COMP472 – Project 2 DEMO

Rhina Kim – 40130779

October 8, 2022

This program is composed of 3 files:

- ***P2_utils.py*** which includes following functionalities:
  - Extract, tokenize, form inverted index from the raw texts
  - All reduction functions (remove numbers and stop words, make lowercase, apply porter stemmer) from Subproject 3
  - Result output function (outputting to console and to files)
- ***P2_compile.py***: all the pipeline steps from Subproject 1 to Subproject 3. All the inputs are given in this file. This file imports *P2_utils.py* to perform necessary functions to update or modify inverted index.

Constants:

```
DIRECTORY = "../reuters21578_extracted/"
OUTPUT_DIRECTORY = "outputs_test/"
MAX_SGM_FILES = 5  # [DEMO] for testing purpose
SAMPLE_QUERIES = ["copper", "Samjens", "Carmark", "Bundesbank"]
```

- *DIRECTORY*: input directory to be read which stores all the *Reuters* corpus
- *OUTPUT_DIRECTORY*: all pipeline outputs are written inside this folder
- *MAX_SGM_FILES*: maximum number of *.sgm* files to be used in this research (just to reduce computation time and make compact result). Only 5 *.sgm* files were used for report section, however, every *.sgm* files are used in this demo section since the program is capable of doing so.
- *SAMPLE_QUERIES*: list of sample queries to be used for single query search which is originally provided by our professor.

Inputs: (*P2_compile.py*, Line 389)

```
### START ###
sgm_files = filter_corpus(DIRECTORY, MAX_SGM_FILES)
# Subproject 1
print("\n===================== Subproject 1 =====================\n")
index = S1_naive_indexer(sgm_files)
# Subproject 2
print("\n\n\n===================== Subproject 2 =====================\n")
S2_naive_query_search(index)
# Subproject 3
print("\n\n\n===================== Subproject 3 =====================\n")
index = S3_compressed_indexer(sgm_files)
S3_compressed_query_search(index)
```

Program runs the input in timely manner (Subproject 1 → Subproject 2 → Subproject 3). Query search mechanism in Subproject 2 is brought again in Subproject 3.

Outputs:

For much simplified looking outputs, refer to ***Deliverables/P2/outputs_test***

```
==================== Pre-processing ====================
[P1] Extracting, Tokenizing, and creating term-documentID pairs from Reuter's collection:
 * Reading ../reuters21578_extracted/reut2-004.sgm *
  - 1000 documents have been successfully extracted.
  - Number of postings: 129102
 * Reading ../reuters21578_extracted/reut2-010.sgm *
  - 1000 documents have been successfully extracted.
  - Number of postings: 134278
 * Reading ../reuters21578_extracted/reut2-011.sgm *
  - 1000 documents have been successfully extracted.
  - Number of postings: 125183
 * Reading ../reuters21578_extracted/reut2-005.sgm *
  - 1000 documents have been successfully extracted.
  - Number of postings: 139044
```

… continues …

```
==================== Subproject 1 ====================
[P2_1-1]: Create term-documentID pairs
--> Total number of postings in F: 2597951

[P2_1-2]: Sort list F and remove duplicates
  - 986711 postings have been removed due to duplicate term-postings pair.
--> Total number of postings in F: 1611240

[P2_1-3]: Complete inverted index via dictionary and pointer of postings
--> Total number of distinct terms(type) in dictionary: 58405
--> Total number of postings in F: 1611240
```

```
==================== Subproject 2 ====================
[P2_2-1]: Sample single term query search using naive index

All queries are NOT lowercased:
* Searching for query "copper" *
  Documents mentioning "copper" exists based on the index.
  - Document position: 12683 7552 12489 13694 18952 12857 18317 12992 13675 6846 6025 18691
5888 2074 6927 800 18225 2782 2880 14274 18392 12513 15166 7775 12223 5203 14297 3613 18631
1552 4744 21293 19781 3454 12926 17007 19502 15932 17714 6225 13897 13670 14302 12243 3862
14499 14476 12861 6395 17118 14572 12024 12215 15831 12692 2764 13003 18280 5209 8601 12910
16123 5435 11945 13877 2006 1184 14805 8569 12963 21327 15556 4431 14852 1607 19878 14398
4816 15988 18394 11273 5827 12633 2186 12725 7724 4291 793 18430 5788 12641 14687 18339 22
18457 15264 816 1148 12980 18849 12651 18758 16971 19786
```

- Term frequency: 104
*** Searching for query "Samjens" ***
Documents mentioning "Samjens" exists based on the index.
- Document position: 17863 19419 17837 18071
- Term frequency: 4
*** Searching for query "Carmark"** *
Documents mentioning "Carmark" exists based on the index.
- Document position: 19758
- Term frequency: 1
*** Searching for query "Bundesbank" ***
Documents mentioning "Bundesbank" exists based on the index.
- Document position: 15364 14956 4828 12084 8673 15625 20868 8145 14848 17306 17620 14849 12471 7767 16942 13629 13461 7310 20500 15763 17821 21511 13404 9282 21280 14770 8144 17230 9975 21496 10751 10382 17243 13577 16977 17906 13949 17119 21277 13398 17899 10697 12875 17065 16268 17881 8714 17915 17396 19557 10741 926 8070 17139 18250 2197 5241 12456 2110 20040 20080 20447 6983 4873 950 10337 942 11234 13471 7025 16976 1959 17246 17448 5176 18090 15444 20033 5201 4297 16128 20893 2979 14109 2686 7031 1969 17898 17242 10344 12447 11820 10359 20925 13571 17272 2662 12801 6108 1540 12455 6453 13379 13244 4062 14931 21508 3020 12780 20038 9787 20071 13531 8677 10765 11212 11900 20158 17445 17265 13468 17883 11776 15539 15048 9295 14987 6426 18452 17259 7051 21422 20377 18095 15253 17598 18299 18086 8137 20907 7554 5253 17248 1971 5290 17064 13512 17249 8444 20764 11818 4830 11238 17193 3514 5371 7555 9946 5251 15384 4113 6446 13496 278 17210 17088
- Term frequency: 166

All queries are lowercased:
*** Searching for query "copper" ***
Documents mentioning "copper" exists based on the index.
- Document position: 12683 7552 12489 13694 18952 12857 18317 12992 13675 6846 6025 18691 5888 2074 6927 800 18225 2782 2880 14274 18392 12513 15166 7775 12223 5203 14297 3613 18631 1552 4744 21293 19781 3454 12926 17007 19502 15932 17714 6225 13897 13670 14302 12243 3862 14499 14476 12861 6395 17118 14572 12024 12215 15831 12692 2764 13003 18280 5209 8601 12910 16123 5435 11945 13877 2006 1184 14805 8569 12963 21327 15556 4431 14852 1607 19878 14398 4816 15988 18394 11273 5827 12633 2186 12725 7724 4291 793 18430 5788 12641 14687 18339 22 18457 15264 816 1148 12980 18849 12651 18758 16971 19786
- Term frequency: 104
*** Searching for query "samjens" ***
No document mentioning "samjens" has found based on the index.
*** Searching for query "carmark" ***
No document mentioning "carmark" has found based on the index.
*** Searching for query "bundesbank" ***
No document mentioning "bundesbank" has found based on the index.

========== Subproject 3 ==========
[P2_3-1]: Implement Lossy Dictionary Compression techniques

(1) Sort list F
--> Total number of postings in F: 2597951

(2) Remove duplicates from F

- 986711 postings have been removed due to duplicate term-postings pair.

--> Total number of postings in F: 1611240

(3) Complete inverted index via dictionary and pointer of postings

(DEFAULT)

--> Total number of postings in F: 1611240

(NO NUMBERS)

- 52282 postings have been removed due to numeric value.

--> Total number of postings in F: 1558958

(LOWERCASE)

--> Total number of postings in F: 1558958

(30 STOPWORDS)

- 53403 postings have been removed due to stop words.

--> Total number of postings in F: 1505555

(150 STOPWORDS)

- 265054 postings have been removed due to stop words.

--> Total number of postings in F: 1240501

(PORTER STEMMER)

--> Total number of postings in F: 1240501

(4) Statistics:

	(Distinct) Terms	Δ% Terms	T% Terms	Nonpositional Postings	Δ% Postings	T% Postings
unfiltered	28752	--	--	377489	--	--
no numbers	27846	3.15	3.15	365432	3.19	3.19
case folding	20587	26.07	28.4	365432	0.0	3.19
30 stop words (*)	20566	0.1	28.47	348451	4.65	7.69
150 stop words (*)	20475	0.54	28.79	283435	22.44	24.92
stemming	14688	28.26	48.91	283435	0.0	24.92

[References]:
- (Distinct) Terms: Number of distinct terms in dictionary
- Nonpositional Postings: Number of nonpositional postings (total postings)
- Δ%: reduction in size from the previous
- T%: culmulative (total) reduction from unfiltered

(*) 30 stoop words and 150 stop words both use case folding as their reference line$›

[P2_3-2]: Sample single term query search using compressed index

All queries are NOT lowercased:

*** Searching for query "copper" ***

Documents mentioning "copper" exists based on the index.

- Document position: 19786 7552 13694 18952 13465 13675 6846 6025 6927 800 14274 12223 14297 3613 16110 19781 17007 19502 13897 14302 20096 12489 14476 12861 12024 15831 2764 18280 12910 16123 5435 11945 1184 14805 21327 4431 13670 14398 13687 14499 5827 12633 2186 4291 793 13608 14136 18430 5788 12215 12692 18339 22 15264 816 1148 12980 18849 12651 14499 5827 4291 12641 12215 12484 22 16123 13665 13757 18849 12651 4373 2186 6846 5888 14687 922 12223 12980 3613 19781 11053 7552 13694 18952 6025 14476 15831 5203 18280 13495 2006 16110 21327 12683 7552 12489 13694 18952 12857 18317 12992 13675 6846 6025 18691 5888 2074 6927 800 18225 2782 2880 14274 18392 12513 15166 7775 12223 5203 14297 3613 18631 1552 4744 21293

19781 3454 12926 17007 19502 15932 17714 6225 13897 13670 14302 12243 3862 14499 14476 12861 6395 17118 14572 12024 12215 15831 12692 2764 13003 18280 5209 8601 12910 16123 5435 11945 13877 2006 1184 14805 8569 12963 21327 15556 4431 14852 1607 19878 14398 4816 15988 18394 11273 5827 12633 2186 12725 7724 4291 793 18430 5788 12641 14687 18339 22 18457 15264 816 1148 12980 18849 12651 18758 16971 19786
- Term frequency: 198

*** Searching for query "Samjens" ***

No document mentioning "Samjens" has found based on the index.

*** Searching for query "Carmark" ***

No document mentioning "Carmark" has found based on the index.

*** Searching for query "Bundesbank" ***

No document mentioning "Bundesbank" has found based on the index.

All queries are lowercased:

*** Searching for query "copper" ***

Documents mentioning "copper" exists based on the index.
- Document position: 19786 7552 13694 18952 13465 13675 6846 6025 6927 800 14274 12223 14297 3613 16110 19781 17007 19502 13897 14302 20096 12489 14476 12861 12024 15831 2764 18280 12910 16123 5435 11945 1184 14805 21327 4431 13670 14398 13687 14499 5827 12633 2186 4291 793 13608 14136 18430 5788 12215 12692 18339 22 15264 816 1148 12980 18849 12651 14499 5827 4291 12641 12215 12484 22 16123 13665 13757 18849 12651 4373 2186 6846 5888 14687 922 12223 12980 3613 19781 11053 7552 13694 18952 6025 14476 15831 5203 18280 13495 2006 16110 21327 12683 7552 12489 13694 18952 12857 18317 12992 13675 6846 6025 18691 5888 2074 6927 800 18225 2782 2880 14274 18392 12513 15166 7775 12223 5203 14297 3613 18631 1552 4744 21293 19781 3454 12926 17007 19502 15932 17714 6225 13897 13670 14302 12243 3862 14499 14476 12861 6395 17118 14572 12024 12215 15831 12692 2764 13003 18280 5209 8601 12910 16123 5435 11945 13877 2006 1184 14805 8569 12963 21327 15556 4431 14852 1607 19878 14398 4816 15988 18394 11273 5827 12633 2186 12725 7724 4291 793 18430 5788 12641 14687 18339 22 18457 15264 816 1148 12980 18849 12651 18758 16971 19786
- Term frequency: 198

*** Searching for query "samjens" ***

No document mentioning "samjens" has found based on the index.

*** Searching for query "carmark" ***

Documents mentioning "carmark" exists based on the index.
- Document position: 19758
- Term frequency: 1

*** Searching for query "bundesbank" ***

Documents mentioning "bundesbank" exists based on the index.
- Document position: 17448 18090 16977 7031 17242 13949 17899 10359 6108 1540 17915 5241 12456 942 11900 17883 15539 18452 18086 17064 17249 11238 17272 17193 6446 17088 8677 11212 14849 12471 15522 9295 6426 16942 13404 21280 1971 17243 12875 4113 17210 15364 926 8145 15364 14956 4828 12084 8673 15625 20868 8145 14848 17306 17620 14849 12471 7767 16942 13629 13461 7310 20500 15763 17821 21511 13404 9282 21280 14770 8144 17230 9975 21496 10751 10382 17243 13577 16977 17906 13949 17119 21277 13398 17899 10697 12875 17065 16268 17881 8714 17915 17396 19557 10741 926 8070 17139 18250 2197 5241 12456 2110 20040 20080 20447 6983 4873 950 10337 942 11234 13471 7025 16976 1959 17246 17448 5176 18090 15444 20033 5201 4297 16128 20893 2979 14109 2686 7031 1969 17898 17242 10344 12447 11820 10359 20925 13571 17272 2662 12801 6108 1540 12455 6453 13379 13244 4062 14931 21508 3020 12780 20038 9787 20071

13531 8677 10765 11212 11900 20158 17445 17265 13468 17883 11776 15539 15048 9295 14987 6426 18452 17259 7051 21422 20377 18095 15253 17598 18299 18086 8137 20907 7554 5253 17248 1971 5290 17064 13512 17249 8444 20764 11818 4830 11238 17193 3514 5371 7555 9946 5251 15384 4113 6446 13496 278 17210 17088

- Term frequency: 210

Explanation:

All the outputs correspond to each subproject parts are specified as such: [***P2_subproject#-pipeline#].***

Pre-processing

The preprocessing section outputs which *.sgm* files from the Reuter's collection have been extracted and tokenized to be used in this project. There are 22 *.sgm* files with each file contains 1000 documents / articles except last file contains 578 documents / articles, which adds up to the **total of 2597951 postings** available.

Subproject 1

Subproject 1 (section **[P2_1-1], [P2_1-2], [P3_1-3]** in the output prompt) outputs the pipeline step of

1. Creating *term-documentID* pairs
2. Sorting list "*F*" and removing duplicates
3. Generate inverted index (naïve indexer)

The output also specifies the total number of postings (non-positional postings) after every pipeline step. For example, in the above outputs, total number of postings in postings list *F* after tokenization process was 2597951. However, the number of postings decreases to 1611240 after the process of duplicate term-postings pair removal, removing total of 986711 postings. This emphasizes the change in size of postings after each operation/processing.

Total number of distinct terms are also specified after generating inverted index (naïve indexer) in the third step of Subproject 1. In this case, it turned out to be 58405 distinct terms from all 22 *.sgm* files. This naïve indexer created in Subproject 1 is passed onto Subproject 2.

Subproject 2

Subproject 2 (section **[P2_2-1]** in the output prompt) outputs the result of single query search. The sample queries used for this Subproject is: [*copper, Samjens, Carmark, Bundesbank*]. According to the output, all sample queries (without case folding) have returned that they exist in the given naïve indexer.

To verify this result **[P2_2-2]**, we will take the query "*Samjens*" as an example:

reut2-017.sgm:

1. <REUTERS … NEWID="**17837**">
   …
   Edelman/Dominion group, known as **Samjens** Acqusition Corp, said

</REUTERS>

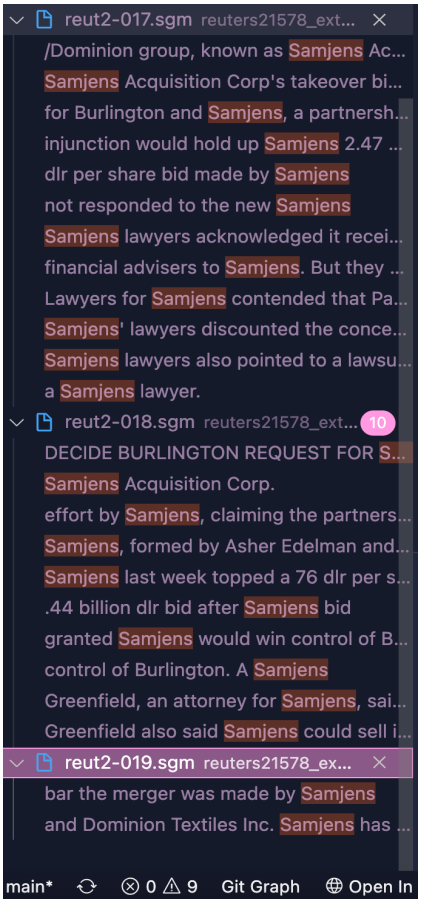
2. <REUTERS … NEWID="**17863**">
   …
   Samjens Acquisition Corp's takeover bid for the company.
   </REUTERS>

reut2-018.sgm:

3. <REUTERS … NEWID="**18071**">
   …
   grant Burlington Industries Inc's request to stop a takeover by **Samjens** Acquisition Corp.
   </REUTERS>

reut2-019.sgm:

4. <REUTERS … NEWID="**19419**">
   …
   The request to bar the merger was made by **Samjens**
   </REUTERS>



Which matches with the above output result of all document positions of "*Sasmjens*", and 4 postings found that corresponds to the term.

Subproject 3

The first part of Subproject 3 (section **[P2_3-1]** in the output prompt) outputs the pipeline step of

1. Creating *term-documentID* pairs, sorting list "*F*", and removing duplicates
2. Generate 6 kinds (6 reduction steps) of inverted index (lossy compressed indexer):
   - Default indexer
   - Indexer without numeric values
   - Indexer with all lowercase
   - Indexer after removing 30 stop words
   - Indexer after removing 150 stop words
   - Indexer after applying porter stemmer
3. Creates statistical properties of terms in table format for every reduction steps

The output also specifies the total number of postings (non-positional postings) after every reduction step. For example, in the above outputs, initial total number of postings in postings list *F* without any reduction process was 2597951. Removing duplicates led to the total number of postings = 1611240.

It is important to remind that, in this Subproject 3, we set the inverted index with 1611240 postings after going through the duplicate term-postings' removal process as new "default" / "unfiltered" index. This is because we want to avoid this process to be used for the statistical analysis of terms, since this process only removes the number of duplicated term-postings pairs yet does not affect the number of "distinct" terms in the inverted index.

Total number of distinct terms, non-positional postings, Δ% and %T of terms and postings are all specified in the statistics table above.
Table 5.1 from textbook:

Introduction to Information Retrieval. Christopher D. Manning, Prabhakar Raghavan and Hinrich Schütze. Cambridge University Press, 2008.

		(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

Statistical Table from this Project 2:

	(Distinct) Terms	Δ% Terms	T% Terms	Nonpositional Postings	Δ% Postings	T% Postings
unfiltered	28752	--	--	377489	--	--
no numbers	27846	3.15	3.15	365432	3.19	3.19
case folding	20587	26.07	28.4	365432	0.0	3.19
30 stop words (*)	20566	0.1	28.47	348451	4.65	7.69
150 stop words (*)	20475	0.54	28.79	283435	22.44	24.92
stemming	14688	28.26	48.91	283435	0.0	24.92

According to the statistical tables from the result output, the reduction size value Δ% and cumulative reduction size value %T for all reduction process both resulted in having similar values compared with the statistical table from the textbook.

Case folding reduction process reduced the number of distinct terms by 26% which is significant reduction. Case folding reduction process in textbook table also displays the significant decrease in the number of distinct terms. However, both non-positional postings for case folding reduction process are not as affecting as those for terms reduction (0% in the Project, 3% in the textbook). The result implies that a lot of uppercase terms were combined with lowercase terms.

Stemming also greatly reduces the number of distinct terms for both tables (28% in the project, 17% in the textbook table) with little effect on number of postings since it just combines the stemmed terms and does not affect a lot on the reduction of non-positional postings.

Unlike case folding and stemming reduction processes, removing stop words does not significantly influence the size of compressed indexer (a.k.a number of distinct terms) yet greatly affects the change in number of non-positional postings. Both tables represent that the more stop words to be removed, the greater the reduction in number of postings. For example, reduction process with 30 stop words resulted in 4.65% in the experimented project table and 14% in the textbook table, whereas same process with 150 stop words resulted in 22.4% in the project table and 30% in the textbook table. This is because only 150 words are removed from massive indexer, yet each word may carry enormous number of postings (especially since the stop words are mostly common words in the article such as “to”, “be”, etc. we can expect that the number of postings for such words can be immense).

Overall, the table given by the textbook and the table resulted from this experiment is similar in terms of the relative percentage in reduction size (Δ%) for every kind of reduction process.

The second part of Subproject 3 (section **[P2_3-2]** in the output prompt) outputs the query search result using compressed indexer. Same sample queries list is used for this experiment: [*copper, Samjens, Carmark, Bundesbank*]. The outputs have two parts: (1) result of query search with default sample queries, and (2) result of query search with lowercased sample queries. Since the query search is performed on compressed indexer where all distinct terms are lowercased, (2) method displays more accurate output in this case, thus we will focus on queries used in method (2).

Brief version of queries search result from the output above:

Search queries with Naïve indexer:

- "*copper*": 104 postings found
- "*Samjens*": 4 postings found
- "*Carmark*": 1 posting found
- "*Bundesbank*": 166 postings found

Search queries with Compressed Indexer:

- "*copper*": 198 postings found
- "*Samjens*": 0 postings found
- "*Carmark*": 1 posting found
- "*Bundesbank*": 210 postings found

For the query "*Samjens*", the postings have reduced to 0 with search queries with compressed indexer. This is because during the stemming process, the term "*Samjens*" turned into "*Samjen*" where all initial 4 postings correspond to the word is now a postings value of "*Samjen*" instead of "*Samjens*". If the output search query of "*Samjens*" to be still desired, this can be improved by proximity search.

Any other queries like "*copper*" or "*Bundesbank*" resulted in significant increase increased number of postings found with compressed indexer. It is observed that the term "*copper*" might have existed with the term "*Copper*" which cannot be combined yet in the naïve indexer, so can "*Bundesbank*". Throughout the compressed indexer, the uppercase and lowercase version of these terms are combined as one distinct term / dictionary as well as its postings list which results in increasing the number of postings for that term.