Intelligent Data Management - Exercise 1

Name: Triet Ho Anh Doan
Date: October 19, 2017

Assignment 1

a. Calculate the IDF for a word if that word appears in:

- 40 documents

$$IDF = log_2\left(\frac{10^7}{40}\right) \approx 18$$

- 10,000 documents

$$IDF = log_2\left(\frac{10^7}{10^4}\right) \approx 10$$

b. Calculate the TF.IDF if that word appears:

- Once.

$$TF = \frac{1}{15}$$
$$IDF = log_2\left(\frac{10^7}{320}\right)$$
$$TF.IDF = TF \times IDF = \frac{1}{15} \times log_2\left(\frac{10^7}{320}\right) \approx 0.9954$$

- Five times.

$$TF = \frac{5}{15} = \frac{1}{3}$$
$$IDF = log_2\left(\frac{10^7}{320}\right)$$
$$TF.IDF = TF \times IDF = \frac{1}{3} \times log_2\left(\frac{10^7}{320}\right) \approx 4.9772$$

c. $c = 7$

Assignment 2

a. Java implementation

```
1  package de.uni.goettingen;
2
3  import java.util.HashMap;
4  import java.util.List;
5
6  public class WordAnalysis {
7
```

```java
    /**
     * Calculate a Term Frequency in a document
     * @param term The term we want to calculate the frequency
     * @param doc The document which contains the word
     * @return The term frequency
     */
    public double TermFrequency(String term, List<String> doc) {

        // If the document is empty, or the word is not in the document
        if (doc.size() == 0 || !doc.contains(term)) {

            // Simply return 0
            return 0;
        }

        // The document has at least 1 word
        double max = 1;

        // Key: word, value: frequency
        HashMap<String, Integer> map = new HashMap<>();

        for (String word : doc) {

            // Put new word to the map
            if (!map.containsKey(word)) {
                map.put(word, 1);
            } else {

                // If the word is already in the map, increase the count
                int freq = map.get(word);
                freq++;

                // Check if its frequency is max
                if (freq > max) {
                    max = freq;
                }
                map.put(word, freq);
            }
        }

        // Get the input frequency and calculate the TF
        int termFreq = map.get(term);
        return termFreq / max;
    }

    /**
     * Calculate the Invert Document Frequency
     * @param docs All documents we have
     * @param term The term we want to calculate
     * @return The IDF value
     */
    public double InvertDocumentFrequency(List<List<String>> docs, String
     ↪  term) {

        // Number of documents containing the term
        double count = 0;

        // For each document
```

```
65          for (List<String> doc : docs) {
66
67              // For each word in that document
68              for (String word : doc) {
69
70                  // If the document contains the term
71                  if (term.equalsIgnoreCase(word)) {
72                      count++;
73                      break;
74                  }
75              }
76          }
77
78          return Math.log(docs.size() / count) / Math.log(2);
79      }
80  }
```

b. The document of each function is included in the source code. The choice of parameters is based on what we need and to make those functions independent of each other.