Search_Engine_Report

과 목 명 :	파이썬프로그래밍및실습
담당교수 :	김미수 교수님
제 출 일 :	2023. 10. 27
학 과 :	토목공학과
학 번 :	192002
이 름 :	나성준

Chonnam National University

- introduction -

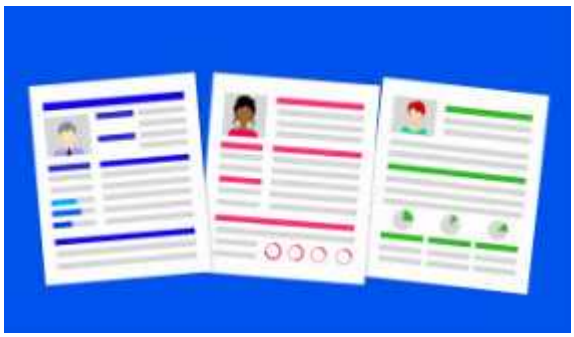
These days, A ton of information is around us.
Science and Industry are developing really fast. IT Technology is growing in the twinkle of an eye. Moreover Many fields are growing. So The quantity of the knowledge is becoming huge.
In this situation, The contents can be overlap each other.
Using this point, We can find the informations more faster.
Get the similarity between the file's contents and my keyword,
And then the most similar sentence may have the information which we find.

In this project, We will make the program for the researching the similarity between the files. Let's begin.

- get to the point -

First, We need to understand the algorithm of the program.

The purpose of this program is comparing similarity between the files.



The files are composed of sentences.
The sentences are composed of the words.
In here, We can find the answer.
We have two files.
If the number of the words which are overlapping in two files is big, it means that two files are similar.

So we will use this logic and make the program.

- Make the functions -

First function. making query.

```
"""
#making the function which makes the query
def preprocess(sentence):
preprocessed_sentence = sentence.strip().split(" ")
return preprocessed_sentence
"""
```

Second function. Read the lines from the files.

```
"""
#making the function which reads the lines from the files
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
"""
```

Third function. Calculating the similarity

```
"""
#making the function which examines the similarity
def calc_similarity(preprocessed_query, preprocessed_sentences):
    score_dict ={}
    for i in range(len(preprocessed_sentences)):
        #No config large_or_small_character
        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)

        #prepare_the_population
        file_token_set = set(file_tokens_pairs[i])

        #make_the_population
        all_tokens = query_token_set | file_token_set

        #pick_up_the_same_tokens
        same_tokens = query_token_set & file_token_set

        #get_the_similarity
        similarity = len(same_tokens) / len(all_tokens)

        score_dict[i] = similarity
    return score_dict
"""
```

Calculating the similarity doesn't distinguish the upper or lower character.

- Coding the Algorithm with the functions -

```
"""
# 1. Indexing
file_name = "searchfile.txt"
file_tokens_pairs = indexing(file_name)

# 2. Input the query from users
query = input("영어 쿼리를 입력하세요.")
preprocessed_query = preprocess(query)
query_token_set = set(preprocessed_query)

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

# 4. Sort the sicilarity 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:
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
"""
```

If you have a file which wants to calculate the similarity, just change the file_name.

Put the file's name with the file's address. And then, Input the keywords which you want to find.

After that, You can see the table which has the sentence's ranks which contain the keyword.

- The Result -

I want to find the key words "time" in the file "search_engine". So I will input "time" in program.

```
영어 쿼리를 입력하세요.time
```

And then, I get the result from the program.

```
rank    Index   score   sentence
1       138     0.14285714285714285     My family will spend more time outdoors.
2       443     0.125   Show time for Jaws are 2:30 and 7:00.
3       532     0.125   He had a difficult time understanding the words.
4       75      0.09090909090909091     Almost every time he is at bat, he hits a home run.
5       499     0.08333333333333333     It took a long time and finally they reached the thirtyeighth floor.
6       104     0.07692307692307693     I have a number of books to read, but I have no time to read them.
7       695     0.07142857142857142     Ramadan is considered to be a time for practicing self-control and showing devotion to Go
d.
8       691     0.06666666666666667     At that time I was looking for an old book called English birds by Johnson.
9       113     0.058823529411764705    At the same time you can compare our way of thinking with the foreign peoples' ways of th
inking, too.
10      569     0.058823529411764705    When males sing a lot, it means they don't have to spend much time hunting for food.
```

Like this, We can find the contents easily with this program.