Comparing Different Keywords Extraction Methods

Wuhao Wang (wuhwa469@student.liu.se)

Abstract

Key-words extraction is an important task in NLP fields. The aim of this project is to compare the performance of different currently popular keyword extraction methods including TF-IDF, Keybert, TextRank and Yake. 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

Keyword extraction techniques can capture a set of words from a piece document to present the core information of the document. In many natural language processing tasks, such a set of words is sufficient to represent the information of a document and saves time and memory compared to using the raw document directly. For example, in the text classification task, one typical method is to calculate the cosine distance between the word vectors of keywords and the word vectors of class labels and assign the text to the label with the closest distance. The qualified keywords here are enough to represent the document and it can help save time and memory compared to using the whole document. Besides document classification, many other tasks in the text mining context such as text searching, text comparison, and abstract generation require an efficient and precise keyword extraction algorithm to pre-process input text data.

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. In this project, four methods are 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 is known 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(Devlin et al., 2018). 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.

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.

The results show that Yake and Keybert are the best models among these four methods and TF-IDF is the fastest method. TextRank is not flexible because we can not control the number of keywords to extract and it also has the worst performance.

2 Related work

Nikazad-Khasmakhi(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

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

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 key-words 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 shows TextRank is more time efficient than keybert but lower f1-score.

3 Dataset

Inspec(Hulth, 2003) consists of 2,000 abstracts of scientific journal papers from Computer Science collected between the years 1998 and 2002. The keywords of each document in this dataset can be divided into two parts. First, the golden keywords are assigned by the paper's author freely, but these keywords have to be picked from the paper's abstract. Another one, the absent golden keywords, is created by Anette Hulth. The absent golden keywords are related to the article abstract but do not appear in the abstract. Both of these two parts are considered ground truth keywords and there is no label indicating which keyword is absent.

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.

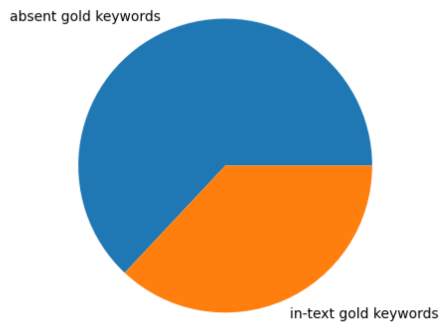

Figure 1: Absent gold keywords

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.

4 Evaluation

In the competition (Augenstein et al., 2017) held by Association for Computational Linguistics in Vancouver, Canada, f1-score is used 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 and it can be calculated by:

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

Tp and fp mean true positive and false positive respectively 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 key-word,**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. 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. The experiment time is recorded based on AMD Ryzen5800 CPU.

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. From Figure 2, the best model is found when the number of keywords is 32. The corresponding precision is 0.31, recall is 0.299 and f1-score is 0.285. It consumes 40 minutes to predict 39 times on this dataset(around 1 minute per time).

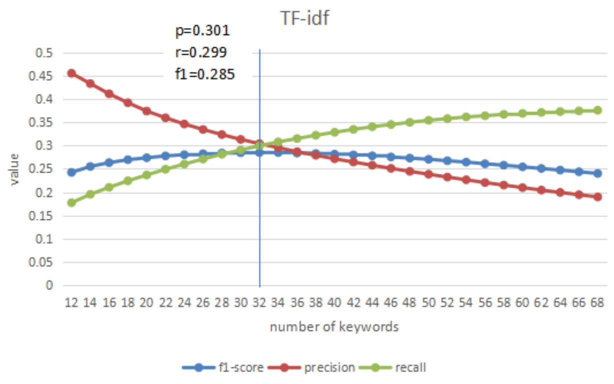

Figure 2: TF-IDF results

5.2 Yake

For Yake model, this project is referring to the official package[3]. Yake is an unsupervised method and it does not require a corpus to train. Yake considers a word from five different perspective(Campos et al., 2020b)(Campos et al., 2020a)(Campos et al., 2018): the Casing of the word, the position of the word in a sentence, the frequency of the word appears, the distance of the word to stopwords, the frequency of the terms show up in different sentences. By combining all the local features together, Yake then sorts the candidate word by the scores. The results of yake are as follows. It consumes around 2.5 hours to evaluate 39 times on this dataset(around 4.5 minutes per time).

[2]https://github.com/explosion/spaCy
[3]https://github.com/LIAAD/yake

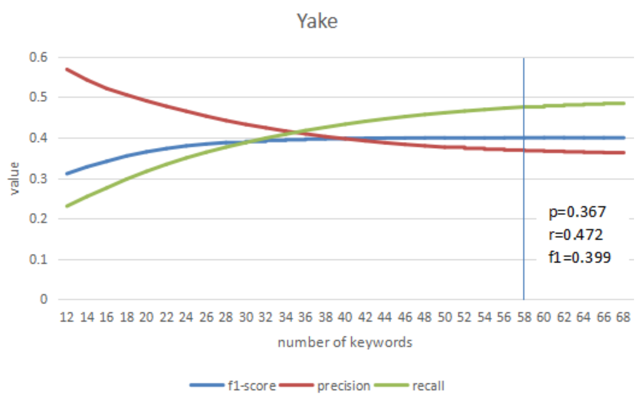

Figure 3: Yake results

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, and the correlation between words will be learned. 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(Grootendorst, 2020). Two different pre-trained models are compared here. Figure 4 is the results of **en-web-core-lg**[5] and Figure 5 is the results of **en-web-core-sm**[6]. The highest f1-score is labeled with a blue straight line. Note that **en-web-core-lg** consumes 4.5 hours to evaluate this dataset and **en-web-core-sm** consumes around 2 hours.

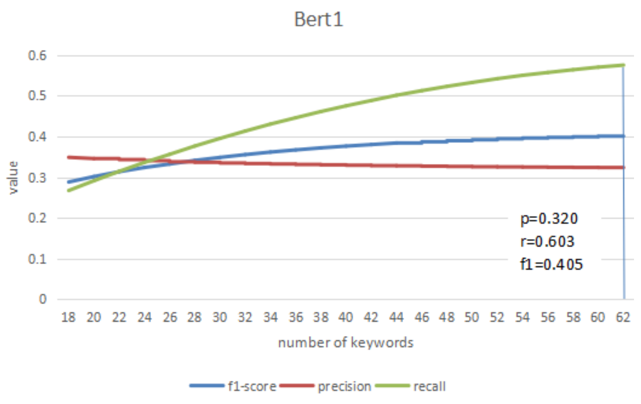

Figure 4: Keybert with weblg

[4]https://github.com/MaartenGr/KeyBERT
[5]https://spacy.io/models/en#en_core_web_lg
[6]https://spacy.io/models/en#en_core_web_sm

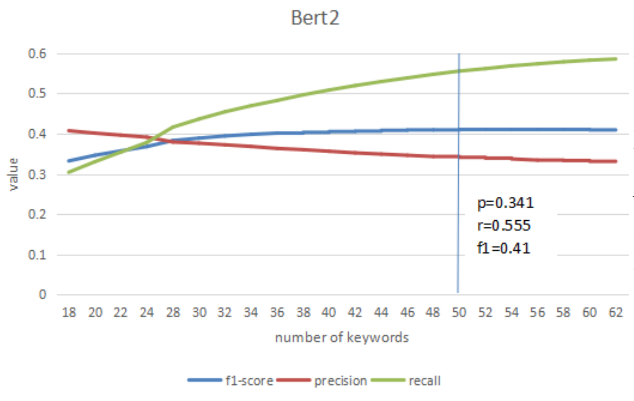

Figure 5: Keybert with websm

5.4 TextRank

For the TextRank model, this project is referring to two different implementations, one is summaNLP[7], another one is spacy [8]. 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

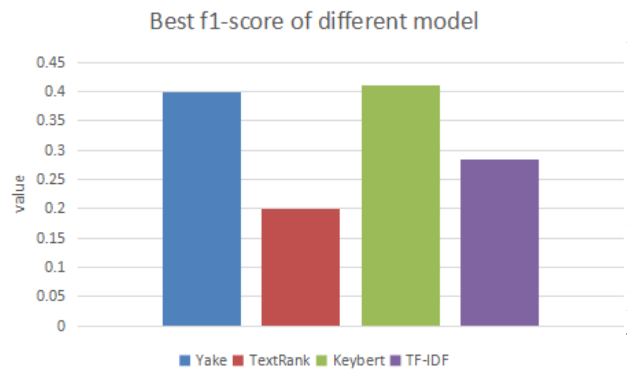

Figure 6: Highest F1-Score Of Each Model

TF-IDF method is the simplest method among the methods discussed in this project. The best f1-score of TF-IDF method is lower than Keybert and Yake mentioned in this project. But TF-IDF consumes the least amount of time.

TextRank algorithm performs the worst in this project. Yake is also a graph-based algorithm but also applies TF-IDF value of each word as a weight. Yake performs better than TextRank and TF-IDF methods but achieves a lower f1-score than Keybert.

The best model in this project is Keybert. We experiment with two different pre-trained models and the smaller one, whose results are shown in Figure 5, achieve a higher f1-score. From the results, we can see that Keybert's performance highly depends on the pre-trained model. Due to device and time limitations, those advanced models can not be tested.

Nikzad-Khasmakhi (Nikzad-Khasmakhi et al., 2021) also mentioned that the textual-based model could outperform the graph-based model(e.g. Yake and TextRank) on the Inspec dataset. Because Inspec is the abstract of a group of scientific papers, even the keywords will not show up many times. This means algorithms that require frequency information can not work well. Bert, the transformer family model, would consider contextual correlation, which is more helpful in this scenario (Devlin et al., 2018).

There is a large model performance gap between ours and the SOTA model, which can reach 0.69 f1-score. The SOTA model in the Inspec dataset is an embedded model (Nikzad-Khasmakhi et al., 2021). They extend Keybert by applying an extra graph model in the word vector extraction stage. They use Bert and a graph model to extract word vectors and concatenate them together. Experimental results show that such an embedded model achieves a higher f1-score than both single models.

The major limitations of the present study are the type of documents is too fixed. All the documents are scientific paper abstracts. This document would have its own style and format, which means some words may have a fixed pattern of position in a sentence. In this case, algorithms like Yake would be biased.

7 Conclusion

In the graph-based model, TextRank performs poorly and is not suitable as a model for keyword extraction. The Yake algorithm achieves similar results as the text-based model (Keybert), but the final results are not as good as the text-based model.

[7]https://github.com/summanlp/TextRank
[8]https://spacy.io/universe/project/spacy-pyTextRank

Among the text-based models, the TF-IDF method is the most time-efficient but does not achieve a good f1-score. the performance of the Keybert model depends on the selection of the base model(e.g., the 'web-sm' model in the experiment takes less time and achieves a higher f1-score than the 'web-lg' model).

A better solution would be to combine the graph-based model and the text-based model, as Nikzad-khasmakhi mentioned(Nikzad-Khasmakhi et al., 2021), by concatenating the outputs of the two models together.

8 Future Work

In future work, it might be important to see how to develop embedded models. In Nikzad-khasmakhi's method(Nikzad-Khasmakhi et al., 2021), they just concatenate the output tensor of two base models. But there might be other ways to do better. For example, one could get these two vectors into another neural network to learn the linguistic knowledge contained in these two tensors and get a better intermediate representative.

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.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

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.