COMP 6791 2013

**Project 1**

**Documentation /Report**

**Lab In Charge – Felix-Herve Bachand**

**f\_bachan@encs.concordia.ca**

**Submitted by: Anish Chavan- 6421180**

**Implementation language :python**

**Problem defination:**

Implement the SPIMI algorithm with disk block merging.

2. Compile an inverted index for Reuters21578 without using any compression techniques (and

make memory size a parameter that you can artificially reduce to test your code)

3. Implement the lossy dictionary compression techniques of Table 5.1 in the textbook and

compile a similar table for Reuters-21578. Are the changes similar? Discuss your findings.

(Note that stemming is not required here, if you run out of time before you get the Porter

stemmer to work, that is ok for this assignment, the remaining table is fine.)

4. Implement a simple scheme to retrieve matching documents for a few queries. Techniques

from Chapters 1-3 are suitable. Show the queries you used and discuss your findings.

**Architecture:**

Project consist of 5 files search file representing following 5 classes represinting the functionalities.

*Class mainmark:*

Its responsible for calling all the functions in other classes and implement critical function of reading from the files and creating inverted index and storing on the hard disk.

*Class mark1:*

This class is responsible for many reading the indexes chunk of data at time and processing it and calling tokenizer.

*Class Tokenizer:*

Responsible for tokenizing the words of the document

Class mark8:

*Its responsible for performing spimi on the small chunk of data and thus writing the result to the disk.*

Class Parser1:

Its responsible for parsing the data between <BODY> and </BODY>

**Lossy compression schemes:**

The main class consist of function which is Indexcompression().

The main task of the function is:

* To remove the numbers from the dictionary.
* To remove the stop words which are listed in stopword.txt

**Datastructures and libraries use:**

The following data structures were used.

dict() in built python dictionary.

list() in built python data structure

Pickle, String were few of the libraries used .

**SPIMI: writing to disk:**

SPIMI was used to create small indexes from the chunks read from the file .

the chunk size is set dynmaic and user can choose the chunk size for fast processing .

***Code Sample:-***

*with open(filename1) as f:*

*while True:*

*chunk = f.read(chunksize1)*

*if chunk =="":*

*break*

*send=par.parser()*

*def IndexWriter(self,newlist):*

*print("============================================================================new index created =========================================")*

*spimi.count=spimi.count+1*

*pickle.dump(newlist,open("index%d"%spimi.count+".p","wb"*

**SPIMI: reading from disk:**

Once the reading job was done it was task to read all the files and merge them in single index and write them on the disk again.

this each index was read and simultaneously wrote on the disk