SENSEVAL-3: Third International Workshop on the Evaluation of Systems
for the Semantic Analysis of Text, Barcelona, Spain, July 2004
Association for Computational Linguistics

The University of Jaén Word Sense Disambiguation System*

Manuel García-Vega
Universidad de Jaén
Av. Madrid 35
Jaén, Spain, 23071
mgarcia@ujaen.es

Miguel A. García-Cumbreras
Universidad de Jaén
Jaén, Spain, 23071
Av. Madrid 35, 23071
magc@ujaen.es

M. Teresa Martín-Valdivia
Universidad de Jaén
Av. Madrid 35
Jaén, Spain, 23071
maite@ujaen.es

L. Alfonso Ureña-López
Universidad de Jaén
Av. Madrid 35
Jaén, Spain, 23071
laurena@ujaen.es

Abstract

This paper describes the architecture and results of the University of Jaén system presented at the SENSEVAL-3 for the English-lexical-sample and English-All-Words tasks. The system is based on a neural network approach. We have used the Learning Vector Quantization, which is a supervised learning algorithm based on the Kohonen neural model.

1 Introduction

Our system for SENSEVAL-3 uses a supervised learning algorithm for word sense disambiguation. The method suggested trains a neural network using the Learning Vector Quantization (LVQ) algorithm, integrating several semantic relations of WordNet (Fellbaum, 1998) and SemCor corpus (Miller et al., 1993). The University of Jaén system has been used in English-lexical-sample and English-All-Words tasks.

2 Experimental Environment

The presented disambiguator uses the Vector Space Model (VSM) as an information representation model. Each sense of a word is represented as a vector in an *n*-dimensional space where *n* is the number of words in all its contexts.

The accuracy of the disambiguator depends essentially on the word weights. We use the LVQ algorithm to adjust them. The input vector weights are calculated as shown by (Salton and McGill, 1983) with the standard *tf·idf*, where the documents are the paragraphs. They are presented to the LVQ network and, after training, the output vectors (called prototype or codebook vectors) are obtained, containing the adjusted weights for all senses of each word.

Any word to disambiguate is represented with a vector in the same way. This representation must be compared with all the trained word sense vectors by applying the cosine similarity rule:

$$sim(\mathbf{w}_k, \mathbf{x}_i) = \frac{\mathbf{w}_k \cdot \mathbf{x}_i}{|\mathbf{w}_k| \cdot |\mathbf{x}_i|} \quad [1]$$

The sense corresponding to the vector of highest similarity is selected as the disambiguated sense.

To train the neural network we have integrated semantic information from two linguistic resources: SemCor corpus and WordNet lexical database.

2.1 SemCor

Firstly, the SemCor (the Brown Corpus labeled with the WordNet senses) was fully used (the Brown-1, Brown-2 and Brown-v partitions). We used the paragraph as a contextual semantic unit

* This paper has been partially supported by the Spanish Government (MCYT) Project number TIC2003-07158-C04-04

and each context was included in the training vector set.

The SENSEVAL-3 English tasks have used the WordNet 1.7.1 sense inventory, but the SemCor is tagged with an earlier version of WordNet (specifically WordNet version 1.6).

```
climb\2#1 158 a_hundred\5 1 absorb\2 1 advance\2 1 ... walk\2 1
want\1 1 warn\2 1 warped\5 1 way\1
2 west\3 1 whip\1 2 whir\1 1
wraithlike\5 1
climb\2#1 45 abruptly\4 1 absence\1
1 ... stop\2 1 switch_off\2 1
there\4 1 tube\1 1 two\5 1 unexpectedly\4 1 water\1 1
...
climb\2#2 33 adjust\2 1 almost\4 1
arrange\2 1 ... procedure\1 1 revetment\1 1 run\2 1 sky\1 1
snatch\2 1 spread\2 1 stand\2 1
truck\1 1 various\5 1 wait\2 1
wing\1 1
...
climb\2#3 3 average\2 1 feel\2 1
report\2 1
```

Figure 1. SemCor context for "climb".

Therefore it was necessary to update the SemCor word senses. We have used the automatically mapped version of Semcor with the WordNet 1.7.1 senses found in WordNet site[1].

```
climb\2#1 10 arise\2 1 come_up\2 1
go\2 1 go_up\2 1 lift\2 1 locomote\2
1 move\2 1 move_up\2 1 rise\2 1
travel\2 1
climb\2#1 3 climb_up\2 1 go_up\2 1
mount\2 1
climb\2#1 5 mountaineer\2 1 ramp\2 1
ride\2 1 scale\2 1 twine\2 1
climb\2#2 7 clamber\2 1 scramble\2 1
shin\2 1 shinny\2 1 skin\2 1 sputter\2 1 struggle\2 1
...
climb\2#5 2 go up\2 1 rise\2 1
```

Figure 2. WordNet artificial paragraph

[1] http://www.cogsci.princeton.edu/~wn/

Figure 1 shows the common format for the all the resource input paragraphs. For each word, the pos and sense are described, e.g. "climb\2#1" is the verb "climb" with sense 1. In addition, it has 158 different words in its context and all of them are shown like the pair word-frequency.

2.2 WordNet

'Semantic relations from WordNet 1.7.1 were considered, in particular synonymy, antonymy, hyponymy, homonymy, hyperonymy, meronymy, and coordinate terms to generate artificial paragraphs with words along each relation.

For example, for a word with 7 senses, 7 artificial paragraphs with the synonyms of the 7 senses were added, 7 more with all its hyponyms, and so on.

Figure 2 shows these artificial paragraphs for the "climb" verb.

3 Learning Vector Quantization

The LVQ algorithm (Kohonen, 1995) performs supervised learning, which uses a set of inputs with their correctly annotated outputs adjusting the model when an error is committed between the model outputs and the known outputs.

The LVQ algorithm is a classification method based on neural competitive learning, which allows the definition of a group of categories on the input data space by reinforced learning, either positive (reward) or negative (punishment). In competitive learning, the output neurons compete to become active. Only a single output neuron is active at any one time.

The general application of LVQ is to adjust the weights of labels to high dimensional input vectors, which is technically done by representing the labels as regions of the data space, associated with adjustable prototype or codebook vectors. Thus, a codebook vector, $\mathbf{w}_k$, is associated for each class, k. This is particularly useful for pattern classification problems.

The learning algorithm is very simple. First, the learning rate and the codebook vectors are initialised. Then, the following procedure is repeated for all the training input vectors until a stopping criterion is satisfied:

- Select a training input pattern, $\mathbf{x}$, with class d, and present it to the network

- Calculate the Euclidean distance between the input vector and each codebook vector $\|\mathbf{x}\text{-}\mathbf{w}_k\|$
- Select the codebook vector, $\mathbf{w}_c$, that is closest to the input vector, $\mathbf{x}$.

$$\|\mathbf{x}-\mathbf{w}_c\| = \min_k\{\|\mathbf{x}-\mathbf{w}_k\|\} \quad [2]$$

This codebook vector is the winner neuron and only this neuron updates its weights according the learning equation (equation 3). If the class of the input pattern, $\mathbf{x}$, matches the class of the winner codebook vector, $\mathbf{w}_c$ (the classification has been correct), then the codebook vector is moved closer to the pattern (reward), otherwise it is moved further away.

Let $\mathbf{x}(t)$ be a input vector at time t, and $\mathbf{w}_k(t)$ the codebook vector for the class k at time t. The following equation defines the basic learning process for the LVQ algorithm.

$$\mathbf{w}_c(t+1) = \mathbf{w}_c(t) + s\cdot\alpha(t)\cdot[\mathbf{x}(t)-\mathbf{w}_c(t)] \quad [3]$$

```
climb\2#1 1921 a\1#0 0.01883
aarseth\1#0 0.03259 abelard\1#0 ...
yorkshire\1#0 0.03950 young\3#0
0.00380 zero\1#0 0.01449
climb\2#2 235 act\1#0 -0.11558
alone\4#0 -0.07754 ... windy\3#0 -
0.00922 worker\1#0 -0.02738 year\1#0
-0.03715 zacchaeus\1#0 -0.02344
climb\2#3 1148 abchasicus\1#0
0.04127 able\3#0 -0.00945 ...
young\3#0 -0.00275 zero\1#0 -0.00010
climb\2#4 258 age\1#0 -0.04180 air-
space\1#0 -0.02862 alone\4#0 -
0.01920 apple\1#0 -0.04242 ...
world\1#0 -0.14184 year\1#0 -0.04113
young\3#0 -0.04831 zero\1#0 -0.06230
...
```

Figure 3. Codebook vectors for "climb" domain

where $s = 0$, if $k \neq c$; $s = 1$, if $\mathbf{x}(t)$ and $\mathbf{w}_c(t)$ belong to the same class ($c = d$); and $s = -1$, if they do not ($c \neq d$). $\alpha(t)$ is the learning rate, and $0<\alpha(t)<1$ is a monotically decreasing function of time. It is recommended that $\alpha(t)$ should initially be rather small, say, smaller than 0.1 (Kohonen, 1995) and $\alpha(t)$ continues decreasing to a given threshold, u, very close to 0.

The codebook vectors for the LVQ were initialized to zero and every training vector was introduced into the neural network, modifying the prototype vector weights depending on the correctness in the winner election.

All training vectors were introduced several times, updating the weights according to learning equation. $\alpha(t)$ is a monotonically decreasing function and it represents the learning rate factor, beginning with 0.1 and decreasing lineally:

$$\alpha(t+1) = \alpha(t) - \frac{\alpha(0)}{P} \quad [4]$$

where P is the number of iterations performed in the training. The number of iterations has been fixed at 25 because at this point the network is stabilized.

The LVQ must find the winner sense by calculating the Euclidean distances between the codebook vectors and input vector. The shortest distance points to the winner and its weights must be updated.

4 English Tasks

The training corpus generated from SemCor and WordNet has been used to train the neural networks. All contexts of every word to disambiguate constitute a domain. Each domain represents a word and its senses. Figure 3 shows the codebook vectors generated after training process for "climb" domain.

We have generated one network per domain and after the training process, we have as many domains as there are words to disambiguate adjusted. The network architecture per domain is shown in Figure 4. The number of input units is the number of different terms in all contexts of the given domain and the number of output units is the number of different senses.

The disambiguator system has been used in English lexical sample and English all words tasks.

For the English lexical sample task, we have used the available SENSEVAL-3 corpus to train the neural networks. We have also used the contexts generated using SemCor and WordNet for each word in SENSEVAL-3 corpus. For the Eng-

lish all word task, we have only used the complete contexts of both SemCor and WordNet resources. The corpus has been tagged and lemmatized using the Tree-tagger (Schmid, 1994).

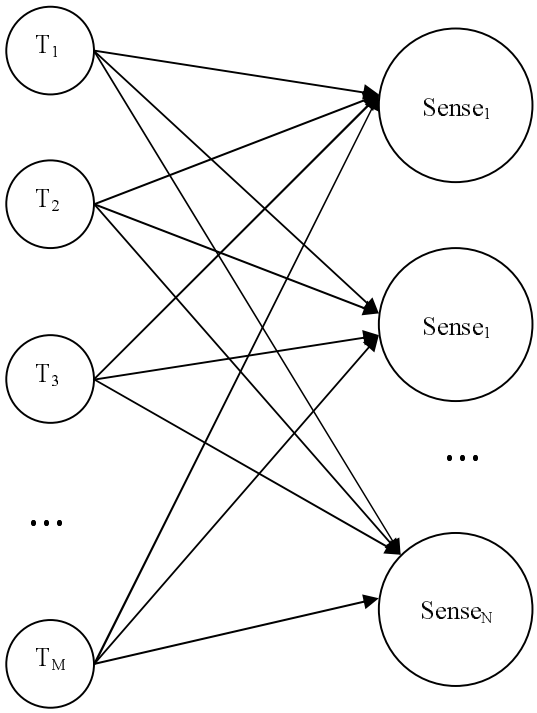

Figure 4. The network architecture

Once the training has finished, the testing begins. The test is very simple. We establish the similarity between a given vector of the corpus evaluation with all the codebook vectors of its domain, and the highest similarity value corresponds to the disambiguated sense (winner sense). If it is not possible to find a sense (it is impossible to obtain the cosine similarity value), we assign by default the most frequent sense (e.g. the first sense in WordNet).

The official results achieved by the University of Jaén system are presented in Table 1 for English lexical sample task, and in Table 2 for English all words.

ELS	Precision	Recall	Coverage
Fine-grained	0.613	0.613	99.95%
Coarse-grained	0.695	0.695	99.95%

Table 1. Official results for ELS.

EAW	Precision	Recall	Coverage
With U	0.590	0.590	100%
Without U	0.601	0.588	97.795%

Table 2. Official results for EAW.

5 Conclusion

This paper presents a new approach based on neural networks to disambiguate the word senses. We have used the LVQ algorithm to train a neural network to carry out the English lexical sample and English all words tasks. We have integrated two linguistic resources in the corpus provided by the organization: WordNet and SemCor.

References

Fellbaum, C. 1998. WordNet: An Electronic Lexical Database. The MIT Press

Kohonen, T. 1995. Self-Organization and Associative Memory. 2nd Ed, Springer.Verlag, Berlín.

Kohonen, T., J. Hynninen, J. Kangas, J. Laaksonen, K. Torkkola. 1996. Technical Report, LVQ_PAK: The Learning Vector Quantization Program Package. Helsinki University of Technology, Laboratory of Computer and Information Science, FIN-02150 Espoo, Finland.

Miller G., C. Leacock, T. Randee, R. Bunker. 1993. A Semantic Concordance. Proc. of the 3rd DARPA Workshop on Human Language Technology.

Salton, G. & McGill, M.J. 1983. Introduction to Modern Information Retrieval. McGraw-Hill, New York.

Schmid, H., 1994. Probabilistic Part-of-Speech Tagging Using Decision Trees. In Proceedings of International Conference on New Methods in Language Processing.