F. Martínez, M.T. Martín, V. M. Rivas, M.C. Díaz, L.A. Ureña
Depto. de Informática, Universidad de Jaén. Avda. Madrid, 35, 23071, Jaén (Spain)

Using Neural Networks for Multiword Recognition in IR

Abstract: In this paper, a supervised neural network has been used to classify pairs of terms as being multiwords or non-multiwords. Classification is based on the values yielded by different estimators, currently available in literature, used as inputs for the neural network. Lists of multiwords and non-multiwords have been built to train the net. Afterward, many other pairs of terms have been classified using the trained net. Results obtained in this classification have been used to perform information retrieval tasks. Experiments show that detecting multiwords results in better performance of the IR methods.

1. Introduction

In this paper we present a new approach to solve the task of effectively detecting multiwords. A multiword is a succession of words whose sense taken as a whole differs from the sum of the senses of its single words. Thus, a multiword can be considered in fact as a new concept.
There exists a second kind of multiword composed of a set of words that complement their senses. Nevertheless, considering this kind of successions as multiwords does not provide information useful for information retrieval tasks. For this reason, this kind of multiwords have not been considered in the present work.
Multiword detection can be successfully used in many different tasks. Information Retrieval (IR) methods, for instance, use the word as the information basic unit; thus, detecting multiwords in corpus and queries make IR systems getting better results. Cross Language Information Retrieval can be also improved by detecting multiwords, given that translating word by word results in lost of information. Natural Language Processing can also be helped when multiwords are correctly detected, because it makes easier text understanding.
The new approach presented in this paper uses neural nets to discriminate against pairs of terms that really are multiwords from those that are not. We propose a well-known supervised neural network: the Kohonen's Learning Vector Quantization (LVQ) widely used for classification tasks (Kohonen, 1995), (Kohonen, 1992). Inputs for the nets are the values yielded by estimators used in literature to perform this same task, and the output the nets provide is a class determining if that values corresponds to a multiword or a non-multiword. Nets learning is performed by training them with the values yielded by the cited estimators when they are applied to pairs of terms known to be either multiwords or non-multiwords. In order to test this new method, the network has been used to classify new pairs of terms; then, the information obtained has been used to perform some IR tasks. Experiments show that results obtained in these task are better than those obtained using only the estimators.
The rest of the paper is organized as follows: section 2 gives a little introduction to the state of the art, briefly showing some of the currently available methods used to detect multiwords. These methods include the different estimators that will be lately used in our method. Section 3 describes a new estimator developed for this work as well as the neural network that has been used. Section 4 shows the experiments carried out and the results obtained. Finally, section 5 outlines some conclusions, and also future research lines.

2. A new approach

Multiword recognition has been explored by many researchers as a way to improve traditional Text Retrieval, in general with a moderate degree of success. However, David Hull and Gregory Grefenstette (Hull, 1996) show that multiword detection and correct translation largely improve the precision in a CLIR system.

Usually methods for automatizing terminological procedures have traditionally been statistical (Hull, 1996), (Ballesteros, 1998), and based on the co-occurrence of each particular pair of words in the text of work or corpus. Other works (Adriani, 1999) obtain the degree of similarity between terms using the co-occurrence factor, and the standard *tf*idf* term weighting formula. Recently, hybrid approaches incorporating linguistic information have been developed: Diana Maynard and Sophia Ananiadou (Maynard, 2000) make use of different types of contextual information: syntactic, semantic, terminological and statistical. Nevertheless, managing different types of information must be done by integrating them in any given way. The most straightforward way is by using a linear function, although this does not mean it is the best way this problem can be faced.

For any of the features (syntactical, semantical, terminological and statistical) wanted to be integrated to perform multiword detection, there are some well-known estimators. This paper introduces a neural network based approach that integrates terminological and statistical estimators. Multiword detection is then thought as a categorization problem where only two categories have to be managed: multiword and non-multiword. Consequently, classifying a pair of terms turns into a two step process: firstly, obtain the values yielded by the different estimators; secondly, use those values as inputs for the neural network, and obtain the class to which the pair of terms belongs. More precisely, the estimators that have been used in this work are the following:

1. *Pearson's* χ^2. A variant of the χ^2 statistic (Hull, 1996)
2. Measure the importance of co-occurrence of the elements in a set by the *em* metric (Ballesteros, 1998)
3. *Dice similarity coefficient* obtain the degree of similarity or association-relation between terms using a term association measure and the tf.idf weighting formula (Adriani, 1999).
4. The *mutual information ratio*, or association ratio, μ (Johansson, 1996).
5. Finally, a new estimator, a variant of Dice similarity coefficient based on the Simpson index, has been developed.

2.1. A new estimator: Simpson Similarity coefficient

Roughly, Dice index is based on the association between two terms by calculating the coefficient of the intersection of two sets and their union. Usually, this approach is convenient to estimate the correlation between words, but not always. "Bill Clinton", for instance, is a multiword, but "Bill" is a very common word, so the term frequency is very high, and "Bill" set is huge. In the other hand, "Clinton" is not too frequent, so "Clinton" set is small. Thus, the coefficient of the intersection and the union of both sets will be small, because "Clinton" set is small. In other way, Simpson index estimates the association between two sets by calculating the coefficient of the intersection of two sets and the *smaller* of them, so "Bill Clinton" will reach a high value for the Simpson coefficient, and a low value for the Dice coefficient.

$$DICE:\ xy = 2\frac{\sum_{i=1}(w'_{xi}\cdot w'_{yi})}{\sum_{i=1}w_{xi}^2+\sum_{i=1}w_{yi}^2} \qquad SIMPSON:\ xy = 2\frac{\sum_{i=1}(w'_{xi}\cdot w'_{yi})}{\min\left(\sum_{i=1}w_{xi}^2,\sum_{i=1}w_{yi}^2\right)}$$

where:
w_{xi} = the weight of term x in the document i.
w_{yi} = the weight of term y in document i.
$w'_{xi} = w_{xi}$ if term y also occurs in document i, or 0 otherwise.
$w'_{yi} = w_{yi}$ if term x also occurs in document i, or 0 otherwise.
n = the number of documents in the collection.

2.2. Neural Network approach: The LVQ algorithm

The LVQ algorithm is a classification method based on neural competitive learning, which allows to define a group of categories on the space of input data by a reinforced learning, either positive (prize) or negative (punishment). LVQ uses supervised learning to define class regions in the input data space. To this end a subset of similarly labeled codebook vectors is placed into each class region.

Given a sequence of input data, an initial group of reference vectors w_k (codebook) is selected. In each iteration, a input vector x_i is selected and the vectors W are updated, so that they fit x_i in a better way. The LVQ algorithm works as follows:

For each class, k, a weight vector w_k is associated. In each repetition, the algorithm selects an input vector, x_i , and compares it with every weight vector, w_k, using the euclidean distance $||x_i-w_k||$, so that the winner will be the weight vector w_c nearest to x_i, being c its index:

$$\left\|x_i - w_i\right\| = \min_k\left\{\left\|x_i - w_k\right\|\right\}$$

The classes compete between them in order to find the most similar to the input vector, so that the winner is the one with less euclidean distance regarding the input vector. Only the winner class will modify its weights using a reinforced learning algorithm, either positive or negative, depending on the classification being correct or not. Thus, if the winner class belongs to the same class of the input vector (the classification has been correct), it will increase the weights, coming slightly close to the input vector (prize). On the contrary, if the winner class is different from the input vector class (the classification has not been correct), it will decrease the weights, coming slightly far from the input vector (punishment).

Let $x_i(t)$ be an input vector at time t, and $w_k(t)$ represents the weight vector for the class k at time t. The following equation defines the basic learning process for the LVQ algorithm.

$$w_c(t+1) = w_c(t) + s\cdot\alpha(t)\cdot\left[x_i(t) - w_c(t)\right]$$

where $s = 0$, if $k \neq c$; $s = 1$, if $x_i(t)$ and $w_c(t)$ belong to the same class; and $s = -1$, if they do not, and where $\alpha(t)$ is the learning rate, being $0<\alpha(t)<1$, a monotonically decreasing function of time. It is recommended that $\alpha(t)$ be rather small initially, say, smaller than 0.5, and that it decreases to a given threshold, u, very close to 0 (Kohonen, 1995).

The experiments showed in section 4, were carried out using the implementation described in LVQ_PAK documentation (Kohonen, 1991) with default parameters. Thus, every experiment

started with the same number of codebooks per class (10 for class 0 and 10 for class 1) and the learning rate being initialized to 0.3.

3. Experiments and results

In order to train and test the neural nets, a set of samples composed of input-output pairs had to be built, every sample corresponding to a pair of terms. In one hand, input values were obtained by applying the different estimators described in section 3. In the other, every output value consisted on a single number classifying the sample as multiword or non-multiword. In ours experiment only multiwords with two relevant terms have been used, and stop words have been removed from the multiword

Obtaining a list of multiwords was done by resorting to WordNet (Miller, 1995), a lexical database where multiwords can be found. Nevertheless, not all the pairs of terms said to be multiwords really were. For this reason, each multiword returned by WordNet was newly searched in the electronic dictionary Encarta[1] to remove pairs of terms that, even appearing together very frequently, were not real multiwords.

Non-multiwords list (needed to train the nets) was taken from the corpus used in CLEF 2000[2] . Pairs of terms were taken from this corpus and then searched in the list of multiwords previously described, checking that they did not appear in it. If they did not appear, they were once more searched in the electronic dictionary to assure they did not formed a multiword.

Once both multiwords and non-multiwords lists have been created, the above cited estimators were applied to them, obtaining the file with the samples to be used with the supervised network. This file was split to use 75% of the samples to train the neural network and the remaining 25% to validate it.

Los Angeles Time 1994 collection, borrowed from the English CLEF 2000 collection, was used to test the method. This collection is composed of 113,005 articles of the 1994 edition of Los Angeles Times, and 40 queries (Title + Description) with relevance judgments. The collection was indexed twice using *Zprise* software[3], with Okapi (Robertson, 2000) weighting formula. First index was created without carrying out multiword detection, while second index uses multiword detection, as depicted above.

Table 1 shows average precision reached by both methods. It shows that multiwords usage improves precision scarcely. Anyway, a more detailed analysis of the results leads to conclude that multiwords detection is useful for IR task. Table 2 shows the precision reached by some queries, and the detected multiwords for each one.

Original query set	*Query set with multiwords detection*
0.375	0.410

Table 1 – Average Precision

Query	*Original AvgP.*	*AvgP. With Multiwords*	*Detected multiwords*
#7	0.3969	0.4452	"world soccer"
#9	0. 1022	0.2027	"war ii" "ii war" "war rwanda" "world war"
#3	0.3912	0.3220	"decisions made", "hard soft"
#32	0.4126	0.2511	"women priest", "change direction"

Table 2 – Four detailed queries.

As table 2 shows, query #7 gained 5% of absolute precision because "world soccer" was effectively detected as a multiword. Results were even better in query #9, in which "world war", "war rwanda" "war ii" and "ii war" multiwords were correctly detected. As can be seen, precision obtained in this query by the new method is twice the precision obtained without multiword detection.
In the other side, query #3 lost 7% of precision with multiword inclusion. Bigrams "decisions made" and "hard soft" are in fact non-multiwords, but the neural network method marked both of them as being. Finally, query #32 lost 16% of precision. "women priest" and "change direction" because, once more, they are not multiwords.

4. Conclusions and future work.

This paper presents a new method to detect multiwords. This method uses the values obtained by estimators, present in literature and developed to perform this same task, as inputs for a neural net that automatically determines whether those values belong to a real multiword or simply to a pair of terms that appear together in a document.
Results show that automatic multiword detection is useful for IR. Nevertheless, the method used must get a higher accuracy, because bad detection of multiwords damages precision of the IR system. Conservative methods must be used to assess multiwords. Classifying multiwords as non-multiwords is better than recognizing too many multiwords. In other words, multiword detection must improve precision over recall.
Future lines of research include the use of new kind of neural networks, such as Radial Basis Function Nets (Broomhead, 1988) (Rivas, 2001), as well as RCE (Zboril, 2000), and also unsupervised training networks as Self-Organization Maps (Kohonen, 1995).
New estimators based on semantic information can be used to improve the results. Others applications for this method must also be investigated, especially its influence in Cross Language Information Retrieval.

Notes

[1] Encarta is available at http://www.encarta.com [2/2/2002]. Encarta has been used because it includes proper nouns that are considered to be multiwords.
[2] Cross Language Evaluation Forum (CLEF) aims at promoting research and development in CLIR. For more information, see: http://www.clef-campaign.org
[3] ZPrise is a software developed by NIST. It is available at http://www.itl.nist.gov/iaui/894.02/works/papers/zp2/zp2.html [2/2/2002]

References

(Adriani, 1999) Adriani, M. and C.J. van Rijsbergen. Term Similarity Based Query Expansion for Cross Language Information Retrieval. In *Proceedings of Research and Advanced Technology for Digital Librarie*s, Third European Conference (ECDL'99), p. 311-322. Springer Verlag: Paris, September 1999.

(Ballesteros, 1998) Ballesteros, L. and Croft, W.B. *Resolving ambiguity for cross-language retrieval*. In: Croft, W.B., Moffat, A., van Rijsbergen, C.J., Wilkinson, R. and Zobel, J. eds. Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. New York, NY: ACM, 1998, 64--71.

(Broomhead, 1988) D.S. Broomhead, D.Lowe. *Multivariable Functional Interpolation and Adaptative Networks*. In Complex Systems, vol. 11, pp.321-355.. 1988

(Hull, 1996) David A. Hull, Gregory Grefenstette. *Experiments in Multilingual Information Retrieval*. In Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. 1996

(Johansson, 1996). Christer Johansson. *Good Bigrams*. In Proceedings from the 16th International Conference on Computational Linguistics (COLING-96). Copenhagen: 592-597. 1996

(Kohonen, 1991) T. Kohonen, J. Hynninen,J. Kangas, J. Laaksonen, K. Torkkola. *LVQ_PAK: The Learning Vector Quantization program package*. Helsinki University of Technology Laboratory of Computer and Information Science. Finland, 1991-1995.

(Kohonen, 1992) T. Kohonen, J. Kangas, J. Laaksonen, K. Torkkola. *LVQ_PAK: A program package for the correct application of Learning Vector Quantization algorithms*. In Proc. of the International Joint Conference on Neural Networks, Pages I 725-730, Baltimore, June 1992. IEEE.

(Kohonen, 1995) T. Kohonen, *Self-Organization and Associative Memory*. 2nd Ed. Springer-Verlag, Berlin, 1995.

(Maynard, 2000) Diana Maynard and Sophia Ananiadou. *TRUCKS: a model for automatic term recognition*, Journal of Natural Language Processing, December 2000.

(Miller, 1995) G. Miller. *WORDNET: A lexical database for English*. Communications of the ACM, 38 (11), 1995.

(Salton, 1983) Salton, Gerard, and McGill, Michael J. *Introduction to Modern Information Retrieva*l, New York: McGraw-Hill, 1983.

(Rivas, 2001) V.M. Rivas, J.J. Merelo, P.A. Castillo. *Evolving RBF Neural Networks.* Lecture Notes in Computer Science, vol. 2064, pp.506-513. 2001

(Robertson, 2000) Robertson, S. E., Walker, S. & Beaulieu, M. *Experimentation as a way of life: Okapi at TREC*. Information Processing & Management, 36(1), 95-108. 2000

(Zboril, 2000) Zboril, F. Zboril,F. *The use of the RCE network in a Pattern Recognition*. Proceedings of MOSIS 2000, pp. 65-70. 2000