TP1- Text Retrieval

Posted: Feb 22 – Deadline: Mar 7

1 Goal

In this exercise, we introduce two basic models in information retrieval namely, the *boolean* and *vector models*. Given tasks should help you understand better how to find a discriminative set of *index terms* (keywords) for collection of documents. We study how to represent each document and query as a vector. In order to do this, we perform stemming by applying the *Porter algorithm* and visualize frequency of words using the *tag clouds* over the stems already obtained.
Next, we extract the most informative words from documents based on the *term frequency-inverse documents frequency* (tf-idf) weighting scheme. Then, we complete the *boolean* and *vector models* to predict that the certain documents are relevant to the particular query.

2 Introduction

2.1 Useful terminologies

Terms a set of words on which the vector representation is based. They are also referred to *index* or *index set*. For example, the terms are "computer", "software" and "fishing".

Term weight is a scalar parameter that represents the significance of a given term in a given document.

Term-document matrix a matrix consisting of all vector representations of the document in a corpus. By convention, rows correspond to terms and columns correspond to documents.

Document ranking is ranking of documents based on the similarity to a certain query.

Frequency is number of occurrences of a word in a text.

Corpus is a collection of documents.

Rank of a word is a word's ordinal number in a list sorted by decreasing frequency.

Vocabulary is a set of all unique words in a corpus.

2.2 Stemming

A *stem* is the part of a word to which affixes (suffixes and prefixes) can be attached to form new words. A few examples of words that have the same stem are given below
- 'house', 'houses', 'housing'
- 'set', 'sets', 'reset', 'setting', 'setters'
- 'tall', 'taller', 'tallest'
- 'steam', 'steamers', 'steamier', 'steaming'

Stemming is the process of extracting for each given word its corresponding stem. Several popular implementations for the English language can be found at `http://tartarus.org/~martin/PorterStemmer/`.

2.3 Stop words

Stop words are those words which are judged to be so common that they have very little information or discrimination power. For the French language, de, et, la, que, vous, etc. are examples of *stop words*. Such words are usually useless for information retrieval. Thus, in order to enhance system performance, stop words can be left out of consideration. In online systems they are not indexed, and therefore are not searchable. A list of English *stop words* is given in the file `english.stop`.

2.4 Tag Clouds

Tag clouds are visual representation of words (tags) in documents (websites). Tags will be represented by different font size based on the importance of each word (frequency). More important words appear larger in the visual representation. Generally tags are listed in alphabetical order. The importance of a word is given by its popularity (frequency). To compute the font size s_i for a word i, we will use the formula

$$s_i = f_{max} \cdot \frac{t_i - t_{min}}{t_{max} - t_{min}}, \tag{1}$$

where f_{max} is the maximum font size, t_i is the word count, t_{min} is the minimum count, and t_{max} is the maximum count.

2.5 Term Frequency-Inverse Documents Frequency

The *tf-idf* is a weight used in information retrieval and text mining. This weight is a statistical measure used to evaluate how important a word is for a document in a collection or corpus. The importance increases proportionally to the number of times a word appears in the document but is offset by the frequency of the word in the corpus. Variations of the *tf-idf* weighting scheme are often used by search engines as a central tool in scoring and ranking a documents relevance given a user query.
The *term frequency* tf_{ij} is the weight of a term t_i in a document d_j computed according to

$$tf_{ij} = \frac{freq_{ij}}{\max\limits_k freq_{kj}}, \tag{2}$$

where $freq_{ij}$ is the number of occurrences of the term t_i in the document d_j. As a result, the most frequent term $t_{\bar{i}}$ in a document d_j always has $tf_{\bar{i}j} = 1$.
The *inverse document frequency* idf_i is a measure of the general importance of the term t_i in the whole corpus. It is obtained by dividing the number of all documents by the number of documents containing the term t_i, and then taking the logarithm of that quotient as

$$idf_i = \log \frac{N}{n_i}, \tag{3}$$

where N is the total number of documents, and n_i is the number of documents in which the term t_i appears.
Using the *term frequency* tf_{ij} and the *inverse document frequency* idf_i as described above, we obtain the most used term-weighting scheme in text retrieval defned as

$$w_{ij} = tf_{ij} \cdot idf_i. \tag{4}$$

A high weight w_{ij} is reached by a high frequency of the term t_i in the document d_j and a low frequency of the term t_i in the whole collection of documents. Hence, the weights will tend to filter out common terms that appear in many documents in the collection.

2.6 Boolean and Vector model

The *boolean model* is a simple retrieval model based set theory on *boolean algebra.* The concept of a set is intuitive, *index terms* and queries are specified as *boolean* expressions. The *boolean* model provides easy tools that are easy to be grasped by a common user of an information retrieval system.
The *vector model* represents documents and queries as vectors. The dimensionality of each vector is fixed, so that we can compare similarity of each document d_j and each query q_l based on the *cosine of the angle* between two vectors that represent them. It is expressed according to following rules

$$sim(d_j, q_l) = \frac{\vec{d_j} \circ \vec{q_l}}{||\vec{d_j}|| \cdot ||\vec{q_l}||}. \tag{5}$$

Each dimension corresponds to a unique term chosen in advance. Its value describes the significance of the term with respect the document. In the simplest case, it can be the number of times the term occurs in the document. The *vector model* disregards the order in which the terms appear in a document.

3 List of tasks

1. Use e.g. 15 articles from `NASA` corpus to obtain raw data (after the *tokenisation*).
2. Perform stemming using *Porter algorithm* and visualize frequency of words using the *tag clouds* for 50 most frequent words.
3. Compute *term frequency* tf_{ij} and *tf-idf* w_{ij} for each document. First, choose and compare the top p stems according to *term frequency* or *tf-idf* for each document. Then, compare the extracted keywords among all documents. Explain differences based on keyword criteria.
4 Build *boolean* and *vector models* based on top p stems, then provide k queries to each IR system. Compare rankings of relevant articles.
5. First, remove *stop words*, then perform stemming and create the *tag clouds* again but for 50 most frequent words.
6.Compute again *term frequency* tf_{ij} and *tf-idf* w_{ij} for each document. First, choose and compare the top p stems according to *term frequency* or *tf-idf* for each document. Then, compare the extracted keywords among all documents. Explain differences based on keyword criteria.
Based on these, build new *boolean* and *vector models.* Then provide k the same queries to each IR system. Compare current rankings of relevant articles with obtained before.
7.Provides final remarks and conclusions.

4 Software requirements

4.1 Implement Required functions

`createTermDocumentMatrix.m` reads all `.txt` from a specifed directory (where your corpus is), creates a boolean or a vector representation for each document, and puts them together into a term-document matrix.

`compQueryBoolean.m` compares the boolean representation of a query to the documents in the term-document matrix, and prints out similarities and filenames of the top N documents.

`compQueryVector.m` compares the vector representation of a query to the documents in the term-document matrix based on the cosine similarity, and prints out similarities and filenames of the top-N documents.

`similarityMeasure.m` nested in `compQueryVector.m`, that measures the cosine similarity between the query and documents vector representation in collections, and returns a similarity measure of the top N documents.

`queryBooleanRepresentation.m` returns a boolean representation of a query.

`queryVertorRepresentation.m` returns a vector representation of a query.

5 Assessment

The assessment is based on your report. It should include all experimental results, your answers to all questions, and your analysis and comments of the experimental results. It should be around 5 pages for this assignment. Please try to detail the report by giving examples and conclusions. Please archive your report and codes in "PrenomNomTP1.zip", and upload to `http://moodle.unige.ch` in the section for TP1 before Sunday, March 7, 2021, 23:59PM. Later submission will not be accepted.

6 Resources

6.1 Corpus

`NASA` collection covers 141 short articles in `nasa.tar.gz` file.