Available at
www.ElsevierComputerScience.com
POWERED BY SCIENCE DIRECT•
Neurocomputing 55 (2003) 665–679

NEUROCOMPUTING

www.elsevier.com/locate/neucom

LVQ for text categorization using a multilingual linguistic resource

M. Teresa Martín-Valdivia*, Manuel García-Vega, L. Alfonso Ureña-López

Departamento de Informática, Jaén University, Campus Las Lagunillas, S/N E-23071, Jaén, Spain

Received 22 October 2001; accepted 11 May 2002

Abstract

Neural learning has been used with effectiveness in natural language processing tasks. Particularly, the Widrow–Hoff and the Kivinen–Warmuth exponentiated gradient (based on neural learning rules) algorithms have been used in text categorization, improving the results obtained by the well-known Rocchio's algorithm. The high performance of competitive learning algorithms, recently applied to solve information retrieval problems, leads us to use them in the specific text categorization tasks. This paper presents a multilingual categorization system based on neural learning, using the polyglot Bible as training collection, both in Spanish and English. The method we suggest is based on using the LVQ algorithm to build a classifier that learns the training multilingual collection. We have performed experiments with the four algorithm which show that the ideas we describe are promising and are worth further investigation.
© 2002 Elsevier B.V. All rights reserved.

Keywords: Text categorization (TC); Vector space model (VSM); Learning vector quantization (LVQ); Multilingual information retrieval (MIR); Polyglot Bible

1. Introduction

Nowadays, about 90% of companies' information is found in text format [4]. We can find text in documents, manuals, reports, circulars, e-mails and also in Web pages. It is estimated that the amount of text available in Web pages is about a terabyte [1]. Therefore, it is essential to manage automatically this enormous amount of information by using natural language processing (NLP) techniques.

* Corresponding author.
E-mail address: maite@ujaen.es (M.T. Martín-Valdivia).

One of the main tasks of NLP is the automatic analysis of content that can be defined as a group of techniques to examine the information objects and provide subsequent access to them [34]. Text categorization is a very interesting task for NLP, which consists of the assignment of one or more pre-existing categories to each document [17]. Text categorization has been the focus of much research in recent years and the most categorization systems use collections of training documents to predict the categories of new documents [35,26,36]. With the increasing availability of electronically accessible information, the use of linguistic resources in text categorization is receiving more and more attention. These resources include training corpora and lexical databases. Training corpora, such as Reuters-21,578 [17], Ohsumed [9], TREC [30] are document collection manually labelled. On the other hand, lexical databases, such as WordNet [22], EuroWorNet [32], EDR [37], are repositories that accumulate information about the lexical items of one or several languages and have been integrated successfully in text categorization [31,7].

One of the most widely used algorithms in text categorization is the Rocchio algorithm [25]. It is a very simple algorithm and good results have been obtained [29,7]. However, a recent work of Lewis et al. [18] shows that two machine learning algorithms (Widrow–Hoff (WH) [33] and Kivinen–Warmuth (KW) [13]) are more effective than the widely used Rocchio algorithm in several categorization and routing tasks.

In contrast, the algorithms of competitive learning, based on the Kohonen model, have been used successfully in tasks of NLP [21,16]. The Kohonen model [14] presents two variants: *Self-Organizing Map* or SOM and *Learning Vector Quantization* or LVQ. Although both of them use competitive learning, the main difference lies in that SOM uses a non-supervised learning method, while in LVQ, it is supervised.

Although the versatility of this type of neural network is very wide, which allows us to classify all types of information from literary [10] to economic [12], the Kohonen model has two limitations. Firstly, the learning process is long and hard, and secondly, it is necessary to repeat the whole learning process to learn new data.

This paper proposes the use of a competitive learning algorithm to train a collection of documents for text categorization. Since it is supervised learning, we have chosen the Kohonen LVQ algorithm. In order to test the effectiveness of LVQ algorithm, we have developed some experiments with a resource, widely spread, freely available and translated into all languages: the Bible. We have compared Rocchio, WH, KW and LVQ algorithms. The results obtained show that algorithms based on neural learning are more effective than Rocchio's algorithm, and that the LVQ algorithm, which uses a competitive learning rule, is the one which obtains the best accuracy.

Research on information retrieval (IR) is developed by several models such as the vector space model (VSM), the probabilistic model and the boolean model. In this paper, we will use the VSM, which is considered an effective model in the IR field [28].

The rest of the paper is organized as follows. Firstly, the TC task is presented, with a brief introductory summary to VSM. Secondly, the polyglot Bible is described as a multilingual linguistic resource for TC. Then, the four algorithms used in the experiments are described. In Section 5, the experiments carried out to evaluate the algorithm's performance are detailed. The results of this evaluation are described in Section 6, closing with the conclusions and future lines of work.

2. Text categorization with VSM

VSM was originally developed for IR, although it can be used in other TC tasks [3]. The aim of VSM for IR is representing a natural language expression as a weight vector of terms, where each weight measures the importance of the term in the natural language expression, which can be either a document or a query. The semantic proximity among documents and queries is calculated with the cosine of the angle formed by their vectors.

In an analogous way, we can consider in TC that a document belongs to a particular category if the similarity between the document and the category is higher than a given step or it is simply the best evaluated.

So, having three groups of N terms, M documents and L queries, the weight vector for document j is $(w_{1j}, w_{2j}, \ldots, w_{Nj})$ and the weight vector for query k is $(q_{1k}, q_{2k}, \ldots, q_{Nk})$. The similarity between document j and query k is obtained by the formula

$$sim(w_j, q_k) = \frac{\sum_{i=1}^{m} w_{ij} \cdot q_{ik}}{\sqrt{\sum_{i=1}^{m} w_{ij}^2 \cdot \sum_{i=1}^{m} q_{ik}^2}} \tag{1}$$

The term weights for the document vectors are calculated by making use of the well-known formula based on term frequency [27]:

$$w_{ij} = tf_{ij} \log_2 \left(\frac{M}{df_i} \right), \tag{2}$$

where tf_{ij} (term frequency) is the number of occurrences of term i in document j, and df_i (document frequency) is the number of documents in the collection in which term i occurs.

3. Parallel corpora as multilingual resource for text categorization

Parallel corpora are a multilingual resource, more and more available in Internet [8], which are being used in NLP for different tasks such as disambiguation [5], IR [23], automatic translation [20], etc.

The Polyglot Bible [24] is one of these resources, with some specially attractive characteristics: freely available, translated into all languages, translations are practically perfect and it is divided into verses.

We propose in this paper a text classifier in Spanish and English, based on a multilingual parallel corpus. For that, we have used two translations of the Bible, *Reina Valera* edition for Spanish, and *American Standard Version* for English. So, we have created a bilingual Bible, mixing these versions verse by verse (Fig. 1).

Considering the verse as semantic unit, this Bible contains very high quality information for both languages, as in a same verse we can find the translation in both languages without any mistake. This new document is the basis for the construction of a multilingual classifier, in our case, bilingual, so a query can be made with docu-

```
010 1 1 In the beginning God created the heavens and the earth. EN
el principio crió Dios los cielos y la tierra.
010 1 2 And the earth was waste and void; and darkness was upon the
face of the deep:´ and the Spirit of God moved upon the face of the
waters Y la tierra estaba desordenada y vacía, y las tinieblas
estaban sobre la haz del abismo, y el Espíritu de Dios se movía
sobre la haz de las aguas.
010 1 3 And God said, Let there be light: and there was light. Y
dijo Dios: Sea la luz: y fué la luz.
010 1 4 And God saw the light, that it was good: and God divided
the light from the darkness  Y vió Dios que la luz era buena  y
```

Fig. 1. Bilingual Bible generated.

ments either in Spanish or in English to obtain the appropriate category, regardless of the language used.

In our experiments, we have divided the bilingual Bible into two parts (75% and 25%), the first part being used as training and the second one for the evaluation of our classifier. In addition, we have proposed 66 different categories, coinciding with the 66 books in the Bible. As each book of the Bible is divided into chapters and these into verses, enough examples of each category appear in order to justify the use of an appropriate training algorithm.

Although we have used the polyglot Bible, this is easily extrapolated to any other parallel corpus such as the Hansard Canadian Parliamentary collection [8] to build a multilingual information system.

4. Training algorithms for text categorization

Although there are many algorithms used in text classification [7,19], we have selected the Rocchio, WH and KW algorithms because they have been used in many other studies [3,31,18] and therefore we can contrast its efficiency with the LVQ algorithm.

4.1. The Rocchio algorithm

The Rocchio algorithm generates a new weight vector **w** from an existing one $\mathbf{w}_0$, and a collection of training documents. The *i*th component of vector **w** is calculated with the formula:

$$w_i = \alpha w_{0,i} + \beta \frac{\sum_{j \in C_k} x_{j,i}}{n_{C_k}} + \gamma \frac{\sum_{j \notin C_k} x_{j,i}}{n - n_{C_k}}, \tag{3}$$

where C_k is the number of documents that constitutes the *k*th category, $w_{0,i}$ is the initial weight of the *i*th term of category k, $x_{j,i}$ is the weight of the *i*th term in document j, and n_{C_k} is the number of documents labelled with the *k*th category. Parameters α, β and γ control the relative impact of the original weight vector, the positive examples and the negative examples, respectively.

As in [18], we have used values $\alpha = 1$, $\beta = 16$ and $\gamma = 4$. We limit the categorizer in order not to make use of negative weights, so in the end, the weight w_{ki} will be positive or will return to 0 if it is negative.

Initial vector w_0 is frequently taken as null vector, but this can be initialized with a set of initial weights calculated with an external resource.

4.2. The Widrow–Hoff algorithm

The WH algorithm starts with a group of existing weight vectors $\mathbf{w}_k(t)$, with $k \in [1, 66]$, which will be updated with the processing of all training documents. The weight of jth component of $\mathbf{w}_k(t+1)$ vector for a given category, k, is obtained from the ith training document and the present weight vector, as the formula:

$$w_{kj}(t+1) = w_{kj}(t) - 2\eta[\mathbf{w}_k(t) \cdot \mathbf{x}_i - y_i]x_{ij}, \tag{4}$$

where $\mathbf{x}_i$ is the ith training document, w_{kj} is the present weight of jth term of k category, y_i is 1 if ith document is assigned to the correct category and 0 if not. Constant η is the learning rate, which controls how quickly the weight vector is allowed to change and how much every new model influences it. A usual value used for n is $\frac{1}{4}X^2$, being X the maximum norm of vectors that represent the training documents.

4.3. The Kivinen–Warmuth algorithm

The exponentiated gradient algorithm was introduced by Kivinen–Warmuth and it is similar to WH, as it maintains a weight vector $\mathbf{w}_k$ and introduces the training vectors one at a time. However, with the KW algorithm, $\mathbf{w}_k$ vector components cannot be negative. The rule with which $\mathbf{w}_k$ weights are updated, similar to the WH algorithm, is

$$w_{kj}(t+1) = \frac{w_{kj}(t)\exp(-2\eta(\mathbf{w}_k \cdot \mathbf{x}_i - y_i)x_{i,j})}{\sum_{j=1}^{N} w_{kj}(t)\exp(-2\eta(\mathbf{w}_k \cdot \mathbf{x}_i - y_i)x_{i,j})}. \tag{5}$$

As we have seen before, the learning rate $\eta > 0$ controls the impact of each new training example.

4.4. The LVQ algorithm

The LVQ algorithm is notable for its heuristics simplicity and it directly adapts to the TC task. The LVQ algorithm is a classification method based on neural competitive learning, which permits the definition of a group of categories on the input data space using reinforced learning, either positive (prize) or negative (punishment).

Given a sequence of input documents, an initial group of reference vectors $\mathbf{w}_k$ is selected. In each iteration, a $\mathbf{x}_i$ document is selected and the vectors $\mathbf{w}$ are updated, so that it will adapt better to $\mathbf{x}_i$. The LVQ algorithm works as follows.

For each class, k, a weight vector $\mathbf{w}_k$ is associated. In each repetition, the algorithm selects a $\mathbf{x}_i$ input document and compares it with each weight vectors $\mathbf{w}_k$, using the euclidean distance $\|\mathbf{x}_i - \mathbf{w}_k\|$, so that the winner will be the weight vector $\mathbf{w}_c$ nearest to $\mathbf{x}_i$, with c as the index of that weight vector:

$$\|\mathbf{x}_i - \mathbf{w}_c\| = \min_k\{\|\mathbf{x}_i - \mathbf{w}_k\|\}. \tag{6}$$

The classes compete among themselves in order to find which is most similar to the input vector, so that the winner will be that one with less euclidean distance with respect to the input document. Only the winner class will modify its weights using a reinforced learning algorithm, either positive or negative, depending on whether the classification is correct or not. Thus, if the winner class belongs to the same class as the input vector (the classification has been correct), it will increase the weights, moving slightly closer 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, moving slightly further from the input vector (punishment).

Let $\mathbf{x}_i(t)$ be an input document at time t, and $\mathbf{w}_k(t)$ the weight vector for the class k at time t. The following equations define the basic learning process for the LVQ algorithm:

$$\mathbf{w}_c(t+1) = \mathbf{w}_c(t) + s \cdot \alpha(t)[\mathbf{x}_i(t) - \mathbf{w}_c(t)], \tag{7}$$

where $s = 0$, if $k \neq c$; $s = 1$, if $\mathbf{x}_i(t)$ and $\mathbf{w}_c(t)$ 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 monotically decreasing function of time. It is recommended that $\alpha(t)$ should initially be rather small, say, smaller than 0.1 [14], and $\alpha(t)$ continues decreasing to a given threshold, u, very close to 0. In our experiments $\alpha(0)$ has initialized in 0.005, decreasing linearly to $u = 0.001$, according to the following equation:

$$\alpha(t+1) = \alpha(t) - \frac{\alpha(0) - u}{K}, \tag{8}$$

where K is the number of classes.

5. Experiments

5.1. Corpus generation

In order to compare the four algorithms, we have used a group of files generated from the bilingual Bible. As the Bible is divided into books, chapters and verses, we have created a division which we have adjusted to our experiments. We have generated a total of 1189 training documents and 1189 evaluation documents, each of them belonging to one of the 66 possible classes, that is, each Bible book defines a different class.

For each book and for each chapter, we have created two files: one file for training and one file for evaluation. The generation of each file is carried out considering units of four verses, three for the training file and one for the evaluation one. Each document is of a given class, corresponding to the book it belongs to.

In order to minimize the vector space dimension, we have removed stop words, running the SMART[1] stoplist and we have extracted the word stems using the SMART stemming algorithms [2,6]. After pre-processing all documents in the collection, a total of 22,054 words, which constitute the application domain, are obtained.

5.2. Initializations

The input vectors **x** are generated applying VSM to 1189 training files, taken as a unique corpus. The query vectors **q** for the evaluation are obtained applying the formula based on term frequency [27] as it is shown in Section 2.

In order to initialize the weight vectors **w**, different methods can be used: set to zero (as in the Rocchio algorithm), set randomly, or taking a given specific value. Specifically, for the LVQ algorithm, the initial weight vector can include certain knowledge of the problem to be solved, thus improving the results [15]. Therefore, we have decided to initialize the training weight vectors of the three algorithms based on neural learning, with this additional information, mixing all the training files of the same class j in a single file. So, 66 files are obtained to generate the initial weights of training vectors, applying VSM to them. Each weight vector $\mathbf{w}_j$ has the same dimension as input vectors **x**, with as many weight vectors as categories, so the weight vector $\mathbf{w}_1$ corresponds to class 1, vector $\mathbf{w}_2$ to category 2, and so on until vector $\mathbf{w}_{66}$ that represents class 66.

5.3. Training and evaluation

The training with the Rocchio algorithm is carried out applying the equation in Section 4.1 to each category.

For the algorithms based on neural learning, all input vectors **x** are processed during the training, as many times as categories exist, in our case 66 times $(t = 1,\dots,K)$.

After the training, the categorization is carried out calculating the similarity of each **q** evaluation vector with the weight vectors **w**, selecting as solution the category with highest similarity.

These vectors **q** are of three different types. Firstly, we have selected the files generated as evaluation in the process of bilingual corpus generation, obtaining the first evaluation group, labelled as *Spanish–English*. Secondly, we have processed these vectors **q**, dividing the words of each language into two files. In this way, we obtain two groups of evaluation documents, labelled as Spanish and English, respectively.

[1] SMART is one of the most well-known Information Retrieval experimental systems of public domain. It is based in the vector model and has a module for the evaluation effectiveness using test collections. It can be downloaded through the ftp in Cornell University (ftp://ftp.cs.cornell.edu/pub/smart).

Table 1
Precision obtained with the four algorithms

	Spanish		English		Spanish–English	
	Microavg	Macroavg	Microavg	Macroavg	Microavg	Macroavg
Rocchio	0.56	0.63	0.61	0.76	0.58	0.62
WH	0.64	0.59	0.71	0.78	0.67	0.61
KW	0.66	0.61	0.73	0.77	0.70	0.61
LVQ	0.74	0.66	0.77	0.80	0.75	0.66

6. Results

In order to evaluate the effectiveness of algorithms, two widely used measurements in TC have been used: microaveraging precision and macroaveraging precision [29,19]. Microaveraging precision is defined as follows:

$$P_{\text{microavg}} = \frac{tdc}{tdc + tdi}, \tag{9}$$

where *tdc* is the number of documents correctly classified and *tdi* is the number of documents wrongly classified. Macroaveraging precision is calculated as the average of microaveraging precisions of all categories:

$$P_{\text{macroavg}} = \frac{\sum P_{\text{microavg}}}{K}. \tag{10}$$

Macroaveraging gives equal weight to each category, whereas microaveraging gives equal weight to each document in the collection.

Table 1 shows the results obtained after the training of each algorithm for all evaluation files. As can be seen, the LVQ algorithm is better than the others. In addition, the three algorithms based on a neural approach obtain better results than the widely used Rocchio algorithm. Neural learning algorithms work better because they carry out repetitive training. Furthermore, the LVQ algorithm gives a prize or punishment depending on its behaviour in training time. It is supervised learning which directs the weights modification according to the system behaviour at each moment, making classes compete among themselves.

The results of bilingual queries average the values obtained by means of monolingual queries experiments. This was as expected, as both languages are almost disjoints.

The best microaveraging is obtained when the evaluation is made with queries only in English; the worst appears with queries only in Spanish. We interpret this fact as the choice of a Spanish stemmer less appropriate than that for English. Efficient stemming algorithms for English have been developed over the past 25 years, but most of this work will not generalize to other languages [11]. On the other hand, the differences seen in the results obtained with the LVQ algorithm for the three query types are not very different (Spanish 0.74, Spanish–English 0.75 and English 0.77) which shows how the

Table 2
The improvement percentage of the LVQ algorithm regarding the other three algorithms

	Rocchio	WH	KW
	%	%	%
S	40.00	26.24	21.80
E	43.76	21.43	14.33
S–E	40.68	23.71	18.01
Average	41.48	23.79	18.05

stemmer choice affects the LVQ algorithm less than the others. These results deserve further investigation in our future work. We must therefore improve our utilization of the linguistic resources.

Table 2 shows the improvement percentage of the LVQ algorithm with respect to the other evaluated algorithms, meaning as M improvement percentage that

$$M = \frac{E - E_{\mathrm{LVQ}}}{E} \times 100, \tag{11}$$

where E_{LVQ} is the number of errors obtained with LVQ algorithm and E is the number of errors found in the other algorithms.

Figs. 2–4 show the confusion matrices for the three groups of LVQ queries: Spanish queries, English queries and bilingual queries, respectively. Each row shows the classification obtained for all documents in one category, i.e., row 1 shows the results obtained for all documents in the category 1. Although we have only included the values greater than zero, we can see the great precision of the LVQ algorithm in focusing the diagonal. Perfect behaviour would show a diagonal matrix.

7. Conclusions and future works

This paper has presented a multilingual categorization system, using a neural learning competitive algorithm.

A direct evaluation of our categorization method based on competitive learning has been carried out, obtaining very important results and better performance than those obtained by other algorithms successfully used in categorization. In fact, our method improves about 27.77% in comparison with others.

We have also displayed an automatic evaluation method of categorization that allows us to compare the effectiveness of different approaches.

The study of the LVQ algorithm in other NLP tasks (disambiguation) and the application of SOM model for other tasks that require non-supervised learning (automatic generation of summaries) will focus our future work. Also, we plan to perform more experiments with others linguistic resources (such as Reuters) in order to check the potential of our method.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66
1	37	1	2				1	1				2					1								2																																					1	2
2		34		1	4																		1																																												
3			26																							1																																							
4			5	26		4																			1																																									
5	1			1	28	2													1				1																																										
6					2	21								1																																																					
7						2	18							1																																																			
8								4																																																											
9			1		1	1			27	1																																																							
10									3	16			3						1					1																																									
11		1										18	3																																																						
12											1	20		2											2																																									
13	1								2	4			20	2																																																		
14		1									7	6		20											1		1																																							
15													1	2	4	1			1								1																																							
16				1	1										1	1	9																																																			
17																	10																																																
18																		32		2	1		5		1																						1																			
19	4	1		1	4			1				1						2	104	4	2	1	7	1	2	1		1					2	1		1		4	1																								2		2
20																					29	1				1																																								
21																					2	10																																												
22											1												7																																											
23	1	1	1										4		1					1	2			42	2		2		1				1	1			1		2	1	1																									1	
24																									3	47	1												1																														
25																			1					1	3																																									
26			2																					2	3		39	1							1																															
27																								1			1	9																																						1	
28												1								1	1	2						9																																					
29																									1					2																																					
30																									3	1						5																																				
31																																1																																		
32																			1					1								2																																	
33																			1					1		1			2				1				1																											
34																			1														2																															
35																			2															1																														
36																								2	1																																									
37																																					2																												
38						1																		1					1									1	10																											
39																																						1	3																											
40														1																										18	4	4	1																								
41																																								7	7	1	1																								
42																																								4	5	15																									
43																																											21																						
44	1								2																																2			23																					
45																				1																									9	1		1	1				1	1						1					
46																																													1	13	1			1															
47		1																																											1		9						1									1				
48																																												1			1	4																			
49																																																	1	4		1															
50																																																		3				1													
51																																																	1		2					1									
52																																																		2		2	1													
53																																																				1	2													
54																																																	1					1	3	1											
55																																												1				1			1			1											
56																																												1		1						1													
57																																																				1														
58							1																																														1					10								1
59																																														1														4								
60																																															1														4							
61																																																								1				1					1	
62																																																														5				
63																																																															1			
64																																																																1		
65																																																	1																		
66																																																			1																21

Fig. 2. LVQ confusion matrix for Spanish queries.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66
1	35	2	1		4			1					1															1			2																																		1	2
2		32	1	1	5																		1																																											
3			23	2	1																					1																																								
4		1	5	26		4																																																												
5		1		1	28	2								1									1																																											
6					1	22																		1																																										
7						2	18							1																																																				
8								4																																																										
9			1		1	1			26	1																				1																																				
10					1				2	16			4						1																																															
11		1								1	17	3																																																						
12												20		3										2																																										
13	1								4	3			16	4					1																																															
14		2			1						5	4		23	1																																																			
15													1		5	2							1														1																													
16														1	2	9																																1																		
17																	10																																																	
18					2													33	1		1				3				1						1																															
19	2		3		2			1				1							105	7	2	1	6		1	3			1				3	1	1	1		1									1							1			1		2							3
20																		1		29			1																																											
21																				2	10																																													
22											1											6				1																																								
23	1	1	1		5							3		1					3	1			35	4		2		2	1							3		2																								1				
24																			1					48						1					1			1																												
25																			1				1		3																																									
26			4																				1	2		38	1							1		1																														
27																				1						1	9							1																																
28					1							1											1					10		1																																				
29																							1						2																																					
30																			1				2	2						3			1																																	
31																															1																																			
32																			2													2																																		
33					1														1							3			2																																					
34																			1															2																																
35					1														1				1																																											
36					1									1																						1																														
37																																					2																													
38						1																	2			1		1									1	8																												
39																								1														1	2																											
40								1																																18	6		2	1																						
41																																								4	11		1																							
42																																								3	3	18																								
43																																											19																			2				
44		1																																										27																						
45																																													16																					
46																																														16																				
47																																															13																			
48																																																6																		
49																																																	6																	
50																																																		4																
51																																																			4															
52																																																				5														
53																																																					3													
54																																																						6												
55																																																							4											
56																																																								3										
57																																																									1									
58																																																										13								
59																																																											5							
60																																																												5						
61																																																													3					
62																																																														5				
63																																																															1			
64																																																																1		
65																																																																	1	
66																																											1																							21

Fig. 3. LVQ confusion matrix for English queries.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66
1	37	1	1		3			1										1								1					2																																		1	2
2		33		1	5																	1																																												
3			24	1	1																					1																																								
4			5	27		4																																																												
5		1		1	28	2								1									1																																										
6					2	22																																																												
7						2	18							1																																																				
8								4																																																										
9			1		1	1			27	1																																																							
10					1				2	16			4						1																																															
11		1									18	3																																																				
12												21		2										2																																										
13	1								4	3			16	4					1																																														
14		2									7	4		22	1																																																			
15													1	1	5	2							1																																										
16														1	1	11																																																		
17																	10																																																		
18					1														37	1		2				1																																								
19	3		1		2			1				1						1	114	8	2	1	4	1		2		1	1				2					3	1																												1
20																				31																																															
21																				2	10																																														
22											1										7																																															
23	1	1	1		2						4		1					2	1			41	3				2		1						2		1	1														1												1	
24																			1				2	48						1																																				
25																			1				1		3																																										
26			3																				1	2		39	1							1		1																														
27																				1						2	9																																							
28					1							1									1						11																																						
29																							1						2																																				
30																							2	1						5			1																																		
31																															1																																				
32																			2													2																																		
33					1														1					1		2			1										1																													
34																			1														2																																	
35																			1				1													1																														
36														1					1																		1																														
37																																					2																													
38						1																	1					1										1	10																												
39																								1																3																										
40								1																																	16	3	2	5	1																								
41																																									8	4	1	3																										
42																																									6	5	13																												
43																																													20																						1				
44	1								2																																1	2	1	1	20																								
45	1																																															12				1				1	1														
46																																																		13		1		2																
47		1																																														1		9							1									1				
48																																																						6																			
49																																																						1	5																			
50																																																								3						1												
51																																																						1	2									1										
52																																																								2		2	1																
53																																																												3															
54																																																													4	2														
55																																																													1	2	1													
56																																															1									2																
57																																																										1																
58							1																																																				1				10									1
59																																															1																4										
60																																															1									1						3									
61																																																										1												1				1	
62																																																																							5				
63																																																									1																	
64																																																		1																					
65																																																																						1					
66																																													1								1															20

Fig. 4. LVQ confusion matrix for Spanish–English queries.

Acknowledgements

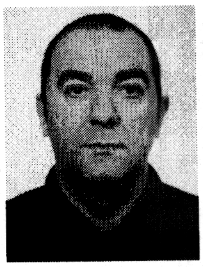
The authors wish to thank the anonymous reviewers for their helpful comments on an earlier version of this paper.

References

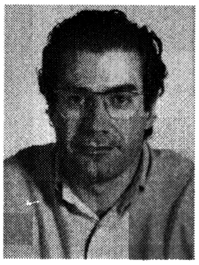
[1] R. Baeza-Yates, B. Ribeiro-Neto, Modern Information Retrieval, ACM Press Books, New York, 1999.
[2] C. Buckley, Implementation of the smart information retrieval system, Technical Report 85-686, Cornell University, 1985.
[3] M. Buenaga, J.M.B. Gómez, Díaz, Using WordNet to complement training information in text categorization, in: Proceedings of Second International Conference on Recent Advances in Natural Language Processing (RANLP), Amsterdam, Netherlands, 1997.
[4] Corp. Oracle., Managing text with Oracle8(tm) context cartridge, in: An Oracle Technical White Paper, 1997.
[5] M.W. Davis, On the effective use of large parallel corpora in cross language text retrieval, in: G. Grefenstette (Ed.), Cross-Lingual Information Retrieval, Kluwer Academic Publisher, Dordrecht, 1998.
[6] W. Frakes, R. Baeza, Information Retrieval: Data Structures and Algorithms, Prentice–Hall, Englewood Cliffs, NJ, 1992.
[7] J.M. Gómez, M. de Buenaga, L.A. Ureña, M.T. Martín, M. García, Integrating Lexical knowledge in learning based text categorization, in: Proceedings of the Sixth International Conference on the Statistical Analysis of Textual Data, France, 2002.
[8] G. Grefenstette, Cross-Lingual Information Retrieval, Kluwer Academic Publisher, Dordrecht, 1998.
[9] W. Hersh, C. Buckley, T.J. Leone, D. Hickman, Oshumed: an interactive retrieval evaluation a new large text collection for research, in: Proceedings of ACM SIGIR, Dublin, Ireland, 1994.
[10] T. Honkela, V. Pulkki, T. Kohonen, Contextual relations of words in Grimm tales, analysed by self-organizing map, in: Proceedings of International Conference on Artificial Neural Networks, ICANN-95, Paris, 1995.
[11] D. Hull, G. Grefenstette, Experiments in multilingual information retrieval, in: Proceedings of ACM, SIGIR'96, Zurich, Switzerland, 1996.
[12] S. Kaski, T. Kohonen, Exploratory data analysis by the self-organizing map: structures of welfare and poverty in the world, in: REFENES (Eds.), Neural Networks in Financial Engineering: Proceedings of the Third International Conference on Neural Networks in the Capital Markets, London, 1995, pp. 498–507.
[13] J. Kivinen, M.K. Warmuth, Exponentiated gradient versus gradient descent for linear predictors, Inform. Computat. 132 (1997) 1–63.
[14] T. Kohonen, Self-Organization and Associative Memory, 2nd Edition, Springer, Berlin, 1995.
[15] T. Kohonen, The self-organizing map, Neurocomputing 21 (1998) 1–6.
[16] T. Kohonen, S. Kaski, et al., Self organization of massive document collection, IEEE Trans. Neural Networks 11 (3) (2000) 574–585.
[17] D.D. Lewis, Representation and learning in information retrieval, Ph.D. Thesis, Department of Computer and Information Science, University of Massachusetts, 1992.
[18] D.D. Lewis, R.E. Schapire, J.P. Callan, R. Papka, Training algorithms for linear text classifiers, in: SIGIR'96: Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Zurich, Switzerland, 1996.
[19] C. Manning, H. Schütze, Foundations of Statistical Natural Language Processing, MIT Press, Cambridge, MA, 2000.
[20] J.S. McCarley, S. Roukos, Fast documents translations for cross language information retrieval, in: Proceedings of AMTA98, Springer, Berlin, 1998.
[21] D. Merkl, Text classification with self-organizing maps: some lessons learned, Neurocomputing 21 (1998) 61–77.
[22] G.A. Miller, WordNet: a Lexical database for English, Commun. ACM 38 (1995) 39–41.

[23] J.Y. Nie, P. Isabelle, M. Simard, R. Durand, Cross-language information retrieval based on parallel texts and automatic mining of parallel texts from the web, ACM-SIGIR Conference, Berkeley, CA, 1999, pp. 74–81.
[24] P. Resnik, M.D. Olsen, M. Diab, The Bible as parallel corpus: annotating the "Book of 2000 Tongues", Comput. Humanities 33 (1–2) (1999) 129–153.
[25] J.J. Rocchio Jr., Relevance feedback in information retrieval, in: G. Salton (Ed.), The SMART Retrieval System: Experiments in Automatic Document Processing, Prentice-Hall, Englewood Cliffs, NJ, 1971.
[26] M. Sahami, Editor of Proceedings of the AAAI'98/ICML'98 Workshop on Learning for Text Categorization, Wisconsin, 1998.
[27] G. Salton, Automating Text Processing: the Transformation, Analysis and Retrieval of Information by Computer, Addison-Wesley, Reading, MA, 1989.
[28] G. Salton, M.J. McGill, Introduction to Modern Information Retrieval, McGraw-Hill, New York, 1983.
[29] F. Sebastiani, A tutorial on automated text categorisation, in: Amandi, Zunino (Eds.), Proceedings of ASAI-99, First Argentinian Symposium of Artificial Intelligence, Buenos Aires, Argentina, 1999.
[30] K. Sparck-Jones, Reflections on TREC, Inform. Process. Management 31 (1995).
[31] L.A. Ureña, M. de Buenaga, M. García, J.M. Gómez, Integrating and evaluating WSD in the adaptation of a Lexical database in text categorization, in: Proceedings of the First Workshop on Text, Speech, Dialogue, Brno, Czech Republic, 1998.
[32] P. Vossen, EuroWorNet: a multilingual database for information retrieval, in: Proceedings of the DELOS Workshop on CLIR, Budapest, Hungary, 1997.
[33] B. Widrow, S. Sterns, Adaptative Signal Processing, Prentice-Hall, Englewood Cliffs, NJ, 1985.
[34] R. Wilensky, The UC Berkeley digital library project: re-thinking scholarly information dissemmination and use, European Conference on Research and Advances Technology for Digital Libraries (ECDL'99), 1999.
[35] Y. Yang, An evaluation of statistical approaches to text categorization, Inform. Retrieval J. 1 (1999).
[36] Y. Yang, X. Liu, A re-examination of text categorization methods, in: Proceedings of ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR), 1999.
[37] T. Yokoi, The EDR electronic dictionary, Commun. ACM 38 (1995) 42–44.

M. Teresa Martín-Valdivia is Assistant Professor in the Department of Computer Science at Jaén University (Spain). He received M.S. degree in Computer Science from the University of Granada in 1992. Her research interests include the application of neural networks, information retrieval and text categorization.



Manuel García Vega is Assistant Professor in the Department of Computer Science at Jaén University (Spain). He received M.S. degree in Computer Science from the University of Granada in 1991. His current research includes Word Sense Disambiguation, Text Categorization, Information Retrieval and Management Systems of Natural Language Processing.



L. Alfonso Ureña López is Assistant Professor in the Department of Computer Science at Jaén University (Spain). He received M.S. degree in Computer Science from the University of Granada in 1991, and Ph.D. in Computer Science from Software Engineering Department of Granada University in 2000. His Ph.D. Thesis won the winner of the 2001 Awards of the Spanish Society for Natural Language Processing. His current research includes Word Sense Disambiguation, Information Retrieval, Text Categorization and Management Systems of Natural Language Processing and Human Computer Interaction. Dr. Ureña is Director of the Computer Science Department at University of Jaén and Director of Research Group of Intelligent Systems. He is author or co-author of more than 40 scientific publications. He serves as a technical reviewer in the for several journals and in the Program Committee of some major conferences.