Part-of-Speech Tagging Using Evolutionary Computation

Ana Paula Silva[1], Arlindo Silva[1], and Irene Rodrigues[2]

[1] Escola Superior de Tecnologia do Instituto Politécnico de Castelo Branco, Portugal
{dorian,arlindo}@ipcb.pt
[2] Universidade de Évora, Portugal
ipr@uevora.pt

Abstract. Part-of-speech tagging is a task of considerable importance in the field of natural language processing. Its purpose is to automatically tag the words of a text with labels that designate the appropriate parts-of-speech. The approach proposed in this paper divides the problem into two tasks: a learning task and an optimization task. Algorithms from the field of evolutionary computation were adopted to tackle each of those tasks. We emphasize the use of swarm intelligence, not only for the good results achieved, but also because it is one of the first applications of such algorithms to this problem. This approach was designed with the aim of being easily extended to other natural language processing tasks that share characteristics with the part-of-speech tagging problem. The results obtained in two different English corpora are among the best published.

Keywords: Part-of-speech Tagging, Disambiguation Rules, Evolutionary Algorithms, Particle Swarm Optimization, Natural Language Processing.

1 Introduction

The words of a language are usually grouped in lexical categories or parts-of-speech (POS). A tagger is a system that should receive a text, made of sentences, and, as output, should return the same text, but with each of its words associated with the correct POS tag. These tags are acronyms for the lexical categories chosen for labeling the words. The process of classifying words into their POS, and labeling them accordingly, is known as POS tagging, or, simply, tagging. In most languages, each word has a set of lexical categories that represent the roles that they can assume in a sentence. When the cardinality of this set is greater than one, we say that the word is ambiguous. The context of a word, i.e., the lexical categories of the surrounding words, is the fundamental piece of information for determining its role in a sentence. For instance, the word *wind* can assume the function of a **verb**, if it follows the word *to*, or can be used as a **noun** if it is preceded by a determiner like *the*. According to this, most taggers take into consideration the context of a word to decide which should be its tag.

G. Terrazas et al. (eds.), *Nature Inspired Cooperative Strategies for Optimization (NICSO 2013)*, Studies in Computational Intelligence 512,
DOI: 10.1007/978-3-319-01692-4_13,

However, each of the words belonging to a word's context can also be used in different ways, and that means that, in order to solve the problem, a tagger should have some type of disambiguation mechanism that allows it to choose the proper POS tags for all the words of a sentence.

The methods used for solving the POS tagging problem can be divided into two distinct groups, based on the information they use. In one group, we can gather the approaches that use statistical information about the possible contexts of the various word tagging hypotheses. Most of the stochastic taggers are based on hidden Markov models. In the other group, we find rule based taggers [1–3]. The rules are usually discovered automatically, and its purpose is to correct errors resulting from an initial basic tagging. Brill's tagger [1] is perhaps the most popular tagger based on rules.

More recently, several works following an evolutionary approach have been published. These taggers can also be divided by the type of information they use to solve the problem: statistical information [4, 5], and rule-based information [2]. In the former, an evolutionary algorithm is used to assign the most likely tag to each word of a sentence, based on a training table that basically has the same information that is used in the traditional probabilistic approaches. The later is inspired by Brill's rule based tagger. In this case a genetic algorithm (GA) is used to evolve a set of transformations rules, which will be used to tag a text in much the same way as in Brill's tagger. While in [4, 5], the evolutionary algorithm is used to discover the best sequence of tags for the words of a sentence, using an information model based on statistical data, in [2] the evolutionary algorithm is used to evolve the information model itself, in the form of a set of transformation rules.

Although the POS tagging problem is a task that has had a special attention in the field of natural language processing (NLP), the evolutionary approach deserves, in our opinion, a more thorough study. We believe that this study should include the application of other algorithms from the evolutionary computation field. Moreover, previous work suggest the exploitation of these algorithms on two key aspects of the task: the information gathering and the automatic process to perform the tagging, according to the information collected. In this paper, we present a new evolutionary approach to the POS tagging problem. Our strategy implies a division of the problem into two different tasks: a learning task and an optimization task. These are tackled using not only evolutionary algorithms, but also particle swarm optimization (PSO), resulting, as far as we know, in the first attempt to approach this problem using swarm intelligence. Although focusing mainly on the POS tagging problem, we believe that this work may be the foundation for a new paradigm to solve other NLP tasks. This paradigm is based, however, in two fundamental assumptions:

– With the help of a classification algorithm, it is possible to generalize, from linguistic resources, the information typically used in the probabilistic approach, by learning a set of disambiguation rules. These rules will not play the role of a classifier, instead they will be used as an heuristic to help solve the task in question.

– It is possible to formalize the main problem as a search problem and use the rules discovered in the first phase as an heuristic to guide the search for a solution in the problem state space.

The field of evolutionary computation includes a set of global optimization algorithms that have been applied, with recognized success, to a vast and varied set of problems in areas such as optimization, search and learning. These algorithms are characterized by being easily adapted to different representations and tasks. They are also global optimization algorithms, hence outperforming many of the greedy approaches. Therefore, they present themselves as a suitable tool to integrate the approach we propose here, since they can be used in both phases of the strategy. Also the inherent versatility of these algorithms contributes to strengthen the possibility of applying this approach to other NLP tasks.

2 Rules Discovery Using Evolutionary Computation

It is our belief that the information stored in the training tables of the probabilistic approach can be interpreted as a set of instances. Each of these instances is typically described by a set of measurable attributes related to the tags of the surrounding words, and is associated with a numerical value that identifies the number of times each one occurs in the training corpus. Naturally, this information is specific to the corpus from which it was collected and does not show any degree of generalization, instead it can easily be interpreted as an extensive and comprehensive collection of information. Hence we are convinced that it is admissible to investigate the possibility of generalizing this information using a classification algorithm. From this generalization we expect to be able to reduce the amount of information needed to solve the problem and also to improve the tagging accuracy. The learned rules may be used, in a similar way to the training table, to guide the search of the POS tagging problem state space. They aim not to classify a given word, but rather assess the quality of a particular classification.

Previous experience with classification rules discovery [6, 7], using evolutionary computation, has led us to define the classification algorithm based on a covering algorithm. The outline of the algorithm used is defined in Algorithm 1. As we can see, the set of rules is obtained by executing the search algorithm m times. This algorithm is responsible for determining the best classification rule for the set of training examples it receives as input. At each execution, the rule obtained is stored, along with its quality value, and the set of positive examples is updated by eliminating all the instances covered by the rule. The search algorithm will be executed as many times as necessary, so that all positive examples are covered, i.e., the set of positive examples is the empty set. We divided the problem into n distinct classification problems, n being the number of different tags used in the annotated corpus, from which the rules will be learned and that define the tag set E. Each tag $e \in E$ presented in the corpus determines a classifying object, with possible classes taking values from the

discrete set $Y = \{Yes, No\}$. Two different search algorithms were tested: one based on a GA and another based on a PSO. A more detailed description of the implemented algorithms can be found in [8, 9].

Algorithm 1. Covering Algorithm

Require: PosExemples, NegExemples
Ensure: RulesSet
  while PosExemples $\neq \varnothing$ **do**
    ⟨BestRule, Quality⟩ ← SearchAlgorithm(PosExemples, NegExemples)
    PosExemples ← Remove(PosExemples,BestRule)
    RulesSet ← Add(RulesSet, ⟨BestRule, Quality⟩)
  end while

2.1 Prediction Attributes and Representation

The use of rules allows, in addition to the grammatical categories of the surrounding words, the consideration of other aspects. Although a word context is perhaps the most determinant piece of information to identify its lexical category, there are also some other aspects that can be helpful. The internal structure of a word may give useful clues as to the word's class [10]. For example, *-ing* is a suffix that is most commonly associated with gerunds, like *walking, talking, thinking, listening*. We also might guess that any word ending in *-ed* is the past participle of a verb, and any word ending with *'s* is a possessive noun. Taking these observations into account, we considered as prediction attributes two distinct groups. The first group includes six attributes related with the context: the lexical categories of the third, second and first words to the left, and the lexical categories of the first, second, and third words to the right of a particular word. The second group comprises the following information about the words: if the word is capitalized, if the word is the first word of the sentence, if the word has numbers or '.' and numbers, and some words' terminations like ***ed***, ***ing***, ***es***, ***ould***, ***'s***, ***s***. The possible values for each of the first group's attributes are the values of the corpus tag set from which the search algorithm will learn the rules. This set will depend on the annotated corpus used, since the tag set will vary for different annotated corpora. The remaining attributes were defined as boolean.

The training sets were built from the Brown corpus. For each word of the corpus, we collected the values of every attribute in the rule's antecedent, creating a specific training example. Next, for each tag of the tag set, we built a training set made by positive and negative examples of that tag. The building process used to define each of the training sets was the following: for each example e_i of the set of examples, with word w and tag t, if w is an ambiguous word, with S the set of all its possible tags, then put e_i in the set of positive examples of tag t, and put e_i in the set of negative examples of all the tags in S, except t.

We used a binary representation for the rules. The attributes related with the context were codified, each one, by six bits. The first bit indicates whether the attribute should or should not be considered, and the following five bits represent

the assumed value of the attribute in question. We adopted a table of 20 entries to store the tag set, and used the binary value represented by five bits to index this table. If the value exceeds the number 20, we used the remainder of the division by 20. The remaining attributes were encoded by 18 bits, two bits for each of the nine attributes. In the same way, the first bit indicates if the attribute should or shouldn't be considered, while the second bit, indicates whether the property is, or is not, present. We adopted a Michigan approach, thus, in both implementations of the search algorithm, each particle/individual represents a rule using the codification described. In short, each particle/individual was composed by $6 \times 6 + 2 \times 9 = 54$ bits.

2.2 Search Algorithm

For the PSO based search algorithm we adopted the binary version presented by Kennedy [11]. The genetic algorithm based version follows the classical GA with binary representation [12]. We used, as genetic operators, the two point crossover (with 0.75 probability) and the binary mutation (with 0.01 probability). The selection scheme used was a tournament selection with tournaments of size two and $k = 0.8$.

The formula used to evaluate each rule, and therefore to set its quality, is expressed in Equation 1. This formula penalizes a particle/individual that represents a rule that ignores the first six attributes, which are related with the word's context, forcing it to assume a more desirable form. The others are evaluated by the well known F_β-measure (see Equation 2). The F_β-measure can be interpreted as a weighted average of precision and recall. We used $\beta = 0.09$, which means we put more emphasis on precision than recall.

$$Q(X) = \begin{cases} F_\beta(X) \text{ if } X \text{ tests at least one of the first six attributes} \\ -1 \qquad\quad \text{otherwise} \end{cases} \tag{1}$$

$$F_\beta(X) = (1 + \beta^2) \times \frac{precision(X) \times recall(X)}{\beta^2 \times precison(X) + recall(X)} \tag{2}$$

3 POS-Tagger

By definition, a POS-tagger should receive as input a non annotated sentence, $\mathbf{w}$, made of n words, w_i, and should return the same sentence, but now with all the w_i marked with the appropriate tag. Assuming we know all the possibilities, W_i, of tagging each of the words w_i of the input sentence, the search space of the problem can be defined by the set $W_1 \times W_2 \times \cdots \times W_m$. Therefore the solution can be found by searching the problem state space. We believe that this search can be guided by the disambiguation rules found earlier. We tested two different global search algorithms: a genetic algorithm (GA-Tagger) and a binary particle swarm optimizer (PSO-Tagger).

The taggers developed were designed to receive as inputs a sentence, $\mathbf{w}$, a set of sets of disambiguation rules, D_t, and a dictionary, returning as output

the input sentence with each of its words labeled with the correct POS tag. The search algorithm evolves a swarm/population of particles/individuals, that encode, each of them, a sequence of tags for the words of the input sentence. The quality of each particle/individual is measured using the sets of disambiguation rules given as input. Again, a more detailed description of the implemented taggers can be found in [8, 9].

3.1 Representation

The representation used in the two implemented algorithms is slightly different. In the GA-Tagger, we adopted a symbolic representation. An individual is represented by a chromosome $\mathbf{g}$ made of a sequence of genes. The number of genes in a chromosome equals the number of words in the input sentence. Each gene, g_i, proposes a candidate tag for the word, w_i, in the homologous position. The possible alleles for gene g_i, are the elements of the set W_i.

Since we adopted the binary version of the PSO algorithm, we used, in this case, a binary representation. To encode each of the tags belonging to the tag set, we used a string of 5 bits. Therefore, a particle that proposes a tagging for a sentence with n ambiguous words will be represented by $n \times 5$ bits. Each five bits of a particle encode a integer number that indexes a table with as much entries as the possible tags for the correspondent ambiguous word. If the integer number, given by the binary string, exceeds the table size, we use as index the remainder of the division by the table size value.

3.2 Tagging Evaluation

The quality of the overall tagging, $\mathbf{t}$, is given by the sum of the evaluation results of each tag assignment, t_i for each word w_i. A particle/individual representing a sequence of n tags, $\mathbf{t}$, for a sentence with n words will give rise to a set of n pairs $\langle \mathbf{x}_i, t_i \rangle$, with $\mathbf{x}_i$ denoting the correspondent 15-tuple collecting the values of the 15 attributes presented in the antecedent of the disambiguation rule. The quality of each tag assignment, t_i, is measured by assessing the quality of the pair $\langle \mathbf{x}_i, t_i \rangle$, with $\mathbf{x}_i$ using Equation 3.

$$h(\langle \mathbf{x}_i, t_i \rangle) = \begin{cases} q_k & \text{If } \langle r_k, q_k \rangle \in D_{t_i} \text{ and } r_k \text{ covers } \mathbf{x}_i \\ 0 & \text{Otherwise} \end{cases} \tag{3}$$

The quality of a particle/individual is given by Equation 4, with T representing the set of all n pairs $\langle \mathbf{x}_i, t_i \rangle$.

$$Quality(T) = \sum_{j=1}^{n} h(T_j) \tag{4}$$

4 Experimental Results

We developed our system in Python and used the resources available on the NLTK (Natural Language Toolkit) package in our experiences. The NLTK

package provides, among others, the Brown corpus and a sample of 10% of the WSJ corpus of the Penn Treebank. It also provides several Python modules to process those corpora. The experimental work was done in two phases. First the disambiguation rules were discovered and, after that, the POS taggers were tested. The results achieved in each phase are presented in the next subsections.

4.1 Disambiguation Rules

As we said before, tagged corpora use many different conventions for tagging words. In order to be able to use the disambiguation rules learned from the Brown corpus to tag text from other corpora, we used the *simplify_tags=True* option of the *tagged_sentence* module of NLTK corpus readers. When this option is set to *True*, NLTK converts the respective tag set of the corpus used to a uniform simplified tag set, composed by 20 tags. This simplified tag set establishes the set of classes we use in our algorithm. We ran the covering algorithm for each one of these classes and built, for each one, the respective sets of positive and negative examples.

We processed 90% of the Brown corpus in order to extract the training examples, and, for each word found, we built the corresponding instance. The total number of examples extracted from the corpus equaled 929286. We used 6 subsets of this set (with different cardinality) to conduct our experiments. We used sets of size: $3E4$, $4E4$, $5E4$, $6E4$, $7E4$ and $8E4$, which we identified with labels A, B, ..., F. For each subset, we built the sets of positive and negative examples for each tag, using the process described in the previous section.

We tested the classification algorithm both with the GA and the PSO implementation of the search algorithm. We ran the classification algorithm two times with each different implementation for each of the training sets. The GA was run with populations of size 200 for a maximum of 80 generations and the PSO with swarms of 20 particles over 200 generations. In Table 1 we present the average number of rules achieved by both algorithms and the correspondent reduction, considering the total number of positive examples (+) adopted.

Although the publications describing previous evolutionary approaches, based on training tables, do not clearly indicate the number of entries of those tables, their size is explicitly mentioned as a sensitive point concerning the algorithm

Table 1. Average number of rules discovered by the classification algorithm

		Average number of rules			
Set	+	GA	Reduction	PSO	Reduction
A	25859	2719	89.49%	2715.5	89.49%
B	33513	3081	90.81%	3124.5	90.68%
C	41080	3358.5	91.82%	3327.0	91.90%
D	48612	3735.5	92.32%	3696.5	92.39%
E	55823	4137	92.59%	4033.0	92.78%
F	63515	4399	93.07%	4288.5	93.25%

time execution [4]. While unknowing these values, the total number of positive examples considered from each of the training sets adopted, can give us an idea of the size of these tables, since the information used is similar. However, while the large training set in our case has a total of $8E4$, the previous approaches use sets with typically more than $1.5E5$. As we can see in Table 1, the rules discovered by both algorithms, allowed a significant reduction (around 90%) in the number of positive examples considered. The results also show that there are no significant differences in the number of rules discovered by the GA and the PSO.

In order to evaluate the quality of the heuristic represented by each of the discovered rules sets, we use them as input for the implemented taggers. At this point, our goal was to compare the accuracy results given by the taggers for each of the rules sets, but at the same time confirm the second hypothesis of our approach. We executed 10 times each of the taggers for each of the rules sets with the same test set. We ran the GA-Tagger with 50 individuals during 10 generations and the PSO-Tagger with swarms of 10 particles during 50 generations. The best accuracy results were systematically achieved by the PSO-Tagger and they are presented in Tables 2 and 3. We observed that the best tagging was achieved with the rules discovered from the set F during the first execution of the classification algorithm based on a GA (GA F.1)). The best set of rules discovered by the classification algorithm based on a PSO was achieved from the training examples of set C during the first run (PSO C.1).

Table 2. Tagging accuracy results achieved using the rules discovered by the GA

Set	Number of rules	Average	Best	Standard deviation
GA A.1	2740	0.9655128	0.9659605	$2.3120E-4$
GA A.2	2698	0.9647239	0.9652956	$3.7227E-4$
GA B.1	3059	0.9651449	0.9654286	$2.6196E-4$
GA B.2	3103	0.9644358	0.9649854	$2.4651E-4$
GA C.1	3355	0.9664569	0.9667583	$2.6329E-4$
GA C.2	3362	0.9654596	0.9658718	$2.6350E-4$
GA D.1	3742	0.9664258	0.9667139	$1.7882E-4$
GA D.2	3362	0.9661023	0.9664480	$3.2624E-4$
GA E.1	4166	0.9669666	0.9672458	$1.7858E-4$
GA E.2	4108	0.9666209	0.9669356	$2.4496E-4$
GA F.1	4440	**0.9672369**	**0.9677334**	$2.3248E-4$
GA F.2	4358	0.9671128	0.9677334	$2.7878E-4$

This first set of experiments enable us to identify the best heuristic and also to confirm that it is possible to formalize the POS tagging problem as a search problem and use the disambiguation rules as an heuristic to guide the search for a solution in the state space of the problem. We also concluded that the classification algorithm based on a GA was more successful than the one based on a PSO. Also we can observe, from the results achieved with the rules discovered

Table 3. Tagging accuracy results achieved using the rules discovered by the PSO

Set	Number of rules	Average	Best	Standard deviation
PSO A.1	2695	0.9635227	0.9641876	$2.5759E-4$
PSO A.2	2736	0.9629022	0.9633011	$2.8265E-4$
PSO B.1	3148	0.9628668	0.9631682	$2.6163E-4$
PSO B.2	3101	0.9649366	0.9651183	$1.5132E-4$
PSO C.1	3385	**0.9669356**	**0.9673345**	$2.3172E-4$
PSO C.2	3269	0.9650962	0.9654286	$2.3937E-4$
PSO D.1	3664	0.9655749	0.9660048	$2.3643E-4$
PSO D.2	3729	0.9661378	0.9664923	$2.4005E-4$
PSO E.1	3958	0.9650740	0.9654286	$2.0365E-4$
PSO E.2	4108	0.9654286	0.9658275	$2.4809E-4$
PSO F.1	4309	0.9655394	0.9658718	$2.1435E-4$
PSO F.2	4268	0.9656901	0.9660934	$2.0836E-4$

by the GA, a correlation between the size of the training examples set and increasing accuracy values. This allows us to expect that better heuristics could be learned by the GA using larger training sets.

4.2 POS Tagging Results

We tested the PSO-Tagger and the GA-Tagger on a test set made of 22562 words of the Brown corpus using the best set of rules found (AG F.1). We ran the PSO-Tagger 20 times with swarms of 10 and 20 particles during 50 and 100 generations. The GA-Tagger was also executed 20 times with populations of 50 and 100 individuals during 10 and 20 generations. These values were chosen so that we could test both algorithms with similar computational effort, considering the number of necessary evaluations the effort measure.

The results achieved are shown in Table 4. As we can see, the best average accuracy was achieved with the PSO-Tagger using a swarm of 20 particles evolving during 50 generations. The best accuracy result returned by the GA-Tagger is

Table 4. Tagging accuracy results achieved by both POS-taggers on a test set made of 22562 words of the Brown corpus using as heuristic the set GA F.1

Tagger	Part/Ind	Generations	Average	Best	Standard Deviation
PSO-Tagger	10	50	0.9672658	0.9679550	$2.6534E-4$
		100	0.9673123	0.9676004	$1.9373E-4$
	20	**50**	**0.9674896**	**0.9678220**	$1.9158E-4$
		100	0.9673921	0.9678663	$2.1479E-4$
GA-Tagger	50	10	0.9672170	0.9675561	$1.9200E-4$
		20	0.9672968	0.9674231	$1.1707E-4$
	100	10	0.9672591	0.9675561	$1.4097E-4$
		20	**0.9672835**	**0.9675117**	$1.0978E-4$

worst than the best result obtained with the PSO-Tagger and it needs the double number of evaluations required by the PSO-Tagger. However, the accuracy values displayed by the GA-Tagger are still very competitive when compared with others published using similar approaches.

We also tested the taggers on a test set of the WSJ corpus of the Penn Treebank made of 1000 sentences, in a total of 25184 words, using the rules discovered from the Brown corpus (see Table 5). As expected, the results achieved by the two taggers on the WSJ corpus, using as heuristic the disambiguation rules learned from the Brown corpus, are inferior to the ones obtained on the Brown corpus. However, we believe that they allow us to conclude that the discovered rules are sufficiently generic so that they can be used in different corpora. This conviction emerges from comparing the obtained results with those published by other evolutionary approaches (see Table 6). Indeed, we found that the accuracy achieved is comparable with the best published results. It is also important to stress that this values are achieved with no previous training on this corpus. The accuracy values for the WSJ corpus presented in Table 6 were achieved using all the corpus available in the NLTK package, in a total of 100676 words, setting the parameters of the algorithm with the values that provided the best results in the initial set of experiments presented in Table 5.

Table 5. Best tagging accuracy results achieved by both POS-taggers on a test set made of 25184 words of the WSJ corpus using as heuristic the set AG F.1

Tagger	Part/Ind	Generations	Average	Best	Standard Deviation
PSO-Tagger	20	50	**0.9659943**	**0.9668837**	$3.1277E-4$
GA-Tagger	100	20	**0.9660598**	**0.9663675**	$2.4541E-4$

An overview of the accuracy values achieved by the taggers in both English corpora used, is presented in Table 6, along with the results published by works using similar approaches. These results only reveal that the accuracy values obtained by the two taggers are competitive with those of past approaches. We can not directly compare our results with those published since we have no access to the test set used in the experiments made in the cited works. Nevertheless, we may conclude that for comparable size words sets (in the case of the evolutionary approaches), taken from the same corpora, the results obtained in this work are among the best published. The values shown in Table 6 were converted to percentage values and rounded to the second decimal place, so that they could be more easily compared with the ones presented in the publications cited.

Table 6. Results achieved by the two taggers on two english corpora along with the ones published by similar approaches. (Araujo - [4]; Alba, Alba-GA, Alba-PGA, Alba - [5]; Wilson - [2]; Brill - [1])

Corpus	Tagger	Training set	Test set	Best
Brown	PSO-Tagger	80000	22562	**96.78**
	GA-Tagger	80000	22562	96.76
	Araujo	185000	2500	95.40
	Alba-GA	165276	17303	96.67
	Alba-PGA	165276	17303	96.75
WSJ	PSO-Tagger	∅	100676	96.67
	GA-Tagger	∅	100676	96.66
	Wilson	600000	=Training	89.80
	Brill	600000	150000	**97.20**
	Alba	554923	2544	96.63

5 Conclusions

We described a new evolutionary approach to the POS tagging problem, which we tested using two distinct algorithms from the evolutionary computation field: a GA and a PSO. We would like to emphasize the fact that, to the best of our knowledge, this was the first attempt to apply a PSO to solve the POS tagging problem, and that, in general, there are few approaches based on swarm intelligence to solve NLP tasks.

The experimental work carried out in order to study the influence of the algorithms' parameters in the taggers' output, namely the number of generations and the number of particles/individuals, allowed us to conclude that the algorithms can find promising solutions even with reduced resources. In fact, we could not identify a clear trend of improving accuracy with increasing number of evaluations. We also observed that the best tagging accuracy was displayed by the PSO-Tagger, which allows us to conclude that swarm intelligence based algorithms can also show good results when applied to NLP problems. The results displayed by the GA-Tagger also proved to be competitive with the best ones published in previous works following a evolutionary approach.

The experiments made using the WSJ corpus and the disambiguation rules extracted from the Brown corpus gave us an idea of the degree of generalization achieved by the adopted classification algorithm. From those results, we were able to confirm that the rules obtained are sufficiently generic to be applied on different corpora. The attained generalization also reflected a substantial reduction in the information volume needed to solve the problem, while contemplating, besides the typical context information, other aspects related, not to the POS tags, but to the characteristics of the words. Although we did not present any example of the learned rules, we would like to point out the advantages of representing the information in the typical classification rule format, when compared

to the numerical values used in the probabilistic approaches. The comprehensibility of the learned rules, which can be represented by predicate logic, allows its easy application in different contexts.

It is our conviction that the presented approach can be viewed as a new paradigm for solving a set of NLP tasks that share some of the features of the POS tagging problem and that are currently mainly solved by probabilistic approaches. Therefore, we are planning to extend this method to other tasks that also need some kind of disambiguation in the resolution process, like noun-phrase chunking, the named-entity recognition problem, sentiment analysis, etc.

References

[1] Brill, E.: Transformation-based error-driven learning and natural language processing: a case study in part-of-speech tagging. Comput. Linguist. 21, 543–565 (1995)
[2] Wilson, G., Heywood, M.: Use of a genetic algorithm in brill's transformation-based part-of-speech tagger. In: Proceedings of the 2005 Conference on Genetic and Evolutionary Computation, GECCO 2005, pp. 2067–2073. ACM, New York (2005)
[3] Nogueira Dos Santos, C., Milidiú, R.L., Rentería, R.P.: Portuguese part-of-speech tagging using entropy guided transformation learning. In: Teixeira, A., de Lima, V.L.S., de Oliveira, L.C., Quaresma, P. (eds.) PROPOR 2008. LNCS (LNAI), vol. 5190, pp. 143–152. Springer, Heidelberg (2008)
[4] Araujo, L.: Part-of-speech tagging with evolutionary algorithms. In: Gelbukh, A. (ed.) CICLing 2002. LNCS, vol. 2276, pp. 230–239. Springer, Heidelberg (2002)
[5] Alba, E., Luque, G., Araujo, L.: Natural language tagging with genetic algorithms. Inf. Process. Lett. 100(5), 173–182 (2006)
[6] Sousa, T., Neves, A., Silva, A.: Swarm optimisation as a new tool for data mining. In: Proceedings of the 17th International Symposium on Parallel and Distributed Processing, IPDPS 2003, p. 144. IEEE Computer Society, Washington, DC (2003)
[7] Sousa, T., Silva, A., Neves, A.: Particle swarm based data mining algorithms for classification tasks. Parallel Computing 30(5), 767–783 (2004)
[8] Silva, A.P., Silva, A., Rodrigues, I.: Biopos: Biologically inspired algorithms for pos tagging. In: Proceedings of the 1st International Workshop on Optimization Techniques for Human Language Technology, OPTHLT/COLING 2012, Mumbai, India, pp. 1–16 (December 2012)
[9] Silva, A.P., Silva, A., Rodrigues, I.: Tagging with disambiguation rules: A new evolutionary approach to the part-of-speech problem. In: Proceedings of the 4th International Conference on Evolutionary Computation Theory and Applications, IJCCI 2012, Barcelona, pp. 5–14 (2012)
[10] Steven Bird, E.K., Loper, E.: Natural Language Processing with Python. O'Reilly Media (2009)
[11] Kennedy, J., Eberhart, R.C.: Swarm intelligence. Morgan Kaufmann Publishers Inc., San Francisco (2001)
[12] Holland, J.: Genetic Algorithms. Scientific American 267(1) (July 1992)