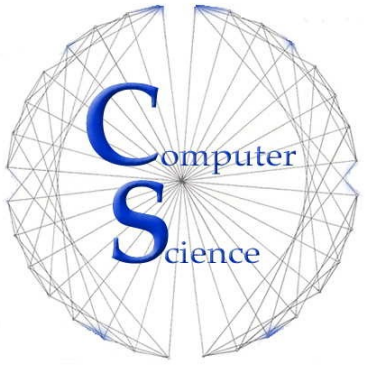


Information Retrieval Project

Prashanth Sandela
Saravana Kumar Gajendran
Sai Kiran Reddy

Advisor Professor Akash Nanvati

Table Of Contents

To: Professor Akash Nanvati
San José State University

Date: 05/02/2014

From: Prashanth Sandela (009403563)
Saravana Kumar Gajendran (009407099)
Sai Kiran Reddy (009325758)
Computer Science Department
San José State University

Subject: Information Retrieval Final Project

Introduction :

Downloaded copy of wikipedia dataset (45 Gb). Took pages only if namespace is 0. Ignored files, images, template etc as per professor advice. Pagerank is used from previous assignment. Tried to fit the index in memory, used compression for the index. Implemented a ranking algorithm to return top 10 ranked documents for the query. we have covered interesting part of our coding in the document and entire coding is also attached is separated file.We followed 64-bit Java.

Below is the System configuration we have worked. Entire coding is in Java, Eclipse is used.

System configuration		
Processor	2.4 i5	2.9 i7
Memory	8GB	8GB
OS	OS X	ubuntu
Hard Disk	256 SSD	1TB

1. XML Parser :

Inside create page class we have specified the path to process the xml file and generate the tokens and write it to output file ParseXML method is used to process the XML file and remove the namespace and not redirects.

```
public static void parseXML() throws XMLStreamException, FactoryConfigurationError,
    XMLEvent xmlEvent = null;
    Characters chars = null;
    boolean pageIdNext = false;
    boolean isText = false;
    boolean isTitle = false;
    boolean isId = false;
    boolean redirect = false;
    StringBuilder pageContent = new StringBuilder();
    String id = "";
    int count = 1;
    boolean isNs = false;
    String ns = "";
    StringBuilder idTitle = new StringBuilder();
    StringBuilder urls = new StringBuilder();
    String title = "";

    Pattern pattern = Pattern
            .compile("\\b((http\\:\\/\\/|~\\/|\\/))+(en.wikipedia.org)+"
                    + "(:[\\d]{1,5})?"
                    + "((((\\/([-\\w~!$+|.,=]|%[a-f\\d]{2})+)+|\\/)+|\\?|#)?"
                    + "((\\?([-\\w~!$+|.,*:]|%[a-f\\d{2}])+=?"
                    + "([-\\w~!$+|.,*:=]|%[a-f\\d]{2})*)"
                    + "(&(?:[-\\w~!$+|.,*:]|%[a-f\\d{2}])+=?"
                    + "([-\\w~!$+|.,*:=]|%[a-f\\d]{2})*)*)*"
                    + "(#([-\\w~!$+|.,*:=]|%[a-f\\d]{2})*)?\\b");
```

1.1 CreateFile method is used to write the output to file.

```
public static void createFile(String filename, String string)
{
    BufferedWriter bw = null;
    try {
        bw = new BufferedWriter(new FileWriter(outputFolderPath + "/'
        bw.write(string);
    } catch (Exception e) {
        System.out.println("Error in 3: " + e);
    }
    finally
    {
        try
        {
            if ( bw != null)
                bw.close( );
        }
        catch ( IOException e)
        {
            System.out.println("Error in 4: " + e);
        }
    }
}
```

This Method is used to create a file

1.2 RemoveDuplicates Method : In this methods we have created a array and two strings "pref and next" to remove the duplicates, we have used bit operator inside if loop to accomplish the task.

```
private static String removeDuplicates(StringBuilder urls) {
    String[] strArray;
    StringBuilder sb = new StringBuilder();
    strArray = urls.toString().split("\n");
    Arrays.sort(strArray);
    String prev = " ";
    String next = "";

    int len = strArray.length;
    for (int i = 0; i < len; i++) {
        next = strArray[i].trim();
        if (!prev.equalsIgnoreCase(next) && !next.isEmpty() && next != "\n")
            sb.append(next + "\n");
        }
        prev = strArray[i];
    }
    return sb.toString();
```

Removes the duplicates

2.General.java : It contains frequently used functions.

- 2.1 Createfile() method takes filename and string as arguments. readStream() method takes filename as argument and returns BufferedReader.
- 2.2 getTime() method is used to know the start and finish times of the process.WhereSimpleDateFormat() method is used to denote the format.

```
public class general {
    public static void createFile(String filename, String string, Boolean append)
    {
        BufferedWriter bw = null;
        try {
            bw = new BufferedWriter(new FileWriter(filename, append ? true : false));
            bw.write(string);
            System.out.println("File Written: " + filename);
        } catch (Exception e) {
            System.out.println("Error in 3: " + e);
        }
        finally
        {
            try
            {
                if ( bw != null)
                    bw.close( );
            }
            catch ( IOException e)
            {
                System.out.println("Error in 4: " + e);
            }
        }
    }

    public static BufferedReader readStream(String filename) throws UnsupportedEncodin
        BufferedReader br = new BufferedReader(new InputStreamReader(new FileInputStre
        return br;
    }

    public static String getTime(){
        Date dNow = new Date();
        SimpleDateFormat ft = new SimpleDateFormat("hh:mm:ss SSS");
        return ft.format(dNow);
    }
```

In general class we have created three methods for creating a file reading a file (BufferedReader) and Get Time method

3.1 External sort(Sorttokens.java):

It is used to sort the tokens. External sorting can handle massive amount of data. It is used when the data being sorted do not fit into the RAM and instead they must reside in the external memory. Here small chunks of data enough to be fit in the RAM are read, sorted, and written out to a temporary file.Later sorted subfiles are merged into a single large file.

```
private static void externalSort(short tempFiles, short currFile) throws
{

    System.out.println("Sorted File: " + currFile + " of " + tempFiles);

    BufferedReader br1 = general.readStream(tokens + "InterFile.txt");
    BufferedReader br2 = general.readStream(tokens + "tmp_" + currFile +

    String line1, line2;
    String word1, word2;
    StringBuilder sb = new StringBuilder();

    long count = 0;
    line1 = br1.readLine();
    line2 = br2.readLine();
    while( true )
    {

        if( line1 == null  || line2 == null )
        {
            System.out.println("exiting one file ");
            break;
        }

        if(count % 20000 == 0)
            System.out.println("ExternalSort: " + count);

        short sr1 = (short) line1.indexOf("=");
        short sr2 = (short) line1.indexOf("=");

        if(sr1 == -1 || sr2 == -1)
            break;

        word1 = line1.substring(0, ( sr1 == -1 ) ? 0 : sr1 ).trim();
        word2 = line2.substring(0, line2.indexOf("=")).trim();
```

3.2 GenerateTokens : It is used to generate and sort the tokens in alphabetic order. We used TreeMap, SortedMap, SortedSet in this process.

```
public static void generateTokens() throws IOException
{
    String[] a = {"a", "b", "c", "d", "e", "f", "g", "h", "i", "j", "k"

    for(String fp: a)
    {
        fp = fp + ".txt";
        BufferedReader br = general.readStream(tokens + fp);
        String line = null;
        int count = 1;
        short tempFiles = 1;

        SM = new TreeMap<String, SortedSet<Integer>>();

        while((line = br.readLine()) != null){

            String[] temp = line.trim().split(" ");
            if(SM.get(temp[0]) == null)
            {
                SortedSet<Integer> ss = new TreeSet<Integer>();
                ss.add(Integer.parseInt(temp[1]));
                SM.put(temp[0], ss);
            }
            else
            {
                SM.get(temp[0]).add(Integer.parseInt(temp[1]));
            }

            count++;
            if(count % 20000 == 0)
            {
                System.out.println(count  + " " + " Size:" + SM.size()
            }
            if(count % 5000000 == 0){
```

Sorts the words from a to z

3.3 SortInt :
It sort integers using sort set

```
public static String sortInt(String string)
{

    String[] str = string.trim().split(",");

    if(str.length == 1)
        return string;

    SortedSet<Integer> ss = new TreeSet<Integer>();

    for(String s: str)
    {
        if(s != null)
            ss.add(Integer.parseInt(s));
    }

    return ss.toString().replaceAll("], ", "\n").replaceAll("\\[|\\]|\\{|\\}| ", "");
}
```

4. Compression:

We used dictionary compression and posting list compression to achieve this.The main goal of compressing the dictionary is to fit it in the main memory to support high query throughput.

In dictionary compression, we reduced the number of bytes occupied by posting list. 11 million words occupied 90 MB of memory. When they are stored as a single line, it occupied 90 MB.

4.1 Dictionary compression

- we implemented it by replacing multiple lines by storing dictionary in single line.
- we followed concept of front coding and blocks.

Below is code of implementation

```
public static void dictionaryCompression(String[] tempWords)
{
    short len = (short) tempWords[0].length();
    for(int j = 0; j < len; j++)
    {
        boolean found = true;
        for(int i = 1; i < blockSize; i++)
        {
            if(  !tempWords[i].substring(0,j + 1).equals(tempWords[0].substring(0,j + 1)) )
            {
                found = false;
                //System.out.println(i);
                break;
            }
        }

        if(!found || j == len - 1)
        {
            wordPointer[wordPointerInc] = token.length();
            String comm = j + tempWords[0].substring(0, j);

            for(int i = 0; i < blockSize; i++)
            {
            //  System.out.println(( tempWords[i].length() - j ) + tempWords[i].substring(j));
                comm += ( tempWords[i].length() - j ) + tempWords[i].substring(j);
            }
            //System.out.println("Term Doc= " + comm);
            token.append(comm);
            wordPointerInc++;
            break;
        }
    }
}
```

4.2 Posting List :

In posting list compression, we reduced the number of bytes used in the memory by taking the difference of successive numbers in the lists and storing them in the memory.

- Current index is calculated by deference of previous index by current index
- Followed compression technique in byte encoding format in Encode function.

```
public static void  postingListCompression(String line)
{
    wordCount++;
    int prevDocId = 0;
    int curDocId = 0;
    StringBuilder sb = new StringBuilder();
    String str[] = line.split(",");
    for(String s: str)
    {
        if( s != null && !s.isEmpty())
        {
            curDocId = Integer.parseInt(s);
            int docIdDiff = curDocId - prevDocId;
            sb.append(docIdDiff + ",");
            prevDocId = curDocId;
        }

    }
    tokenSet[wordCount] = encode(sb.toString());
}
```

5. Web Graph:

We have used webgraph to compute the PageRank, our webgraph describes inlinks between pages in Wikipedia data set

```
private static void populateinlinksGraph() throws IOException
{
    BufferedReader br = general.readStream(filePath1);
    String line;
    int index;
    int lineCount = 0;
    while( (line = br.readLine()) != null)
    {
        line = line.trim();
        index = line.indexOf("\t");
        System.out.println(index);
        if(index != -1)
        {
            int Id = Integer.parseInt(line.substring(0,index));

            String OTtitle = line.substring(index + 1).trim();
            System.out.println(OTtitle);
            if(urlTitle.containsKey(OTtitle))
            {
                Integer key = urlTitle.get(OTtitle);
                if(inLinks.get(key) != null)
                {
                    inLinks.get(key).add(Id);
                }
                else
                {
                    SortedSet<Integer> ss = new TreeSet<Integer>();
                    ss.add(Id);
                    inLinks.put(key, ss);
                }

            }
        }
```

6. Page Rank :

The PageRank is the results of our algorithm based on the webgraph that we have computed previously.

```
private static void computePageRank() {

    String[] idArr = ids.toString().split(",");
    int count = 0;
    for(String id: idArr)
    {
        if(count == 0)
            pageRank.add((float) 1);
        else
            pageRank.add((float) 0);
    }

    count = 0;
    Float[] pageR = (Float[]) pageRank.toArray();
    pageRank = new ArrayList<Float>();
    for(String id: idArr){
        count++;
        float Sum = 0;
        String[] line = idWeight.get(id).substring(0,idWeight.get(id).indexOf(";")).split(",");
        for(String s: line)
        {
            int j = ids.indexOf(s);
            for(int k = 0; k < j; k++ ){
                Sum += zeroEntry * pageR[count];
            }
            Sum += (fixedTerm + line.length * epsilon)/(line.length * n);
        }
        pageRank.add(Sum);
    }
}
```

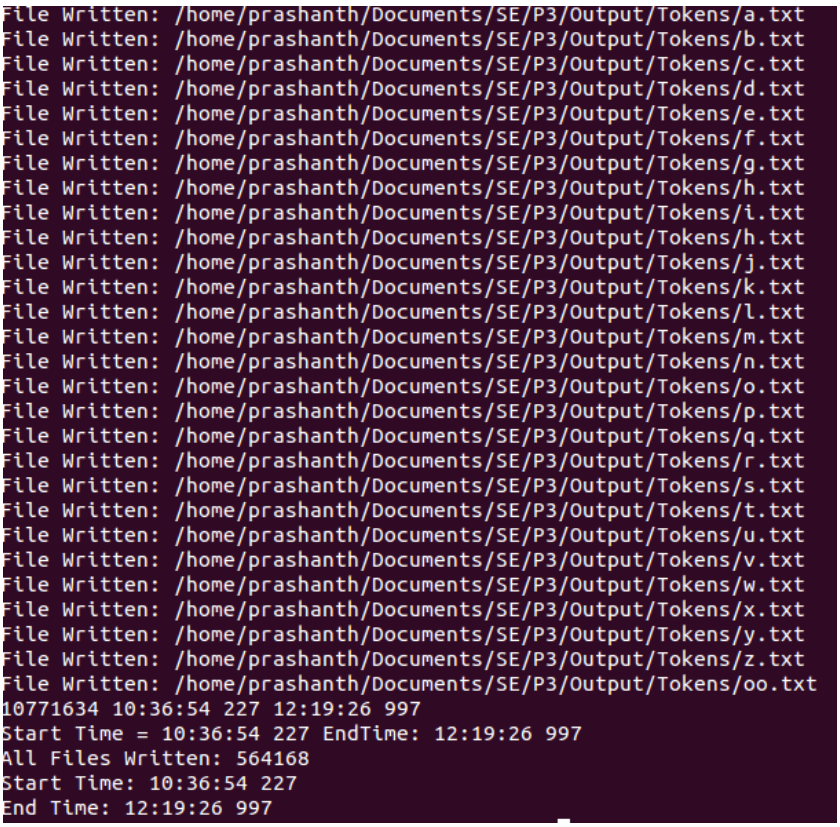
7. Web Search :

This program initiate compression process stores everything into memory, loads indexes and computes web search on input query.

```
private static void getTitles(int[] indexes) throws IOException {

    SortedSet<Integer> ss = new TreeSet<Integer>();
    String[] sets = new String[10];

    for(int i: indexes)
    {
        ss.add(i);
    }

    BufferedReader br = general.readStream(idUrls);
    String line;
    int i =0;
    while( (line = br.readLine()) != null)
    {
        int index = line.indexOf(",");
        int id = Integer.parseInt(line.substring(0, index));
        if( ss.contains(id))
        {
            sets[i] = line.substring(0, index + 1);
        }
        i++;
    }

    System.out.println("Top 10 URL's");
    for(String s: sets)
    {
        System.out.println("http://en.wikipedia.com/wiki?title=" + s.replace(" ", "_"));
    }

}
```

8 Outputs:

8.1 XML Parser Output:



8.2 Sorted output:

```
2300000  Size:148802 z.txt Time: 01:06:09 623
2320000  Size:149709 z.txt Time: 01:06:09 642
2340000  Size:150606 z.txt Time: 01:06:09 664
2360000  Size:151363 z.txt Time: 01:06:09 681
Generated for z.txt
File Written: /home/prashanth/Documents/SE/P3/Output/Tokens/tmp_1.txt
File Written: /home/prashanth/Documents/SE/P3/Output/Tokens/InterFile.txt
Sorted File: 1 of 1
ExternalSort: 0
Writing remaining files
File Written: /home/prashanth/Documents/SE/P3/Output/Tokens/tmp.txt
File Written: /home/prashanth/Documents/SE/P3/Output/Tokens/tmp.txt
Done. Start Time: 10:57:23 715 End Time: 01:06:11 929
```

8.3 Dictionary Compression Output:

```
Max Memory: 6371
Allocate Memory: 5721
Free Memory : 272
File write: Done
Total Tokens:11865654

Start Time : 05:58:19 173
End Time   : 05:02:48 113
```

8.4 Compression Outputs:

```
Result       : Done
Total Tokens : 118656547
Start Time   : 12:06:13 179
End Time     : 12:13:45 153
```

8.5 Web Graph

```
Writing to File System:
Writing idTitle File
Written WebGraph File: Count= 1000000
File Written: /home/prashanth/Documents/SE/P3/Output/4_WebGraph/idTitle.txt
Written WebGraph File: Count= 2000000
File Written: /home/prashanth/Documents/SE/P3/Output/4_WebGraph/idTitle.txt
Written WebGraph File: Count= 3000000
File Written: /home/prashanth/Documents/SE/P3/Output/4_WebGraph/idTitle.txt
Written WebGraph File: Count= 4000000
File Written: /home/prashanth/Documents/SE/P3/Output/4_WebGraph/idTitle.txt
File Written: /home/prashanth/Documents/SE/P3/Output/4_WebGraph/idTitle.txt
Writing WebGraph File
File Written: /home/prashanth/Documents/SE/P3/Output/4_WebGraph/webgraph.txt
Start Time: 10:48:05 057 End Time: 10:48:33 830
```

8.6 Page Rank

```
Started
Writing into file: ~/prashanth/Documents/SE/src/P3/Output/5_pageRank/pageRank.tx
t
Start time: 10:05:34 End Time: 10:20:27
```

8.7 Total Number of Tokens

```
 69156336 a.txt
 48394270 b.txt
 93318974 c.txt
 57084416 d.txt
 40266095 e.txt
 41736896 f.txt
 33006911 g.txt
 35864682 h.txt
 38058404 i.txt
 16760179 j.txt
 12358584 k.txt
 49118462 l.txt
 55279290 m.txt
 28548509 n.txt
 22494896 o.txt
 76573931 p.txt
  5374432 q.txt
 61182709 r.txt
 95868583 s.txt
 44425288 t.txt
 14596056 u.txt
 14234706 v.txt
 32795876 w.txt
   866410 x.txt
  6824985 y.txt
  2359661 z.txt
996549541 total
```

8.8 Total Unique Tokens Count :

```
  769179 a.txt_sorted.txt
  720888 b.txt_sorted.txt
  778408 c.txt_sorted.txt
  540113 d.txt_sorted.txt
  388702 e.txt_sorted.txt
  398930 f.txt_sorted.txt
  479381 g.txt_sorted.txt
  454266 h.txt_sorted.txt
  322584 i.txt_sorted.txt
  250784 j.txt_sorted.txt
  532855 k.txt_sorted.txt
  445534 l.txt_sorted.txt
  794975 m.txt_sorted.txt
  426064 n.txt_sorted.txt
  288653 o.txt_sorted.txt
  715778 p.txt_sorted.txt
  100596 q.txt_sorted.txt
  482604 r.txt_sorted.txt
 1062544 s.txt_sorted.txt
  650357 t.txt_sorted.txt
  214131 u.txt_sorted.txt
  290266 v.txt_sorted.txt
  318625 w.txt_sorted.txt
   74799 x.txt_sorted.txt
  135787 y.txt_sorted.txt
  151312 z.txt_sorted.txt
11788115 total
```

8.9 Web Search :

```
http://en.wikipedia.com/wiki?title=Phish_tours
http://en.wikipedia.com/wiki?title=Temagami_greenstone_belt
http://en.wikipedia.com/wiki?title=Volleyball_at_the_2009_Summer_Universiade_Men
http://en.wikipedia.com/wiki?title=Jamey_Scott
http://en.wikipedia.com/wiki?title=Charlie_Waitt
http://en.wikipedia.com/wiki?title=Athletics at the 2009 Mediterranean Games
http://en.wikipedia.com/wiki?title=Michael Giacchino
http://en.wikipedia.com/wiki?title=Temagami greenstone belt
http://en.wikipedia.com/wiki?title=Mosconi Cup
http://en.wikipedia.com/wiki?title=Republic of Salé
Start time: 03:25:37 End Time: 03:37:33
```