Available online at www.sciencedirect.com
ScienceDirect
Information Processing Letters 100 (2006) 173–182
Information Processing Letters
www.elsevier.com/locate/ipl

Natural language tagging with genetic algorithms

Enrique Alba [a,*], Gabriel Luque [a], Lourdes Araujo [b]

[a] *Departamento de Lenguajes y Ciencias de la Computación, Universidad de Málaga, Spain*
[b] *Departamento Sistemas Informáticos y Programación, Universidad Complutense, Spain*

Received 25 May 2006; received in revised form 30 June 2006; accepted 10 July 2006

Available online 10 August 2006

Communicated by L. Boasson

Abstract

This work analyzes the relative advantages of different metaheuristic approaches to the well-known natural language processing problem of part-of-speech tagging. This consists of assigning to each word of a text its disambiguated part-of-speech according to the context in which the word is used. We have applied a classic genetic algorithm (GA), a CHC algorithm, and a simulated annealing (SA). Different ways of encoding the solutions to the problem (integer and binary) have been studied, as well as the impact of using parallelism for each of the considered methods. We have performed experiments on different linguistic corpora and compared the results obtained against other popular approaches plus a classic dynamic programming algorithm. Our results claim for the high performances achieved by the parallel algorithms compared to the sequential ones, and state the singular advantages for every technique. Our algorithms and some of its components can be used to represent a new set of state-of-the-art procedures for complex tagging scenarios.
© 2006 Elsevier B.V. All rights reserved.

Keywords: Genetic algorithms; CHC algorithm; Natural language processing; Part-of-speech tagging; Parallel algorithms

1. Introduction

Part-of-speech (POS) tagging or simply "tagging" is a basic task in natural language processing (NLP). Tagging aims to determine which is the most likely lexical tag for a particular occurrence of a word in a sentence. There are many NLP tasks which can be improved by applying disambiguation to the text [1]. The ambiguity of syntactic analysis or partial parsing—a kind of analysis limited to particular types of phrases—is highly simplified in the absence of lexical ambiguity. Many partial parses work on the output of a tagger [2] by testing the appearance of regular expressions of tags, which define the searched patterns. For instance, the word *can* can be a noun, an auxiliary verb or a transitive verb. The category assigned to the word will determine the structure of the sentence in which it appears and thus its meaning.

Other important applications of tagging are information retrieval [3] and question answering. For example, before identifying the documents relevant for the requested information, an information retrieval system needs to represent those documents according to some criterion. Tagging and partial parsing can be very useful here to perform such an organization and obtain the representative terms.

Moreover, tagging is a difficult problem by itself, since many words belong to more than one lexical class. To give an idea, according to [4], over 40% of the words

[*] Corresponding author.
E-mail addresses: eat@lcc.uma.es (E. Alba), gabriel@lcc.uma.es (G. Luque), lurdes@sip.ucm.es (L. Araujo).

0020-0190/$ – see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.ipl.2006.07.002

This	the	therapist	may	pursue	in	later	questioning
DT	AT	NN	NNP	VB	RP	RP	VB
QL			MD	VBP	NNP	RB	NN
					RB	JJ	JJ
					NN	JJR	
					FW		
					IN		

Fig. 1. Tags for the words in a sentence extracted from the Brown corpus. Underlined tags are the correct ones, according to the Brown corpus. Tags correspond to the tag set defined in the Brown corpus: DT stands for determiner/pronoun, AT for article, NN for common noun, MD for modal auxiliary, VB for uninflected verb, etc.

appearing in the hand-tagged Brown corpus [5] are ambiguous.

Because of the importance and difficulty of this task, a lot of work has been carried out to produce automatic taggers. Automatic taggers, usually based on Hidden Markov Models, rely on statistical information to establish the probabilities of each scenario. The statistical data are extracted from previously hand-tagged texts, called *corpus*. These stochastic taggers [1,6] neither require knowledge of the rules of the language nor try to deduce them, and thus they can be applied to texts in any language, provided they can be previously trained on a corpus for that language.

The context in which the word appears helps to decide which is its more appropriate tag, and this idea is the basis for most taggers. For instance, consider the sentence in Fig. 1, extracted from the Brown corpus. The word *questioning* can be disambiguated as a common name if the preceding tag is disambiguated as an adjective. But it might happen that the preceding word to be ambiguous, so there may be many dependencies which must be resolved simultaneously.

The statistical model considered in this work amounts to maximize a global measure of the probability of the set of contexts (a tag and its neighboring tags) corresponding to a given tagging of the sentence. Then, we need a method to perform the search of the tagging which optimizes this measure of probability.

The aim of this article is to check and compare different metaheuristic algorithms to perform such a search, such as a genetic algorithm (GA), a CHC algorithm, and a simulated annealing (SA). One of the advantages of using an evolutionary algorithm (EA) as the search algorithm for tagging is that these algorithms can be applied to any statistical model, even if they do not rely on the Markov assumption (i.e., the tag of a word only depends on the previous words), as it is required in classic search algorithms for tagging, such as the widely used Viterbi [7]. Genetic algorithms have been previously applied to the problem [8,9], obtaining accuracies as good of those of typical algorithms used for stochastic tagging. CHC is a non-traditional genetic algorithm, which presents some particular features: CHC guarantees the survival of the best solutions found, does not allow the mating of similar solutions, and uses specialized operations.

One of the aims of this work is to investigate if the particular mechanisms of CHC for diversity can improve the selection of different sets of tags. From previous work, it has been observed that words incorrectly tagged are usually those which require one of their more rare tags, or which appear in an infrequent context. We plan to use a quality function based on the probability of the contexts of a sequence of tags assigned to a sentence. A priori, it should be difficult for a GA to find appropriate tags within high probability contexts. CHC allows simultaneously changing several tags of the sequence, which can hopefully lead to explore combinations of tags very different from those of the ancestors and then to better results. Thus, it is interesting to study what is more advantageous; the smooth exploration of the GA or the more disruptive one of CHC. We have also compared the results of the GAs with those obtained from Simulated Annealing, in order to ascertain the suitability of the evolutionary approach compared with other optimization methods.

For most tagging applications, the whole process of search is time consuming, what made us to include a parallel version of the algorithms. We also compare the results of our approaches with the ones of Viterbi, a classical method for solving this problem, in order to test the accuracy of all our methods in a wider spectrum of techniques.

The rest of the paper proceeds as follows: Section 2 describes the kind of statistical models to which our algorithms can be applied. Sections 3, 4 and 5 describe the GA, CHC, and SA algorithms, and Section 6 discusses the parallel version of these algorithms. Section 7 presents the details on how the algorithms are applied to tagging. Section 8 describes and discusses the computational results, and Section 9 draws the main conclusions of this work.

2. A probabilistic approach for tagging

Statistical tagging, probably the most extended approach nowadays, is based on statistical models defined on a number of parameters, which take their values from probabilities extracted on tagged texts. The goal of these models is to assign to each word of a sentence the most likely lexical tag according to the *context* of the word, i.e., according to the tags of other words surrounding the considered one. Therefore, we can collect statistics on the number of occurrences of the contexts resulting of assigning their different valid part-of-speech to the considered word, and then choose the more likely one. However, the surrounding words may also be ambiguous, and thus, we need some kind of statistical model to select the "best" tagging for the whole sequence according to the model. More formally, the part-of-speech tagging problem can be stated as

$$t_{1,n} = \arg\max_{t_{1,n}} P(t_{1,n}|w_{1,n}),$$

where $\arg\max_x f(x)$ is the value of x which maximizes $f(x)$, and $t_{1,n}$ is the tag sequence of the words $w_{1,n}$ which compose the sentence being tagged.

If we assume that the tag of a word only depends on the previous tag, and that this dependency does not change throughout the time, we can adopt a Markov Model for tagging. Let w_i the word at position i in the text, t_i the tag of w_i, $w_{i,j}$ the words appearing from position i to j, and $t_{i,j}$ the tags for the words $w_{i,j}$. Then, the model states that

$$P(t_{i+1}|t_{1,i}) = P(t_{i+1}|t_i).$$

If we also assume that the probability of a word appearing at a particular position only depends on the part-of-speech tag assigned to that position, the optimal sequence of tags for a sentence can be estimated as

$$\begin{aligned} t_{1,n} &= \arg\max_{t_{1,n}} P(t_{1,n}|w_{1,n}) \\ &= \prod_{i=1}^{n} P(w_i|t_i)P(t_i|t_{i-1}). \end{aligned}$$

Accordingly, the parameters of the Markov model tagger can be computed from a training corpus. It can be done by recording the different contexts of each tag in a table called *training table*. This 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 tagset.

The Markov model for tagging described above is known as a *bigram* tagger because it makes predictions based on the preceding tag, i.e., the basic unit considered is composed of two tags: the preceding tag and the current one. This model can be extended in such a way that predictions depend on more than one preceding tag. For example, a *trigram* model tagger makes its predictions depending on the two preceding tags.

Once the statistical model has been defined, most taggers use the Viterbi algorithm [7] (a dynamic programming algorithm) to find the tag sequences which maximize the probability according to the selected Markov model.

We here offer an alternative approach to tagging which can be used instead of the Viterbi algorithm. This approach relies on using evolutionary algorithms, which include a number of search templates based on the production of offsprings and the survival of the fittest. These heuristic techniques provide us a general method that can be applied to any statistical model. For example, they can be applied to perform tagging according to the Markov model described above or not. In this way, they can also be applied to other models for which there is no known efficient algorithm. For instance, they can be applied to a model which has been proven to improve the results over a Markov one [10], in which the context of a word is composed of both, the tag of the preceding words and also the tag of the following words. The potential problem in using evolutionary algorithms (EAs) is that many of them do not guarantee to reach the optimum solution but a reasonably good approximation, according to the resources assigned (time and memory). In addition, the probability of error can be systematically decreased in EAs by increasing the number of points explored with a fine tuning of the algorithm parameters.

According to these considerations, we are going to explore different metaheuristic techniques for tagging. The statistical model we consider amounts to maximize a global measure of the probability of the set of contexts (a tag and its neighboring tags) corresponding to a given tagging of the sentence. The contexts considered are very general and widely applicable since they are composed of a certain number of tags on the left and another on the right of the word at the considered position.

3. Genetic algorithm

Genetic algorithms (GAs) [11] are stochastic search methods that have been successfully applied in many real applications. A GA is an iterative technique that applies stochastic operators on a pool of individuals (tentative solutions). A fitness function allocates a real value to every individual indicating its suitability to the problem. Traditionally, GAs are associated to the use of a binary representation, but nowadays you can find GAs that use other types of representations. A GA usually applies a recombination operator on two solutions, plus a mutation one that randomly modifies the individual contents to promote diversity and thus reaching new portions of the search space not implicitly present in the previous generations.

4. CHC algorithm

CHC [12] is a variant of a genetic algorithm with a particular way of promoting diversity. It uses a highly disruptive crossover operator to produce new individuals maximally different from their parents. It is combined with a conservative selection strategy which introduces a kind of inherent elitism. The main features of this algorithm are:

- The mating is not restricted to the best individuals, but parents are randomly paired in a mating pool. However, recombination is only applied if the Hamming distance between the parents is above a certain threshold (*incest prevention*).
- CHC uses a *half-uniform crossover* (HUX), which exchanges half of the differing genes.
- CHC guarantees survival of the best individuals selected from the set of parents and offsprings.
- Mutation is not applied directly. Instead, CHC uses a re-start mechanism when the population remains unchanged after a given number of generations.

5. Simulated annealing

Simulated annealing (SA) [13] is a stochastic search technique that can be seen as a hill-climber with an internal mechanism to escape from local optima. In SA, the solution s' is accepted as the new current solution s if $\delta \geqslant 0$ holds, where $\delta = f(s') - f(s)$. To allow escaping from a local optimum, moves that decrease the energy function are accepted with a decreasing probability $\exp(\delta/T)$ if $\delta < 0$, where T is a parameter called the "temperature". The decreasing values of T are controlled by a cooling schedule, which specifies the temperature values at each stage of the algorithm, what represents an important decision for its application. Here, we are using a proportional method for updating the temperature ($T_k = \alpha \cdot T_{k-1}$, where α indicates the decrease speed of the temperature).

6. Parallel metaheuristics

A parallel EA (PEA) is an algorithm composed of multiple EAs, regardless of their population structure. Each component (usually a traditional EA) subalgorithm includes an additional phase of *communication* with a set of subalgorithms [14]. In this work, we have chosen a *distributed EA* (dEA) because of its popularity and because it can be easily implemented in clusters of machines. In distributed EAs (also known as Island model) there exists a small number of islands performing separate EAs, and periodically exchanging individuals after a number of isolated steps (*migration frequency*). Concretely, we use a static ring topology in which the best individual is migrated, and asynchronously included in the target population only if it is better than the local worst-existing solution.

The parallel SA (PSA) is also composed of multiple asynchronous SAs. Each component SA, starts off from a different random solution and exchanges the best solution found (*cooperation* phase) with its neighboring SA in the ring.

7. Evolutionary design for tagging

The first step in designing an evolutionary algorithm is to define the data structure included into the individuals which compose the population. Genetic operators on them must also be defined, as well as a selection policy based on a measure of the individual quality, or "fitness".

7.1. Individuals

Tentative solutions here are made of sequences of genes. Each gene corresponds to each word in the sentence to be tagged. Fig. 2 shows some example individuals for the sentence in Fig. 1.

Each gene represents a tag and additional information useful in the evaluation of the solution, such as counts of contexts for this tag according to the training table. Each gene's tag is represented by an index to a vector which contains the possible tags of the corresponding word. The composition of the genes depends on the chosen coding, as Fig. 3 shows. In the integer coding the gene is just the integer value of the index. In the binary coding the gene is the binary representation

Sent.	This	the	therapist	may	pursue	in	later	questioning
Ind. 1:	DT	AT	NN	NNP	VBP	IN	JJ	VB
Ind. 2:	DT	AT	NN	MD	VB	RB	RB	NN
Ind. 3:	QL	AT	NN	NNP	VB	FW	JJ	JJ

Fig. 2. Potential individuals for the sentence in Fig. 1.

Word	Tag index							int	bin
	0	1	2	3	4	5	...		
This	DT	QL						0	000
the	AT							0	000
therapist	NN							0	000
may	NNP	MD						1	001
pursue	VB	VBP						0	000
in	RP	NNP	RB	NN	FW	IN		5	101
				...					

Fig. 3. Integer and binary codings of a possible selection of tags chosen for the words of a sentence extracted from the Brown corpus. The selected tags appear underlined.

of the index. As in the texts we have used for experiments the maximum number of tags per word is 6, we have used a binary code of 3 bits.

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 produces, along with the tags assigned to its neighbor words, the most probable context, according to the training text.

7.2. Fitness evaluation

The fitness of an individual is a measure of the total correctness probability of its sequence of tags, according to the data from the training table. It is computed as the sum of the fitness of its genes, $\sum_i f(g_i)$. The fitness of a gene is defined as

$$f(g) = \log P(T|LC, RC),$$

where $P(T|LC, RC)$ is the probability that the tag of gene g is T, given that its context is formed by the sequence of tags LC to the left and the sequence RC to the right (the logarithm is taken in order to make fitness additive). This probability is estimated from the training table as

$$P(T|LC, RC) \approx \frac{occ(LC, T, RC)}{\sum_{T' \in \mathcal{T}} occ(LC, T', RC)},$$

where $occ(LC, T, RC)$ is the number of occurrences of the list of tags LC, T, RC in the training table, and $\mathcal{T}$ is the set of all possible tags of g_i.

A particular, sequence LC, T, RC may not be listed in the training table, either because its probability is strictly zero (if the sequence of tags is forbidden for some reason) or, most likely, because there is insufficient statistics. In these cases we proceed by successively reducing the size of the context, alternatively ignoring the rightmost and then the leftmost tag of the remaining sequence (skipping the corresponding step whenever either RC or LC are empty) until one of these shorter sequences matches at least one of the training table entries or until we are left simply with T. In this latter case we take as fitness the logarithm of the frequency with which T appears in the corpus (also contained in the training table).

7.3. Genetic operators

For the GA, we use a one point crossover, i.e., a crossover point is randomly selected and the first part of each parent is combined with the second part of the other parent thus producing two offsprings. Then, a mutation point is randomly selected and the tag of this point is replaced by another of the valid tags of the corresponding word. The new tag is randomly chosen according to its probability (the frequency at which it appears in the corpus).

The CHC algorithm applies HUX crossover, randomly taking from each parent half of the tags in which they differ and exchanging them.

Individuals resulting from the application of the genetic operators along with the old population are used to create the new one.

8. Computational experiments

We have used as the set of training texts for our taggers the Brown [5] and Susanne [15] corpora, two of the most widespread in linguistics. For the Brown corpus we have used a training set of 165 276 words, a data set to tune the parameters of 2477 words, and a test set of 2421 words. For the Susanne corpus we have used a training set of 145 778 words, a data set to tune the parameters of 2489 words, and a test set of 2510 words. The CHC algorithm has been run with a crossover rate of 50%, without mutation. Whenever convergence is achieved, 90% of population is renewed. The GA applies the recombination operator with a rate of 50%, and the mutation operator with a rate of 5%. In the parallel version, the migration occurs every 10 generations. We made preliminary tests with different parameter settings for determining the best values for each algorithms. The analysis of other specific operators is deferred for a future work.

Tables 1, 2, and 3 show the results obtained with CHC, GA, and SA algorithms, using both, integer and binary codings. The two upper rows correspond to the Brown text with two different contexts (1-0 is a context which considers only the tag of the preceding word and 2-0 considers the tag of the two preceding words) and the two lower rows are the results for the Susanne text. The two texts contain 2500 words approximately. Figures represent the best result out of 30 independent runs. The globally best result for each row appears in boldface. **Integer** stands for the integer representation and **Binary** for the binary representation with a code of 3 bits. For CHC and GA, measures have been taken for two population sizes. Furthermore, sequential and parallel versions with 4 islands are analyzed. In evolutionary algorithms, the population size of each island is the global population size divided by the number of islands.

Table 1
Tagging accuracy obtained with the CHC algorithm for two test texts

Context		Integer				Binary			
		PS = 32		PS = 64		PS = 32		PS = 64	
		Seq.	Par.	Seq.	Par.	Seq.	Par.	Seq.	Par.
Brown	1-0	91.02	91.74	91.35	91.55	94.98	**95.32**	94.67	94.62
	2-0	91.31	91.31	91.83	92.18	**95.35**	**95.35**	95.18	95.03
Susanne	1-0	91.43	91.19	92.32	92.68	94.35	**95.01**	93.61	94.41
	2-0	93.42	93.75	93.53	93.56	94.37	**94.82**	93.96	94.31

PS stands for Population Size.

Table 2
Accuracy obtained with the GA for the two test texts

Context		Integer				Binary			
		PS = 32		PS = 64		PS = 32		PS = 64	
		Seq.	Par.	Seq.	Par.	Seq.	Par.	Seq.	Par.
Brown	1-0	95.83	**96.01**	94.34	94.95	93.15	93.02	92.97	93.02
	2-0	96.13	**96.41**	95.14	95.42	94.86	94.82	94.54	94.89
Susanne	1-0	96.41	**96.74**	95.46	96.25	95.12	95.32	94.96	94.54
	2-0	**97.32**	**97.32**	96.91	97.01	95.39	95.43	95.34	95.39

PS stands for Population Size.

Looking at Table 1, the first conclusion is that the binary coding always achieves a higher accuracy with respect to the integer one. This suggests that the integer representation is not appropriate for CHC, probably because the low number of genes of the latter interferes with the CHC mechanism to avoid crossover between similar individuals. Regarding the parallel executions, we can observe that the parallel version usually provides more accurate results, particularly for a population of 32 individuals, because for such a small population the higher diversity introduced by parallelism is beneficial. In general, the accuracy obtained for Brown text is better than for Susanne text, probably because the statistics provided by the Susanne corpus are poorer[1] and the CHC mechanism for diversity is limited by these data. Anyway, the best results are always obtained using binary coding and parallel executions for any instance.

Table 2 shows the results obtained with the GA. In this case, the integer representation provides the best results. Again, parallel versions improve the sequential results, obtaining the best results when the population is composed of 32 individuals. We can observe that for this algorithm, the accuracy increases when the context is larger. The same trend was observed before for CHC, but not so conclusively as for the GA. Also, we can notice that unlike the previous results for CHC, the results for Susanne text are more accurate than the ones for Brown text.

Table 3
Accuracy obtained with the SA algorithm for the two test texts (best result out of thirty independent runs)

Context		Integer		Binary	
		Seq.	Par.	Seq.	Par.
Brown	1-0	91.41	**91.83**	91.25	91.58
	2-0	91.92	**92.28**	91.68	91.72
Susanne	1-0	91.03	**91.87**	89.79	90.53
	2-0	**92.31**	**92.31**	91.31	91.74

Table 3 presents the data obtained with the SA algorithm. The SA algorithm performs 5656 iterations using a Markov chain of length 800 and with a decreasing factor of 0.99. In the parallel version, each SA component exchanges the best solution found with its neighbor SA in the ring every 100 iterations. We can observe that SA always provides worse results than any of the evolutionary algorithms, thus proving the advantages of the evolutionary approach. Anyhow, the parallel SA is still better in accuracy than the sequential version, as we also noticed for CHC and GA. In general, the results obtained for this algorithm are very poor, indicating that SA is not able to solve this problem adequately.

Table 4 presents the best value and the average for the configuration which provides the best results of each algorithm, i.e., parallel implementation using integer cod-

[1] Due to the large tagset of this corpus.

Table 4
Comparison of the results of all the algorithms for the two test texts

Context		GA-Int		CHC-Bin		SA-Int		Viterbi
		Best	Mean	Best	Mean	Best	Mean	Best
Brown	1-0	96.01	95.26	95.32	94.91	91.83	90.95	**97.04**
	2-0	96.41	96.23	95.35	94.80	92.28	92.14	**97.48**
Susanne	1-0	**96.74**	96.51	95.01	94.72	91.87	91.43	96.36
	2-0	**97.32**	96.84	94.82	94.38	92.31	91.45	96.53

Table 5
Execution times of the different versions of the algorithms (in seconds)

Context		GA-Int		CHC-Bin		SA-Int		Viterbi
		Seq.	Par.	Seq.	Par.	Seq.	Par.	Seq.
Brown	1-0	12.31	7.02	20.48	9.60	5.84	2.98	0.47
	2-0	47.93	21.19	60.92	26.44	17.32	7.93	0.47
Susanne	1-0	10.86	6.32	17.28	8.03	4.37	1.85	1.21
	2-0	75.12	24.31	123.94	58.31	32.12	10.42	6.53

ing for GA and SA, and binary one for CHC. We do not show the standard deviation because the fluctuations in the accuracy of different runs are always within the 1% interval, claiming that all the algorithms are very robust. Also, we include the results of the Viterbi method (as said before, a typical algorithm widely used for stochastic tagging) to perform a comparison between our evolutionary approach and a classical tagging method. The results of this algorithm have been obtained with the TnT system [16], a widely used trigram[2] tagger.

First we compare our algorithms, and latter, we compare our results with the Viterbi ones. We can observe in Table 4 that the GA has reached the globally best results for all the test texts and contexts, though the differences are small. This proves that the exploration of the search space given by the classical crossover and mutation operators are enough for this specific problem.

Now, we compare the Viterbi method against the GA (which obtains the best results of all our approaches). Viterbi obtains the best results for the Brown corpus, although the difference of accuracy with respect to GA is around 1%. However, for the Susanne corpus, which provided poorer statistics, the GA results outperform Viterbi's one. In this way, the heuristic nature of the genetic algorithm is illustrated as useful for tagging where traditional algorithms have low accuracy.

After these results another finding that is worth mentioning is that the accuracy obtained with the parallel versions of GA, around 97%, is a very good result [6] according to the statistical model used. We must take into account that the accuracy is limited by the statistical data provided to the search algorithm. Moreover, the goal of the model is to maximize the probability of the context composed by the tags assigned to a sentence, but it is only an approximate model. The correct tag for a word is not always the most probable one (though most times it is), and the algorithm is conditioned by this fact, but sometimes it is not the one which provides the most probable context either, and it is just in these cases when the tagger fails.

Let us now analyze Table 5, the average execution time for the configurations of the GA, CHC, and SA algorithms, the ones providing the best results for each of them, also including the execution time of the Viterbi method. We can observe that the execution time increases with the size of the context. We can also observe that GA is faster than CHC. Probably, this is due to two reasons: first, binary codings are slower than integer ones, because they require a decodification step prior to apply the fitness function, and second, CHC needs additional computations to detect the converge of the population or to detect incest mating. SA is the fastest of our algorithms. The reason of this is that the SA operates on a single solution, while the rest of the methods are population-based and in addition they execute more complex operators. The table also shows that the parallel implementation reduces the execution time considerably (between 42% and 78%), and this reduction is increasingly beneficial for larger contexts.

If we compare our approach with the Viterbi algorithm we can observe that the classical tagging algorithm is rather fast compared to the rest of our proposed

[2] It considers the previous two tags for deciding on the current tag.

Table 6
Accuracy and execution times obtained with the GA for the two test texts using two new contexts

Context		Seq.			Par.		
		Best	Mean	Time	Best	Mean	Time
Brown	1-1	96.72	96.41	56.92	**96.78**	96.56	19.98
	2-1	**96.43**	96.22	210.36	**96.43**	96.27	67.28
Susanne	1-1	98.36	98.11	77.14	**98.59**	98.39	21.25
	2-1	97.78	97.31	283.49	**98.01**	97.64	78.17

algorithms, though the differences for the Susanne corpus are smaller. Viterbi, and in general any specific algorithm for a problem, will always be more efficient than an evolutionary algorithm, which is a general technique that can be applied to any variant of the problem. However this generality provides some important advantages, such as the applicability to extensions of the model, such as contexts with tags on both sides, in our case. Another advantage concerning the execution time is that there exists parallel versions of the evolutionary algorithm which highly reduce the execution time without requiring addition design effort. In this way the evolutionary algorithm can reach generality and efficiency at the same time.

Finally, in order to offer a through analysis, we have also tested our best algorithm (a GA using integer coding) with two more complex contexts (1-1 and 2-1). Table 6 shows the results of these experiments. Viterbi cannot applied in this case because this algorithm is designed to search the data sequence which maximizes the observed data according to a Markov model, i.e., a model in which the current state only depends on the previous one. If we consider tags on the right of the word being tagged, our model is not a Markov process any more and Viterbi cannot be applied. This alone is a strong reason to further research with metaheuristics. As we saw before, the parallelism allows to improve the accuracy and, at the same time, the execution time is reduced considerably (see Table 5). In this case, we observe that the increase of the length of the context (from 1-1 to 2-1) provokes a larger execution times.

In the above experiments, the test data sets used had a small/medium size. Now, we want to analyze the behavior of our approaches with two larger instances. We now use a test set of 17 303 words from the Brown corpus, and of 15 006 words from the Susanne corpus. These values represent the 10% of the whole data approximately. We have perform 30 independent runs for these instances using the best configuration of our methods. The results of these experiments are shown in Table 7. Several conclusions can be extracted from these results.

Table 7
Accuracy and execution times obtained with all the methods for larger test data sets

Corpus alg.		Seq.			Par.		
		Best	Mean	Time	Best	Mean	Time
Brown	GA	96.67	96.37	334.1	**96.75**	96.61	113.7
	CHC	95.37	95.07	372.2	**95.48**	95.11	158.3
	SA	92.03	91.68	149.5	**92.17**	91.87	46.6
Susanne	GA	98.32	98.105	410.9	**98.61**	98.37	128.7
	CHC	95.12	94.77	658.3	**95.19**	94.81	190.5
	SA	91.89	91.37	189.2	**92.03**	91.52	59.8

First of all, we notice that the algorithms show a similar behavior to the one presented in previous experiments, i.e., the parallel versions outperform the serial ones, and the GA is the best approach to tackle this problem. In fact, we have carried out statistical analyses using t-test with a confidence level of 95%, which indicate that there is no statistically significant difference in the solution quality found between these experiments and the previous ones (with the exception of some results of the SA method). This demonstrates that our proposals have a very good scalability. Scalability is a very important feature of any algorithm if it is expected to be actually useful for the research community. However, as we expected, the execution time is higher than in previous experiments because the number of words in the test data sets has been incremented drastically.

Though what we propose in this paper is a search method valid for different tagging models, and thus our goal is not to compete with other models, in order to give an idea of the quality of the particular model that we have used, we present a comparison with the accuracy results of other systems evaluated on the same corpus. This corpus is the Wall Street Journal section of the Penn Treebank [17]. We have used a training set of 554 923 words and a test set of 2544 words. Table 8 compares our results, obtained with the GA and the integer representation, with the results obtained by Pla and Molina [18] and Halteren et al. [19]. Pla and Molina proposed a lexicalized HMM taking into account a set of selected words empirically obtained. However, the results shown in Table 8 correspond to the non-lexicalized model, because we do not use lexicalization. Halteren et al. propose a combination of different methods for tagging using several voting strategies. The figure which appears in the table for this work corresponds to the results for a single method, which is an HMM. These results confirm the conclusions obtained in the previous experiments. We can observe that in general, our approach has a better behavior when it uses a more complete information, i.e., using a higher context or us-

Table 8
Comparison with other systems tagging the Wall Street Journal section of the Penn Treebank

Context	GA-Int	Pla and Molina [18]	Halteren et al. [19]
1-0	95.79	96.13	–
2-0	96.39	96.44	**96.63**
1-1	**96.63**	–	–

"–" indicates that this information is not provided by the corresponding paper.

ing tags on the right of the word being tagged. In fact, our general method outperforms the results presented by [18] when right tags are used. Also it is able to find the best known solution [19] using 1-1 contexts.

We can observe that our general approach offers competitive results with respect to the specific methods presented in [18] and [19]. Nevertheless, the quality of our results can be improved by introducing in the model the refinements proposed by the other works, such as lexicalization, combination of different models, etc. Moreover, our method allows extending those models with new features such as right-hand contexts, what can lead to further improvements. We have also observed that our approaches can deal with larger instances adequately.

The obtained results show that a generic metaheuristic such as our genetic algorithm is able to solve the tagging problem with the same accuracy as an specific method which was designed for this problem. In addition, GAs can perform the search of the best sequence of tags for any context-based model, even if it does not fulfill the Markov assumption. Thus, it is a general method with a proved high quality, and even still able of hybridization with problem-dependent operations to yield more accurate results (future line of research).

9. Conclusions

This work compares different optimization methods to solve an important natural language task: the categorization of each word in a text. The optimization methods considered here have been a genetic algorithm (GA), a CHC algorithm, and a simulated annealing (SA). We have compared their results with a widely used method for tagging such as Viberti.

Results obtained allow extracting a number of conclusions. The first one is that the integer coding performs better than the binary one for the GA and the SA, while the binary one is the best for the CHC algorithm. Parallelism has also proven to be useful, always throwing the more accurate results even with small populations, and reducing the execution time of all the algorithms. The GA has been found to be better than CHC, indicating that the exploration of the search space achieved by the classical genetic operators is enough for this problem. The two evolutionary algorithms have outperformed SA. Also, we have observed that our evolutionary approach is able of outperform classical algorithms such as Viberti in some cases. These results showed that a metaheuristic such as our genetic algorithm is able to solve the tagging problem with the same accuracy as the Viterbi method (an specific method for this problem) with additional scenarios for application forbidden to other techniques.

For the future, we plan to investigate other genetic operators for the evolutionary algorithms considered herein, as well as other kinds of metaheuristic methods for tagging.

Acknowledgements

The two first author are partially supported by the Ministry of Science and Technology and FEDER under contract TIN2005-08818-C04-01 (the OPLINK project). The third author has been supported by the project TIC2003-09481-C04.

References

[1] C.D. Manning, H. Schütze, Foundations of Statistical Natural Language Processing, MIT Press, Cambridge, MA, 2000.

[2] F. Pla, A. Molina, N. Prieto, Tagging and chunking with bigrams, in: Proc. of the 17th Conference on Computational Linguistics, 2000, pp. 614–620.

[3] R.A. Baeza-Yates, B.A. Ribeiro-Neto, Modern Information Retrieval, Addison-Wesley, Reading, MA, 1999.

[4] S.J. DeRose, Grammatical category disambiguation by statistical optimization, Computational Linguistics 14 (1988) 31–39.

[5] F.W. Nelson, H. Kucera, Manual of information to accompany a standard corpus of present-day edited American English, for use with digital computers, Tech. rep., Dep. of Linguistics, Brown University, 1979.

[6] E. Charniak, Statistical Language Learning, MIT Press, Cambridge, MA, 1993.

[7] G.D. Forney, The Viterbi algorithm, Proceedings of the IEEE 61 (3) (1973) 268–278.

[8] L. Araujo, Part-of-speech tagging with evolutionary algorithms, in: Proc. of Int. Conf. on Intelligent Text Processing and Computational Linguistics, in: Lecture Notes in Computer Science, vol. 2276, Springer-Verlag, Berlin, 2002, pp. 230–239.

[9] L. Araujo, G. Luque, E. Alba, Metaheuristics for natural language tagging, in: K. Deb, et al. (Eds.), Genetic and Evolutionary Computation Conference (GECCO-2004) Seattle, Washington, in: Lecture Notes in Computer Science, vol. 3102, Springer, Berlin, 2004, pp. 889–900.

[10] E.F.T.K. Sang, Memory-based shallow parsing, J. Mach. Learn. Res. 2 (2002) 559–594.

[11] L. Davis (Ed.), Handbook of Genetic Algorithms, Van Nostrand Reinhold, New York, 1991.

[12] L.J. Eshelman, The CHC adaptive search algorithm: How to have safe search when engaging in nontraditional genetic recombination, in: Proceedings of the First Workshop on Foundations of Genetic Algorithms, Morgan Kauffman, Los Altos, CA, 1991, pp. 265–283.

[13] S. Kirkpatrick, C.D. Gelatt, M.P. Vecchi, Optimization by simulated annealing, Science 220 (4598) (1983) 671–680.

[14] E. Alba (Ed.), Parallel Metaheuristics. A New Class of Algorithms, Wiley and Sons, New York, 2005.

[15] G. Sampson, English for the Computer, Clarendon Press, Oxford, 1995.

[16] T. Brants, TnT: a statistical part-of-speech tagger, in: Proceedings of 6th Conference on Applied Natural Language Processing, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2000, pp. 224–231.

[17] M.P. Marcus, B. Santorini, M.A. Marcinkiewicz, Building a large annotated corpus of English: The Penn Treebank, Computational Linguistics 19 (2) (1994) 313–330.

[18] F. Pla, A. Molina, Improving part-of-speech tagging using lexicalized HMMs, Nat. Lang. Eng. 10 (2) (2004) 167–189.

[19] H. van Halteren, W. Daelemans, J. Zavrel, Improving accuracy in word class tagging through the combination of machine learning systems, Comput. Linguist. 27 (2) (2001) 199–229.