Keyword Identification in Spanish Documents using Neural Networks

Germán Aquino[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, 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 an algorithm for keyword extraction from documents written in Spanish.This algorithm combines autoencoders, which are adequate for highly unbalanced classification problems, with the discriminative power of conventional binary classifiers. In order to improve its performance on larger and more diverse datasets, our algorithm trains several models of each kind through *bagging*.

Keywords: Keyword Extraction, 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. Keywords are widely used in search engines as they help in the process of searching, indexing, and retrieving information [1]. 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 [2].

In this paper a new algorithm for keyword extraction from text documents written in Spanish language is presented. This algorithm is based on a classification model capable of learning the structural features of the terms considered keywords, and to recognize terms having these features in unseen documents. A combination of discriminant classifiers and autoencoders is used to build a classification model that assigns a score to each term of a document. This score is used to construct a ranking of the terms considered most informative for a given document.

This paper is organized as follows. Some algorithms for keyword extraction are described in Section 2. The proposed algorithm is explained in detail in Section 3.

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

2 Related Work

The problem of keyword extraction has been treated from the machine learning discipline since a few decades ago [2][3][4]. This approach aims to transform text data into a structured representation suitable for learning algorithms. Such algorithms work with a feature set computed 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 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 and the inherent ambiguity of natural language makes keyword extraction a very difficult problem to solve. Many of the mistakes made by the keyword extraction algorithms, specially those which apply supervised classification schemes, are due to redundancy (in the case of several semantically-equivalent terms are selected) and over-generalization (in the case of selection of terms that contain important terms but are not keywords themselves). The flexibility of the vocabulary used and the ambiguity of the human language makes very difficult for automatic classifiers to distinguish between two seemingly equivalent terms, and to see a relation between subtly related terms [5].

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 was reported by Peter Turney [2]. 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.

Among the most recent algorithms for keyword extraction there is Maui, developed by Olena Medelyan [6][7]. Maui is also a supervised classification algorithm that computes a set of features for the candidate terms. Maui uses a *stemmer* and a *stoplist* of the given language and it is built on top of the machine learning platform Weka [8] and uses bagged decision trees to classify terms.

In a previous work [9] we introduced a keyword extraction algorithm that relies on auto-associative neural networks or autoencoders [10] to identify keywords. This algorithm uses only the elements belonging to the minority class, the class of the keywords, to build a recognition model as opposed to discriminative models obtained using conventional neural networks and other machine learning algorithms. The autoencoder approach has the advantage that it handles naturally the imbalance inherently present in the keyword extraction problem, and also it enables to control the number of keywords extracted from each document and to rank them. Also, it is much faster than other algorithms as it processes only the examples of the minority class.

The algorithm presented in this paper is also a supervised machine learning algorithm, and it is an improvement over our previous approach as it combines qualities of both discrimination-based (supervised) and recognition-based (unsupervised) classifiers in order to improve performance on larger and less regular datasets. The potentially large variance present in the training and testing examples is handled through the use of *bagging* [11] in order to average the classification decisions of different classifiers. As its predecessor, the proposed algorithm does not use *stoplists* to rule out insignificant or malformed terms but instead it applies *part-of-speech (POS) tagging* to allow the correct identification of noun phrases present in the text.

3 Description of the Algorithm

In this work autoencoders are used to classify terms in two classes, 'keyword' and 'non-keyword'. Autoencoders are adequate for unbalanced classification problems and one-class recognition problems [12]. To enhance their recognition capabilities, several autoencoders are combined by the use of bagging and also a set of discriminant classifiers is used.

An autoencoder processes examples of only one class. Autoencoders try to find an approximation of the training set to itself, finding in the process an approximation to the identity function of such training set. This allows them to assign a reconstruction error that characterizes the similarity between a new element and the training set. On the other hand, discriminant classifiers attempt to find a possibly non-linear boundary in the feature space of the training examples in order to define regions in such space for each class. Here, the decision of discriminant classifiers is used to weight the reconstruction error assigned to the examples by the autoencoders. Both kinds of classifiers made decisions through a voting scheme, which will be explained further in Section 3.3.

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 the 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. In this work we will use 'term' and 'N-gram' interchangeably. In the N-grams extraction task the Fürnkranz algorithm [13] is applied for avoiding the generation of every possible N-gram from the text and increasing the efficiency in the generation of N-grams. This algorithm requires the specification of the maximum length of the terms considered and the minimum frequency such terms must have in a document to be eligible as keywords.

In order to further reduce the number of terms to be processed, after the feature calculation phase we apply a filter which discards N-grams that do not start or end with nouns or adjectives. This filtering discards sequences of words that are not eligible as keywords, for example '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 *POS* tags to each word of the document based on its use. To this end we apply a *maximum entropy* model trained with the tool OpenNLP [14] using a tagged corpus as training set. This filtering greatly reduces the required processing time, since it discards an important number of terms that should not be considered as keywords.

The POS tagging model for Spanish was trained using the tagged corpus Conll-2002 [15] and the grammatical tags defined by the EAGLES group [16]. 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 (NER), which is the problem of finding person names, places, organizations and similar information in the text.

3.2 Term characterization

The features computed for each N-gram consist of several frequential and positional quantities extracted from the text. Most of these features are computed using only the information present in each document, but some of them require the processing of the entire training corpus for their computation. The features are:

1. **Term length:** the number of individual words composing the N-gram.
2. **Term Frequency (TF):** the rate between the frequency of the term and the number of words in a document.
3. **Inverse Document Frequency:** it measures how common is a given term by counting how different documents in the corpus contain it.
4. **Term Frequency – Inverse Document Frequency (TF-IDF) [17]:** consists in weighting the term frequency with the inverse document frequency. TF-IDF favors terms that are infrequent in the corpus but frequent in the given document.
5. **First Occurrence:** the relative position of the first occurrence of the term in the text. It is calculated as the ratio between the number of words that appear before the first occurrence of the given term and the number of words of the document.
6. **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.
7. **Occurrence 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.
8. **Occurrence 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 that have more or less lexical words. It is the ratio between the number of words of a term *t* that appear in the title and the length of *t*.

9. **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.
10. **Normalized Frequency (Z-Score) [18]:** 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.
11. **Last occurrence:** the last position in the text in which the term appears.
12. **Spread:** the difference between first and last occurrences.
13. **Normalized frequency:** the frequency of the term normalized by the highest frequency of any term in the document.
14. **Lowest position in sentence:** considering all the positions a term occupied in each of its sentences, this is the closest to the beginning of the sentence, normalized using the sentence length.
15. **Highest position in sentence:** similar to the previous one, but considering the position closest to the end of the sentence.
16. **Shortest sentence length:** the length of the shortest sentence a term appears in, normalized by the highest length of any sentence.
17. **Longest sentence length:** similar to the previous one, but considering the longest sentence a term appears in.
18. **Log frequency:** a non-linear monotonic function is applied to the term frequency in order to reduce the impact of its absolute value but at the same time to keep its magnitude.
19. **Condition of being a named entity:** this is a boolean feature that indicates if the term is a named entity or not. To identify named entities in the document a NER OpenNLP model is applied.
20. **Keyphraseness [3]:** the number of times a given term was chosen as a keyword in the training set. It makes sense if the testing documents belong to the same domain as the training documents, which should be the case to obtain a reasonable performance.

3.3 Keyword Identification

As mentioned earlier, the proposed method is a supervised classification algorithm. It uses the feature vectors of the terms of the training document set in order to build a classification model to be applied to the feature vectors of a testing document set.

In the proposed method three ensembles of classifiers are used. The first ensemble is composed of conventional bagged *multi-layer perceptrons*, trained using sampling with replacement from the training set. In order to cope with the imbalance problem, the number of elements that are sampled from the majority class is proportional to the sampled number of elements in the minority class. As all of these sampled smaller training sets are different, the resulting classifiers will yield different views on the original feature space. Given the large variance present in the problem domain and the intrinsic non-deterministic nature of neural networks, *bagging* helps to improve the performance of the obtained models, giving more consistent and more robust predictions. These classifiers are trained to distinguish important terms from non-important ones.

The other two ensembles are composed of autoencoders. The first of these two ensembles attempts to characterize the set of elements belonging to the minority class (the positive set), which in our case are the feature vectors of the terms designed as keywords in the training set. The other ensemble attempts to characterize the set of elements belonging to the majority class (the negative set), which is naturally much more diverse. Both ensembles are also trained applying *bagging*, and the autoencoders of the majority class are trained with larger samples in order to provide more accurate estimates of the complete set.

Autoencoders are neural networks that have as many output units as they have input units, so given an input vector X they can produce an approximate vector X'. The difference between the original vector and the approximate vector can be characterized by the reconstruction error, which is the sum of the squared differences between both vectors. 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.

The autoencoders are trained in the same way as conventional neural networks. In this work we used *Resilient Backpropagation* [19] as training algorithm, both for the autoencoders and the multi-layer perceptrons. This algorithm allows a faster convergence, providing better results, and at the same time it eliminates the need to specify a learning rate.

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. As we are using two sets of autoencoders, one for the positive class and one for the negative class, we have two scores for each term of the testing set. Let Pos_e be the reconstruction error of the term in respect to the positive set, and Neg_e the reconstruction error in respect to the negative set. An informative term should minimize Pos_e, as it should be similar to the elements in the positive set, and at the same time it should maximize Neg_e, its dissimilarity to the negative set. Hence, an informative term should minimize $Pos_e - Neg_e$, and this is the score used to construct the term ranking. The selection scheme employed gives preference to the terms chosen by the discriminant classifiers as informative terms, and then their reconstruction error is considered.

The use of the reconstruction error as a selection mechanism 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 allows the user 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 Experimental Results

Some experiments were carried out to assess the performance of the proposed method. A dataset formed by a set of scientific articles published between 2005 and 2013 in Argentine Congress of Computer Science (CACIC) [20] was used in these experiments. The dataset includes 888 documents written in Spanish language and contains 130792 terms from which 1683 are labeled as keywords, giving an imbalance rate of 1.28%, that is, less than 2% of all terms belong to the minority class. We also used a dataset composed of 166 scientific articles from the Workshop of Researchers in Computer Science (WICC) [21]. This dataset was used to measure the performance of the previous version of our method [9], and it is used here to assess that the new version is indeed superior.

The metrics used were precision, recall and f_1-measure calculated for each of the four algorithms.These metrics were applied considering as a hit the match between a term selected by an algorithm and a term designated as keyword by the authors of the given document. Thus, a false positive occurs 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 recallmeasures the proportion of keywords correctly identified by the method. 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.

The evaluation methodology 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 both algorithms to extract 5 keywords as this is the average number of keywords per document on the dataset.

In our experiments we used 15 multi-layer perceptrons as discriminant classifiers, 5 autoencoders for the positive set, and 10 autoencoders for the negative set. All these neural networks were trained using 20 hidden neurons, a maximum of 50 epochs, and the logistic function as activation function in the hidden and output layers. The implementation used of Maui is the one developed by its authors. For Maui we applied the Spanish stemmers and stoplists provided with the implementations. For the previous version of our method, the autoencoder was configured to use 15 hidden neurons, a maximum of 100 epochs, and the same activation functions as the new version. In these experiments the terms extracted by all methods have a maximum length of 4 words and a minimum frequency of 3 occurrences in their respective documents.

The results of the 30 runs of the cross-validation for each algorithm on each dataset are shown in the Figure 1, identifying the proposed algorithm as AE*, for autoencoder. The previous version of our method is simply denoted as AE.

The tests results show that the proposed algorithm outperforms Maui on these datasets. It can be seen also that it handles properly larger and more diverse datasets than its predecessor. One of the main goals of our algorithm is to capture the largest possible number of descriptive terms, and this goal is quantified by the recall metric. A high recall is important because it allows capturing 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 beneficial, since the quality of the extracted terms will be inferior. Therefore it is necessary to find a balance between precision and recall.

In order to verify that these differences are statistically significant, we ran a Kolmogorov-Smirnov test on the results of the precision, recall and f-measure obtained from the cross-validation procedure for both methods, and we ran a t-test on the difference of the means of the samples for the three metrics. The tests showed that the mean for the three metrics obtained by our method are higher than the ones obtained by Maui with a significance level of 0.05, as the obtained p-values are 1.3669e-30, 3.7699e-40 and 4.2676e-35 respectively.

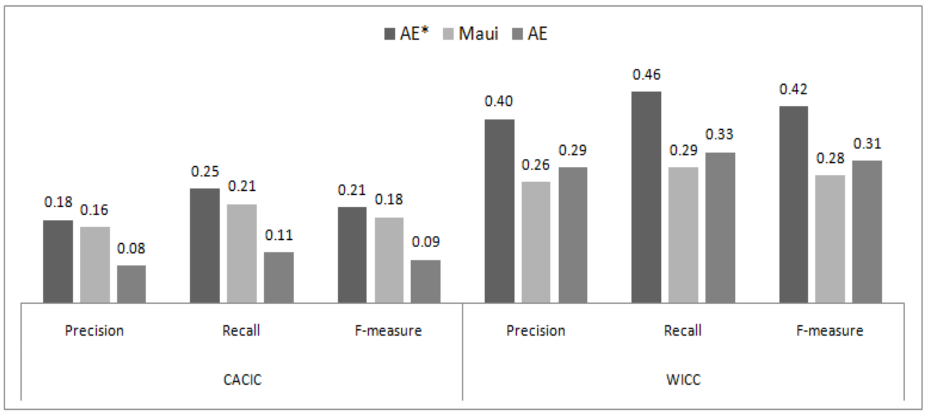

Figure 1. Average precision, recall and f_1-measure of the three methods on the CACIC dataset.

In the Table 2 there are shown the lists of keywords extracted of both methods for a set of documents from the CACIC dataset, and these keywords are compared to the real keywords assigned by the authors of the respective documents. The matches between an extracted keyword and a real one are highlighted in bold. It is important to notice that some of these documents have fewer keywords than the specified number of keywords to extract. This necessarily means that the methods will have false positives errors, despite the selected terms may be considered descriptive by a human observer. It is also noteworthy that some terms are semantically equivalent to the true keywords, but as they are not exact matches are hence considered false positives too. The high variability of the keyword assignment criteria of the authors, combined with the ambiguity of the human language contributes to the high difficulty of the keyword extraction problem. These issues could be addressed by the use of semantic knowledge bases that could map related terms to the same concept, and by the definition of more advanced scoring criteria for performance assessment than exact matching.

Table 2. Comparative results of the keyword extraction methods performance on some sample cases.

Documents in dataset	Keywords assigned by authors	Keywords extracted by AE*	Keywords extracted by Maui
Una implementación paralela de las Transformadas DCT y DST en GPU.	-procesamiento paralelo **-GPU** **-CUDA** **-procesamiento de señales** **-DCT**	-transformadas **-GPU** **-CUDA** -GPU CUDA **-procesamiento de señales**	-MPI **-DCT** -transformadas -DST **-CUDA**
Programación híbrida en clusters de multicore.	-arquitecturas paralelas **-programación híbrida** **-cluster** **-multicore** **-jerarquía de memoria**	**-cluster** **-multicore** -programación **-programación híbrida** **-jerarquía de memoria**	**-jerarquía de memoria** **-cluster** **-multicore** -pasaje de mensajes -caso de estudio
Evaluación de variantes en modelo destinado a anticipar la conveniencia de trazar proyectos de software.	**-ingeniería de software** **-análisis ROC** **-trazabilidad de requerimientos**	-trazabilidad -métricas **-análisis ROC** **-ingeniería de software** **-trazabilidad de requerimientos**	-ROC -trazabilidad -métricas -variantes -factores
Autorregulación del aprendizaje en entornos mediados por TIC.	**-autorregulación** **-TIC** **-aprendizaje**	**-autorregulación** **-TIC** **-aprendizaje** -intervención -autorregulación del aprendizaje	**-aprendizaje** **-TIC** -propuesta de intervención **-autorregulación** -intervención
Integración segura de MANETs con limitaciones de energía a redes de infraestructura.	-MANET **-bluetooth** **-IPSec** **-energía** **-seguridad**	**-bluetooth** **-IPSec** -MANETs **-energía** -ad hoc	**-seguridad** **-Bluetooth** **-IPSec** -consumo -consumo de energía

5 Conclusions and Future Work

In this paper we presented a new algorithm for keyword extraction from Spanish documents. The main feature of our proposal is the use of autoencoders to capture the properties of important terms, yielding comparable or even better results than other well known keyword extraction algorithms. Autoencoders classification decisions are further reinforced by the use of discriminant classifiers. 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 are the POS tagging and NER models, thus replacing these models with models trained with documents in another language would allow us to apply our method in such language.

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.

We are currently working on the term representation to include features related to the grammatical structure of a given language, as the use of parsing trees in order to find head noun phrases in sentences. We are also interested in incorporating the use of knowledge bases in order to find semantic relations between pairs of terms and to identify their degree of generality or specificity in a given domain.

References

1. Gutwin, C., Paynter, G., Witten, I., Nevill-Manning, C., Frank, E.: Improving Browsing in Digital Libraries with Keyphrase Indexes. Journal of Decision Support Systems, Vol.27, no 1-2, pp.81--104. (1999)
2. Turney, P.D.: Learning Algorithms for Keyphrase Extraction. Information Retrieval, vol. 2,303--336 (2000).
3. 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).
4. Hulth, A.: Improved automatic keyword extraction given more linguistic knowledge. In Proceedings of the 2003 Conference on Empirical Methods in NLP, pp. 216--223 (2003).
5. Hasan, K. S., Ng V.: Automatic Keyphrase Extraction: A Survey of the State of the Art. In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 1262--1273 (2014).
6. Medelyan, O.: Human-competitive automatic topic indexing. Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, vol. 3, pp. 1318--1327, Association for Computational Linguistics (2009).
7. Kim, S. N., Medelyan, O., Kan, M., Baldwin, T. SemEval-2010 Task 5: Automatic Keyphrase Extraction from Scientific Articles. In Proceedings of the 5th International Workshop on Semantic Evaluation. pp. 21--26 (2010).
8. WEKA, http://www.cs.waikato.ac.nz/ml/weka/, accessed in July 2015.
9. Aquino, G, Hasperué, W, Lanzarini, L. Keyword Extraction using Auto-associative Neural Networks. XX CongresoArgentino en Ciencias de la Computación (2014).
10. Japkowicz, N, Myers, C, Gluck, M.: A Novelty Detection Approach to Classification. Proceedings of the Fourteenth Joint Conference on Artificial Intelligence, pp. 518--523 (1995).
11. Breiman, L.: Bagging Predictors. Machine Learning, pp. 123--140 (1996).
12. Japkowicz, N.: The Class Imbalance Problem: Significance and Strategies. Proceedings of the 2000 International Conference on Artificial Intelligence (ICAI), pp. 111--117 (2000).
13. Fürnkranz, J.: A Study Using n-gram Features for Text Categorization (1998).
14. OpenNLP, http://opennlp.apache.org/, accessed in July 2015.
15. Conference on Computational Natural Language Learning (CoNLL-2002), http://www.clips.ua.ac.be/conll2002/ner/, accessed in July 2015.
16. Expert Advisory Group on Language Engineering Standards (EAGLES), http://www.ilc.cnr.it/EAGLES96/home.html, accessed in July 2015.
17. Salton, G., Buckley, C.: Term-weighting approaches in automatic text retrieval. Information Processing and Management, pp. 513--523 (1988).
18. 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, pp. 600--607 (1998).
19. Riedmiller, M.: Advanced Supervised Learning in Multi-layer Perceptrons - From Backpropagation to Adaptive Learning Algorithms (1994).
20. Congreso Argentino en Ciencias de la Computación, http://redunci.info.unlp.edu.ar/cacic.html, accessed in July 2015.
21. Workshop de Investigadores en Ciencia de la Computación, http://redunci.info.unlp.edu.ar/wicc.html, accessed in July 2015.