eg-GRIDS: Context-Free Grammatical Inference from Positive Examples using Genetic Search

Georgios Petasis[1], Georgios Paliouras[1], Constantine D. Spyropoulos[1], and Constantine Halatsis[2]

[1] Software and Knowledge Engineering Laboratory,
Institute of Informatics and Telecommunications,
National Centre for Scientific Research (N.C.S.R.) "Demokritos",
P.O BOX 60228, Aghia Paraskevi, GR-153 10, Athens, Greece.
{petasis, paliourg, costass}@iit.demokritos.gr
[2] Department of Informatics and Telecommunications,
University of Athens,
TYPA Buildings, Panepistimiopolis, GR-157 84, Athens, Greece.
halatsis@di.uoa.gr

Abstract. In this paper we present eg-GRIDS, an algorithm for inducing context-free grammars that is able to learn from positive sample sentences. The presented algorithm, similar to its GRIDS predecessors, uses simplicity as a criterion for directing inference, and a set of operators for exploring the search space. In addition to the basic beam search strategy of GRIDS, eg-GRIDS incorporates an evolutionary grammar selection process, aiming to explore a larger part of the search space. Evaluation results are presented on artificially generated data, comparing the performance of beam search and genetic search. These results show that genetic search performs better than beam search while being significantly more efficient computationally.
Keywords: grammatical inference, context-free grammars, minimum description length, genetic algorithms, positive examples

1 Introduction

In this paper we present eg-GRIDS, an algorithm for inducing context free grammars solely from positive sample sentences. This algorithm is an enhancement of the e-GRIDS algorithm [10], which was based on the GRIDS algorithm as it appeared in [6], which in turn is based on earlier work by Wolff [18], [19] and his SNPR system. The presented algorithm, similar to its predecessors, uses simplicity as a criterion for directing inference, a set of operators for exploring the search space and a basic beam search strategy. However, eg-GRIDS extends e-GRIDS by employing additional search operators and an evolutionary grammar selection process in addition to the basic beam search strategy, aiming to explore a larger part of the search space. Evaluation results on artificially generated data from context-free grammars suggest that eg-GRIDS performs better than e-GRIDS and that the evolutionary grammar selection process significantly

improves the processing efficiency and scalability of eg-GRIDS over its previous version.

The majority of grammatical inference algorithms presented in the literature share a common methodology. Based on an initial set of positive training examples, an overly specific grammar is constructed that is able to recognise only these examples. Then, a set of operators generalises this initial grammar, usually with respect to a set of *negative examples*, i.e. sentences that should not be recognised by the grammar. The existence of negative examples is a requirement of many algorithms, due to the need to limit the extent of generalisation. If in a given inference step a grammar is produced whose language is larger than the unknown target language, this is irreversible since no positive example could ever supply information to detect this error. Overly general grammars can be detected if negative examples are available, since the language of such a grammar is likely to include some negative examples. Thus, a learning algorithm that uses negative examples in this manner should primarily prevent *overspecialisation (or over-fitting)*, as overgeneralisation can be controlled by the negative examples. On the other hand, a learning algorithm that has to learn solely from positive examples must prevent both overspecialisation and *overgeneralisation*. However, as the absence of negative evidence often arises in practice, two solutions have been proposed in order to alleviate this problem:

- Restricted classes of formal languages have been proven to be learnable from positive examples, such as reversible languages [1], k-testable languages [3], code regular and code linear languages [2], pure context-free languages [5], [17] and strictly deterministic automata [20].
- Various heuristics aiming to avoid overgeneralisation without the use of negative examples have been proposed [12], [6].

The eg-GRIDS algorithm presented here belongs to the latter category and uses simplicity as a heuristic for directing the inference process, based solely on positive information. Viewing grammars as code, the heuristic utilised in eg-GRIDS, based on Minimum Description Length (*MDL*) [11], seeks to compress the grammar itself, as well as the encoding of the training sentences by the grammar.

Section 2 discusses work related to our approach, whereas section 3 presents the architecture, heuristics, search operators and search strategies employed by eg-GRIDS. Section 4 reports on an evaluation of eg-GRIDS on examples generated from an artificial[3] context-free grammar. Finally, section 5 concludes and outlines plans for future research.

2 Related work

eg-GRIDS shares some of its central features with earlier work in grammatical inference. We have already mentioned that eg-GRIDS originates from GRIDS

[3] The term "artificial grammar" is used to describe a grammar devised solely for evaluation purposes, which does not necessarily correspond to any real-world problem.

[6] which is in turn based on SNPR [18], [19]. Both GRIDS and SNPR are also biased towards "simple" grammars, as they use *MDL* for scoring and selecting the most plausible grammars.

Although the majority of the work in grammatical inference focuses on regular grammars, a significant number of algorithms exist that infer context-free grammars. Stolcke and Omohundro [16], [15] have presented an approach which infers probabilistic context-free grammars. Using a Bayesian framework, their system employs learning operators similar to those employed by eg-GRIDS in order to find a grammar with maximal posterior probability given the training example set, a criterion essentially equivalent to the *MDL*. In [13] an algorithm for inducing context-free grammars from positive and negative structured examples is presented. A structured example is simply an example with parentheses that indicate the shape of the derivation tree of the grammar (structural information). The learning algorithm employs a genetic search and the CYK algorithm [4] for converging to a final grammar. An efficient successor of this algorithm can be found in [7]. A more recent version of this algorithm [14] operates on partially structured examples instead of complete ones and uses a tabular representation, leading to a more flexible and applicable algorithm, compared to its predecessor. The algorithm has been successfully applied to various simple languages as well as to DNA sequence modelling. *Synapse* [8] is another algorithm that is based on CYK. Synapse learns incrementally from positive and negative examples following a top-down search organisation, while preliminary results show that it is able to infer both ambiguous and unambiguous context-free grammars for simple languages in reasonable time.

3 The eg-GRIDS algorithm

GRIDS [6] infers context-free grammars solely from positive example sets. It incorporates a beam search towards simple grammars with the help of two learning operators. Being based on GRIDS, eg-GRIDS shares some features with its predecessor:

- grammatical knowledge representation (context-free grammars);
- bias towards simple grammars, based on the same principle, i.e. minimum description length (*MDL*);
- two basic learning operators; and
- beam search.

However, eg-GRIDS also has some notable differences with GRIDS:

- it optimises the beam-based search process by using the results of a theoretical analysis of the dynamic behaviour of the learning operators [10];
- it incorporates new learning operators that can lead to more compact grammars; and
- it implements an additional genetic search strategy.

3.1 A bias towards "simple" grammars

As eg-GRIDS uses no negative evidence, an additional criterion is needed to direct the search through the space of context-free grammars and avoid overly general grammars. As we have mentioned above, this criterion provides a bias towards *simple* grammars. Following the GRIDS algorithm, we adopt the approach of *minimum description length (MDL)*, which directs the search process towards grammars that are compact, i.e., ones that require few bits to be encoded, while at the same time they encode the example set in a compact way, i.e. few bits are required to encode the examples using the grammar.

In order to use *MDL*, we must measure the encoding length of the grammar and of the example set, as encoded by the grammar. Assuming a context-free grammar G and a set of examples (sentences) T that can be recognised (parsed) by the grammar G, the total description length of a grammar (henceforth *model description length*, abbreviated as *ML*) is the sum of two independent lengths:

- The grammar description length (*GDL*), i.e. the bits required to encode the grammar rules and transmit them to a recipient who has minimal knowledge of the grammar representation, and
- the derivations description length (*DDL*), i.e. the bits required to encode and transmit all examples in the set T as encoded by grammar G, provided that the recipient already knows G.

The first component of the *ML* heuristic directs the search away from the sort of trivial grammar that has a separate rule for each training sentence, as this grammar will have a large *GDL*. However, the same component leads to the other sort of trivial grammar, a grammar that accepts all sentences. In order to avoid this, the second component estimates the *derivation power* of the grammar, by measuring the way the *training examples* are generated by the grammar, and helps to avoid overgeneralisation by penalising general grammars. The higher the derivation power of the language, the higher its *DDL* is expected to be. The initial overly specific grammar is trivially best in terms of *DDL*, as usually there is a one-to-one correspondence between the examples and the grammar rules, i.e. its derivation power is low. On the other hand, the most general grammar has the worst score, as it involves several rules in the derivation of a single sentence, requiring substantial effort to track down all the rules involved in the generation of the sentence.

Although *MDL* aims at a minimally compact representation of *both the model and the data* simultaneously, it does not provide means for creating the models, i.e. given a set of models that describe the same example set, *MDL* can only be used as an evaluation metric to decide which model is better. As a result, *MDL* cannot provide any help on how the space of possible models should be searched, in order to converge to a satisfactory model. In grammatical inference, *MDL* simply offers a mechanism for comparing grammars and selecting the one that is more "compact" with respect to both the length of the grammar as well as the encoding of the training set by the grammar.

3.2 Architecture of eg-GRIDS and the learning operators

The architecture of eg-GRIDS is summarised in figure 1. Like many other grammar inference algorithms, eg-GRIDS uses the training sentences in order to construct an *initial, "flat" grammar*. This initial grammar is constructed by simply converting each one of the training examples into a grammatical rule[4]. As a result, the number of initial rules corresponds to the number of training examples. This initial grammar is overly specific, as it can recognise only the sentences contained in the training set. After the initial grammar has been created, eg-GRIDS tries to generalise this initial grammar, with one of the two search processes: beam or the genetic search. Both search strategies utilise the same search operators in order to produce more general grammars.

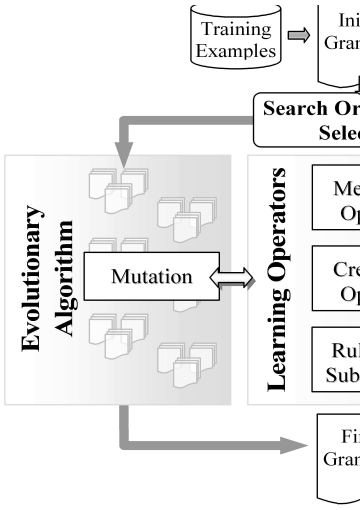

Fig. 1. The architecture of the eg-GRIDS algorithm.

Currently, eg-GRIDS supports five search operators:

Merge NT: merges two non-terminal symbols into a single symbol, thereby replacing all their appearances in the head and the body of rules.

Create NT: creates a new non-terminal symbol X, which is defined as a sequence of two or more existing non-terminal symbols. X is defined as a new production rule that decomposes X into its constituent symbols.

Create Optional NT: duplicates a rule created by the "Create NT" operator and appends an existing non-terminal symbol at the end of the body of the rule, thus making this symbol optional.

[4] The body of each rule is a sequence of non-terminal symbols, as each terminal is mapped initially to a unique non-terminal.

Detect Center Embedding: aims to capture the center embedding phenomenon. This operator tries to locate the most frequent four-gram[5] of the form "$AABB$". Once such a four-gram is located, the operator creates a new non-terminal symbol X as the operator "Create NT" would have done. However, *assuming* that this four-gram was created through center embedding involving symbol X, this operator additionally creates a new production rule of the form "$X \rightarrow AAXBB$" and replaces all symbol sequences that match the pattern "$A+\ X?\ B+$" with X.

Rule Body Substitution: examines whether the body of a production rule R is contained in bodies of other production rules. In such a case, every occurrence of the body of rule R in other rule bodies is replaced by the head of rule R.

The five operators create grammars that have either the same or greater expressiveness than their parent grammar. As the operators never remove rules from a grammar, the resulting grammars have at least the same coverage as the parent grammar, i.e. they can recognise at least the same set of sentences.

The choice of operators was motivated mostly by practical needs. The "Merge NT" and "Create NT" operators are fundamental: in principle these two operators can create any CFG structure, subject to the search involved for finding the right sequence of operators. The "Rule Body Substitution" operator helps to reduce redundancy in the grammar. Finally, the "Create Optional NT" and "Detect Center Embedding" operators operate as heuristics, and try to accelerate the search process focusing on important predetermined structures. Clearly, many more similar heuristics can be devised. However, it should be noted that all operators simply *suggest* new grammars. The final choice over what suggestions should be accepted is always left to the *MDL*.

3.3 Search strategies

Beam search The first search strategy implemented by eg-GRIDS is a beam search. Having an initial hypothesis (the initial grammar) in the search beam, eg-GRIDS uses five "modes" (one mode for each search operator) in order to explore the space of context-free grammars.

During learning, eg-GRIDS constantly alternates among the five modes, where each mode is characterised by the repetitive application of the same operator. Each mode initiates with a beam (beam A) containing the current grammars. For each grammar in beam A, all possible ways of applying the operator corresponding to the mode are examined. For example, the mode for the "Create NT" operator considers all ways of creating new symbols from all possible n-grams found in a grammar[6], while the mode for the "Merge NT" considers all

[5] Since bigrams and trigrams are quite common (frequent) structures and their presence can be attributed to a large number of phenomena, we assume that four-grams are the smallest n-grams that indicate possible existence of center embedding.

[6] All possible n-grams are examined, with n ranging from two (bigrams) to the length of the longest rule in the grammar.

ways of merging non-terminal symbols by repeatedly applying the "Merge NT" operator. For each operator application, a successor grammar is created and the best grammars according to the *MDL* form the new beam (beam B).

After all grammars in beam A have been examined, beams A and B are compared. If any of the grammars in beam B scores better than a grammar in beam A, beam A is replaced by beam B and the algorithm continues in this mode. However, if none of the grammars in beam B scores better than the grammars in beam A, then beam B is discarded and eg-GRIDS switches to another mode.

The algorithm continues alternating among the five modes until it is unable to produce a successor grammar that scores better than the ones in beam A. At that stage, the learning process terminates.

Genetic search The second search strategy implemented by eg-GRIDS is based on genetic algorithms. A genetic algorithm consists of four basic elements:

1. the evolutionary process, describing among other things which individuals should survive, reproduce or die;
2. the representation for individuals;
3. the set of genetic operators; and
4. the objective function for scoring individuals.

Regarding the evolutionary process, a variety of genetic algorithms is available in eg-GRIDS[7]. For the representation of individuals we have chosen to use the context-free grammars directly, rather than a special mapping that converts a grammar into a string. As a result, the choice of objective function was straightforwardly *MDL*. Regarding the set of genetic operators, eg-GRIDS can be considered to perform "informed mutation", using its five search operators.

More specifically, the genetic algorithm selects a grammar and an offspring grammar is produced from it as follows:

1. A search operator is randomly[8] chosen from the five operators.
2. The chosen operator is applied to the selected grammar, producing the offspring.
3. In case the chosen search operator requires additional information, e.g. the two non-terminal symbols that should be merged by the "Merge NT" operator, the required parameter values are randomly chosen from the set of all suitable values, e.g. all non-terminals for the "Merge NT" operator.

4 Experimental evaluation

In this section we evaluate the eg-GRIDS algorithm experimentally, focusing mainly on its performance on learning from examples generated from artificial context-free grammars.

[7] The eg-GRIDS software uses the GAlib genetic algorithm package, written by Matthew Wall at the Massachusetts Institute of Technology. Available from http://lancet.mit.edu/ga/.

[8] All random selections in our experiments were based on a uniform distribution.

4.1 Evaluation metrics for artificial grammars

Evaluation in grammatical inference presents peculiarities and common metrics that are used for supervised learning tasks, like recall and precision, are not directly applicable. Alternatively, in order to evaluate an inferred grammar we have to compare it against the "correct" grammar, so as to identify their similarity. However, even if the "correct" grammar is known, which is not the case in most real-world situations, the problem of determining whether two context-free grammars are equivalent is not an easy task: Given two context-free grammars G_1 and G_2, there exists no algorithm that can determine whether G_1 is more general than G_2 (i.e. $L(G_1) \supseteq L(G_2)$ or if $L(G_1) \cap L(G_2) = \emptyset$, where $L(G)$ the language of grammar G [9], [4].

As a result, during evaluation we mainly focus on measuring three aspects of the inferred grammar [6]:

- errors of omission (failures to parse sentences generated by the "correct" grammar), which indicate that an overly specific grammar has been learned,
- errors of commission (failures of the "correct" grammar to parse sentences generated by the inferred grammar), which indicate that an overly general grammar has been learned, and
- ability of the inferred grammar to parse correctly sentences longer than the sentences used during training, which indicates the additional expressiveness of the learned grammar.

In experiments with artificial context-free grammars, where the "correct" grammar is known, to estimate these figures we use the original (or "correct") grammar G_O and the learned grammar G_L to generate a large number of sentences. Errors of omission can be estimated as the fraction of the number of sentences generated by G_O that are not parsed by G_L to the total number of sentences generated by G_O. Errors of commission can be estimated as the fraction of the number of sentences generated by G_L that are not parsed by G_O to the total number of sentences generated by G_L. Errors of omission and errors of commission measure the *overlap* of the two grammars. In the ideal case, both of these figures must be zero, indicating that all sentences generated by one grammar can be parsed by the other. In order to estimate the third figure, example sentences must be generated from G_O that have greater length than the ones used for training. This figure can then be estimated as the fraction of the number of sentences that were successfully parsed by G_L to the total number of generated sentences.

4.2 The balanced parenthesis language

Our experiments examine the learning behaviour of eg-GRIDS on the Dyck language, not only because it is a context-free language but also because e-GRIDS comes very close into learning it [10], but fails to converge to the correct

grammar for a small number of the used training sets. Hence, we evaluate eg-GRIDS using the Dyck language with $k = 1$:

$$S \rightarrow S\ S|(\ S\ )| \in \tag{1}$$

In the experiments that we have conducted, we have generated a large number of unique sentences[9] top-down from the above grammar, using a uniform distribution for selecting randomly when expanding ambiguous non-terminals. The generated example sentences were randomly shuffled. Then, we defined an arbitrary maximum length $L_{max} = 20$ tokens[10] for the examples. The resulting set (subset A) was used for evaluation according to the first two figures, i.e. errors of omission and commission. Thus, a second test set was created (subset B), containing example sentences with lengths greater than L_{max} but lower than a second arbitrary maximum length (25 tokens). This second set is the set that was used in order to calculate the third figure, i.e. the ability to parse sentences longer than the ones used for training. All sets were populated by randomly selecting example sentences from the generated sentences. Special care was taken in order to ensure that the same sentence did not appear both in the training and the test sets.

As the Dyck language with $k = 1$ cannot be used to generate a large number of example sentences if we restrict the maximum sentence length, we have performed a ten-fold cross validation. Thus, subset A is split into ten subsets of equal size and the experiment is repeated 10 times: each time 9 subsets of A are used for training eg-GRIDS and the learned grammar is evaluated on the 10^{th} unused set, augmented with a test set created from subset B.

Figures 2 and 3 show the results of this experiment for the e-GRIDS algorithm (eg-GRIDS' predecessor), as presented in [10]. As it is evident from figure 2, e-GRIDS comes very close to learning the target grammar, as its performance approaches 0.95 with a training set size of 900 example sentences and remains around 0.90 for greater example set sizes. Regarding example sentences longer than the ones used for training (figure 3), e-GRIDS exhibits a similar behaviour. Finally, e-GRIDS never converges to a grammar that could produce ungrammatical sentences, i.e. it has zero errors of commission.

The same experiment was repeated for the eg-GRIDS algorithm, using the extended operator set and the genetic grammar selection strategy, with the "steady-state" genetic algorithm, an algorithm supporting overlapping populations. The first observation from the results was a significant improvement in terms of errors of omission. There were no such errors for sentences with length up to 20 words, even for very small training sets, while the situation is similar even for longer test sentences, as can be seen in figure 4.

An interesting difference between the two algorithms is the fact that eg-GRIDS seems to converge to more general grammars than the target grammar for small training set sizes, thus producing ungrammatical sentences (figure 5).

[9] All example sets used in our experiments contain unique example sentences, i.e. an example set of size 900 contains 900 distinct example sentences.

[10] A token is either a single left parenthesis "(" or a right one ")".

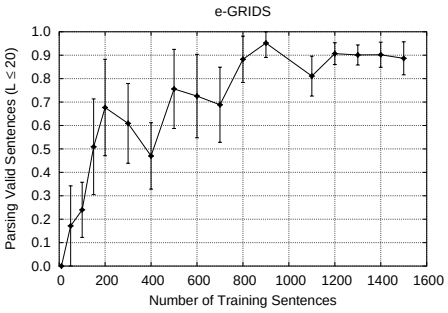

Fig. 2. Probability of parsing a valid sentence of length up to 20 words. (1-errors of omission)

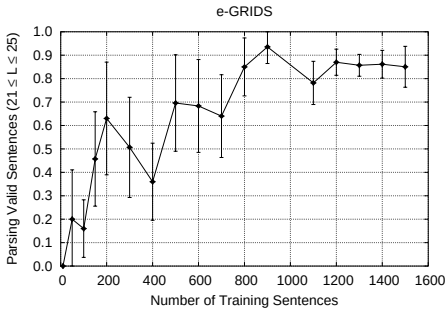

Fig. 3. Probability of parsing a valid sentence of length between 21 and 25 words. (1-errors of omission)

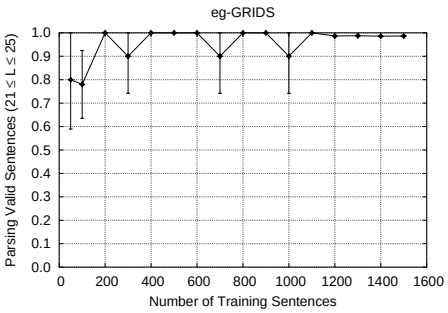

Fig. 4. Probability of parsing a valid sentence of length between 21 and 25 words. (1-errors of omission)

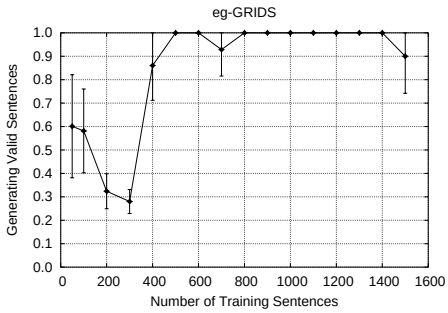

Fig. 5. Probability of generating a valid sentence. (1-errors of commission)

Genetic search allows eg-GRIDS to search a larger space of possible grammars, mainly due to the diversity of the grammars maintained in evolving populations. We believe that this diversity helped eg-GRIDS to converge to "better" grammars than e-GRIDS, even if these grammars are more general than the target one for small training set sizes.

5 Conclusions

In this paper, we presented the eg-GRIDS algorithm for inducing context-free grammars solely from positive evidence. eg-GRIDS utilises a heuristic based on minimum description length to avoid overgeneralisation, a consequence of the absence of negative evidence. Its main advantages are the fact that implements a genetic search strategy and has a richer set of search operators, offering eg-GRIDS the ability to adapt easier to a wider range of tasks and perform better than its predecessors.

Regarding the learning performance of eg-GRIDS, experiments have been conducted with the help of artificially generated examples. Our results have shown that eg-GRIDS is able to infer grammars that perform well, based on relatively small sets of training examples. The addition of the genetic search process, besides the increase in the performance of the algorithm, has also increased significantly the processing efficiency of the algorithm, and thus its scalability to more complex tasks. Preliminary results show that eg-GRIDS is more than an order of magnitude faster than e-GRIDS, due to the less exhaustive nature of its search. Future work will focus on the effect of the newly added search operators. Preliminary results suggest that eg-GRIDS can achieve very good performance on the Dyck language with k=2, a task beyond the capabilities of e-GRIDS.

Concluding, the work presented here has resulted in a new algorithm that alleviates some of the shortcomings of its predecessors, with special attention given to robustness and efficiency. We believe that eg-GRIDS will be useful in modelling various subparts of natural languages and identifying these subparts in texts, a task that cannot be easily modeled by other machine learning approaches, at least those that expect fixed-length vectors as input. Interesting tasks that fit this description include noun phrase chunking, named-entity recognition and information extraction.

References

1. D. Angluin, "Inference of reversible languages", *Journal of ACM*, vol. 29, pp. 741–765, 1982.
2. J. D. Emerald, K. G. Subramanian, D. G. Thomas, "Learning Code regular and Code linear languages", *Proceedings of International Colloquium on Grammatical Inference (ICGI-96)*, Lecture Notes in Artificial Intelligence 1147, Springer – Verlag, pp. 211–221, 1996.
3. P. García, E. Vidal, "Inference of K-testable languages in the strict sense and applications to syntactic pattern recognition", *Journal of IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12(9), pp. 920–925, 1990.

4. J. Hopcroft, J. Ullman, *Introduction to Automata Theory, Languages and Computation.* Addison – Wesley, 1979.
5. T. Koshiba, E. Makinen, Y. Takada, "Inferring pure context-free languages from positive data", *Technical report A-1997-14*, Department of Computer Science, University of Tampere, 1997.
6. P. Langley, S. Stromsten, "Learning Context-Free Grammars with a Simplicity Bias", *Proceedings of the Eleventh European Conference on Machine Learning (ECML 2000)*, Lecture Notes in Artificial Intelligence 1810, Springer – Verlag, pp. 220–228, Barcelona, Spain, 2000.
7. F. Mäkinen, "On the structural grammatical inference problem for some classes of context-free grammars", *Information Processing Letters*, 42, pp. 193–199, 1992.
8. K. Nakamura, T. Ishiwata, "Synthesizing Context Free Grammars from Simple Strings Based on Inductive CYK Algorithm", *Proceedings of the Fifth International Colloquium on Grammatical Inference (ICGI 2000), Grammatical Inference: Algorithms and Applications, Oliveira A. (ed.)*, Portugal, 2000. Springer.
9. R. Parekh, V. Honavar, "Grammar Inference, Automata Induction, and Language Acquisition", R. Dale, H. Moisl, and H. Somers (eds.), *Handbook of Natural Language Processing*, chapter 29, pp. 727–764, Marcel Dekker Inc., 2000.
10. G. Petasis, G. Paliouras, V. Karkaletsis, C. Halatsis and C.D. Spyropoulos, "e-GRIDS: Computationally Efficient Grammatical Inference from Positive Examples," Grammars, Special Issue, 2004. Available from http://217.125.102.104/special4.asp.
11. J. Rissanen, *Stochastic Complexity in Statistical Inquiry*, World Scientific Publishing Co, Singapore, 1989.
12. H. Rulot, E. Vidal, "Modelling (sub)string-length-based constraints through grammatical inference methods", Devijer and Kittler (eds.), Springer – Verlag, 1987.
13. Y. Sakakibara, "Efficient learning of context-free grammars from positive structural examples", *Information and Computation*, 97, pp. 23–60, 1992.
14. Y. Sakakibara, H. Muramatsu, "Learning Context-Free Grammars from Partially Structured Examples", *Proceedings of the Fifth International Colloquium on Grammatical Inference (ICGI 2000), Grammatical Inference: Algorithms and Applications, Oliveira A. (ed.)*, Portugal, 2000. Springer.
15. A. Stolcke, "Bayesian Learning of Probabilistic Language Models", *PhD Thesis, University of California at Berkley*, 1994.
16. A. Stolcke, S. Omohundro, "Inducing Probabilistic Grammars by Bayesian Model Merging", *Proceedings of International Colloquium on Grammatical Inference (ICGI-94)*, Lecture Notes in Artificial Intelligence 862, Springer – Verlag, pp. 106–118, 1994.
17. N. Tanida, T. Yokomori, "Inductive Inference of Monogenic Pure Context-free languages", Lecture Notes in Artificial Intelligence 872, Springer – Verlag, pp. 560–573, 1994.
18. G. Wolff, "Grammar Discovery as data compression", *Proceedings of the AISB/GI Conference on Artificial Intelligence*, pp. 375–379, Hamburg, West Germany, 1978.
19. G. Wolff, "Language Acquisition, Data Compression and Generalisation", *Language and Communication*, 2, pp. 57–89, 1982.

20. T. Yokomori, "On Polynomial-Time Learnability in the Limit of Strictly Deterministic Automata", *Journal of Machine Learning*, vol. 19, pp. 153–179, 1995.