

## Evolving Neural Word Sense Disambiguation Classifiers with a Letter-Count Distributed Encoding

A. Azzini and C. da Costa Pereira and 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  
E-mail: azzini,dragoni,pereira,tettamanzi@dti.unimi.it*

We propose a supervised approach to word sense disambiguation (WSD), 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.

*Keywords:* evolutionary algorithms; artificial neural networks; supervised learning; word sense disambiguation.

### 1. Introduction

The automatic disambiguation of word senses consists of assigning the most appropriate meaning to a polysemous word, i.e., a word with more than one meaning, 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 a distributed encoding scheme, based on the way words are written, to represent the context in which a word occurs. The paper is organized as follows: Section 2 briefly describes the EA used in this work, whereas Section 3 explains its application to WSD, followed, in Section 4, by the results of experiments.

Section 5 concludes with a discussion of the results.

## 2. The Neuro Evolutionary Algorithm

Their tolerance for noise, their ability to generalize, and their well-known suitability for classification tasks<sup>1</sup> 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,<sup>15</sup> and one of the most promising approaches to using EAs to design and optimize NNs jointly evolves network architecture and weights, without any intervention by an expert.<sup>1</sup>

We use an evolutionary approach for NN design, previously validated on different benchmarks and real-world problems,<sup>3,4</sup> which uses a variable-size representation of individuals. A population of classifiers — the individuals — is maintained, 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 takes advantage of 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.

### 2.1. 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. The input size is 26 neurons (see Section 3). 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).

Individuals are not constrained to a pre-established topology, and the population is initialized with different hidden layer sizes and different num-

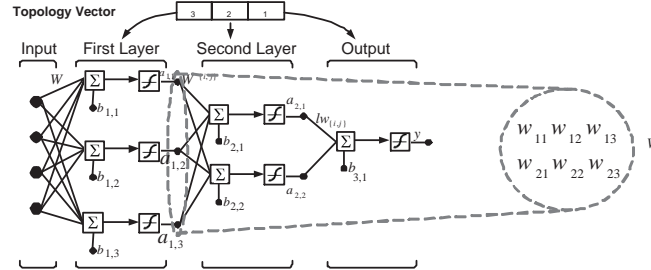


Fig. 1. Genotype representation of the NN: example of a topology vector and its associated weight matrix  $W$ .

bers 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, that could reduce the performances. 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,<sup>5</sup> 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.

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 for every element of $\mathbf{W}^{(i)}$ , $i = 1, \dots, l$ .
$b_{ij}$	Bias of the $j$ th neuron in the $i$ th layer.
$\mathbf{Var}(b_{ij})$	Variance of the bias of the $j$ th neuron in the $i$ th layer.

## 2.2. 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 over-fitting, 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.

### 2.3. Genetic Operators

The evolutionary process is based on the genetic operators of selection and mutation, described in detail in a previous work,<sup>3</sup> and here briefly reported. Recombination is not used, due to its detrimental effects on NNs.<sup>15</sup>

The selection method implemented in this work uses elitism, allowing the best individual to survive unchanged into 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 perturbs the structure maintaining the overall network behavior unchanged.

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*.<sup>6</sup> 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<sup>3</sup>) are eliminated. Topology Mutation defines then 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.

### 3. Application to Word Sense Disambiguation

The algorithm presented in Section 2 is used to evolve 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.

#### 3.1. Dataset

We used IXA Group's web corpus,<sup>2</sup> which comprises all WordNet noun senses. Its construction is inspired by the "monosemous relatives" method.<sup>10</sup> From this web corpus we extracted all data relevant 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<sup>9</sup> and localist schemes.<sup>7,11</sup> 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 disambiguate an entire text.

Examples of distributed schemes are microfeatures<sup>14</sup> and the word-space model.<sup>13</sup> 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 a distributed *letter count* representation

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.

The activation of the input neurons is 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

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  $\rightarrow$  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  $n = 2$ ). 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.

#### 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., executions 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. The best settings found for *memory* are 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,<sup>8</sup> 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.

Table 2. Disambiguation Accuracy.

Word	# of Senses	Exp. Acc.	Dataset Size	Max. Dist.	Min. Dist.	Avg. Dist.	Neuro-Gen Accuracy	Mihalcea "Basic" <sup>12</sup>
bar	11	-	59156	23	4	12	33.93	31.45
bum	4	57	25761	11	5	8	42.64	37.20
channel	6	43	34629	17	3	12	32.89	43.18
circuit	7	35	35457	19	5	13	42.16	40.35
church	3	62	9366	17	6	12	46.94	52.77
day	9	35	64072	11	2	5	35.26	45.52
detention	2	75	3971	12	12	12	92.34	79.16
dyke	2	75	5560	13	13	13	97.40	38.61
feeling	7	35	14269	12	3	7	32.06	39.21
grip	5	52	7464	19	10	14	48.93	53.46
hearth	3	62	6856	10	7	8	50.88	44.82
holiday	2	75	17215	6	6	6	88.56	84.61
lady	3	62	20145	7	2	5	46.61	61.53
material	6	43	44962	12	4	8	68.36	37.28
memory	5	52	12963	17	6	10	57.12	—
mouth	8	33	25309	17	3	11	53.26	50.00
post	6	43	21781	20	5	11	59.05	37.93
yew	2	75	11295	17	17	17	75.34	70.00

The results reported in the last column of Table 2 are relevant to the results obtained by another technique recently reported in the literature, namely the basic classifiers,<sup>12</sup> that are created by using combinations of features commonly used in several WSD classifiers.

Out of the 18 words for which comparison data are available, Mihalcea's basic classifiers outperform our evolved NNs 5 times only, namely for the words *channel*, *church*, *day*, *feeling*, and *lady*.

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, we consider the graphical representation of the corresponding row-wise normalized confusion 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<sup>12</sup> 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 18 words considered with the two representation schemes are shown in Figure 2.

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.

In the representation of the word *memory* 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*, *mouth*, *day* 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 explanation. 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

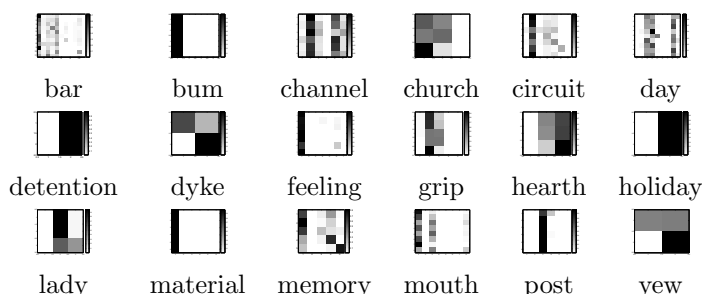


Fig. 2. Confusion matrices of the words analyzed by the evolutionary approach.

always guessing the most common sense pays off very well.

The performance of “lazy” classifiers would be judged satisfactory with respect to other related works, as indicated in Table 2. Indeed, together with *holiday*, the neuro-genetic approach provides satisfactory results with *detention*, by obtaining an accuracy of 92.34 as opposed to 79.16 obtained from Mihalcea.<sup>12</sup> It should be argued that a “lazy” strategy would be anyway the winning strategy for real-world applications, when what really matters is simply being correct most of the times. Nevertheless, in those cases in which 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*, *yew* and *dyke*, as shown in Figure 2. In particular, for the latter, the system reaches an accuracy of 97.40, as opposed to 38.61 obtained by Mihalcea,<sup>12</sup> which is, by the way, worse 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. The next step will be to apply this approach to a larger set of polysemous words and to investigate different input encoding strategies.

## References

1. A. Abraham. Meta learning evolutionary artificial neural networks. In *Neurocomputing*, volume 56, pages 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. In *Proc. of the Genetic and Evolutionary Computation Conference, GECCO'06*, volume 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 *Proc. of the European Workshop on Evolutionary Computation in Image Analysis and Signal Processing, 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, *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 2–9, San Mateo, CA, 1991. Morgan Kaufmann.
6. T. Bäck and H. Schwefel. Evolutionary computation: An overview. In *Proceedings of the IEEE International Conference on Evolutionary Computation*, 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 *Proc. of the Eighteenth International Joint Conference on Artificial Intelligence*, 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. M. Lesk. Automated sense disambiguation using machine-readable dictionaries: How to tell a pine cone from an ice cream cone. In *Proc. SIGDOC Conference*, 1986.
12. R. Mihalcea. Co-training and self-training for word sense disambiguation. In *Proc. of the Conference on Natural Language Learning*, 2004.
13. H. Schütze. Word space. In *Proc. of the 1993 Conference on Advances in Neural Information Processing Systems, NIPS '93*, pages 895–902, San Francisco, CA, 1993. Morgan Kaufmann.
14. D. L. Waltz and Pollack. Massively parallel parsing: A strongly interactive model of natural language interpretation. *Cognitive Science*, 9:51–74, 1985.
15. 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.