IST 2018/2019

Processamento e Recuperação de Informação

Lab 02: Information Retrieval Models

The main goal of this lab is to **implement a simple, memory-based, version of the Vector Space Model for Information Retrieval**, using the Python language.

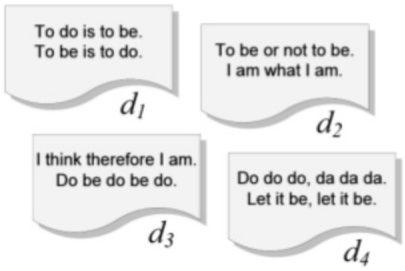
Your application should take as input a file containing several textual documents, one document per line. The documents should be read from disc and indexed into memory. Once the documents are stored, the application should read a set of keywords from *stdin* and use it as a query over the indexed documents. The documents should be presented ordered according to their similarity to the query.

1

Implement a function that reads a file from disc (as described above) and creates an in-memory *inverted index* of its contents. The efficient construction of disk-based inverted indexes will be covered later on the PRI course and, for now, you should implement a simple in-memory data structure.

The inverted index essentially corresponds to a dictionary that contains, for each term, the documents where it occurs and the corresponding document frequency. The following figure schematically illustrates an inverted index created over a collection of four documents. The figure shows each term, from the vocabulary used in the collection, in association to the documents where the term occurs, and together with the occurrence frequency.

Vocabulary	n_i	Occurrences as inverted lists
to	2	[1,4],[2,2]
do	3	[1,2],[3,3],[4,3]
is	1	[1,2]
be	4	[1,2],[2,2],[3,2],[4,2]
or	1	[2,1]
not	1	[2,1]
I	2	[2,2],[3,2]
am	2	[2,2],[3,1]
what	1	[2,1]
think	1	[3,1]
therefore	1	[3,1]
da	1	[4,3]
let	1	[4,2]
it	1	[4,2]



When reading the file from disk, assume that the terms in each document are separated by any sequence of non-alphanumeric characters. Each document read should be assigned a unique identifier.

2

Using the above function, implement a script to print some statistics on the documents indexed, namely:

- The total number of documents;
- The total number of terms;
- The total number of individual terms;

The script should also take a set of terms as command line arguments and, for each term, print the following statistics:

- Document frequency (DF);
- Maximum and minimum term frequency (TF);
- Inverse Document Frequency(IDF), according to the formula $\log(N/DF)$, where N is the total number of documents.

If needed, change your indexing function so that these values can be easily obtained (e.g., you can consider indexing the vocabulary used in the document collection, associating each term to the corresponding IDF score, separately from the list that contains the documents where the term occurs, together with the corresponding frequency).

> **Notes:**
>
> To read command line arguments in Python, you should use the `sys.argv` list, from the `sys` module.
>
> To find the minimum and maximum, try the `min` and `max` functions.

3

Implement 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

With the above functions, create an application that indexes a set of documents, reads user queries from *stdin*, and produces a list of documents (or document ids) ordered by their similarity to the query.

Note 1: You can use function `raw_input` to read data from stdin.

Note 2: You can test your application with the "Time" IR test collection, available from `http://ir.dcs.gla.ac.uk/resources/test_collections/time/`

5 Pen and Paper Exercise

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:

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

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 1: 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.

Note 2: After completing the exercise, considering the same user query, you can also experiment with the application of other retrieval models:

- The simplest version of the Binary Independence Model (BIM), considering the blind assumptions for $P(k_i|R)$ and $P(k_i|\overline{R})$.
- The Okapi BM25 model, considering a simple estimate for the IDF component, and with $b = 0.75$ and $k_1 = 1.2$.
- A simple unigram language modeling approach, without considering parameter smoothing.

The corresponding equations are given below.

$$\text{BIM}(q,d) = \sum_{i=1}^{|q|} \text{IDF}(i), \quad \text{with} \quad \text{IDF}(i) = \log\left(\frac{|D|}{|\{d' \in D : i \in d'\}|}\right)$$

$$\text{BM25}(q,d) = \sum_{i=1}^{|q|} \text{IDF}(q_i) \cdot \frac{f(q_i,d) \cdot (k_1+1)}{f(q_i,d) + k_1 \cdot \left(1 - b + b \cdot \frac{|d|}{\text{avgdl}}\right)}, \text{ with IDF as defined above}$$

$$\text{LM}(q,d) = P(\{t_1,\ldots,t_{|q|}\}|d) = \prod_{1<=i<=|q|} P(t_i|d) = \prod_{1<=i<=|q|} \frac{f(t_i,d)}{|d|}$$