University of Leeds
SCHOOL OF COMPUTING
RESEARCH REPORT SERIES
Report 2001.05

Comparing Two Unsupervised Grammar Induction Systems: Alignment-Based Learning vs. EMILE

by

Menno van Zaanen[1] **& Pieter Adriaans**[2]

March 2001

[1]School of Computing
menno@comp.leeds.ac.uk
[2]Department of Mathematics, Computer Science, Physics and Astronomy
University of Amsterdam
Plantage Muidergracht 24
1018 TV Amsterdam, The Netherlands
pieter.adriaans@ps.net

Abstract

In this paper we set out to compare two unsupervised grammar induction systems: Alignment-Based Learning (ABL) and EMILE. Both are motivated from a different background and with a different goal. ABL starts out from the linguistic notion of substitutability (Harris, 1951) aiming to learn a maximum number of correct constituents. On the other hand, EMILE stems from a mathematically sound theory of substitution classes which makes it possible to prove that EMILE's learned string language converges to the string language from which samples are taken. Apart from a theoretical comparison of the systems, the results on several corpora are compared.

1 Introduction

In this paper we study unsupervised grammar induction from text. Existing grammar learning methods can be grouped (like other learning methods) into supervised and unsupervised methods, and methods that only learn from positive data versus methods that use complete information (positive as well as negative). Unsupervised methods only use plain (or pre-tagged) sentences, while supervised methods are first initialised with structured sentences.

Since the landmark paper of Gold (Gold, 1967) we know that only a very limited class of languages can be identified from positive data alone. This fact, together with the observation that young children are rarely corrected by their parents when they start to learn their native language, has induced a widespread interest in theoretical as well as experimental research into constraints that make certain more interesting classes of languages (regular, context-free, context-sensitive) learnable.

Apart from this theoretical motivation for our research, there is also a more practical goal. In practice, supervised methods generate better results, since they can adapt their output to the structured examples from the initialisation phase, whereas unsupervised methods do not have any idea what the output should look like. Although unsupervised methods perform worse than supervised methods, unsupervised methods are necessary for the time-consuming and costly creation of treebanks for which no corpus nor grammar yet exists.

Grammar induction tools could also be used in situations where it is in principle impossible to use annotated corpora such as deciphering texts in an unknown language, or the analysis of non-linguistic data sets such as DNA sequences or sequences of error messages. From these facts it is clear that there is a strong practical motivation to develop real-life unsupervised grammar induction algorithms that work on text.

The main goal in this paper is the introduction and comparison of two approaches to unsupervised grammar induction: EMILE 4.1 (Adriaans, 1992) and ABL (Alignment-Based Learning) (van Zaanen, 2000a).

2 Previous Work

Both ABL and EMILE can be seen as learning approaches that have their roots in the idea of distributional analysis (Harris, 1951).

Early attempts try to induce context-free grammars from text but in 1967 Gold (Gold, 1967) proved that this is in general impossible. Gold's concept of identification in the limit has been amended with the notion of PAC learning (Probably approximately correct learning (Valiant, 1984)) in the eighties and PACS learning (PAC learning under simple distributions (Li and Vitányi, 1991)) in the nineties of the last century.

Various authors have worked on the unsupervised grammar induction problem. We will give a (very) short overview here.

There have been several approaches from the machine learning perspective, using for example genetic algorithms (Lankhorst, 1994) or neural networks (Honkela et al., 1995). Memory based learning (MBL) keeps track

of the possible contexts and assigns word types based on that information (Daelemans, 1995). Magerman and Marcus (1990) describe a method that finds constituent boundaries using mutual information values of the part of speech n-grams within a sentence and Redington et al. (1998) present a method that bootstraps syntactic categories using distributional information.

Algorithms that use the minimum description length (MDL) principle build grammars that describe the input sentences using the minimal number of bits. This idea stems from the information theory. Examples of these systems can be found in (Grünwald, 1994; de Marcken, 1996) and (Wolff, 1988).

The system by Wolff (1988) performs a heuristic search while creating and merging symbols directed by an evaluation function. Similarly, Cook et al. (1976) describe an algorithm that uses a cost function that can be used to direct search for a grammar. Stolcke and Omohundro (1994) describe a more recent grammar induction method that merges elements of models using a Bayesian framework. Chen (1995) presents a Bayesian grammar induction method, which is followed by a post-pass using the inside-outside algorithm (Baker, 1979; Lari and Young, 1990), while Pereira and Schabes (1992) apply the inside-outside algorithm to a partially structured corpus.

3 Description of the systems

In this section two grammar induction systems are described. We start out with an explanation of EMILE, followed by an overview of ABL. After describing the two systems, the differences and similarities are discussed.

3.1 The EMILE approach

The EMILE 4.1[3] algorithm is motivated by the concepts behind categorial grammar. It falls into the PACS paradigm. The theoretical concepts used in EMILE 4.1 are elaborated on in articles on EMILE 1.0/2.0 (Adriaans, 1992) and EMILE 3.0 (Adriaans, 1999). More information on the precursors of EMILE 4.1 may be found in the above articles, as well as in the E. Dörnenburg's Master's Thesis (1997). The EMILE 4.1 algorithm was designed and implemented by Vervoort (2000). Some other experiments on large corpora are reported in (Adriaans et al., 2000).

The general idea behind EMILE is the notion of identification of substitution classes by means of clustering. If a language has a context-free grammar then expressions that are generated from the same non-terminal can be substituted for each other in each context where that non-terminal is a valid constituent. Conversely, if we have a sufficiently rich sample from this language than one expects to find classes of expressions that cluster together in comparable contexts.

In figure 1 we illustrate how EMILE finds clusters and contexts in the ATIS corpus. Actually, this type of clustering can be seen as a form of text compression (Grünwald, 1994). The context *What is* occurs with the expressions *a family fare* and *coach fare for flight 1943*.

What is a family fare
What is coach fare for flight 1943

What is {*a family fare* |
coach fare for flight 1943}

Figure 1: Example clustering expressions in EMILE

This finding gives rise to the hypothesis (possibly unjustified) that these two expressions are generated from the same non-terminal. If we find enough traces of a whole group of expressions in a whole group of contexts the probability of this hypothesis grows.

For a sentence of length n the maximal number of different contexts and expressions is $1/2n(n+1)$.[4] The complexity of a routine

[3] EMILE 4.1 is a successor to EMILE 3.0, conceived by Adriaans. The original acronym stands for Entity Modelling Intelligent Learning Engine. It refers to earlier versions of EMILE that also had semantic capacities. The name EMILE is also motivated by the book on education by J.-J. Rousseau.

[4] Note that EMILE's cluster routine does a more extensive search for patterns than a k-gram routine that distinguishes only $n-(k-1)$ elements in a sen-

that clusters all contexts and expressions is polynomial in the number of contexts and expressions.

The EMILE family of algorithms works efficiently for the class of shallow context-free languages with characteristic contexts and expressions provided that the sample is taken according to a simple distribution (Adriaans, 1999).

An expression of a type T is *characteristic* for T if it only appears with contexts of type T. Similarly, a context of a type T is *characteristic* for T if it only appears with expressions of type T.

In the example one might see the context *What is* as characteristic for nominal phrases. One might see the phrase *a family fare* as characteristic for nominal contexts.

A distribution is simple if it is recursively enumerable. A class of languages C is *shallow* if for each language L it is possible to find a grammar G, and a set of sentences S inducing characteristic contexts and expressions for all the types of G, such that the size of S and the length of the sentences of S are logarithmic in the descriptive length of L (relative to C).

Languages with characteristic contexts and expressions for each syntactic type are called *context- and expression-separable*. A sample is *characteristic* if it allows us to identify the right clusters that correspond with nonterminals in the original grammar.

Samples generated by arbitrary probability distributions are very likely to be non-characteristic. One can prove, however, that if the sample is drawn according to a simple distribution and the original grammar is shallow then the right clusters will be found in a sample of polynomial size, i.e. one will have a characteristic sample of polynomial size.

Natural languages seem to be context- and expression-separable for the most part, i.e. if there are any types lacking characteristic contexts or expressions[5], these types are few in number, and rarely used. Furthermore, there is no known example of a syntactical construction in a natural language that cannot be expressed in a short sentence. Hence the conjecture that natural languages are (mostly) context- and expression-separable, and shallow seems tenable. This explains why EMILE works for natural language.

3.1.1 The EMILE 4.1 algorithm

The EMILE 4.1 algorithm consists of two main stages: *clustering* and *rule induction*.

In the clustering phase all possible contexts and expressions of a sample are gathered in a matrix. Starting from random seeds, clusters of contexts and expressions, that form correct sentences, are created.[6] If a group of contexts and expressions cluster together they receive a type label. This creates a set of proto-rules. In our example: *[0]⇒ What is [19]* and *[19] ⇒ a family fare*. The sentence type [0] can be rewritten as *What is* concatenated to an expression of type [19].

In the rule induction phase a concise method for rule creation is used.[7] The matrix is checked for characteristic expressions and new rules are derived by substitution of types for characteristic sub-expressions in typed expressions (Vervoort, 2000). Suppose for instance that the expression *a family fare* is characteristic for type [19]. We may then form the rule *[201] ⇒ the price of [19] to Rome* from the rule *[201] ⇒ the price of a family fare to Rome*.

In (Adriaans, 1999) it is shown that the EMILE 3.0 algorithm can PACS learn shallow context-free (or categorial) languages with context- and expression separability in time polynomial to the size of the grammar. EMILE 4.1 is an efficient implementation of the main characteristics of 3.0.

3.2 Alignment-Based Learning

ABL is based on Harris's idea of substitutability, which states that if two constituents are of the same type then they can be substituted by each other (Harris, 1951). To illustrate this,

tence

[5] After rewriting types such as 'verbs that are also nouns' as composites of basic types.

[6] A set of parameters and thresholds determines the significance of, and the amount of noise in the clusters.

[7] EMILE 3.0 uses a much more elaborate sound rule induction algorithm, but it is impossible to implement this routine efficiently.

consider the sentence *What is a family fare*. If the noun phrase *a family fare* is replaced by another noun phrase, for example *the payload of an African Swallow*, the resulting sentence is the (syntactically) correct sentence *What is the payload of an African Swallow*.

ABL now tries to find constituents by using a reversed version of Harris's implication: if parts of sentences can be substituted by each other then they are constituents of the same type.[8]

As an example of how ABL finds constituents, consider the two sentences from the ATIS corpus in figure 2. From these two sentences we may assume that *a family fare* can be replaced by *the payload of an African Swallow*, since both sentences occur in the corpus and thus are both valid. The dissimilar parts of the sentences can be replaced by each other, so ABL now assumes that these parts are constituents of the same type.

What is a family fare
What is the payload of an African Swallow

What is (a family fare)$_X$
What is (the payload of an African Swallow)$_X$

Figure 2: Example bootstrapping structure in ABL

The implementation of the ABL system described here consists of two distinct steps *Alignment Learning* and *Selection Learning*, which will both be described next. For a more elaborate review see (van Zaanen, 2000a) or (van Zaanen, 2000b).

3.2.1 Alignment Learning

During the alignment learning phase, the system effectively builds a search space of possible constituents.

The string-edit distance algorithm (Wagner and Fischer, 1974) is used to find the parts of the sentences that are similar in both sentences and the parts of the sentences that are dissimilar. The dissimilar parts can be substituted in pairs and are thus stored as possible constituents.

Possible constituents are stored as pairs of brackets in the sentence and its non-terminal type. Two substitutable constituents get the same non-terminal (since they are considered to be of the same type). Each time a new constituent is found, a new non-terminal is introduced.

Instead of comparing only two sentences (as in the example), ABL compares every sentence in the corpus to all other sentences in the corpus (in pairs). Already learned structure is stored together with the sentence, but this information is not used when aligning to other sentences.

At some point it may occur that a constituent is learned that was already known (apart from the type) because of alignment against an earlier sentence. In this case, all occurrences of the two types (the old and the new one) are merged (i.e. they are equated). This reduces the number of constituents in the learned treebank.

Three different alignment methods have been implemented, but since the difference in results of the three systems is relatively small, we have chosen to use the default alignment system as described in (van Zaanen, 1999), which is exactly the system just described.[9]

3.2.2 Selection Learning

The alignment learning phase in the algorithm may at some point generate constituents that overlap with other constituents. The second sentence in figure 3 receives two overlapping structures. One constituent is learned by comparing against the first sentence and the other (overlapping) constituent is found by aligning with the third. Note that all possible constituents (including the overlapping ones) are stored during alignment learning.

This problem can be solved by selecting the correct constituents (or at least the better constituents) out of the possible constituents. Selecting constituents can be done in several different ways.

[8] Although the original implication may always hold, the reversed implication breaks down on certain cases as described in (van Zaanen, 2000a).

[9] The other systems use a slightly different γ cost function to find the alignment or learn using all possible alignments (when more alignments are possible).

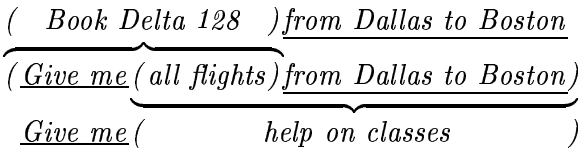

Figure 3: Overlapping constituents

van Zaanen (2000b) describes five different selection methods. One method assumes that earlier learned constituents are correct. The other four methods are probabilistic in nature.

The probabilistic methods select constituents based on their probability. First, the probability of the (overlapping) constituents is computed. Then the probability of the combination of constituents is computed using the geometric mean (instead of using the product as is done in SCFGs (Booth, 1969).)[10]

The probabilistic methods differ in the way the constituent probabilities are computed. The *leaf* method computes the probability of a constituent counting the number of times the words in the constituent occur as a constituent in the learned text, normalised by the total number of constituents.

$$P_{leaf}(c) = \frac{|c' \in C : yield(c') = yield(c)|}{|C|}$$

where C is the entire set of constituents.

The *branch* method computes the probability of a constituent using the occurrences of the words in the constituent *and* its nonterminal (i.e. it is a normalised version of P_{leaf}).

$$P_{branch}(c|root(c) = r) = \frac{|c' \in C : yield(c') = yield(c) \wedge root(c') = r|}{|c'' \in C : root(c'') = r|}$$

The two other methods are similar to the *leaf* and *branch* methods. Only when two or more possible combinations of constituents have the same probability, these methods select the combination with the largest amount of constituents.

[10] Using the geometric mean reduces the *trashing* effect (Caraballo and Charniak, 1998).

In this evaluation, we chose the intuitively best system: plain *branch* (not the extended the system that selects the largest amount of constituents in case of multiple possibilities).

3.3 Theoretical comparison between ABL and EMILE

While ABL directly (and greedily) structures sentences, EMILE tries to find grammar rules[11] and it only finds a grammar rule when enough evidence is found. This duality is actually the main difference of the two systems. This results in ABL being slower (and thus working on smaller corpora) in contrast to EMILE, which is developed to work on much larger corpora (say over 100,000 sentences).

The inner working of the algorithms is completely different. EMILE finds a grammar rule when enough information is found to support the rule. In contrast, ABL stores all possible constituents and then after all possible constituents are found, the "best" constituents are selected.

Given the two sentences *What is a family fare* and *What is coach fare for flight 1943* ABL will try to align *What is* and *fare*, whereas EMILE will try to cluster *coach fare for flight 1943* and *a family fare*.

One other interesting feature is that both systems can learn recursive structures. To our knowledge, no other systems do this.

4 Test Environment

The two systems have been tested on two treebanks: the *Air Traffic Information System (ATIS)* treebank (Marcus et al., 1993) and the *Openbaar Vervoer Informatie Systeem*[12] *(OVIS)* treebank (Bonnema et al., 1997).

The Penn treebank ATIS is an English treebank consisting of 716 sentences. The larger OVIS treebank contains exactly 10,000 Dutch sentences. Removing all sentences consisting of only one word results in a corpus of 6,797 sentences. The sentences of both corpora contain mainly imperatives and questions.

[11] Using these grammar rules the original corpus is parsed and the resulting structured sentences are used for evaluation.

[12] "Openbaar Vervoer Informatie Systeem" stands for "Public Transport Information System".

To evaluate the systems, the sentences from the treebanks are stripped from their structure and the plain sentences are fed to the grammar induction systems. The generated treebanks are then compared to the original treebanks.

To compare the treebanks, we use the PARSEVAL measures as described in (Black et al., 1991). The commonly used EVALB program is used to compute the PARSEVAL measures and the EVALB measures. Additionally, the *F-score* is computed.

Note that ABL generates empty constituents, but EMILE does not. To simplify evaluation, we have removed the empty constituents from all treebanks.

To our knowledge, there is no "standard" evaluation test bench for unsupervised grammar induction systems. Instead, we have tried to use as many "standard" components used by others to evaluate their system. This justifies the use of the ATIS treebank, the PARSEVAL measures and the EVALB program.

Instead of testing the unsupervised methods on English text only, we have chosen to use the Dutch OVIS treebank as well. This allows us to see how well the systems perform on a less(er) known language. In our view, unsupervised methods are best used in fields where no structured corpora yet exist (since supervised systems cannot be used there). In addition, it is roughly ten times as large as the ATIS treebank.

Using these components unfortunately does not give results that can be compared to other existing results directly. Hopefully, people evaluating their unsupervised grammar will take an approach similar to ours making further comparison possible.

5 Results

An overview of the results of both systems on the two treebanks can be found in table 1. The figures in the tables represent the mean values of the metric followed by their standard deviations (in brackets). Each result is computed using ten fold cross validation.

Note that the OVIS and ATIS corpora are certainly not characteristic for the underlying grammars. It is therefore impossible to learn a perfect grammar for these corpora from the data in the corpora.[13]

As can be seen from the results, EMILE is a slow learner on small corpora (hence the low recall), but the learned constituents are reasonably good (higher precision than recall and also a high 0 CB and $\leq$2 CB measures and a low average crossing).[14] On the other hand, ABL learns faster, resulting in a higher recall and a slightly lower precision on the ATIS corpus. On the OVIS, ABL has a high recall and precision, but the CB, 0 CB and $\leq$2 CB measures are still lower than EMILE's. This indicates that ABL learns greedily, but ABL also tends to learn some incorrect constituents.

Remember that although the results may not look very good, these measures were actually designed to evaluate *supervised* methods. These results cannot be compared to the results of supervised systems under any circumstances. Unsupervised learning of structure is a much harder problem to solve, since the systems do not have a clue whatsoever what the resulting structure should look like.

6 Conclusion

First of all, the results of the system may seem low compared to the results of supervised systems, but this does not mean that unsupervised methods are not useful. Supervised methods need structured corpora to train or initialise, but these resources are often not readily available (for example when analysing unknown languages). Since unsupervised methods like EMILE or ABL do not need these resources, they can still be used.

In general ABL greedily searches for constituents, while EMILE is much more cautious. EMILE needs to have a certain support in the context-expression matrix for the construction of a new type, while ABL only needs

[13] It is our hypothesis that one needs a corpus of at least 50.000.000 sentences to get an acceptable grammar of the English language on the basis of the EMILE algorithm.

[14] EMILE first has to learn a grammar and then parses the sentences, while ABL produces structured sentences directly. The current parser of EMILE is non-probabilistic.

Table 1: Results: UR=unlabelled recall, UP=unlabelled precision, F=F-score, CB=average crossing brackets, 0 CB=no crossing brackets, ≤2 CB=two of fewer crossing brackets

		UR	UP	F	CB	0 CB	≤2 CB
ATIS	EMILE	16.814 (0.687)	**51.588** (2.705)	25.351 (1.002)	**0.841** (0.108)	**47.362** (4.765)	**93.408** (2.195)
	ABL	**35.564** (0.020)	43.640 (0.023)	**39.189** (0.021)	2.120 (0.000)	29.050 (0.000)	64.996 (0.069)
OVIS	EMILE	36.893 (0.769)	49.932 (1.961)	41.433 (3.213)	**0.714** (0.053)	**56.897** (2.475)	**93.189** (0.695)
	ABL	**61.536** (0.007)	**61.956** (0.008)	**61.745** (0.007)	1.220 (0.000)	42.046 (0.017)	85.210 (0.000)

to identify expressions in a pair of sentences to learn structure.

In texts with length of the OVIS and ATIS corpora there is not enough information for the EMILE approach to converge to a grammar. In general EMILE is much faster and less greedy than ABL. For large corpora (say >100K sentences) ABL is currently not feasible, while this is well within the possibilities of EMILE.[15]

Since the underlying ideas of EMILE and ABL match rather well it should be possible to develop a hybrid version using the best qualities of both algorithms. This hybrid grammar induction tool would combine the greedy power of ABL locally with the scalability of EMILE on large data sets.

Acknowledgements

We would like to thank Marco Vervoort for his work in generating the EMILE results on both the ATIS and OVIS corpus.

References

P. Adriaans, M. Trautwein, and Vervoort M. 2000. Towars high speed grammar induction on large text corpora. In G.K. Hlaváč, V. Feffrey and J. Wiedermann, editors, *SOFSEM 2000: Theory and Practice of Informatics*, volume 1963 of *Lecture Notes in Computer Science*, pages 173–186. Berlin: Springer Verlag.

Pieter Willem Adriaans. 1992. *Language Learning from a Categorial Perspective.* Ph.D. thesis, Universiteit van Amsterdam, nov.

Pieter Adriaans. 1999. Learning shallow context-free languages under simple distributions. ILLC Report PP-1999-13, Institute for Logic, Language and Computation, Amsterdam.

J. K. Baker. 1979. Trainable grammars for speech recognition. In J. J. Wolf and D. H. Klatt, editors, *Speech Communication Papers for the Ninety-seventh Meeting of the Acoustical Society of America*, pages 547–550.

E. Black, S. Abney, D. Flickenger, C. Gdaniec, R. Grishman, P. Harrison, D. Hindle, R. Ingria, F. Jelinek, J. Klavans, M. Liberman, M. Marcus, S. Roukos, B. Santorini, and T. Strzalkowski. 1991. A procedure for quantitatively comparing the syntactic coverage of english grammars. In *Speech and Natural Language — Proceedings of a Workshop*, pages 306–311, feb.

R. Bonnema, R. Bod, and R. Scha. 1997. A DOP model for semantic interpretation. In *Proceedings of the Association for Computational Linguistics/European Chapter of the Association for Computational Linguistics, Madrid*, pages 159–167. Sommerset, NJ: Association for Computational Linguistics.

T. Booth. 1969. Probabilistic representation of formal languages. In *Conference Record of 1969 Tenth Annual Symposium on Switching and Automata Theory*, pages 74–81.

Sharon A. Caraballo and Eugene Charniak. 1998. New figures of merit for best-first probabilistic chart parsing. *Computational Linguistics*, 24(2):275–298.

Stanley F. Chen. 1995. Bayesian grammar induction for language modeling. In *Proceedings of the Association for Computational Linguistics*, pages 228–235.

C. M. Cook, A. Rosenfeld, and A. R. Aronson. 1976. Grammatical inference by hill climbing. *Informational Sciences*, 10:59–80.

Walter Daelemans. 1995. Memory-based lexical acquisition and processing. In P. Steffens, editor, *Machine Translation and the Lexicon*, volume 898 of *Lecture Notes in Artificial Intelligence*, pages 85–98. Berlin: Springer Verlag.

[15] However, to date no sufficiently large treebanks exists.

Carl G. de Marcken. 1996. *Unsupervised Language Acquisition*. Ph.D. thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA, sep.

E. Dörnenburg. 1997. Extensions of the emile algorithm for inductive learning of context-free grammars. Master's thesis, University of Dortmund.

E.M. Gold. 1967. Language identification in the limit. *Information and Control*, 10:447–474.

Peter Grünwald. 1994. A minimum description length approach to grammar inference. In G. Scheler, S. Wernter, and E. Riloff, editors, *Connectionist, Statistical and Symbolic Approaches to Learning for Natural Language*, volume 1004 of *Lecture Notes in AI*, pages 203–216. Berlin: Springer Verlag.

Zellig Harris. 1951. *Methods in Structural Linguistics*. Chicago, IL: University of Chicago Press.

T. Honkela, V. Pulkki, and T. Kohonen. 1995. Contextual relations of words in grimm tales, analyzed by self-organizing map. In *Proceedings of the International Conference on Artificial Neural Networks*.

M.M. Lankhorst. 1994. Grammatical inference with a genetic algorithm. In *Proceedings of the 1994 EUROSIM Conference on Massively Parallel Applications and Development*, pages 423–430.

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

M. Li and P. Vitányi. 1991. Learning simple concepts under simple distributions. *SIAM Journal of Computing*, pages 911–935.

D. Magerman and M. Marcus. 1990. Parsing natural language using mutual information statistics. In *Proceedings of the National Conference on Artificial Intelligence*, pages 984–989. Cambridge, MA: MIT Press.

M. Marcus, B. Santorini, and M. Marcinkiewicz. 1993. Building a large annotated corpus of english: the Penn treebank. *Computational Linguistics*, 19(2):313–330.

F. Pereira and Y. Schabes. 1992. Inside-outside reestimation from partially bracketed corpora. In *Proceedings of the Association for Computational Linguistics*, pages 128–135, Newark, Delaware.

Martin Redington, Nick Chater, and Steven Finch. 1998. Distributional information: A powerful cue for acquiring syntactic categories. *Cognitive Science*, 22(4):425–469.

Andreas Stolcke and Stephen Omohundro. 1994. Inducing probabilistic grammars by bayesain model merging. In *Second International Conference on Grammar Inference and Applications*, pages 106–118. Berlin: Springer Verlag. Alicante, Spain.

L.G. Valiant. 1984. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142.

M. van Zaanen. 1999. Bootstrapping structure using similarity. In Paola Monachesi, editor, *Computational Linguistics in the Netherlands*, pages 235–245.

M. van Zaanen. 2000a. ABL: Alignment-based learning. In *Proceedings of the 18th International Conference on Computational Linguistics COLING*, pages 961–967, aug.

M. van Zaanen. 2000b. Bootstrapping syntax and recursion using alignment-based learning. In *Proceedings of the Seventeenth International Conference on Machine Learning ICML*, pages 1063–1070, jul.

Marco R. Vervoort. 2000. *Games, Ealks and Grammars*. Ph.D. thesis, Universiteit van Amsterdam, sept.

Robert A. Wagner and Michael J. Fischer. 1974. The string-to-string correction problem. *Journal of the Association for Computing Machinery*, 21(1):168–173, jan.

J. G. Wolff. 1988. Learning syntax and meanings through optimization and distributional analysis. In I.M. Schlesinger Levy, Y. and M.D.S. Braine, editors, *Categories and Processes in Language Acquisition*, pages 85–98. Hillsdale NJ: Lawrence Erlbaum.