A Text Feature Based Automatic Keyword Extraction Method for Single Documents

Ricardo Campos[1,2(✉)], Vítor Mangaravite[2], Arian Pasquali[2], Alípio Mário Jorge[2,3], Célia Nunes[4], and Adam Jatowt[5]

[1] Polytechnic Institute of Tomar, Tomar, Portugal
ricardo.campos@ipt.pt
[2] LIAAD – INESC TEC, Porto, Portugal
{vima,arrp}@inesctec.pt
[3] DCC – FCUP, University of Porto, Porto, Portugal
amjorge@fc.up.pt
[4] University of Beira Interior, Covilhã, Portugal
celian@ubi.pt
[5] Kyoto University, Kyoto, Japan
adam@dl.kuis.kyoto-u.ac.jp

Abstract. In this work, we propose a lightweight approach for keyword extraction and ranking based on an unsupervised methodology to select the most important keywords of a single document. To understand the merits of our proposal, we compare it against RAKE, TextRank and SingleRank methods (three well-known unsupervised approaches) and the baseline TF.IDF, over four different collections to illustrate the generality of our approach. The experimental results suggest that extracting keywords from documents using our method results in a superior effectiveness when compared to similar approaches.

Keywords: Keyword extraction · Information extraction · Feature extraction

1 Introduction and Related Work

With the massive explosion of data, manually processing documents turned out to be an impossible task. As a direct consequence, several automatic solutions have emerged over the last few years, some following a supervised approach, of which a well-known example is KEA [11], others following an unsupervised methodology [5–7, 9], with TextRank [7], Rake [8], and SingleRank [10] being probably the most well-known solutions. Although the above-mentioned works offer first insights into how this problem can be answered, the task of extracting keywords is yet to be solved. In this work, we present an alternative approach that attempts to overcome the results of the above-mentioned works, while not being dependent on an external source or on linguistic tools. We follow an unsupervised methodology supported by a heuristic approach, which can easily scale to different collections, domains, and languages in a short time span. Our contributions are as follows: (1) we propose an unsupervised keyword extraction method named Yake! which builds upon text statistical features, to extract keywords (both single-word and multi-word terms) from single documents,

G. Pasi et al. (Eds.): ECIR 2018, LNCS 10772, pp. 684–691, 2018.
https://doi.org/10.1007/978-3-319-76941-7_63

thus without the need to rely on a document collection; (2) YAKE! may work for domains and languages for which there are no ready methods as it neither requires a training corpora, nor it depends on any external sources (such as WordNet) or linguistic tools (e.g., NER or PoS taggers).

2 YAKE! Architecture

The proposed method has four main components: (1) Text pre-processing; (2) Feature extraction; (3) Individual term weighting; (4) Candidate keywords generation.

2.1 Text Pre-processing

In the text pre-processing phase, we apply a tokenization process which splits the text into individual terms whenever an empty space or a special character (e.g., brackets, comma, period, etc.) delimiter is found.

2.2 Feature Extraction

Second, we devise a set of five features to capture the characteristics of each individual term. Although these features may be applied to any language, they are particularly suited to Western ones for which some characteristics we devise are particularly tuned. This is the case of *word casing*, which, in Western languages, reflects an important signal of a word. The features we considered are: (1) *Casing*; (2) *Word Position*; (3) *Word Frequency*; (4) *Word Relatedness to Context*; and (5) *Word DifSentence*. A more detailed study of the individual contribution of each of these features, will be conducted in the future. In the following we shortly describe each one of them.

2.2.1 Casing (W_{Case})

In this work, we give particular attention to any word starting with a capital letter (excluding ones at the beginning of sentences) or to any acronym (that is, where all letters of the word are capital) under the assumption that these words tend to be more relevant. Instead of counting them twice we only consider the maximum occurrence within the two of them. Equation 1 reflects this casing aspect:

$$W_{Case} = \frac{\max(TF(U(w)), TF(A(w)))}{\log_2(TF(w))} \quad (1)$$

where $TF(U(w))$ is the number of times the candidate word w starts with an uppercase letter, $TF(A(w))$ is the number of times the candidate word w is marked as an acronym and $TF(w)$ is the frequency of *w*.

2.2.2 Word Position ($W_{Position}$)

Considering the positions of the sentence where the word occurs may be an important feature for the keyword extraction process as the early parts of documents (especially, scientific and news publications) tend to contain a high rate of relevant keywords. We calculate this weight using the following Equation:

$$W_{Position} = \log_2(\log_2(2 + Median(Sen_w))) \quad (2)$$

where Sen_w indicates the positions of the set of sentences where the word w occurs, and Median is the median function. The result is an increasing function, where values tend to increase smoothly as words are positioned at the end of the document, meaning that the more often a word occurs at the beginning of a document the less its $W_{Position}$ value. Conversely, words positioned more often at the end of the document (likely less relevant) will be given a higher $W_{Positional}$ value. Note that a value of 2, is considered in the equation to guarantee that $W_{Positional} > 0$.

2.2.3 Word Frequency (W_{Freq})

This feature indicates the frequency of the word w within the document. It reflects the belief that the higher the frequency, the more important the word is. To prevent a bias towards high-frequency in long documents the TF value of a word w is divided by the mean of the frequencies (MeanTF) plus one time their standard deviation (σ) as in Eq. 3. Our purpose is to score all those words that are above the mean of the terms (balanced by the degree of dispersion given by the standard deviation).

$$W_{Freq} = \frac{TF(w)}{MeanTF + 1 * \sigma} \quad (3)$$

2.2.4 Word Relatedness to Context (W_{Rel})

W_{Rel} quantifies the extent to which a word resembles the characteristics of a stopword. To compute this measure, we resort to the number of different terms that occur in a window of size n to the left (and right) side of the candidate word. The more the number of different terms that co-occur with the candidate word (on both sides), the more meaningless the candidate word is likely to be. W_{Rel} is defined in Eq. 4:

$$W_{Rel} = \left(0.5 + \left(\left(WL * \frac{TF(w)}{MaxTF}\right) + PL\right)\right) + \left(0.5 + \left(\left(WR * \frac{TF(w)}{MaxTF}\right) + PR\right)\right) \quad (4)$$

More precisely, WL [WR] measures the ratio between the number of different words that co-occur with the candidate word (on the left [right] hand side) and the number of words that it co-occurs with. TF(w) is the frequency of the word with regards to the maximum term frequency within all words (MaxTF), and PL [PR] measures the ratio between the number of different words that co-occur with the candidate word (on the left [right] hand side) and the MaxTF. In practical terms, the more insignificant the candidate word is, the higher the score of this feature will be. Thus, stopwords-like terms will easily obtain higher scores.

2.2.5 Word DifSentence ($W_{DifSentence}$)

This feature quantifies how often a candidate word appears within different sentences. It is computed using the following equation:

$$\mathrm{W}_{\mathrm{DifSentence}} = \frac{\mathrm{SF}(w)}{\#\mathrm{Sentences}} \tag{5}$$

where SF(w) is the sentence frequency of the word (w), i.e., the number of sentences where (w) appears, and #Sentences is the total number of sentences in the text.

2.3 Individual Term Weighting

In the third step, we heuristically combine all these features into a single measure (see Eq. 6) such that each term is assigned a weight S(w). The smaller the value S(w), the more important the word (w) would be. This weight will feed the process of generating keywords to be explained in the next section.

$$\mathrm{S(w)} = \frac{\mathrm{W}_{Rel} * \mathrm{W}_{Position}}{\mathrm{W}_{Case} + \frac{\mathrm{W}_{Freq}}{\mathrm{W}_{Rel}} + \frac{\mathrm{W}_{DifSentence}}{\mathrm{W}_{Rel}}} \tag{6}$$

By looking at the equation, we can observe that both $\mathrm{W}_{\mathrm{Freq}}$ and $\mathrm{W}_{\mathrm{DifSentence}}$ are offset by $\mathrm{W}_{\mathrm{Rel}}$. The motivation behind this offset is to assign a high weight to words that appear frequently and appear in many sentences (likely indicative of their importance) as long as the word is relevant (i.e., for which $\mathrm{W}_{\mathrm{Rel}}$ is low). Indeed, some words may occur plenty of times and in many sentences and yet be useless (e.g., stopwords or similar). These terms should be penalized. Likewise, the position of a word in sentences occurring at the top of a document is an important feature that is taken into account, when multiplying $\mathrm{W}_{Rel} * \mathrm{W}_{Position}$.

2.4 Candidate Keyword List Generation

The fact that a keyword may consist of more than one word, forces us to consider a further step where the final score of a keyword (be it one, two or n-terms) is determined. To collect the candidate keywords, we consider a sliding window of 3-g, generating a contiguous sequence of 1, 2 and 3-g candidate keywords. In addition, keywords beginning or ending with a stopword will not be considered. It is also important to mention that no conditions are set in respect to the minimum frequency or sentence frequency that a candidate keyword must have. This means that we can have a keyword considered as significant/insignificant with either one occurrence or with multiple occurrences. Each candidate keyword will then be assigned a final S(kw), such that the smaller the score the more meaningful the keyword will be. Equation 7 formalizes this:

$$\mathrm{S(kw)} = \frac{\prod_{w \in kw} S(w)}{\mathrm{TF(kw)} * \left(1 + \sum_{w \in kw} S(w)\right)} \tag{7}$$

where S(kw) is the score of a candidate keyword with a maximum size of 3 terms, determined by multiplying (in the numerator) the score of the first term of the candidate keyword by the subsequent scores of the remaining terms, such that the smaller this

multiplication the more meaningful the keyword will be. This is divided by the sum of the S(w) scores to average out with respect to the length of the keyword, such that longer n-grams do not benefit just because they have a higher *n*. The result is further divided by TF(kw)- term frequency of the keyword - to penalize less frequent candidates. The final step in determining suitable candidate keywords is to eliminate similar candidates. For this, we use the *Levenshtein distance* [4] which measures the similarity between two strings. Among the strings considered similar (in our case, two strings are considered similar if their Levenshtein distance is above a given threshold) we keep the one that has the lowest S(kw) score. Finally, the method will output a list of potential relevant keywords, formed by 1, 2 or 3-g, such that the lower the S(kw) score the more important the keyword will be.

3 Evaluation

To evaluate the effectiveness of our method and its generality, we tested it on 4 different datasets characterized by different sizes of documents, data and languages: SemEval2010 [3], 500N-KPCrowd-v1.1 [5], WICC [1] and Schutz2008 [9]. SemEval2010 [3], which is probably one of the most well-known collections in this kind of evaluation, consists of 244 full scientific computer science papers ranging from 6 to 8 pages collected from ACM (8,020 tokens per document on average, the longest documents used in our experiments). Schutz2008 [9] in turn, consists of 1,231 papers, but this time belonging to the medical domain (selected from PubMed Central). A different collection is 500N-KPCrowd-v1.1 [5], which despite containing short documents (393 tokens per document on average) represents a different type of data: 500 English broadcast news stories from 10 different categories. Finally, WICC [2], is a Spanish dataset composed of 1,640 computer scientific articles published between 1999 and 2012 (which makes this not only the largest collection among all the datasets considered, but also a different one due to its language). In our experiments, we retrieve keywords with a maximum keyword size of 3-g and make use of a stopword corpus list. In addition, we consider a Levenshtein threshold of 0.8. To have a fair evaluation, we compare our method against TextRank[1] [7], RAKE[2] [8] and SingleRank (See footnote 2) [10], which are the state-of-the-art of unsupervised approaches. In addition, we also compare against TF.IDF (See footnote 2) which, despite being unsupervised, demands the existence of more than one document. Note that, unlike our method, TextRank, SingleRank and the implementation we used for TF.IDF make use of a PoS tagger. A python implementation of YAKE! is also available at PyPi[3]. This will enable researchers not only to test our method but also to compare their approach against ours, thus guaranteeing the reproducibility of the research. An online version and API of our method is also available here: http://bit.ly/YakeDemoECIR2018 [2].

[1] Implementation available at http://www.hlt.utdallas.edu/~saidul/code.html.

[2] Implementation available at https://github.com/zelandiya/RAKE-tutorial.

[3] Implementation available at https://pypi.python.org/pypi/yake.

3.1 Results

For the task of evaluating our proposal, we follow the traditional match evaluation scheme. That is, for each single document, we exactly match the keywords in the ground truth with those retrieved by tested methods, and calculate precision, recall, and F1-score. In the experiments, we assess the effectiveness over top 10 keywords retrieved by each method under f different collections to study the effect of document length, different types of data and languages. Table 1 presents the results for all datasets. We apply a paired sample t-test considering a significance level of 0.05. ▼ indicates a statistically significant improvement of the results of YAKE! method over corresponding baselines.

Table 1. SemEval2010, Schutz2008, 500N-KPCrowd and WICC results

Method	SemEval2010			Schutz2008			500N-KPCrowd			WICC		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1
YAKE!	0.153	0.103	0.123	0.217	0.058	0.091	0.251	0.063	0.101	0.050	0.141	0.073
TextRank	0.101▼	0.067▼	0.081▼	0.198▼	0.052▼	0.082▼	0.265	0.063	0.103	0.018▼	0.058▼	0.027▼
TF.IDF	0.036▼	0.023▼	0.028▼	0.100▼	0.028▼	0.043▼	0.223▼	0.060▼	0.095▼	0.026▼	0.067▼	0.037▼
SingleRank	0.035▼	0.022▼	0.027▼	0.082▼	0.024▼	0.037▼	0.190▼	0.054▼	0.084▼	0.017▼	0.045▼	0.024▼
RAKE	0.007▼	0.004▼	0.005▼	0.013▼	0.004▼	0.006▼	0.120▼	0.038▼	0.058▼	0.004▼	0.012▼	0.006▼

The results illustrate the effectiveness of the proposed method, with YAKE! achieving both higher precision, recall and F1-M in comparison to the baselines. Overall, one can note that except for the 500N dataset and the TextRank method, for which the t-test does not show any significant improvement of one method over another, all the remaining results reflect the fact that YAKE! method achieves better and statistically significant results. Among all the datasets, the best results are achieved in the 500N dataset by TextRank with 0.265 for P@10, followed very closely by YAKE! with 0.251. However, as previously referred there is no statistically significant difference between any of the two methods, meaning that they are rather equivalent, beyond being evidently superior when compared to the remaining methods.

The results further confirm that regardless the type of data, YAKE! method tends to have relatively stable effectiveness. To study this effect, one can look at the results of Schutz2008, which in contrast to 500N (a collection of broadcast news) is composed of full text research articles. Despite a slight drop, results are still relatively good with YAKE! achieving 0.217 of P@10 still significantly better than the 0.198 achieved by the TextRank method.

The effect of the document length is then studied by running our experiments under the SemEval2010 collection (the largest documents here studied –8.020 per document on average, twice the double of the Schtuz2008 collection and 25 times more than the 500N). Although the results have dropped considerably, still, they are significantly better than the ones of the second-best approach (TextRank), which only achieves 0.101 of P@10 (significantly lower than 0.154 obtained by our method). This proves that, although our method performs better than any of the baselines, still the effect of document length significantly impacts the results obtained. This should be studied in

the future. It is also important to stress out that, while TextRank depends and benefits from NLP techniques, such as PoS taggers, YAKE! simply takes as input a set of plain features extracted from the text. These may be understood as an advantage over baselines anchored on PoS taggers, which may not be available or may perform poorly for some minor languages for which there is either a lack of interest or lack of open source tools.

Finally, we wanted to evaluate the effectiveness of our method under a different language. To this regard we consider the WIC dataset. Once again YAKE! returns the best results, although in this case a score of only 0.05 of P@10 has been obtained, which has much to do with the fact that only 3.57 of gold keywords per document have been defined in the collection (the smallest number of gold keywords among all the datasets). While, one may be tempted to claim that the results are quite low when compared to other IR tasks, it should be taken into account that unlike other IR core research areas, the realm of keyword extraction is a different one, with the tendency to have lower scores. One of the reasons for this is that an exact match between the ground-truth and the methods keyword is usually used as a rule-of-thumb thus impeding partial matches. An additional reason is that some of the keywords of the ground-truth cannot simply be found in the text, thus making it impossible to have an exact match. Thus, any increase in the effectiveness of current solutions, would always represent a significant contribution over state-of-the-art solutions.

4 Conclusions

In this paper, we propose a novel approach to extract keywords from single documents. Based on the experiments, we could confirm that YAKE! achieves better results in comparison to four state-of-the-art unsupervised keyword extraction algorithms, over a large number of text documents in four different datasets. Unlike supervised approaches, which require a training corpus, YAKE! is fully unsupervised. Moreover, the fact that it only leverages features drawn from the text itself together with its independence with regards to natural language processing techniques makes it suitable for other text collections, including different domains and languages. As future work, we plan to investigate how our method performs in comparison with the most popular supervised approaches like KEA [11].

Acknowledgements. This work is partially funded by the ERDF through the COMPETE 2020 Programme within project POCI-01-0145-FEDER-006961, and by National Funds through the FCT as part of project UID/EEA/50014/2013 and of project UID/MAT/00212/2013. It was also financed by MIC SCOPE (171507010) and by Project "TEC4Growth - Pervasive Intelligence, Enhancers and Proofs of Concept with Industrial Impact/NORTE-01-0145-FEDER-000020" which is financed by the NORTE 2020, under the PORTUGAL 2020, and through the ERDF.

References

1. Aquino, G., Lanzarini, L.: Keyword identification in Spanish documents using neural networks. J. Comput. Sci. Technol. **15**(2), 55–60 (2015)
2. Campos, R., Mangaravite, V., Pasquali, A., Jorge, A., Nunes, C., Jatowt, A.: YAKE! collection-independent automatic keyword extractor. In: Pasi, G., Piwowarski, B., Azzopardi, L., Hanbury, A. (eds.) ECIR 2018, LNCS, vol. 10772, pp. 806–810. Springer, Cham (2018)
3. Kim, S., Medelyan, O., Kan, M.-Y., Baldwin, T.: SemEval-2010 task 5: automatic keyphrase extraction from scientific articles. In: SemEval 2010, Sweden, pp. 21–26 (2010)
4. Levenshtein, V.: Binary codes capable of correcting deletions, insertions, and reversals. Sov. Phys. Dokl. **10**(8), 707–710 (1966)
5. Marujo, L., Viveiros, M., Neto, J.: Keyphrase cloud generation of broadcast news. In: arXiv (2013)
6. Matsuo, Y., Ishizuka, M.: Keyword extraction from a single document using word co-occurrence statistical information. J. Artif. Intell. Tools **13**(1), 157–169 (2004)
7. Mihalcea, R., Tarau, P.: TextRank: bringing order into texts. In: EMNLP 2004, pp. 404–411 (2004)
8. Rose, S., Engel, D., Cramer, N., Cowley, W.: Automatic Keyword Extraction from Individual Documents. Text Mining: Theory and Applications. Wiley, Chichester (2010)
9. Schutz, A.T.: Keyphrase extraction from single documents in the open domain exploiting linguistic and statistical methods. Master thesis, National University of Ireland (2008)
10. Wan, X., Xiao, J.: Single document keyphrase extraction using neighborhood knowledge. In: AAAI 2008, 13–17 July, pp. 855–860 (2008)
11. Witten, I., Paynter, G., Frank, E., Gutwin, C., Nevill-Manning, C.: KEA: practical automatic keyphrase extraction. In: Proceedings of the JCDL 2004, 7–11 June, pp. 254–255 (1999)