Peter El-Jiz
6250661

	Distinct Terms			Non Positional Postings		
	#	Δ	T	#	Δ	T
Unfiltered	112420			1912577		
No Punctuation, No dash	54008	-51.96	-51.96	1871225	-2.16	-2.16
No Numbers	49949	-7.51	-55.57	1676335	-10.41	-12.35
Case Folding	43072	-13.76	-61.69	1620641	-3.32	-15.26
Stop Words	42779	-0.68	-61.94	1140096	-29.65	-40.38
Stemming	32404	-24.25	-71.17	1118817	-1.86	-41.5

Compression Techniques Findings:
In the end, I used all compression techniques, I first filtered the text, got rid of noisy data (weird characters, punctuations, joining words separated by a dash into one word). I then got rid of all numbers, made all words lowercase, and filtered the most common words (words that appear in my stop list).
Using lossy compression techniques allow us to store more crucial information (vs redundant) in memory at a time. By halving the memory requirements, I reduce the number of operations the algorithm needs to process all files.
Using all the techniques above, we end up gaining a 71% decrease in terms, and a 41% decrease of postings.

Data Structure of my Inverted Index:
In this project, I used data structures provided by the JAVA API, I did not write any of my own. For parsing the documents and collecting term – document pairs, I used a LinkedHashMap<String, ArrayList<String>>. A LinkedHashMap preserves insertion order. The first String represents the docID, and the ArrayList<String> represents the words (after being processed) in that document. I divide the available memory according to a certain percentage. Some of the memory is allocated for parsing list task (will keep adding to the list, till it reaches allocated memory for it). The remaining memory is used for the SPIMI construction.

Elements are retrieved from the list; terms are added to a new Hash Map structure. A posting list is created for each term (if it doesn't exist already) and document IDs are added to it. After we're done processing a document, we remove it from list (freeing up memory). Once we're done processing all documents, or run out of memory, we return the new list to the main file (so as to not lose the added elements to it), and save the processed ones to disk.
Since the docIDs are read in ascending order. The postings list will be in sorted order. We use a Tree map for sorting the term (keys), because Tree Maps maintain natural sorting order; thus terms will also be sorted.

The final Inverted Index DS is a tree map of (term – postings list), term is the key, the posting list is the value. Posting list is an ArrayList<Integer>.

Peter El-Jiz
6250661

Describe how your implementation of SPIMI writes this data structure to disk (include the file names and line numbers where this is implemented in your code).

Since JAVA DS implement the Serializable Interface, I write my Maps as objects to disk. The blocks are encoded and may not be human-read. The file name is SpimiHelper.java, the line is , the method is
public static LinkedHashMap<String, ArrayList<String>>
SPIMIConstruction(LinkedHashMap<String, ArrayList<String>> list, File outFile):

The code snippet is:

```
try
{
FileOutputStream fileOutStream = new FileOutputStream(outFile);
ObjectOutputStream objectOutStream = new ObjectOutputStream(fileOutStream);
objectOutStream.writeObject(sortedTerms);
objectOutStream.close();
fileOutStream.close();
}
catch (FileNotFoundException e)
{
// TODO Auto-generated catch block
e.printStackTrace();
}
catch (IOException e)
{
// TODO Auto-generated catch block
e.printStackTrace();
}
```

Describe how your implementation of SPIMI merges these disk files to perform queries (again, include the file names and line numbers where this is implemented in your code):

My merge implementations merges 2 saved files at a time. The way this is done is:
I first list all block files in my blocks directory. Every 2 blocks are merged and saved as a new file (a combination of both), the new merged block file is then merged with the 3rd block:
(ie: Files: block0.blk, block1.blk, block2.blk, block3.blk, block4.blk
block0.blk and block1.blk are merged: the new merged file is saved as block0&1.blk;
block0&1.blk and block2.blk are merged: the new merged file saved as block0&1&2.blk
block0&1&2.blk and block3.blk are merged: the new file is block0&1&2&3.blk)
block0&1&2&3.blk + block4.blk ➔ block0&1&2&3&4.blk.
The file name is retained)
The garbage collector is called to free up some memory and take care of any remnants)

Peter El-Jiz
6250661

block0&1&2&3&4.blk is deserialized (read into memory) into the TreeMap<String, ArrayList<Integer> finalIndex.

The snippet of code for the merging of n block files can be seen in Spimi.java on line 113.

The merging of two block files goes as follows: In Spimi.java 3 File objects are created, (the 1st two referring to the block files to be merged, and the 3rd referring to the output file). Those File Objects are passed to a static method in SpimiHelper.java called mergeTwoBlocks(File file1, File file2, File outFile).

In mergeTwoBlocks: file1 and file2 are deserialized and stored in 2 Tree Maps which will be merged.

Here is the actual merge code:

```
//Merge Code
Iterator i1 = file1List.entrySet().iterator();
Iterator i2 = file2List.entrySet().iterator();


Entry i1Entry = null;
Entry i2Entry = null;

if ((i1.hasNext()) && (i2.hasNext()))
{
	i1Entry = (Entry) i1.next();
	i2Entry = (Entry) i2.next();
}

TreeMap<String, ArrayList<Integer>> temporary = new TreeMap<String, ArrayList<Integer>>();


while ((i1.hasNext()) && (i2.hasNext()))
{
	//i1 term
	String i1Term = (String) i1Entry.getKey();//i1Entry.getKey();

	//i2 term
	String i2Term = (String) i2Entry.getKey();

	//System.out.println("i1: " + i1Term);
	//System.out.println("i2: " + i2Term);



	if (i1Term.equalsIgnoreCase(i2Term)) //equal terms
	{
		//Merge Postings List
```

```
                    //i1 documents
                    ArrayList<Integer> i1Postings = (ArrayList<Integer>) i1Entry.getValue();

                    //i2 documents
                    ArrayList<Integer> i2Postings = (ArrayList<Integer>) i2Entry.getValue();


                    //add i2's documents to i1 (i2's document ids > i1's) in natural ascending order.
                    Iterator p = i2Postings.iterator();
                    while (p.hasNext())
                    {
                            Integer docValue = (Integer) p.next();
                            if (!i1Postings.contains(docValue))
                                    i1Postings.add(docValue);
                    }

                    i1Entry = (Entry) i1.next();
                    i2Entry = (Entry) i2.next();

            }
            else
                    if (i1Term.compareTo(i2Term) < 0) //i1Term < i2Term
                    {
                            //advance i1
                            i1Entry = (Entry) i1.next();

                            //no need to add anything, because the element exists in the i1Postings

                            //i2 stays the same
                    }
                    else
                    {

                            ArrayList<Integer> i1Postings = (ArrayList<Integer>) i1Entry.getValue();
                            ArrayList<Integer> i2Postings = (ArrayList<Integer>) i2Entry.getValue();

                            //add i2
                            temporary.put(i2Term, (ArrayList<Integer>) i2Entry.getValue());

                            //advance i2
                            i2Entry = (Entry) i2.next();

                            //i1 stays the same
                    }

}


//Add elements that we couldn't add, because we couldn't modify our ds while iterating
file1List.putAll(temporary);


while (i2.hasNext())
{
```

```
        ArrayList<Integer> i2Postings = (ArrayList<Integer>) i2Entry.getValue();
        String i2Term = (String) i2Entry.getKey();

        if (!file1List.containsKey(i2Term))
        {
                Iterator p = i2Postings.iterator();
                ArrayList<Integer> postings = new ArrayList<Integer>();

                while (p.hasNext())
                {
                        Integer docValue = (Integer) p.next();
                        postings.add(docValue);
                }

                file1List.put(i2Term, postings);

        }
        else
        {
                Iterator p = i2Postings.iterator();
                ArrayList<Integer> postings = file1List.get(i2Term);

                while (p.hasNext())
                {
                        Integer docValue = (Integer) p.next();
                        postings.add(docValue);
                }

                file1List.put(i2Term, postings);

        }



        //file1List.put((String) i2Entry.getKey(), (ArrayList<Integer>) i2Entry.getValue());
        i2Entry = (Entry) i2.next();

}
```

How this code works is as follows:
We iterate through both tree maps.
If our keys are equal, we add all elements from (file2's posting list for that key) to (file1's posting list for that key) if it is not already in the file1's list. If it is in file1's list, we simply skip the posting and move to the next posting

If file1's key is < file2's key:
This means that the element does not exist in file2 (since it is an ascending order). The iterator will keep going to the next element till file1's key and file2's key are either equal (same elements and merge) or till file2's key > file1's key (meaning the element does not exist in file1 and we have to add it). Since we cannot add to a an element over which we are still iterating, we add it to a temporary map from which

we'll later add back to file1's list.
If we're done iterating over either file1 or file2, we exit the loop and move on to:
Adding the elements from temp back to file1's list
Adding any remaining elements (if any) in file2 back in file1.
(file1's list might be shorter than file2, so we have to make sure that all elements of file2 are properly handled)

As long as file2 still has elements we haven't iterated over:
get file2's key and posting list. Check if file1 does not have the element
If the element does not exist in file1, add it (term + posting list)
If the element does exist in file1, get file1's posting list, get file2's posting list. Iterate over file2's posting list, and add each element to file1's posting list. (Since the documents are read in ascending order, we do not have to worry about duplicate documents, or sorting them, and add adds to the tail of the list, so we have nothing to worry about).

Discuss how your implementation of SPIMI influences performance, number of files created on disk, size of files created on disk, or anything else that seems relevant.
In computer science, we speak of a big O notation, which represents algorithmic speed. Usually we do a compromise of memory (space) / speed. The bigger the memory capacity, the more we can sophisticate our algorithm. Also since I had the memory space available, I decided to use it, and optimize my algorithmic speed. When it comes to parsing and reading in the file, I take advantage of the memory and write less efficient and more memory demanding code. I'm reading in the whole reuters file (using the scanner class) and then processing my string. (look for NEWID= and <BODY>, get data in between, etc). I could have done more efficient (but boilerplate, unfortunately) code, and read in line by line, but it was easier for me to do it that way. I paid the consequences of having to be careful with the memory allocation (parsing-saving to blocks). For the saving part to run without errors, I have to leave it 20-25 MB of JVM memory. Depending on how much I leave the saving to blocks. The less I leave memory time for saving, the less Blocks I have, but the bigger they end up. The more I leave memory time for saving, the more blocks I have, the less they are in size.

Give an example of a query or two:
Enter your query type: AND or OR
and
Enter your query: (without the operators)
cocoa bank zone

Document ID: 10936

Peter El-Jiz
6250661

The Governor of the Central Bank of
West African CFA franc zone countries (UMOA) today urged that the bank should be involved in all negotiations with the International Monetary Fund but said the zone's seven member countries were not thinking of ditching the fund.

Abdoulaye Fadiga told reporters after a regular one-day meeting of UMOA finance ministers here that the bank's participation in future credit talks with the IMF would promote a greater coherence in Fund aid granted to the region.

But he stresed that UMOA, which last month called for a new relationship with the IMF, was not contemplating a break with the fund.

"Member states (of UMOA) never envisaged breaking with the Fund," he said.

Fadiga said an extraordinary UMOA meeting held in Ivory Coast's inland capital, Yamoussoukro, last month, stresssed the need for growth-oriented IMF adjustment policies.

"Borrowing to pay off debts cannot be a solution," he added.

UMOA members are Benin, Burkina Faso, Ivory Coast, Mali, Niger, Senegal and Togo. The zone's common-curreny, the CFA franc, is pegged to the French franc at the rate of 50 CFA francs to one French franc.

Fadiga said many of the region's problems stem from low commodity prices and a weak dollar.

He applauded last week's agreement between producers and consumers on the functioning of a new international cocoa pact, saying negotiation of other commodity accords would benefit UMOA countries.

He also urged action to persuade West Africans to transfer more money through the regular banking system rather than using non-official channels.
Reuter

Document ID: 10995
The Governor of the Central Bank of
West African CFA franc zone countries (UMOA) today urged that the bank should be involved in all negotiations with the International Monetary Fund but said the zone's seven member countries were not thinking of ditching the fund.

Abdoulaye Fadiga told reporters after a regular one-day meeting of UMOA finance ministers here that the bank's participation in future credit talks with the IMF would promote a greater coherence in Fund aid granted to the region.

But he stresed that UMOA, which last month called for a new relationship with the IMF, was not contemplating a break with

the fund.
"Member states (of UMOA) never envisaged breaking with the Fund," he said.
He said an extraordinary UMOA meeting held in Ivory Coast's inland capital, Yamoussoukro, last month, stresssed the need for growth-oriented IMF adjustment policies.
"Borrowing to pay off debts cannot be a solution," Fadiga added.
UMOA members are Benin, Burkina Faso, Ivory Coast, Mali, Niger, Senegal and Togo. The zone's common-curreny, the CFA franc, is pegged to the French franc at the rate of 50 CFA francs to one French franc.
Fadiga said many of the region's problems stem from low commodity prices and a weak dollar.
He applauded last week's agreement between producers and consumers on the functioning of a new international cocoa pact, saying negotiation of other commodity accords would benefit UMOA countries.
Reuter

Enter your query type: AND or OR
or
Enter your query: (without the operators)
aapplied zurack

Document ID: 8024
The four-times-a-year "triple witching hour" did not jolt Wall Street as much as it has in the past.
Market averages finished sharply higher as stock index futures, index options and options on individual stocks expired simultaenously. Some analysts warned part of the gain may be retraced next week. But there were signs Wall Street is getting used to the phenomeon which causes a huge burst of activity in the final minutes. Officials of the New York Stock Exchange said the day went smoothly and efficiently.
"This has been one of the few times that the consensus has been right," said Jim Creber, trader at Miller, Tabak, Hirsch and Co.
He expects a "follow-through" Monday and Tuesday to the advance which lifted the Dow Jones Industrial Average 34 points for the day and 75 points for the entire week.
Creber, whose firm was one of the first to get involved in arbitrage activity involving index futures and options, said the general trend of the market has been upward for months.

"Every time the market comes in, somebody comes along with more money," he said.

He said investors adding to Individual Retirement Accounts prior to a mid-April tax deadline and buying from Japanese investors are apparently helping push stocks higher.

Ron Feinstein, an analyst with Dean Witter Reynolds Inc, said reports of heavy Japanese buying, just prior to the end of the fiscal year in Japan, fueled bullish sentiment.

He said investors who had long positions in stocks hedged with short positions in index futures rolled the expiring March futures over into June contracts. But he added different players with other goals also were active and there was no simple explanation of the market's gyrations.

For example, Feinstein noted the June contract for the June Standard and Poor's 500 stock index future hit 300 about 15 minutes prior to the close of NYSE trading. In 12 minutes, the contracted dipped to 297.50. "That could have been a wave of selling from institutional people making a roll," he said.

It was the first time a nearby contract of the S and P 500 hit 300. The cash index closed at a record 298.17. "It looks like the market is going to continue to go higher," he said.

"Triple expirations didn't really mean that much, it was a strong day for stocks," said Steve Chronowitz of Smith Barney, Harris Upham and Co.

Chronowitz said the stock market has been underpinned by "good solid buying interest" which will cushion any pullback.

Other investors who were long futures and short stocks bought stocks on the close today instead of rolling over, said Mark Zurack of Goldman, Sachs and Co. He cautioned against "over-dramatizing" the day's activity, which said was more a reflection of fundamental strength in the markets.

Leigh Stevens of PaineWebber Group Inc said what he saw was mostly covering of short positions in stocks as index options and individual options expired. He said there could be a decline early next week in the stock market.

"It looked like it worked well today," said Howard Kramer, asssistant director of market regulation for the Securities and Exchange Commission. But he said all of the data will have to be analyzed next week.

He said there was relatively little commotion at the close, with about 50 mln shares changing hands in the final minute compared to 85 mln in the triple expirations three months earlier. He noted the Dow industrials jumped about 12 points in the closing minutes, a modest move for a 2333-point index.

Kramer pointed out an SEC-NYSE measure to curb volatility, disclosure of "market-on-close" orders in 50 major stocks 30

minutes prior to the end of trading, showed imbalances of modest proportions. The disclosures are aimed at evening out volatility by attracting orders on the opposite side of the imbalance.

The data showed more buy orders than sell orders for 47 stocks, a preponderance of sell orders for only one stock, and no significant imbalance for two stocks. The NYSE had tightened a rule governing what type of market on close orders can be accepted in the final half hour.

Reuter

Document ID: 19643

D and N Savings Bank said it
aapplied to the Federal Home Loan Bank Board for approval to open three new branches inside K mart Corp <KM> stores in Michigan.

It said one of the branches being applied for is in Grand Rapids, Mich., which would bring to five the number of K mart locations in that city with D and N Bank Marts.

It said the other applications are for branches in Flint K mart stores.

Reuter