IST 2017

Processamento e Recuperação de Informação

Course Project: Part 2

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

The second part of the course project for *Information Processing and Retrieval* will continue to explore different alternatives for addressing **extractive summarization**.

General instructions

For each exercise, 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 An approach based on graph ranking

Several previous studies (e.g., TextRank[1]) have proposed to leverage graph centrality metrics (e.g., approaches similar to PageRank) to address extractive document summarization problems. These methods are based on representing documents as a graph, where nodes correspond to sentences, and where edges encode similarity between sentences (e.g., an edge between two nodes encodes the fact that the associated sentences have a similarity above a particular threshold).

In this first exercise, you should develop a program that uses a PageRank-based method for performing extractive summarization, taking the following considerations into account:

- You should start by creating a graph where nodes correspond to sentences from the document being processed, and where edges encode the fact that a pair of lower-cased sentences has a cosine similarity above the threshold value of 0.2 (we suggest you experiment also with different thresholds for this particular parameter);
- Sentences should then be ranked according to a variation of the PageRank algorithm for undirected graphs, which computes a score for each sentence according to an iterative procedure based on the following equation:

[1] `http://web.eecs.umich.edu/~mihalcea/papers/mihalcea.emnlp04.pdf`

$$\mathrm{PR}(p_i) = \frac{d}{N} + (1-d) \times \sum_{p_j \in \mathrm{Links}(p_i)} \frac{\mathrm{PR}(p_j)}{\mathrm{Links}(p_j)}$$

In the equation, $p_1, \ldots, p_N$ are the sentences under consideration, $\mathrm{Links}(p_i)$ is the set of sentences that are related to p_i (i.e., the set of nodes linked to p_i in the graph), and N is the total number of sentences. Notice that the PageRank definition considers a uniform residual probability for each node, usually set to $d = 0.15$.

- The iterative procedure should be applied up to a maximum um 50 iterations, and finally the top-5 scoring sentences should be returned as the summary.

You program should apply the summarization method to a textual document of your choice, reading it from a text file stored on the hard drive. The summary should be returned by presenting the selected sentences, according to the order by which they originally appear in the document.

2 Improving the graph-ranking method

The PageRank procedure from the previous exercise can be extended in order to consider a non-uniform prior probability for each sentence, and also in order to consider weighted edges in the graph, indicating the degree of association between the sentences. In this case, PageRank can be computed through the following iterative procedure:

$$\mathrm{PR}(p_i) = d \times \frac{\mathrm{Prior}(p_i)}{\sum_{p_j} \mathrm{Prior}(p_j)} + (1-d) \times \sum_{p_j \in \mathrm{Links}(p_i)} \frac{\mathrm{PR}(p_j) \times \mathrm{Weight}(p_j, p_i)}{\sum_{p_k \in \mathrm{Links}(p_j)} \mathrm{Weight}(p_j, p_k)}$$

In the equation, $\mathrm{Prior}(p_i)$ corresponds to a prior probability for node p_i, and $\mathrm{Weight}(p_i, p_j)$ corresponds to the weight of the edge between nodes p_i and p_j.

For this exercise, you should develop a program for comparing different alternatives of the PageRank approach for document summarization, evaluating them through the mean average precision over the TeMário[2] dataset that was also used in the first part of the project. Variants that you can consider include, for example:

- Non-uniform prior weights with basis on the position of the sentence in the document (e.g., first sentence, second sentence, etc.), under the intuition that sentences in the first positions are often more descriptive of the entire contents of the document;
- Non-uniform prior weights based on the cosine similarity towards the entire document, leverating either TF-IDF or BM25 weights when representing sentences (i.e., computing TF-IDF or BM25 scores, similarly to what was done on the first part of the project);

[2] `http://www.linguateca.pt/Repositorio/TeMario/`

- Edge weights based on the cosine similarity between pairs of sentences, again leveraging TF-IDF or BM25 weights when building the representations;
- Edge weights based on the number of noun phrases shared between the sentences, extracted through a procedure similar to the one used in the first part of the project;
- Edge weights based on a probabilistic model similar to a naïve Bayes classifier, which assigns a probability to each sentence conditioned on the entire document;
- Edge weights based on computing similarities from *dense* representations for the sentences, for instance obtained through a latent semantic indexing procedure based on a Singular Value Decomposition (SVD), or obtained from distributional word representations (e.g., leveraging existing datasets of pre-trained word embeddings[3]).

The evaluation for this exercise will value creative solutions, proposed to improve the results when using graph ranking for extractive summarization. In the report, you should present the mean average precision scores for the original versus the improved methods, with basis on summaries built through the concatenation of the top-5 highest ranked sentences.

3 A supervised learning-to-rank approach

Extractive document summarization can also be formulated as a supervised learning-to-rank problem. In this case, the sentences from a given document can be represented through a set of descriptive features (e.g., through different ranking scores), and training data can be used to learn an optimal combination of the features for the purpose of selecting the most relevant sentences to include in the summary.

For this exercise, you should develop a program implementing one such supervised approach for document summarization, leveraging a point-wise learning to rank strategy based on one of the algorithms introduced in the classes (e.g., the Perceptron algorithm). The following features can be considered in the ranking model:

- Position of the sentence in the document;
- Cosine similarity of the sentence towards the entire document, leveraging TF-IDF or BM25 weights when building the representations, or leveraging representations built through a latent semantic indexing procedure based on the SVD;
- Estimates produced though a model similar to a naïve Bayes classifier, which assigns a probability to each sentence conditioned on the entire document;
- Graph centrality scores.

[3] `https://github.com/nlx-group/lx-dsemvectors`

After training, the inferred learning-to-rank model can be used to score sentences, and the top-5 scoring candidates should be returned as the summary.

You should use the same dataset from the previous exercise for evaluating the quality of the obtained results (e.g., through the mean average precision). For training data, you can use the documents and the summaries made available in a second corpus[4] that extends the original TeMário dataset. The evaluation for this exercise will also value creative solutions for improving the results (e.g., solutions involving different sets of features).

4 A practical application

This exercise concerns with the development of a program for illustrating extractive multi-document summarization in a practical application. Your program should parse the XML/RSS feeds from the *World news* section of several different sources: the New York Times[5], CNN[6], the Washington Post[7], and Los Angels Times[8]. Your program should parse the XML contents and extract the textual contents for the news articles in the feeds. Representations should be built from the sentences in the different news articles, and a summary of the current news should be produced by taking the top-5 most relevant sentences.

The extractive summarization method can be based on either of the programs developed in this second part of the course project.

As a result, your program should generate an HTML page illustrating the summary. The evaluation for this exercise will value creative solutions for presenting the results (e.g., connecting sentences in the summary with the original documents, explaining the reason why a particular sentence was included in the summary, etc.)

[4] http://conteudo.icmc.usp.br/pessoas/taspardo/TeMario_2006.rar

[5] http://www.nytimes.com/services/xml/rss/index.html

[6] http://edition.cnn.com/services/rss/

[7] https://www.washingtonpost.com/rss-feeds/2014/08/04/ab6f109a-1bf7-11e4-ae54-0cfe1f974f8a_story.html

[8] http://www.latimes.com/local/la-los-angeles-times-rss-feeds-20140507-htmlstory.html