Totally Automated Keyword Extraction

Tayfun Pay
Graduate Center of New York
New York, USA
tpay@gradcenter.cuny.edu

***Abstract*—We develop and analyze an unsupervised and domain-independent method for extracting keywords from single documents. Our approach differs from the previous ones in the way of identifying candidate keywords, pruning the list of candidate keywords with several filtering heuristics and selecting keywords from the list of candidate keywords according to dynamic threshold functions. We were able to obtain better precision and recall on the same data set with our method than the previously studied ones.**

***Keywords*-data mining; text mining; text analysis;**

I. Introduction

Keywords consist of sequence of one or more words and are concise representation of important facts in documents. Keywords can be, utilized to classify a given document [1], employed in journal articles for indexing [2], or used as search queries in search engines [3]. However, most documents do not contain any assigned keywords, whereas the ones that do are usually restricted to certain keywords and rely on the perception of the curator. Therefore, the importance of automatically extracting keywords from documents, a branch of data mining, has been becoming increasingly important in the data driven world of the 21^{st} century. There have been numerous studies done during the recent years, such as the ones in [4], [5], [6], and [7], that sought for a fast, credible and selective process to extract keywords.

Extracting keywords from a document usually involves several steps. First, candidate keywords need to be identified with some approach, such as n-grams. After that, the decisive features need to be calculated, such as TF-IDF. Then, depending on whether the method is supervised or unsupervised, either the learning algorithm is executed with the training data and the keywords are extracted from the test data accordingly, or some type of a threshold function is used to select the top n ranking candidate keywords.

One of the pioneer works in this field was presented in [8], where a supervised approach was employed to extract keywords from titles and abstracts of journal articles. It was shown in [8] that adding linguistic knowledge on top of statistics provides better results when the extracted keywords are compared to the manually assigned keywords.

Then an unsupervised method called TextRank was proposed in [9], which has the PageRank algorithm of [10] embedded in its core. This method creates a text graph, where the nodes are single words and the edges among them are created according to the specified co-occurrence window. It has a post-processing step that creates keywords with multiple words by consulting the original text. The TextRank approach in [9] with several different features provided better precision and recall on the same data set than the various approaches that were proposed in [8].

More recently, another unsupervised method called RAKE was detailed in [11]. This approach employs a stop-list to obtain a list of candidate keywords from a document. Then it calculates certain metrics on candidate keywords before selecting the keywords according to the threshold function. The approach in [11] with several different stop-lists provided better precision on the same data set than the ones in [8] and [9]. Even though, they implicitly called their approach "corpus independent" in [11], they still obtained their best results by using a stop-list generated from the training data set in [8].

What we propose is an unsupervised and domain-independent method for extracting keywords from single documents. First, we employ a straightforward approach to obtain candidate keywords. Then we employ several filtering heuristics to eliminate as many candidate keywords as possible that could have returned false-positives. We still use the same approach in [11] to calculate the features. However, we employ dynamic threshold functions to decide which candidate keywords are selected as keywords. We call our method totally automated keyword extraction (TAKE).

We employed TAKE to extract keywords from the test data set in [8], which was also carried out in [9] and [11]. TAKE produced better precision and recall on this data set than the previously studied methods. Then we created our own data set from the titles and abstracts of journal articles in [12], where TAKE once again provided superior results. Overall, we regard TAKE to readily perform well on other short and precise documents, such as memos, emails, social media comments and textual customer feedback.

II. Our Methods

Our totally automated keyword extraction (TAKE) method is divided into the following four steps: candidate keyword extraction, filtering of candidate keywords, feature calculation of candidate keywords and selection of keywords from

candidate keywords using the threshold function.

A. Candidate Keyword Extraction

Most of the important facts in documents can be summarized by $nouns$ along with $adjectives$ that depict them. Therefore, we construe that any candidate keyword is a noun-phrase. Then we define a noun-phrase as any sequence of words that consist of at least one noun preceded by zero or more adjectives, (<JJ*><NN+>), with no restriction on the number of words. We employ the default part-of-speech tagger in the NLTK library [13], without any training, to obtain the respective part of speech tag of each word in the given text. Then we extract all of the noun-phrases as candidate keywords for the document. We also consider one special case, two adjacent noun-phrases (< JJ*><NN+> <JJ*><NN+>) as candidate keywords. We should make a special note that we consider all types of $nouns$ (NN, NNS, NNP or NNPS) and $adjectives$ (JJ, JJR or JJS), where the pos-tag abbreviations are from the NLTK library [13].

There are of course several setbacks to this naive approach, one of being the structure of the candidate keywords. For example, a candidate keyword that has the structure NN-of-JJ-NN might be more meaningful in the context of the given document. Another impediment is that we are heavily dependent on the pos-tagger to correctly identify the part-of-speech tag of each word in the context of the given document. For example, the pos-tagger might identify a $noun$ as a $verb$ and this might cause us to totally miss that candidate keyword or extract a shorter candidate keyword.

B. Filtering Heuristics

There is no statistical significance of candidate keywords that have a frequency of one, especially if these candidate keywords consist of a single word. Most of these candidate keywords, if not all, need to be removed from the list of candidate keywords by observing some other features. The most natural thing to do is to look at the location of these type of candidate keywords in the given document.

- Location Based Filtering Heuristic: If a candidate keyword with a frequency of one and word count of one does not appear in the beginning 10% of the text, then we remove it from the list of candidate keywords.

The location based filtering heuristic can of course be revised to explicitly include the first sentence or some fixed number of sentences. Furthermore, it can also be amended to include some arbitrary sections of the text, such as the middle and/or the end of the text. There is obviously no limitation to the way this heuristic can be composed. We will use the above definition because this precise interpretation works well when performing keyword extraction from titles and abstracts of journal articles.

Another way to filter out insignificant candidate keywords would be to consult a stop-list, such as the one compiled in [14]. Stop-list is a collection of the most frequently occurring words in a corpus and/or a language, where these words would not be meaningful to include in as a keyword in most context, such as the words the, of, a, in, an and etc.

- Stop-List Based Filtering Heuristic: We go through the list of candidate keywords and remove any candidate keyword that contains a word from the fox stop-list [14].

The stop-list based filtering heuristic can also be revised to include more than one stop-list or a stop-list constructed from a given corpus. However, the latter approach would make the method domain-dependent.

C. Feature Calculation

We employ the same methodology as in [11] for feature calculation.

- Frequency: The total number of occurrences of the given $word$ in candidate keywords.
- Degree: The total number of co-occurrences of the given $word$ with other words in candidate keywords.
- Word-Score: Degree/Frequency
- Candidate-Keyword-Score: Sum of the Word-Score of all of the $words$ that make up the candidate keyword.

These metrics are best illustrated with an example. Let's assume that the candidate keywords for some document are as follows: "*robotic search algorithms, image processing algorithms, image processing, approximation algorithms, randomized algorithms, machine learning, artificial intelligence, perception heuristics, computer vision, image, robotic, artificial, computer, perception, vision* " . Table I shows the calculated metrics of each word in these candidate keywords.

Table I
FREQUENCY, DEGREE AND SCORES OF EACH WORD

$word$	$frequency$	$degree$	$wordscore$
robotic	2	4	2
search	1	3	3
algorithms	4	10	2.5
image	3	6	2
processing	2	5	2.5
approximation	1	2	2
randomized	1	2	2
artificial	2	3	1.5
intelligence	1	2	2
machine	1	2	2
learning	1	2	2
perception	2	3	1.5
heuristics	1	2	2
computer	2	3	1.5
vision	2	3	1.5

The $words$ that appear often among candidate keywords have a higher frequency and the $words$ that appear in longer

and/or many candidate keywords have a higher degree. The word score metric combines and to a certain extent normalizes the frequency and degree metrics. However, the greedy presumption of more the better is still embedded in the word score metric. Nevertheless, Table II shows the candidate keyword scores as calculated by consulting the rightmost column in Table I.

Table II
SCORE OF EACH CANDIDATE KEYWORD

candidate keyword	*candidate keyword score*
robotic search algorithms	7.5
image processing algorithms	7
image processing	4.5
approximation algorithms	4.5
randomized algorithms	4.5
machine learning	4
artificial intelligence	3.5
perception heuristics	3.5
computer vision	3
image	2
robotic	2
artificial	1.5
computer	1.5
perception	1.5
vision	1.5

D. Threshold Functions

The methods in [8], [9] and [11] employ a static threshold function that selects the top 1/3 highest scoring candidate keywords as keywords for the given document. If we look at the example in the previous section, this method would select the top five candidate keywords as keywords. This static way of doing keyword selection from candidate keywords undermines the contextual properties of each document, such that it treats each document in the same manner. However, it could very well be the case that for a document with twenty candidate keywords, we should select fifteen of them as keywords, whereas for a document with thirty candidate keywords, we should select ten of them as keywords. Therefore, we propose two alternate dynamic threshold functions.

- Mean-Threshold: Calculate the mean of the candidate keyword-scores and select all candidate keywords that score higher than the mean as keywords.
- Median-Threshold: Calculate the median of the candidate keyword-scores and select all candidate keywords that score higher than the median as keywords.

If we once again consult the example in the previous section, the Mean-Threshold function would select the top eight candidate keywords as keywords for the document, since the mean is 3.46; and the Median-Threshold function would select the top six candidate keywords as keywords for the document, since the median is 3.5.

III. ANALYSIS

We used the same data set that was introduced in [8] and then used in [9] and [11] for evaluating their keyword extraction methods, which provided us an unambiguous comparison with these methods. This data set contains 2000 titles and abstracts for journal papers from Computer Science and Information Technology, which is divided into a training set, validation set and testing set that contain 1000, 500, and 500 documents, respectively. As our method is unsupervised, like the ones in [9] and [11], we only used the 500 abstracts from the testing set. The only modification we had to do was to add a period at the end of the abstracts because all of their last sentences were missing one. This was necessary for the pos-tagger to work properly.

The second data set was constructed from the one that was introduced in [12] that contains 2304 journal papers from Association for Computing Machinery. We wanted to preserve the same procedure as before, so we selected 1080 papers that contained more than half of the manually assigned keywords in their abstracts and only used the titles and abstracts of these papers during our evaluation.

For both of these data sets, we tested to see if we were able to absolutely extract the manually assigned keywords with our method TAKE. For example, if one of the manually assigned keywords for the document was "$machine\ learning\ algorithms$" and if we only extracted "$machine\ learning$", then this was counted as a miss. However, if we extracted "$machine\ learning\ algorithm$", that is without the last s, then this was counted as a hit. We also removed all of the brackets, parentheses and similar special characters and changed all of the upper case characters into lower case characters. We did not perform any type of stemming.

We employed both of our dynamic threshold functions and observed the following parameters: extracted keywords, correct keywords, precision, recall and f-measure. Precision tells us how many of the extracted keywords are correct and is calculated as (correct-keywords/extracted-keywords). Recall tells us how many of the manually assigned keywords are detected and is calculated as (correct-keywords/manually-assigned-keywords). As noted in [8], [9] and [11] both the precision and recall are equally important, so the f-measure is calculated as (2*precision*recall/precision+recall).

In the first data set, the 500 titles and abstracts from the testing set in [8], there are 4912 manually assigned keywords of which only 3837 are present in the titles and abstracts. In the second data set, there are 6021 manually assigned keywords of which only 3905 are present in the titles and abstracts. We will use the total number of manually assigned keywords in the calculation of recall and f-measure, so the highest achievable recall for the first and the second data sets are 78.1 and 64.9, respectively. We should note that the recall and f-measure calculations in [8] were done with the

total number of manually assigned keywords present in the abstracts, which makes them inherently higher and skewed.

Table III
PRECISION RECALL AND F-MEASURE FOR DATA SET 1

method	*precision*	*recall*	*f-measure*
TAKE - (T=Mean)	**50.4**	33.7	40.4
TAKE - (T=Median)	44.3	**46.9**	**45.6**
RAKE - (ka-stoplist) [11]	33.7	41.5	37.2
RAKE - (fox-stoplist) [11]	26.0	42.2	32.1
TextRank (UnD. w=2) [9]	31.2	43.1	36.2
TextRank (UnD. w=3) [9]	28.2	38.6	32.6
Supervised (Ngram w/tag) [8]	25.2	*51.7*	*33.9*
Supervised (Chunk w/tag) [8]	29.7	*37.2*	*33.0*
Supervised (Pattern w/tag) [8]	21.7	*39.9*	*28.1*

Table IV
EXTRACTED AND CORRECT KEYWORDS FOR DATA SET 1

method	*extracted*	*correct*
TAKE - (T=Mean)	3279	1653
TAKE - (T=Median)	5197	2304
RAKE - (ka-stoplist) [11]	6052	2037
RAKE - (fox-stoplist) [11]	7893	2054
TextRank (UnD. w=2) [9]	6784	2116
TextRank (UnD. w=3) [9]	6715	1897
Supervised (Ngram w/tag) [8]	7815	1973
Supervised (Chunk w/tag) [8]	4788	1421
Supervised (Pattern w/tag) [8]	7012	1523

A. Data Set 1

Tables III and IV illustrate the performance of TAKE compared to the other keyword extraction methods presented in [8], [9] and [11] on data set 1. When TAKE is employed with the Mean threshold function, we achieved the best precision and better f-measure compared to any of the previously studied methods. Even though, the total number of correct keywords is the lowest when we used TAKE with the Mean threshold function, they are very explicit. On the other hand, when TAKE is employed with the Median threshold function, we obtained a better precision, recall and evidently better f-measure compared to any of the previously examined methods. The total number of correct keywords is also the highest when we used TAKE with the Median threshold function.

B. Data Set 2

Tables V and VI illustrate the performance of TAKE compared to keyword extraction method presented in [11] on data set 2. Once again, the best precision was achieved by using TAKE with the Mean threshold function and the best recall was achieved by using TAKE with the Median threshold function. However, this time the f-measures were similar, where TAKE with the Mean threshold function had a small edge. We also tested RAKE with the fox-stop list, where we obtained a considerably low precision, but comparable recall, just like in the previous data set.

Table V
PRECISION, RECALL AND F-MEASURE FOR DATA SET 2

method	*precision*	*recall*	*f-measure*
TAKE - (T = Mean)	**28.9**	32.8	**30.7**
TAKE - (T = Median)	21.6	**42.5**	28.6
RAKE - (fox-stoplist)	16.9	41.0	23.9

Table VI
EXTRACTED AND CORRECT KEYWORDS FOR DATA SET 2

method	*extracted*	*correct*
TAKE - (T = Mean)	6825	1972
TAKE - (T = Median)	11839	2561
RAKE - (fox-stoplist)	14626	2474

IV. CONCLUSION

In this paper, we detailed a totally automated keyword extraction method that is unsupervised and truly domain independent. Even though, our method performed better than the previously studied methods on established benchmarks [8], [9], [11], there are of course rooms for improvement. Furthermore, if we decide to extend this method to extract keywords from full journal articles, rather than just titles and abstracts of journal articles, the filtering heuristics as well as the dynamic threshold functions that we employed would need to be revisited. However, we expect our method to still perform well on short and concise documents, such as memos, emails, social media comments and textual customer feedback. Keyword extraction is one of the first crucial steps in making sense of the given data. Extracting more quality keywords from documents would intrinsically improve the subsequently employed machine learning and data mining techniques, such as the ones in [15], [16], [17], [18], [19].

REFERENCES

[1] K. M. Hammaouda, D. N. Matute, and M. S. Kamel, "Corephrase: Keyphrase extraction for document clustering." *International Workshop on Machine Learning and Data Mining in Pattern Recognition.*, pp. 265–274, 2005.

[2] S. N. Kim and M. Y. Kan, "Re-examining automatic keyphrase extraction approach in scientific articles." *Proceedings of the workshop on multiword expressions: Identification, interpretation, disambiguation and applications.*, pp. 7–16, 2009.

[3] Z. Gong and Q. Liu, "Improving keyword based web image search with visual feature distribution and term expansion." *Knowledge and Information Systems.*, vol. 21, pp. 113–132, 2009.

[4] Y. Matsuo and M. Ishizuka, "Keyword extraction from a single document using word co-occurrence statistical information." *International Journal on Artificial Intelligence Tools* ., vol. 13, pp. 157–169, 2004.

[5] C. Wang, L. Zhang, M.and Ru, and S. Ma, "An automatic online news topic keyphrase extraction system." *Proceedings of the 2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology.*, pp. 214–219, 2008.

[6] P. Whitney, D. Engel, and N. Cramer, "Mining for surprise events within text streams." *Proceedings of the Ninth SIAM International Conference on Data Mining.*, pp. 617–627, 2009.

[7] G. Bordea and B. P., "Deriunlp: A context based approach to automatic keyphrase extraction." *Proceedings of the 5th International Workshop on Semantic Evaluation.*, pp. 146–149, 2010.

[8] A. Hulth, "Improved automatic keyword extraction given more linguistic knowledge." *Proceedings of the 2003 conference on empirical methods in natural language processing.*, pp. 216–223, 2003.

[9] R. Mihalcea and P. Tarau, "Textrank: Brining order into texts." *Association for Computational Linguistics.*, 2004.

[10] S. Brin and L. Page, "The anatomy of a large-scale hypertextual web search engine." *Computer Networks and ISDN Systems.*, vol. 30, pp. 1–7, 1998.

[11] S. Rose, D. Engel, N. Cramer, and W. Cowley, "Automatic keyword extraction from individual documents." *Text Mining.*, pp. 1–20, 2010.

[12] M. Krapivin, A. Aliaksandr, and M. Maurizio, "Large dataset for keyphrases extraction." *University of Trento*, 2009.

[13] S. Bird and E. Loper, "Nltk: the natural language toolkit." *ETMTNLP 02 Proceedings of the ACL-02.*, vol. 1, pp. 63–70, 2002.

[14] C. Fox, "A stop list for general text." *ACM SIGIR Forum.*, vol. 24, pp. 19–21, 1989.

[15] S. J. Pan, X. Ni, J. Sun, Q. Yang, and Z. Chen, "Cross-domain sentiment classification via spectral feature alignment." *WWW10 Proceedings of the 19th international conference on world wide web.*, pp. 751–760, 2010.

[16] Y. Liu, D. Xu, W. I. Tsang, and J. Luo, "Textual query of personal photos facilitated by large-scale web data." *IEEE Transactions on Pattern Analysis and Machine Intelligence.*, vol. 33, pp. 1022–1036, 2011.

[17] S. J. Pan and Q. Yang, "A survey on transfer learning." *IEEE Transactions on Knowledge and Data Engineering.*, vol. 22, pp. 1345–1359, 2010.

[18] M. Shao, D. Kit, and Y. Fu, "Generalized transfer subspace learning through low-rank constraint." *International Journal of Computer Vision.*, vol. 109, pp. 74–93, 2014.

[19] Z. Ding, M. Shao, and Y. Fu, "Missing modality transfer learning via latent low-rank constraint." *IEEE Transactions on Image Processing.*, vol. 24, pp. 4322–4334, 2015.