Python Programming and Practice
: Project practice Report

Major : IoT Artificial Intelligence Convergence

School ID : 182207

Name : Seokyoung Kim

- Simple Search Engine

1. Introduction

1. 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.
2. Goal: To develop a basic search engine that retrieves sentences similar to the user's query.

2. 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
      • Similarity is based on the count of the same “word."
   ④ Rank the sentences based on similarity.
   ⑤ Output the top 10 ranked sentences to the user from the ranked sentences.

3. Design and Implementation

1. Implementation Details:
(Detailed explanation of how each requirement was implemented, including relevant code snippets)

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

The **‘preprocess’** function is defined to preprocess a sentence by converting it to lowercase, splitting it into words and creating a set of words. This preprocessed data is stored in the **‘file_tokens_pairs’** list.

- Preprocess function.

```
# 1. Preprocess function
def preprocess(sentence):
    # Convert the sentence to lowercase, split it into words, and create a set of words.
    return set(sentence.strip().lower().split(" "))
```

```
# 1. Indexing
file_name = "jhe-koen-dev.en"
file_tokens_pairs = indexing(file_name)
```

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

The user is prompted to input an English query using the **'input'** function, and the provided query is preprocessed using the **'preprocess'** function.

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

- Indexing function.

```
# 2. Indexing function
def indexing(file_name):
    file_tokens_pairs = []
    lines = open(file_name, "r", encoding="utf8").readlines()  # Read the lines from a file
    for line in lines:
        tokens = preprocess(line)
        file_tokens_pairs.append(tokens)
        #print(tokens)
    return file_tokens_pairs
```

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

The **'calc_similarity'** function calculates the similarity between the preprocessed query and each preprocessed sentence from the search target. It uses a set-based approach, where it calculates the Jaccard similarity coefficient.

- Calculate similarity function.

```
# 3. Calculate similarity function
def calc_similarity(preprocessed_query, preprocessed_sentences):
    score_dict = {}
    query_token_set = set(preprocessed_query)

    for i, sentence_tokens in enumerate(preprocessed_sentences):
        all_tokens = query_token_set | sentence_tokens  # Union of query tokens and sentence tokens.
        same_tokens = query_token_set & sentence_tokens  # Intersection of query tokens and sentence tokens.
        similarity = len(same_tokens) / len(all_tokens)  # Calculate similarity
        score_dict[i] = similarity  # Store the similarity score

    return score_dict
```

In the main part of the code

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

④ Rank the sentences based on similarity.

The similarity scores calculated in step 3 are stored in a dictionary where the keys are the index of the sentence in the search target. These scores are then sorted in descending order using the **'sorted'** function.

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

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

The code prints the top 10 ranked sentences to the user. If the top-ranked sentence has a similarity score of 0.0, it informs the user that there is no similar sentence.

```
# 1. Indexing
file_name = "jhe-koen-dev.en"
file_tokens_pairs = indexing(file_name)

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

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

# 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:
    print("There is no similar sentence.")
else:
    print("rank", "Index", "score", "sentence", sep="\t")
    rank = 1
    for i, score in sorted_score_list[:10]:
        print(rank, i, score, ' '.join(file_tokens_pairs[i]), sep="\t")
        rank += 1
```

output results

```
영어 쿼리를 입력하세요 : Hello My name is Seokyoung Kim
rank    Index   score   sentence
1       679     0.42857142857142855     my name is mike.
2       526     0.25    my is bob brother.
3       538     0.25    my hobby traveling. is
4       453     0.2222222222222222      sketching mother them. is my
5       241     0.2     father running is my with so-ra.
6       336     0.2     at is my family the park.
7       212     0.18181818181818182     sister waiting betty for is my me.
8       505     0.16666666666666666     sister annie years little old. five is my
9       610     0.14285714285714285     yell, would i ready!" is voice my and raise "lunch
10      190     0.125   sunday. is it
```

My code provides a basic implementation of a text similarity search system that allows users to input an English query and find the most similar sentences from the search target based on word overlap. The similarity scores are calculated using the Jaccard coefficient, and the top 10 ranked sentences are presented to the user.

4. Testing

1. Test Results for Each Functionality: (Description of test cases and corresponding screenshots for each requirement)

- Progress function test

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

Test code

```
영어 쿼리를 입력하세요 : Hello
{'hello'}
There is no similar sentence.
```

- Calculate similarity function test

```python
# 3. Calculate similarity function
def calc_similarity(preprocessed_query, preprocessed_sentences):
    score_dict = {}
    query_token_set = set(preprocessed_query)

    for i, sentence_tokens in enumerate(preprocessed_sentences):
        all_tokens = query_token_set | sentence_tokens  # Union of query tokens and sentence tokens.
        same_tokens = query_token_set & sentence_tokens  # Intersection of query tokens and sentence tokens.
        similarity = len(same_tokens) / len(all_tokens)  # Calculate similarity
        score_dict[i] = similarity  # Store the similarity score
    print(score_dict)  Test code

    return score_dict
```

```
영어 쿼리를 입력하세요 : Hello
{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: 0.0, 18: 0.0, 19: 0.0, 20: 0.0, 21: 0.0, 22: 0.0, 23: 0.0, 24: 0.0, 25: 0.0, 26:
0.0, 27: 0.0, 28: 0.0, 29: 0.0, 30: 0.0, 31: 0.0, 32: 0.0, 33: 0.0, 34: 0.0, 35: 0.0, 36: 0.0, 37: 0.0, 38: 0.0, 3
9: 0.0, 40: 0.0, 41: 0.0, 42: 0.0, 43: 0.0, 44: 0.0, 45: 0.0, 46: 0.0, 47: 0.0, 48: 0.0, 49: 0.0, 50: 0.0, 51: 0.0,
52: 0.0, 53: 0.0, 54: 0.0, 55: 0.0, 56: 0.0, 57: 0.0, 58: 0.0, 59: 0.0, 60: 0.0, 61: 0.0, 62: 0.0, 63: 0.0, 64: 0.
0, 65: 0.0, 66: 0.0, 67: 0.0, 68: 0.0, 69: 0.0, 70: 0.0, 71: 0.0, 72: 0.0, 73: 0.0, 74: 0.0, 75: 0.0, 76: 0.0, 77:
0.0, 78: 0.0, 79: 0.0, 80: 0.0, 81: 0.0, 82: 0.0, 83: 0.0, 84: 0.0, 85: 0.0, 86: 0.0, 87: 0.0, 88: 0.0, 89: 0.0, 9
0: 0.0, 91: 0.0, 92: 0.0, 93: 0.0, 94: 0.0, 95: 0.0, 96: 0.0, 97: 0.0, 98: 0.0, 99: 0.0, 100: 0.0, 101: 0.0, 102:
0.0, 103: 0.0, 104: 0.0, 105: 0.0, 106: 0.0, 107: 0.0, 108: 0.0, 109: 0.0, 110: 0.0, 111: 0.0, 112: 0.0, 113: 0.0,
114: 0.0, 115: 0.0, 116: 0.0, 117: 0.0, 118: 0.0, 119: 0.0, 120: 0.0, 121: 0.0, 122: 0.0, 123: 0.0, 124: 0.0, 125:
0.0, 126: 0.0, 127: 0.0, 128: 0.0, 129: 0.0, 130: 0.0, 131: 0.0, 132: 0.0, 133: 0.0, 134: 0.0, 135: 0.0, 136: 0.0,
137: 0.0, 138: 0.0, 139: 0.0, 140: 0.0, 141: 0.0, 142: 0.0, 143: 0.0, 144: 0.0, 145: 0.0, 146: 0.0, 147: 0.0, 148:
0.0, 149: 0.0, 150: 0.0, 151: 0.0, 152: 0.0, 153: 0.0, 154: 0.0, 155: 0.0, 156: 0.0, 157: 0.0, 158: 0.0, 159: 0.0,
160: 0.0, 161: 0.0, 162: 0.0, 163: 0.0, 164: 0.0, 165: 0.0, 166: 0.0, 167: 0.0, 168: 0.0, 169: 0.0, 170: 0.0, 171:
0.0, 172: 0.0, 173: 0.0, 174: 0.0, 175: 0.0, 176: 0.0, 177: 0.0, 178: 0.0, 179: 0.0, 180: 0.0, 181: 0.0, 182: 0.0,
183: 0.0, 184: 0.0, 185: 0.0, 186: 0.0, 187: 0.0, 188: 0.0, 189: 0.0, 190: 0.0, 191: 0.0, 192: 0.0, 193: 0.0, 194:
0.0, 195: 0.0, 196: 0.0, 197: 0.0, 198: 0.0, 199: 0.0, 200: 0.0, 201: 0.0, 202: 0.0, 203: 0.0, 204: 0.0, 205: 0.0,
206: 0.0, 207: 0.0, 208: 0.0, 209: 0.0, 210: 0.0, 211: 0.0, 212: 0.0, 213: 0.0, 214: 0.0, 215: 0.0, 216: 0.0, 217:
0.0, 218: 0.0, 219: 0.0, 220: 0.0, 221: 0.0, 222: 0.0, 223: 0.0, 224: 0.0, 225: 0.0, 226: 0.0, 227: 0.0, 228: 0.0,
229: 0.0, 230: 0.0, 231: 0.0, 232: 0.0, 233: 0.0, 234: 0.0, 235: 0.0, 236: 0.0, 237: 0.0, 238: 0.0, 239: 0.0, 240:
0.0, 241: 0.0, 242: 0.0, 243: 0.0, 244: 0.0, 245: 0.0, 246: 0.0, 247: 0.0, 248: 0.0, 249: 0.0, 250: 0.0, 251: 0.0,
252: 0.0, 253: 0.0, 254: 0.0, 255: 0.0, 256: 0.0, 257: 0.0, 258: 0.0, 259: 0.0, 260: 0.0, 261: 0.0, 262: 0.0, 263:
0.0, 264: 0.0, 265: 0.0, 266: 0.0, 267: 0.0, 268: 0.0, 269: 0.0, 270: 0.0, 271: 0.0, 272: 0.0, 273: 0.0, 274: 0.0,
275: 0.0, 276: 0.0, 277: 0.0, 278: 0.0, 279: 0.0, 280: 0.0, 281: 0.0, 282: 0.0, 283: 0.0, 284: 0.0, 285: 0.0, 286:
0.0, 287: 0.0, 288: 0.0, 289: 0.0, 290: 0.0, 291: 0.0, 292: 0.0, 293: 0.0, 294: 0.0, 295: 0.0, 296: 0.0, 297: 0.0,
298: 0.0, 299: 0.0, 300: 0.0, 301: 0.0, 302: 0.0, 303: 0.0, 304: 0.0, 305: 0.0, 306: 0.0, 307: 0.0, 308: 0.0, 309:
0.0, 310: 0.0, 311: 0.0, 312: 0.0, 313: 0.0, 314: 0.0, 315: 0.0, 316: 0.0, 317: 0.0, 318: 0.0, 319: 0.0, 320: 0.0,
321: 0.0, 322: 0.0, 323: 0.0, 324: 0.0, 325: 0.0, 326: 0.0, 327: 0.0, 328: 0.0, 329: 0.0, 330: 0.0, 331: 0.0, 332:
0.0, 333: 0.0, 334: 0.0, 335: 0.0, 336: 0.0, 337: 0.0, 338: 0.0, 339: 0.0, 340: 0.0, 341: 0.0, 342: 0.0, 343: 0.0,
344: 0.0, 345: 0.0, 346: 0.0, 347: 0.0, 348: 0.0, 349: 0.0, 350: 0.0, 351: 0.0, 352: 0.0, 353: 0.0, 354: 0.0, 355:
0.0, 356: 0.0, 357: 0.0, 358: 0.0, 359: 0.0, 360: 0.0, 361: 0.0, 362: 0.0, 363: 0.0, 364: 0.0, 365: 0.0, 366: 0.0,
367: 0.0, 368: 0.0, 369: 0.0, 370: 0.0, 371: 0.0, 372: 0.0, 373: 0.0, 374: 0.0, 375: 0.0, 376: 0.0, 377: 0.0, 378:
0.0, 379: 0.0, 380: 0.0, 381: 0.0, 382: 0.0, 383: 0.0, 384: 0.0, 385: 0.0, 386: 0.0, 387: 0.0, 388: 0.0, 389: 0.0,
```

- Indexing function test

```python
# 2. Indexing function
def indexing(file_name):
    file_tokens_pairs = []
    lines = open(file_name, "r", encoding="utf8").readlines()  # Read the lines from a file
    for line in lines:
        tokens = preprocess(line)
        file_tokens_pairs.append(tokens)
        print(tokens)  Test Code
    return file_tokens_pairs
```

```
{'the', 'and', "you'll", 'fruit', 'generally', 'be', 'usual', 'picking', 'us', 'work.', 'farm', 'helping', 'all', '
do'}
{'the', 'and', 'streets', 'filled', 'middle', 'with', 'very', 'in', 'garbage.', 'not', 'ages,', 'were', 'cities', '
clean,'}
{'catch', 'be', 'may', 'or', 'apron', 'sooner', 'with', 'for', 'progressive', 'strings,', 'their', 'moment', 'yet',
'will', 'up', 'the', 'society', 'they', 'hiding', 'but', 'behind', 'world.', 'later'}
{'you', 'the', 'said', 'answered?"', 'cow', 'what', 'minister.', 'do', 'know'}
{'and', 'different', 'countries.', 'very', 'poland', 'like', 'may', 'italy', 'seem'}
{'the', 'and', 'mr.', 'smith', 'in', 'day', 'whole', 'oxford.', 'stayed', 'i'}
{'the', 'signal', 'gave', 'sight', 'an', 'of', 'red', 'him', 'a', 'idea.', 'traffic'}
{'they', 'pumpkins', 'so', 'used', 'instead.'}
{'might', 'much', 'they', '2.', 'affairs:', 'of', 'particular', 'not', 'me', 'occasion', 'a', 'state', 'offer', 'mo
ney.'}
{'interested', "you'll", 'in', "i'm", 'especially', 'information', 'so', 'about', 'skills,', 'hope', 'i', 'horse-ri
ding', 'this.', 'learning,', 'include'}
{'gave', 'the', 'through', 'darkness.', 'devil', 'his', 'candle', 'him', 'single', 'a', 'way', 'instead,', 'light',
'to'}
```

2. Final Test Screenshot:: ([Comprehensive screenshot demonstrating the overall functionality of the program)

- If there is no similar sentence

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

- If there is similar sentence

```
영어 쿼리를 입력하세요 : Hello My name is Seokyoung Kim
rank    Index   score   sentence
1       679     0.42857142857142855     my name is mike.
2       526     0.25    my is bob brother.
3       538     0.25    my hobby traveling. is
4       453     0.2222222222222222      sketching mother them. is my
5       241     0.2     father running is my with so-ra.
6       336     0.2     at is my family the park.
7       212     0.18181818181818182     sister waiting betty for is my me.
8       505     0.16666666666666666     sister annie years little old. five is my
9       610     0.14285714285714285     yell, would i ready!" is voice my and raise "lunch
10      190     0.125   sunday. is it
```

```
영어 쿼리를 입력하세요 : Hello My name is Misoo Kim
rank    Index   score   sentence
1       679     0.42857142857142855     my name is mike.
2       526     0.25    my is bob brother.
3       538     0.25    my hobby traveling. is
4       453     0.2222222222222222      sketching mother them. is my
5       241     0.2     father running is my with so-ra.
6       336     0.2     at is my family the park.
7       212     0.18181818181818182     sister waiting betty for is my me.
8       505     0.16666666666666666     sister annie years little old. five is my
9       610     0.14285714285714285     yell, would i ready!" is voice my and raise "lunch
10      190     0.125   sunday. is it
```

5. Results and Conclusion

1. Result: The successful completion of the search engine development project is a notable achievement. It signifies that all the planned tasks and objectives were effectively executed, leading to a functioning search engine.

2. Conclusion: The experience of developing the search engine was quite challenging, akin to first learning basic arithmetic and then facing the complexities of multiplication. It required not only a strong foundation but also the ability to handle more intricate and demanding tasks, demonstrating the depth of knowledge and skills required in the project.