Project 1 Report

Shunyu Wang

40043915

COMP 479 Fall 2019

**Summary of Approach**

The procedure for the python script is

1. Collect all source files and grep each raw document which is enclosed by “reuters” and then clean the source documents by retrieving a list of [newid, document\_body] pair.
2. Input the list from step one into the SPIMI algorithm and it will generate block files according to pre-determined capacity of memory for the global dictionary where temporary inverted-index is held.
3. Now all block files are generated in the disk. The terms in dictionary in each block file are sorted.
4. All block files then are put into merge\_blocks algorithm. The memory will only holds the current top line of each file and build the inverted-index. Since all terms in each file are sorted, it could be guaranteed that the minimum term will be first written into index file. Then the algorithm will read the next line from the files where the previous term comes. Each file is closed after the end of the line is reached and then after all files are exhausted, the algorithm is done. The direct result is index files are created according to the predefined size of index file. Meanwhile, all end words of index files are output as splitting words for directing search of query words.

**Lossy Dictionary Compression Technique Experiments**



The experiments prove the rationale behind the table

Test Queries

1. Single Query – And and Or query for the single word should return the same results.

For example, python3 project\_1\_query.py -o "gpa"

1. And query

python3 project\_1\_query.py -a "This is a good visit"

[INFO] Spliting\_words: ['khafji']

[INFO] System arguments: ['project\_1\_query.py', '-a', 'This is a good visit']

[INFO] Query: ['good', 'visit']

[INFO] Mode: -a

[INFO]----Results------ ['916', '10643', '12472', '14840', '17305', '17477', '17779', '17896']

1. python3 project\_1\_query.py -o "violates grasp"

[INFO] Spliting\_words: ['khafji']

[INFO] System arguments: ['project\_1\_query.py', '-o', 'violates grasp']

[INFO] Query: ['grasp', 'violates']

[INFO] Mode: -o

[INFO]----Results------ ['10486', '17680', '18359', '2537', '3409', '4323', '7037', '8361', '9989', '10583', '12190', '13227', '15469']

Note: The most frequent ids will be returned first

**Test Queries for Demo**

**Below are queries for demo,**

AND Challenge Queries:

**• Jimmy Carter**

[INFO] Spliting\_words: ['khafji']

[INFO] System arguments: ['project\_1\_query.py', '-a', 'Jimmy Carter', '-v']

[INFO] Query: ['carter', 'jimmy']

[INFO] Mode: -a

index: 0 carter=854 965 1993 2534 3380 5465 5525 10252 11114 11118 11330 12136 12677 13540 15158 15473 16887 17023 17325 17363 18005 18438 18962 19432 20614 21308

index: 1 jimmy=12136 13540 17023 18005 19432 20614

[INFO]----Results------ ['12136', '13540', '17023', '18005', '19432', '20614']

**• Green Party**

[INFO] Spliting\_words: ['khafji']

[INFO] System arguments: ['project\_1\_query.py', '-a', 'Green Party']

[INFO] Query: ['green', 'party']

[INFO] Mode: -a

[INFO]----Results------ []

**• Innovations in telecommunication**

[INFO] Spliting\_words: ['khafji']

[INFO] System arguments: ['project\_1\_query.py', '-a', 'Innovations in telecommunication', '-v']

[INFO] Query: ['innovations', 'telecommunication']

[INFO] Mode: -a

index: 0 innovations=247 3192 5230 5386 7051 8483 16128

index: 1 telecommunication=3552 6535 12143 12188 12459 17460 19016 19035

[INFO]----Results------ []

OR Challenge Queries:

**• environmentalist ecologist**

[INFO] Spliting\_words: ['khafji']

[INFO] System arguments: ['project\_1\_query.py', '-o', 'environmentalist ecologist']

[INFO] Query: ['ecologist', 'environmentalist']

[INFO] Mode: -o

[INFO]----Results------ ['5774']