HOMEWORK1

Program Description

Zhu, Kai

CS6322 | kxz160030@utdalla.edu

Program Description

In my program, I use python and its strong lib integration to support tokenization and stemming implementation. It will first scan the target folder, grab all Cranfield files in target folder. Then use read_file() to handle each individual file. For each file, the program will read line by line, then read by tokenize().

For the tokenize part, after several round parsing pilot test, I combine some basic scenrios and my observations to generate a parsing and token rule as below:

1. Replace all SGML tags with space value
2. Replace all punctuation followed or behind space with space value
3. Replace all [] {} () with space value
4. Break Possessives into two part based on two scenario
    a) If word end with ‘s ‘m ‘d, then break into two part by ‘.
    b) If word end with ‘ll ’re ‘ve n’t, then break into two part by this condition.
5. Replace all – or _ with space value.

With the token list returned by tokenize(), it will store in global token table based on hash table which could provide linear access complexity. If one token has been found before, then its value will be accumulated by 1. If not exist, then this token will be added into token table.

After tokenization, my program will call stemming(), which use the porter stemming algorithm to further handle the token list, I use the stemming package in PYPI.[1]

For Stemming part, stemming() will go through the new generated token list to further handler the toke list and combine the same stemmer together to build a stem table.

Program Overview

1. How long the program took to acquire the text characteristics?
   My laptop is I5-2.7G, my running time on text acquiring is around 1 seconds.
   On csgrads1 server, running time on text acquiring is around 2.7 seconds.

2. How the program handles:
   A. Upper and lower case words (e.g. "People", "people", "Apple", "apple");
      All words convert to lower case before into tokenize().
   B. Words with dashes (e.g. "1996-97", "middle-class", "30-year", "tean-ager")
      Words separate by dashes into two word and replace dash with space value.
   C. Possessives (e.g. "sheriff's", "university's")
      Keep ‘s remaining, word separate int two part.

[1] https://pypi.python.org/pypi/stemming/1.0

D. D. Acronyms (e.g., "U.S.", "U.N.")
Keep mid dot remaining, increasing probability for further query.

3. Major algorithms and data structures.
    1. Use hash table to store token with frequency and stem with frequency. Which provide linear time on access, search, insertion and comparison. (Python dictionary)
    2. Use priority queue to generate top 30 frequency tokens or stems. (Python heapq)

Questions

Tokenization

1. The number of tokens in the Cranfield text collections: 236278
2. The number of unique (e.g. distinct) tokens in the Cranfield text collection: 10630
3. The number of tokens that occur only once in the Cranfield text collection: 4674
4. The 30 most frequent word tokens in the Cranfield:
    1. the 19449
    2. of 12714
    3. and 6671
    4. a 5969
    5. in 4642
    6. to 4560
    7. is 4113
    8. for 3491
    9. are 2428
    10. with 2263
    11. on 1943
    12. flow 1848
    13. at 1834
    14. by 1755
    15. that 1570
    16. an 1388
    17. be 1271
    18. pressure 1207
    19. boundary 1156
    20. from 1116
    21. as 1113
    22. this 1081
    23. layer 1002
    24. which 975
    25. number 973
    26. results 885
    27. it 855

28. mach 824
29. theory 788
30. shock 712

5. The average number of word tokens per document: 168.77

Stemming

1. The number of distinct stems in the Cranfield text collection: 7829
2. The number of stems that occur only once in the Cranfield text collection: 3561
3. The 30 most frequent stems in the Cranfield:
    1. the 19449
    2. of 12714
    3. and 6671
    4. a 5969
    5. in 4642
    6. to 4560
    7. is 4113
    8. for 3491
    9. are 2428
    10. with 2263
    11. flow 2079
    12. on 1943
    13. at 1834
    14. by 1755
    15. that 1570
    16. an 1388
    17. pressur 1382
    18. be 1368
    19. number 1347
    20. boundari 1185
    21. layer 1134
    22. from 1116
    23. as 1113
    24. result 1087
    25. this 1081
    26. it 1042
    27. effect 996
    28. which 975
    29. method 886
    30. theori 881
4. The average number of word tokens per document: 5.59