Exercise Sheet 1[1]

Exercise 1

See http://vulcano.informatik.uni-freiburg.de/wiki/teaching/SearchEnginesWS0910/StudentIntros

Exercise 2

- A collection of all RFCs from http://www.rfc-editor.org/download.html has been used. The collection contains 5540 documents (490 MB ASCII text).
- The content of different german news websites has been retrieve with the programm `harvestman`. The websites where:
  - http://www.stern.de/,
  - http://www.derstandard.at/,
  - http://www.sueddeutsche.de/,
  - http://www.focus.de/,
  - http://www.tagesschau.de/,
  - http://www.ftd.de/,
  - http://www.taz.de/,
  - http://www.heute.de/,
  - http://www.welt.de/,
  - http://www.nzz.ch/ and
  - http://www.zeit.de/.

  The collection containts 5362 documents.
- Text files from http://www.textfiles.com/ have been used. The collection contains 48799 documents with (1.1 GB ASCII text).

Only tokens composed of letters have been regarded.

Exercise 3

Uploaded on web page, see there.

Exercise 4

Algorithm: The algorithm is explaned in regard to the current implementation in Java. The implementation does execute a complete search as nothing more precise is stated on the exercise sheet.

As each *two-word with one hit queries* shall be found, look at each pair (i, j) of words from the index, where i is lexicographical greater than j.

```
for(int i = 0; i < index.size(); i++) {

    for(int j = i + 1; j < index.size(); j++) {
```

[1] Version 1.1

```
        IndexElement i1 = index.get(i);
        IndexElement i2 = index.get(j);
        if(!i1.content.equals(i2.content)) {
            Vector<String> hits =
            doItersection(i1.fileNames, i2.fileNames);
            if(hits.size() == 1) {
              /// Pair found
              System.out.println(i1.content + " -- " + i2.content);
            } } } }
```

Calculate the inersection of both lists. The lists are sorted so we can iterate over both list at once and in each step shift one or both iterators until finished with one of both lists.

```
/// Prepare return value
Vector<String> intersection = new Vector<String>();
/// Calculate intersection of the hit lists
Iterator<String> it1 = fileNameList1.iterator();
Iterator<String> it2 = fileNameList2.iterator();
/// Initial values
String s1 = ""; String s2 = "";
/// Compare metric value
int c = s1.compareTo(s2);
/**
* c == 0: s1 eq s2, c < 0: s1 lex_< s2, c > 0: s1 lex_> s2
*
*  @see Java API
*/
while(
(c == 0 && it1.hasNext() && it2.hasNext()) ||
(c < 0 && it1.hasNext()) ||
(c > 0 && it2.hasNext())) {
    if(c == 0) {
        /// Hits are equal
        /// Move the iterators
        s1 = it1.next();
        s2 = it2.next();
    } else if(c < 0) {
        /// s1 lex_< s2
        s1 = it1.next();
    } else {
        /// s1 lex_> s2
        s2 = it2.next();
    }
    c = s1.compareTo(s2);
    if(c == 0) {
        /// Intersection found
        intersection.add(s1);
```

```
        }
    }
```

Analysis: There are $\frac{n\cdot(n-1)}{2}$ ordered pairs (i,j) of words in a set N of n word. Let h_i be the number of files that contain word number i. The list of files H_i, that contain word number i is ordered. In the worst case both lists for a pair of words (i,j) must be iterated in

$$\mathcal{O}\left(h_i + h_j\right) \quad .$$

Therefore the overall complexity of regarding all such pairs is

$$\mathcal{O}\left(\frac{n\cdot(n-1)}{2}\cdot\left(h_i+h_j\right)\right) = \mathcal{O}\left(\sum_{i=1}^{N}\sum_{j=i+1}^{N}\left(h_i+h_j\right)\right) \quad .$$

Exercise 5

The plot of the frequencies is shown below. Multiple data sets have been regared for comparison.

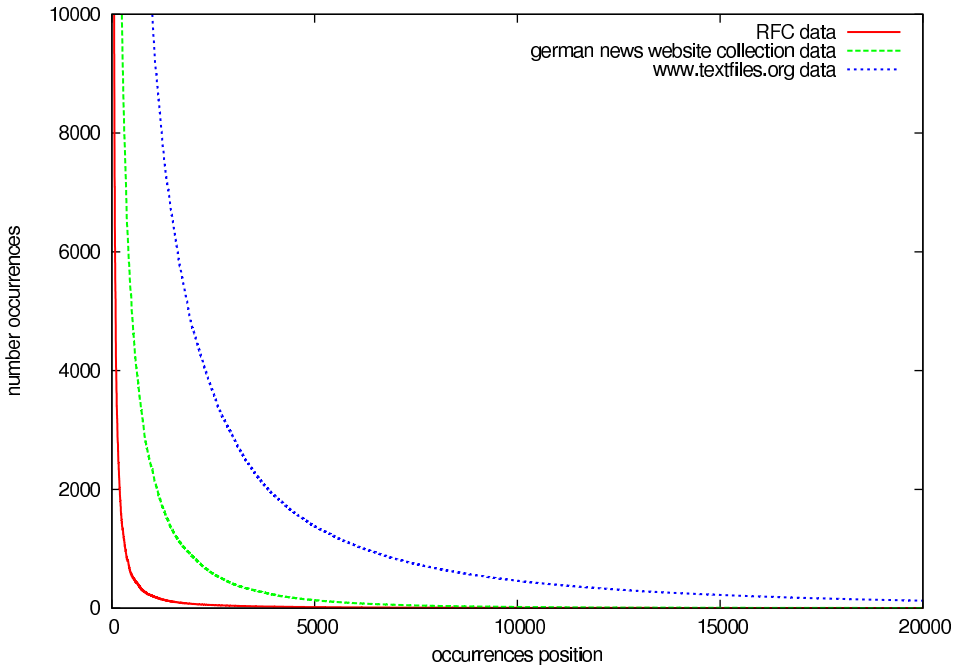

Figure 1: Frequency to position plot

The constant factor ε from Zipf's law has been estimated by least squares fitting for each collections. The error was defined as

$$\text{error}\left(\varepsilon\right) = \sum_{i=1}^{N+1}\left(\underbrace{\varepsilon\cdot N\cdot\frac{1}{i}}_{\text{from Zipf}} - \underbrace{occ\left(i\right)}_{\text{from data set}}\right)^2$$

where N is the number of words occurences in the collection and $occ\left(i\right)$ is the number of the over all occurrences of the $i-$most frequent word $(\forall i : occ\left(i\right) \geq occ\left(i+1\right))$.

Collection	ε
RFC	0.105
news websites	0.076
www.textfiles.com	0.076