IST 2016

Processamento e Recuperação de Informação

Course Project: Part 2

One important information extraction task relates to the extraction of relevant topical phrases from textual documents, commonly known as **automatic keyphrase extraction**.

For a given document (or for a given document collection), the extracted keyphrases should be relevant to the major topics being described, and the set of keyphrases should provide good coverage of all the major topics. The fundamental difficulty lies thus in determining which keyphrases are the most relevant and provide the best coverage.

The second part of the course project for *Information Processing and Retrieval* will continue to explore different alternatives for addressing the task of automatic keyphrase extraction.

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 the automatic extraction of keyphrases. These methods are based on representing documents as a graph, where nodes correspond to keyphrase candidates, and where edges encode co-occurrence between candidates (e.g., an edge between two nodes encodes the fact that the associated candidates co-occur within a same sentence).

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

- You should start by creating a graph where nodes correspond to n-grams (where $1 \leq n \leq 3$, and ignoring stop-words) from the document being processed, and where edges encode co-occurrences within the same sentence;
- Candidates should then be ranked according to a variation of the PageRank algorithm for undirected graphs, which computes a score for each candidate 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 keyphrase candidates under consideration, $\mathrm{Links}(p_i)$ is the set of candidates that co-occur with p_i (i.e., the set of nodes lined to p_i in the graph), and N is the total number of candidates. 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 candidates should be returned as the keyphrases.

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

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 candidate, and also in order to consider weighted edges in the graph, indicating the degree of association between the candidates. 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 keyphrase extraction, evaluating them through the mean average precision over one of the datasets[2] from the list that was given to you 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 candidate in the document (e.g., in the first sentence, in the second sentence, etc.), under the intuition that candidates in the first sentences are more likely to constitute good keyphrases;
- Non-uniform prior weights based on TF-IDF or BM25 scores for the candidates (i.e., computing TF-IDF or BM25 scores, similarly to what was done on the first part of the project);
- Edge weights based on number of co-occurrences involving the candidates;

[2] `https://github.com/zelandiya/keyword-extraction-datasets`

- Edge weights based on distributional word similarity between the words in the candidates, for instance 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 keyphrase extraction. In the report, you should present the mean average precision scores for the original versus the improved methods.

3 A supervised learning-to-rank approach

Keyphrase extraction can also be formulated as a supervised learning-to-rank problem. In this case, candidates for 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 relevant keyphrases.

For this exercise, you should develop a program implementing one such supervised approach for keyphrase extraction, 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 candidate in the document;
- Term frequency, inverse document frequency, TF-IDF, or BM25 scores for the candidate;
- Phraseness and informativeness scores;
- Graph centrality scores.

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

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 all the remaining datasets that are available. 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 keyphrase extraction in a practical application. Your program should parse one of the XML/RSS feeds from The New York Times[4], extract the titles and descriptions for the news articles in the feed, and show the most relevant keyphrases currently in the news.

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

[3] http://nlp.stanford.edu/projects/glove/

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

As a result, your program should generate an HTML page illustrating the most relevant key-phrases (e.g., using either a list, a chart, word clouds, etc.). The evaluation for this exercise will value creative solutions for presenting the results.