IST 2019/2020

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, eventually using implementations for particular algorithms from external libraries) 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 and punctuation) from the document being processed, and where edges encode co-occurrences within the same sentence;

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

- 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:

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

  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. You can use existing implementations for the PageRank algorithm (e.g., from the `NetworkX` package), but keep in mind that you should use a variant for unweighted graphs, as shown in the previous equation.

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 length or the position of the candidate in the document (e.g., in the first sentence, in the second sentence, etc.), under the intuition that longer candidates in the first sentences are more likely to constitute good keyphrases;

[2] `https://github.com/boudinfl/ake-datasets`

- 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 the number of co-occurrences involving the candidates, computed over a background collection;
- 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, leveraging one of the datasets from the set of evaluation resources that was suggested in the first part of the project. Again, you can use existing implementations for the weighted PageRank algorithm (e.g., from the `NetworkX` package), conforming to the presented equation.

3 An unsupervised rank aggregation approach

Keyphrase extraction can also be formulated as an unsupervised learning-to-rank (i.e., rank aggregation) 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 a combination method such as CombSUM/CombMNZ or Reciprocal Rank Fusion can be used for scoring and selecting the most relevant keyphrases. For this exercise, you should develop a program implementing one such unsupervised approach. The following features can be considered in the aggregation model:

- Term frequency, inverse document frequency, TF-IDF, or BM25 scores for the candidate;
- Graph centrality scores (e.g., computed with the PageRank method).

For combining the ranking features, you should preferentially use a Reciprocal Rank Fusion approach, corresponding to the following equation:

$$\text{RRFScore}(k \in D) = \sum_{f \in R} \frac{1}{50 + \text{rank}(f_k)} \tag{1}$$

The fusion method can be used to score candidate phrases, 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), and you should compare the unsupervised approach against the supervised method from the first part of the project, using the same set of features.

The evaluation for this exercise will also value creative solutions for improving the obtained results (e.g., solutions involving different sets of features).

[3] `https://fasttext.cc/docs/en/english-vectors.html`

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. 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 (e.g., you can consider clustering the news articles according to keyphrases, use different types of graphical presentations, etc.)

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