A Novel, Language-Independent Keyword Extraction Method

Germán Aquino[1], Waldo Hasperué[1,2], César Estrebou[1] and Laura Lanzarini[1]

[1] III-LIDI. School of Computer Science. UNLP. Argentina
[2] CONICET scholarship
{gaquino, whasperue, cesarest, laural}@lidi.info.unlp.edu.ar

Abstract. Obtaining the most representative set of words in a document is a very significant task, since it allows characterizing the document and simplifies search and classification activities. This paper presents a novel method, called LIKE, that offers the ability of automatically extracting keywords from a document regardless of the language used in it. To do so, it uses a three-stage process: the first stage identifies the most representative terms, the second stage builds a numeric representation that is appropriate for those terms, and the third one uses a feed-forward neural network to obtain a predictive model. To measure the efficacy of the LIKE method, the articles published by the Workshop of Computer Science Researchers (WICC) in the last 14 years (1999-2012) were used. The results obtained show that LIKE is better than the KEA method, which is one of the most widely mentioned solutions in literature about this topic.

Keywords: Text Mining, Document characterization, Back-propagation, WICC.

1 Introduction

Text mining presents interesting challenges to solve, since the lack of structure in the texts analyzed makes it difficult to extract information from them. Nowadays, given the large number of texts that are published each day, be these scientific articles, books, journals, periodicals or web pages, facing these challenges can prove to be interesting, as well as developing strategies that allow obtaining information from relevant texts.

One way of briefly describing the topic of a document is by means of a list of keywords. The keywords in a document are of the utmost importance, since they allow carrying out several tasks, such as searching for a specific topic, classifying documents, clustering [1], summarization [2] [3] [4], etc.

Even though most of the times the author of the document is the one in charge of proposing the list of keywords, as in the case of scientific publications, there are other times when this list is not present at all and, therefore, it would be interesting to have an automated method that can propose a list of keywords by analyzing the text of the document.

Within text mining, there have been various alternatives proposed for the task of extracting keywords. There are statistical methods that typically do not have prior training with the documents; in such cases, only statistical information is collected from the words that are present in the document to identify which of them can be chosen as keywords. The most widely used statistical methods include TF-IDF [5] [6] [7] [8], word co-occurrence [9], etc.

On the other hand, there are machine learning-based methods that, from a given corpus, carry out a training process and generate a model that allows performing classifications afterwards or, in the case of keyword extraction, establishing which words in the document are candidates to be chosen as keywords. In these cases, each document in the initial corpus must have a list of keywords that are used as positive cases during training. Some of the machine learning methods used in this type of tasks are Naïve Bayes [8] [10], Support Vector Machine [11], etc.

The methods that analyze the linguistic aspects of the documents are those that offer the most interesting solutions, since they combine lexical analysis, syntactic analysis, etc. [12] [13]; however, their disadvantage is that they are strongly dependant on the language used to write the documents.

One of the main concepts pertaining to the specific task of extracting keywords from documents is that of n-grams. Any word within a document is a unigram, while any sequence of two or more words forms an n-gram, where *n* indicates the number of words in the sequence. When extracting keywords, any n-gram in the document is a potential keyword for that document. Most of the techniques that carry out this task perform calculations and measurements on each n-gram in the document, and then process them by means of a machine learning technique [14] or by assigning a given score [15] to obtain a model that can be used as predictor for future documents.

In order to extract keywords, some methods require a corpus from which to generate a first model [7] [8], while others do so from a single document [9]. There are techniques that carry out numeric and/or statistical calculations for all n-grams in the document [16] [17], while others exploit certain linguistic information [12] [13]. Most of the existing strategies pre-process the documents with stemming and word filtering techniques by means of a stop-word list.

In this paper, a novel method is proposed, called LIKE (Language Independent Keyword Extraction), which uses texts from documents from a given corpus to obtain a model that can be used to extract keywords. To this end, it uses a three-phase algorithm. The first phase consists in extracting any n-grams that are detected as candidates for keyword, the second phase calculates a set of numerical features for each n-gram that was detected, and the third and final stage uses those features to produce a model by training a feed-forward network. This trained network is used as model to decide, given a new document, possible keywords. LIKE is independent from the source language of the documents, provided that the same language is maintained throughout each individual document, since it does not carry out the usual pre-processing steps of stemming or word filtering using a stop-word list.

This article is organized as follows: in Section 2, LIKE is described; in Section 3, the results obtained in the experiments carried out are presented; and in Section 4, conclusions and future work are presented.

2 LIKE

The method proposed in this paper, called LIKE, is a three-phase method that allows extracting a list of keywords by analyzing the text in a document. LIKE is independent from the source language of the documents, since it does not carry out the pre-processing steps of stemming or word filtering using a stop-word list. LIKE analyzes the documents in a document corpus to train a back-propagation network that will then be used as model to determine the list of keywords for new documents.

LIKE starts by identifying all existing n-grams in each document in the corpus. Since the number of all possible n-grams would be excessively high, a strategy is required to reduce this number. In this proposal, the method presented by [18] is used, since it allows identifying a much lower number of n-grams.

During the second stage, each n-gram obtained in the previous stage is transformed into a features numeric vector; these vectors are labeled as one of two classes of data. One class includes the vectors corresponding to those n-grams that are part of the list of keywords proposed by the author(s) of the document, while the other class is formed by the vectors corresponding to all remaining n-grams. The third stage consists in using these two data classes to train a back-propagation network.

2.1 Phase 1. N-gram Extraction

The first phase of the method proposed consists in identifying the n-grams in the corpus. For each document, all existing n-grams are extracted. An n-gram is considered to be valid if it is formed by consecutive words within the same sentence with no punctuation marks between any of them.

In general, the number of existing n-grams is excessively high. For the tests carried out for this work, which has a corpus of 96 documents, more than 580,000 n-grams can be extracted. Therefore, a strategy to identify a lower number of n-grams is required. In this proposal, the algorithm presented in [18] is used. This strategy, inspired in the Apriori algorithm, builds sets of elements from other smaller sets. In this algorithm, the maximum n value (number of words in the n-gram) and the minimum occurrence frequency for each n-gram have to be determined. N-grams are built from the (n-1)-grams that meet the requirement of a minimum specified frequency. To do this, it is assumed that an n-gram whose frequency is k is built from the intersection of two (n-1)-grams whose frequency is at least k, i.e., an n-gram cannot be more frequent than its parts. For each n-gram, the first n-1 and last n-1 words are taken, and it is checked that these (n-1)-grams meet the minimum allowed frequency criterion. If this criterion is not met, the n-gram is discarded. Finally, there are n runs on the text, first to obtain 1-grams, then 2-grams based on these, then 3-grams, and so forth.

The use of this strategy in LIKE allows identifying a low number of n-grams in each document. In the experiments that were carried out for this work, the total number of n-grams for the entire corpus was reduced to little more than 70,000.

The result of this phase is a list of n-grams, which are then labeled. Once this list of keywords is known for each document, a label is assigned to each n-gram to indicate if the n-gram is a keyword or not. Thus, a two-class set of data is generated.

2.2 Phase 2. N-gram Characterization

The purpose of this phase is converting each of the n-grams that were identified in the previous phase into a features vector. In this article, we propose that the eight features detailed below are calculated.

i) TF (Term Frequency): TF is perhaps, together with IDF, one of the descriptors most widely-used to characterize n-grams. Term Frequency is the number of times the n-gram occurs in the document divided by the total word count of the document.

ii) IDF (Inverse Document Frequency): It is the ratio between the number of documents in the corpus that include the n-gram *d*(*g*) and the total number of documents *D*.

$$IDF(g) = \log(\frac{D}{d(g)})$$

iii) First occurrence of the term: This represents the relative position of the first occurrence of the n-gram. It is calculated as the number of words before the first occurrence of the n-gram divided by the total word count of the document.

iv) Position within the sentence: This is the relative position of the n-gram in the sentence that contains it. The same as the previous one, it is calculated as the number of words before the occurrence of the n-gram in the sentence divided by the total word count of the sentence itself. If the same n-gram appears several times in different sentences in the same document, then all n-gram occurrences are averaged.

v) Part of the title: This feature is a binary value that indicates if the n-gram appears in the title of the document or not.

vi) Part of the n-gram present in the title: This feature (only valid for n-grams with two or more words) counts the number of words in the n-gram that also appear in the title, regardless of the order of the words in the title. This number of occurrences is normalized by the number of words in the n-gram. In the case of unigrams, the same as the previous feature, this is a binary value that indicates either presence or absence.

vii) NSL (Normalized Sentence Length): This is the length of the sentence where the n-gram appears divided by the length of the longest sentence in the document. If the n-gram appears in more than one sentence in the document, all occurrences are averaged.

viii) Z-score: This is a statistical measure that normalizes the frequency of the n-gram. It requires knowing the mean and standard deviation of the frequency for each n-gram.

$$Z\text{-}score(g) = \frac{freq(g) - \mu}{\sigma}$$

If the n-gram appears in more than one sentence in the document, all occurrences are averaged.

Of the eight features proposed, only two (IDF and Z-score) require the corpus in order to be calculated.

The result of this phase is a features vector for each of the n-grams identified in the previous phase.

2.3 Phase 3. Creating the Model

The third phase of the method proposed consists in creating a model that can learn from a given corpus and allows classifying n-grams from new documents as possible keywords or not. The prediction model is built by training a back-propagation network.

The problem that arises when trying to use the set of vectors obtained in the previous phase as data for training the back-propagation network is that the classes in this data set are not balanced, since the "not a keyword" class has a lot more elements than the "is a keyword" class. In the corpus used to carry out the experiments presented here, the ratio of elements in both classes was approximately 150 to 70,000.

When there is a data set with unbalanced classes, training a back-propagation network is not an easy task, since the training set prevents the generation of a model that can accurately predict the data in the minority class. In light of this problem, several solutions have been proposed ([19] [20] [21]). In particular, the solution described in [21] proposes that, before the training process, a clustering operation is performed on the data in the larger class in order reduce its number of elements. In this work, the idea in [21] is used – the data in the larger class are clustered using the k-means algorithm.

Be u the number of data present in the minority class, the value of k is then established as $k=u/10$. A clustering of k clusters is performed, and 10 random elements are extracted from each resulting cluster. These $10*k$ elements thus selected form a new data set that replaces the original data from the majority class. Following this methodology, the back-propagation network can be trained using a data set whose classes have similar numbers of elements.

To train the back-propagation network, the classic algorithm is used. After several tests and empirical observation, a decision was made to use seven neurons for the hidden layer, the logsig function as transfer function for the hidden layer, the tansig function for the output layer, an alfa of 0.25, and a maximum of 1,000 iterations. The best results were obtained with this configuration.

The result of this training process is a model that can predict keywords for new documents. The procedure to establish the keywords for a new document is as follows: first, the n-grams are extracted as described in Section 2.1; then, features vectors are calculated for each n-gram as explained in Section 2.2; and finally, these new vectors are presented to the trained network to determine if a given n-gram is a keyword for the document or not.

3 Results

The method proposed here was tested using as corpus all papers submitted to WICC (http://redunci.info.unlp.edu.ar/wicc.html) between 1999 and 2012. Only those articles written in English were included in the corpus (96 articles). The rationale for using only articles written in English was that, at a later stage, the results obtained with this method would be compared with those obtained with other keyword extraction method that is widely used in the literature: KEA [8]. Even though KEA can be adapted to work with languages other than English, since it depends on a stemmer and a stop-word list, those developed by the authors were used, which are in English.

KEA [8] is an automated keyword extraction algorithm that identifies candidate words by using lexical methods to calculate a set of features, and then apply an automated learning algorithm that allows predicting which candidates are good keywords.

The same as LIKE, KEA builds a prediction model using a training corpus with specified keywords, and it then uses this model to extract keywords from new documents.

KEA allows the free extraction of keywords, as well as the extraction of keywords using a vocabulary list that is controlled by means of a thesaurus. For these tests, the first mode was used, establishing as parameter a number of three keywords per document. In order to train KEA, a stop-word list and a stemmer are required. The stop-word list contains words of low semantic content (conjunctions, articles, prepositions, etc.) that should not be considered as keyword candidates.

The first step in the KEA method consists in filtering out the words that appear in the stop-word list, and then apply a stemming process to reduce to their syntactic root all n-grams that were not filtered out. The next step is to calculate the features of all candidate words, which include: TF-IDF, the initial position of the n-gram in the text and the length of the n-gram (the number of individual words that form the n-gram). Based on this representation, KEA uses Naïve Bayes as learning algorithm.

Both LIKE and KEA were trained using the same corpus. From that corpus, some documents were selected randomly for the training stage and the rest were used for testing. For each test, accuracy, recall and f-measure are calculated.

Both methods were run with the 10-fold cross-validation procedure, and average accuracy, recall and f-measure were obtained. The 10-fold cross-validation procedure was run 30 separate times with both methods in order to measure the statistical significance of the various results obtained.

One of the greatest disadvantages of LIKE (also present in KEA) is that, for each n-gram, two features are calculated whose result depends on the entire corpus (IDF and Z-score). Depending on a corpus for calculating features is not desirable, so two versions of the LIKE method were run – LIKE-8, which uses the eight features proposed in this article (see Section 2.2), and LIKE-6 which uses only the six features that do not depend on the corpus (i.e., all but IDF and Z-score).

Table 1 shows the average accuracy, recall and f-measure for the 30 separate runs with LIKE-8, LIKE-6 and KEA. With these results, a statistical test was carried out to determine the statistical significance for LIKE-8 vs. KEA, LIKE-6 vs. KEA and LIKE-8 vs. LIKE-6 (Table 2). As it can be seen in Table 2, both LIKE-8 and LIKE-6 achieved better results than KEA, while the version that used all eight attributes improved only accuracy and f-measure results when compared to the version that used only those six that are not corpus-dependent.

Table 1. Average precision, recall and f-measure for LIKE-8, LIKE-6 and KEA (standard deviation indicated between parentheses).

	LIKE-8	LIKE-6	KEA
Precision	0.76 (0.051)	0.65 (0.101)	0.52 (0.006)
Recall	0.75 (0.094)	0.72 (0.141)	0.37 (0.004)
f-measure	0.74 (0.053)	0.68 (0.116)	0.43 (0.005)

Table 2. Results of the statistical significance for precision, recall and f-measure for LIKE-8 vs. KEA, LIKE-6 vs. KEA and LIKE-8 vs. LIKE-6. For α=0.01 the "+" sign indicates that the result is statistically significant, while the "–" sign indicates that there is no statistical significance (p-value indicated between parenthesis).

	LIKE-8 vs. KEA	LIKE-6 vs. KEA	LIKE-8 vs. LIKE-6
Precision	+ ($5.48x10^{-22}$)	+ ($5.19x10^{-08}$)	+ ($4.12x10^{-06}$)
Recall	+ ($1.50x10^{-19}$)	+ ($4.08x10^{-14}$)	- (0.4832)
f-measure	+ ($2.18x10^{-24}$)	+ ($2.51x10^{-12}$)	+ (0.0096)

4 Conclusions and Future Work

The novel automated method LIKE for extracting keywords from the text of a set of documents has been presented. This method extracts n-grams from the documents and then calculates a series of features to convert them into numeric vectors. It then uses these vectors as data to train a back-propagation network and thus obtain a model that works as predictor and that can be used to extract keywords from new documents.

In this paper, the calculation of eight features is proposed for each n-gram, with only two of these being dependent on the entire corpus. LIKE was trained using the eight features, and then a second test was carried out using only the six features that do not depend on the corpus. Articles written in English submitted to the WICC between 1999 and 2012 were used for the experiments. The results obtained were compared with KEA, and it was shown that both the six-feature and the eight-feature LIKE models were better. When comparing the results obtained with both versions of LIKE, using all eight features turned out to be superior than using just six when calculating precision and f-measure, while for the recall parameter, neither of the versions appeared to be better than the other.

The main advantage of the method presented here is that it does not depend on the language of the texts analyzed, since it does not pre-process them because it does not use stemming or a stop-word list.

As future work, it would be interesting to study in detail the n-grams that are negatively classified so as to determine their nature and analyze the possibility of detecting grammar structures that help improve the performance of the method. Also, if less candidate n-grams are identified, the majority class of negative cases would be reduced and this could possibly lead to being able to omitting the clustering stage before training the network.

Another aspect to be studied in relation to the method proposed here is the possibility of assigning keywords from a list of controlled vocabulary. Different authors may choose different key words in articles dealing with the same topic, so it would be interesting if an automated assignment method were available to assign key words from a list of controlled vocabulary. This would ensure that documents on related topics would have the same key words, which would in turn improve the results obtained in future searches, classifications or statistical analyses.

References

1. Tonella, P., Ricca, F., Pianta, E., Girardi, C.: Using keyword extraction for Web site clustering. In: Conference Using keyword extraction for Web site clustering, pp. 41 - 48. (2003)
2. D'Avanzo, E., Magnini, B., Vallin, A.: Keyphrase Extraction for Summarization Purposes: The LAKE System at DUC-2004. Proceedings of the 2004 Document Understanding Conference (2004)
3. Wan, X., Yang, J., Xiao, J.: Towards an Iterative Reinforcement Approach for Simultaneous Document Summarization and Keyword Extraction. In: Conference Towards an Iterative Reinforcement Approach for Simultaneous Document Summarization and Keyword Extraction, pp. 552-559. (2007)
4. Zha, H.: Generic summarization and keyphrase extraction using mutual reinforcement principle and sentence clustering. In: Conference Generic summarization and keyphrase extraction using mutual reinforcement principle and sentence clustering, pp. 113-120. (2002)
5. Islam, M.R., Islam, M.R.: An improved keyword extraction method using graph based random walk model. 11th International Conference on Computer and Information Technology 225-229 (2008)
6. Kaur, J., Gupta, V.: Effective Approaches For Extraction Of Keywords. International Journal of Computer Science Issues 7, (2010)
7. Liu, Y., Ciliax, B.J., Borges, K., Dasigi, V., Ram, A., Navathe, S., Dingledine, R.: Comparison of two schemes for automatic keyword extraction from MEDLINE for functional gene clustering. Proc IEEE Comput Syst Bioinform Conf 394-404 (2004)
8. Witten, I.H., Paynter, G.W., Frank, E., Gutwin, C., Nevill-Manning, C.G.: KEA: practical automatic keyphrase extraction. In: Conference KEA: practical automatic keyphrase extraction, pp. 254-255. (1999)
9. Matsuo, Y., Ishizuka, M.: Keyword Extraction From A Single Document Using Word Co-Occurrence Statistical Information. In: Conference Keyword Extraction From A Single Document Using Word Co-Occurrence Statistical Information, pp. 392-396. (2003)
10. Frank, E., Paynter, G.W., Witten, I.H., Gutwin, C.: Domain-specific keyphrase extraction. proc. Sixteenth International Joint Conference on Artificial Intelligence 668--673 (1999)
11. Wu, C., Marchese, M., Wang, Y., Krapivin, M., Wang, C., Li, X., Liang, Y.: Data Preprocessing in SVM-Based Keywords Extraction from Scientific Documents. Fourth International Conference on Innovative Computing, Information and Control (ICICIC), pp. 810 - 813 (2009)
12. Csomai, A., Mihalcea, R.: Linguistically Motivated Features for Enhanced Back-of-the-Book Indexing. Proceedings of the 9th SIGdial Workshop on Discourse and Dialogue (2008)

13. Kireyev, K.: Semantic-based estimation of term informativeness. Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics 530-538 (2009)
14. Hulth, A.: Improved automatic keyword extraction given more linguistic knowledge. Proceedings of the 2003 conference on Empirical methods in natural language processing 216-223 (2003)
15. Wang, C., Zhang, M., Ru, L., Ma, S.: An Automatic Online News Topic Keyphrase Extraction System. IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology 1, 214-219 (2008)
16. HaCohen-Kerner, Y., Gross, Z., Masa, A.: Automatic extraction and learning of keyphrases from scientific articles. Proceedings of the 6th international conference on Computational Linguistics and Intelligent Text Processing 657-669 (2005)
17. Hu, X., Wu, B.: Automatic Keyword Extraction Using Linguistic Features. Sixth IEEE International Conference on Data Mining Workshops 19 - 23 (2006)
18. Fürnkranz, J.: A study using n-gram features for text categorization. Austrian Research Institute for Artifical Intelligence 3, 1-10 (1998)
19. Castro, C.L., Braga, A.P.: Novel Cost-Sensitive Approach to Improve the Multilayer Perceptron Performance on Imbalanced Data. IEEE Transactions on Neural Networks and Learning Systems 24, 888-899 (2013)
20. Lin, M., Tang, K., Yao, X.: Dynamic Sampling Approach to Training Neural Networks for Multiclass Imbalance Classification. IEEE Transactions on Neural Networks and Learning Systems 24, 647 - 660 (2013)
21. Zhang, Y.-p., Zhang, L.-N., Wang, Y.-C.: Cluster-based majority under-sampling approaches for class imbalance learning. 2nd IEEE International Conference on Information and Financial Engineering (ICIFE) 400-404 (2010)