# Information Processing and Retrieval

Instituto Superior Técnico 2020

# Lab 2: Information Retrieval Models

The main goal of this lab is to implement a simple, memory-based, version of the Boolean and Vector Space Models for Information Retrieval.

Your application should take as input a folder containing several text documents, index them and, once indexed, the application should support simple queries using `stdin`. Query results using Boolean and Vector Space Models should be presented in the `stdout`. Considering the Vector Space Model, the retrieved documents should be further subjected to ranking.

## 1 Indexing

**Recover** the homework exercises **4.1** and **4.2** from previous lab. In this exercise, we implemented a function to read text documents from a disk directory and create a simple, in-memory, non-optimized *inverted index* for the given collection. The inverted index is essentially a dictionary (hash structure) that contains, for each term, the number of documents where it occurs and the corresponding list of document identifiers.

### 1.1

Extend the developed inverted index:

(a) the postings of our inverted index should not only contain the document identifier but also the number of times the selected term occurs in the document. To this end, and using Python, each entry in the dictionary should point to a list of tuples *(doc ID, #occurrences)*;

(b) in addition to the inverted index, consider maintaining a auxiliary map *from* document identifiers *to* pairs *(#tokens, #terms)*. Consider using a dictionary of tuples in python to this end;

### 1.2

Let us visualize simple collection statistics. Using the auxiliary map containing basic document statistics, draw an histogram with the distribution of terms per document.

```
import matplotlib.pyplot as plt
x = [value1, value2, value3,....]
plt.hist(x, bins = number of bins)
plt.show()
```

### 1.3

Program a function that, given a term $t$, prints its inverse Document Frequency, $IDF_t$, according to the formula $\log(N/DF_t)$, where $DF_t$ is the document frequency of term $t$ and $N$ is the total number of documents.

Extend the dictionary in our inverted index to, in addition to the document frequency, further store the corresponding IDF score.

## 2 Boolean model

Create an IR system that receives a query and returns the matched documents.

### 2.1

Program a function that receives a single term in the command line as arguments and returns a list of documents where the inputted term occurs.

**Note**: to read command line arguments in python, you can use sys.argv list from sys module.

### 2.2

Extend your program to receive a query of multiple terms and return a list of documents that contain all of the inputted terms.

**Homework**: after accomplishing the remaining exercises of this lab, consider returning to this question in order to make use of the principles for efficient intersection of posting lists introduced in lectures. First, select less frequent terms. Second, intersect them in linear time.

## 3 Vector space model

Program a function that takes as input a list of terms and computes the *dot product* similarity between the query formed by those terms and each indexed document. The function should return a list of pairs (document id, similarity).

The following pseudo-code shows how this can be implemented efficiently.

```
• Set $A \leftarrow \{\}$
• For each query term $t \in Q$
    – Set $I_t \leftarrow$ the inverted list of $t$
    – Set $idf_t \leftarrow \log(N/DF_t)$
    – For each $(d, TF_{d,t})$ pair in $I_t$
        * If $A_d \notin A$ then
            · Set $A_d \leftarrow 0$
        · Set $A \leftarrow A \cup \{A_d\}$
      * Set $A_d \leftarrow A_d + TF_{d,t} \times idf_t$
• Return $A$, where each $A_d$ contains the similarity between the query and
  document $d$.
```

## 4 Pen and paper exercise (*homework*)

Consider the following collection of 4 text documents.

| number | text document |
|---|---:|
| 1 | shipment of gold damaged in fire |
| 2 | delivery of silver arrived in silver truck |
| 3 | shipment of silver arrived in truck |
| 4 | truck damaged in fire |

Calculate the representations for the documents in the collection according to the Vector Space Model, using TF-IDF weights for the different terms. The TF-IDF score of a term $t$ in a document $d$ from a collection $D$ can be computed through the following equation:

$$\text{TF-IDF(t, d) = frequency(t,d)} \times \log\left(\frac{|D|}{|\{d' \in D : t \in d'\}|}\right)$$

According to the vector space model and using the document representations (i.e., computing the cosine similarity towards the document vectors), find which document in the collection is the most relevant to the following query: `silver truck`.

**Note**: For simplification, the TF-IDF formula above is not considering the normalization of the TF component with basis on the occurrence frequency of the most frequent term in the document, as shown in the corresponding lecture.