A New Approach to the POS Tagging Problem Using Evolutionary Computation

Ana Paula Silva
EST-IPCB
dorian@ipcb.pt

Arlindo Silva
EST-IPCB
arlindo@ipcb.pt

Irene Rodrigues
Universidade de Évora
ipr@di.uevora.pt

Abstract

The purpose of part-of-speech tagging is to automatically tag the words of a text, written in a certain language, with labels that usually take the form of acronyms that designate the appropriate parts-of-speech. In this paper we propose a new approach to the problem that divides it in two different tasks: a learning task and an optimization task. We tackled each of those tasks with evolutionary computation techniques: genetic algorithms and a particle swarm optimizer. 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. We believe that this approach is generic enough so that it can be though as an alternative approach to solve other natural language processing tasks that share some fundamental characteristics with the part-of-speech tagging problem. The results obtained in two different English corpora are among the best published ones.

1 Introduction

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. Typically, 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 *fly* 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. 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 (Brill (1995); Wilson and Heywood (2005); Nogueira Dos Santos et al. (2008)). The rules are usually discovered automatically, and its purpose is to correct errors resulting from an initial basic tagging. Brill's tagger (Brill (1995)) 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 (Araujo (2002); Alba et al. (2006)), and rule-based information (Wilson and Heywood (2005)). 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 Araujo (2002) and Alba et al. (2006), 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 Wilson and Heywood (2005) the evolutionary algorithm is used to evolve the information model itself, in the form of a set of transformation rules.

Proceedings of Recent Advances in Natural Language Processing, pages 619–625,
Hissar, Bulgaria, 7-13 September 2013.

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.

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 (Sousa et al. (2004)), using evolutionary computation, has led us to define the classification algorithm based on a covering algorithm. 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 $Y = \{Yes, No\}$. The covering algorithm receives as input a set of positive examples and a set of negative examples. It then invokes the search algorithm with the current sets of examples. 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. Therefore, the complete set of rules is obtained by executing the search algorithm m times. Two different search algorithms were tested: one based on a GA and another based on a PSO.

2.1 Prediction attributes and representation

As prediction attributes we used two groups of information. 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 con-

sidered, 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

As we said in the previous section, we implemented the search algorithm in two different ways: using a genetic algorithm and a particle swarm optimizer. For the PSO based search algorithm we adopted the binary version presented by Kennedy (Kennedy and Eberhart (2001)). The genetic algorithm based version follows the classical GA with binary representation (Holland (1992)). 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 by the well known F_β-measure (see Equation 1). 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. Each time the search algorithm is invoked by the covering algorithm it returns the best rule found and a numerical value that represents the value of the F_β-measure to that rule. This value will be used as the quality value of the rule by the POS-Tagger, which we will present in the next section.

$$F_\beta(X) = (1 + \beta^2) \times \frac{P(X) \times R(X)}{\beta^2 \times P(X) + R(X)} \quad (1)$$

$$P(X) = \frac{TP}{TP + FP} \quad (2)$$

$$R(X) = \frac{TP}{TP + FN} \quad (3)$$

In Equation 3 TP represents the number of true positives, i.e. the number of instances covered by the rule that are correctly classified; FP represents the number of false positives, i.e. the number of instances covered by the rule that are wrongly classified; FN the number of false negatives, i.e. the number of instances not covered by the rule, whose class matches the training target class.

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.

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 par-

ticle 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 disambiguation rule's antecedent. 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 4.

$$h(\langle \mathbf{x}_i, t_i \rangle) = \begin{cases} q_k & \text{If } \langle r_k, q_k \rangle \in D_{t_i} \\ & \text{and } r_k \text{ covers } \mathbf{x}_i \\ 0 & \text{Otherwise} \end{cases} \quad (4)$$

The quality of a particle/individual is given by Equation 5, with T representing the set of all n pairs $\langle \mathbf{x}_i, t_i \rangle$.

$$Quality(T) = \sum_{j=1}^{n} h(T_j) \quad (5)$$

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 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 discovery

We used a simplified tag set, composed by 20 tags. This simplified tag set establishes the set of classes we use in our classification 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 time execution (Araujo (2002)). 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.

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

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%

Table 2: 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 AG 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**	$1.0978E-4$

evaluations the effort measure.

The results achieved are shown in table 2. 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 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. 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 3). 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 3 were achieved using all the corpus available in the NLTK package, in a total of 100676 words.

Table 3, presents the accuracy values achieved by the taggers in both English corpora used, 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 3 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.

5 Conclusions

We described a new evolutionary approach to the POS tagging problem, which we tested using two distinct algorithms from the evolutionary compu-

Table 3: Results achieved by the two taggers on two english corpora along with the ones published by similar approaches. (Araujo - Araujo (2002); Alba, Alba-GA, Alba-PGA, Alba - Alba et al. (2006); Wilson - Wilson and Heywood (2005); Brill - Brill (1995)).

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

tation 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 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 words' characteristics. Although we didn't 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

Alba, E., Luque, G., and Araujo, L. (2006). Natural language tagging with genetic algorithms. *Inf. Process. Lett.*, 100(5):173–182.

Araujo, L. (2002). Part-of-speech tagging with evolutionary algorithms. In Gelbukh, A., editor, *Computational Linguistics and Intelligent Text Processing*, volume 2276 of *Lecture Notes in Computer Science*, pages 187–203. Springer Berlin / Heidelberg.

Brill, E. (1995). Transformation-based error-driven learning and natural language processing: a case study in part-of-speech tagging. *Comput. Linguist.*, 21:543–565.

Holland, J. (1992). Genetic Algorithms. *Scientific American*, 267(1).

Kennedy, J. and Eberhart, R. C. (2001). *Swarm intelligence*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Nogueira Dos Santos, C., Milidiú, R. L., and Rentería, R. P. (2008). Portuguese part-of-speech tagging using entropy guided transformation learning. In *Proceedings of the 8th international conference on Computational Processing of the Portuguese Language*, PROPOR '08, pages 143–152, Berlin, Heidelberg. Springer-Verlag.

Sousa, T., Silva, A., and Neves, A. (2004). Particle swarm based data mining algorithms for classi-

fication tasks. *Parallel Computing*, 30(5):767–783.

Wilson, G. and Heywood, M. (2005). 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 '05, pages 2067–2073, New York, NY, USA. ACM.