Term extraction and TopMine model

Jaroslav Kotliarov

September 2023

Abstract

This project was done to evaluate the effectiveness of the TopMine model in the task of extracting terms from scientific and other texts. The work used models of classes implemented in the *topmine* module. The set of articles and lists of terms for them were obtained from open sources. As a result of the algorithm's work, lists of thematic phrases for each text were obtained, which were compared with lists of terms. Code is here: `https://github.com/kotyaro508/NLP-2023`.

1 Introduction

There are already NLP models designed to extract thematic phrases from texts. However, there is a question about whether these phrases can be called terms. At first glance, the concepts «term» and «thematic phrase» are equivalent. This leads to the idea of using one of the models to obtain thematic phrases from an paper in order to compare the result with a list of potential terms for this paper.

1.1 Team

Jaroslav Kotliarov did the work.

2 Related Work

As already stated, the key element in this work is the TopMine model. More details about its structure are described in [1]. This model has its own *topmine* module in Python. An implementation of the module with two important components of this algorithm can be found here `https://github.com/anirudyd/topmine/tree/master`. Based on the results of using this model, we will additionally talk about its advantages and disadvantages.

3 Model Description

The TopMine algorithm consists of two parts.

Using the PhraseMining operation, candidate topic phrases are obtained from a set of documents. Documents are submitted for entry in pre-processed form. It is necessary to leave only those words that carry a semantic load in the sentence, and write them down with a space.

The PhraseMining operation consists of two steps. The first step involves searching and analyzing frequently occurring phrases. At the second step, they are segmented using the bottom-up method. At the same time, low-quality phrases are filtered.

After PhraseMining is completed, the PhraseLDA function is applied to the results of dividing documents into phrases. This is necessary to get an approximate representation of the distribution of terms across documents as a combination of some distribution of terms across topics and distribution of topics across documents. The use of the PhraseLDA function assumes that these two distributions can be considered as random variables distributed according to the Latent Dirichlet Allocation rule. The output from the function is the distribution of phrases in each document by topic.

4 Dataset

A link to Google Drive with preprocessed texts of articles, as well as the correct sets of terms for them, is in the project's GitHub repository. The texts of the articles are taken from the openly published ACL RD-TEC dataset `https://aclanthology.org/W14-4807/`.

5 Experiments

5.1 Metrics

In the final part of the work it is necessary to compare pairs of lists for which it would be appropriate to use metrics showing the proximity of two sets (such as Jaccard distance). However, in this work it is proposed to use standard binary classification metrics

$$precision = \frac{TP}{TP + FP}, recall = \frac{TP}{TP + FN},$$

$$F1 = \frac{2 * precision * recall}{precision + recall}.$$

For the entire set of texts, it is proposed to take average values

$$score\_total = \frac{1}{N} \sum_{i=1}^{N} score_i.$$

5.2 Experiment Setup

First of all, you need to install the *topmine* module.

Then, to feed data into an object of the class *PhraseMining* from the module *topmine.phrase_mining*, you need to convert the text archive into a list of strings.

When creating the object itself for the resulting list, you can specify the maximum phrase length parameter. After creating an instance the *.mine*() class method is used to obtain a breakdown of strings into phrases and a complete vocabulary of all words found in the archive. Then an object of the *topmine.phrase_lda.PhraseLDA* class is created requiring two output objects from the previous step. Additionally, you can specify the number of topics as well as the number of iterations of the main method of this class. After the *.ran*() method is called, from the output of which a list of lists of topic numbers of each phrase of each document is selected.

To obtain a list of thematic phrases for each text, a self-written algorithm is used. To do this, we first identify the topic of each document as the topic with the largest number of phrases in that document. Then, from the phrases, only those that relate to the topic of the document are selected. The phrases themselves are obtained using phrasal indices from the vocabulary. Since each list may contain many cognates, lemmatization should be performed.

Ultimately, the precision, recall and F1-score metrics are calculated.

6 Results

While working with the algorithm, I discovered that identifying the topic of the text as the most common among its constituent phrases is a good solution. If you look at the distribution of document phrases by topic, then in most cases there is one topic whose number of phrases in a given document is significantly greater than all others.

To compare the effectiveness of the approach I proposed for this problem, quality metrics of another algorithm are provided. You can find out more about it by following this link [2]. The results are shown in Table 1.

Algorithm	precision	recall	F1-score
TopMine based	0.17	0.29	0.21
BERT and LSTM based	0.76	0.82	0.79

Table 1: Metrics of the quality of work of the algorithm based on TopMine and the algorithm based on BERT and LSTM.

7 Conclusion

For this work I searched for methods for extracting terms from scientific and other articles. I chose the TopMine algorithm which, as I found out, solves a slightly different problem. I pre-processed the data, applied two models from the *topmine* module and received lists of words for each text which I compared with the lists of terms I found in an open source. As a result, it turned out that this algorithm did not give a good result with the input parameters that I chose for it. There are suspicions that if for the existing dataset we choose other values for hyperparameters such as the number of topics or the number of iterations for the *PhraseLDA* class, then the results may be much better. But if you look at the words that are obtained at the output of the algorithm, you can understand why the «thematic phrase» and the «term» are not identical concepts. After all, it is reasonable to believe that numbers or proper names are not terms, although they may well characterize the topic in which they are mentioned.

In the end, I would like to note the features of the algorithm for solving this problem. On the one hand, it is very simple. You can easily navigate the structure of the *PhraseMining* and *PhraseLDA* classes if you open their implementation via a link, and it was difficult to make a mistake when writing their work. On the other hand, this algorithm cannot be applied to one document since in order to say that a phrase is a topic, it is necessary to assign this document to one of the topics. At the same time, the algorithm does not have the *.fit*() and *.predict*() methods, so it is not designed to determine its topic for one new text and extract thematic phrases from it. In such cases, you need to re-assemble a large dataset of texts and wait a long time for the model to finish working with it.

References

[1] El-Kishky, Ahmed, et al (2014). Scalable topical phrase mining from text corpora. *Department of Computer Science The University of Illinois at Urbana Champaign*.

[2] Sharma, P., Li, Y. (2019). Self-Supervised Contextual Keyword and Keyphrase Retrieval with Self-Labelling.