

# Intelligent Data Management - Exercise 1

Name: Triet Ho Anh Doan

Date: October 17, 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
```

```

8      /**
9      * Calculate a Term Frequency in a document
10     * @param term The term we want to calculate the frequency
11     * @param doc The document which contains the word
12     * @return The term frequency
13     */
14     public double TermFrequency(String term, List<String> doc) {
15
16         // If the document is empty, or the word is not in the document
17         if (doc.size() == 0 || !doc.contains(term)) {
18
19             // Simply return 0
20             return 0;
21         }
22
23         // The document has at least 1 word
24         double max = 1;
25
26         // Key: word, value: frequency
27         HashMap<String, Integer> map = new HashMap<>();
28
29         for (String word : doc) {
30
31             // Put new word to the map
32             if (!map.containsKey(word)) {
33                 map.put(word, 1);
34             } else {
35
36                 // If the word is already in the map, increase the count
37                 int freq = map.get(word);
38                 freq++;
39
40                 // Check if its frequency is max
41                 if (freq > max) {
42                     max = freq;
43                 }
44                 map.put(word, freq);
45             }
46         }
47
48         // Get the input frequency and calculate the TF
49         int termFreq = map.get(term);
50         return termFreq / max;
51     }
52
53     /**
54     * Calculate the Invert Document Frequency
55     * @param docs All documents we have
56     * @param term The term we want to calculate
57     * @return The IDF value
58     */
59     public double InvertDocumentFrequency(List<List<String>> docs, String
60     ↪ term) {
61
62         // Number of documents containing the term
63         double count = 0;
64
65         // 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);
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.