IST 2017

Processamento e Recuperação de Informação

Course Project: Part 1

An important task within the areas of information extraction and retrieval is **automatic document summarization**, which concerns the shortening of text document(s) through an automated method, in order to create a summary with the major points of the original document(s).

Most systems follow an extractive approach, in which a model is used to select the most relevant sentences from the original document(s), and these are then presented, in the order by which they originally appear, as the resulting summary. The fundamental difficulty lies thus in determining which are the sentences that capture the main ideas behind the document(s).

The course project for *Information Processing and Retrieval* will explore different alternatives for addressing the task of **automatic single-document extractive summarization**.

General instructions

For each of the following four exercises, you should develop a Python program (i.e., create a file named `exercise-number.py` with the necessary instructions) that addresses the described challenges. When delivering the project, you should present a `.zip` file with all four Python programs, together with a PDF document (e.g., created in LaTeX) describing your approaches to address each of the exercises.

The PDF file should have a maximum of two pages and, after the electronic project delivery at Fénix, you should also deliver a printed copy of the source code plus the project report (e.g., using two-sided printing and two pages per sheet).

1 Simple approach based on TF-IDF

As a first exercise, you should implement an approach for selecting the most relevant sentences for forming a summary, based on TF-IDF vectors.

- Segment the document into its constituent sentences, and represent each sentence according to the vector space model for information retrieval, considering TF-IDF weights for index terms corresponding to individual words. Use a normalized TF score, and a logarithmically scaled IDF score, as discussed in the classes;
- Represent the textual document to summarize according to the same vector space model;
- Score each sentence against the entire document, according to the cosine similarity between the vectors (i.e., the sentence vector versus the document vector);
- Select the top-3 highest scoring sentences in order to form the summary (i.e., the sentences that are more similar to the entire contents of the document);

- Return the summary to the user, by presenting the 3 sentences according to the order by which they appear in the original document.

You program should apply the aforementioned summarization method to an English textual document of your choice, reading it from a text file stored on the hard drive.

Notice that this exercise focuses on the representation of sentences (i.e., the individual sentences can be seen as equivalent to the documents that would be returned in a traditional retrieval system). When estimating the IDF scores, you should consider sentences instead of documents, and thus compute the *inverse sentence frequency*. Notice also that you can use Python libraries such as `scikit-learn` or `nltk` in the development of your program, although normalized and logarithmically scaled TF-IDF scores should still be used.

2 Evaluating the simple approach

The previous summarization method should now be applied to the Portuguese documents in the TeMário dataset available at `http://www.linguateca.pt/Repositorio/TeMario/`.

You should compare the procedure from the previous exercise, against a simple alternative in which the IDF scores are estimated with basis on the entire collection of documents in the aforementioned dataset. Taking the source texts with the titles, you should generate summaries containing the 5 most relevant sentences, and afterwards check if these sentences also appear in the ideal extractive summaries that are provided.

Leveraging the aforementioned ground-truth summaries, your program should print the precision, recall, and F1 scores achieved by the two alternatives for the simple extractive summarization method (i.e., the overall evaluation scores, produced over all documents in the corpus). Your program should also print the Mean Average Precision (MAP) score.

3 Improving the simple approach

Write a Python program that improves on the extractive summarization method from Exercise 1, considering the following two aspects, **as well as other aspects that you deem to be relevant** (i.e., the evaluation for this exercise will value creative solutions, proposed to improve the results for automated summarization). Compare your improved version, against the alternative from the previous exercise, using the TeMário corpus.

3.1 Representing sentences in the vector space model

The vector space representations of the sentences should now consider longer word n-grams (i.e., bi-grams, where $1 \leq n \leq 2$), instead of just individual words. Besides words and bi-grams, the representations should also consider noun phrases, i.e. sequences of words matching a parts-of-speech regular expression pattern like $\{$`(<JJ>* <NN.*>+ <IN>)? <JJ>* <NN.*>+`$\}$+.

Parts-of-speech tags for the words in the textual documents can, for instance, be obtained through the Portuguese resources associated to the `nltk` package[1].

3.2 Scoring sentences

The phase concerned with scoring sentences according to their relevance should now consider the BM25 term weighting heuristic, instead of TF-IDF. Recall the BM25 term weighting formula:

$$\text{score}(D,t) = \text{IDF}(t) \cdot \frac{f(t,D)\cdot(k_1+1)}{f(t,D)+k_1\cdot\left(1-b+b\cdot\frac{|D|}{\text{avgdl}}\right)}$$

$$\text{IDF}(t) = \log\frac{N-n(t)+0.5}{n(t)+0.5}$$

In the previous equations, $f(t,D)$ is the frequency for element t in document D (or instead in sentence D, when representing sentences), $|D|$ is the length of D in terms of the number of words, and $avgdl$ is the average document length in terms of the number of words (or instead the sentence length when representing a collection of sentences) in a background text collection.

The parameters k_1 and b are free parameters, usually set to $k_1 = 1.2$ and $b = 0.75$.

In the IDF formula, N is the total number of documents (or instead the number of sentences, when representing a collection of sentences) in a background collection, and $n(t)$ is the number of elements, from this background, containing the term t.

4 A more sophisticated approach

In a paper entitled *the use of MMR, diversity-based reranking for reordering documents and producing summaries*[2], Carbonell and Goldstein proposed Maximal Marginal Relevance (MRR) as a measure to increase the diversity of documents returned by an information retrieval system, and then described a summarization method based on MMR.

The proposed method iteratively (i.e., through a greedy procedure that selects one sentence at a time) constructs the resulting set of sentences S by selecting the ones that maximize the following objective function, which simultaneously attempts to select sentences that are relevant (i.e., that are similar to the entire document) and diverse (i.e., that are dissimilar to the sentences that have been selected so far, and hence are non-redundant).

$$\text{MMR}(s) = (1-\lambda)\times\text{sim}(s,d) - \lambda\sum_{v\in S}\text{sim}(s,v) \tag{1}$$

In the previous equation, d corresponds to the document representation, s represents the sentence being evaluated, and S represents the set of sentences selected so far. The function $\text{sim}(s_1,s_2)$ represents the cosine similarity between representations s_1 and s_2.

[1] `http://www.nltk.org/howto/portuguese_en.html`

[2] `https://dl.acm.org/citation.cfm?id=291025`

Leveraging the aforementioned idea, your program should build document summaries by taking the top 5 sentences. You may tune the λ parameter for optimal performance, and you may choose to use either TF-IDF or BM25 term weights, in the similarity measure used within MRR.

You should also evaluate the new method on the document collection from Exercise 2, comparing it against a simple procedure that just selects the first 5 sentences that occur in each document (i.e., in news articles, the first sentences are usually the most informative).