TP2- Text Retrieval

Pablo Strasser
e-mail: Pablo.Strasser@unige.ch
Exercice based on Michal Muszynski.

March 11, 2019

1 Goal

In this exercise, we introduce two other basic models in information retrieval namely, the *probabilistic* and *latent semantic indexing models.* Given tasks should help you understand better how to find a discriminative set of *index terms* (keywords) for collection of documents and queries. We study how to summarize the content of each document. 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 *probabilistic* and *latent semantic indexing 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 Probabilistic model

The *probabilistic model* is a simple retrieval model based on probabilistic framework. The idea is as follows. We consider a user query and a set of documents that contains only relevant documents for this query. We can think of this set as the ideal answer set. If there were given descriptors of ideal answer set, we would not have a problem with retrieving its documents. Querying process specifies the properties of an ideal answer set. In most cases, our problem is that we do not know exactly what these properties are. We know only that there are index terms whose semantic properties should be used to describe these properties. At the beginning, these properties are not known for particular query. We make an effort to guess what they could be. The initial values allows us to generate a preliminary probabilistic description of the ideal answer set.

We assume that there is a given user query q and a document d_j in the data base. We attempt to estimate the probability that the user will find the document d_j as interesting. The fundamental assumption is that this probability of relevance depends on the query and the document representations only. Moreover, there is a ideal subset of all documents which the user chooses as the answer for this query.

For the *probabilistic model*, the index term weight variables are all binary. A query q is a subset of index terms. Let R be the set of initially guessed documents to be relevant. Let $\bar{R}$ be the complement set of R. It means that $\bar{R}$ contains all non-relevant documents in the collection. Let $P(R|d_j)$ be the probability that the document d_j is relevant to the query q and $P(\bar{R}|d_j)$ be the probability that the document d_j is non-relevant to the query q. The similarity measure $sim(d_j, q)$ between the document d_j and the query q is defined as the radio

$$sim(d_j, q) = \frac{P(R|d_j)}{P(\bar{R}|d_j)}. \tag{5}$$

Applying *Bayes' rule*, we get the following expression

$$sim(d_j, q) = \frac{P(d_j|R) \cdot P(R)}{P(d_j|\bar{R}) \cdot P(\bar{R})}, \tag{6}$$

where $P(d_j|R)$ corresponds to the probability of randomly selected the document d_j from the set R and $P(R)$ corresponds to the probability that a selected document is relevant. $P(d_j|\bar{R})$ and $P(\bar{R})$ are analogous and complementary. Assuming independence of *index terms* and taking logarithms, we can finally write

$$sim(d_j, q) = \sum_{i=1}^{k} w_{iq} \cdot w_{ij} \cdot \left(\log \frac{P(t_i|R)}{1 - P(t_i|R)} + \log \frac{1 - P(t_i|\bar{R})}{P(t_i|\bar{R})} \right), \tag{7}$$

where $P(t_i|R)$ corresponds to the probability that *index term* t_i is present in a document randomly selected from the set R. The probabilities associated with the set $\bar{R}$ have meanings that are analogues to the ones just described.

Since we do not know the set R at the beginning, it is necessary to choose a method for initially computing the probabilities $P(t_i|R)$ and $P(t_i|\bar{R})$. Now, we discuss some alternative approaches to computing them.

We only consider one method in which we assume initially that $P(t_i|R)$ is constant for all index terms, e.g. equal 0.5 and the distribution of *index terms* among non-relevant documents can be approximated by the distribution of index terms among all documents in the collection. These two assumptions provide formulas

$$P(t_i|R) = 0.5, P(t_i|\bar{R}) = \frac{n_i}{N}, \tag{8}$$

where N is the total number of documents, and n_i is the number of documents in which the term t_i appears. Based on it, we retrieve initial probabilistic *ranking* that is improved as follows.

Let V be a subset of the documents initially retrieved and ranked by the probabilistic model, for instance, the top r ranked documents where r is a certain *threshold*. Next, let V_i be the subset of V composes of the documents in V which contain the *index term* t_i.

Further, we can improve recursively our initial guesses values of $P(t_i|R)$ and $P(t_i|\bar{R})$. This can be accomplished with satisfying the following assumptions. We can approximate $P(t_i|R)$ by the distribution of index term t_i among the documents retrieved so far, and we can approximate $P(t_i|\bar{R})$ by considering that all the non-retrieved documents are not relevant. With theses assumptions, we can express updating rules as follows

$$P(t_i|R) := \frac{|V_i|}{|V|}, P(t_i|\bar{R}) := \frac{n_i - |V_i|}{N - |V|}. \tag{9}$$

The last formulas for $P(t_i|R)$ and $P(t_i|\bar{R})$ have a problem for small values of V and V_i which appears in practice. To solve it, we can add some adjustment factor which obtains

$$P(t_i|R) := \frac{|V_i| + 0.5}{|V| + 1}, P(t_i|\bar{R}) := \frac{n_i - |V_i| + 0.5}{N - |V| + 1}. \tag{10}$$

An alternative is to take the fraction $\frac{n_i}{N}$ as the adjustment factor which yields

$$P(t_i|R) := \frac{|V_i| + \frac{n_i}{N}}{|V| + 1}, P(t_i|\bar{R}) := \frac{n_i - |V_i| + \frac{n_i}{N}}{N - |V| + 1}, \tag{11}$$

where $| \cdot |$ means a number of set elements.

2.7 Latent semantic indexing model

The idea of *latent semantic indexing model* is that the process of matching documents to a given query could be done based on concept matching instead of index term matching. This would allow us to retrieve the documents even when they are not indexed by query index terms. For example, a document could be retrieved because it shares concepts with another document which is relevant to the given query.*Latent semantic indexing model* addresses these issues. This approach maps each documents and query vector into a lower dimensional space which is associated with concepts. The reduce space may be better to retrieve information than the original one.
Let k be the number of index terms in the collection of documents and N be the the total number of documents. Define M as a matrix with k rows and N columns. Each element M_{ij} of this matrix is assigned a weight w_{ij} associated with the index term t_i and the document d_j. The weight w_{ij} could be generated using *tf-idf* weighting technique.
Latent semantic indexing model proposes to decompose the *term-document* matrix M using *singular value decomposition* as follows

$$M = S \cdot \Delta \cdot D^T. \tag{12}$$

The matrix S is the matrix of eigenvectors obtained from $M \cdot M^T$.The matrix D is the matrix of eigenvectors derived from $M^T \cdot M$. The matrix Δ is an $r \times r$ diagonal matrix of singular values where $r = min(k, N)$ is the rank of M.
Consider now only the l largest singular values of Δ and keep with their corresponding columns in S and D, respectively. The result is matrix M_l given by

$$M_l = S_l \cdot \Delta_l \cdot D_l^T, \tag{13}$$

where l, $l < r$, is the dimensionality of reduced concept space. The selection of a value for l tries to balance two opposing effects. It means that l should be large enough to fit all the structure in the real data. On the other hand, it should be small enough to filter out non-relevant details of data.
The relation between any two documents in the reduced space of dimensionality l can

be derived from the matrix $M_l^T \cdot M_l$, where element (i, j) quantifies the relationship between documents d_i and d_j.
To rank documents with regard to a given user query, we simply our model in which the query is considered as a document in the original matrix M.

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.*
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.
4 Build *probabilistic* (test different updating rules) and *latent semantic indexing model* based on top p stems, then provide s queries to each IR system. Compare rankings of relevant articles.
5. First, remove *stop words*, then perform stemming.
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. Based on these, build new *probabilistic* (test different updating rules) and *latent semantic indexing model.* Then, provide s the same queries to each IR system. Compare current rankings of relevant articles with obtained before.
7.Provides final remarks and conclusions, compare current results with *boolean* and *vector models* (TP1).

4 Software requirements

4.1 Implement Required functions

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

2. Reads all `.txt` from a specifed directory (where your corpus is), creates a latent semantic indexing representation for all documents and queries, and puts them together into a term-document matrix.

3. Compares similarity between a given query and the documents, and returns similarity values and filenames of the top N relevant documents.

4. Compares similarity between a given query and the documents, and returns similarity values and filenames of the top N relevant documents. (A QUERY IS TREATED AS A SHORT DOCUMENT IN TERM-DOCUMENT MATRIX)

5. Returns a boolean representation of a query for probabilistic model.

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` under TP2 before Monday, March 25, 2019.

6 Resources

6.1 Corpus

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