COMP 6791 Assignment 2 Report

Zexin Peng
Student number: 40166520

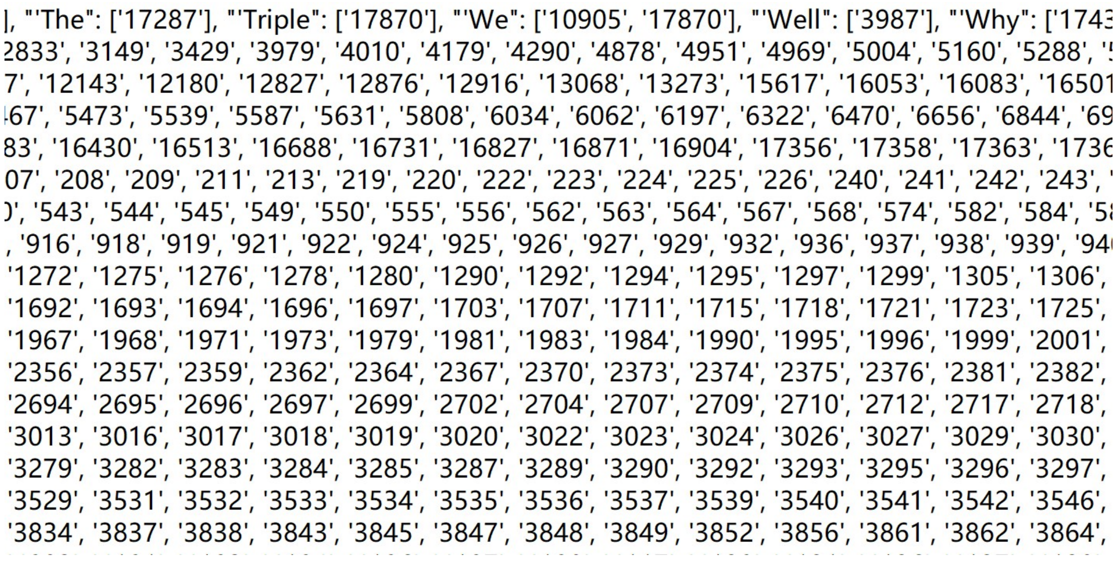
1. Description

My project consists of three python files representing three subprojects respectively. After the execution of the 'subject1.py', it will save the dictionary and posting list in a txt file. By the way, it should be executed at the first step, otherwise, the other two subprojects will not run correctly. If you run the 'subject2.py', you need to enter the term you want to query, and then you will get its posting list. For the 'subject3.py', you will get the statistical result after dictionary and posting list compression, and then you could enter the term then get its posting list.

2. Design

2.1 Subproject1

The first step is reading the "sgm" file one-by-one. After reading a file, I use regular expression to extract the document ID, title and content information from it. For every document, I concatenate the title and content. Then I tokenize to get all tokens in this document, and remove all punctuations in the token within this list "=\'+!()-[]{};:",<>./`?@#$%^&*_~ ". After the removal, I append it with its document ID into a list. We will get a list which has all term-documentID pairs in the whole collection. At the next step, we sort the list and remove all duplications in the list. Finally, for every term-document ID pair in the list, we put them into a dictionary and store the final dictionary into a file. Below is a section in the file. We could see that the posting list has been arranged in order too.



```
], "'The": ['17287'], "'Triple": ['17870'], "'We": ['10905', '17870'], "'Well": ['3987'], "'Why": ['1743
2833', '3149', '3429', '3979', '4010', '4179', '4290', '4878', '4951', '4969', '5004', '5160', '5288', '
7', '12143', '12180', '12827', '12876', '12916', '13068', '13273', '15617', '16053', '16083', '16501
67', '5473', '5539', '5587', '5631', '5808', '6034', '6062', '6197', '6322', '6470', '6656', '6844', '69
83', '16430', '16513', '16688', '16731', '16827', '16871', '16904', '17356', '17358', '17363', '1736
07', '208', '209', '211', '213', '219', '220', '222', '223', '224', '225', '226', '240', '241', '242', '243', '
)', '543', '544', '545', '549', '550', '555', '556', '562', '563', '564', '567', '568', '574', '582', '584', '5
, '916', '918', '919', '921', '922', '924', '925', '926', '927', '929', '932', '936', '937', '938', '939', '94
'1272', '1275', '1276', '1278', '1280', '1290', '1292', '1294', '1295', '1297', '1299', '1305', '1306',
'1692', '1693', '1694', '1696', '1697', '1703', '1707', '1711', '1715', '1718', '1721', '1723', '1725',
'1967', '1968', '1971', '1973', '1979', '1981', '1983', '1984', '1990', '1995', '1996', '1999', '2001',
'2356', '2357', '2359', '2362', '2364', '2367', '2370', '2373', '2374', '2375', '2376', '2381', '2382',
'2694', '2695', '2696', '2697', '2699', '2702', '2704', '2707', '2709', '2710', '2712', '2717', '2718',
'3013', '3016', '3017', '3018', '3019', '3020', '3022', '3023', '3024', '3026', '3027', '3029', '3030',
'3279', '3282', '3283', '3284', '3285', '3287', '3289', '3290', '3292', '3293', '3295', '3296', '3297',
'3529', '3531', '3532', '3533', '3534', '3535', '3536', '3537', '3539', '3540', '3541', '3542', '3546',
'3834', '3837', '3838', '3843', '3845', '3847', '3848', '3849', '3852', '3856', '3861', '3862', '3864',
```

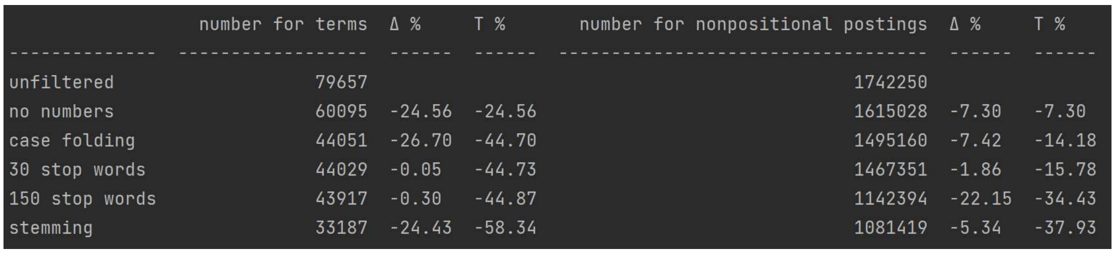
2.2 Subproject2

For subproject2, I just read the dictionary and posting list from the file obtained from subproject1. The program will search the term in the dictionary and return its posting list after the user enters the query term.

2.3 Subproject3

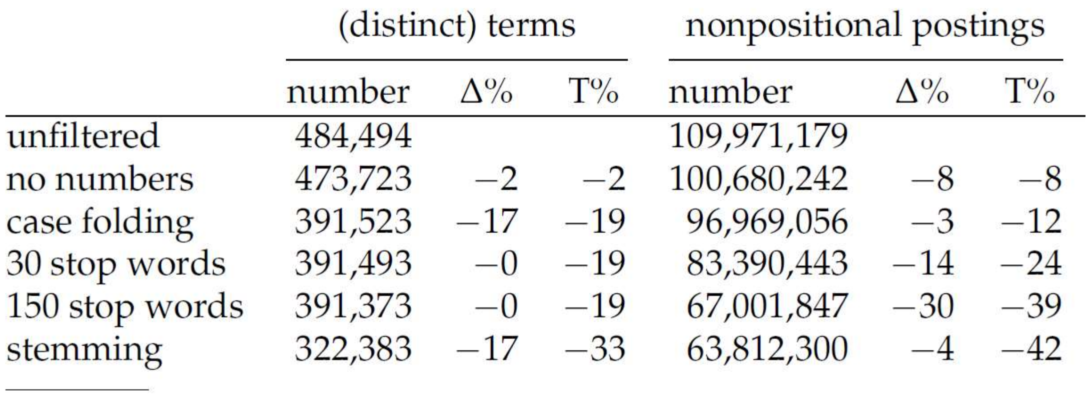
In my subproject3, after reading the dictionary and posting list into the memory, I do numbers removal, case folding, 30 stop words, 150 stop words removal and stemming, and analyze the statistical data of them and put them into the table one by one. For numbers removal and stop words removal, I delete all the numbers and stop words from the dictionary and its posting list. As for case folding and stemming, I iterate all terms in the dictionary and do case folding or stemming then merge its original posting list to its corresponding. The merge operation of posting list should keep the ascending order as well.
For stop lists, I use the stop word list in the nltk package.
At the end, you could enter the query term, the program will do the stemming first then search it in the dictionary and show its posting list.

3. Table result:



```
                  number for terms  Δ %     T %       number for nonpositional postings  Δ %     T %
-------------  -----------------  ------  ------  ---------------------------------  ------  ------
unfiltered                  79657                                            1742250
no numbers                  60095  -24.56  -24.56                            1615028  -7.30   -7.30
case folding                44051  -26.70  -44.70                            1495160  -7.42   -14.18
30 stop words               44029  -0.05   -44.73                            1467351  -1.86   -15.78
150 stop words              43917  -0.30   -44.87                            1142394  -22.15  -34.43
stemming                    33187  -24.43  -58.34                            1081419  -5.34   -37.93
```

Table from my program

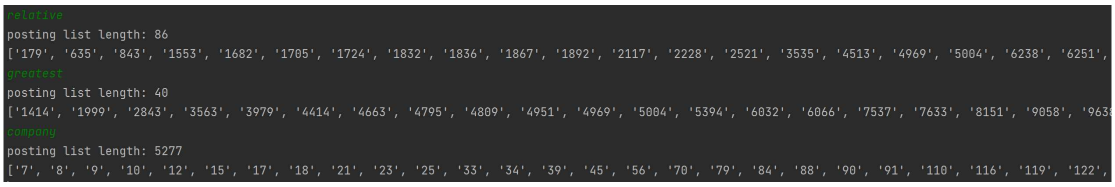


	(distinct) terms			nonpositional postings		
	number	Δ%	T%	number	Δ%	T%
unfiltered	484,494			109,971,179		
no numbers	473,723	−2	−2	100,680,242	−8	−8
case folding	391,523	−17	−19	96,969,056	−3	−12
30 stop words	391,493	−0	−19	83,390,443	−14	−24
150 stop words	391,373	−0	−19	67,001,847	−30	−39
stemming	322,383	−17	−33	63,812,300	−4	−42

Table in the textbook

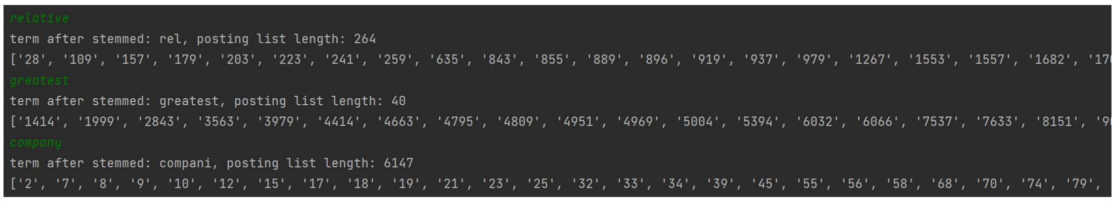
Compared to the table from the textbook, the effect of number removal in my program is much greater than that in textbook, and other two dictionary compression methods case folding and stemming in my program also have a little better performance than that in the textbook. The total size of the dictionary has 57% reduction after all compression methods in my program compared with 33% in the textbook.
As for the statical data of positional postings, I have roughly the same result with the textbook but a slight difference in 30 stop words, because the first 30 stop words of the stop words list only have some words that does not occur so frequent in the collection. Below is the first 30 stop words in the stop word list.
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've", "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his', 'himself', 'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its']

4. Query comparison:



```
relative
posting list length: 86
['179', '635', '843', '1553', '1682', '1705', '1724', '1832', '1836', '1867', '1892', '2117', '2228', '2521', '3535', '4513', '4969', '5004', '6238', '6251',
greatest
posting list length: 40
['1414', '1999', '2843', '3563', '3979', '4414', '4663', '4795', '4809', '4951', '4969', '5004', '5394', '6032', '6066', '7537', '7633', '8151', '9058', '963
company
posting list length: 5277
['7', '8', '9', '10', '12', '15', '17', '18', '21', '23', '25', '33', '34', '39', '45', '56', '70', '79', '84', '88', '90', '91', '110', '116', '119', '122',
```

Query in subproject2



```
relative
term after stemmed: rel, posting list length: 264
['28', '109', '157', '179', '203', '223', '241', '259', '635', '843', '855', '889', '896', '919', '937', '979', '1267', '1553', '1557', '1682', '17
greatest
term after stemmed: greatest, posting list length: 40
['1414', '1999', '2843', '3563', '3979', '4414', '4663', '4795', '4809', '4951', '4969', '5004', '5394', '6032', '6066', '7537', '7633', '8151', '9
company
term after stemmed: compani, posting list length: 6147
['2', '7', '8', '9', '10', '12', '15', '17', '18', '19', '21', '23', '25', '32', '33', '34', '39', '45', '55', '56', '58', '68', '70', '74', '79',
```

Query in subproject3

As we could see, the word ‘greatest’ has no change after stemmed, and no words after stemmed could be ‘greatest’, so the posting list we get has no change as well. The other two words get longer posting list, because after compression, we merge some terms and their posting lists because they are the same after stemmed, so the length of the posting list increases.

5. What I have learnt:

I have practiced the method of document tokenization, and I have also learnt the process of constructing the dictionary and its posting list. By the compression, my heuristics is that dictionary compression could reduce the size of dictionary and posting list dramatically, because there is much useless information in the collections like numbers, stop words and punctuations, so it could improve the efficiency of searching terms in the dictionary. The result term list will be slightly longer than that without compression, because some posting lists have been merged. The efficiency has also improved because the merged lists also have many document ID in common.