Keyword Extraction using Auto-associative Neural Networks

Germán Aquino[1, 2], Waldo Hasperué[1, 2], Laura Lanzarini[1],

[1] Instituto de Investigación en Informática – III–LIDI Facultad de Informática – Universidad Nacional de La Plata - Argentina
[2] CONICET – Consejo Nacional de Investigaciones Científicas y Técnicas
{gaquino, whasperue, laural}@lidi.info.unlp.edu.ar

Abstract. The large amount of textual information digitally available today gives rise to the need for effective means of indexing, searching and retrieving this information. Keywords are used to describe briefly and precisely the contents of a textual document. In this paper we present a new algorithm for keyword extraction. Its main goal is to extract keywords from text documents written in Spanish quickly and without requiring a large training set. This goal was achieved using auto-associative neural networks, also known as *autoencoders*, trained using only the terms designated as keywords in the training set, so that these networks can learn the features characterizing the important terms in a document.

Keywords: Keyword Extraction, Text Mining, Neural Networks, Autoencoders.

1 Introduction

The large amount of textual information digitally available today gives rise to the need for effective means of indexing, searching and retrieving text documents quickly and without having a user to read them entirely, which in many cases is not feasible. Keywords are used to describe briefly and precisely the contents of a text document, so that a user can find documents relevant to him/her without having to read them beforehand. However, there are many documents without keywords and the task of manually assigning keywords to them is slow, difficult and highly subjective. For this reason it is beneficial to have tools that assist professional indexers by providing a list of terms candidates to be keywords. Some of the most known keyword extraction methods are described in Section 2.

In this paper we present a new algorithm to extract keywords from text documents written in Spanish. This algorithm uses a set of example documents to build a classification model capable of learning the structural features of the terms considered keywords, and to recognize terms having these features in unseen documents. The model is built by training an autoencoder, which assigns a reconstruction error to each term of a document. This allows us to select the terms with lower reconstruction error as keywords. The proposed algorithm is explained in detail in Section 3.

One of the main goals of this algorithm is that such list of candidate terms contain the largest possible number of descriptive terms. Also it is desirable that the classification model can be constructed using a small training set.

The results of the experiments carried out are presented in Section 4, and Section 5 summarizes conclusions and future work.

2 Related Work

The discipline of machine learning has been attempting to tacke the problem of keyword extraction since a few decades ago. This approach aims to transform text data into a structured representation suitable for learning algorithms. Such algorithms work with a feature set calculated for each term of a document and consider keyword extraction as a classification problem, determining whether each term is a keyword or not. Supervised learning methods usually use the terms designated as keywords by the authors of the training documents as examples of one class, and the rest of the terms as examples of the other class. The latter, the class of the terms that are not keywords, is naturally much more numerous than the other class. This imbalance in the number of elements of each class, together with the inherent ambiguity of natural language, makes keyword extraction a very difficult problem to solve.

In order to find a suitable representation for learning algorithms, many keyword extraction methods apply *stemming*, which consists of reducing each term to its morphological root, and filter terms using a *stoplist*, which is a list of terms with low semantic value (*stopwords*) such as articles, prepositions, conjunctions and pronouns.

One of the first advances in considering keyword extraction as a classification problem to be solved through machine learning comes from Peter Turney [1]. Turney developed an algorithm called GenEx that applies a set of rules whose parameters are tuned in a first stage using a genetic algorithm. These rules are used to rank terms and select the ones that have the highest score in the second stage. GenEx has a pre-processing step in which *stemming* is applied to terms and *stopwords* are filtered.

Another significant example is KEA, developed by Ian Witten and Eibe Frank [2]. KEA uses TF-IDF [3] and position in the text as features of the terms, and a third attribute that represents the number of times a given term was selected as keyword in the training set, which requires that all the documents belong to the same domain for this attribute to have meaning. This algorithm uses a Naïve Bayes classifier that calculates for each term the probability of being chosen as keyword. Like GenEx, KEA also applies *stemming* and *stopword* filtering.

In a previous paper we introduced a keyword extraction algorithm called LIKE [4]. That algorithm has the particularity of being completely language-independent as it does not use specific linguistic resources. As in the aforementioned works, we treated keyword extraction as a classification problem using a neural network to classify terms. Since the number of terms that are considered keywords in the training set is much lower than the terms that are not, the number of elements belonging to each class is heavily unbalanced, which greatly hinders the training of the neural network [5]. To deal with this problem a clustering algorithm was applied on the elements of

the majority class to equate its number to the amount of elements of the minority class before training the network.

One of the most recent proposals on the keyword extraction area is Maui, developed by Olena Medelyan [6]. Maui is an updated version of KEA which incorporates more attributes and replaces the Naïve Bayes classifier with bagged decision trees [7]. Like KEA, Maui uses a *stemmer* and a *stoplist* of the given language and it is built on top of the machine learning platform Weka [8]. Optionally, Maui gives the ability to use Wikipedia Miner [9] to extract information from Wikipedia and use it as a knowledge base to calculate more features such as the number of articles containing a given term.

Unlike the aforementioned methods, the algorithm introduced in this paper does not use *stemming* nor *stoplists*, as such mechanisms are very well developed for English but are not on the same level of quality for Spanish and other languages. Also, the construction and validation of these resources must be supervised by linguists and language professionals to ensure its correctness. Nevertheless, we applied another mechanism to filter malformed or insignificant terms. This mechanism consists in tagging each word of the document according to its grammatical function (noun, adjective, verb, article, etc) and discarding the terms starting or ending with an article, preposition, conjunction, pronoun or verb, following the idea that keywords are usually formed by nouns and adjectives.

3 Description of the Algorithm

In this work, instead of considering keyword extraction as a *discrimination* problem (distinction between the elements belonging to each class), we consider it a *recognition* problem (learning of the properties that determine the membership of an element to a group) and to that end we apply an auto-associative classifier [10]. Auto-associative classifiers train using only the elements of the class of interest, which in our case are the terms designated as keywords by the autors of the documents of the training set, and are therefore more tolerant to class imbalance. This has the additional benefit that the training is conducted much more quickly than in the discriminant classifiers, since the latter need to analyze the examples of both classes.

3.1 Pre-processing

The first step of the proposed algorithm consists in splitting the text in sentences and words using two list of delimiters provided as parameters. These delimiters can be any character sequence and will not be part of extracted terms. Once sentences and words are obtained the algorithm proceeds to compute the features for the terms.

Terms are represented by N-grams, which are sequences of N consecutive words in the same sentence, and for each one we compute a set of features relative to position and frequency of the term in the document. To avoid generating every possible N-gram from the text, many of which are probably not valid terms, we apply an algorithm of efficient generation of N-grams [11] reducing processing time. This

algorithm requires the maximum length of the terms considered and the minimum frequency that a term must have in a document to be eligible as keyword.

In order to further reduce the number of terms to be processed we apply a filter which discards N-grams that start or end with a word of a given grammatical category, like articles and prepositions. This filtering discards sequences of words that are not eligible as keywords, such as the Spanish sequence "de forma que". This process is similar to the application of a *stoplist*, with the difference that we do not use an exhaustive list of terms to rule out but instead we assign *part-of-speech (POS)* tags to each word of the document based on its use. To this end we apply a *maximum entropy* model [12] trained with the tool OpenNLP [13] using a tagged corpus as training set. This filtering greatly reduces processing time, since it discards an important number of terms that should not be chosen as keywords.

The POS tagging model for Spanish was trained using the tagged corpus Conll-2002 [14] and the grammatical tags defined by the EAGLES group [15]. The corpus was provided in the 2002 *Conference on Computational Natural Language Learning* to be used to train and evaluate algorithms of named entitity recognition (such as person names, places and organizations). The EAGLES group *(Expert Advisory Group on Language Engineering Standards)* is an organization with the goal of providing standars for large-scale language resources manipulation, both text and speech, computational linguistic formalisms and software tools.

3.2 Term representation

The defined representation consists in 8 attributes calculated for each term of the documents. These attributes are the same as those we used in LIKE.

1 - Term Frequency (TF): the rate between the frequency of the term and the number of words in a document.

2 - Term Frequency – Inverse Document Frequency (TF-IDF): consists in weighting TF with the frequency a term has in the entire corpus. TF-IDF favors terms that are unfrequent in the corpus but frequent in the given document.

3 - First Ocurrence: the relative position of the first occurrence of the term in the text. It is calculated as the rate between the number of words that appear before the first occurrence of the given term and the number of words of the document.

4 - Position in Sentence: a measure of the relative position of a term in the sentences it appears in. For each sentence *s* that contains term *t*, we count the number of words that appear in *s* before *t*, and we average these values.

5 – Ocurrence in Title: this attribute is set to 1 if the term appears literally in the document title and 0 otherwise. It represents the notion that terms appearing in the title are important and hence are candidates to be keywords.

6 – Ocurrence of Members in Title: this attribute, like the previous one, relates the importance of a term with its appearance in the title. The difference is that this attribute considers occurrences in the title of the individual words of the term. This allows considering terms whose occurrences in the title are not literal, such as when the words are in a different order or there are more or less lexical terms. It is the rate between the number of words of a term *t* that appear in the title and the length of *t*.

7 - Normalized Sentence Length: it is a measure of the length of the sentences in which a given term appears in, calculated by averaging the lengths of these sentences. Such lengths are also normalized by dividing them by the length of the longest sentence in the document.

8 - Normalized Frequency (Z-Score) [16]: consists in normalizing the term frequency using its mean frequency in the training corpus and its standard deviation. It measures the difference between the frequency of a term and its mean frequency in the corpus.

Of these 8 features only TF-IDF and Z-Score depend on the entire training corpus for their calculation. The rest of the attributes use only information of the given document.

3.3 Description of the Classifier

The classifier used in this work is an auto-associative neural network, or *autoencoder*. Neural networks aim to learn a function that maps from elements of an input set to elements of an output set. In autoencoders the output set is the same as the input set, so that these networks attempt to learn the identity function of the training set.

Given an input vector $X \in R^n$ the network produces an approximated vector X', which presents a reconstruction error defined as the sum of squared differences:

$$e(X) = \sum_{i=1}^{n} (X_i - X_i')^2$$

As training is carried out using the elements of the class of interest it is expected that new elements that are similar to the ones in the training set have a lower reconstruction error than those that are not. By determining a threshold, this allows to classify the elements as belonging to one class or the other [17]. In our case the class of interest is the class of terms that are keywords, and hence the training set is formed by the feature vectors of the terms designated as keywords by the authors of the training documents.

The autoencoder has an equal number of input and output neurons, and is trained in the same way as a conventional neural network. In this work we used *Resilient Backpropagation* [18][19] as training algorithm. This algorithm uses a velocity coefficient for each weight of each neuron, so that it does not need a global learning rate. On weight update it only uses the sign of the gradient of the error surface instead of its magnitude, so that descent through the error surface is independent of the shape of this surface. This facilitates convergence to a minimum when the error surface is irregular. As Resilient Backpropagation removes the need of setting a learning rate, the remaining free parameters of the network are the hidden and output activation functions and the size of the hidden layer.

As we mentioned earlier, the autoencoder assigns a reconstruction error to each element of a testing set, which represents the similarity between the element and those of the training set. Instead of determining a cutoff threshold to accept or reject a term

as keyword we opted to select the R terms with lowest reconstruction error from each document of the testing set. This provides two benefits: first, we obtain a *ranking* of the extracted terms, and second, it is guaranteed that each document of the testing set will have terms to represent it, which does not necessarily hold with the use of a global threshold or a discriminant classifier. Besides, R is a parameter of the algorithm which gives more control and enables to adjust the output of the algorithm when more precision or more recall is preferred. By default, the number of terms to extract is the average number of keywords of the documents of the training set.

4 Experiments and Results

To assess the performance of the proposed method we used a set of scientific articles published between 1999 and 2012 in the Workshop of Researchers in Computer Science (WICC) [20], and another set of scientific articles published in the XVII Argentine Congress of Computer Science (CACIC) [21] in 2011. With these documents we formed three datasets of articles in Spanish: two of them from the WICC corpus, of 43 and 166 documents respectively, and one formed by 72 articles from the CACIC corpus. In this paper these datasets are called *wicc1*, *wicc2* and *cacic2011*.

The description of these three datasets is shown in the Table 1. The class imbalance can clearly be seen, which is a factor that severely affects training of traditional learning algorithms.

Table 1. Description of the datasets used in the tests

	wicc1	*wicc2*	*cacic2011*
Number of documents	43	166	72
Total number of keywords	210	747	327
Total number of terms	7297	33428	34875
Number of keywords/number of terms rate	0.028	0.022	0.009

We performed tests of the proposed algorithm over the three datasets and we compared its performance with the performances of KEA 5.0, Maui 1.2 and LIKE. The implementations of KEA and Maui are the ones developed by its respective authors.

The metrics used were precision, recall and f_1-measure calculated for each of the four algorithms. These metrics were applied considering a hit the matching between a term selected by an algorithm and a term designated as keyword by the authors of the given document. Thus, a false positive ocurrs when a method identifies as keyword a terms that is not included in the list of keywords by the author, and a false negative when the method fails to extract a keyword contained in that list. In our case precision measures the proportion of extracted terms that match assigned keywords, and *recall* measures the proportion of keywords correctly identified by the method.

The methodology of evaluation we applied is 10-fold cross validation. This evaluation process was repeated 30 times to obtain a significative sample over which we can average the results. We configured all algorithms, with the exception of LIKE,

to extract three keywords because this is the average number of keywords per document on the three datasets. Since LIKE is a discriminant classifier without a ranking criterion, the number of extracted terms cannot be limited.

In our experiments we used an autoencoder with 10 hidden neurons, a maximum of 500 epochs, and the logistic function as activation function in the hidden and output layers. For LIKE, we also used 10 hidden neurons and a maximum of 500 epochs, and a learning rate of 0.1, the logistic function in the hidden layer and the hyperbolic tangent in the output layer. For KEA and Maui we applied the Spanish stemmers and stoplists provided with the implementations. In these experiments the terms extracted by the four methods have a maximum length of four words and a minimum frequency of three ocurrences in the document.

The average precision, recall and f_1-measure of the 30 runs of the cross-validation for each algorithm on each dataset are shown in the Table 2, identifying the algorithm introduced in this work as AE, for *autoencoder*.

Table 2. Precision, recall and f_1-measure of each method in the three datasets.

	wicc1			*wicc2*			*cacic2011*		
	P	R	F	P	R	F	P	R	F
AE	**0.362**	0.343	**0.344**	0.266	0.249	**0.256**	0.138	0.177	**0.153**
KEA	0.132	0.082	0.101	0.178	0.119	0.142	0.038	0.025	0.030
Maui	0.342	0.210	0.258	**0.308**	0.207	0.247	**0.179**	0.119	0.142
LIKE	0.117	**0.587**	0.175	0.137	**0.680**	0.218	0.019	**0.660**	0.035

The tests results show that the algorithm proposed in this paper has the highest f_1-measure in the three datasets. F_1-measure is the harmonic mean between precision and recall, and therefore it is a good measure of the global performance of a given method in similar datasets. Also it can be seen that even if the f_1-measure achieved by Maui is close to the one of our algorithm, in the three cases we achieved higher recall.

In order to verify that these differences are statistically significant, we conducted hypothesis tests on the difference of the means of the three metrics using the 30 samples of cross-validations of our algorithm and Maui. The tests showed that the differences are significant in all cases with a significance level of 0.05.

A high recall is important because it allows to comprise the maximum possible of eligible terms, which in turn gives the possibility of suggesting descriptive terms that were not chosen by the authors. However, getting a high recall at the expense of precision is not benefitial, since the quality of the extracted terms will be inferior. Hence it is necessary to find a balance between precision and recall.

On the other hand, in the *cacic2011*dataset all algorithms had a lower performance because, as we show on Table 1, this is a much more unbalanced dataset than the other two, since it has less keywords in relation to the total number of terms and thus it is a more difficult dataset.

5 Conclusions and Future Work

In this paper we presented a new algorithm for keyword extraction from Spanish documents. Our proposal uses only examples of the class of interest to train the classifier, and therefore training is conducted in a faster and more efficient way giving comparable or even better results than other algorithms. It is important to highlight the need to use a representative, comprehensive and noise-free training set to achieve better performance, even when such training set is small.

Given the inherent difficulty of this problem due to class imbalance and subjectivity of natural language, we consider interesting that an algorithm can learn from examples which are the features that determine whether a term is important or not.

Also, we consider important to achieve a high recall so that the algorithm can capture more terms eligible by different human observers, with the goal to act as a recommendation system of possible keywords. The only language-dependent of our method is the POS tagging model, thus replacing this model with a model trained with documents in another language would allow us to apply our method in such language.

Finally, given that the number of terms to extract is a parameter of the algorithm the user can adjust the expected level of precision or recall from the terms suggested by the system.

As future work we aim to enrich the term representation including semantic attributes related to the grammatical structure of the language considered. We are also interested in refining the grammatical filtering in order to reduce further the class imbalance and to consider the use of more sophisticated classification strategies.

References

1. Turney, P.D.: Learning Algorithms for Keyphrase Extraction. Information Retrieval, vol. 2, 303--336 (2000).
2. Witten, I.H., Paynter, G.W., Frank, E., Gutwin C., Neville-Manning, C. G.: KEA: Practical Automatic Keyphrase Extraction. In Proceedings of the 4th ACM Conference on Digital Libraries, pp. 254--255 (1998).
3. Salton, G., Buckley, C.: Term-weighting approaches in automatic text retrieval. Information Processing and Management, 513--523 (1988).
4. Aquino, G, Hasperué, W, Estrebou, C, Lanzarini, L. A Novel Language-Independent Keyword Extraction Method. XIX Congreso Argentino en Ciencias de la Computación (2013).
5. Japkowicz, N.: The Class Imbalance Problem: Significance and Strategies. Proceedings of the 2000 International Conference on Artificial Intelligence (ICAI), 111--117 (2000)
6. Medelyan, O.: Human-competitive automatic topic indexing. Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, vol. 3, 1318--1327, Association for Computational Linguistics (2009).
7. Breiman, L.: Bagging Predictors. Machine Learning, 123--140 (1996).
8. WEKA, http://www.cs.waikato.ac.nz/ml/weka/, accedido en Junio de 2014.
9. Wikipedia Miner, http://wikipedia-miner.cms.waikato.ac.nz/, accessed in June 2014.
10. Japkowicz, N, Myers, C, Gluck, M.: A Novelty Detection Approach to Classification. Proceedings of the Fourteenth Joint Conference on Artificial Intelligence, 518--523 (1995).

11. Fürnkranz, J.: A Study Using n-gram Features for Text Categorization (1998).
12. Ratnaparkhi, A.: A Maximum Entropy Model for Part-Of-Speech Tagging (1996).
13. OpenNLP, http://opennlp.apache.org/, accessed in June 2014.
14. Conference on Computational Natural Language Learning (CoNLL-2002), http://www.clips.ua.ac.be/conll2002/ner/
15. Expert Advisory Group on Language Engineering Standards (EAGLES), http://www.ilc.cnr.it/EAGLES96/home.html, accessed in June 2014.
16. Andrade, M.A., Valencia, A.: Automatic extraction of keywords from scientific text: application to the knowledge domain of protein families. Bioinformatics, vol. 14, no. 7, 600--607 (1998).
17. Manevitz, L, Yousef, M.: One-class Document Classification via Neural Networks. Neurocomputing, vol. 70, no. 7-9, 1466--1481, Elsevier Science Publishers B. V. (2007).
18. Riedmiller, M.: Advanced Supervised Learning in Multi-layer Perceptrons - From Backpropagation to Adaptive Learning Algorithms (1994).
19. Igel, C, Hüsken, M.: Improving the Rprop Learning Algorithm (2000).
20. Workshop de Investigadores en Ciencias de la Computación, http://redunci.info.unlp.edu.ar/wicc.html, accessed in June 2014.
21. Congreso Argentino en Ciencias de la Computación, http://cacic.info.unlp.edu.ar/, accessed in June 2014.