Information Retrieval

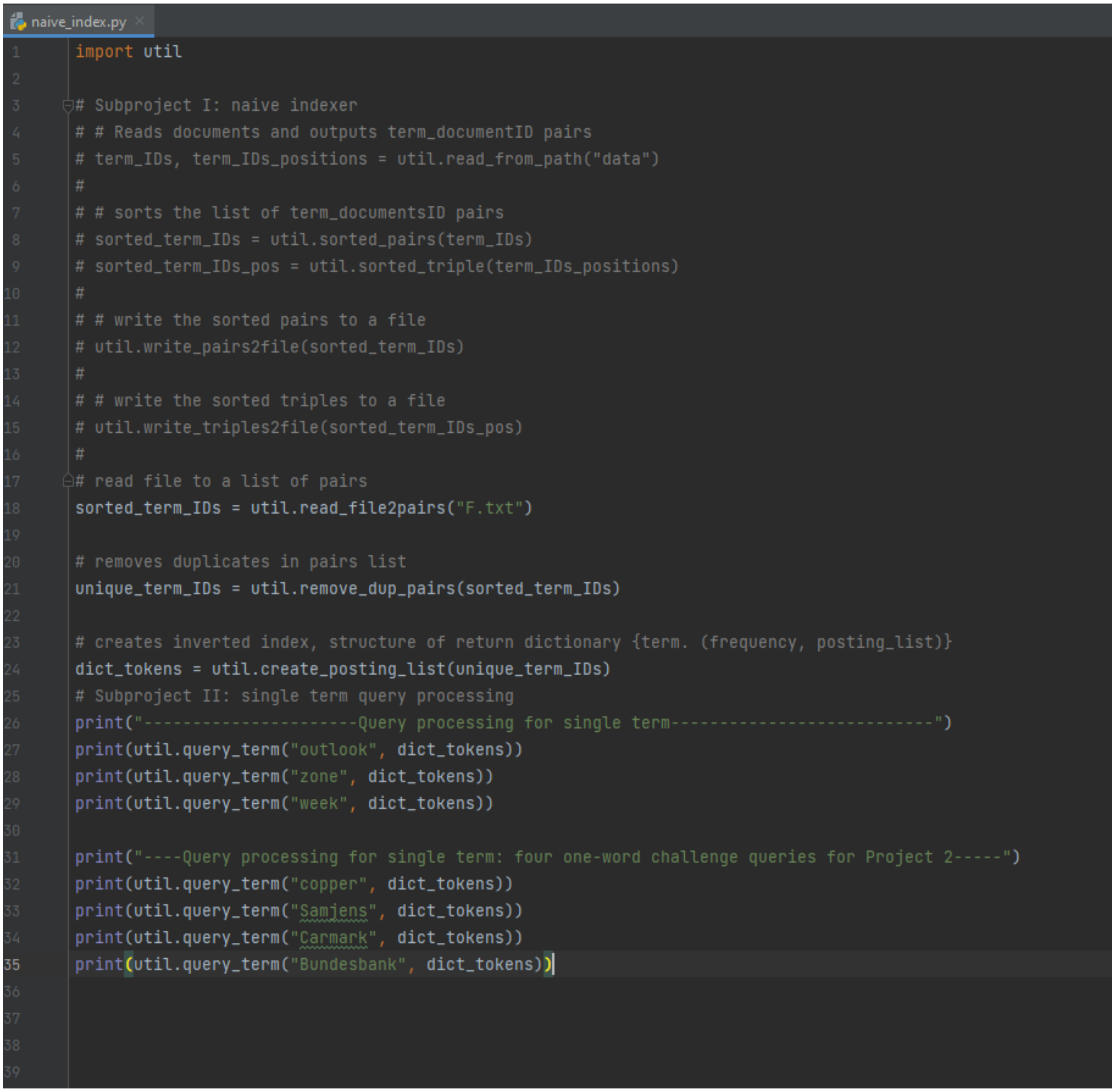
COMP 479 Project 2
Text preprocessing with NLTK DEMO

Lin Ling (40153877)

A report submitted in partial fulfilment of the requirements of Comp479.

Concordia University

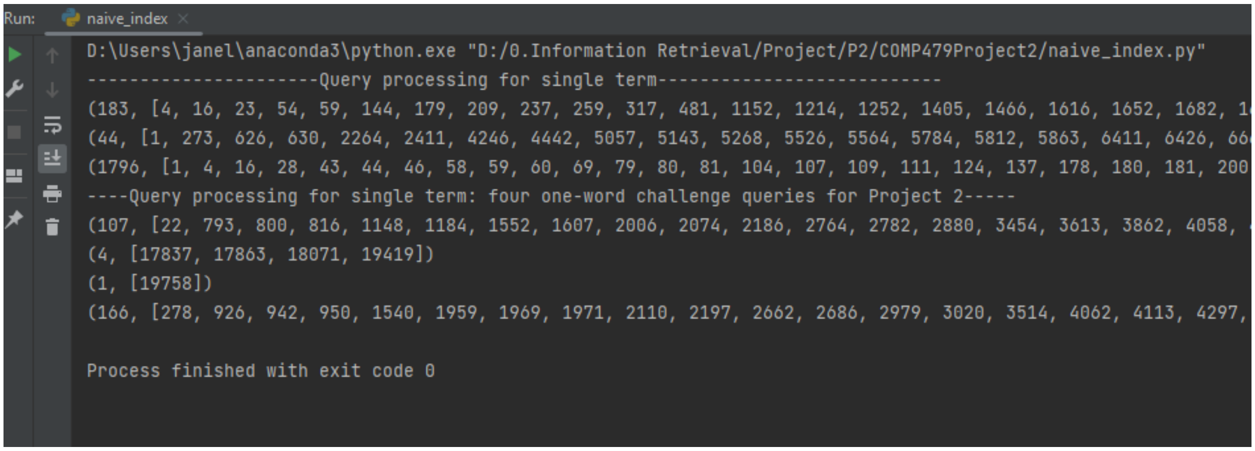
Source Code: naïve_index.py



```
1   import util
2
3   # Subproject I: naive indexer
4   # # Reads documents and outputs term_documentID pairs
5   # term_IDs, term_IDs_positions = util.read_from_path("data")
6   #
7   # # sorts the list of term_documentsID pairs
8   # sorted_term_IDs = util.sorted_pairs(term_IDs)
9   # sorted_term_IDs_pos = util.sorted_triple(term_IDs_positions)
10  #
11  # # write the sorted pairs to a file
12  # util.write_pairs2file(sorted_term_IDs)
13  #
14  # # write the sorted triples to a file
15  # util.write_triples2file(sorted_term_IDs_pos)
16  #
17  # read file to a list of pairs
18  sorted_term_IDs = util.read_file2pairs("F.txt")
19
20  # removes duplicates in pairs list
21  unique_term_IDs = util.remove_dup_pairs(sorted_term_IDs)
22
23  # creates inverted index, structure of return dictionary {term. (frequency, posting_list)}
24  dict_tokens = util.create_posting_list(unique_term_IDs)
25  # Subproject II: single term query processing
26  print("----------------------Query processing for single term---------------------------")
27  print(util.query_term("outlook", dict_tokens))
28  print(util.query_term("zone", dict_tokens))
29  print(util.query_term("week", dict_tokens))
30
31  print("----Query processing for single term: four one-word challenge queries for Project 2-----")
32  print(util.query_term("copper", dict_tokens))
33  print(util.query_term("Samjens", dict_tokens))
34  print(util.query_term("Carmark", dict_tokens))
35  print(util.query_term("Bundesbank", dict_tokens))
36
37
38
39
```

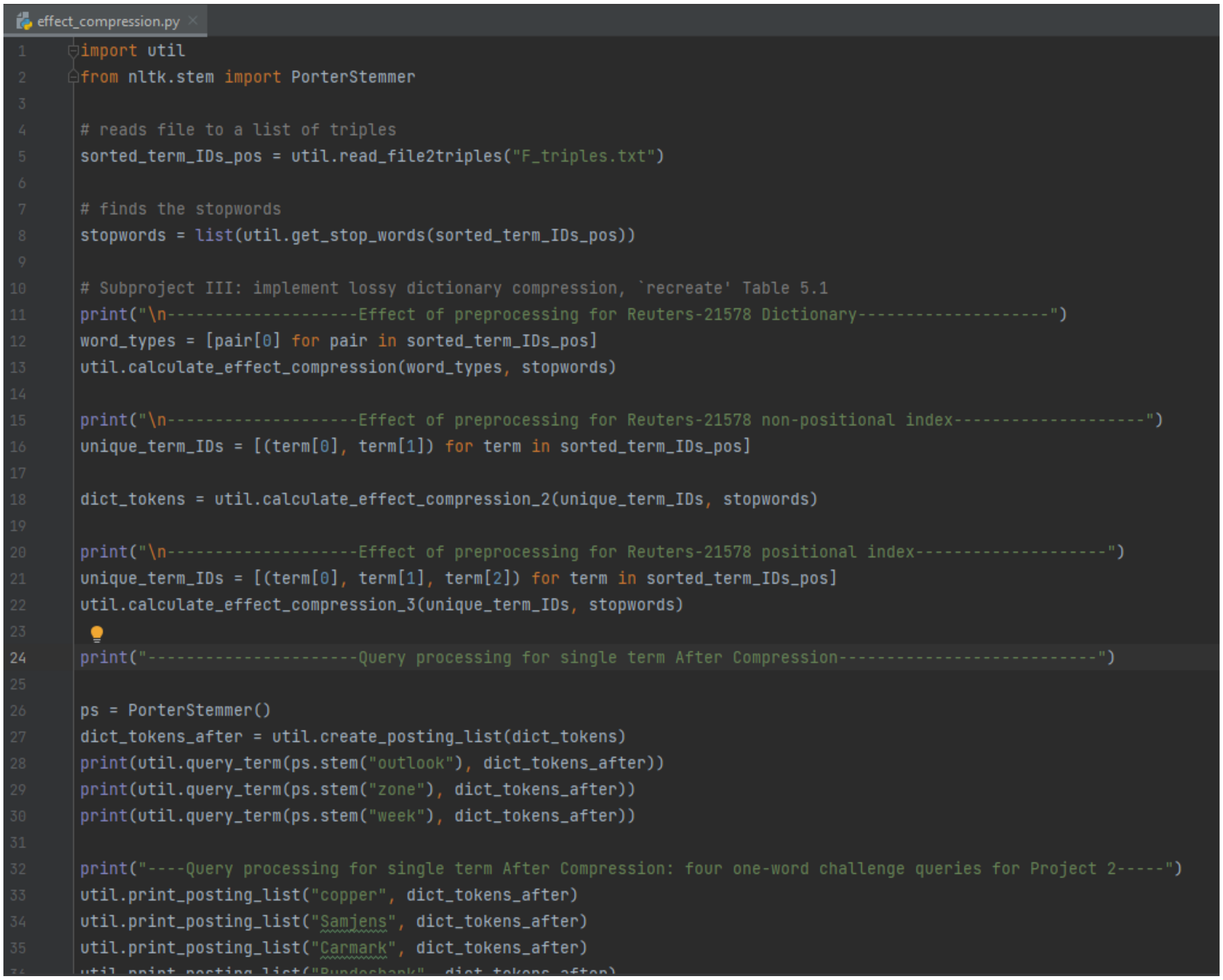
Execute the naïve_index.py will print out the query processing results of three sample queries by myself and four one-word challenge queries.

Note: For saving time, comment out the code from line 3 to line 17, as it will take a long time to read 21578 files in data folder and generate the list of term_ids and term_ids_positions to a file which already run and got the file F.txt and F_triples.txt



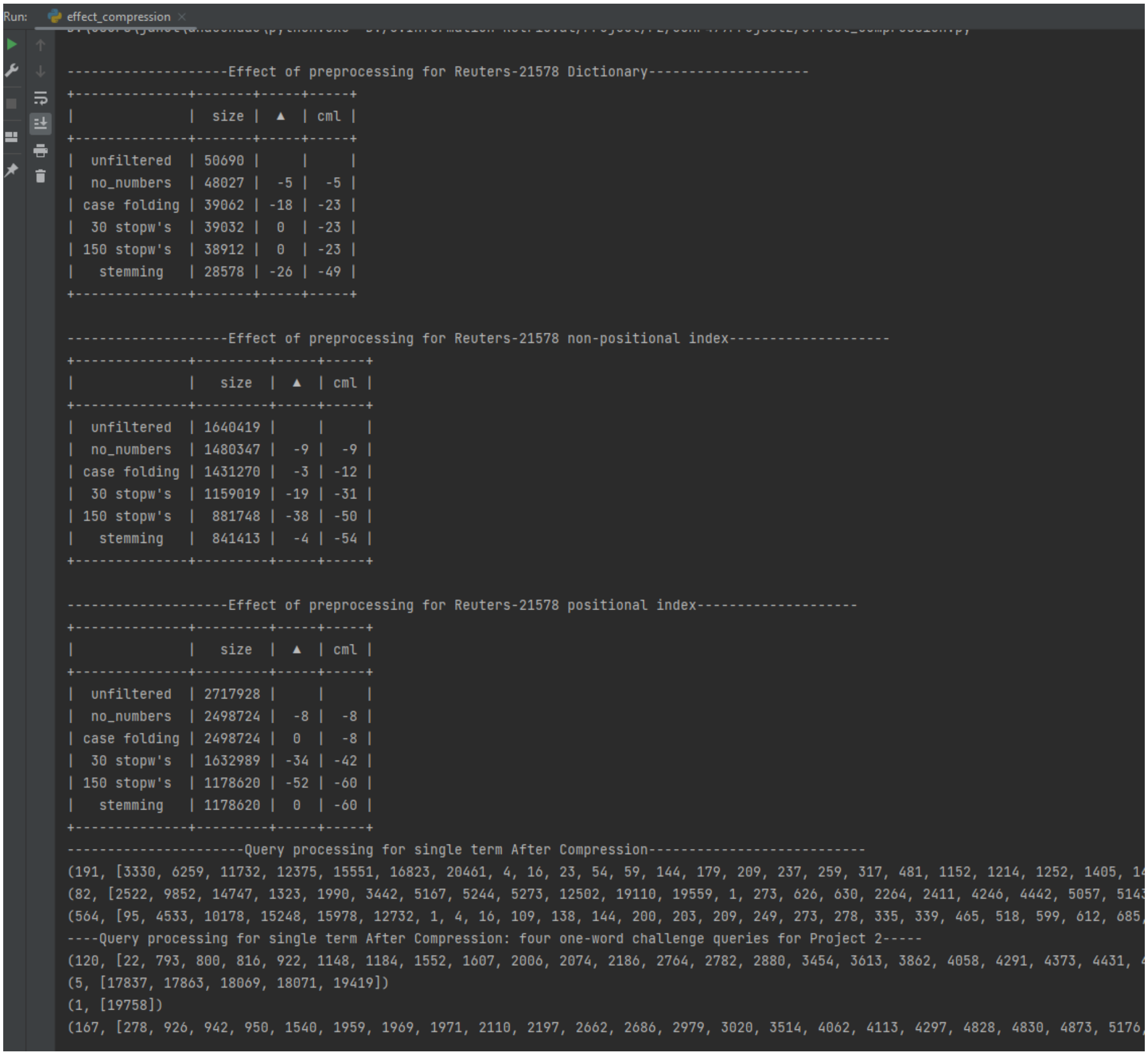
```
D:\Users\janel\anaconda3\python.exe "D:/0.Information Retrieval/Project/P2/COMP479Project2/naive_index.py"
----------------------Query processing for single term---------------------------
(183, [4, 16, 23, 54, 59, 144, 179, 209, 237, 259, 317, 481, 1152, 1214, 1252, 1405, 1466, 1616, 1652, 1682, 1
(44, [1, 273, 626, 630, 2264, 2411, 4246, 4442, 5057, 5143, 5268, 5526, 5564, 5784, 5812, 5863, 6411, 6426, 66
(1796, [1, 4, 16, 28, 43, 44, 46, 58, 59, 60, 69, 79, 80, 81, 104, 107, 109, 111, 124, 137, 178, 180, 181, 200
----Query processing for single term: four one-word challenge queries for Project 2-----
(107, [22, 793, 800, 816, 1148, 1184, 1552, 1607, 2006, 2074, 2186, 2764, 2782, 2880, 3454, 3613, 3862, 4058,
(4, [17837, 17863, 18071, 19419])
(1, [19758])
(166, [278, 926, 942, 950, 1540, 1959, 1969, 1971, 2110, 2197, 2662, 2686, 2979, 3020, 3514, 4062, 4113, 4297,

Process finished with exit code 0
```

Source code: effect_compression.py



```python
import util
from nltk.stem import PorterStemmer

# reads file to a list of triples
sorted_term_IDs_pos = util.read_file2triples("F_triples.txt")

# finds the stopwords
stopwords = list(util.get_stop_words(sorted_term_IDs_pos))

# Subproject III: implement lossy dictionary compression, `recreate' Table 5.1
print("\n--------------------Effect of preprocessing for Reuters-21578 Dictionary--------------------")
word_types = [pair[0] for pair in sorted_term_IDs_pos]
util.calculate_effect_compression(word_types, stopwords)

print("\n--------------------Effect of preprocessing for Reuters-21578 non-positional index--------------------")
unique_term_IDs = [(term[0], term[1]) for term in sorted_term_IDs_pos]

dict_tokens = util.calculate_effect_compression_2(unique_term_IDs, stopwords)

print("\n--------------------Effect of preprocessing for Reuters-21578 positional index--------------------")
unique_term_IDs = [(term[0], term[1], term[2]) for term in sorted_term_IDs_pos]
util.calculate_effect_compression_3(unique_term_IDs, stopwords)

print("----------------------Query processing for single term After Compression---------------------------")

ps = PorterStemmer()
dict_tokens_after = util.create_posting_list(dict_tokens)
print(util.query_term(ps.stem("outlook"), dict_tokens_after))
print(util.query_term(ps.stem("zone"), dict_tokens_after))
print(util.query_term(ps.stem("week"), dict_tokens_after))

print("----Query processing for single term After Compression: four one-word challenge queries for Project 2-----")
util.print_posting_list("copper", dict_tokens_after)
util.print_posting_list("Samjens", dict_tokens_after)
util.print_posting_list("Carmark", dict_tokens_after)
util.print_posting_list("Bundesbank", dict_tokens_after)
```

After implementation, it will generate three tables to show the changed rate by different data retaining in different lossy compression steps. In the meantime, it will print out the query processing results of three sample queries by myself and four one-word challenge queries after lossy compression steps.



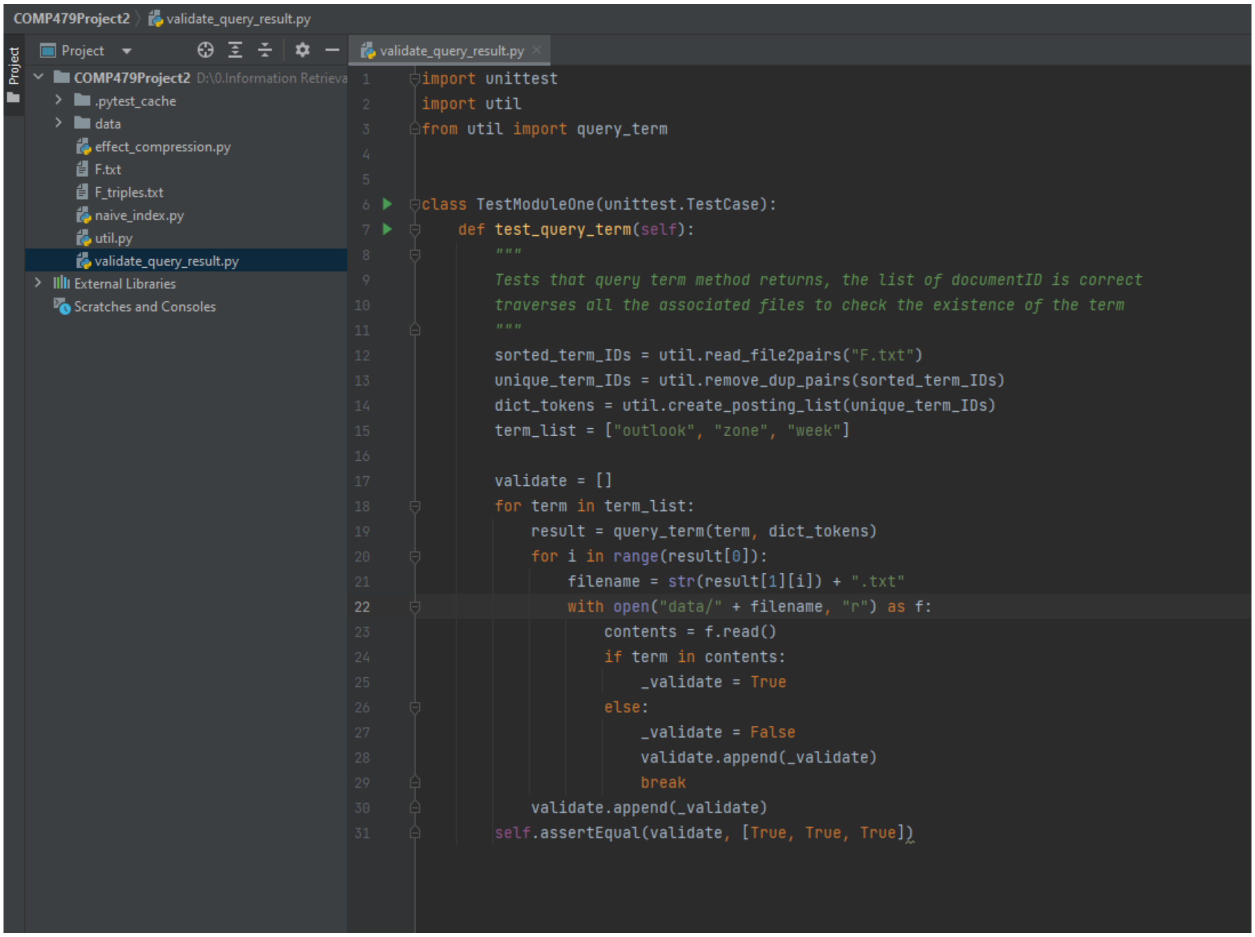
```
--------------------Effect of preprocessing for Reuters-21578 Dictionary--------------------
+-------------+-------+-----+-----+
|             |  size |  ▲  | cml |
+-------------+-------+-----+-----+
|  unfiltered | 50690 |     |     |
|  no_numbers | 48027 |  -5 |  -5 |
| case folding | 39062 | -18 | -23 |
|  30 stopw's  | 39032 |  0  | -23 |
| 150 stopw's  | 38912 |  0  | -23 |
|   stemming   | 28578 | -26 | -49 |
+-------------+-------+-----+-----+

--------------------Effect of preprocessing for Reuters-21578 non-positional index--------------------
+-------------+---------+-----+-----+
|             |   size  |  ▲  | cml |
+-------------+---------+-----+-----+
|  unfiltered | 1640419 |     |     |
|  no_numbers | 1480347 |  -9 |  -9 |
| case folding | 1431270 |  -3 | -12 |
|  30 stopw's  | 1159019 | -19 | -31 |
| 150 stopw's  |  881748 | -38 | -50 |
|   stemming   |  841413 |  -4 | -54 |
+-------------+---------+-----+-----+

--------------------Effect of preprocessing for Reuters-21578 positional index--------------------
+-------------+---------+-----+-----+
|             |   size  |  ▲  | cml |
+-------------+---------+-----+-----+
|  unfiltered | 2717928 |     |     |
|  no_numbers | 2498724 |  -8 |  -8 |
| case folding | 2498724 |  0  |  -8 |
|  30 stopw's  | 1632989 | -34 | -42 |
| 150 stopw's  | 1178620 | -52 | -60 |
|   stemming   | 1178620 |  0  | -60 |
+-------------+---------+-----+-----+
----------------------Query processing for single term After Compression---------------------------
(191, [3330, 6259, 11732, 12375, 15551, 16823, 20461, 4, 16, 23, 54, 59, 144, 179, 209, 237, 259, 317, 481, 1152, 1214, 1252, 1405, 14
(82, [2522, 9852, 14747, 1323, 1990, 3442, 5167, 5244, 5273, 12502, 19110, 19559, 1, 273, 626, 630, 2264, 2411, 4246, 4442, 5057, 5143
(564, [95, 4533, 10178, 15248, 15978, 12732, 1, 4, 16, 109, 138, 144, 200, 203, 209, 249, 273, 278, 335, 339, 465, 518, 599, 612, 685,
----Query processing for single term After Compression: four one-word challenge queries for Project 2-----
(120, [22, 793, 800, 816, 922, 1148, 1184, 1552, 1607, 2006, 2074, 2186, 2764, 2782, 2880, 3454, 3613, 3862, 4058, 4291, 4373, 4431, 4
(5, [17837, 17863, 18069, 18071, 19419])
(1, [19758])
(167, [278, 926, 942, 950, 1540, 1959, 1969, 1971, 2110, 2197, 2662, 2686, 2979, 3020, 3514, 4062, 4113, 4297, 4828, 4830, 4873, 5176,
```

Source Code: util.py

This file contains all the function which used in naïve_index.py and effect_compression.py.

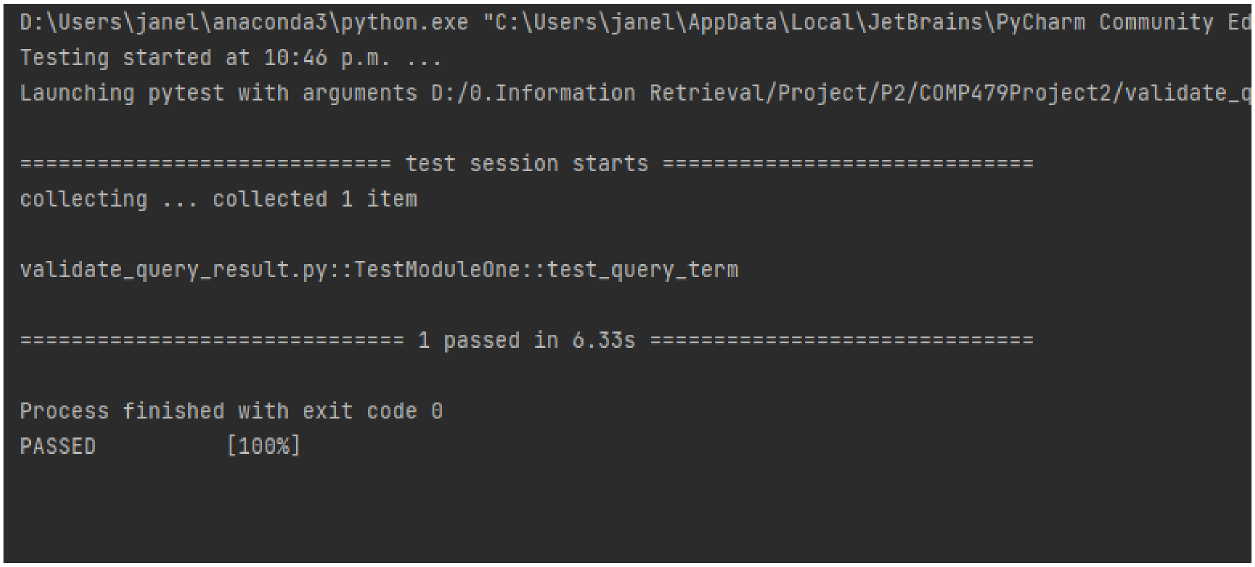
Source Code: Validate_query_result.py

Do the test case by unit test for validate query result



```python
import unittest
import util
from util import query_term


class TestModuleOne(unittest.TestCase):
    def test_query_term(self):
        """
        Tests that query term method returns, the list of documentID is correct
        traverses all the associated files to check the existence of the term
        """
        sorted_term_IDs = util.read_file2pairs("F.txt")
        unique_term_IDs = util.remove_dup_pairs(sorted_term_IDs)
        dict_tokens = util.create_posting_list(unique_term_IDs)
        term_list = ["outlook", "zone", "week"]

        validate = []
        for term in term_list:
            result = query_term(term, dict_tokens)
            for i in range(result[0]):
                filename = str(result[1][i]) + ".txt"
                with open("data/" + filename, "r") as f:
                    contents = f.read()
                    if term in contents:
                        _validate = True
                    else:
                        _validate = False
                        validate.append(_validate)
                        break
            validate.append(_validate)
        self.assertEqual(validate, [True, True, True])
```

Test results:



```
D:\Users\janel\anaconda3\python.exe "C:\Users\janel\AppData\Local\JetBrains\PyCharm Community Ed
Testing started at 10:46 p.m. ...
Launching pytest with arguments D:/0.Information Retrieval/Project/P2/COMP479Project2/validate_q

============================= test session starts ==============================
collecting ... collected 1 item

validate_query_result.py::TestModuleOne::test_query_term

============================== 1 passed in 6.33s ===============================

Process finished with exit code 0
PASSED          [100%]
```