Assignment 3

Introduction to Web Science
Dr. Michael Nelson
Fall 2014

Sybil Melton

October 3, 2014

Contents

List of Tables

Listings

1 Question

Download the 1000 URIs from assignment #2. Use a tool to remove (most) of the HTML markup.

1.1 Answer

I decided to use boilerpipe to download the URIs and extract the content. The built-in function ArticleExtractor yielded the best results when I tested it with www.cnn.com. Since it is a Java program, I continued to use Java for the entire program.[1] I took the suggestion of hashing the URI for output file names, to prevent any problems with characters in the URI. Each mapping was written to "mapping.txt" to keep track of the hash to URI translations. Listing 1 is the function used.

```java
private static String getHash(String uri, PrintWriter map) throws IOException{
        //http://www.mkyong.com/java/java-sha-hashing-example/
        StringBuilder sb = new StringBuilder();
        try {
            MessageDigest md = MessageDigest.getInstance("MD5");
            md.update(uri.getBytes());
            byte byteData[] = md.digest();
            for (int i = 0; i < byteData.length; i++) {
                sb.append(Integer.toString((byteData[i] & 0xff) + 0x100, 16).substring(1));
            }
        } catch (NoSuchAlgorithmException ex) {
            Logger.getLogger(getDownload.class.getName()).log(Level.SEVERE, null, ex);
        }
        map.println(uri + "\t" + sb.toString());
        return sb.toString();
    }
```

Listing 1: Hash Function

Using the URI list file name as an argument, the main function opened each URL and sent it to the boilerpipe function, BoilerpipeSAXInput. The extracted text was written to an output file with the hashed URI file name.[2] Listing 2 is the main function.

```java
/*
 * Uses boilerplate to download URL and remove the HTML markup.
 * https://code.google.com/p/boilerpipe/
 * Documentation at
 * http://boilerpipe.googlecode.com/svn/trunk/boilerpipe-core/javadoc/1.0/index.html
 */
/**
 *
 * @author sybil melton
 */
import org.xml.sax.InputSource;

import de.l3s.boilerpipe.document.TextDocument;
import de.l3s.boilerpipe.extractors.ArticleExtractor;
import de.l3s.boilerpipe.sax.BoilerpipeSAXInput;

import java.io.InputStream;
import java.net.URL;
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.util.logging.Level;
import java.util.logging.Logger;

public class getDownload {
    public static void main(final String[] args) throws Exception {

        String inFile = args[0];
        String u = "";
        out = null;
        map = new PrintWriter(new BufferedWriter
                    (new FileWriter("mapping.txt")));
        String outFile = "";
        BufferedReader file = new BufferedReader(new FileReader(inFile));
            String text = "";
            while (file.ready()) {
                text = file.readLine();
                u = "http://" + text;
                //Hash the URI to get the filename it will be output to
                outFile = "./files/"+getHash(text, map);
                System.out.println(text);
                //set URL for boilerplate to fetch
                URL url;
                url = new URL(u);

                final TextDocument doc;
              try{
                InputStream urlStream = url.openStream();
                final InputSource is = new InputSource(urlStream);
                final BoilerpipeSAXInput in = new BoilerpipeSAXInput(is);
                doc = in.getTextDocument();

                //http://www.cs.carleton.edu/faculty/dmusican/cs117s03/iocheat.html
                    try{
                        out = new PrintWriter(new BufferedWriter
                                (new FileWriter(outFile)));
                        out.println(ArticleExtractor.INSTANCE.getText(doc));
                    }catch(FileNotFoundException e){
                        System.out.println(e);
                    }catch(SecurityException e) {
                        System.out.println(e);
                        System.out.println("ERROR: couldn't open URL " + url);
                    }finally {
                        if(out != null){
                            out.close();
                        }
                    }
              }catch(IOException e) {
                   System.out.println("ERROR: couldn't open URL " + url);
                }
        }
        map.close();
    }
    private static PrintWriter out;
    private static PrintWriter map;
}
```

Listing 2: Main Function

2 Question

Choose a query term (e.g., “shadow”) that is not a stop word (see week 5 slides) and not HTML markup from step 1 (e.g., “http”) that matches at least 10 documents (hint: use “grep” on the processed files). If the term is present in more than 10 documents, choose any 10 from your list. (If you do not end up with a list of 10 URIs, you’ve done something wrong).

As per the example in the week 5 slides, compute TFIDF values for the term in each of the 10 documents and create a table with the TF, IDF, and TFIDF values, as well as the corresponding URIs. The URIs will be ranked in decreasing order by TFIDF values.

2.1 Answer

I used Perl as my chosen scripting language to search each file for a query term. The script was run with the search word as an argument. If the term was found, it increased a counter for each occurrence to keep track of the Term Frequency. Additionally, each word was counted in the file, so the TF was calculated in the script to be apart of the result set. The hashed file name was looked up in “mapping.txt” to get the URI it belonged to. The results were printed to the console.[3],[4] Listing 3 is the script used.

```perl
#!/usr/bin/perl
# http://perl.about.com/od/filesystem/a/perl_parse_tabs.htm
use strict;
use warnings;

if (@ARGV) {
        my ($i,$input,$unique);
        $input = $ARGV[0];
        my @files = <files/*>;
        my $mapFile = "mapping.txt";
        foreach my $file (@files) {
                open (IN, "$file") or die "couldn't open input file $file\n";
                $unique=0;
                while (my $line = <IN> ) {
                   chomp $line;
                   my @strings = $line =~ /$input/g;
                   foreach my $s (@strings) {
                         $unique++;
                   }
                }
                if($unique>0){
                        print $unique;
                        open (MAPFILE, "$mapFile") or die "couldn't open input file $mapFile\n";
                        while(<MAPFILE>){
                          chomp;
                          my $f = substr($file , 6, 32); #http://perldoc.perl.org/functions/substr.html
                          (my $uri, my $name) = split("\t");
                          if($name =~ /$f/){
                                print "\t" . $uri . "\n";
                          }
                        }
                close(MAPFILE);
                }
        close(IN);
        }
}
exit;
```

Listing 3: Search Script

I was able to find many files using the search term “music”. I used Google for the DF estimation, which has approximately 42 billion webpages indexed.[5] My search on Google for “music” yielded an estimate of 67.7 million webpages.

The IDF(music) was calculated as:

$$IDF = log_2(42B/67.7M) = 9.277$$

Table 1 lists the URI in decreasing order by TFIDF value.

TFIDF	TF	IDF	URI
0.158	0.017	9.277	www.turlockjournal.com
0.130	0.014	9.277	royradio.net
0.111	0.012	9.277	somafm.com/digitalis/played
0.074	0.008	9.277	gashouseradio.com
0.074	0.008	9.277	www.wfmz.com
0.056	0.006	9.277	www.comicbookmovie.com/fansites/nailbiter111/news
0.027	0.003	9.277	www.lonelyplanet.com/usa/new-york-city/travel-tips-and-articles/76899
0.009	0.001	9.277	thelionwords.wordpress.com
0.009	0.001	9.277	www.cypheredwolf.com
0.009	0.001	9.277	tropicomx.com

Table 1: 10 Hits for the term “music”, ranked by TFIDF.

3 Question

Now rank the same 10 URIs from question #2, but this time by their PageRank. Use any of the free PR estimaters on the web, such as:

http://www.prchecker.info/check_page_rank.php

http://www.seocentro.com/tools/search-engines/pagerank.html

http://www.checkpagerank.net/

Briefly compare and contrast the rankings produced in questions 2 and 3.

3.1 Answer

I used www.prchecker.info to retrieve the PageRanks. Seven of the original ten I chose came back with a rank of zero or NA, so I went through the list of URIs and I ended up with eight with a page rank over zero. Table 2 lists the URI in decreasing order by PageRank value. The table order loosely follows the order in Table 1. For example, the URI with the top TFIDF score is number two in PageRank and the two URIs with a PageRank of zero are in the bottom three in TFIDF values.

PageRank	URI
0.6	www.wfmz.com
0.5	www.turlockjournal.com
0.4	somafm.com/digitalis/played
0.4	www.comicbookmovie.com/fansites/nailbiter111/news
0.3	gashouseradio.com
0.2	www.cypheredwolf.com
0.2	royradio.net
0.1	www.lonelyplanet.com/usa/new-york-city/travel-tips-and-articles/76899
0.0	thelionwords.wordpress.com
0.0	tropicomx.com

Table 2: 10 hits for the term "music", ranked by PageRank.

4 Question

Compute the Kendall Tau_b score for both lists (use "b" because there will likely be tie values in the rankings). Report both the Tau value and the "p" value.

4.1 Answer

Using R, I input the values of TFIDF and Page rank into two separate vectors. Using the package Kendall, it calculated a tau of 0.94 and p of 0.00045812. With a tau number so close to +1, there is a high association between the two measured quantities.[7] In the case of ties, both packages "Kendall and cor produce the same result but cor.test produces a p-value which is not as accurate."[6]

References

[1] Boilerpipe demo classes, Retrieved September 27, 2014 from https://code.google.com/p/boilerpipe/source/browse/#svn/tags/BOILERPIPE_1_0_0/boilerpipe-core/src/demo/de/l3s/boilerpipe/demo

[2] Java Console and File Input/Output Cheat Sheet, Retrieved September 27, 2014 from http://www.cs.carleton.edu/faculty/dmusican/cs117s03/iocheat.html

[3] Parsing Text Files, Kirk Brown, Retrieved September 28, 2014 from http://perl.about.com/od/filesystem/a/per_parse_tabs.htm

[4] How can I search multiple files for a string in Perl?, Sinan Ünür, Retrieved September 28, 2014 from http://stackoverflow.com/questions/1512729/how-can-i-search-multiple-files-for-a-string-in-perl

[5] The size of the World Wide Web (The Internet), Retrieved September 28, 2014 from http://www.worldwidewebsize.com/

[6] A.I. McLeod, Package 'Kendall'. 2011, Retrieved September 28, 2014 from http://cran.r-project.org/web/packages/Kendall/Kendall.pdf

[7] The size of the World Wide Web (The Internet), Retrieved September 28, 2014 from http://en.wikipedia.org/wiki/Kendal_tau_rank_correlation_coefficient#Tau-b