Concordia

Comp 479-A1

Report

Name: Poon Kai Alan, Fok

ID: 26816797

As the assignment requirement, I have to use spimi algorithm to read the documents and split by serval blocks and merge from the blocks to invert index and positing list. And I assume the document has title and body are important. If the document doesn’t content body and title. I assume it is an invalid document.

1. Tokenization And Blocking

I remove all punctuation in body and title in the document and tokenize the body and title. By tokenizing, I used nltk tokenize function and string.punctuation.

temp = str((content.body.text).translate({ord(i): None for i in '\x7f'}))

tokens = nltk.word\_tokenize((temp).translate({ord(i): None for i in string.punctuation}))

After I tokenized the context and title, I put the token to be term and put it into dictionary. Term will be a key in the dictionary and value will be a list. The ID(NEWID) of document will be push to the list. And it will look for another document. When the program read 500 documents, the program will create a block and put the dictionary to the block, then clean up the dictionary and continuous.

Before the program goes the merging process, and the program will count how many link in every block and put zzzzzzzzzzzzzz----:::[] at the bottom in every block to signal the program that is the bottom for all the blocks.

1. Spimi Merging

After all the blocks are created, the program will go to merge process which in Merge.py. . In Merge.py, it contains 3 lists, fullreadbuffer, linecount and indexupdate. Based on the spimi algorithm, the merging process has to read line by line in every block, put the line into a buffer it has to find the smallest term in the reading buffer, and put the term and their posting list in the invert dictionary

The first line of all the blocks will be put into the fullreadbuffer, and find out what is the term is smallest. All smallest term and positing list in the fullreadbuffer will be push to invert index. And the indexupdate will track which index in fullreadbuffer has the smallest term and storage this index to indexupdate. Indexupdate will signal the linecount.

self.lineCounting[index] = self.lineCounting[index] + 1

linecount will tell the blocks which had a smallest term and already put the line which smallest term to read and push the next line to the fullreadbuffer. Since the program know which line in the block that it has to retrieve, I import the library - linecach and use the library function to read line in the block directly. When the problem find the smallest term in the fullreadbuffer is zzzzzzzzzzzzzz----:::[], it means the program already reached the bottom of all blockese. The program will stop merging.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Term | ∆% | T% | Posting list | ∆% | T% |
| unfiltered | 88259 |  |  | 1720266 |  |  |
| No number | 69185 | -21.61 | -21.61 | 1602383 | -6.85 | -6.85 |
| Case folding | 52450 | -24.18 | -40.57 | 1485755 | -7.27 | -13.63 |
| 30 stop words | 52422 | -0.053 | -40.60 | 1452363 | -2.24 | -15.57 |
| 150 Stop words | 52301 | -0.002 | -40.74 | 1143634 | -21.25 | -33.51 |

1. Query

For the query, I created the Query.py and it runs separately in the program. User has to put the query in merge in simpleQuery.

In the Challenge Queries:

simpleQuery = "Jimmy AND Carter"

Jimmy AND Carter [13540, 20614, 12136, 19432, 18005, 17023]

simpleQuery = "Green AND Party"

Green AND Party [21577]

simpleQuery = "Innovations AND in AND telecommunication"

Innovations AND in AND telecommunication []

simpleQuery = "environmentalist OR ecologist"

environmentalist OR ecologist [5774]

1. What I learn in this project

I learnt that computer is so powerful today and we can merge everything by using our ram without optimize the algorithm. However, in the old day, everything is much harder and we need to care all the memory by using optimizing algorithm.