Comparing Different Keywords Extraction Methods

Wuhao Wang (wuhwa469@student.liu.se)

Abstract

Key-words extraction is an important task in NLP fields, many advanced tasks are based on key-words extraction. In this project, TF-IDF, Keybert, TextRank and Yake will be compared. The dataset is from other NLP competitions and the evaluation is based on the f1-score. We found that Keybert and Yake are with the highest f1-score while TF-IDF is the most efficient method. Textrank is not flexible since it can not customize the number of keywords.

1 Introduction

Key-words detection is always an important task in many text applications. In this project, four methods will be discussed and compared. The first methods are TF-IDF[1], which basically calculates and compares the frequency of a specific word in all the text data and one text data. A higher TF-IDF value means higher importance of this word to the text.

Another model isknown as Yake(Campos et al., 2020b)(Campos et al., 2020a)(Campos et al., 2018). Yake can be considered as an extension of TF-IDF, this algorithm will calculate TF-IDF value and also cosine similarity, then a word graph will be constructed to obtain the weights of each candidate word.

The third one is a self-supervised learning method called KeyBert (Grootendorst, 2020). In this approach, document-level vector representations are extracted by BERT. Subsequently, word vectors are extracted for the N-gram words/phrases, and then, cosine similarity will be applied to find the most similar word/phrase to the document. Finally, the selected words will be identified as the words that best describe the entire document.

[1] https://en.wikipedia.org/wiki/Tf/$%$E2/$%$80$%$93idf

Last Algorithm is TextRank(Mihalcea and Tarau, 2004). The TextRank algorithm is a graph-based sorting algorithm. The workflows are as follows: putting text segmentation into several constituent units (e.g. sentences), building a node connection graph, using similarity between sentences as the edge Weights, calculating the TextRank value of the sentence through loop iterations, and finally extracting the sentences with high rankings and combine them into text summaries.

1.1 Motivation

The main purpose of this project is to explore the performance and characteristics of different models so that a reasonable choice of models can be made in practical scenarios. Keyword extraction is usually used as a base task for other natural language processing tasks, so the selection of an efficient model can help a lot to improve the overall task timeliness

2 Related work

Mohammad-Reza Feizi-Derakhshi(Nikzad-Khasmakhi et al., 2021) used bert model to do key-phrase extraction. The bert model with graph-text-based embedding can reach a 0.70 f1-score. MingXi Zhang(Zhang et al., 2020) tested the Textrank algorithm in many key-words extraction datasets, but they only have the precision as evaluation. In 2018, Bijoyan Das and Sarit Chakraborty(Das and Chakraborty, 2018) used TF-IDF to grab keywords and use those keywords to do different tasks. In a classification task, they reach 96.83% accuracy. Shahzad Qaiser and Ramsha Ali (Qaiser and Ali, 2018)used TF-IDF to find the words with the highest relevance to the documents, they compared their rank results with the golden label but they did not mention evaluation scores like f1-score. SusieXi Rao, Piriyakorn Pirytamwong and Parijat Ghoshal (Rao et al., 2022) did experiment on different algorithms

including textrank and keybert. They use k-means to help textrank clusterings and used a self-trained model as a keybert language model. The result show textrank is more time efficient than keybert but lower f1-score.

3 Dataset

The data set used in this project is Inspec (Hulth, 2003). The language is English, and the original text is collected from scientific papers within computer science fields. Below are the numeric features of the data.

N-docs	TG	TPD
2000	29230	128.2

N-docs is the number of documents contained in the data, **TG** is the number of gold keywords in the dataset, **TPD** is an average number of tokens in each document.

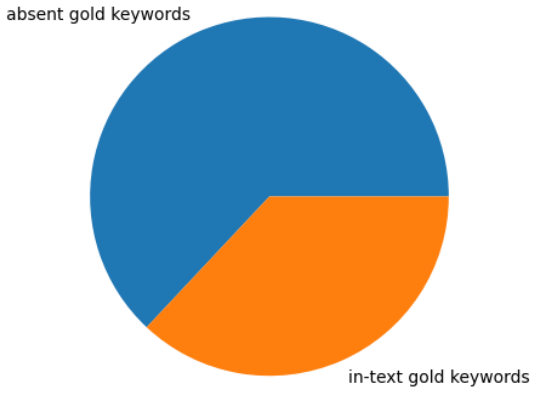

Figure 1: Absent gold keywords

The advantage of using this data is that the dataset contains only 2000 documents with an average number of tokens of 128. This means the training is less computational density than a huge dataset. But the dataset contains 37 percent golden keywords (see graph above) that are absent in the resource document, this will automatically lead to a lower recall rate.

4 Evaluation

In the competition (Augenstein et al., 2017) held by Association for Computational Linguistics in Vancouver, Canada, they use the f1-score to evaluate the results. In this project, all the model performances will be evaluated by f1-score in this project. The f1-score is more robust than accuracy. The f1-score can be calculated by:

$$f1 = \frac{2}{recall^{-1} + precision^{-1}} = \frac{2tp}{2tp + fp + fn} \quad (1)$$

Tp here means true positive, fp here means false positive and fn means false negative. In this project, **true positive** means if a gold key-word is predicted as a gold key-word, **false negative** means if a gold key-word is predicted as a non-gold keyword,**false positive** means if a non-gold keyword is predicted as a gold keyword. As mentioned before, this dataset contains an average of 37 percent golden keywords that are absent in the document, this means the false negative will be higher than 0.37. This will make the f1-score of each model lower than the usual dataset.

5 Experiment

In this section, the results of each method will be presented and discussed. TF-IDF, Keybert, and Yake support a customized number of keywords. We will select 5 benchmarks (10,12,16,18,20) to test each model's f1 score. The textrank methods implemented by spacy or summa do not support the number of keywords as a hyperparameter, so the results from two different implementations will be compared.

5.1 TF-IDF

For TF-IDF method, this project is referring to the spacy package[2]. TF-IDF will calculate the frequency of a term showing up in a document(tf) and also ratio of how many documents contains the term(idf). A higher tf-idf value is supposed to imply a more important term. The results of TF-IDF method are as follows. N is the number of keywords that the model would extract, and f1 is the corresponding f1-score.

[2] https://github.com/explosion/spaCy

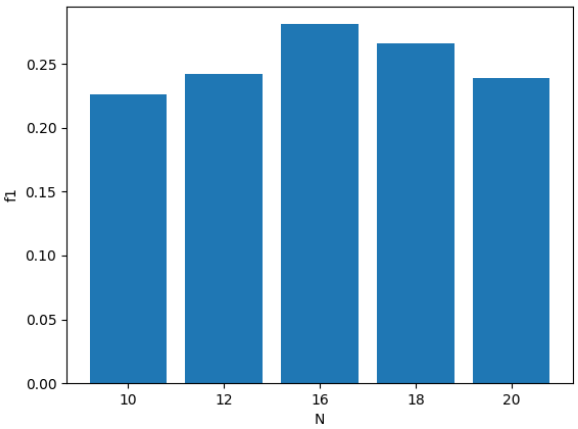

Figure 2: TF-IDF results

5.2 Yake

For yake model, this project is referring to the official package[3]. Yake is an unsupervised method, it will first find similar articles by calculating tf-idf and their cosine distance. With in those selected words, candidate words will be selected by calculating the global affinity graph. The keywords will be then selected by sorting candidate words according to global affinity graph value from high to low. The results of yake are as follows. N is the number of keywords that the model would extract, and f1 is the corresponding f1-score. Note that there are many other parameters that can be tuned, see the appendix for more details.

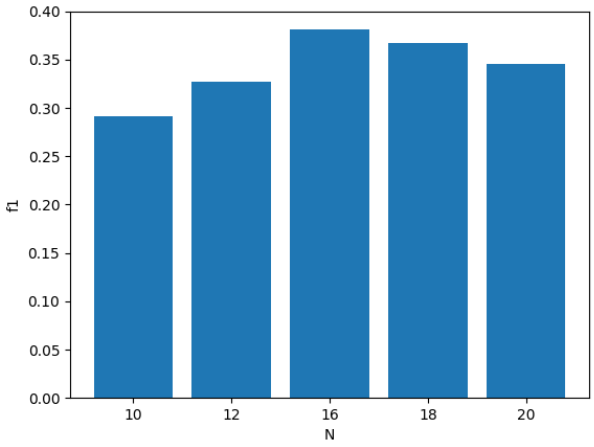

Figure 3: Yake results

5.3 Keybert

For Keybert model, this project is referring to the official package[4]. Keybert uses bert skeleton. A pre-trained network will be applied to extract word vectors. Then N-gram will be calculated and compared will the document vector. Keywords will then be selected by sorting the cosine distance between the candidate word vector and the document. The results of Keybert are as follows. N is the number of keywords that the model would extract, and f1 is the corresponding f1-score. Note that there are many other parameters that can be tuned, see the appendix for more details.

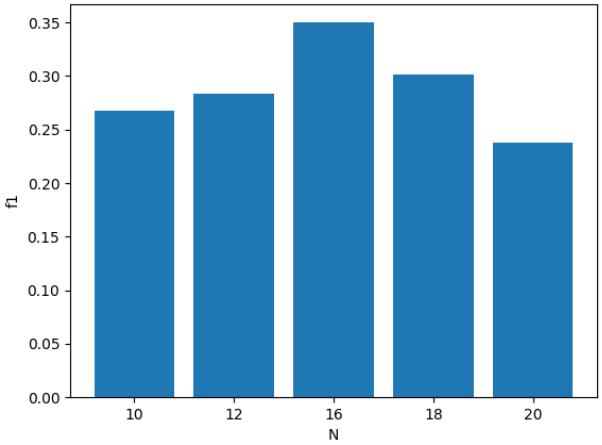

Figure 4: Keybert results

5.4 Textrank

For the textrank model, this project is referring to two different implementations, one is summaNLP[5], another one is spacy [6]. By partitioning the text into several component units (words, sentences) and building a graph model, and using a voting mechanism to rank the important components of the text, keyword extraction and digestion can be achieved using only the information of a single document itself. The Textrank algorithm does not offer "number of keywords" parameter, the results of two different implementations are as follows:

implementations	summaNLP	Spacy
f1	0.200	0.179

6 Discussion

All the models have the highest f1 score when the number of keywords is 16. To clearly compare each model's performance, the highest f1 score of each model is selected below.

[3] https://github.com/LIAAD/yake
[4] https://github.com/MaartenGr/KeyBERT
[5] https://github.com/summanlp/textrank
[6] https://spacy.io/universe/project/spacy-pytextrank

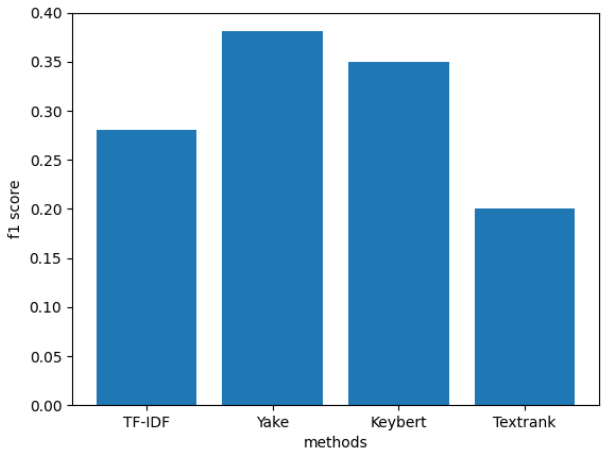

Figure 5: Highest Score Of Each Model

Yake is based on TF-IDF, the weights of each word come from document-level statistics. But Yake considers more features than TF-IDF and the f1-score also reflects that Yake is more advanced than TF-IDF.

Keybert uses BERT to extract the word vectors and considers the similarity between words and documents. So, the result highly depends on which pre-trained network is used.

Textrank is based on a graph algorithm. Like a transformer-based model, it will divide the text into different parts. The different thing is that Textrank will consider those elements as points and edges in a graph. Then a voting system will select the most important words.

7 Conclusion

From the experiment results, one can easily see that yake and keybert are much better than the other two methods. For this dataset, every document contains an average of 15 golden keywords. And for textrank which does not receive the "number of keywords" parameter, the f1-score is lower because of too few key-words predictions. In this project, We also try to set "number of key-words = 5" for other models and the results are quite similar to textrank. In one word, the TF-IDF method is the fastest method among these four methods while keybert and yake have the highest f1-score.

For simple keyword extraction like getting key-words from a paper, tf-idf value is enough since this scenario normally needs several single words to identify the research fields, and tf-idf method is the simplest method with the highest time efficiency. For more complex works where contextual information is needed like extracting keywords from a piece of news or a pice of product description, we need keybert, Yake, or textrank since those methods build a language model and take context into consideration. When it comes to key-phrase extraction, tf-idf methods can hardly work since this method has no representation of a phrase.

Note that in Inspec dataset, 37.7% of keywords do not show up in the text, which means the 'fn' in the f1-score is at least 0.37. That is an important reason why all the models can not have a high f1-score.

The drawback of TF-IDF is obvious. TF-IDF method can only get every single keyword. When it comes to key phrase extraction(KPE), the TF-IDF method can not work.

8 Future Work

In this project, the dataset semeval2017 (Augenstein et al., 2017) , which contains key phrases, is also applied to all of these four models. We pick the same hyperparameters as in the Inspec datasets, but the length of a key phrase is set in a range from 1 to 4.

models	TF-IDF	Keybert	TextRank	Yake
f1	0.04	0.13	0.12	0.15

Based on this project, there are multiple future tasks. The most important thing is to think about how can we use the TF-IDF method to predict key phrases. Maybe this can be achieved by combining the candidate keywords and re-calculate the weights.

Another thinking is about how to improve other models' f1-score in keyphrase extraction. Maybe this can be achieved by applying some pre-processing procedure or using a more advanced pre-trained word2vec model in keybert. There is also a possibility to ensemble different models and get more robust ensembled models.

In the semeval-2017 competition, the highest team reach the f1-score of 0.68. It is a quite good result but is still far away from the practical usage standard.

References

Isabelle Augenstein, Mrinal Das, Sebastian Riedel, Lakshmi Vikraman, and Andrew McCallum. 2017. SemEval 2017 task 10: ScienceIE - extracting keyphrases and relations from scientific publications. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 546–555, Vancouver, Canada. Association for Computational Linguistics.

Ricardo Campos, Vítor Mangaravite, Arian Pasquali, Alípio Jorge, Célia Nunes, and Adam Jatowt. 2020a. Yake! keyword extraction from single documents using multiple local features. *Inf. Sci.*, 509(C):257–289.

Ricardo Campos, Vítor Mangaravite, Arian Pasquali, Alípio Mário Jorge, Célia Nunes, and Adam Jatowt. 2018. Yake! collection-independent automatic keyword extractor. In *Advances in Information Retrieval*, pages 806–810, Cham. Springer International Publishing.

Ricardo Campos, Vítor Mangaravite, Arian Pasquali, Alípio Jorge, Célia Nunes, and Adam Jatowt. 2020b. Yake! keyword extraction from single documents using multiple local features. *Information Sciences*, 509:257–289.

Bijoyan Das and Sarit Chakraborty. 2018. An improved text sentiment classification model using TF-IDF and next word negation. *CoRR*, abs/1806.06407.

Maarten Grootendorst. 2020. Keybert: Minimal keyword extraction with bert.

Anette Hulth. 2003. Improved automatic keyword extraction given more linguistic knowledge. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*, EMNLP '03, page 216–223, USA. Association for Computational Linguistics.

Rada Mihalcea and Paul Tarau. 2004. TextRank: Bringing order into text. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 404–411, Barcelona, Spain. Association for Computational Linguistics.

Narjes Nikzad-Khasmakhi, Mohammad-Reza Feizi-Derakhshi, Meysam Asgari-Chenaghlu, Mohammad-Ali Balafar, Ali-Reza Feizi-Derakhshi, Taymaz Rahkar-Farshi, Majid Ramezani, Zoleikha Jahanbakhsh-Nagadeh, Elnaz Zafarani-Moattar, and Mehrdad Ranjbar-Khadivi. 2021. Phraseformer: Multimodal key-phrase extraction using transformer and graph embedding.

Shahzad Qaiser and Ramsha Ali. 2018. Text mining: Use of tf-idf to examine the relevance of words to documents. *International Journal of Computer Applications*, 181.

Susie Xi Rao, Piriyakorn Piriyatamwong, Parijat Ghoshal, Sara Nasirian, Emmanuel de Salis, Sandra Mitrović, Michael Wechner, Vanya Brucker, Peter Egger, and Ce Zhang. 2022. Keyword extraction in scientific documents.

Mingxi Zhang, Xuemin Li, Shuibo Yue, and Liuqian Yang. 2020. An empirical study of textrank for keyword extraction. *IEEE Access*, 8:178849–178858.

A Appendices

A.1 Hyperparameter

For Keybert

hp	div	maxsum	model
value	0.2	False	'all-MiniLM-L6-v2'

For Yake

hp	threshold	algorithm	Size
value	0.9	'seqm'	1