# Evolving Natural Language Grammars without Supervision

Lourdes Araujo, Jesús Santamaría

*Abstract*— **Unsupervised grammar induction is one of the most difficult works of language processing. Its goal is to extract a grammar representing the language structure using texts without annotations of this structure. We have devised an evolutionary algorithm which for each sentence evolves a population of trees that represent different parse trees of that sentence. Each of these trees represent a part of a grammar. The evaluation function takes into account the contexts in which each *sequence of Part-Of-Speech tags* (POSseq) appears in the training corpus, as well as the frequencies of those POSseqs and contexts. The grammar for the whole training corpus is constructed in an incremental manner. The algorithm has been evaluated using a well known Annotated English corpus, though the annotation have only been used for evaluation purposes. Results indicate that the proposed algorithm is able to improve the results of a classical optimization algorithm, such as EM (Expectation Maximization), for short grammar constituents (right side of the grammar rules), and its precision is better in general.**

## I. Introduction and state of the art

Grammar inference (GI) in natural language processing amounts to extracting a representation of the language structure from a collection of texts or corpora. If this collection has been manually annotated with the parsing of the sentences, i.e. it is a treebank, the task is termed supervised GI, otherwise it is referred to as unsupervised GI. GI is a highly complex problem, even in the supervised case. Regular grammars (RG) cannot be correctly identified from positive examples alone [1], since exclusively inferring from positive examples leads to over-generalization, so that, the grammar obtained successfully parses the training examples but also accepts many illegal examples. Inferring context free grammars (CFG) offers additional challenges due to *undecidable decision problems*[1], e.g. for two given CFGs, G1 and G2, there exists no algorithm that can determine whether G1 is more general than G2 [5]. All these problems found in formal language GI are also present in the much more complex case of natural language GI.

In spite of the above mentioned difficulties several statistical and heuristic approaches have been devised for grammar induction, specially for the case of supervised GI [6], [7], [8], [9], [10]. In this case the grammar rules can be automatically extracted from the treebank, which provides information of statistical preference, and in the case of considering stochastic grammars, i.e. grammars with probabilities added to their rules, the treebank also allows the estimation of such probabilities.

Lourdes Araujo and Jesús Santamaría are in Dept. de Lenguajes y Sistemas Informáticos. UNED, Madrid, Spain (email (lurdes|jsant)@lsi.uned.es)

[1]A good introduction to the theoretical problems related to the induction of formal grammars can be found in [2], [3], [4].

The availability of a treebank with syntactic annotations is, obviously, a great help to the GI. However these treebanks are not available in many languages since the manual annotation of thousands of sentences is a very expensive and time-consuming process. Furthermore, the grammar inferred from a collection of texts usually depends on the type of those texts (newspapers, tales, poetry, etc.). So different types of treebanks are required even for the same language. These reasons have led some researchers to pay attention to the unsupervised GI problem. Some of these works have focused on finding patters of words [11] more than syntactic structures. There are other works whose goal is to identify sequences of lexical units or Part-Of-Speech (POS) tags which represent equivalence classes covering substitutable units. Systems adopting this approach aim at identifying for each sentence non contradictory sequences of POS tags which maximize a given statistical model, usually the parsing probability. These systems apply the Expectation-Maximization (EM) algorithm [12] to perform the optimization. EM is an iterative method which applies two steps at each iteration. The first one is the expectation (E) step, which computes an expectation of the likelihood with respect to the current estimate of the distribution for the hidden variables. The second one is a maximization (M) step, which computes the parameters which maximize the expected likelihood found on the E step. These parameters are used to determine the distribution of the hidden variables in the next E step. Zaanen [13] proposed the Alignment Based Learning algorithm in which the EM algorithm is directly applied to raw text, without POS tags, as other systems do. This system uses minimal string edit distance between sentences to propose units, from which the most probable combination is chosen. The system has only been applied to small corpora. The work by Clark [14] is also within this category. The proposed algorithm learns a phrase-structure grammar from tagged texts by clustering sequences of tags together based on local distributional information, and selects clusters that satisfy a mutual information criterion related to the entropy of a random variable associated with the tree structures. Paskin [15] proposes a model to reduce the complexity of the EM algorithm to $O(n^3)$. It is based upon grammatical bigrams, i.e. syntactic relationships between pairs of words and introduce new independence assumption into the traditional models. However, the quality of the results for the completely unsupervised case is low. One of the most successful proposals in this area is the one by Klein and Manning [16]. It starts from a corpus labeled only with POS tags. The key idea of the model proposed in

978-1-4244-8126-2/10/$26.00 ©2010 IEEE

this work is that constituents[2] appear in constituent contexts. In particular, the model exploits the fact that long constituents often have short, common equivalents, which appear in similar contexts and whose constituency as a grammar rule is more easily found. The model aims at transferring the constituency of a sequence directly to its containing context. The context tends to contain new sequences that can be identified as constituents in the next step of the iterative identification process. The authors have devised a statistical model according to these ideas and then applied the EM algorithm to search for the set of POSseqs (sequences of POS tags) that optimize the fit of the probabilistic model to the data. However, the EM algorithm presents some serious problems: it is very slow[3], and is easily trapped in local maxima. Lari and Young [17] found that applying EM to recover very simple context-free grammars, the learned grammar would require several times the number of non-terminals to recover the structure of a target grammar, and in many cases the grammar obtained was a quite different variant of the target grammar. Carroll and Charniak (1992) [18] show the high tendency of this algorithm to be trapped in local optima performing several experiments in which they run the EM algorithm from random starting point, and obtaining a different and poor quality grammar in every case.

There have been several works applying evolutionary algorithms to the GI problem [19]. Many of them have focused on formal languages [20], [21], [22], [23], [24], which provide a more accessible benchmark to investigate. Some other [25], [26], [27] have been devised for very simple fragment of the language in the form of a small set of simple sentences. There have been also proposals dealing with real language [28], [29], though for a particular aspect of it. Araujo and Serrano [29] have applied EAs for natural language GI restricted to particular components that are noun phrases (NPs). De Pauw [28], [30] has developed the GRAEL system, an agent-based evolutionary technique for induction and optimization of grammars for natural language, able to generate new grammar rules in a guided manner.

We have adopted Klein and Manning's ideas, i.e. the constituency character is mutually assessed by the sequence of tags composing it and the context in which the sequence appears. We do consider, as in the above mentioned work, sequences of POS tags. It is not a hard assumption since there are very accurate POS taggers (about 97%) for many languages. We investigate how to use an evolutionary algorithm instead of the EM algorithm to fit the training data to the model, thus circumventing the problems that EM presents. Our evolutionary algorithm works with a population of parse trees for each sentence. The parse tree selected at the end of the evolution represents a part of the grammar, which is thus built in an incremental manner. The algorithm evolves the population selecting those POSseqs which optimize the presence of some sequences and forms of the parse tree extracted from the training data. We do not assign a type to the POSseqs but they are clustered on classes of units that can be placed in similar contexts. Experiments have been carried out using part of the Penn Treebank [31] for English, but the annotations are only used for evaluation purposes, not for training. For the evaluation we have used the usual measures in grammar induction, such as recall and precision, which measure different aspects of the similarity between the parse trees given by the evaluated system and the reference tree provided by a human expert and which is part of the corpus.

The rest of the paper is organized as follows: section II presents the evolutionary algorithm implemented, describing the adopted individual representation, the fitness function and the different genetic operators; section III is devoted to the experimental setup; section IV presents and discusses the results of the experiments, and section V draws the main conclusions of this work.

## II. The Evolutionary Algorithm

TABLE I
ALPHABETICAL LIST OF PART-OF-SPEECH TAGS USED IN THE PENN TREEBANK, THE CORPUS USED IN OUR EXPERIMENTS

| | |
|---|---|
| CC | Coordinating conjunction |
| CD | Cardinal number |
| DT | Determiner |
| EX | Existential there |
| FW | Foreign word |
| IN | Preposition / subordinating conjunction |
| JJ | Adjective |
| JJR | Adjective, comparative |
| JJS | Adjective, superlative |
| LS | List item marker |
| MD | Modal |
| NN | Noun, singular or mass |
| NNS | Noun, plural |
| NNP | Proper noun, singular |
| NNPS | Proper noun, plural |
| PDT | Predeterminer |
| POS | Possessive ending |
| PRP | Personal pronoun |
| PP$ | Possessive pronoun |
| RB | Adverb. |
| RBR | Adverb, comparative |
| RBS | Adverb., superlative |
| RP | Particle |
| SYM | Symbol (mathematical or scientific) |
| TO | To |
| UH | Interjection |
| VB | Verb, base form |
| VBD | Verb, past tense |
| VBG | Verb, gerund / present participle |
| VBN | Verb, past participle |
| VBP | Verb, non-3rd ps. sing. present |
| VBZ | Verb, 3rd ps. sing. present |
| WDT | wh-determiner |
| WP | wh-pronoun |
| WP$ | Possessive wh-pronoun |
| WRB | wh-adverb |

Our EA extracts the grammar representing the language structure in an incremental manner, considering sentences one by one. For a given sentence the algorithm creates a

[2]Constituents are language units in which we can arrange the structure of a sentence.

[3]It is $\Omega m^3 n^3$, where $m$ is the length of the sentence, and $n$ is the number of non-terminal in the grammar

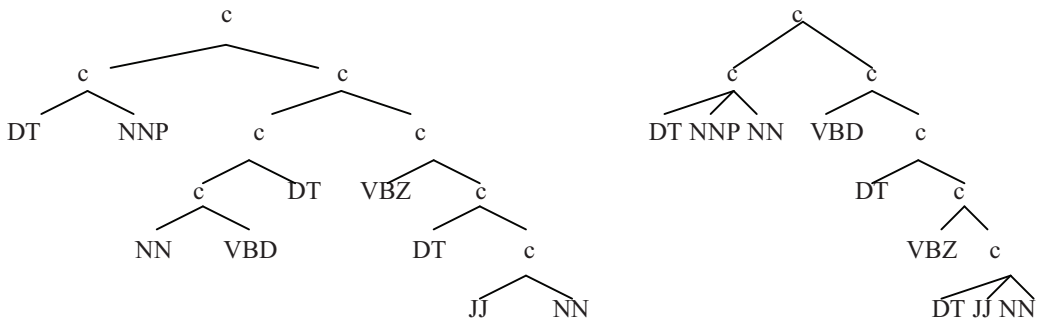



Fig. 1. Example of two possible individuals for the sentence with POS tags DT NNP NN VBD DT VBZ DT JJ NN

population of individuals which are possible parse trees for that sentence, i.e. trees which have as leaves the whole sequence of POS tags of the sentence. Figure 1 shows two possible individuals for the sentence DT NNP NN VBD DT VBZ DT JJ NN. The meaning of the POS tags appearing in the examples are shown in Table I. The branching of an individual determines the sequences of tags which are considered to be constituent (i.e. the grammar rules without specifying the left hand side) and its contexts (the tag on the left and on the right). Note the difference between POSseq and constituent:

- A POSseq is any sequence of POS tags appearing in the corpus.
- A constituent is a sequence of POS tags grouped together in the parse of a sentence, i.e. the left hand side of some grammar rule. In the tree on the left of Figure 1 we can distinguish the following constituents: DT, NNP, NN, VBD, DT, VBZ, DT, JJ, NN, (DT, NNP), (NN, VBD), (NN, VBD, DT), (JJ, NN), (DT,JJ,NN), (VBZ, DT, JJ, NN), (NN, VBD, DT, VBZ, DT, JJ, NN) and (DT, NNP, NN, VBD, DT, VBZ, DT, JJ, NN). That is, they are the sequences of tags under any *c* symbol plus each POS tag.

Figure 2 shows the constituents for the individual on the left.

The initial population is created at random, with the unique condition of having only legal trees for the sentence, i.e. a graph in which two vertices are connected by exactly one path, and with the whole sequence of the sentence tags as leaves.

### *A. The Fitness Function*

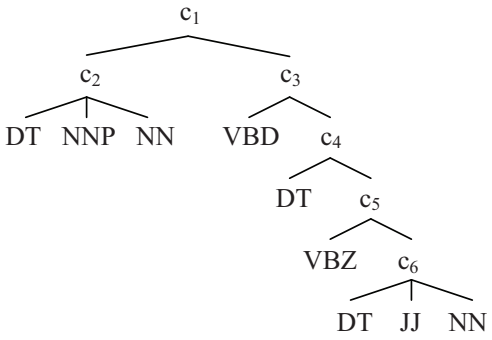
Our fitness function tries to capture the ideas of Klein and Manning's model described above, as well as some other observations on the form of the parse trees, and adopts the form of a linear combination of different aspects of the model. Specifically we have considered the following factors:

- *Constituents appear in constituent contexts*. We have observe that there exists some POS-tags which trigger new levels in the parse tree. For example, in the tree on the right of Figure 1 this is the case of VBD, DT and VBZ. These kind of POS-tags, that we call *separators* are the beginning of a constituent which finishes at the end of the last level created. In order to identify the set of possible separators without using the parse trees, i.e. without supervision, we have considered the more frequent POSseqs in our corpus:

| | |
|---|---|
| DT NN | 2222 |
| JJ NN | 1352 |
| IN DT | 894 |
| NN IN | 892 |
| DT JJ | 834 |
| JJ NNS | 797 |
| NN NN | 795 |
| CD CD | 776 |
| NN VBZ | 760 |
| ... | ... |

We can observe that the first two sequences have a frequency clearly higher than the rest of them. Accordingly we consider that these sequences are frequent enough to be safely taken as constituents. We call them *discriminant constituent* (DC). Extensions of the DCs are POSseqs starting and ending with the same tags (or their variants NNS, NNP, JJR, etc.) than a DC, such as (DT JJ NNS), (DT JJ NNS NN), (JJ, JJR, NNP), etc. Taking this assumption, we identify as separators those POS-tags which appear more frequently (more than 75%) outside the DCs or their extensions than inside. We have used the corpus is WSJ10, composed of 6842 sentences. WSJ10 is a subset of the Wall Street Journal section of the Penn Treebank, containing only those sentences of 10 words or less after removing punctuation and null elements, such as $, ", etc.. Applying the above described criterion to this corpus we have obtained the following sets of separators for each DC:

| DC | Separators |
|---|---|
| DT NNx | MD, PRP, IN, RB, RBR, CC, TO, VB, VBD, VBN, VBZ, VBP, VBG, EX, LS, RP, UH, WP, WRB |
| JJx NNx | CC, EX, FW, IN, LS, MD, PDT, PRP, RB, RBR, RBS, RP, SYM, TO, UH, VB, VBD, VBG, VBN, VBP, VBZ, WDT, WP, WRB |




| Node | Constituent |
|---|---|
| $c_1$ | $c_2$ $c_3$ |
| $c_2$ | DT NNP NN |
| $c_3$ | VBD $c_4$ |
| $c_4$ | DT $c_5$ |
| $c_5$ | VBZ $c_6$ |
| $c_6$ | DT JJ NN |

Fig. 2. Constituents of the individual on the left

We have included in the fitness function a factor SEP for each POSseq $ps$ found in the sentence, which increases the fitness if $ps$ is composed of a separator followed by a sequence of POS-tags arriving to the end of the sentence ($es(ps)$). The value of the factor SEP for a POSseq $ps$ depends on the set of separators to which the first tag (ft(ps)) of the sequence belongs to:

$$SEP(ps) = \begin{cases} 3 & \text{if } es(ps) \text{ and} \\ & ft(ps) \in separ(DT, NN) \text{ and} \\ & ft(ps) \in separ(JJ, NN) \\ 2 & \text{if } es(ps) \text{ and} \\ & ft(ps) \in separ(DT, NN) \\ 1 & \text{if } es(ps) \text{ and} \\ & ft(ps) \in separ(DT, NN) \\ 0 & \text{otherwise} \end{cases}$$

- *Long constituents often have short, common equivalents, which appear in similar contexts and whose constituency as a grammar rule is more easily found.* To capture this idea we have considered again the discriminant constituents and their extensions, *extend(X,Y)*. The fitness function is increased with a factor CONS for each POSseq $ps$ corresponding to a DC or one of their extensions. The value of CONS depends on the DC:

$$CONS(ps) = \begin{cases} 2 & \text{if } ps \in extend(DT, NN) \\ 1 & \text{if } ps \in extend(JJ, NN) \\ 0 & \text{otherwise} \end{cases}$$

Then, the fitness function takes the following form:

$$\sum_{ps \in tree} SEP(ps) + CONS(ps)$$

where $ps$ represents each POSseq in the parse tree of the individual.

### B. Genetic Operators

This section presents the crossover and mutation operators used in the EA: one crossover operator, and three mutation operators (called *ungrouping*, *grouping*, and *group-ungroup*).

*1) Crossover:* We can not apply the most classical crossover operators used in genetic programming to combine trees since we have to guarantee that the resulting offspring are legal parse tree for the sequence of tags of the sentence. Therefore we have designed a specific operator that combines trees taking the constituents of a range of lengths from one parent and the constituents of the remaining range from the other parent. For a sentence of length $n$ the possible constituents range from 1 to n. At this point, a random integer $1 \leq p \leq n-1$ is generated within this range. The first offspring is then composed of the constituents of length between 1 and $p-1$ from the first parent and of the constituents of length between $p$ and $n$ from the second parent. Analogously, the second offspring is composed of the constituents of length between $p$ and $n$ from the first parent and the other ones from the second parent.

Figure 3 shows an example with $p = 2$. In this case, the sentence under consideration is[4] DT NN NN IN JJ NNS. We can see that the crossover of the parent on the left (called PA) with the one on the right (called PB), produces the individuals on the bottom (called CA and CB). CA takes the constituents of size 1 and 2 from PA, but PA has not constituents of size 2 so it does not provide any constituent to CA. PB, which provides to CA constituents of length 3 and higher, has only one constituent of size 3 (IN JJ NNS), which is therefore given to CA. The sentence tags not included in the constituents provided by any parent are taken as one-tag constituents. The other offspring, CB, is the result of crossing PB with PA, taken the constituents of size 2 from PB (JJ NNS) and those of size 3 from PA (DT NN NN and IN JJ NNS). By the way, this tree SB is the correct one according to the linguists' judgements contained in the Penn treebank.

Offspring generated may not be valid trees with regard to the sentence being considered. For example, let us consider the individuals in Figure 4 and assume that $p$ has taken the value 2. Then, the parent of the left will provide the constituents of size 2 or smaller: (DT,NN), (NN,IN) and (JJ,NNS), for the first offspring and the parent on the right

[4]Lexical labels of this example correspond to the sentence: *an insurance company with financial problems*

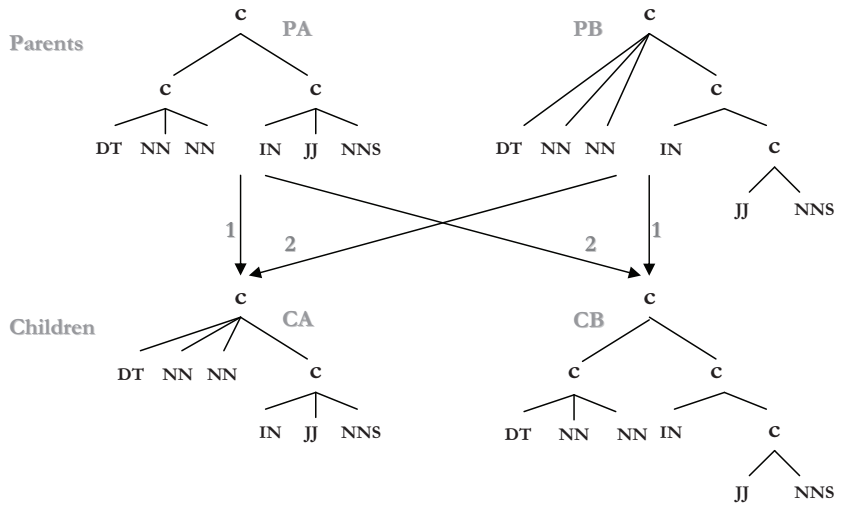



Fig. 3. Crossover example

will provide the constituents of size 3 or larger: (IN, JJ, NNS), and then there is a conflict for the tag IN at the end of the second constituent from the first parent, and at the beginning of the constituent from the second parent. These cases are solved by randomly taking one of the constituents.

*2) Mutation:* Our system implements three mutation operators, aiming respectively at enabling the ungrouping of tags, the grouping of them, and also a change in the way of grouping them:

1) **Ungrouping**: Changes a randomly chosen constituent of N tags by N constituents of only one tag.
2) **Grouping**: Searches the POSseqs which do not lead to conflict if they are made constituents, and randomly selects one of them.
3) **Group-Ungroup**: It randomly chooses a constituent size for which there are some constituent in the tree. Then randomly chooses one of the constituents of the selected size to be ungrouped. At the same time the operator randomly chooses a POSseq of the selected size which does not lead to conflict if it becomes a constituent, and grouped it as a constituent.

The mutation operator which is applied to each individual is randomly selected among the three ones described above.

## III. Experimental Setup

### A. The Evaluation Method

The methodology used to evaluate our system (in addition to qualitative assessments by inspection) is to compare the tree structures produced by the system to the gold-standard trees produced by linguists which can be found in the Penn Treebank. Because we do not assign class name to our constituents, i.e. a left hand side symbol for the grammar rules, as the linguists do in treebanks, the comparison ignores the class labels considering only groups of tags.

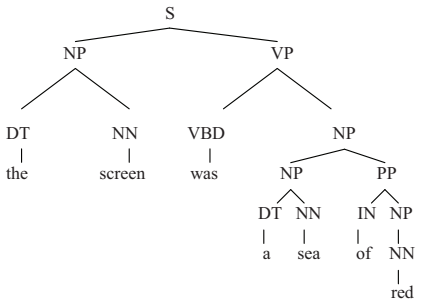



Fig. 5. Parse tree produced by a linguist

We performed the experiments on the 6842 sentences[5] of the WSJ10 selection from the Penn treebank Wall Street Journal section. Notice that because we work with an unsupervised process, we can use the same sources to obtain both the statistical values as well as the test data.

In order to evaluate the quality of the obtained grammar we have used the most common measures for parsing and grammar induction evaluation: recall, precision and their harmonic mean F1. They are defined assuming a bracket representation of a parse tree. For example, the bracket representation for the sentence in Figure 5 is
S(NP(DT, NN), VP(VBD, NP(NP(DT, NN),PP(IN, NP(NN)))))

While the unlabeled bracket representation for the same sentences is
c(c(DT, NN), c(VBD, c(c(DT, NN),c(IN, c(NN)))))

*Precision* is given by the number of brackets in the parse to evaluate which match those in the correct tree and *recall* measures how many of the brackets in the correct tree are

[5]More precisely sequences of POS tags

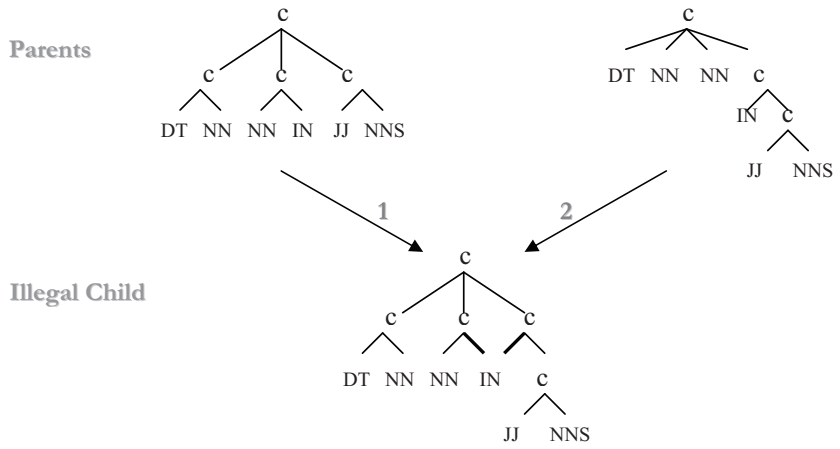



Fig. 4. Illegal Crossover example

in the parse. These measures have a correspondence for unlabeled trees, which have been the ones considered in this work, in which the label assigned to each POSseq is not checked. Constituents which could not be wrong (those of size one and those spanning the whole sentence) have not been included in the measures.

The definitions of Unlabeled Precision (UP) and Recall (UR) of a proposed corpus $P = [P_i]$ against a gold corpus $G = [G_i]$ are:

$$UP(P,G) = \frac{\sum_i |\text{brackets}(P_i) \cap \text{brackets}(G_i)|}{\sum_i |\text{brackets}(P_i)|},$$
$$UR(P,G) = \frac{\sum_i |\text{brackets}(P_i) \cap \text{brackets}(G_i)|}{\sum_i |\text{brackets}(G_i)|}.$$

Finally, $F_1$ is given by:

$$F_1 = \frac{2 \cdot UP(P,G) \cdot UR(P,G)}{UP(P,G) + UR(P,G)}.$$

## IV. Experimental results

After a number of experiments, for which the results are shown in Figure 6 we have taken the parameter set appearing in Table II. The results are quite stable for a population size of 150 individuals and 50 generations. The best results are obtained for a crossover rate of 0.65 and a mutation rate of 0.06, respectively. A run with these parameters takes about 27 hours.

The results obtained by our system for different constituent sizes are shown in Table III and are graphically represented in Figure 7. They have been obtained with the parameters appearing in Table II. We can observe that UR follows an almost continuous trend to rise with the size of the constituents, reaching 98.34% for constituents of size 9. However, Precision is higher for constituent sizes 3 and 4, becoming lower for longer constituents. F1 maintains the level for all the constituent size, which is a nice property.

Figure 8 compares our results with those of the Klein and Manning system (CCM) which uses the EM algorithm for optimization, and which, as far as we know, has provided the best results for unsupervised GI using a monolingual corpus.

TABLE II
Genetic Algorithm parameter set

| Population size | Max. generations | Crossover rate | Mutation rate | Elite size |
|---|---|---|---|---|
| 150 | 50 | 0.65 | 0.06 | 1 |

TABLE III
Results obtained for each constituent size. $UR$ stands for unlabeled recall, $UP$ for unlabeled precision, and $F_1$ for the harmonic mean.

| Constituent Size | $UR$ (%) | $UP$ (%) | $F_1$ (%) |
|---|---|---|---|
| 2 | 64.42 | 58.32 | 61.22 |
| 3 | 63.53 | 79.30 | 70.62 |
| 4 | 61.68 | 71.52 | 66.24 |
| 5 | 63.73 | 65.00 | 64.36 |
| 6 | 68.12 | 57.17 | 62.16 |
| 7 | 74.78 | 58.08 | 65.38 |
| 8 | 80.07 | 54.10 | 64.57 |
| 9 | 98.34 | 61.32 | 75.54 |
| Total | 65.60 | 64.21 | 64.89 |

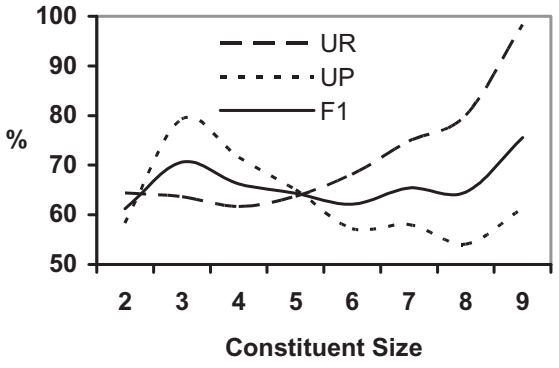



Fig. 7. Results obtained vs. constituent size. *UP*: Unlabeled Precision, *UR*: Unlabeled Recall, $F_1$: Harmonic Mean of UP and UR.

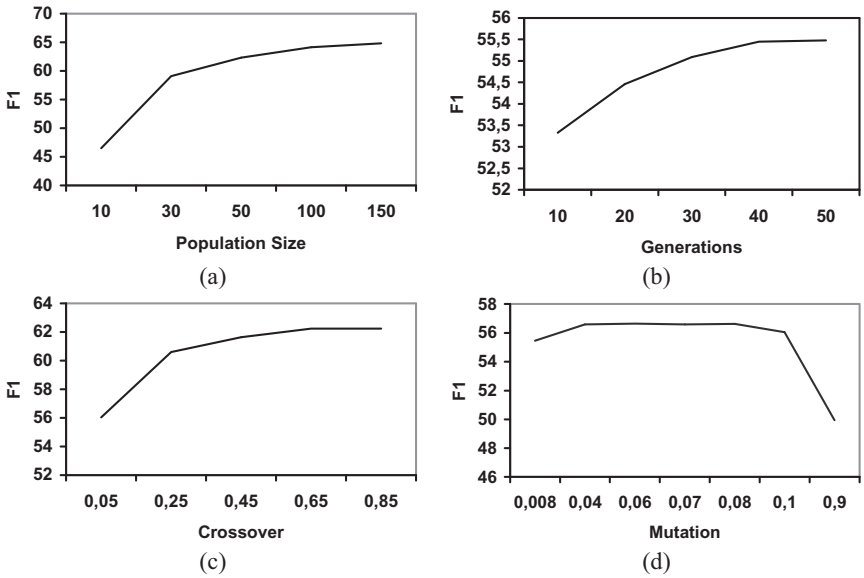



Fig. 6. Results of F1 obtained for (a) different population sizes, (b) number of generations, (c) crossover rates, and (d) Mutation rates. Default parameters appear in Table II

We can observe that for short and medium size constituents (2, 3, 4, 5, and 6) our evolutionary system performs better than the one using EM, though for large constituents (7, 8, and 9) the F1 results are a bit higher for CCM. Concerning precision, we can also observe that our results are much better than those of CCM for short constituents. We consider this a promising result since the identification of short constituents, which appear very frequently, is particularly relevant. Moreover we are currently working on considering other discriminant constituents conditioned to the presence of other elements in the parse tree being evaluated.

## V. Conclusions

This work has investigated a particular task of language learning: unsupervised grammar extraction by induction. The system does not require to be trained with a parsed corpus and thus it can be applied to any language or to any particular kind of texts. We have applied an EA to select the most probable grammar for a given corpus according to a model that takes into account both the frequencies of the sequences of POS tags and the contexts in which they appear, as well as the relationships between both data. The system performance is high for short constituents but decreases a bit the longest constituents. We are currently working to palliate this effect by introducing new factors to the fitness functions and tuning the weights assigned to each of them.

We also plan to test the system on other corpora and languages, and well as to consider other individual representations that can improve the system efficiency.

## Acknowledgements

This paper has been funded in part by the Spanish MICINN projects NoHNES (Spanish Ministerio de Educación y Ciencia - TIN2007-68083) and QEAVis-Catiex (TIN2007-67581-C02-01), as well as by the Regional Government of Madrid under the Research Network MA2VICMR (S2009/TIC-1542).

## References

[1] E. M. Gold, "Language identification in the limit," *Information and Control*, vol. 10, no. 5, pp. 447–474, 1967.

[2] K. S. Fu and T. L. Booth, "Grammatical inference: introduction and survey_part i," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 8, no. 3, pp. 343–359, 1986.

[3] ——, "Grammatical inference: introduction and survey_part ii," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 8, no. 3, pp. 360–375, 1986.

[4] R. Parekh and V. Honavar, "Grammar inference, automata induction, and language acquisition," in *Handbook of natural Language processing*, Dale, Moisle, and Somers, Eds. Marcel Dekker Inc., 2000. [Online]. Available: citeseer.ist.psu.edu/rajesh00grammar.html

[5] J. E. Hopcroft and J. D. Ullman, *Introduction To Automata Theory, Languages, And Computation.* Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1990.

[6] E. Charniak, "Statistical parsing with a context-free grammar and word statistics," in *Proc. of the 14th National Conference on Artificial Intelligence*. AAAI Press/MIT Press, 1997, pp. 598–603.

[7] ——, "A maximum-entropy-inspired parser," in *Proceedings of the first conference on North American chapter of the Association for Computational Linguistics*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2000, pp. 132–139.

[8] M. Collins, "Three generative, lexicalised models for statistical parsing," in *Proc. of the Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 1997, pp. 16–23.

[9] ——, *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. Dissertation, University of Pennsylvania, 1999.

[10] A. Ratnaparkhi, "A linear observed time statistical parser based on maximal entropy models," in *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing*, C. Cardie and R. Weischedel, Eds. Somerset, New Jersey: Association for Computational Linguistics, 1997, pp. 1–10. [Online]. Available: citeseer.ist.psu.edu/ratnaparkhi97linear.html

[11] Z. Solan, D. Horn, E. Ruppin, and S. Edelman, "Unsupervised learning of natural languages," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 102,

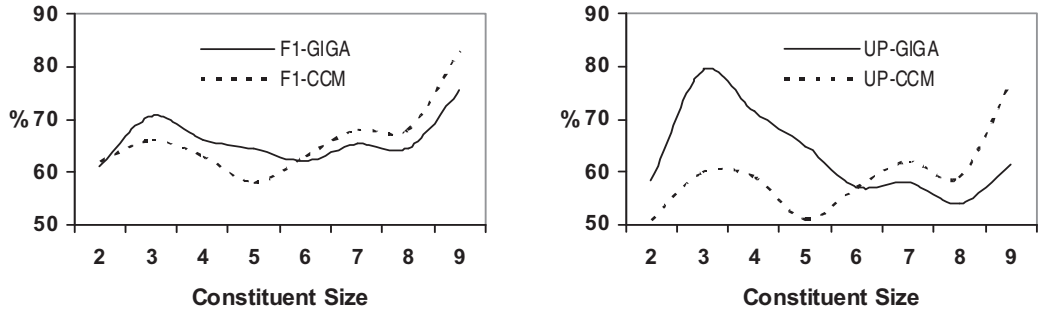



Fig. 8. Comparison between Klein & Manning's system (CCM) and our system (GIGA) with respect to $F_1$ (a) and to Precision (b)


no. 33, pp. 11 629–11 634, August 2005. [Online]. Available: http://dx.doi.org/10.1073/pnas.0409746102

[12] A. Dempster, N. Laird, and D. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Royal statistical Society B*, vol. 39, pp. 1–38, 1977.

[13] M. van Zaanen and L. J. Leeds, "Learning structure using alignment based learning," in *Universities of Brighton and Sussex*, 2000, pp. 75–82.

[14] A. Clark, "Unsupervised induction of stochastic context-free grammars using distributional clustering," in *ConLL '01: Proceedings of the 2001 workshop on Computational Natural Language Learning*. Morristown, NJ, USA: Association for Computational Linguistics, 2001, pp. 1–8.

[15] M. A. Paskin, "Grammatical bigrams," in *Advances in Neural Information Processing Systems 14*. MIT Press, 2001.

[16] D. Klein and C. D. Manning, "Natural language grammar induction with a generative constituent-context model," *Pattern Recognition*, vol. 38, no. 9, pp. 1407–1419, 2005.

[17] K. Lari and S. J. Young, "The estimation of stochastic context-free grammars using the inside-outside algorithm," *Computer Speech and Language*, vol. 4, pp. 35–56, 1990.

[18] G. Carroll and E. Charniak, "Two experiments on learning probabilistic dependency grammars from corpora," in *Working Notes of the Workshop Statistically-Based NLP Techniques*. AAAI, 1992, pp. 1–13.

[19] L. Araujo, "How evolutionary algorithms are applied to statistical natural language processing," *Artif. Intell. Rev.*, vol. 28, no. 4, pp. 275–303, 2007.

[20] P. J. Wyard, "Context free grammar induction using genetic algorithms." in *ICGA*, 1991, pp. 514–519.

[21] M. M. Lankhorst, "Breeding Grammars. Grammatical Inference with a Genetic Algorithm," Dept. of CS. University of Groningen, Groningen, The Netherlands, Technical Report, 1994.

[22] T. E. Kammeyer and R. K. Belew, "Stochastic context-free grammar induction with a genetic algorithm using local search." in *FOGA*, 1996, pp. 409–436.

[23] B. Keller and R. Lutz, "Evolving stochastic context-free grammars from examples using a minimum description length principle," in *Workshop on Automata Induction Grammatical Inference and Language Acquisition, ICML-97.*, 1997. [Online]. Available: citeseer.ist.psu.edu/59698.html

[24] ——, "Evolutionary induction of stochastic context free grammars." *Pattern Recognition*, vol. 38, no. 9, pp. 1393–1406, 2005.

[25] T. Smith and I. Witten, "A genetic algorithm for the induction of natural language grammars," in *Proc. IJCAI-95 Workshop on New Approaches to Learning Natural Language*, Montreal, Canada, 1995, pp. 17–24.

[26] R. M. Losee, "Learning syntactic rules and tags with genetic algorithms for information retrieval and filtering: An empirical basis for grammatical rules," *Information Processing and Management*, vol. 32, no. 2, pp. 185–197, 1996. [Online]. Available: citeseer.ist.psu.edu/losee00learning.html

[27] E. E. Korkmaz and G. Ucoluk, "Genetic programming for grammar induction," in *2001 Genetic and Evolutionary Computation Conference Late Breaking Papers*, E. D. Goodman, Ed., San Francisco, California, USA, 9-11 2001, pp. 245–251. [Online]. Available: citeseer.ist.psu.edu/korkmaz01genetic.html

[28] G. De Pauw, "Evolutionary computing as a tool for grammar development." in *Proceedings of GECCO 2003, LNCS 2723*. Berlin Heidelberg: Springer-Verlag, 2003, pp. 549–560. [Online]. Available: http://www.cnts.ua.ac.be/Publications/2003/De03f

[29] L. Araujo and J. I. Serrano, "Highly accurate error-driven method for noun phrase detection," *Pattern Recognition Letters*, vol. 29, no. 4, pp. 547–557, 2008.

[30] G. De Pauw, "Grael: An agent-based evolutionary computing approach for natural language grammar development." in *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence*, Acapulco, Mexico, 2003, pp. 823–828. [Online]. Available: http://www.cnts.ua.ac.be/Publications/2003/De03d

[31] M. P. Marcus, B. Santorini, and M. A. Marcinkiewicz, "Building a large annotated corpus of english: The penn treebank," *Computational Linguistics*, vol. 19, no. 2, pp. 313–330, 1994. [Online]. Available: citeseer.ist.psu.edu/marcus93building.html