Automatic Keyphrase Extraction - Part 2
Information Processing and Retrieval

Martin Mirakyan
ist194771
mirakyanmartin@gmail.com

Abstract

This report is the continuation of the experiments on creating an automatic keyphrase extraction system within the scope of the Information Processing and Retrieval course. We run experiments by testing different approaches and building solutions on top of the first part of the project. We evaluate all the experiments on the test data to show which of those perform the best. In the end, we do a small practical application of the keyword extraction system on real-world data and generate a simple UI to display the results.

1 Introduction

Keyword extraction systems have different ways of finding and ranking keyphrases from texts. For the second part of the project, we have experimented with graph-based approach TextRank (Mihalcea and Tarau, 2004), and unsupervised rank aggregation methods to rank candidate keyphrases.

For each of the methods, we've experimented with different features and parameters to improve the performance. To evaluate the methods, we have stemmed both the predictions and the target keyphrases before comparing to each other. All the codes developed in the scope of this project are open-sourced and available on GitHub [1].

2 Exercise 1 - An approach based on graph ranking

For the first exercise, we had to create a simple keyphrase extraction method that will extract keyphrases from a text of our choice. Similar to the first part of the project we have selected the famous alice.txt text. The graph ranking approach is based on, first, selecting all the unigrams, bigrams and trigrams from the sentences in the document as candidates, second, considering each candidate as a node in a graph and connecting the nodes if they appear in the same sentence, and then calculating the centrality scores for each candidate. After running the algorithm, the top 5 candidates are selected. For our selected document Alice.txt the results were: alice, said, little, went, like.

3 Exercise 2 - Improving the graph-ranking method

The second part aims to improve the initial approach by calculating the centrality scores with weights in the graph, while also giving priority to some candidates. To evaluate the system we calculate the Mean Average precision score for each document in the Inspec collection of the ake-datasets [2].

For the priorities and the weights between the nodes, we have experimented with different methods and evaluated them on the test split of the Inspec dataset.

Method	MAP
Unweighted, No personalization	0.020
Unweighted, TF-IDF score as personalization	0.019
Unweighted, Position score as personalization	0.025
Co-occurance weighting, No personalization	0.003
Fasttext cosine similarity, No personalization	0.026
Fasttext, Position score	**0.031**

Table 1: Mean Average Precision score for different parameters of PageRank algorithm on Inspec test set

If the method is unweighted, the default value for all the weights is set to 1.

We have experimented with setting the scores for personalization to the TF-IDF scores of the candidate which improved the results. Yet, setting the personalization score of the candidate to the position in the document demonstrated better

[1] https://github.com/MartinXPN/Tecnico/tree/master/InformationProcessing/Project2

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

results. The position score PS for candidate c in document d was calculated as:

$$pos = d.findPosition(c)$$

$$PS(d, c) = \frac{1}{1 + \log(1 + pos)}$$

For the weights between the candidates, we have experimented with co-occurrence weighting which did not perform good, while setting the weight equal to the cosine similarity between the two candidate vectors obtained from fastText (Bojanowski et al., 2016) performed much better.

Finally, we experimented with combining the best methods for personalization and weighting - position score as personalization and fasttext vector cosine similarity as weights between candidates. The ensemble method demonstrated the best performance among all the methods.

4 Exercise 3 - An unsupervised rank aggregation approach

The aim of this exercise is to create an ensemble method that aggregates different keyphrase extraction scores to improve upon them.

We have used Reciprocal Rank Fusion method to combine the TF, IDF, TD-IDF, BM25, Length, and Centrality scores of the candidates. Centrality score was calculated by the best method obtained from the previous exercise - the ensemble method.

To evaluate the method, we have compared the Mean Average Precision of the Reciprocal Rank Fusion with the ones obtained from the first part of the project by xgboost.

Method	MAP
xgboost	0.07
RRFScore	**0.08**

Table 2: Mean Average Precision score for different methods to extract keyphrases evaluated on Inspec test set

As demonstrated in the table the RRF method performed much better than the xgboost supervised classification method.

5 Exercise 4 - A practical application

Finally, we create a python script that will generate an HTML document to display some particles with extracted keyphrases from their descriptions.

To do that we have used the best keyphrase extraction method obtained from the previous exercises and applied it on the New York Times XML/RSS Dance feed.

Some results are displayed below.

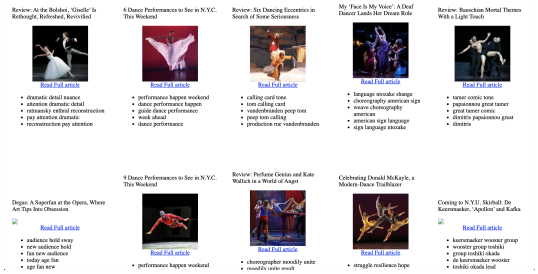
6 Dance Performances to See in N.Y.C. This Weekend

Read Full article

- performance happen weekend
- dance performance happen
- guide dance performance
- week ahead
- dance performance

Most of the keyphrases extracted convey useful information.

Below is the full UI of the generated page.



References

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*.

R. Mihalcea and P. Tarau. 2004. TextRank: Bringing order into texts. In *Proceedings of EMNLP-04and the 2004 Conference on Empirical Methods in Natural Language Processing*.