AUTOMATIC KEYWORD EXTRACTION USING ARTIFICIAL NEURON NETWORK AND FEATURE EXTRACTION

Nguyen Van Son[1,*], Le Thanh Huong[2], Nguyen Chi Thanh[2]

Abstract: *The growth of the internet and the digital library have created massive electronic documents that the manual analysis of these documents is no longer feasible. Hence, extracting keywords from these documents is vital to have the first concept about these documents and search for relevant documents. However, extracting a limited number of keywords from a document that is enough to find all related documents is challenging. Automatic keyword extraction is an important research direction in data mining, natural language processing, and search engine optimization. The advantages of traditional methods of keywords extraction, based on statistics and linguistic rules, are domain-independent and do not require any training data. However, keywords generated by this method are not good enough to find all relevant documents. This paper proposes a new approach that exploits word-level handcrafted features and machine learning models to solve this task. To evaluate the proposed solution, we compare our results with the latest supervised and unsupervised methods for automatic keyword extraction.s. Experiment results show that our model achieves the best results on the 9/20 data corpus. It points out that our proposed approach is promising.*

Keywords: Feature extraction; Keyword extraction; Artificial neural network; Supervised learning.

1. INTRODUCTION

The advent of the internet has produced enormous digital documents, of which 80% is unstructured documents[1]. Keyword extraction aims to create a set of keywords representing text documents that can be used effectively in many tasks such as automatic indexing, clustering and filtering, search engine optimization, etc. There are a large number of documents on the internet without keywords. However, the manual assignment of high-quality keywords is costly, time-consuming, and error-prone. Because of that, automatic keyword extraction has been raised as an important task in natural language processing. Researches on automatic keyword extraction are mainly divided into two categories: *unsupervised* and *supervised learning*. The supervised methods consider this problem as a binary classification task, whereas the unsupervised ones are mostly based on TF-IDF [3], clustering, and graph-based ranking.

Unsupervised learning approaches usually use the term frequency-inverse document frequency, clustering, and graph-based to rank and select keywords. Rafiei et al.[1] and Kim [2] uses the TF.IDF weight to identify keywords in the single document. Research [4] selects three groups of six words with the highest values in the entire document, the longest sentence in each paragraph, and the title. Research [5,6,7] are graph-based ranking models applied to extract keywords. Novel unsupervised approaches, such as RaKUn [9] and YAKE! [8], work reasonably well, and have some advantages over supervised approaches, as they are language and genre independent. YAKE! [8] method implements keyword extraction based on each word's importance through feature extraction and employs a heuristic measure to determine their relevance. Based on phrase embedding, Key2Vec [13] propose processing text documents for training multi-word phrase embeddings for the

[1] https://lawtomated.com

thematic representation of scientific articles and ranking of keyphrases extracted from them using theme-weighted PageRank.

Existing supervised learning approaches typically extracting features from documents then apply machine learning algorithms. The tradition researches [10-14] have fed linguistic and lexical features as input into the machine learning algorithm such as Decision Tree, Naïve Bayes, K-nearest Neighbours, and Support Vector Machines (SVM). However, the drawback for the supervised learning method is that it depends on a large labelled dataset. The deep learning approaches are supervised learning approaches that are constructed by neural networks with multiple layers. Research [15] proposed a deep neural network model to extract keywords in the Chinese language. An extended Long short term memory(LSTM) model called center-based LSTM applies a self-attention mechanism to capture multi-aspects sentence-level information when determining whether given the word is a keyword. Very recently, two supervised models named CopyRNN and CatSeqD proposed by Meng et al. [16] and Yuan et al. [17]. Meng [16] employs an one-to-seq approach, named CopyRNN, where an input text is matched with a concatenated sequence made of all the keywords for a specific text. The CatSeqD [17] method is also a one-to-seq approach incorporated two diversity mechanisms (called semantic coverage and orthogonal regularization) into the model. These mechanisms constrain the over-all internal representation of a generated keyword sequence to be semantically similar to the overall meaning of the source text and therefore force the model to produce different keywords.

In general, the unsupervised methods have the advantage of not requiring any training data and can achieve results in any domain. A drawback of this approach is that the quality of the keywords achieved depends on the document's length and quality. On domain-specific data, supervised methods have shown better performances. However, the model used for training and prediction has many effects on the quality of the final keywords.

The method that we introduce in this paper is a combination of handcrafted feature extraction techniques and an artificial neural network model. We exploit the advantage of the unsupervised learning method by proposing a set of effectiveness features. Therefore, our result has achieved excellent performance and works much more efficiently than unsupervised algorithms based on simple word frequency statistics. Besides, the selection of the appropriate neural network model is also the advantage of the proposed solution.

The main contributions of the paper include:

- Propose a set of powerful features at the word level using for extracting keywords. The features we considered are *(1) Noun phrase; (2) Name Entity; (3) Trigram; (4) Length; (5) Position; (6) Spread; (7) Frequency; (8) TF.IDF; and (9) Casing.*

- Propose a binary classification artificial neural network model to extract keywords. The proposed model is independent on the language, length, and quality of the document.

The rest of this paper is structured as follows. We present our proposed approaches in Section 2, including pre-processing, feature extraction methods, and the model for query and filtering. Section 3 presents the results and a discussion on the results. Finally, our conclusion and future works will be introduced in Section 4.

2. PROPOSED METHOD

The approach proposed in this paper has three main steps illustrated in Figure 1: *(1) pre-processing, (2) features extraction, and (3) keywords extraction.* First, the document is pre-processed and building keyword candidates. Then, an input feature vector is created based on features defined in section 2.2. Finally, a ranking algorithm is applied to select top-k keywords. We explain and give a detailed description of each step implementations.

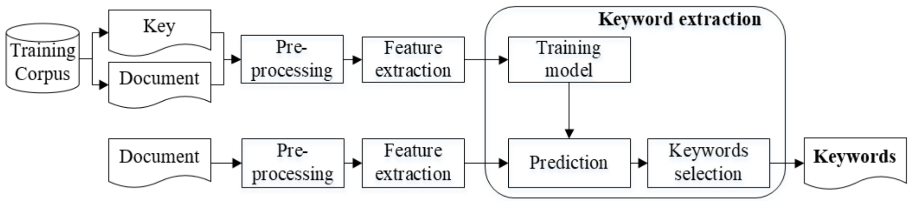

Figure 1. *Processing steps for keyword extraction*

2.1. Pre-processing

Stopwords are the most common words in any natural language. In analyzing and building natural language processing models, these stopwords might not add much value to the meaning of the document. Besides, based on the observation that numbers and special characters rarely play a role as keywords, but Noun Phrases, Named Entities, or repeated words play an important role in a sentence. Because of that, we remove stopwords, numbers, and special characters from the document. To create candidate keywords for a document, we extract the Noun phrases, Named Entities, and Trigrams which frequencies not less than a threshold *t*. After pre-processing, we obtain a set of candidate keywords.

2.2. Feature extraction

Given an input document *D* and its set of candidate keywords $U = (u_1,u_2,...,u_n)$, with *n* is the number of candidate keywords. We propose nine features to capture characteristics of each term, including: *(1) Noun phrase; (2) Name Entity; (3) Trigram; (4) Length; (5) Position; (6) Spread; (7) Frequency; (8) TF.IDF;* and *(9) Casing*. We use the *NLTK*[2] toolkit to extract Noun phrase, Name Entity, and Trigram. Each feature will be described for each candidate keyword u_i below.

2.2.1. Noun phrase

We extract all Noun phrases from the input document, tokenize and define a co-occurrence vector for each candidate keyword. Given *NP* is the set of words in Noun phrases, this feature is determined by Equation (1).

$$NP_i = \begin{cases} 1 \ if \ u_i \in NP \\ 0 \ otherwise \end{cases} \quad (1)$$

2.2.2. Named Entity

Named Entity Recognition (NER) is an information extraction method in which entities that are present in the text are classified into pre-defined entity types like "Person"," Place", "Organization", etc. We extract all the Named Entities of the text, tokenize and also define the co-occurrence vector for each candidate keyword. Given *NE* is the set of words in Named Entities, the Named Entity feature is determined by Equation (2).

$$NE_i = \begin{cases} 1 \ if \ u_i \in NE \\ 0 \ otherwise \end{cases} \quad (2)$$

2.2.3. Trigram

A trigram is a particular case of the n-gram, where n is 3. Looking at most frequent n-grams can give a better understanding of the context in which the word was used. We define *NG* as trigrams that appear at least *t* times[3], tokenize and define the co-occurrence vector

[2] http://www.nltk.org/
[3]The value of t will be discussed in Section 3.2

where the words appear in the NG are assigned the value 1, and the other position assigns a value of 0. The Trigram feature is determined by Equation (3).

$$NG_i = \begin{cases} 1 \; if \; u_i \in NG \\ 0 \; otherwise \end{cases} \tag{3}$$

2.2.4. Length

There are many studies related to word length and the effect of length on the importance of words. Sigurd et al. [18] studied the relationship between word length and word occurrence frequency in experimental data warehouses. Research results show that words with a length of 3 characters have the most frequency. NEW et. at [19] studied the effects of word length on word choice. Research results show that words with a length of 3-5 letters are used most commonly. In this work, we consider word length as a feature. We do not specify how long a word is essential. We combine this word length feature with others to train on the set of gold keywords to determine the word importance. We define each word's length as the number of characters of that word. The word length feature is determined by Equation (4).

$$LE_i = \frac{len(u_i)}{max\_length} \tag{4}$$

where *len(u_i)* is the number of characters of term u_i, and m*ax_length* is a pre-defined constant value.

2.2.5. Position

Another indicator of the importance of a candidate term is its position [10,20]. There are many approaches in determining the importance of words based on word position, such as the position of the sentence containing the word appearing [8] or the first position of the word appearing in the text [10]. Completing documents (such as articles and essays) usually consists of three sections: introduction, body, and conclusion. We realize that essential words often appear at the beginning and the conclusion of the document. Besides, the exact definition of each section of the text is approximate, so we define the Equation (5) to determine the word position weight based on the first occurrence relate to the middle position of the document.

$$PO_i = \frac{abs(first\_occurrencse(u_i) - L)}{\mathrm{L}} \tag{5}$$

where *first_occurrences(u_i)* return the first occurrences of term u_i; L is the middle position of the text. The *abs* function returns the absolute (positive) value of a number.

2.2.6. Spread

One more feature relates to the position where we consider the importance of a word. We found that the word to be important when it appears in many places in the text. In this case, the positions that interest us are the first and last appearance. We define Spread as the number of words between the first and last occurrence of a keyword divided by the total number of words in the text. The Spread feature is determined by Equation (6).

$$SP_i = \frac{last\_occurrences(u_i) - first\_occurrences(u_i)}{len(D)} \tag{6}$$

where *first_occurrences(u_i), last_occurrences(u_i)* return the first and last occurrences of term u_i, respectively, *len(D)* is the number of characters in document *D*.

2.2.7. Term frequency

Research [20] indicates that the frequency of term occurrence in a document furnishes a useful measurement of word significance. This state reflects the belief that the higher the frequency of a candidate term, the greater its importance. To prevent a bias towards high-frequency in long documents, we define the term frequency value of a candidate term u_i as the number of times the term u_i appears divided by the total number of candidate keywords. The Term frequency feature is determined by Equation (7).

$$TF_i = \frac{TF(u_i)}{\mathrm{n}} \tag{7}$$

2.2.8. TF-IDF

TF-IDF stands for Term Frequency and Inverse Term Frequency [3]. This method helps understand the importance of a word in a document. It is often used as a weighting factor in searches of information retrieval, text mining, and user modelling. In this paper, we use the TF-IDF value of a word as a weight to determine the word's importance in the document. The TF-IDF feature is determined by Equation (8).

$$TF.IDF_i = \; TF(u_i) \times IDF(u_i) \tag{8}$$

where *TF(u_i)* returns the term frequency of term u_i; *IDF(u_i)* returns the inverse document frequency of term u_i;

2.2.9. Casing

Research [8] stated that the underlying rationale is that uppercase terms tend to be more relevant than lowercase ones. The term that beginning with a capital letter or all the letters is capitals is considered a relevant term. However, we observe that such determination is not robust enough to evaluate plural abbreviations and chemical formulas. In this paper, we identify relevant words if there is at least one capital letter. The Casing feature is determined by Equation (9).

$$CA_i = \begin{cases} 1 \; if \; u_i \; include \; any \; capitalization \; character \\ 0 \quad otherwise \end{cases} \tag{9}$$

2.3. Keyword extraction

After extracting the nine feature vectors as above, we combine the feature vectors into a 2-dimensional matrix of size *n×9* where *n* is the number of candidate keywords, is defined as

$$F = \begin{pmatrix} NP_1 & NE_1 & NG_1 & LE_1 & PO_1 & SP_1 & TF_1 & TF.IDF_1 & CA_1 \\ NP_2 & NE_2 & NG_2 & LE_2 & PO_2 & SP_2 & TF_2 & TF.IDF_2 & CA_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ NP_n & NE_n & NG_n & LE_n & PO_n & SP_n & TF_n & TF.IDF_n & CA_n \end{pmatrix} \tag{10}$$

In the next step, the feature matrix *F* is used as the input for a binary classifier to predicts the probability of being an important word within the candidate keywords. To choose the best classifier for our task, we applied four classification models, including SVM, Naïve Bayes, ANN, and LSTM. The output values are in the range (0,1) for each word. The output values for all words are then sorted in decreasing order. The top-k candidate keywords will be chosen as the final keywords. Here the value of *k* depends on how many keywords needed to be extracted.

SVM and Naïve Bayes are two of the three binary classifiers we use to evaluate the effectiveness of proposed features. In this study, we use the *sklearn*[4] library for the SVM

[4] https://scikit-learn.org/

and Naïve Bayes classifiers. For the Naïve Bayes classifier, we use optimized default parameters from the *sklearn* library. With the SVM classifier, we experimented with different kernels and found when the kernel function is sigmoid, and the margin is default got the best results.

Next, we introduce in detail two more complex binary classification models, ANN and LSTM. Our ANN architecture for the binary classification is shown in Figure 2.

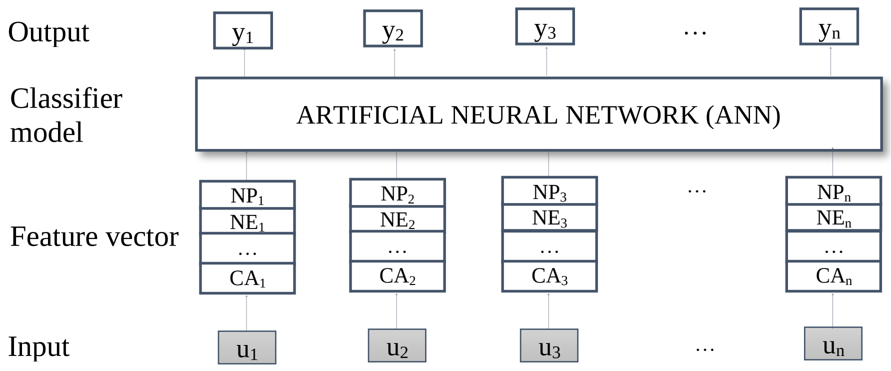

Figure 2. *Keyword extraction propose model*

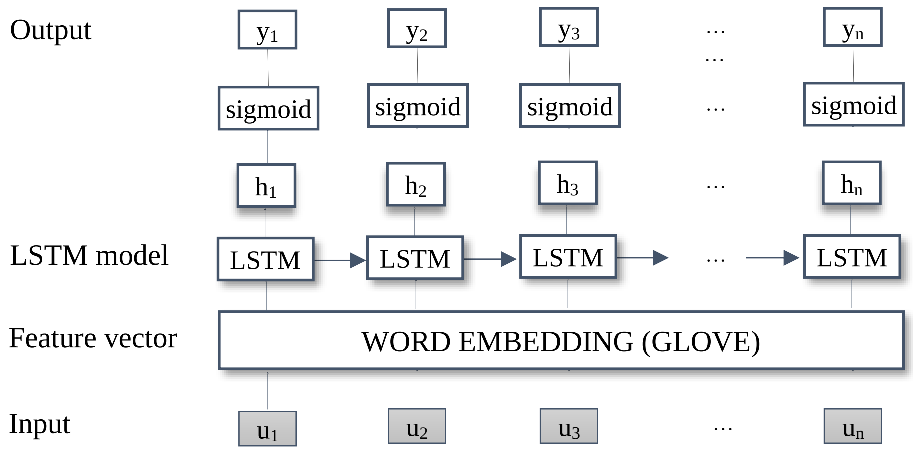

Figure 3. *The binary classification model based on LSTM*

Here u_i represents for the *i*-th word from the candidate keywords. $y = (y_1, y_2, \dots, y_n)$ is the output of the binary classification model ($0 \le y_i \le 1$), and *n* is the number of candidate keywords. The ANN model employed in our study is a feed-forward network with sigmoid activation functions in the hidden layers. According to Bishop [22], more than one hidden layer is often not necessary, so our architectures have only one hidden layer. The ANN is trained using a back-propagation algorithm with *Adam* optimizer, the *learning rate is 0.001*, and the *number of the epoch is 25***.**

The LSTM architecture for our binary classification is shown in Figure 3. For each word in the candidate keywords, we use Glove [23] size of 100 to create a word representation vector. The word representation matrix is reshaped into three dimensions as the input vector of the LSTM model. The specific values of the three-dimensional vector are: *batch_size* equal to the number of words, *time_steps* equal 1, and *seq_len* equal to the number of features (seq_len=100).

3. EXPERIMENT RESULTS AND DISCUSSION

The experiment conducted here aimed to evaluate the proposed method and to compare it with state-of-the-art methods. Our experiment includes two main phases. First, we experiment to evaluate the effectiveness of the proposed features set. Second, we evaluate the efficiency of the selected model. Evaluations are reported using standard metrics of precision (P), recall (R) and F1-score (F).

To evaluate the proposed solution's effectiveness, we carried out experiments over 20 different datasets [8]. These datasets have fully gold keywords that have been created or indexed by taggers.

To reduce the amount of computation, we create the candidate keywords based on words from the Named Entities, Noun Phrases, and Trigrams *(with optimized value t=5)* then use these candidate keywords in both training and prediction stages. In the training stage, we extend the candidate keywords by adding gold keywords.

In the first experiment phase, we divide each dataset into two parts in a 70/30 ratio for training and testing, respectively. We applied four machine learning models: SVM, Naïve Bayes, and ANN to choose the best one. The F1-score value for the top-10 keywords with each dataset is presented in Table 1. In the second experiment phase, we conduct experiments on data corpus in the English language. The LSTM classification model is used in this phase. The results present in the last column in Table 1.

Through the results shown in Table 1, we found that: *(1) Using the handcraft features obtain better results than the solution using the learned feature from the simple LSTM model; (2) The ANN model gives the best results among candidate models.* We select the best results *(the ANN column)* to compare with state-of-the-art approaches. In Table 2, we present a comparison between our results with other researches using the same datasets. It shows that our approach has a remarkable improvement comparing to the others.

Table 1. *The F1-score values for top-10 keywords*

#	**Corpus**	**SVM**	**Naïve Bayes**	**ANN**	**LSTM**
1	110-PT-BN-KP	0.176	0.263	**0.367**	0.194
2	500N-KPCrowd-v1.1	0.101	0.156	**0.201**	0.085
3	Cacic	0.013	0.152	**0.400**	0.023
4	Citeulike180	0.018	0.110	**0.226**	-
5	Fao30	0.016	0.122	**0.216**	0.044
6	Fao780	0.011	0.111	**0.272**	0.038
7	Inspec	0.348	0.393	**0.424**	0.380
8	Kdd	0.207	0.220	**0.249**	0.229
9	Krapivin2009	0.013	0.161	**0.433**	0.020
10	Nguyen2007	0.022	0.203	**0.407**	0.025
11	Pak2018	0.013	0.056	**0.085**	-
12	PubMed	0.017	0.116	**0.185**	0.122
13	Schutz2008	0.025	0.135	**0.278**	0.032
14	Semeval2010	0.017	0.147	**0.313**	0.015
15	SemEval2017	0.272	0.350	**0.362**	0.213
16	theses100	0.012	0.127	**0.270**	0.046
17	Wicc	0.013	0.214	**0.435**	-
18	Wiki20	0.014	0.122	**0.222**	0.097
19	WikiNews	0.086	0.298	**0.314**	0.115
20	WWW	0.233	0.246	**0.275**	-

* *The symbol "-" mean corpus is not in English*

Table 2. *Our approach compared to some other state-of-the-art results*

Dataset	YAKE!	Single Rank	KEA	Key2Vec	RAKUN	CopyRNN	CatSeq	Our
110-PT-BN-KP	**0.500**	0.275	0.215	-	-	-	-	0.367
500N-KPCrowd-v1.1	0.173	0.157	0.159	-	**0.428**	-	-	0.201
Inspec	0.316	0.378	0.150	**0.486**	0.054	0.289	0.333	0.424
Krapivin2009	0.170	0.097	0.171	-	-	0.266	0.285	**0.433**
Nguyen2007	0.256	0.158	0.221	-	0.096	-	-	**0.407**
PubMed	0.106	0.039	**0.216**	-	0.075	-	-	0.185
Schutz2008	0.196	0.086	0.182	-	**0.418**	-	-	0.278
WWW	0.172	0.097	0.072	-	-	-	-	**0.275**
KDD	0.156	0.085	0.063	-	0.046	-	-	**0.249**
SemEval2010	0.211	0.129	0.215	0.290	0.091	0.318	**0.366**	0.313
SemEval2017	0.329	**0.449**	0.201	-	0.112	-	-	0.362
Cacic	0.196	0.087	0.155	-	-	-	-	**0.400**
Citeulike180	0.256	0.066	**0.317**	-	0.250	-	-	0.226
Fao30	0.184	0.066	0.139	-	**0.233**	-	-	0.216
Fao780	0.187	0.085	0.114	-	0.094	-	-	**0.272**
Pak2018	**0.086**	0.022	0.043	-	-	-	-	0.085
Theses100	0.111	0.060	0.104	-	0.069	-	-	**0.270**
Wicc	0.256	0.133	0.167	-	-	-	-	**0.435**
Wiki20	0.162	0.038	0.134	-	0.190	-	-	**0.222**
WikiNews	**0.450**	0.248	0.248	-	-	-	-	0.314

** The best results are highlighted in bold*

** The symbol "-" mean the result is not published*

Table 1 presents the results of the two-phase experiment of the classification algorithm on 20 corpora. Table 2 compares the results of the proposed method with the published methods using the same experiment dataset. The best (the highest) results obtained by a particular keyword extraction method are indicated as the only boldface. Table 2 shows that our system's results are reached the best on the 9/20 data corpus. It indicates that our system is suitable for many different corpora types regardless of the language or length of the document and achieves better results for corpus with many documents (such as Krapivin2009, Nguyen2007, or Cacic). The results prove that our proposed feature set and the ANN model are great promise solutions for automatic keyword extraction problems.

3.3. Error Analysis

The errors in the output of the system can be divided into the following types:

- Data size is a very crucial part of training neural networks. Large datasets can help us better learn model parameters and improve the optimization process and imparts generalization. In our case, several corpora have small data such as Wiki20 (20 documents), Fao30 (30 documents), and Pak2018 (50 documents). We need many iterations before finding the optimum values, but the predicted result is quite low because of overfitting results.

- Some document files have no gold keywords, or some gold keywords do not belong to the content file *(34/437 files in WWW corpus; 10/15 files in the Pak2018 corpus)*. These errors are the cause of the F1-score decrease.

4. CONCLUSION

Keyword extraction algorithms have become critical components in many computer science applications. This paper proposed an approach that uses feature extraction techniques and an artificial neural network model for automatic keywords extraction. By experimenting with different machine learning models, ANN stands out to be the best one. The key to the paper's success is the appropriate selection of features and the classification model. The proposed method was evaluated using the popular keyword extracting corpora and widely accepted evaluation metrics: *precision*, *recall*, and *F1-score*. The solution achieves the best results on 9/20 corpus compared to state-of-the-art researches.

Through testing, we recognize some weaknesses of the proposed solution. We plan to improve F1-score by using semantic information in the future model.

REFERENCES

[1]. Rafiei, Javad, et al. *"Source retrieval plagiarism detection based on noun phrase and keyword phrase extraction."* PAN (2015).

[2]. Kim, Su Nam, et al. *"Semeval-2010 task 5: Automatic keyphrase extraction from scientific articles."* Proceedings of the 5th International Workshop on Semantic Evaluation. 2010.

[3]. Chowdhury, Gobinda G. *"Introduction to modern information retrieval."* Facet publishing, 2010.

[4]. Suchomel, Simon, and Michal Brandejs. *"Heterogeneous Queries for Synoptic and Phrasal Search."* CLEF (Working Notes). 2014.

[5]. Mihalcea, Rada, and Paul Tarau. *"Textrank: Bringing order into text."* Proceedings of the 2004 conference on empirical methods in natural language processing. 2004.

[6]. Page, Lawrence, et al. *"The PageRank citation ranking: Bringing order to the web."* Stanford InfoLab, 1999.

[7]. Bougouin, Adrien, Florian Boudin, and Béatrice Daille. *"Topicrank: Graph-based topic ranking for keyphrase extraction."* 2013.

[8]. Campos, R., Mangaravite, V., Pasquali, A., Jorge, A., Nunes, C., & Jatowt, A. (2020). "*YAKE! Keyword extraction from single documents using multiple local features."* Information Sciences, 509, 257–289. Elsevier.

[9]. Škrlj, Blaž, Andraž Repar, and Senja Pollak. *"RaKUn: Rank-based Keyword extraction via Unsupervised learning and Meta vertex aggregation."* International Conference on Statistical Language and Speech Processing. Springer, Cham, 2019.

[10]. Hulth, Anette. *"Improved automatic keyword extraction given more linguistic knowledge."* Proceedings of the 2003 conference on Empirical methods in natural language processing. 2003.

[11]. Dietterich, Thomas G. *"Ensemble methods in machine learning."* International workshop on multiple classifier systems. Springer, Berlin, Heidelberg, 2000.

[12]. Beliga, Slobodan. *"Keyword extraction: a review of methods and approaches."* University of Rijeka, Department of Informatics, Rijeka (2014): 1-9.

[13]. Kovačević, Aleksandar, et al. *"Automatic extraction of metadata from scientific publications for CRIS systems."* Program (2011).

[14]. Witten, Ian H., et al. *"Kea: Practical automated keyphrase extraction."* Design and Usability of Digital Libraries: Case Studies in the Asia Pacific. IGI global, 2005. 129-152.

[15]. Zhang, Yu, et al. *"Keywords extraction with deep neural network model."* Neurocomputing 383 (2020): 113-121.

[16]. Meng, Rui, et al. *"Does order matter? an empirical study on generating multiple keyphrases as a sequence."* arXiv preprint arXiv:1909.03590 (2019).

[17]. Yuan, Xingdi, et al. *"One Size Does Not Fit All: Generating and Evaluating Variable Number of Keyphrases."* arXiv preprint arXiv:1810.05241 (2018).

[18]. Sigurd, Bengt, Mats Eeg-Olofsson, and Joost Van Weijer. *"Word length, sentence length and frequency–Zipf revisited."* Studia Linguistica 58.1 (2004): 37-52.

[19]. New, Boris. *"Reexamining the word length effect in visual word recognition: New evidence from the English Lexicon Project."* Psychonomic bulletin & review 13.1 (2006): 45-52.

[20]. Graves, Alex, Abdel-rahman Mohamed, and Geoffrey Hinton. *"Speech recognition with deep recurrent neural networks."* 2013 IEEE international conference on acoustics, speech and signal processing. IEEE, 2013.

[21]. V. N. Vapnik. *"The Nature of Statistical Learning Theory."* Springer, New York, 2 nd edition, 2000

[22]. Bishop, Christopher M. *"Neural networks for pattern recognition."* Oxford university press, 1995.

[23]. Pennington, Jeffrey, Richard Socher, and Christopher D. Manning. *"Glove: Global vectors for word representation."* Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP). 2014.

TÓM TẮT

TỰ ĐỘNG TRÍCH RÚT TỪ KHÓA SỬ DỤNG MẠNG NƠ RON NHÂN TẠO VÀ KỸ THUẬT TRÍCH RÚT ĐẶC TRƯNG

Sự phát triển của internet và thư viện số đã tạo ra những tài liệu điện tử đồ sộ mà việc phân tích thủ công những tài liệu này không khả thi. Để có khái niệm đầu tiên về các tài liệu này và tìm kiếm các tài liệu liên quan thì việc trích xuất từ khóa từ các tài liệu này là rất quan trọng. Tuy nhiên, việc trích rút một số hữu hạn từ khóa phục vụ cho việc tìm kiếm nhằm thu được tất cả tài liệu liên quan vẫn đang là một thách thức. Tự động trích rút từ khóa là một hướng nghiên cứu quan trọng trong lĩnh vực khai thác dữ liệu, xử lý ngôn ngữ tự nhiên và tối ưu hóa công cụ tìm kiếm. Các phương pháp trích rút từ khóa truyền thống dựa trên phương pháp thống kê và đặc trưng ngôn ngữ có ưu điểm là không phụ thuộc vào miền dữ liệu và không yêu cầu bất kỳ dữ liệu huấn luyện. Tuy nhiên, các từ khóa được tạo bằng phương pháp này không đủ tốt để tìm tất cả các tài liệu liên quan. Bài báo này đề xuất một cách tiếp cận mới khai thác thế mạnh của kỹ thuật trích rút đặc trưng thực hiện ở cấp độ từ và mô hình học máy để giải quyết bài toán này. Để đánh giá hiệu quả của giải pháp đề xuất, chúng tôi so sánh kết quả của mình với các phương pháp trích rút từ khóa dựa trên phương pháp học có giám sát và không giám sát mới nhất hiện nay. Kết quả thử nghiệm cho thấy mô hình của chúng tôi đạt được kết quả tốt nhất trên 9/20 kho dữ liệu thử nghiệm. Điều đó chỉ ra rằng phương pháp đề xuất của chúng tôi hoàn toàn khả thi.

Từ khóa: Trích rút đặc trưng; Trích rút từ khóa; Mạng nơ ron nhân tạo; Học có giám sát.

Received date, 13th April, 2016
Revised manuscript, 10th May, 2016
Published, 09th June, 2016

Author affiliations:

[1] Institute of Information Technology, Institute of Military Science and Technology Ministry of Defence;

[2] School of Information and Computer Science Technology Hanoi University of Science and Technology Hanoi;

*Corresponding author: sonnv78@gmail.com