

# Evolving Neural Networks for Word Sense Disambiguation

A. Azzini, C. da Costa Pereira, M. Dragoni, and A. G. B. Tettamanzi  
Università degli Studi di Milano, Dipartimento di Tecnologie dell’Informazione  
Via Bramante 65, I-26013 Crema (CR), Italy  
{azzini, dragoni, pereira, tettamanzi}@dti.unimi.it

## Abstract

*We propose a supervised approach to word sense disambiguation based on neural networks combined with evolutionary algorithms. Large tagged datasets for every sense of a polysemous word are considered, and used to evolve an optimized neural network that correctly disambiguates the sense of the given word considering the context in which it occurs. The viability of the approach has been demonstrated through experiments carried out on a representative set of polysemous words.*

## 1 Introduction

The automatic disambiguation of word senses consists in assigning the most appropriate meaning to a polysemous word, i.e., a word with more than one meanings, within a given context. This consists of two steps: (i) considering the possible senses of the given word; and (ii) assigning each occurrence of the word to its *appropriate* sense.

We propose a supervised approach to word sense disambiguation based on neural networks (NNs) combined with evolutionary algorithms (EAs). We dispose of large tagged datasets describing the contexts in which every sense of a polysemous word occurs, and use them to evolve an optimized NN that correctly disambiguates the sense of a word given its context. To this aim, we use an EA to automatically design NNs, with two kinds of distributed encoding schemes, based on the way words are written, to represent the context in which a word occurs.

The paper is organized as follows: Section 2 explains the application of the evolutionary approach to the word sense disambiguation (WSD), whereas Section 3 describes the neuro-genetic algorithm used to approach this problem, followed, in Section 4, by the results of experiments. Section 5 concludes with a discussion of the results.

## 2 Word Sense Disambiguation

Their tolerance for noise, their ability to generalize, and their well-known suitability for classification tasks [1] make NNs natural candidates for approaching WSD. A large number of successful applications demonstrates that NN design is improved by considering it in conjunction with EAs, since neural and evolutionary techniques can be combined in a synergetic way [14]. The most promising approach to using EAs to design and optimize NNs is the one which jointly evolves network architecture and weights: a completely functioning network can be evolved without any intervention by an expert [1].

The algorithm presented in Section 3 evolves a specific NN specializing in disambiguating one given target polysemous word. The same process must be repeated for each polysemous word one is interested in disambiguating. Since our aim here is to demonstrate the feasibility of this approach, we will focus on a small, but representative, set of target polysemous words.

We used IXA Group’s web corpus [2], which comprises all WordNet noun senses. Its construction is inspired by the “monosemous relatives” method [10]. The method usually relies on information in WordNet in order to retrieve examples from large corpora or the web. The advantage of using this corpus instead of other traditional corpora specifically designed for WSD is that, due to the way the corpus is constructed, we get an extensive coverage of word usage in all domains. Furthermore, since the corpus is not tagged by hand, it gives wider guarantees of accuracy and objectivity.

From this web corpus we extracted all data relative to the words listed in Table 2, with their number of senses. For each of these words 75% of the records have been used for training, 12.5% for testing, and 12.5% for validation.

Two kinds of representations commonly used in connectionism are distributed [9] and localist schemes [7]. The major drawback of the latter is the high number of inputs needed to disambiguate a single sentence, because every input node has to be associated to every possible sense; in fact, this number increases staggeringly if one wants to dis-

ambiguates an entire text.

Examples of distributed schemes are microfeatures [13] and the word-space model [12]. Distributed schemes are very attractive in that they represent a context as an activation pattern over the input neurons. Therefore, unlike with localist schemes, the number of input neurons required does not have to equal the size of the vocabulary. On the contrary, there is an intrinsic idea of *compression*, whereas each input neuron encodes a given *feature* of a word or sentence and thus can be reused to represent many different words or senses. Of course, the more the input patterns are compressed into a low-dimensional space, the more information is lost. However, despite such information loss, enough information may still be there to allow meaningful processing by the NN. We used two distributed representation schemes, both based on the way words are written, corresponding to two different degrees of compression: (i) a *positional* scheme, whereby 156 input neurons, divided in six groups of 26 input neurons each, encode the first six letters of a word, after deleting as many vowels, starting from the last one but except the first one, as required to reduce the word to six letters, because consonants carry a heavier distinctive load; if, even after deleting all the vowels but the first, the word is still more than six letter long, only the first six letters are actually represented (thus *representation* would be encoded as REPRSN, but *cotton* would be encoded as COTTON); if the word is shorter than six letters, the representation is padded with blanks; and (ii) a *letter count* scheme, which brings the distributed representation idea to one extreme, by using the number of occurrences of the 26 letters of the alphabet as features.

In both cases, the activations of the input neurons are obtained by summation of the activation patterns representing the words occurring in a given context, excluding the target word, after removing stop words and stemming the remaining words. Additional fields of the training set (one for each output neuron) correspond to the  $n$  senses of the target word. They are all set to zero except the one corresponding to the correct sense. For example, starting from a sentence ‘part aqueduct system’, where the target word, *tunnel*, has two senses, (1) “a passageway through or under something” and (2) “a hole made by an animal”, the associated record would be

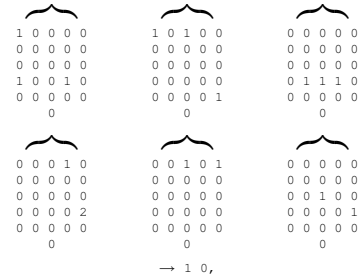
- for the letter count encoding,

2 0 1 1 2 0 0 0 0 0 0 0 1 0 0 1 1 1 2 3 2 0 0 0 1 0  
 → 1 0,

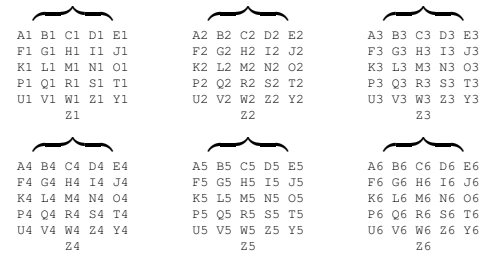
in which the first 26 numbers represent the occurrences of the letters of the alphabet, and the last 2 the two output senses (here  $n = 2$ );

- for the positional encoding, after reducing all words

to their 6-letter representatives ‘PART’, ‘ACQDCT’, and ‘SYSTEM’,



where the numbers on the left-hand side of the arrow represent the occurrences of the letters in the six positions, i.e., A1, . . . , Z6, here displayed according to the template



The rationale for this type of sentence encoding is that, since every word can be regarded as an activation pattern of the input layer of a NN, the overlap of different patterns from the same sentences allows the network to detect significant information about the context.

The success of the experiments presented in Section 4 will serve as an empirical proof that even such extremely compressed representations preserve enough context information to allow disambiguation of word sense.

### 3 The Neuro Evolutionary Algorithm

We use an evolutionary approach for NN design, previously validated on different benchmarks and real-world problems [4, 3], which uses a variable-size representation of individuals. A population of classifiers is maintained — the individuals, encoding multilayer perceptrons (MLPs), a type of feed-forward NN. The evolutionary process is based on the joint optimization of NN structure and weights, and uses the error back-propagation (BP) algorithm to decode a *genotype* into a *phenotype* NN. Accordingly, it is the genotype which undergoes the genetic operators and which reproduces itself, whereas the phenotype is used *only* for calculating the genotype’s fitness. The rationale for this choice is that the alternative of using BP, applied to the genotype, as a kind of “intelligent” mutation operator would boost exploitation while impairing exploration, thus making the algorithm too prone to being trapped in local optima.

**Table 1. Individual Representation.**

Element	Description
topology	String of integer values that represent the number of neurons in each layer.
$\mathbf{W}^{(0)}$	Weights matrix of the input layer neurons of the network.
$\mathbf{Var}^{(0)}$	Variances for every element of $\mathbf{W}^{(0)}$ .
$\mathbf{W}^{(i)}$	Weights matrix for the $i$ th layer, $i = 1, \dots, l$ .
$\mathbf{Var}^{(i)}$	Variances every element of $\mathbf{W}^{(i)}$ , $i = 1, \dots, l$ .
$b_{ij}$	Bias of the $j$ th neuron in the $i$ th layer.
$\text{Var}(b_{ij})$	Variance of the bias of the $j$ th neuron in the $i$ th layer.

### 3.1 Evolutionary Encoding

Each individual is encoded in a structure in which basic information is maintained as illustrated in Table 1. For each simulation a new population of a specific subset of NN architectures, MLPs, is created. The genotype is represented by an input layer and a number of hidden layers together with the output layer, equal to the layer vector size. The input and output sizes are pre-established at network definition and maintained unchanged during the entire evolutionary process. The input size is 26 or 156 depending on the encoding scheme (see Section 2). The output size is given by the number of senses of the target word. The number of hidden nodes in the  $i^{th}$  hidden layer corresponds to the number specified in the  $i^{th}$  element in the topology vector (see Figure 1, top), while the chromosome of the individual is defined by the corresponding MLP (see Figure 1, bottom).

**Figure 1. Genotype representation of the NN.**

Individuals are not constrained to a pre-established topology, and the population is initialized with different hidden layer sizes and different numbers of neurons for each individual according to two exponential distributions, in order to maintain diversity among all the individuals in the new population. The number of neurons in each hidden layer is constrained to be greater than or equal to the number of network outputs, in order to avoid hourglass structures, whose performance tends to be poor. Indeed, a layer with fewer neurons than the outputs destroys information which later cannot be recovered. Initial weights and biases are extracted from a normal distribution. Like in evolution strategies [5], for each connection weight and neuron bias encoded in the genotype, an associated variance is also encoded, which determines the probability distribution of mutations and is used to self-adapt the mutation step.

### 3.2 Fitness Function

We adopt the convention that a lower fitness means a better NN, mapping the objective function into an error minimization problem. The fitness of an individual is calculated based on the confusion matrix, by means of the following equation:

$$f = N_{\text{outputs}} - \text{Trace}, \quad (1)$$

where  $N_{\text{outputs}}$  is the number of output neurons and Trace is the sum of the diagonal elements of the row-wise normalized confusion matrix, which represents the conditional probabilities of the predicted outputs given the actual outputs.

Following the commonly accepted practice of machine learning, the problem data are partitioned into three sets: training, test and validation set, used respectively to train, to stop the training, thus avoiding overfitting, and to assess the generalization capabilities of a network. In the neuro-genetic approach the fitness is calculated for each individual on the test set after applying BP.

### 3.3 Genetic Operators

The evolutionary process is based on the genetic operators of selection and mutation. Recombination is not used, due to the detrimental effects on NNs [14].

The selection method implemented in this work follows the breeder genetic algorithm. Elitism is also used, allowing the best individual to survive unchanged in the next generation and solutions to monotonically get better over time. The algorithm uses truncation selection with a proportion of individuals selected for reproduction of  $\frac{1}{2}$ : starting from a population of  $n$  individuals, the worst  $\lfloor n/2 \rfloor$  (with respect to  $f$ ) are eliminated. The remaining individuals are then duplicated in order to replace those eliminated, and finally, the population is randomly permuted.

Two kinds of NN perturbations are used: a general weight mutation, that perturbs the weights of the neurons before performing any structural mutation and applying BP, and a topology mutation, that is defined with four types of mutation by considering neutral operations for neurons and layer addition and elimination.

- *Weight Mutation* defines a Gaussian distribution for the variance matrix values  $\mathbf{Var}^{(i)}$  of each weight matrix  $\mathbf{W}^{(i)}$ , defined in Table 1, much like in *evolution strategies* [6]. All the weight matrices  $\mathbf{W}^{(i)}$  and the corresponding biases are perturbed by using variance matrices and evolutionary strategies applied to the synapses of each NN. The main idea behind these strategies is to allow a control parameter, like mutation variance, to self-adapt rather than changing their values by some deterministic algorithm.

- *Weight Control* is applied to each NN after weight perturbation. For each hidden layer with a number of neurons greater than the number of outputs, the neurons whose contribution to the network output is negligible (i.e., strongly below-average, as detailed in [3]) are eliminated.
- *Topology Mutation* defines layer addition and elimination and neuron insertion with independent probabilities, corresponding respectively to three algorithm parameters  $p_{\text{layer}}^+$ ,  $p_{\text{layer}}^-$  and  $p_{\text{neuron}}^+$ , set at the beginning and maintained unchanged during the entire evolutionary process.

*Layer insertion:* with probability  $p_{\text{layer}}^+$ , a hidden layer  $i$  is randomly selected and a new hidden layer  $i+1$  with the same number of neurons is inserted after it. All the nodes of the parent layer are connected to all the nodes of the offspring by defining an Identity matrix. The previous weight connections of the parent are then shifted to the output connections of the offspring layer.

*Layer elimination:* with probability  $p_{\text{layer}}^-$  a hidden layer  $i$  is randomly selected, with the condition that the NN must have at least two hidden layers. The layer  $i$  is removed from the structure and the connections between the  $(i-1)^{\text{th}}$  layer and the  $(i+1)^{\text{th}}$  layer are redefined by combining their values in order to reflect the influence of the hidden layer that is removed from the NN.

*Neuron insertion:* with probability  $p_{\text{neuron}}^+$ , the  $j^{\text{th}}$  neuron in the hidden layer  $i$  is randomly selected for duplication. A copy of it is inserted into the same layer  $i$  as the  $(N_i + 1)^{\text{th}}$  neuron. The high correlation between parents and offspring is achieved, first, by duplicating the input connection weights of the parent in order to define the connection weights of the offspring. Then, the output weights of the parent are equally divided and the half is assigned to the offspring.

## 4 Experiments and Results

Several experiments have been carried out in order to find out the optimal settings of the genetic parameters  $p_{\text{layer}}^+$ ,  $p_{\text{layer}}^-$ , and  $p_{\text{neuron}}^+$ . For each run of the EA, up to 250,000 network evaluations (i.e., simulations of the network on the whole training set) have been allowed, including those performed by the BP algorithm. The first group of experiments was carried out on the word *memory*, and the best experimental solutions have been found with  $p_{\text{layer}}^+$ ,  $p_{\text{layer}}^-$  and  $p_{\text{neuron}}^+$  equal to 0.05, although they do not differ significantly from other solutions. From the experiments carried out, the best disambiguation ratio has been collected, as well as its average and standard deviation.

The best settings found for *memory* were subsequently used for the disambiguation of the remaining words; the results of these experiments are reported in Table 2. The third column in the table provides, as a benchmark, the expected disambiguation accuracy over all words with a given number of senses, as computed by Galley and McKeown [8], using the semantic concordance corpus (semcor), a corpus extracted from the Brown Corpus and semantically tagged with WordNet senses. For instance, they have calculated that, on average, WSD methods have an accuracy of 75% when applied to all words with two senses.

The results reported in the second-last column of Table 2 are relevant to the best results obtained by another technique recently reported in the literature, namely the basic classifiers used in [11]; these classifiers are created by using combinations of features commonly used in several WSD classifiers.

Out of the 14 words for which comparison data are available, Mihalcea’s basic classifiers outperform our evolved NNs 4 times only, namely for the words *channel*, *day*, *feeling*, and *lady*. The last column of Table 2 shows the accuracy of feedforward neural networks without evolutionary process. Here only one case is better than those based on EANNs, for the word *bar*.

The superiority of the positional encoding scheme over the letter count scheme is evident, but not so overwhelmingly as one would expect given that the information provided as input to the NNs is 6 times more. Furthermore, in all but one case, the letter count encoding scheme outperforms Michalcea’s basic classifiers when the positional encoding does.

It should be noticed that not all senses differ semantically by the same amount. To this aim, the table reports information about distance among different senses of each word, measured as the minimum number of ontology edges connecting two concepts in WordNet. It can be observed that, for instance, the senses of *day* are quite close to each other. This means that confusion of senses may be expected, as the contexts in which semantically close meanings of a word are used may be very similar or even coincide. As a matter of fact, a quick inspection of the results suggests that better accuracy is obtained by our method for words whose senses are more apart.

If one factors distance between senses in, the results shown in Table 2 look quite promising. Indeed, for all applications of WSD, what is more critical is to be able to disambiguate between senses that are semantically far apart.

## 5 Discussion and Concluding Remarks

To provide a more detailed, critical discussion of the results, it will be convenient to consider the graphical representation of the corresponding row-wise normalized confu-

**Table 2. A summary of the results of applying the neuro-genetic approach to the disambiguation of 15 test words, with a comparison to the results obtained by Mihalcea on 14 of the same words.**

Word	# of Senses	Expected Accuracy	Dataset Size	Max. Dist.	Min. Dist.	Avg. Dist.	Letter Count Accuracy	Positional Accuracy	Mihalcea "Basic" [11]	Simple ANN
bar	11	-	59156	23	4	12	33.93	29.57	31.45	<b>34.47</b>
bum	4	57	25761	11	5	8	42.64	<b>47.12</b>	37.20	42.55
channel	6	43	34629	17	3	12	32.89	39.75	<b>43.18</b>	35.88
circuit	7	35	35457	19	5	13	42.16	<b>50.43</b>	40.35	47.01
day	9	35	64072	11	2	5	35.26	29.46	<b>45.52</b>	28.98
detention	2	75	3971	12	12	12	92.34	<b>95.19</b>	79.16	68.68
dyke	2	75	5560	13	13	13	97.40	<b>98.73</b>	38.61	91.67
feeling	7	35	14269	12	3	7	32.06	35.38	<b>39.21</b>	30.26
grip	5	52	7464	19	10	14	48.93	<b>56.77</b>	53.46	41.98
hearth	3	62	6856	10	7	8	50.88	<b>61.94</b>	44.82	47.67
holiday	2	75	17215	6	6	6	88.56	<b>90.75</b>	84.61	88.92
lady	3	62	20145	7	2	5	46.61	50.15	<b>61.53</b>	45.82
material	6	43	44962	12	4	8	68.36	<b>69.73</b>	37.28	68.36
memory	5	52	12963	17	6	10	57.12	<b>62.00</b>	-	46.60
post	6	43	21781	20	5	11	59.05	<b>59.70</b>	37.93	53.38

sion matrices, which may also be regarded as the characteristic matrix  $C$  of an information transmission channel. Unfortunately, the results of other related work previously carried out in the literature [11] are not discussed in terms of the confusion matrix, but only of the coarser (and less accurate) percentage of accuracy.

The confusion matrices for the 15 words considered with the two representation schemes are shown in Figure 2.

A first analysis over the two kinds of representations show how in the most cases the positional choice for the data representation results better than the occurrences one, even with a reduced number of inputs to the neural classifiers.

In the two representations of the word *memory*, for example, the EA classifies with satisfactory performances three of the five senses of the word listed in Table 3, respectively the first, the fourth and the fifth sense. The second and the third senses are not classified as well, in that they tend to be confused with the first sense. A similar observation holds for the words *bar*, *circuit* and *grip* as well. This fact is mainly due to the high similarity among some of the word senses, as well as among all the sentences in the dataset associated to each of them.

The last two senses of *memory* are, instead, very different from the others, and the evolved classifiers perform quite well in recognizing them, even if the occurrences of the last sense account for but a very small percentage, i.e., 1.08%, of the overall dataset used for word *memory*, as shown in Table 3.

A completely different case is given by considering some words with two senses, like *holiday* and *detention*. Indeed, for these two words the evolved NN uses the "shortcut" of always recognizing only one of the two senses, without considering the other. The unbalance of the dataset for the two senses, visible in Table 3, would seem a possible ex-

planation. However, further experiments with artificially balanced datasets for the same words have shown similar results and, after all, the way fitness is calculated neutralizes the bias of unbalanced data. Anyway, in such cases, the "lazy" strategy of always guessing the most common sense pays off very well.

Nevertheless, when the senses of a word are more far apart, the neuro-genetic classifier obtains satisfactory performances by recognizing all the word senses. An example is given by the words *hearth* and *dyke*, as shown in Figure 2. In particular, for the latter, the system reaches an accuracy of 98.73, as opposed to 38.61 obtained by Mihalcea [11], which is, by the way, worst than a random choice between the two senses of that word!

At first sight, creating a single NN for every ambiguous word might seem hardly practical or even infeasible. However, there are just 15,935 polysemous words out of the 117,798 WordNet entries. Evolving a NN for disambiguating a polysemous word takes, on an ordinary desktop PC, two hours on average. Assuming some 30 PCs are available day and night, 45 days would be enough to evolve a NN for each polysemous word. We estimate that the entire set of almost 16,000 NNs would occupy 30 Mbytes. When disambiguating a document, a stored NN would be recalled from the database and executed every time a polysemous word were encountered. Recalling a network can take a few milliseconds, whereas executing it is just a matter of microseconds. Therefore, the approach we propose can be considered realistic and feasible with state-of-the-art technology.

## References

- [1] A. Abraham. Meta learning evolutionary artificial neural networks. In *Neurocomputing*, volume 56, pages

**Table 3. Senses of the words *memory* and *holiday*.**

Word	Occurrences (%)	Sense
memory	3576 (27.59)	1. Something that is remembered.
	2571 (19.83)	2. The cognitive processes whereby past experience is remembered.
	727 (5.61)	3. The power of retaining and recalling past experience.
	5949 (45.89)	4. An electronic memory device.
	140 (1.08)	5. The area of cognitive psychology that studies memory processes.
holiday	1041 (6.05)	1. Leisure time away from work devoted to rest or pleasure.
	16174 (93.95)	2. A day on which work is suspended by law or custom.

(lc)	(p)	(lc)	(p)	(lc)	(p)	(lc)	(p)	(lc)	(p)
	bar		bum		channel		circuit		day
	detention		dyke		feeling		grip		hearth
	holiday		lady		material		memory		post

**Figure 2. Confusion matrices of the words analyzed by the evolutionary approach, with the letter count (lc) encoding and the positional (p) encoding.**

1–38, 2004.

- [2] E. Agirre, E. Alfonseca, and O. Lopez. Approximating hierarchy-based similarity for WordNet nominal synsets using topic signatures. In *Proc. of the 2nd Global WordNet Conference.*, 2004.
- [3] A. Azzini and A. Tettamanzi. A neural evolutionary approach to financial modeling. *GECCO'06*, vol 2, pages 1605–1612. Morgan Kaufmann, San Francisco, CA, 2006.
- [4] A. Azzini and A. Tettamanzi. A neural evolutionary classification method for brain-wave analysis. In *EVOIASP'06*, pages 500–504, 2006.
- [5] T. Bäck, F. Hoffmeister, and H. Schwefel. A survey of evolutionary strategies. In R. Belew and L. Booker, editors, *ICGA 4*, pages 2–9, San Mateo, CA, 1991. Morgan Kaufmann.
- [6] T. Bäck and H. Schwefel. Evolutionary computation: An overview. In *CEC'96*, pages 20–29, Nagoya, Japan, May 1996. IEEE Press.
- [7] G. W. Cottrell. *A Connectionist Approach to Word Sense Disambiguation*. Pitman, London, 1989.
- [8] M. Galley and K. McKeown. Improving word sense disambiguation in lexical chaining. In *IJCAI'03*, pages 1486–1488, 2003.
- [9] G. E. Hinton, J. L. McClelland, and D. E. Rumelhart. Distributed representations. In G. E. Hinton, J. L. McClelland, and D. E. Rumelhart, editors, *Parallel Distributed Processing: explorations in the microstructure of cognition*. MIT Press, Cambridge, MA, 1986.
- [10] C. Leacock, M. Chodorow, and G. A. Miller. Using corpus statistics and WordNet relations for sense identification. *Computational Linguistics*, 24(1):147–165, 1998.
- [11] R. Mihalcea. Co-training and self-training for word sense disambiguation. In *Proc. of the Conference on Natural Language Learning*, 2004.
- [12] H. Schütze. Word space. In *NIPS '93*, pages 895–902, San Francisco, CA, 1993. Morgan Kaufmann.
- [13] D. L. Waltz and Pollack. Massively parallel parsing: A strongly interactive model of natural language interpretation. *Cognitive Science*, 9:51–74, 1985.
- [14] X. Yao and Y. Liu. A new evolutionary system for evolving artificial neural networks. *IEEE Transactions on Neural Networks*, 8(3):694–713, May 1997.