CMPE493 – Introduction to Information Retrieval
Assignment 1 Report
A Simple Document Retrieval System for Boolean and Wildcard Queries

CEMAL BURAK AYGÜN (2014400072) **SPRING 2019**

pre-process.py

For pre-processing the data, I have mainly utilized Python **dictionary** objects and **Regex**. I created a dictionary object named **INDEX_DICT** to represent the (actual) **term-document inverted index**. I also created another dictionary object named **TEMP_INDEX_DICT**. This is a temporary (helper) term-document inverted index used to build the actual one. These dictionary objects map a string to a list of integers (map terms to a list of document ids). Also, I created several integer variables to store some statistics about the quantity of the tokens/terms of the corpus before and after some processing is applied.

pre-process.py traverses all the data files (e.g., reut2-001.sgm) in directory reuters21578 one by one. For each file, it extracts all the **(NEWID, <TEXT>...</TEXT>)** pairs stored in *<REUTERS>...</REUTERS>* blocks using a Regex pattern. For each such pair, it extracts **title** and **body** fields of *<TEXT>* block using Regex patterns and concatenates them in a string **s**. Then, **s** is tokenized as follows: First, all the punctuation characters (stored in **punctuations.txt** file) are replaced by a space character and then the result is split from space.

After getting the tokens of *<TEXT>* block, number of tokens is added to an integer variable (**TOKENS_BEFORE_STOPWORDS**) to calculate *the total number of tokens in the corpus **before** the stopwords are removed*. Also, the number of tokens that are stopwords (words stated in **stopwords.txt** file, case-sensitive) is added to another integer variable (total_stopwords) to calculate *the total number of tokens in the corpus **after** the stopwords are removed* (**TOKENS_AFTER_STOPWORDS** = **TOKENS_BEFORE_STOPWORDS** – total_stopwords). Finally, the tokens are inserted into **TEMP_INDEX_DICT** with **NEWID** being the document id.

Since tokens are inserted into **TEMP_INDEX_DICT** without any processing (like filtering and case-folding) and the keys in a dictionary object is unique (case-sensitive), the size (the number of keys) of **TEMP_INDEX_DICT** becomes equal to *the total number of terms in the corpus **before** the stopwords are removed and case-folding is applied* when all the files are traversed and tokenized.

After **TEMP_INDEX_DICT** is fully constructed, **pre-process.py** continues to create **INDEX_DICT**. The elements of **TEMP_INDEX_DICT** are traversed applying case-folding (lowercasing) and the ones that are not stopwords are copied (or merged) into **INDEX_DICT**. Effectively, the terms in **TEMP_INDEX_DICT** are filtered from stopwords and case-folding is applied and the result is **INDEX_DICT**. So, the size (the number of keys) of **INDEX_DICT** becomes equal to *the total number of terms in the corpus **after** the stopwords are removed and case-folding is applied.*

After **INDEXT_DICT** is constructed, **pre-process.py** dumps it in a **JSON** file named **index.json**. Then, another dictionary object is created which is the **bigram-term inverted index** from the terms (keys) stored in **INDEX_DICT** and that one is dumped into a JSON file named **bigrams.json**.

CMPE493 – Introduction to Information Retrieval
Assignment 1 Report
A Simple Document Retrieval System for Boolean and Wildcard Queries

CEMAL BURAK AYGÜN (2014400072) **SPRING 2019**

Top 20 most frequent terms are calculated as follows: A dictionary object (freq_dict) is created which maps integers to a list of strings (maps frequencies to a list of terms). For example, if a term-document inverted index is { 'a': [1, 3, 5], 'b': [2, 4], 'c': [2, 5, 10] }, then freq_dict will be { 3: ['a', 'c'], 2: ['b'] }. Then, freq_dict is sorted according to its keys (frequency) in decreasing order and the top 20 terms are selected. One freq_dict is created for **TEMP_INDEX_DICT** to get t*op 20 most frequent terms **before** the stopword removal and case-folding* and one is created for **INDEX_DICT** to get t*op 20 most frequent terms **after** the stopword removal and case-folding*.

Overall, the statistics are as follows:

Number of tokens the corpus contains before stopword removal	:	2,902,784
Number of tokens the corpus contains after stopword removal	:	2,286,351
Number of terms (unique tokens) before stopword removal and case-folding	:	61,462
Number of terms (unique tokens) after stopword removal and case-folding	:	45,346

Top 20 most frequent terms before stopword removal and case-folding: <term> (<frequency>)

3 (19162) | of (15480) | said (15344) | the (15052) | Reuter (14225) | and (14188) | to (14112) | a (13346) | in (12828) | lt (11332) | The (11242) | for (11031) | it (9494) | s (8560) | mln (8340) | on (8316) | its (7906) | is (7833) | dlrs (7588) | from (7584)

Top 20 most frequent terms after stopword removal and case-folding: <term> (<frequency>)

3 (19162) | reuter (18885) | said (15435) | to (14845) | lt (11339) | s (10684) | mln (8587) | dlrs (8036) | from (7762) | at (7488) | 1 (7133) | year (6455) | pct (6264) | has (6143) | inc (5998) | company (5507) | 2 (5411) | corp (5350) | u (4770) | 000 (4762)

CMPE493 – Introduction to Information Retrieval
Assignment 1 Report
A Simple Document Retrieval System for Boolean and Wildcard Queries

CEMAL BURAK AYGÜN (2014400072) **SPRING 2019**

```
cba@cba-desktop:/DOSYA/BOUN/18-19-2/CMPE493/HOMEWORK/HW1/workspace$ python3 Code/pre-process.py

[INFO]  Creating term-document inverted index...
        /DOSYA/BOUN/18-19-2/CMPE493/HOMEWORK/HW1/workspace/reuters21578/reut2-000.sgm was processed.
        /DOSYA/BOUN/18-19-2/CMPE493/HOMEWORK/HW1/workspace/reuters21578/reut2-001.sgm was processed.
        /DOSYA/BOUN/18-19-2/CMPE493/HOMEWORK/HW1/workspace/reuters21578/reut2-002.sgm was processed.
        /DOSYA/BOUN/18-19-2/CMPE493/HOMEWORK/HW1/workspace/reuters21578/reut2-003.sgm was processed.
        /DOSYA/BOUN/18-19-2/CMPE493/HOMEWORK/HW1/workspace/reuters21578/reut2-004.sgm was processed.
        /DOSYA/BOUN/18-19-2/CMPE493/HOMEWORK/HW1/workspace/reuters21578/reut2-005.sgm was processed.
        /DOSYA/BOUN/18-19-2/CMPE493/HOMEWORK/HW1/workspace/reuters21578/reut2-006.sgm was processed.
        /DOSYA/BOUN/18-19-2/CMPE493/HOMEWORK/HW1/workspace/reuters21578/reut2-007.sgm was processed.
        /DOSYA/BOUN/18-19-2/CMPE493/HOMEWORK/HW1/workspace/reuters21578/reut2-008.sgm was processed.
        /DOSYA/BOUN/18-19-2/CMPE493/HOMEWORK/HW1/workspace/reuters21578/reut2-009.sgm was processed.
        /DOSYA/BOUN/18-19-2/CMPE493/HOMEWORK/HW1/workspace/reuters21578/reut2-010.sgm was processed.
        /DOSYA/BOUN/18-19-2/CMPE493/HOMEWORK/HW1/workspace/reuters21578/reut2-011.sgm was processed.
        /DOSYA/BOUN/18-19-2/CMPE493/HOMEWORK/HW1/workspace/reuters21578/reut2-012.sgm was processed.
        /DOSYA/BOUN/18-19-2/CMPE493/HOMEWORK/HW1/workspace/reuters21578/reut2-013.sgm was processed.
        /DOSYA/BOUN/18-19-2/CMPE493/HOMEWORK/HW1/workspace/reuters21578/reut2-014.sgm was processed.
        /DOSYA/BOUN/18-19-2/CMPE493/HOMEWORK/HW1/workspace/reuters21578/reut2-015.sgm was processed.
        /DOSYA/BOUN/18-19-2/CMPE493/HOMEWORK/HW1/workspace/reuters21578/reut2-016.sgm was processed.
        /DOSYA/BOUN/18-19-2/CMPE493/HOMEWORK/HW1/workspace/reuters21578/reut2-017.sgm was processed.
        /DOSYA/BOUN/18-19-2/CMPE493/HOMEWORK/HW1/workspace/reuters21578/reut2-018.sgm was processed.
        /DOSYA/BOUN/18-19-2/CMPE493/HOMEWORK/HW1/workspace/reuters21578/reut2-019.sgm was processed.
        /DOSYA/BOUN/18-19-2/CMPE493/HOMEWORK/HW1/workspace/reuters21578/reut2-020.sgm was processed.
        /DOSYA/BOUN/18-19-2/CMPE493/HOMEWORK/HW1/workspace/reuters21578/reut2-021.sgm was processed.
[INFO]  /DOSYA/BOUN/18-19-2/CMPE493/HOMEWORK/HW1/workspace/Output/index.json was created.

[INFO]  Creating bigram-term inverted index...
[INFO]  /DOSYA/BOUN/18-19-2/CMPE493/HOMEWORK/HW1/workspace/Output/bigrams.json was created.


% % % % % % % % % % STATISTICS % % % % % % % % % %
NUMBER OF TOKENS BEFORE STOPWORDS REMOVAL       : 2902784
NUMBER OF TOKENS AFTER STOPWORDS REMOVAL        : 2286351

NUMBER OF TERMS BEFORE STOPWORDS REMOVAL AND CASE-FOLDING       : 61462
NUMBER OF TERMS AFTER STOPWORDS REMOVAL AND CASE-FOLDING        : 45346

20 MOST FREQUENT TERMS BEFORE STOPWORD REMOVAL AND CASE-FOLDING:
| 3 (19162) | of (15480) | said (15344) | the (15052) | Reuter (14225) | and (14188) | to (14112) | a (13346) | in (12828) | lt (11332) |
 The (11242) | for (11031) | it (9494) | s (8560) | mln (8340) | on (8316) | its (7906) | is (7833) | dlrs (7588) | from (7584) |

20 MOST FREQUENT TERMS AFTER STOPWORD REMOVAL AND CASE-FOLDING:
| 3 (19162) | reuter (18885) | said (15435) | to (14845) | lt (11339) | s (10684) | mln (8587) | dlrs (8036) | from (7762) | at (7488) |
1 (7133) | year (6455) | pct (6264) | has (6143) | inc (5998) | company (5507) | 2 (5411) | corp (5350) | u (4770) | 000 (4762) |

PRE-PROCESS TIME (SEC): 60.01
```

A screenshot of pre-process.py

process.py

For processing the queries, I have mainly utilized Python **set** objects.

Type-1 (Conjunctive) Queries

A type-1 query consists of ***AND*** of single-words. Every query word is searched in the term-document inverted index (**index.json** file) and the corresponding document ids are collected separately. Then, the (set) intersection of the collected document ids are printed in increasing order as the result.

```
cba@cba-desktop:/DOSYA/BOUN/18-19-2/CMPE493/HOMEWORK/HW1/workspace$ python3 Code/process.py 1 "computer AND technology"
Processing the query: computer AND technology

[10, 67, 361, 367, 635, 1041, 1659, 1737, 2831, 3027, 3038, 3437, 3679, 3866, 3938, 4055, 4160, 4951, 5302, 5643, 5808,
 6337, 6346, 6749, 7916, 7944, 8252, 8365, 8439, 8506, 8601, 9051, 9488, 9568, 9870, 10278, 10301, 10501, 10556, 10557,
 10829, 10875, 10905, 11169, 11206, 11508, 11524, 12665, 12779, 13246, 14328, 15232, 15530, 16558, 17255, 17497, 17798,
 18113, 18684, 18729, 19379, 19425, 19589, 20823, 21147, 21400, 21438]
```

A screenshot of type-1 (conjunctive) query

CMPE493 – Introduction to Information Retrieval
Assignment 1 Report
A Simple Document Retrieval System for Boolean and Wildcard Queries

CEMAL BURAK AYGÜN (2014400072) **SPRING 2019**

Type-2 (Disjunctive) Queries

A type-2 query consists of ***OR*** of single-words. Every query word is searched in the term-document inverted index (**index.json** file) and the corresponding document ids are collected separately. Then, the (set) union of the collected document ids are printed in increasing order as the result.

```
cba@cba-desktop:/DOSYA/BOUN/18-19-2/CMPE493/HOMEWORK/HW1/workspace$ python3 Code/process.py 2 "istanbul OR ANKARA OR Turkey"

Processing the query: istanbul OR ANKARA OR Turkey

[835, 1611, 1975, 2012, 2246, 2517, 2970, 3047, 3518, 3635, 3745, 3792, 3837, 4328, 5130, 5244, 5274, 5574, 5655, 6870, 6935
, 7105, 7951, 9374, 9831, 9908, 10175, 10367, 10395, 10452, 10511, 10522, 10539, 10620, 10621, 10627, 10630, 10641, 10797, 1
0862, 10910, 11777, 11795, 11823, 11885, 11894, 12522, 12877, 12909, 12943, 12948, 13052, 13129, 13179, 14419, 15200, 15338,
 15884, 16191, 16239, 16491, 16504, 16948, 18034, 18061, 18293, 18676, 18994, 19285, 19499, 19559, 19740, 19884, 20760, 2139
2]
```

A screenshot of type-2 (disjunctive) query

Type-3 (Wildcard) Queries

A type-3 query consists of a single word with exactly one wildcard (*) in it. First, the query keyword is split from * character into two strings; before_wildcard and after_wildcard. Then, before_wildcard is prepended with a ***$*** character to mark it as the start of a term and the bigrams of the resulting string are created. The same operation is done for after_wildcard, except the ***$*** character is appended rather than being prepended, to mark it as the end of a term. The (set) union of the bigrams (coming from '$'+before_wildcard and after_wildcard+'$') are searched in the bigram-term inverted index (**bigrams.json** file) and the corresponding terms are collected in a set. Theni collected terms are filtered to eliminate false positive ones. Filtering is simply comparing each term to check if it starts with before_wildcard and end with after_wildcard. The terms that do not satisfy this condition are discarded. Finally, a type-2 query (see above) is issued by ORing the remaining terms.

```
cba@cba-desktop:/DOSYA/BOUN/18-19-2/CMPE493/HOMEWORK/HW1/workspace$ python3 Code/process.py 3 cl*ss
Processing the query: cl*ss

[23, 87, 141, 609, 677, 803, 827, 850, 1018, 1060, 1075, 1219, 1353, 1408, 1438, 1534, 1571, 1672, 1747, 1755, 1803, 1849, 2
020, 2039, 2058, 2079, 2379, 2513, 2523, 2526, 2533, 2547, 2684, 2816, 3025, 3026, 3077, 3094, 3431, 3623, 3868, 3907, 3999,
 4014, 4140, 4243, 4369, 4373, 4430, 4450, 4465, 4565, 4572, 4594, 4614, 4991, 5054, 5121, 5335, 5377, 5379, 5486, 5537, 558
1, 5585, 5589, 5593, 5673, 5748, 5808, 5890, 5948, 6150, 6275, 6290, 6328, 6332, 6340, 6473, 6709, 6721, 7108, 7130, 7181, 7
198, 7228, 7243, 7286, 7358, 7829, 7932, 7974, 8085, 8091, 8130, 8222, 8227, 8251, 8285, 8351, 8362, 8381, 8406, 8558, 8719,
 8733, 8807, 8864, 8904, 9047, 9078, 9230, 9268, 9269, 9322, 9404, 9454, 9508, 9567, 9653, 9876, 10007, 10068, 10208, 10222,
 10234, 10380, 10481, 10582, 10817, 10822, 10854, 10890, 11038, 11386, 11428, 11522, 11654, 11875, 12004, 12067, 12103, 1210
6, 12138, 12257, 12282, 12443, 12535, 12549, 12574, 12876, 12953, 12978, 12989, 13030, 13115, 13125, 13137, 13202, 13221, 13
907, 14073, 14279, 14668, 14729, 14736, 14741, 14979, 15028, 15055, 15088, 15098, 15184, 15190, 15261, 15320, 15369, 15474,
15633, 15780, 15864, 15902, 15992, 16023, 16176, 16214, 16231, 16237, 16262, 16513, 16534, 16590, 16591, 16832, 17294, 17474
, 17542, 17605, 17638, 17901, 18036, 18206, 18300, 18573, 18684, 18696, 18802, 18944, 19117, 19130, 19146, 19292, 19544, 196
46, 19662, 19833, 19838, 19871, 19953, 20322, 20379, 20398, 20604, 20712, 20905, 20921, 21136, 21287, 21404, 21429]
```

A screenshot of type-3 (wildcard) query