**searh\_engine report**

183014 이해건

1. Introduction
   1. Project Purpose and Background

This project is about putting Python into practice, which we learned for seven weeks.

* 1. Goal

Learn how to develop a basic search engine that searches files for sentences that are highly similar to the user's sentences

1. 요구사항 Requirements
   1. User requirements

Search engine that searches for similar sentences when a user enters a sentence

* 1. Functional Requirements:
     1. sentences in the search target to the list after preprocessing
     2. Receive input from the user, store it in the list, and remove duplicate values.
     3. Calculate similarities based on a same token set
     4. Sort the similarity list
     5. Print the result

1. Design and Implementation
   1. Implementation Details
      1. Save sentences in the search target to the list after preprocessing

텍스트, 폰트, 스크린샷, 라인이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 폰트, 라인이(가) 표시된 사진

자동 생성된 설명

* + - 1. Input

Jhe-koen-dev.en file

* + - 1. Result

Converted to a list to which all lines in the file are members

* + - 1. description

Take out the sentences in each file as a variable line and save them to the list

* + 1. Receive input from the user, store it in the list, and remove duplicate values.

텍스트, 폰트, 스크린샷, 친필이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 폰트, 라인이(가) 표시된 사진

자동 생성된 설명

* + - 1. Input

User input = query

* + - 1. Result

User input .strip and .split value

* + - 1. description

The values input by the user are listed and aggregated.

* + 1. similarities based on a same token set



텍스트, 스크린샷, 폰트, 문서이(가) 표시된 사진

자동 생성된 설명

* + - 1. Input

query\_token\_set, file\_tokens\_pairs

* + - 1. Result

Score\_dict

* + - 1. description

Extract the sentences from the list one by one as a repeating statement. The user input value, the set of sentences in the file list, and the intersection are obtained, and the similarity is obtained by dividing it.

* + 1. Sort the similarity list.

텍스트, 폰트, 스크린샷, 라인이(가) 표시된 사진

자동 생성된 설명

* + - 1. Input x
      2. Result x
      3. description

The values calculated for similarity are sorted in descending order in the order of high similarity and stored as sorted\_score\_list.

* + 1. Print the result

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

* + - 1. Input x
      2. Result

If the highest similarity is 0.0, print “There is no similar sentence.”

Or print“rank", "Index", "score", "sentence”

Repeated statements output up to similarity order rank == 10

* + - 1. Description

If the highest similarity is 0.0, print “There is no similar sentence.” Alternatively, all the contents from the ranking to the 10th place with high similarity are output through the repeated statement.

1. Testing
   1. Test Results for Each Functionality
      1. Save sentences in the search target to the list after preprocessing

텍스트, 흑백, 화이트, 블랙이(가) 표시된 사진

자동 생성된 설명

* + 1. Receive input from the user, store it in the list, and remove duplicate values.



* + 1. Calculate similarities based on a same token set

텍스트, 흑백, 패턴, 종이이(가) 표시된 사진

자동 생성된 설명

* + 1. Sort the similarity list

텍스트, 흑백, 신문, 종이이(가) 표시된 사진

자동 생성된 설명

* + 1. Print the result

텍스트, 폰트, 스크린샷, 번호이(가) 표시된 사진

자동 생성된 설명

* + 1. Final Test Screenshot

텍스트, 영수증, 화이트, 폰트이(가) 표시된 사진

자동 생성된 설명

1. Results and Conclusion
   1. Result

Through the similarity of words inside the sentence, a program that searches for similar sentences could be established and supplemented.

* 1. Conclusion

At first, it felt quite difficult, but when I opened each line, it was amazing to see that it was built within what we know, and if I improve my skills later, I want to design something like this.