666667,
          3: 0.42105263157894735,
          4: 0.2857142857142857,
          5: 0.3157894736842105,
          6: 0.3888888888888889,
          7: 0.35294117647058826,
          8: 0.3333333333333333,
          9: 0.28,
          10: 0.3181818181818182,
          11: 0.46153846153846156,
          12: 0.3333333333333333,
          13: 0.3684210526315789,
          14: 0.38095238095238093,
          15: 0.3333333333333333,
          16: 0.35294117647058826,
```

④ Rank the sentences based on similarity.

```
22 sorted(score_dict.items(), key = operator.itemgetter(1), reverse=True)
23
24
```

```
영어 쿼리를 입력하세요.ThIs Is teSt SeTeNce

Out[17]: [(130, 0.5454545454545454),
          (293, 0.5454545454545454),
          (17, 0.5384615384615384),
          (156, 0.5384615384615384),
          (11, 0.5),
          (199, 0.5),
          (282, 0.5),
          (453, 0.5),
          (479, 0.5),
          (589, 0.5),
          (597, 0.5),
          (53, 0.4666666666666667),
          (283, 0.4666666666666667),
          (304, 0.4666666666666667),
          (451, 0.4666666666666667),
          (650, 0.4666666666666667),
          (718, 0.4666666666666667),
```

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

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

```
영어 쿼리를 입력하세요.ThIs Is teSt SeTeNce
rank    Index   score   sentence
1       130     0.5454545454545454      It's a nice city.
2       293     0.5454545454545454      B : When is it?
3       17      0.5384615384615384      It is a chemical that cause cance
r.
4       156     0.5384615384615384      They can go inside things.
5       11      0.5     It shines over the sea.
6       199     0.5     White horses are coming into the ring.
7       282     0.5     They were two nice cameras.
8       453     0.5     My mother is sketching them.
9       479     0.5     The second tried counting sheep.
10      589     0.5     It changes its color at times.
```

2. 최종 테스트 스크린샷

```
영어 쿼리를 입력하세요.Hello my name is minho
rank    Index   score   sentence
1       551     0.7142857142857143      They listened to him and did not
ride on them.
2       251     0.6923076923076923      I am nine years old.
3       101     0.6428571428571429      Then he eats it all by himself.
4       173     0.625   So English is used by many people.
5       350     0.625   George helps his team win many games.
6       358     0.625   George is the best baseball player on his team.
7       538     0.625   My hobby is traveling.
8       596     0.625   It shows how life will be fifty years from now.
9       282     0.6     They were two nice cameras.
10      392     0.6     Highways were only for horses.
```

Results and Conclusion

1. Project Results

1. I learned how to preprocess a string using a method related to the string.
2. I leaned how to open a file and how to read it.
3. We constructed a function that makes all sentences lowercase.
4. The conversion and application methods between list, dictionary, and set covered by Python were learned in the process of developing a search engine.

As a result, the search engine was developed, and through this process, the core concepts and application methods of Python were learned.

2. Realization

When I first learned the dictionary and set in Python, I was ashamed of not being good at converting data types, but now I am very happy to be free to deal with data types in this project after studying. The professor gives me a lot of difficult questions, and I think that's very desirable. After all, in order to work as developers or achieve our dreams, we need the ability to solve difficult problems. So I think we should try to solve the problem rather than complain that the problem is difficult. However, what's a little disappointing is that the deadline for the assignment you give seems too short. I hope the professor will delay the deadline for submitting the assignment by at least a day or two.