Information Retrieval

COMP 479 Project 2
Text preprocessing with NLTK Report

Lin Ling (40153877)

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

Concordia University

INDEX

1. Subproject I: naive indexer

a. Source Code: naïve_index.py

util.py

1.1 Processes documents to lists of tokens and outputs a list F of term-documentID pairs

Uses the modules created in project 1 to read the articles from the sgm files and output 21578 documents file with a list of terms without the punctuation and named with the newID of each article, and call method in util.py, <read_from_path> to get the list of term_documentsID pairs

```
def read_from_path(path):
    """
    reads the document as a list of tokens and outputs term_documentsID pairs
to a list
    :param path: the directory of the data
    :return: term_documentsID pairs, term_documentsID_positions triples
    """
```

1.2 Sorts and remove duplicates

Uses the methods in util.py, <sorted_pairs> <remove_dup_pairs>

```
def sorted_pairs(pairs_list):
    """
    sorts the list of term_documentsID pairs
    :param pairs_list: unsorted term_documentsID pairs
    :return:sorted term_documentsID pairs
```

```
def remove_dup_pairs(pairs_list):

    """
    removes duplicates in pairs list
    :param pairs_list: term_documentsID pairs with duplicate
    :return:term_documentsID pairs without duplicate
    """
```

Writes the sorted pairs to a file F.txt to save the times to read all the files at each execute. And reads the sorted pairs from the file for the following processing.

1.3 Creates inverted index with posting list

Creates inverted index, structure of return dictionary {term. (frequency, posting_list)}

```
def create_posting_list(pairs_list):
    """
    creates inverted index, structure of return dictionary {term. (frequency,
posting_list)}
    :param pairs_list: term_documentsID pairs without duplicate
    :return: dictionary of term associated with posting list
    """
```

2. Subproject II: single term query processing

2.1 Implement a query processor for single term queries

b. Source Code: naïve_index.py

util.py

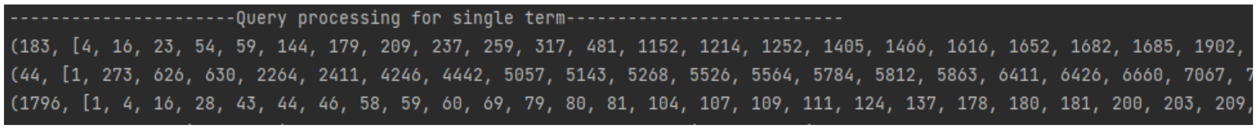
```
def query_term(term, dict_tokens):
    """
    implement a query processor for single term queries
    :param term: single term for query
    :param dict_tokens: the dictionary contains the information of  tokens'
frequency and posting list
    :return: the posting list of the term
    """
```

Three sample queries:

a. “outlook”: frequency :183

b. “zone”: frequency:44

c. “week”: frequency:1796



```
----------------------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, 1685, 1902,
(44, [1, 273, 626, 630, 2264, 2411, 4246, 4442, 5057, 5143, 5268, 5526, 5564, 5784, 5812, 5863, 6411, 6426, 6660, 7067,
(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, 203, 209,
```

Given challenge queries results:

a. “copper”: frequency :107
b. “Samjens”: frequency:4
c. “Carmark”: frequency:1
d. “Bundesbank”: frequency:166:

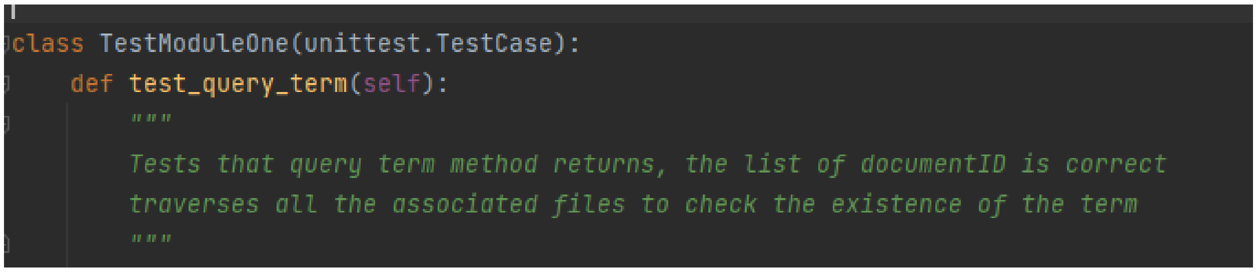
Print out frequency and term posting list:

```
(107, [22, 793, 800, 816, 1148, 1184, 1552, 1607, 2006, 2074, 2186, 2764, 2782, 2880, 3454, 3613, 3862, 4058, 4291, 4431, 4744, 4816, 5203, 5209, 5435, 5788, 5827, 5888, 6025, 6225, 6395, 6846, 6927, 7552, 7724, 7775, 8569, 8601, 9195, 11273, 11945, 12024, 12215, 12223, 12243, 12489, 12513, 12633, 12641, 12651, 12683, 12692, 12725, 12857, 12861, 12910, 12926, 12963, 12980, 12992, 13003, 13670, 13675, 13694, 13877, 13897, 14274, 14297, 14302, 14398, 14476, 14499, 14572, 14687, 14805, 14852, 15072, 15166, 15264, 15556, 15831, 15932, 15988, 16123, 16971, 17007, 17118, 17714, 18225, 18280, 18317, 18339, 18392, 18394, 18430, 18457, 18631, 18691, 18758, 18849, 18952, 19502, 19781, 19786, 19878, 21293, 21327])

(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, 4828, 4830, 4873, 5176, 5201, 5241, 5251, 5253, 5290, 5371, 6108, 6426, 6446, 6453, 6983, 7025, 7031, 7051, 7310, 7554, 7555, 7767, 8070, 8137, 8144, 8145, 8444, 8673, 8677, 8714, 9282, 9295, 9787, 9946, 9975, 10337, 10344, 10359, 10382, 10697, 10741, 10751, 10765, 11212, 11234, 11238, 11776, 11818, 11820, 11900, 12084, 12447, 12455, 12456, 12471, 12780, 12801, 12875, 13244, 13379, 13398, 13404, 13461, 13468, 13471, 13496, 13512, 13531, 13571, 13577, 13629, 13949, 14109, 14770, 14848, 14849, 14931, 14956, 14987, 15048, 15253, 15364, 15384, 15444, 15539, 15625, 15763, 16128, 16268, 16942, 16976, 16977, 17064, 17065, 17088, 17119, 17139, 17193, 17210, 17230, 17242, 17243, 17246, 17248, 17249, 17259, 17265, 17272, 17306, 17396, 17445, 17448, 17598, 17620, 17821, 17881, 17883, 17898, 17899, 17906, 17915, 18086, 18090, 18095, 18250, 18299, 18452, 19557, 20033, 20038, 20040, 20071, 20080, 20158, 20377, 20447, 20500, 20764, 20868, 20893, 20907, 20925, 21277, 21280, 21422, 21496, 21508, 21511])
```

2.2 validate query returns for three sample queries

- Source Code: validate_query_result.py (unit test)

  util.py



```
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
        """
```

The unit test is passed for all three sample queries.

Another validation method: compare the results with my learning partners to validate the query returns.

3. Subproject III: implement lossy dictionary compression

3.1 Lossy dictionary compression table and analysis

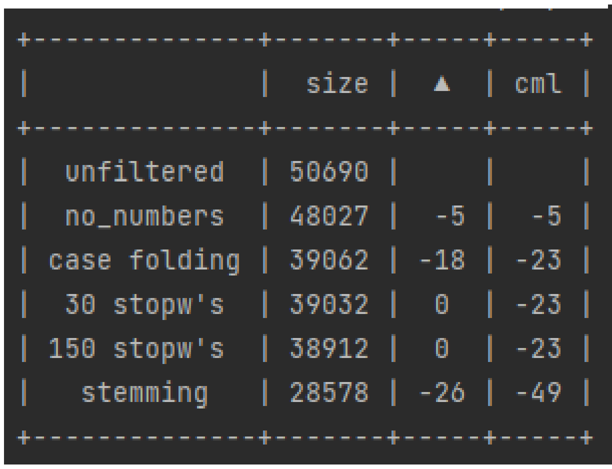
- Source Code: effect_compression.py

  util.py

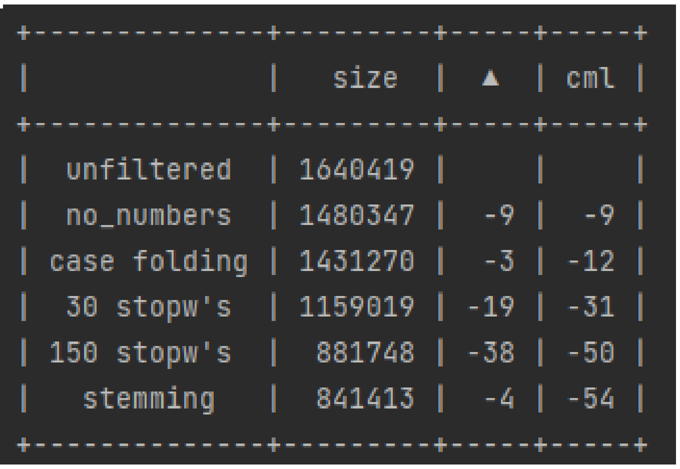
Uses methods in util.py to get the length of the terms, terms paired with document ids and terms connected with document ids and terms position in document after each preprocessing of lossy compression. Prints out the changed rate and accumulated changed rate accordingly.

Table1: Effect of preprocessing for Reuters-21578 Dictionary and non-positional index

Dictionary **Non-positional index**



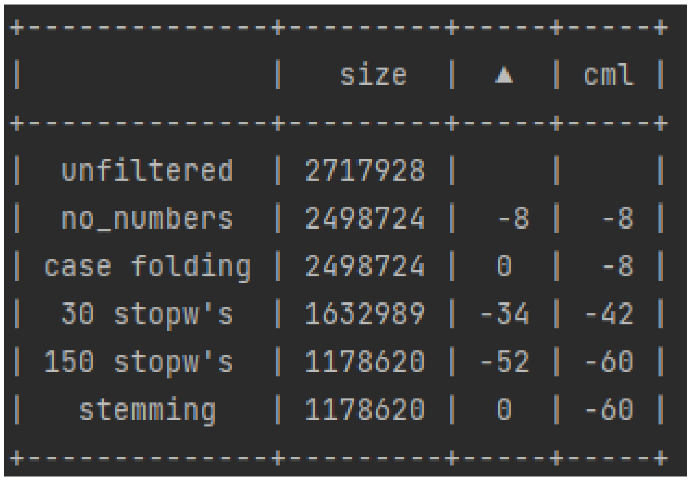
```
+-------------+-------+-----+-----+
|             |  size |  ▲  | cml |
+-------------+-------+-----+-----+
|  unfiltered | 50690 |     |     |
|  no_numbers | 48027 |  -5 |  -5 |
| case folding | 39062 | -18 | -23 |
|  30 stopw's | 39032 |  0  | -23 |
| 150 stopw's | 38912 |  0  | -23 |
|   stemming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



```
+-------------+---------+-----+-----+
|             |   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 |
+-------------+---------+-----+-----+
```

Table2: Effect of preprocessing for Reuters-21578 positional index

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 |
+-------------+---------+-----+-----+
```

From the table 1 and table 2, we could find remove stop words in dictionary would not bring much change at the size as the terms in dictionary is just showed once, so if we remove 30 or 150 stop words which are most common words but just record once in dictionary, it is just remove 30 or 150 terms out of dictionary which takes few changes for the size. But for non-positional index and positional index, remove the stop words will get a sharp shrank for the size. That is because, in the non-positional index, terms paired with document id and in the positional index, terms connected with the document id and their position in the document.

The similar reasons for the differences between numbers non-positional and positional, we could find case folding step will decrease the size 3% in non-positional index but can not bring any change in positional index, because in positional index each term related with document id and their position, lowercase the term could not bring any changes, but in non-positional index, if the terms in same documents could help to decrease the size. The same difference happens in stemming for the same reason.

3.2 Compares the retrieval results for three sample queries before and after the lossy compression

Three sample queries before compression:

a. “outlook”: frequency :183
b. “zone”: frequency:44
c. “week”: frequency:1796

Three sample queries after compression:

a. “outlook”: frequency :191
b. “zone”: frequency:82
c. “week”: frequency:564

Very interesting founding in the term week, the frequency is changed from 1796 to 564. The reason is it’s a stop word, and the 564 is the term like “weekly”, “weeks” stemming to “week”.

Given challenge queries results before compression:

a. “copper”: frequency :107
b. “Samjens”: frequency:4

c. “Carmark”: frequency:1

d. “Bundesbank”: frequency:166:

Given challenge queries results after compression:

a. “copper”: frequency :120

b. “Samjens”: frequency:5

c. “Carmark”: frequency:1

d. “Bundesbank”: frequency:167

Print out frequency and term posting list: some change comparing the queries results before compression, because of the compression.

```
(120, [22, 793, 800, 816, 922, 1148, 1184, 1552, 1607, 2006, 2074, 2186, 2764, 2782, 2880, 3454, 3613,
3862, 4058, 4291, 4373, 4431, 4744, 4816, 5203, 5209, 5435, 5788, 5827, 5888, 6025, 6225, 6395, 6846,
6927, 7552, 7724, 7775, 8569, 8601, 9195, 11053, 11273, 11945, 12024, 12215, 12223, 12243, 12484,
12489, 12513, 12633, 12641, 12651, 12683, 12692, 12725, 12857, 12861, 12910, 12926, 12963, 12980,
12992, 13003, 13465, 13495, 13608, 13665, 13670, 13675, 13687, 13694, 13757, 13877, 13897, 14136,
14274, 14297, 14302, 14398, 14476, 14499, 14572, 14687, 14805, 14852, 15072, 15166, 15264, 15556,
15831, 15932, 15988, 16110, 16123, 16971, 17007, 17118, 17714, 18225, 18280, 18317, 18339, 18392,
18394, 18430, 18457, 18631, 18691, 18758, 18849, 18952, 19502, 19781, 19786, 19878, 20096, 21293,
21327])

(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, 5201, 5241, 5251, 5253, 5290, 5371, 6108, 6426, 6446, 6453, 6983,
7025, 7031, 7051, 7310, 7554, 7555, 7767, 8070, 8137, 8144, 8145, 8444, 8673, 8677, 8714, 9282, 9295,
9787, 9946, 9975, 10337, 10344, 10359, 10382, 10697, 10741, 10751, 10765, 11212, 11234, 11238,
11776, 11818, 11820, 11900, 12084, 12447, 12455, 12456, 12471, 12780, 12801, 12875, 13244, 13379,
13398, 13404, 13461, 13468, 13471, 13496, 13512, 13531, 13571, 13577, 13629, 13949, 14109, 14770,
14848, 14849, 14931, 14956, 14987, 15048, 15253, 15364, 15384, 15444, 15522, 15539, 15625, 15763,
16128, 16268, 16942, 16976, 16977, 17064, 17065, 17088, 17119, 17139, 17193, 17210, 17230, 17242,
17243, 17246, 17248, 17249, 17259, 17265, 17272, 17306, 17396, 17445, 17448, 17598, 17620, 17821,
17881, 17883, 17898, 17899, 17906, 17915, 18086, 18090, 18095, 18250, 18299, 18452, 19557, 20033,
20038, 20040, 20071, 20080, 20158, 20377, 20447, 20500, 20764, 20868, 20893, 20907, 20925, 21277,
21280, 21422, 21496, 21508, 21511])
```