Part-of-Speech Tagging with Evolutionary Algorithms

Lourdes Araujo

Dpto. Sistemas Informáticos y Programación.
Universidad Complutense de Madrid.
lurdes@sip.ucm.es

Abstract. This paper presents a part-of-speech tagger based on a genetic algorithm which, after the "evolution" of a population of sequences of tags for the words in the text, selects the best individual as solution. The paper describes the main issues arising in the algorithm, such as the chromosome representation and the evaluation and design of genetic operators for crossover and mutation. A probabilistic model, based on the context of each word (the tags of the surrounding words) has been devised in order to define the fitness function. The model has been implemented and different issues have been investigated: size of the training corpus, effect of the context size, and parameters of the evolutionary algorithm, such as population size and crossover and mutation rates. The accuracy obtained with this method is comparable to that of other probabilistic approaches, but evolutionary algorithms are more efficient in obtaining the results.

1 Introduction

The process of labeling each word in a sentence with its part-of-speech (noun, verb, etc) is called tagging and is a key step in the parsing process. Part-of-speech tagging is used in many language processing and generation applications: parsing, machine translation, information retrieval, speech recognition and generation, etc. The research in automatic part-of-speech tagging has increased a lot in the last years, probably due to the increasing availability of large tagged corpora.

Because languages are ambiguous and a word may have more than one tag, disambiguation methods are required to proceed with the tagging. There are different approaches for the disambiguation task, which can be classified in statistical [Jel85,DeM90,Mer94,CKPS92,SS94,Cha93] and rule-based[Qui93,Bri97]. Statistical taggers use large amount of data to establish the probabilities of each situation and neither require knowledge of the rules of the language nor try to deduce them. Most of these systems are based on Hidden Markov Models. Rule-based approaches apply language rules to improve the accuracy of the tagging. The Brill system [Bri95] extracts these rules from a training corpus, obtaining competitive performance with stochastic taggers.

The model presented in this paper belongs to the stochastic approach. In this model disambiguation is introduced by assigning different probabilities to a given tag depending on which are the neighbouring tags (context). The second important feature of the model is the use of an evolutionary algorithm to find the tagging of new sentences. Evolutionary algorithms are among the most efficient methods to deal with complex optimization problems, and can improve the performance of the typical algorithms used for the same purpose in other stochastic tagging approaches (such as the widely used of Viterbi).

Evolutionary algorithms mimic the principles of natural evolution: heredity and survival of the fittest individuals. Systems based on evolutionary algorithms maintain a population of potential solutions, and are provided with some selection process based on the fitness of individuals. The population is renewed by replacing individuals with those obtained by applying "genetic" operators to selected individuals. The usual "genetic" operators are *crossover* and *mutation*. Crossover obtains new individuals by mixing, in some problem dependent way, two individuals, called parents. Mutation gives a new individual by performing some kind of change on an individual. The production of new generations continues until resources are exhausted or until some individual in the population is fit enough. Evolutionary algorithms have been shown to be practical search and optimization methods, applied in diverse areas, such as planning or machine learning [Mic94]. They have also been applied to different issues of natural language processing, such as query translation [MD96], inference of context-free grammars [Wya91,TS95] and parsing [Ara00].

This work describes the evolutionary algorithm designed for the tagging task. In this algorithm, individuals are sequences of tags assigned to the words of a sentence. The computation of the fitness of the individuals, what decides their opportunities of surviving for the next generation, is based on the data extracted of a training corpus tagged by hand. These data are organized as *contexts*. The fitness function considers contexts whose length varies when necessary to increase the precision of the algorithm.

The rest of the paper proceeds as follows: Section 2 presents the evolutionary algorithm for tagging, including each genetic operator; Section 3 describes and discusses the experimental results, and Section 4 draws the main conclusions of this work.

2 Evolutionary Tagging

The evolutionary tagger is able to learn from a training corpus so as to produce a table of rules (contexts) called *training table*. Chromosome evaluation is done according to the training table. This table records the different contexts of each tag. The table can be computed by going through the training text and recording the different contexts and the number of occurrences of each of them for every tag in the training text.

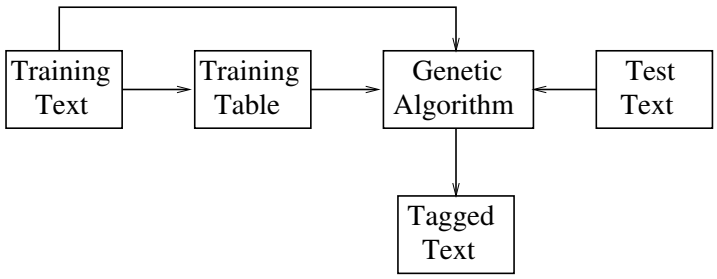

Fig. 1. Evolutionary Algorithm Training Scheme.

Furthermore, the training corpus used to extract this stochastic information, can also be used here to tune the parameters of the evolutionary algorithm in a automatic way (Figure 1 shows a scheme of the process).

The evolution process is run for each sentence in the text to be tagged. Evolution steps aim to maximize the total probability of the tagging of the sentences in the test corpus. The process finishes either if the fitness deviation lies below a threshold value (convergence) or if the evolutionary process has been running for a maximum number of generations. Let us consider the different elements of the algorithm.

2.1 Chromosome Representation

Chromosomes are sequences of genes which correspond to each word in the sentence to be tagged. Each gene is composed of a tag and additional information useful in the evaluation of the chromosome, such as counts of contexts for this tag according to the training table.

Initial Population For a given sentence of the test corpus, the chromosomes forming the initial population are created by randomly selecting from a dictionary one of the valid tags for each word, with a bias to the most probable tag. Words not appearing in the dictionary are assigned the tag which appears more often with that given context in the training text.

2.2 Fitness: Chromosome Evaluation

The fitness function is a critical point in the evolutionary algorithm design, since it determines the quality of the solutions. In our case this function is directly related to the objective to be maximized, the total probability of each solution or chromosome. The raw data to obtain this probability are extracted from the training table. The fitness of each chromosome is evaluated according to the following points:

- Evaluation goes from left to right.

- Let n be the number of words in the sentence. Let w_i be the word in position i of the sentence. Let $f(g_i)$ be the estimated accuracy, or fitness, of the tag in position i or gene g_i. The fitness or measure of the adaptation of a chromosome is computed according to the formula

$$\sum_{i=1}^{n} f(g_i)$$

- In order to compute the fitness, each chromosome position or *gene* is previously evaluated according to the training table, so as to obtain an estimation of its accuracy. This is done in the following way:
  - Let $\mathcal{T}$ be the set of possible tags for the word at the position to be evaluated, i.
  - We are considering contexts of the following form:

$$LC(T_{l_1}, \cdots, T_{l_{l_{LC}}}), T, RC(T_{r_1}, \cdots, T_{r_{l_{RC}}})$$

    where $T \in \mathcal{T}$ is the tag currently assigned to the word at the position being evaluated, LC represents the left part of context, with length l_{LC}, and composed of the tags $T_{l1}, \cdots, T_{l_{l_{LC}}}$, and RC represents the right part of the context, with length l_{RC} and tags $T_{r_1}, \cdots, T_{r_{l_{RC}}}$. Let occ_i be the number of occurrences of this context in the training text, which can be directly obtained from the training table.
    Let sum_i be the sum of all the occurrences of any context of the form:

$$LC(T_{l_1}, \cdots, T_{l_{LC}}), T', RC(T_{r_1}, \cdots, T_{r_{RC}})$$

    $\forall T' \in \mathcal{T}$, which can be computed from the training table.
    Then, the adaptation or fitness of the gene at position i, $f(g_i)$, is computed as:

$$f(g_i) = log(\frac{occ_i}{sum_i})$$

  - If the context does not correspond to any entry of the training table for this tag, then the context size is reduced by one tag and the counts for the new context are calculated from the table by adding all the entries in which the new context is present.
  - If even the shortest context does not appear in the table, then $f(g_i)$ is calculated according to the probability of the tag:

$$\frac{\#T \text{ in any context}}{\sum_{T' \in \mathcal{T}} \#T' \text{ in any context}}$$

    $\#T$ denotes the number of occurrences of T.
- Some entries to the training table are too small —with respect to other entries for the same tag— for them to be significant, and so they can be ignored. In this case, it is necessary to determine a threshold. After some experiments, the adopted value is 5% of the number of occurrences of the most probable context for that tag.

The population is evaluated after each evolution step, and the new average fitness is computed.

2.3 Genetic Operators

Each evolution step some chromosomes from the current population are applied genetic operators to produce new individuals and renew the population. The genetic operators used herein are *crossover*, which combines two chromosomes to generate a new one, and *mutation*, which creates a new chromosome by changing a randomly selected gene in a chromosome of the previous generation. The design of the genetic operators must balance between the inheritance of the ancestors' properties and the exploration of new areas of the search space. The efficiency of the algorithm is very sensitive to the rates of crossovers and mutations performed at each step, which are input parameters.

Crossover The crossover operation can be summarized as follows:

- Two chromosomes are randomly selected in the population with a probability proportional to its fitness.
- Then a crossover point is randomly selected, thus dividing the chromosome in two parts. In the selection of this partition point, positions corresponding to genes with low fitness are preferred.
- At this point two different kinds of crossover are considered, depending on the state of the evolution process:
  - For the first steps of the process (before the average fitness doubles that of the initial population): The first part of one parent is combined with the second part of the other parent thus producing two offsprings. This kind of crossover is expected to produce individuals quite different from the parents, thus helping to explore new areas of the search space.
  - For the following steps: At this stage it is expected to have quite accurate individuals, so the crossover applied is more conservative, trying to maintain the good characteristics of the parents. Thus, now only the tag at the crossover point is exchanged between the parents, producing again two new offsprings.

Mutation The mutation operation can be summarized as follows:

- Mutation is applied to the chromosomes resulting of the crossover operations.
- Mutation is applied to each gene of these chromosomes with a probability given by the mutation rate.
- The tag of the mutation point is replaced by another of the valid tags of the corresponding word. The new tag is randomly chosen according to its probability.

3 Experimental Results

The corpus used to train the tagger has a huge influence on the performance. It must be a good sample of the language and most of the times the corpus used

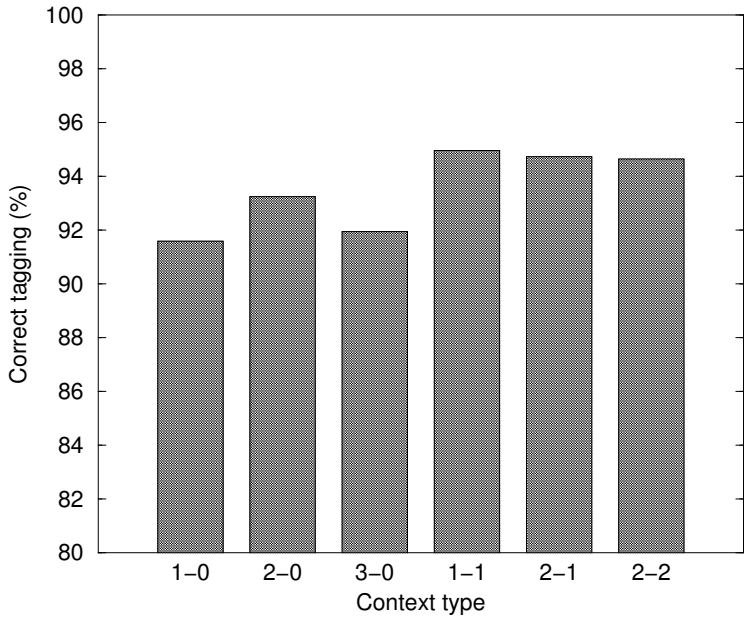

Fig. 2. Accuracy rate obtained for different sizes of the context, using a training corpus of 185000 words, a test text of 2500 words, a population size of 20 individuals, a crossover rate of 50%, and a mutation rate of 5%.

is domain-specific. In this work we have used the *Brown* corpus. The tag set is not too large, what favours the accuracy of the system, but at the same time is large enough to make the system useful.

Different experiments have been carried out in order to study the main factors affecting the accuracy of the tagging: size and shape of the contexts used for the training table, size of the training corpus and evolutionary parameters.

3.1 Influence of the Amount of Context Information

The way in which the context information is used by the tagger influences its performance. This piece of information can be fixed or variable and have different sizes. In order to determine the maximum length of contexts we have carried out a statistical analysis of the correlation between the tag at a given position in the Brown corpus and the tags separated a certain distance d; that is, we have computed $P(X_d|X_0)$ for different values of d. From those data we can determine the smallest value of d for which

$$P(X_d|X_0)/P(X_d) \approx 1$$

i.e. the minimum distance d for which the tag of the word at position 0 has no influence over the tag at position d. From this analysis we observe that $d = 3$ is

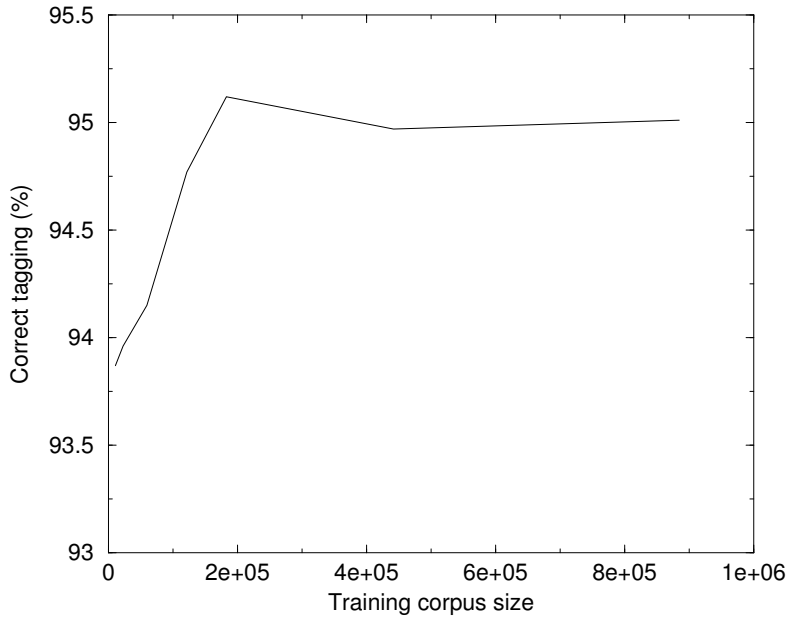

Fig. 3. Accuracy rate reached with different sizes of the training corpus, using contexts of the form 1-1, a test text of 2500 words, a population size of 20 individuals, a crossover rate of 50%, and a mutation rate of 5%.

a safe value and in most cases $d = 1$ is enough. Therefore, contexts longer that 3 are irrelevant.

Figure 2 shows the accuracy rates reached with different context sizes (always with $d \leq 3$) and shapes. Results show that the best performance is reached for small context sizes, such as 1-1, probably because with larger contexts the number of occurrences for many entries of the training table is not significant enough.

3.2 Influence of the Size of the Training Text

The next step is determining the influence of the size of the training text. Though intuitively it may seem obvious that the larger the training corpus, the better, we must take into account that the size of the training table slows down the evolutionary process, so only a significant increase of the accuracy can justify an increase of the training corpus. Figure 3 presents the increase of the accuracy with the size of the training corpus, showing that it saturates beyond a certain size (around 200,000 words).

The best Hidden Markov models typically perform at about the 95% level of correctness [Cha93]. Brill's model [Bri97], based on transformation rules, with a training set of 120,000 words and a separate test set of 200,000 words, obtained

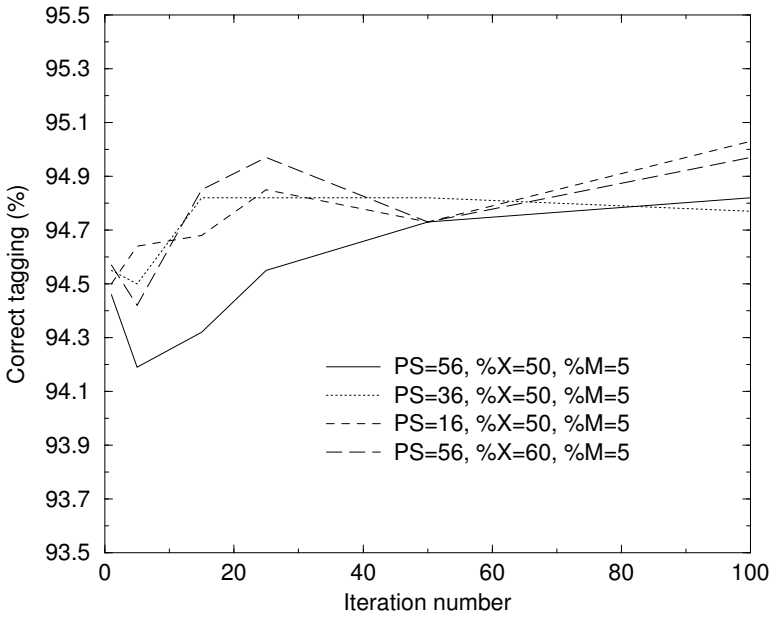

Fig. 4. Accuracy rate reached as a function of the number of iterations. PS stands for the population size, %X for the crossover rate, and %M for the mutation rate.

a tagging accuracy of 95.6%, which increased up to 96.0% by expanding the training set to 350,000 words. Therefore our results are comparable to that of other probabilistic and rule-based approaches, with the advantage that they can be obtained in a very efficient way thanks to the evolutionary algorithm. Besides, in this case the algorithm turns out to be particularly fast because the best tagging can be reached with small populations and just a few iteration steps.

3.3 Study of the Evolutionary Algorithm Parameters

We have also investigated the parameters of the evolutionary algorithm: population size and crossover and mutation rates. Figure 4 shows the results obtained. We have observed that small populations are enough to obtain high accuracy rates, because the sentences are tagged one by one and so in general a small population is enough to represent the variety of possible taggings. This leads to a quicker algorithm. Crossover and mutation rates must be in correspondence with the population size: the larger the population, the higher the rates. It is therefore not only unnecessary but also inconvenient to increase the population.

4 Conclusions

The complexity of the tagging process for texts of important length can be treated by using stochastic methods that provide an approximate solution in a reasonable time, such as genetic algorithms. This work has developed a genetic algorithm that works with a population of potential taggings for each input sentence in the text. The evaluation of individuals is based on a training table composed of contexts extracted from a set of training texts.

Results indicate that the evolutionary approach is robust enough for tagging texts of natural language, obtaining accuracies comparable to other statistical approaches. The tests indicate that the length of the contexts extracted for the training is a determining factor for the results. However, there is a limit beyond which no further improvement can be obtained. Results have also shown the influence of the size of the training texts.

A study of the most frequent errors in the tagging has revealed the following points:

- As expected, words that require a tag that is not the most frequent or that appears in an odd context tend to be wrongly tagged.
- In general, the longer the sentence to be tagged, the better the results, because in large sentences there will be enough contexts to compensate the weight of some erroneous taggings.
- When deciding the size of the training table we must take into account that when tagging sentences whose words require one of its frequent tags and that appear in a frequent context, the longer the training text, the more accurate the tagging. However, when tagging sentences with words that adopt some of their most rare tags, the length of the training corpus can spoil the results.
- Another observation is the correlation between the parameters of the algorithm and the complexity of the texts to be analysed. The more complex the text (number of ambiguous words), the larger the population size required to quickly reach a correct tagging.
- Besides, as the population size increases, higher rates of crossover and mutation are required to maintain the efficency of the algorithm.

References

[Ara00] L. Araujo. Evolutionary parsing for a probabilistic context free grammar. In *Proc. of the Int. Conf. on on Rough Sets and Current Trends in Computing (RSCTC-2000)*, 2000.

[Bri95] E. Brill. Transformation-based error-driven learning and natural language processing: A case study in part of speech tagging. *Computational Linguistics*, 21(4), 1995.

[Bri97] E. Brill. Unsupervised learning of disambiguation rules for part of speech tagging. In S. Armstrong, K. Church, P. Isabelle, S. Manzi, E. Tzoukermann, and D. Yarowsky, editors, *Natural Language Processing Using Very Large Corpora*. Kluwer Academic Press, 1997.

[Cha93] E. Charniak. *Statistical Language Learning*. MIT press, 1993.

[CKPS92] D. Cutting, J. Kupiec, J. Pedersen, and P. Sibun. A practical part-of-speech tagger. In *Proc. of the Third Conf. on Applied Natural Language Processing.* Association for Computational Linguistics, 1992.

[DeM90] C. DeMarcken. Parsing the lob corpus. In *Proc. of the 1990 of the Association for Computational Linguistics.* Association for Computational Linguistics, 1990.

[Jel85] F. Jelinek. Self-organized language modelling for speech recognition. In J. Skwirzinski, editor, *Impact of Processing Techniques on Communications.* Dordrecht, 1985.

[MD96] T. Dunning M. Davis. Query translation using evolutionary programming for multilingual information retrieval II. In *Proc. of the Fifth Annual Conf. on Evolutionary Programming.* Evolutionary Programming Society, 1996.

[Mer94] B. Merialdo. *Tagging english text with a probabilistic model.* 1994.

[Mic94] Z. Michalewicz. *Genetic algorithms + Data Structures = Evolution Programs.* Springer-Verlag, 2nd edition, 1994.

[Qui93] J.R. Quinlan. *C 4.5: Programs for Machine Learning.* Morgan Kaufmann Publisher, 1993.

[SS94] H. Schutze and Y. Singer. Part od speech tagging using a variable memory markov model. In *Proc. of the 1994 of the Association for Computational Linguistics.* Association for Computational Linguistics, 1994.

[TS95] I.H. Witten T.C. Smith. A genetic algorithm for the induction of natural language grammars. In *Proc. IJCAI-95 Workshop on New Approaches to Learning Natural Language*, pages 17–24, Montreal, Canada, 1995.

[Wya91] P. Wyard. Context free grammar induction using genetic algorithms. In *Proc. of the 4th Int. Conf. on Genetic Algorithms*, pages 514–518, 1991.