BioPOS: Biologically Inspired Algorithms for POS Tagging

Ana Paula Silva[1] *Arlindo Silva*[1] *IreneRodrigues*[2]
(1) Polytechnic Institute of Castelo Branco, Portugal
(2) University of Evora, Portugal
`dorian@ipcb.pt, arlindo@ipcb.pt, ipr@uevora.pt`

ABSTRACT
In this paper we present a new biologically inspired approach to the part-of-speech tagging problem, based on particle swarm optimization. As far as we know this is the first attempt of solving this problem using swarm intelligence. We divided the part-of-speech problem into two subproblems. The first concerns the way of automatically extracting disambiguation rules from an annotated corpus. The second is related with how to apply these rules to perform the automatic tagging. We tackled both problems with particle swarm optimization. We tested our approach using two different corpora of English language and also a Portuguese corpus. The accuracy obtained on both languages is comparable to the best results previously published, including other evolutionary approaches.

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

1 Introduction

The part-of-speech (POS) tagging is a fundamental step for the execution of other natural language processing (NLP) tasks, like phrase chunking, parsing, named entity recognition, machine translation, information retrieval, speech recognition, etc. It is the process of classifying words according to the roles they assume in a sentence. The task is not straightforward because words in most languages can assume different roles in a sentence, depending on how they are used. Those roles are normally designated by part-of-speech tags or word classes, such as nouns, verbs, adjectives and adverbs.

The role of a word in a sentence is determined by it's surrounding words (context). For instance, the word *fish* can assume the function of a **verb**, "Men like to fish.", or a **noun**, "I like smoked fish", depending on how we choose to use it on a sentence. This means that in order to assign to each word of a sentence its correct tag, we have to consider the context in which each word appears. 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 we need some type of disambiguation mechanism.

Traditionally, there are two groups of methods used to tackle this task, with respect to the information model used. The first group is based on statistical data concerning the different context possibilities for a word (stochastic taggers) (Brants, 2000; Araujo, 2002, 2004, 2007; Araujo et al., 2004; Alba et al., 2006), while the second group is based on rules that capture the language properties and are used to improve tagging accuracy (Brill, 1995; Wilson and Heywood, 2005; Nogueira Dos Santos et al., 2008).

The simplest stochastic tagger, called unigram tagger, takes only into account the word itself. It assigns the tag that is most likely for one particular token. The tagger works like a simple lookup tagger, assigning to each word the most common tag for that word in the training corpus. To do that, the learning process just counts, for each word, the number of times it appears with each of the possible tags. An n-gram tagger is a generalization of a unigram tagger, whose context is the current word together with the part-of-speech tags of the n-1 preceding tokens. In this case, the training step saves, for each possible tag, the number of times it appears in every different context presented on the training corpus. Since the surrounding words can also have various possibilities of classification, it is necessary to use a statistical model that allows the selection of the best choices for marking the entire sequence, according to the model. Most of the stochastic taggers are based on hidden Markov models, and, because of that, a word's context consists only in the tags of the words that precede it. However, more recently, other taggers have emerged that, although based on stochastic information, have used different models that make possible to consider different context shapes.

One of the most popular taggers based on rules is the one proposed by Brill (Brill, 1995). Brill's rules are usually called transformation rules. The system can be divided into two main components: a list of transformation rules patterns for error correction, and a learning system. The transformation patterns are handmade and provided to the learning algorithm, which will instantiate and order them. The search is made in a greedy fashion. The result is an ordered set of transformation rules, which is then used to perform the tagging. These rules are meant to correct mistakes in a pre-tagged text, usually achieved by a baseline system that marks each word with its most common tag. They are applied in a iterative way until no rule can be fired.

As already observed, the only information an *n*-gram tagger considers from prior context is the tags, even though words themselves might be a useful source of information. It is simply impractical for *n*-gram models to be conditioned by the context words themselves (and not only their tags). On the other hand, Brill's approach allows the inclusion of other type of information besides the context. In fact, the author also used the learning algorithm to achieve a set of lexicalized transformation rules, that includes not only the tags but the words themselves.

There are also some other aspects that can be used to determine a word's category beside it's context in a sentence (Steven Bird and Loper, 2009). The internal structure of a word may give useful clues as to the word's class. For example, in the English language, *-ness* is a suffix that combines with an adjective to produce a noun, e.g., *happy* → *happiness*, *ill* → *illness*. Therefore, if we encounter a word that ends in *-ness*, it is very likely to be a noun. Similarly, *-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.

More recently, several evolutionary approaches have been proposed to solve the tagging problem. These approaches can also be divided by the type of information used to solve the problem, statistical information (Araujo, 2002, 2004, 2007; Araujo et al., 2004; Alba et al., 2006), and rule-based information (Wilson and Heywood, 2005).

Although there is a substantial amount of work about POS tagging applied to the English language, there are, comparatively, a small number of attempts to approach the problem using the Portuguese language. One of the main reasons is the limited number of resources that exist for it. Nevertheless, we can find some work where several of the existing techniques used to solve the POS tagging are tested on the Portuguese language: transformation based learning (Aires et al., 2000), Markov models (Kepler and Finger, 2006), entropy guided transformation learning (Nogueira Dos Santos et al., 2008).

One of the problems of the stochastic taggers, is that they tend to be dependent of the domain in which they are trained. Also, the information used is in the form of probabilistic values, which are less comprehensible than data presented in the form of rules, and only contemplates context information. In this work, we investigate the possibility of extracting, from an annotated corpus, a set of rules similar to classification rules, that could be used to help the decision process needed to perform the tagging. With our approach, we hope to be able to learn rules that prove to be more generic than the information used by the stochastic taggers, so that the tagger performs well in different domains. We also intend to include in these rules, together with the context information, other type of data, namely some aspects related with the word morphology. To accomplish our goals, we chose one technology that has already proven to be successful in data mining tasks, in particular in the discovery of classification rules (Sousa et al., 2004) - the particle swarm optimization (PSO) algorithm.

We also investigate the possibility of using a discrete PSO algorithm to perform the tagging, using the disambiguation rules to guide the evolution of the swarm. The problem of searching for the best tag assignments can be seen as a combinatorial optimization problem, where a solution is evaluated with the help of the disambiguation rules previously learned. We decided to test the application of swarm intelligence to this problem, since other population based algorithms, in particular genetic algorithms, have been successfully applied to many combinatorial optimization tasks. In order to evaluate the success of our approach we tested it on two different languages: English and Portuguese.

The remainder of this paper is organized as follows. In section 2, we describe the two evolutionary approaches to the POS tagging problem that we think are the most closely related to the work reported in this paper. In section 3, we present the general idea behind particle swarm optimization. The particle swarm optimization algorithm for disambiguation rules discovery is outlined in section 4 and the POS-Tagger in section 5. The experimental results are presented in section 6, and finally, in section 7, we make some concluding remarks.

2 Previous Work

In this section we present a brief description of the two approaches more closely related to our work, in that both of them use evolutionary algorithms to tackle some aspect of the part-of-speech problem. We begin with the part-of speech tagger developed by Wilson (Wilson and Heywood, 2005), and then we discuss the evolutionary tagger developed by Araujo (Araujo, 2002).

2.1 Wilson's Tagger

The system proposed in (Wilson and Heywood, 2005) is an evolutionary version of the Brill's tagging system (Brill, 1995) based on non lexical transformation rules. The authors propose a genetic algorithm to learn the rewrite rules, instead of the greedy algorithm described by Brill.

In the genetic algorithm presented, an individual of the population is a list of rewrite rules. The chosen representation was a fixed-length binary representation, in which each individual represents a set of 378 rules. The initial order of the rules is determined randomly. During the evolutionary process the order can be changed by the recombination operator.

Each rule is a sequence of 48 bits.The initial population is randomly generated. Each individual represents an ordered collection of rules that potentially solves the problem of marking the Wall Street Journal corpus. Each rule takes in consideration the tags of the three words to the left and to the right of the current word. An extra bit is used for each of the context elements, to indicate if that element should be considered in the rule. Thus the leftmost bit in the string indicates whether the triggering environment involves examining the third tag back from the target tag position. The next six bits indicate what tag should be sought in that position by using the integer equivalent of the binary representation.

The six bits map onto a 45 part-of-speech tags from the Penn Treebank, and if the integer equivalent of the six bits exceeds 45, the tag is considered to be the remainder of the integer division by 45. The next bit indicates whether to examine the second tag back. What tag to look for there is indicated by the next six bits. The same interpretation is made for the next seven bits, considering the tag of the previous word. The next six bits indicate what tag is to replace that of the target position. The tags sough (and wether or not they should be sough) in the three positions following the target are given by the following 21 bits. The interpretation is the same as that described for the positions previous to the target position.

The performance of an individual is given by the accuracy of the tagging achieved with the set of rules it represents. The experimental conditions used by Wilson attempt to follow the ones used by Brill nonlexicalized tagger. They used only nonlexicalized rules and each individual is composed by 378 rules. In each rule the tags located 3 positions previous to the target and 3 positions following the target position, are examined. They used a subset of the Pen Treebank corpus, composed of 600000 words to learn the rules. A closed vocabulary assumption was adopted. The best result achieved in the training corpus obtained an accuracy of 89.8%. There is no reference to experiments made in a separate test set with the best set of rules evolved by the evolutionary algorithm.

2.2 Araujo's Tagger

The system proposed in (Araujo, 2002) uses an evolutionary algorithm to perform the tagging. In order to tag the words of a text, the algorithm must be run separately for each sentence of the text. In the proposed algorithm, each individual in the population represents a candidate solution to the tagging problem. Thus, each individual comprises a chromosome composed by as many genes as the number of words of the sentence to tag. Each gene refers to the sentence word in the same position, i.e. the gene g_i contains information concerning the word, w_i of the sentence to tag. This information includes a candidate tag for the word, and data related with the context of that part-of-speech. The context information is obtained in an initial step and is stored as a table - the training table. For each part-of-speech of the training corpus the number of times each of one appears in every possible context was counted. The authors considered contexts that include tags appearing before and after a certain part-of-speech. Experiments with different sizes of context were coducted.

The performance of a particular chromosome was measured using a fitness function defined as the sum of the evaluation of each of the genes that compose the chromosome. A gene is evaluated based on the statistical information collected and stored in the training table. The value obtained is an estimative of the accuracy of the tag proposed by the gene for a particular word in the sentence.

The evolutionary process aims to choose, for a particular sentence, the sequence of tags that maximize the total probability, based on the statistical data obtained previously. The authors conclude that the corpus used for training the system has a great influence on the algorithm performance, and state that a corpus which proves to be a good sample of the language should be used. However, that is not always possible, since in most cases the corpora available are specific to a particular domain.

The results presented were achieved with the Brown corpus. The best results presented, using a test set of 2500 words, were below 95.5%.

3 Particle Swarm Optimization

PSO is inspired in the intelligent behavior of agents as part of an experience sharing community, as opposed to an isolated individual reactive response to the environment. The adaptive culture model (Kennedy and Eberhart, 2001), which is PSO's framing theory, states that the process of cultural adaptation is rooted into three principles: evaluate, compare and imitate. Evaluation is the capacity to qualify environmental stimuli and the *sine qua non* condition to social learning. Evaluation itself is both useless and impossible without the ability to compare; all of our metrics are but a comparison to a well-known unit and a single value becomes pointless without the values of it peers. At last, imitation is the rawest form of experience sharing from the receiver's standpoint; it involves not only observation but also realization of purpose and timing adequacy.

In PSO algorithms, a particle decides where to move next, considering its own experience, which is the memory of its best past position, and the experience of its most successful neighbor. There may be different concepts and values for neighborhood; it can be seen as spatial neighborhood where it is determined by the Euclidean distance between the positions of two particles, or as a sociometric neighborhood (e.g.: the index position in the storing array). Although some actions differ from one variant of PSO to the other, the common pseudo-code for PSO is as follows:

The output of this algorithm is the best point in the hyperspace the swarm visited. Since its introduction, the PSO algorithm has been successfully applied to problems in different areas,

Algorithm 1 Generic Particle Swarm Optimization Algorithm

```
Initiate_Swarm()
repeat
  for p= 1 to number of particles do
    Evaluate(p)
    Update_past_experience(p)
    Update_neighborhood_best(p,k)
    for d= 1 to number of Dimensions do
      Move(p,d)
    end for
  end for
until Criterion
```

including antenna design, computer graphics visualization, biomedical applications, design of electrical networks, and many others (Poli, 2008). Amongst the qualities that led to this popularity are its conceptual simplicity, ease of implementation and low computational costs. The algorithm works well with small swarms and tends to converge fast to a solution. In addition, particle swarm optimization has proven itself to be easily adaptable to new domains of application and to work well when hybridized with other approaches. There are several variants of PSO, including algorithms for discrete and continuous domains. The variant used in our work was the discrete PSO introduced by Kennedy (Kennedy and Eberhart, 2001).

3.1 Discrete Swarm Optimization Algorithm

This variant of the PSO considers each bit as a dimension, with 2 possible values 0 or 1. Particle motion is just the toggling between these two values.

There is a velocity value(V_{id}) associated with each dimension/bit; this vale is randomly generated from a range of $[-4.0, 4.0]$ when the particle is created and iteratively updated (1) according to his previous best position (P_{id}) and the best position in the neighborhood (P_{gd}). φ_1 and φ_2 are random weights whose role is to provide diversity between individual learning and social influence.

$$V_{id}(t+1) = V_{id}(t-1) + \varphi_1(P_{id} - x_{id}(t-1)) + \varphi_2(P_{gd} - xi, d(t-1)) \tag{1}$$

To determine if a bit will toggle (2), a random number, ρ, is drawn from a uniform distribution ranging from 0 to 1, and is compared to a normalized value of the velocity associated with this dimension.

$$x_{i,d} = \begin{cases} 1 & \text{if } \rho < S(v_{id}(t)) \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

The sigmoid function (3) is used here to insure that the velocity's value is kept in the range of $[0.0, 1.0]$.

$$S(v_{id}) = \frac{1}{1 + exp(-v_{id})} \tag{3}$$

4 Particle Swarm Optimization Algorithm for Disambiguation Rules Discovery

In this section we describe the use of a discrete PSO (DPSO) algorithm to discover a set of disambiguation rules, that will be used to help the optimization process necessary to solve the part-of-speech tagging problem. We will approach the problem as if we were trying to solve a classification problem, by learning a set of classification rules.

The overall structure of our approach was designed to include two nested algorithms. The innermost algorithm is the classification rule discovery algorithm. Its task is to find and return the rule which better classifies the training examples. As we said before we adopted a DPSO to perform this task. The outmost algorithm is a covering algorithm that receives the training set, divided in two sets - the set of positive examples and the set of negative examples. It then invokes the classification rule discovery algorithm to reduce this set by removing instances of the set of positive examples, that are correctly classified by the rule returned by the classification rule discovery algorithm. This process is repeated until there are no examples to classify (see algorithm 2).

Algorithm 2 Covering Algorithm

```
Require: SetOfPosExamples, SetOfNegExamples
Ensure: SetOfRules
  while SetOfPosExamples $\neq \emptyset$ do
    BestRule = DPSO(SetOfPosExamples, SetOfNegExamples)
    SetOfPosExamples = RemoveExamples(SetOfPosExamples, BestRule)
    SetOfRules = Add(SetOfRules, BestRule)
  end while
```

4.1 Rule Representation

Classification rules are no more than conditional clauses, involving two parts: the antecedent and the consequent. The former is the conjunction of logical tests, and the latter gives the class that applies to instances covered by this rule. These rules take the following format:
`IF` $attrib_a$ = val_1 `AND` $attrib_b$ = val_2 ..... `AND` $attrib_n$ = val_i `THEN` $class_x$.

In evolutionary rule classifier systems there are two distinct approaches to individual or particle representation: the Michigan and the Pittsburgh approaches (Freitas, 2003). In the Michigan approach each individual encodes a single rule, whereas in the Pittsburgh approach each individual encodes a set of rules. In our work, we follow the Michigan approach, and rules are coded using binary strings; each attribute is assigned with the necessary bit number to accommodate its value range. An extra bit is used to represent any value, meaning that no logical test is performed to this attribute when this bit is activated. As we stated in Introduction, our aim is to discover a set of rules that take into consideration not only context information but also information about the words' morphology.

We decided to test our approach on the English and Portuguese languages. Thereby, since the structure of the languages are different, we needed to extract disambiguation rules from a English and a Portuguese corpora. We adopted a slightly different type of rules for the two languages. For the context, we decided to consider the same information that was used in the work of Brill (Brill, 1995) and Wilson (Wilson and Heywood, 2005). Thus, we consider six attributes: 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. These attributes were used for the rules of both languages. For the words' morphology information we decided to include, for the English language as well as for the Portuguese language, the following attributes: 'The word is capitalized?', 'The word is the first word of the sentence?'. In addition, for the English language, we incorporated on the rules' antecedent these attributes: 'The word ends with ***ed***?' 'The word ends with ***ing***?', 'The word ends with ***es***?' , 'The word ends with ***ould***?' , 'The word ends with ***'s***?', 'The word ends with ***s***?', 'The word has numbers or '.' and numbers?'.

The possible values for each of the first six attributes are the values of the corpus tag set from which the evolutionary algorithm will extract the rules. This set will depend on the annotated corpus used, since the set of used labels will vary for different annotated corpora. The maximum number of tags on the tag sets we used was 29, so we used six bits to represent each attribute. 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 n entries, to store the tag set, with n representing the cardinality of the set, and used the binary value represented by five bits to index this table. If the value exceeds the number n, we used the remainder of the division by n.

The remaining k attributes were encoded by $2 \times k$ bits, two bits for each of the k attributes. In the same way, the first bit indicates if the attribute should, or shouldn't be considered and the second bit, indicates whether the property is, or is not, present. In short, for the English language each particle was composed by $6 \times 6 + 2 \times 9 = 54$ bits, and for the Portuguese language each particle was made by $6 \times 6 + 2 \times 2 = 40$ bits.

In general, there are three different ways to represent the predicted class in an evolutionary algorithm (Freitas, 2003). One way is to encode it into the genome of the individual, or the particle, opening the possibility of subjecting it to evolution (de Jong et al., 1993; Greene and Smith, 1993). Another way is to associate all individuals of the population to the same class, which is never modified during the execution of the algorithm. Thus, if we are to find a set of classification rules to predict k distinct classes, we need to run the evolutionary algorithm, at least k times. In each i-th execution, the algorithm only discovers rules that predict the i-th class (Janikow, 1993). The third possibility consists in choosing the predicted class in a more or less deterministic way. The chosen class may be the one that has more representatives in the set of examples that satisfy the antecedent of the rule (Giordana and Neri, 1995), or the class that maximizes the performance of the individual (Noda et al., 1999) . We adopted the second possibility, so we didn't need to encode the rule's consequent. This choice determines that the covering algorithm needs to be ran for each tag of the tag set adopted for the experimental work.

4.2 Rule Evaluation - Establishing Reference Points

Rules must be evaluated during the training process in order to establish points of reference for the training algorithm. The rule evaluation function must not only consider instances correctly classified, but also the ones left to classify and the wrongly classified ones. The formula used to evaluate a rule, and therefore to set its quality, is expressed in equation 4. This formula penalizes a particle 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. 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.

$$Q(X) = \begin{cases} F_\beta(X) & \text{if context(X) = True} \\ -1 & \text{otherwise} \end{cases} \tag{4}$$

$$context(X) = \begin{cases} True & \text{if } X \text{ tests at least one of the first six attributes} \\ False & \text{otherwise} \end{cases} \tag{5}$$

$$F_\beta(X) = (1+\beta^2) \times \frac{precision(X) \times recall(X)}{\beta^2 \times precison(X) + recall(X)} \tag{6}$$

$$precision(X) = \frac{TP}{TP + FP} \tag{7}$$

$$recall(X) = \frac{TP}{TP + FN} \tag{8}$$

where:

- TP - True Positives = number of instances covered by the rule that are correctly classified, i.e., its class matches the training target class;
- FP - False Positives = number of instances covered by the rule that are wrongly classified, i.e., its class differs from the training target class;
- FN - False Negatives = number of instances not covered by the rule, whose class matches the training target class.

4.3 Pre-processing Routines - Data Extraction

For the English language we used the Brown corpus to create the training set that we provided as input to the discrete PSO for the extraction of the disambiguation rules. The corpus used for the Portuguese language was the Mac-Morpho. For each word of both corpus we collected the values for every attribute included in the rule's antecedent, creating a specific training example. Then, for each tag of the tag set, we built a training set composed by positive and negative examples of the tag.

Usually, the set of positive (negative) examples of a class is composed by examples that do (do not) belong to that particular class. However, in our case, we are not interested in finding typical classification rules, our goal is not to solve a classification problem, we just need rules that help to choose the best tag from a set of possible tags. This set is not all the tag set, but a subset of it, usually composed by a small number of elements. When we have a word that has only one possible lexical class, the tagging is straightforward. The problematic words are the ones that are ambiguous. Thus, our training set should only include examples corresponding to ambiguous words. 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.

4.4 Experimental Work

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, for the English language, among others, the Brown corpus and a sample of 10% of the Wall Street Journal (WSJ) corpus of the Penn Treebank. These corpora are the most frequently used to test taggers' performances on the English language and the ones used in the approaches we mentioned earlier. The Mac-Morpho corpus is also available for the Portuguese language.

4.4.1 English Language

The corpus used for the extraction of the disambiguation rules for the English language was the Brown corpus. Tagged corpora use many different conventions for tagging words. This means that the tag sets vary from corpus to corpus. Besides, our approach determines that, to extract the disambiguation rules from a set of annotated texts, we need to run our algorithm for each of the tags belonging to the tag set. However, our intention was to test the rules extracted from the Brown corpus on a different corpus of the same language, namely the WSJ corpus. Since these corpora have different tag sets we will not be able to measure the performance of our tagger on the WSJ corpus with the rules found on the Brown corpus. To avoid this, we decided to use 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 Brown and WSJ corpora to a uniform simplified tag set, composed by 29 tags. This simplified tag set establishes the set of classes we use in our algorithm for the English language. We ran the covering algorithm for each one of the classes that had ambiguous words in the brown corpus.

We processed the Brown corpus and, for each word found, we built the correspondent instance. The total number of examples extracted from the corpus equals 929286. We used 5 subsets of this set (with different cardinality) to conduct our experiments We used sets of size: 30000, 40000, 50000, 60000, 70000. The examples were taken from the beginning of the corpus. For each subset adopted, we built the sets of positive and negative examples for each ambiguous tag, using the process described in the previous section. We used a compacted representation of the examples: for each different example we saved the number of times it appears in the set

The discrete PSO algorithm was run with a swarm of 20 particles for a maximum of 200 generations. These values were established after some preliminary experiments. We ran the algorithm 4 times for each of the sets, and tested the resulting rules by providing them to the PSO-Tagger, which we will describe next. The set that produced the best results on the PSO-Tagger was the one with 50000 tokens. The discovery algorithm found a total number of 3385 rules for this set.

4.4.2 Portuguese Language

Like we said before, we used the Mac-Morpho corpus to extract the disambiguation rules for the Portuguese language. This corpus contains 1.2 million manually tagged words. Its tag set contains 22 POS tags and 10 more tags that represent additional semantics aspects. We carried out our experiments using only the 22 POS tags.

We processed the Mac-Morpho corpus in much the same way we processed the Brown corpus. But in this case we used the first 60% of the Mac-Morpho sentences to build the examples used to define the sets of positive and negative examples for each of the tags of the tag set used in this corpus. A compacted representation was also used. Notwithstanding, to reduce the number of examples of each set, we eliminated the ones that only appeared one, two or three times, depending on the cardinality of the positive and negative sets. Between positive and negative examples, we considered a total of 107200 different examples . We ran the discrete PSO algorithm with 20 particles during 200 generations. The set of rules found that provided the best tagging had 5988 rules.

5 PSO-Tagger

The PSO-Tagger was designed to receive as inputs a sentence, a set of disambiguation rules and a dictionary. The returned output is the input sentence with each of its words marked with the correct POS tag. The discrete PSO algorithm evolves a swarm of particles, that encode, each of them, a sequence of tags to mark the ambiguous words of the input sentence. Since we adopted the discrete version of the PSO algorithm, we used a binary representation for the particles. To encode each of the tags belonging to the tag set, used in the experimental work, 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.

5.1 Representation

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. As we said before, we intend to use the disambiguation rules, found by the discovery algorithm described in the previous section, to guide the evolution of the swarm particles. Since these rules have six attributes related with the word context, and other k attributes concerning morphological properties of the words, we need to build, from the input sentence and from the tags proposed by the particles, the values of each one of the attributes contemplated in the rules' antecedents.

Although the particles only propose tags for the ambiguous words, the tags of the unambiguous ones will be needed to extract the attributes' values. Thus, before optimization begins, the discrete PSO marks each of these words with the correspondent tag, by simple input dictionary lookup. So a particle completes the previous lookup based tagging, and provides a full marked sentence. This sentence can then be used to extract a set of instances composed by the $6 + k$ attributes, so that the disambiguation rules can be applied. This way, each pair w_i/t_i in the full annotated sentence results in a $(6 + k + 1)$-tuple made by the $6 + k$ attributes and by t_i. The English disambiguation rules had $k = 9$, and the Portuguese disambiguation rules $k = 2$. When there is no word in one of the positions contemplated in the context, we adopted the use of an extra tag named 'None'.

5.2 Particles' Evaluation

The quality of a particle is given by the tagging quality of the full input sentence. To evaluate the tagging of the sentence, we use the disambiguated rules to measure the quality of each instance extracted from the sentence. The quality of the overall tagging is given by the sum of the evaluation results for each instance. Let's consider t_i to be the class presented in the last position of the 16-tuple of instance $instance_k$. If R_{t_i} represents the set of disambiguation rules for the lexical category t_i, and $r_k \in R_{t_i}$ a disambiguation rule that covers the instance $instance_k$, then the quality value of $instance_k$ is given by the quality measure Q_k associated with rule r_k.

$$F(instance_k) = \begin{cases} Quality(r_k) & \text{if } r_k \text{ is found} \\ 0 & \text{otherwise} \end{cases} \quad (9)$$

If S_p is the set of all instances extracted from the annotated sentence determined by the particle p, the quality of particle p is given by

$$Fitness(p) = \sum_{i \in S_p} F(i) \quad (10)$$

Although a particle only suggests tags for the ambiguous words in the sentence, the quality of the instances defined by the unambiguous words is affected by the tags established by the particle. Therefore, the overall tagging evaluation should also consider these instances.

6 Experimental Results

We tested our PSO-Tagger on two different languages: English and Portuguese. For the English language we carry out experiments on two different corpora: in a test set of the WSJ corpus of the Penn Treebank made of 8300 tokens, and on a test set of 22562 tokens of the Brown corpus. In both cases the POS-Tagger received as input the set of disambiguation rules learned from the Brown corpus.

For the Portuguese language, we tested the tagger on a test set of the Mac-Morpho corpus with 21466 tokens. In this case the algorithm received as input the set of disambiguation rules learned from the Mac-Morpho training set.

We ran the algorithm 20 times with a swarm of 10 and 20 particles during 50 and 100 generations for the three corpora. The results achieved are shown in table 1. As we can see, the best average accuracy on the WSJ corpus was 96.91% and the best average accuracy on the Brown corpus was 96.72%. The best results on the Brown corpus were achieved with a swarm of 20 particles over 50 generations. A maximum accuracy of 96.75% was found. A swarm of 20 particles over 100 generations gave the best results for the WSJ corpus. In this case, the best tagging found had an accuracy of 97.04%.

For the Portuguese corpus, the best average accuracy obtained was 96.83%, when the POS-tagger was executed with 10 particles over 100 generations. The best accuracy, 96.89%, was found during a run of the algorithm with 20 particles over 50 generations.

Both results allow us to conclude that the PSO-Tagger usually finds a solution very quickly. Like we said before, this algorithm works well with small swarms and tends to converge fast to a solution. Table 2 presents the best results achieved by the approaches we mentioned earlier, along with the best ones achieved by the PSO-Tagger. Observing table 2, we can see that the PSO-Tagger accuracy is very promising, since it is among the best values presented.

Naturally, the difficulty level of the tagging task depends on the number of ambiguous words of the sentence we want to tag. Although it is possible to construct sentences in which every word is ambiguous (Hindle, 1989), such as the following: "Her hand had come to rest on that very book."; those situations are not the most common. After counting the number of ambiguous words that appear in the sentences of the 10% of the Brown corpus we reserved for testing the tagger, we observed that, in average, there are 6.9 ambiguous words per sentence. This

Corpus	Particles	Generations	Average	Standard Deviation	Best
Brown	10	50	96.68	0.022	96.72
		100	96.67	0.028	96.72
	20	50	**96.7**	0.024	**96.75**
		100	96.69	0.024	96.73
WSJ	10	50	96.9	0.054	96.99
		100	**96.91**	0.054	**97.04**
	20	50	96.88	0.053	96.99
		100	96.91	0.031	96.98
Mac-Morpho	10	50	96.82	0.029	96.86
		100	**96.83**	0.029	96.86
	20	50	96.82	0.033	**96.89**
		100	96.81	0.026	96.86

Table 1: Results achieved on the English and Portuguese corpora by the PSO-Tagger after 20 runs with a swarm of size 10 and 20, during 50 and 100 generations.

explain the considerable low number of particles and generations needed to achieve a solution. We could argue that in those conditions the use of a PSO algorithm is unnecessary, and that a exhaustive search could be applied to solve the problem. However, we can not ignore the worst case scenario, where, like we see above, all the words, or a large majority of the words, on a very long sentence may be ambiguous. Furthermore, we observed that the sentence average size of the Brown corpus is of 20.25 tokens, with a maximum of 180. The largest number of ambiguous words on a sentence belonging to this corpus is 68. Even for the smallest degree of ambiguity, with only two possible tags for each word, we have a search space of 2^{68}, which fully justifies the use of a global search algorithm such as a PSO.

The results achieved show that there are no significant differences on the accuracy obtained by the tagger on the two test sets of the English language. At this point, it is important to emphasize that the disambiguation rules used on the tagger were extracted from a subset (different from the test set used in this experiments) of the Brown corpus. Which bring us to the conclusion that the learned rules are generic enough to be used on different corpora, and are not domain dependent. The results achieved for the Portuguese language showed that our strategy also performed well for other languages different from English.

7 Conclusions

We described a new evolutionary approach to the POS tagging problem that achieved competitive results when compared to the best ones published (see table 2). Although there are other approaches to this task based on evolutionary algorithms, in particular genetic algorithms, as far as we know this is the first attempt that uses a PSO algorithm to tackle the POS problem. Our method also differs from previous ones on the information model used to guide the evolutionary process. More specifically, in this work, we used a set of disambiguation rules, including morphological information, in opposition to the stochastic data that is usually adopted in the evolutionary approaches we have found in the literature. We believe that this approach brings a important level of generalization to the model, which results in good tagging performances even when applied to other corpora. Beside the traditional experiments made on English corpora,

Corpus	Tagger	Training Set	Test Set	Average	Best
Brown	PSO-Tagger	50000	22562	96.7	**96.75**
	(Araujo, 2002)	185000	2500	-	95.4
	GA (Alba et al., 2006)	165276	17303	96.37	96.67
	PGA (Alba et al., 2006)	165276	17303	96.61	96.75
WSJ	PSO-Tagger	none	8300	96.91	**97.04**
	(Wilson and Heywood, 2005)	600000	none	-	89.8
	(Brill, 1995)	600000	150000	-	97.2
	(Alba et al., 2006)	554923	2544	-	96.63
Mac-Morpho	PSO-Tagger	107200	21466	96.83	**96.86**
	ETL-Tagger	1M	200K	-	96.75

Table 2: Results achieved by the PSO-Tagger on corpora of English and Portuguese language, along with the results achieved by the approaches more similar to the one presented here. The ETL-Tagger are the one presented in (Nogueira Dos Santos et al., 2008)

we also tested our strategy on a different language, namely the Portuguese language. The results attained showed that the tagger also achieves competitive accuracy when applied to the Portuguese language.

The PSO-Tagger proved to be capable of tackling the combinatorial optimization problem, performing the tagging task with good results, while using limited resources in terms of swarm size and number of generations. Although we consider our results to be promising, we are aware of the necessity of evaluating our approach with a larger tag set and of applying it to more corpora. We also believe that the algorithms' parameters could be better tuned, and for that we intend to conduct a larger set of experiments. Likewise, we think that further experiments, with different sets of attributes for the disambiguation rules, should be conducted. Finally, we think that the overall evolutionary approach could be successfully applied to other disambiguation problems, like the phrase chunking and named-entity recognition problems.

References

Aires, R. V. X., Aluísio, S. M., Kuhn, D. C. e. S., Andreeta, M. L. B., and Oliveira, Jr., O. N. (2000). Combining classifiers to improve part of speech tagging: A case study for brazilian portuguese. In *International Joint Conference, 7th Ibero-American Conference, 15th Brazilian Symposium on AI, IBERAMIA-SBIA 2000, Open Discussion Track Proceedings on AI*, pages 227–236. ICMC/USP.

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.

Araujo, L. (2004). Symbiosis of evolutionary techniques and statistical natural language processing. *Evolutionary Computation, IEEE Transactions on*, 8(1):14 – 27.

Araujo, L. (2007). How evolutionary algorithms are applied to statistical natural language processing. *Artificial Intelligence Review*, 28(4):275–303.

Araujo, L., Luque, G., and Alba, E. (2004). Metaheuristics for natural language tagging. In *Genetic and Evolutionary Computation - GECCO 2004, Genetic and Evolutionary Computation Conference*, volume 3102 of *Lecture Notes in Computer Science*, pages 889–900. Springer.

Brants, T. (2000). Tnt: a statistical part-of-speech tagger. In *Proceedings of the sixth conference on Applied natural language processing*, ANLC '00, pages 224–231, Stroudsburg, PA, USA. Association for Computational Linguistics.

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.

de Jong, K. A., Spears, W. M., and Gordon, D. F. (1993). Using genetic algorithms for concept learning. *Machine Learning*, 13:161–188. 10.1023/A:1022617912649.

Freitas, A. A. (2003). *A survey of evolutionary algorithms for data mining and knowledge discovery*, pages 819–845. Springer-Verlag New York, Inc., New York, NY, USA.

Giordana, A. and Neri, F. (1995). Search-intensive concept induction. *Evol. Comput.*, 3:375–416.

Greene, D. P. and Smith, S. F. (1993). Competition-based induction of decision models from examples. *Machine Learning*, 13:229–257.

Hindle, D. (1989). Acquiring disambiguation rules from text.

Janikow, C. Z. (1993). A knowledge-intensive genetic algorithm for supervised learning. *Machine Learning*, 13:189–228. 10.1007/BF00993043.

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

Kepler, F. and Finger, M. (2006). Comparing two markov methods for part-of-speech tagging of portuguese. In Sichman, J., Coelho, H., and Rezende, S., editors, *Advances in Artificial Intelligence - IBERAMIA-SBIA 2006*, volume 4140 of *Lecture Notes in Computer Science*, pages 482–491. Springer Berlin / Heidelberg. 10.1007/11874850_52.

Noda, E., Freitas, A., and Lopes, H. (1999). Discovering interesting prediction rules with a genetic algorithm. In *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, volume 2, pages 3 vol. (xxxvii+2348).

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.

Poli, R. (2008). Analysis of the publications on the applications of particle swarm optimisation. *J. Artif. Evol. App.*, 2008:4:1–4:10.

Sousa, T., Silva, A., and Neves, A. (2004). Particle swarm based data mining algorithms for classification tasks. *Parallel Comput.*, 30(5-6):767–783.

Steven Bird, E. K. and Loper, E. (2009). *Natural Language Processing with Python*. O'Reilly Media.

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.