Intelligent Data Mining - Exercise 1

Michael Debono Mrđen

18 October 2017

1 Assignment 1: Introducing TF.IDF and hash functions

a. (a) 40 documents
IDF
$= log_2(10,000,000/40)$
$= 18$

(b) 10,000 documents
IDF
$= log_2(10,000,000/10,000)$
$= 10$

b. (a) once
$TF_{wd} = 1/15$
$IDF_w = log_2(10,000,000/320)$
$= 15$
$TF.IDF = 1/15 * 15$
$= 1$

(b) five times
$TF_{wd} = 5/15$
$= 1/3$
$TF.IDF = 1/3 * 15$
$= 5$

c. The hash function h will be suitable if c is not equal to the factors of 15, i.e. 3 and 5.

2 Assignment 2: Implementation

a. Main.java:

```
import java.util.HashMap;

public class Main {

        public static double tf(String word, String
            doc) {
                // create new hashmap to store
                    occurrences of each word
                HashMap<String,Integer> map = new
                    HashMap<String,Integer>();
                // store max occurrences
                int max = 1;
                // split the document into words
                String[] words = doc.split("␣");
                // for all words
                for (String w : words) {
                        // if map does not contain
                            word, put it
                        if (!map.containsKey(w)) {
                                map.put(w, 1);
                        } else {
                                // get current
                                    occurrence count
                                int count = map.get(w
                                    );
                                // increase count
                                count++;
                                // store the count
                                map.put(w, count);
                                // update max if
                                    count is larger
                                if (count > max) {
                                        max = count;
                                }
                        }
                }
                int count = 0;
                // if map contains word, return it's
                    occurrence count
                if (map.containsKey(word)) {
                        count = map.get(word);
                }
                // return the term frequency
                return (double)count / (double)max;
        }
```

```
    public static double log2(double x) {
            // change base formula
            return (Math.log(x) / Math.log(2));
    }

    public static double idf(String word, String
        docs[]) {
            // count the number of documents
                where word occurs
            int count = 0;
            // for all documents
            for (String doc : docs) {
                    // split the document into
                        words
                    String[] words = doc.split("
                        ");
                    // for all words in document
                    for (String w : words) {
                            // if word is found
                            if (w.equals(word)) {
                                    // increase
                                        count
                                    count++;
                                    // don't keep
                                        counting
                                        for the
                                        same
                                        document
                                    break;
                            }
                    }
            }
            // return inverse document frequency
            return log2((double)docs.length / (
                double)count);
    }

}
```

b. For the implementation, documents are considered to be a String and a set of documents an array of String.

   The method Main.tf() takes a word and a document, both as String's, as the TF function is the frequency of a term or word divided by the maximum frequency of all terms in one document.

   The method Main.idf() takes a word as String and a set of documents as

an array of String. The IDF function is based on the number of documents in which a term or word appears, which explains the choice of inputs.