Multiobjective Genetic Programming for Natural Language Parsing and Tagging *

L. Araujo

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

Abstract. Parsing and Tagging are very important tasks in Natural Language Processing (NLP). Parsing amounts to searching the correct combination of grammatical rules among those compatible with a given sentence. Tagging amounts to labeling each word in a sentence with its lexical category and, because many words belong to more than one lexical class, it turns out to be a disambiguation task. Because parsing and tagging are related tasks, its simultaneous resolution can improve the results of both of them. This work aims developing a multiobjective genetic program to perform simultaneously statistical parsing and tagging. It combines the statistical data about grammar rules and about tag sequences to guide the search of the best structure. Results show that any of the implemented multiobjective optimization models improve on the results obtained in the resolution of each problem separately.

1 Introduction

Parsing and tagging are very important tasks in Natural Language Processing (NLP). They are usually a prior step to carry on many NLP processes such as machine translation, information retrieval, speech recognition, etc. Parsing amounts to searching the correct combination of grammatical rules among those compatible with a given sentence. Tagging amounts to labeling each word in a sentence with its lexical category (noun, verb, etc), disambiguating those words which belong to more than one lexical class. Languages are ambiguous and many words belong to more than one lexical class. Thus, disambiguation methods are required to proceed with tagging. The research in automatic part-of-speech tagging [1] and parsing [2] has increased a lot in the past few years, probably due to the increasing availability of large annotated corpora.

Often, tagging is a prior step to parsing. Grammatical ambiguity is highly reduced if lexical ambiguity has been eliminated. In fact, Dalrymple [5] has shown that if a perfect tagger were available, we could obtain a reduction in ambiguity of about 50%.

On the other hand, a perfect disambiguation of a lexical tagging requires taking into account at least the parsing, but sometimes also discourse analysis

* Supported by project TIC2003-09481-C04.

and world knowledge. Thus, the simultaneous resolution of tagging and parsing can improve on the results of both of them.

This work aims at developing an evolutionary multiobjective optimization (EMO) algorithm to perform simultaneously statistical parsing and tagging. It combines the statistical data about grammar rules and about sequences of lexical tags to guide the search of the best structure. The work relies on previous works which apply separately genetic programming (GP) to perform parsing [2] and an evolutionary algorithm for solving tagging [1]. Because, even separately the results provided by these evolutionary systems are comparable or even better than those obtained with algorithms of exhaustive search, it is expected than the EMO genetic program can improve on the results for both problems.

Parsing is essentially a Multi-Objective Optimization process: a search for the most probable combination of grammar rules as well as for the most probable combination of lexical categories for the words of the sentence. An EMO algorithm for this problem can take advantage of several EMO models which have appeared in the literature [6, 4]. Some of the simplest ones, usually known as aggregative functions, amounts to combining the different objectives into a single function or linear fitness combination, i.e., to adding all the objective functions together using different weighting coefficients for each of them, thus transforming the problem into a scalar optimization one. Some alternative approaches are based on the concept of Pareto optimum[1]. Solutions included in the Pareto optimal set are called nondominated. The approaches considered herein have been an aggregative function, and some Pareto-optimum approaches: MOGA (Multi-Objective Genetic Algorithm, in which the rank of an individual corresponds to the number of individuals in the population by which it is dominated) [8], NSGA (Non-dominated Sorting Genetic Algorithm, which considers several layers of classification of individuals) [11], and NSGA-II (which improves NSGA with elitism and a crowded comparison operator to keeps diversity without additional parameters) [7].

The rest of the paper proceeds as follows: Section 2 and 3 are respectively devoted to state the two problems considered: parsing and tagging, presenting the main elements of a genetic program to solve parsing and the objective function for tagging; Section 4 presents and discusses the experimental results, and Section 5 draws the main conclusions of this work.

2 Evolutionary Statistical Parsing

To implement our EMO algorithm for parsing and tagging we have started from the genetic program for parsing, which produces the parse tree for a sentence, which in its turn determines the tagging of the sentence. The fitness function combines the probabilistic evaluation of the parse tree with the context based

[1] A solution $\mathbf{x} \in \mathcal{X}$ is a Pareto optimum (minimum) if there is no better solution $\mathbf{y} \in \mathcal{X}$ for some objective: $\nexists y \in \mathcal{X}$ such that
$f_i(y) \leq f_i(x) (\forall i = 1, \cdots, k)$ and
$f_j(y) < f_j(x)$ for some j.

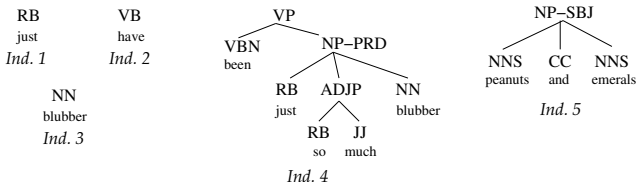

Fig. 1. Examples of individuals for the sentence *To her peanuts and emeralds would have been just so much blubber.*

evaluation of the tagging. This combination is performed in different ways for the different EMO models considered here.

A bottom-up parser starts with the sequence of lexical classes of the words and its basic operation is to take a sequence of symbols that match the right-hand side of the grammar rules. Stochastic grammars [3, 2], whose rules are assigned a probability, allow us evaluating the possible parses of a sentence, and thus solving ambiguity by selecting the most probable one. The stochastic grammar can be extracted from a linguistic corpus syntactically annotated (treebanks).

We have developed a probabilistic bottom-up parser based on genetic programming which works with a population of partial parses, i.e. parses of sentence segments.

2.1 Chromosome Representation

Individuals in our system are parses of segments of the sentence, that is, they are trees obtained by applying the probabilistic context free grammar (PCFG) to a sequence of words of the sentence. Each individual is assigned a syntactic category: the left-hand side of the top-level rule of the parse. The probability of this rule is also registered. Every tree is composed of a number of subtrees, each of them corresponding to the required syntactic category of the right-hand side of the rule. Figure 1 shows some individuals for the sentence *To her peanuts and emeralds would have been just so much blubber*, used as a running example, which has been extracted from the Penn Treebank [9]. We can see that there are individuals composed of a single word, such as *1*, while others, such as *4*, are a parse tree obtained by applying different grammar rules. For the former, the category is the chosen lexical category of the word (a word can belong to more than one lexical class); e.g. the category of *1* is *RB*. For the latter, the category is the left hand-side of the top level rule; e.g. the category of *4* is *VP*.

Initial Population The first step to parse a sentence is to find the possible lexical tags of each word. They are obtained, along with their frequencies, from a dictionary. Because parses are built in a bottom-up manner, the initial population is composed of individuals that are leave trees formed only by a lexical category of the word. The system generates a different individual for each lexical category of the word.

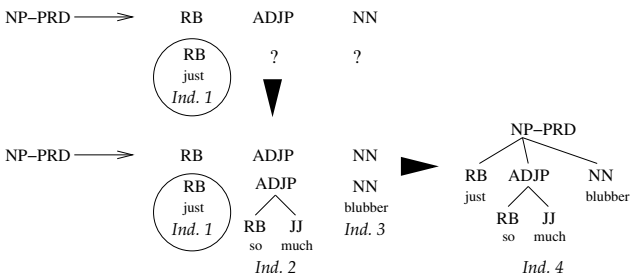

Fig. 2. Example of application of the crossover operator. Individual 1, whose syntactic category is RB, is randomly selected for crossover. The rule NP-PRD → RB ADJP NN is selected among those rules whose right-hand side begins with RB. Finally, the population is searched for individuals corresponding to the remaining syntactic categories of the rule, provided its sequence of words is appropriate to compose a segment of the sentence.

2.2 Genetic Operators

Chromosomes in the population of subsequent generations which did not appear in the previous one are created by means of two genetic operators: *crossover* and *cut*. The crossover operator combines a parse with other parses present in the population to satisfy a grammar rule; cut creates a new parse by randomly selecting a subtree from an individual of the population.

Crossover The crossover operator produces a new individual by combining an individual selected from the population with an arbitrary number of other ones. Notice that the crossover in this case does not necessarily occurs in pairs. The individuals to be crossed are randomly selected. This selection does not consider the fitness of the individuals because some grammar rules may require, to be completed, individuals of some particular syntactic category for which there are no representatives with higher fitness.

Let us assume that the individual *1* of Figure 1 is selected. The syntactic category (label of the root) of this individual is *RB*. The next step requires selecting among the grammar rules those whose right-hand side begins with this syntactic category, i.e *RB*. Some examples from the grammar used in this work are (NP-PRD → RB ADJP NN), (ADJP → RP JJ), (ADVP → RB RB), etc. Let us assume that we choose the first of these rules. Now, the crossover operator searches in the population for individuals whose syntactic category matches the remaining categories at the right-hand side of the rule, and whose sequence of words is the continuation of the words of the previous individual (Figure 2). In the example, we look for an individual of category *ADJP* and for another of category *NN*. The sequence of words of the individual of category *ADJP* must begin with the word *so*, the one following the words of individual *1*. Accordingly, the individual *2* of Figure 2 is a possible candidate (likewise, individuals *3* is also

chosen for the crossover). This process produces the individual *4* of Figure 2, whose syntactic category is the left-hand side of the rule (NP-PRD), and which is composed of the subtrees selected in the previous steps. This new individual is added to the population. With this scheme, the crossover of one individual may produce no descendant at all, or may produce more than one descendant. In the latter case all descendants are added to the population. The process of selection is in charge of reducing the population down to the specified size.

Crossover increases the mean size of the individuals every generation. Though this is advantageous because at the end we are interested in providing as solution individuals which cover the whole sentence, it may also induce some problems. If the selection process removes small individuals which can only be combined in later generations, the parses of these combinations will never be produced. This situation is prevented by introducing the *cut* operator, as well as by applying some constraints in the selection process, which reduces the size of the population by erasing individuals according to their fitness but always ensuring that each of their words is present in at least another individual. The cut operator produces a new individual out of another one by cutting off a subtree of its parse tree at random. The new individual is added to the population.

2.3 Objective Function for Parsing

The fitness function is basically a measure of the probability of the parse. It is computed as the average probability of the grammar rules used to construct the parse:

$$f_{par} = \frac{\sum_{s_i \in T} prob(s_i)}{nn(T)}$$

where T is the tree to evaluate, s_i each of its nodes and $nn(T)$ is the number of nodes. For the lexical category, the probability is the relative frequency of the chosen tag.

3 Objective Function for Tagging

A tagging of the considered sentence is a sequence of tags assigned to the words of the sentence. The model for tagging deals with disambiguation by assigning different probabilities to a given tag depending on which are the neighbouring tags (*context*) on both sides of the word. Figure 3 shows an example of sentence extracted from the Penn corpus. The word *questioning* can be disambiguated as a common name (NN) if the preceding tag is disambiguated as an adjective (JJR). But it might happen that the preceding word were ambiguous, so there may be many dependencies which must be resolved simultaneously. The computation of the fitness (f_{tag}) of a tagging is based on the list of contexts (*training table*) extracted from a training hand-tagged corpus. This table records the different contexts of each tag, along with its number of occurrences, for every tag in the training text. For example, if we consider contexts composed of two tags on the

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

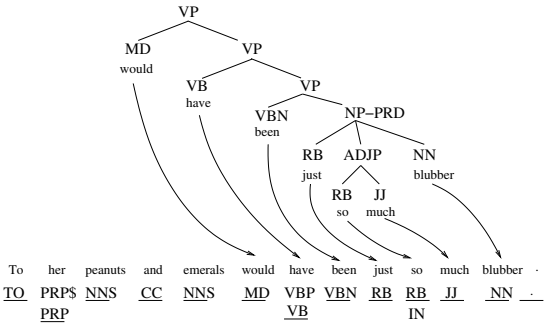

Fig. 4. Example of construction of the tagging of a sentence out the partial parse.

left and two tags on the right, the entry in the table for tag *JJ* (adjective) could have the form:

```
JJ 4557 9519

VBZ RB JJ IN PRP 37
IN DT JJ JJ NNS 20 ...
```

denoting that *JJ* has 4557 different contexts and appears 9519 times in the text, and that in one of those contexts, which appears 37 times, *JJ* is preceded by tags *VBZ* and *RB* and followed by *IN* and *PRP*, and so on until all the 4557 different contexts have been listed.

The genetic parser works with partial parses of the sentences. Thus the tagging obtained from an individual is a partial tagging of the whole sentence. We complete the tagging by choosing tags for the remaining words, randomly selected among the valid tags for the word (according to the dictionary), proportionally to their probabilities. Figure 4 shows an example. The tags selected to compose the tagging corresponding to the individual appear underlined. The tag PRP for the word *her* is randomly selected among those of the word (PRP$, possessive pronoun, and PRP, personal pronoun), while the tag VB (verb in base form) selected for *been* is given by the parse tree. The f_{tag} of an individual is a measure of the total 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 g 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.

The application of an aggregative function to the fitness functions computed for parsing and tagging is straightforward. We only have to take into account that both functions should vary within a similar range of values.

In NSGA, the computation of the sharing function for each front requires the computation of the phenotypic distance between two individuals. Because individuals in our system are partial parses, corresponding in general to different segments of the sentence, they can not be directly compared. Accordingly, a phenotypic distance is defined as the number of words which are parsed in one of the individuals but not in the other.

4 Experimental Results

The system, implemented on a PC in C++ language, has been applied to sentences extracted from two linguistic corpora: the Penn Treebank [9] and the Susanne corpus [10], both databases of English sentences manually annotated with syntactic information. The probabilistic grammar for parsing has also been obtained from the corpora. Each grammar rule is assigned a probability computed as its relative frequency with respect to other rules with the same left-hand side.

In order to evaluate the quality of the obtained parses we have used the most common measures for parsing evaluation, defined assuming a bracket representation [2] of the parse tree: *precision*[3] and *recall*[4].

From the Susanne corpus, we have used a training text set of 470082 words, and a test text of 520 words (17 sentences). From the Penn Treebank, we have taken a set of training text of 16233 words, using a test text of 269 words (11 sentences).

[2] For example, the parse tree for individual *5* of Figure 1 corresponds to the bracket expression NP-SBJ(NNS(peanuts), CC(and), NNS(emeralds)).

[3] Given by the number of brackets in the parse to evaluate matching those of the correct tree.

[4] A coverage measure given by the number of brackets in the correct tree which also appear in the parse.

4.1 Studing the sharing parameters

First of all, we have adjusted the parameters of the EMO algorithms considered in this paper. Table 1 shows the results obtained in both corpora for MOGA with different values of the sharing parameter. This is defined as a percentage of the distance between the best and worst values of the objective parsing. We can observe that the best results on both corpora are obtained with a sharing value of 30%. Table 2 shows the results obtained for NSGA with different values

Table 1. Study (tagging accuracy, precision and recall) of the sharing parameter for MOGA in Penn corpus (PS=450, IT=1500, %X=20, %C=20, TC=5) and in Susanne corpus (PS=200, IT=500, %X=20, %C=10, TC=3)

Sharing	Acc. Tag.	Prec.	Recall
10%	99.61	90.50	90.08
20%	99.61	91.72	89.16
30%	99.61	92.40	90.38
40%	96.12	89.54	92.14

Sharing	Acc. Tag.	Prec.	Recall
10%	99.61	98.04	96.56
20%	99.80	97.83	97.50
30%	99.61	99.74	98.46
40%	99.80	98.81	97.53

of the sharing parameter. In this case, individuals of a same niche are those of the same level which differ in a number of words below a threshold value. A sharing distance of 3 words provides the best results in this case. In the case of

Table 2. Study(tagging accuracy, precision and recall) of the sharing parameter for NSGA in Penn corpus (PS=450, IT=1500, %X=20, %C=20, TC=5) and in Susanne corpus (PS=200, IT=500, %X=20, %C=10, TC=3)

Sharing	Acc. Tag.	Prec.	Recall
1	96.51	90.66	91.15
2	96.51	90.66	91.15
3	99.61	91.98	91.34
4	99.61	87.37	87.23

Sharing	Acc. Tag.	Prec.	Recall
1	99.61	99.74	98.30
2	99.61	99.74	98.30
3	99.80	99.74	98.46
4	99.61	99.74	98.44

the aggregative function, we have also studied the use of different weights for the contributions to the fitness of parsing and tagging. Table 3 shows the results obtained in both corpora. In this case, the best results are obtained by assigning the same weight to both objective functions. A general observation is that the most significant results are obtained with the Penn Treebank. This is due to the large sets of lexical and syntactic tags used in the Susanne corpus, which lead to very specific grammar rules, for which the results are very similar in all cases.

Table 3. Study(tagging accuracy, precision and recall) of the coefficient for the aggregative function in Penn corpus (PS=450, IT=1500, %X=20, %C=20, TC=5) and in Susanne corpus (PS=200, IT=500, %X=20, %C=10, TC=3)

Coef.	Acc. Tag.	Prec.	Recall
1/2+1/2	99.61	89.90	87.93
1/3+2/3	99.61	87.91	89.80
2/3+1/3	100	87.92	87.10

Coef.	Acc. Tag.	Prec.	Recall
1/2+1/2	100	98.45	97.79
1/3+2/3	99.80	98.16	97.54
2/3+1/3	99.61	97.97	97.05

Table 4. Comparison of the results (tagging accuracy, precision and recall) obtained performing parsing separately and with different EMO algorithms. *Chart p.* stands for a classic chart parser algorithm, and *Wout tag.* for parsing without tagging with a genetic programming algorithm. The parameters of the EMO algorithms in Penn corpus (left table) are population size (PS)=450, iteration number (IT)=1500, crossover rate(%X)=20, cut rate(%C)=20) and in Susanne corpus (right table) (PS=200, IT=500, %X=20, %C=10)

Algorithm	Acc. Tag.	Prec.	Recall
Chart p.	99.22	88.66	85.81
Wout tag.	96.51	87.88	90.59
AF	99.61	89.90	87.93
MOGA	99.61	92.40	90.38
NSGA	99.61	91.98	91.34
NSGA-II	100	91.56	90.85

Algorithm	Acc. Tag.	Prec.	Recall
Chart p.	97.30	94.52	96.41
Wout tag.	99.61	83.44	81.82
AF	100	98.45	97.79
MOGA	100	99.74	98.46
NSGA	99.80	99.74	98.46
NSGA-II	99.61	99.23	99.23

4.2 Comparing the models

In order to compare the GP with a classic parser, we have implemented a classic *best-first chart parsing* (BFCP) algorithm. Table 4 shows the precision, recall, and tagging accuracy results obtained for each algorithm (figures represent the best result with respect precision out of ten independent runs). MOGA is run with a sharing parameter of 30%, NSGA with a sharing distance of 3 words, and the aggregative function with a linear combination of 0.5 + 0.5. We can observe that the results of any evolutionary algorithm improve on those of a classic chart parser. We can also observe that all methods provide similar results, though it is MOGA which provides the best results in precision (in particular for the Peen Treebank, for which there is larger differences for different methods). This can be due to the fact that this model allows tuning the weight of each objective function more precisely. MOGA distributes the population over the Pareto-optimal region by a niching method based on objective function values. The sharing distance for NSGA is defined as a function of the sentence words, what can be too coarse for tuning the algorithm. Concerning NSGA-II, the differences among the fitness value of each objective function for individuals in the same level (used to evaluate the crowding distance) can be little significant in some cases because individuals may parse different segments of the sentence.

5 Conclusions

We have tested different EMO algorithms for two related problems of natural language processing: parsing and tagging. Parsing can be regarded as a multi-objective optimization process in nature, since it amounts to searching the most probable combination of grammar rules as well as the most probable combination of lexical categories for the words of the sentence. The implementation of the tested EMO models (aggregative function, MOGA, NSGA and NSGA-II) has been adapted to our particular problems, defining specific sharing parameters and distances, according to the structure of our individuals.

The most relevant result is the improvement achieved by any of the chosen EMO models with respect the resolution of each problem separately, regardless of whether a classic exhaustive search method or an evolutionary algorithm is applied to the isolated problems.

The comparison of the applied EMO models has revealed that all of them provide similar results, though MOGA performs a bit better. We now plan to test other EMO models for these two problems.

References

1. L. Araujo. Part-of-speech tagging with evolutionary algorithms. In *Proc. of the Int. Conf. on Intelligent Text Processing and Computational Linguistics (CICLing-2002), Lecture Notes in Computer Science 2276*, pages 230–239. Springer-Verlag, 2002.
2. L. Araujo. Genetic programming for natural language parsing. In *Proc. of the European Conference on Genetic Programming (EuroGP2004), Lecture Notes in Computer Science 3003*, pages 230–239. Springer-Verlag, 2004.
3. E. Charniak. *Statistical Language Learning.* MIT press, 1993.
4. C. A. Coello Coello, D. A. Van Veldhuizen, and G. B. Lamont. *Evolutionary Algorithms for Solving Multi-Objective Problems.* Kluwer Academic Publishers, 2002.
5. M. Dalrymple. How much can tagging help parsing? Technical report, Department of Computer Science, King's College London, 2004.
6. K. Deb and D. Kalyanmoy. *Multi-Objective Optimization Using Evolutionary Algorithms.* John Wiley & Sons, 2001.
7. K. Deb, A. Pratab, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on Evolutionary Computations*, 6(2):182–197, 2002.
8. Carlos M. Fonseca and Peter J. Fleming. Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization. In *Genetic Algorithms: Proceedings of the Fifth International Conference*, pages 416–423. Morgan Kaufmann, 1993.
9. Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19(2):313–330, 1994.
10. G. Sampson. *English for the Computer.* Clarendon Press, Oxford, 1995.
11. N. Srinivas and Kalyanmoy Deb. Multiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary Computation*, 2(3):221–248, 1994.