Towards High Speed Grammar Induction on Large Text Corpora

Pieter Adriaans[1 2], Marten Trautwein[1], and Marco Vervoort[2]

[1] Perot Systems Nederland BV, P.O.Box 2729, NL-3800 GG Amersfoort, The Netherlands
{Pieter.Adriaans,Marten.Trautwein}@ps.net

[2] University of Amsterdam, FdNWI, Plantage Muidergracht 24, NL-1018 TV Amsterdam, The Netherlands
vervoort@wins.uva.nl

Abstract. In this paper we descri e an efficient and scala le implementation for grammar induction ased on the EMILE approach ([2], [3],[4], [5], [6]). The current EMILE 4.1 implementation ([11]) is one of the first efficient grammar induction algorithms that work on free text. Although EMILE 4.1 is far from perfect, it ena les researchers to do empirical grammar induction research on various types of corpora.
The EMILE approach is ased on notions from categorial grammar (cf. [10]), which is known to generate the class of context-free languages. EMILE learns from positive examples only (cf. [1], [7], [9]). We descri e the algorithms underlying the approach and some interesting practical results on small and large text collections. As shown in the articles mentioned a ove, in the limit EMILE learns the correct grammatical structure of a language from sentences of that language. The conducted experiments show that, put into practice, EMILE 4.1 is efficient and scala le. This current implementation learns a su class of the shallow context-free languages. This su class seems sufficiently rich to e of practical interest. Especially Emile seems to e a valua le tool in the context of syntactic and semantic analysis of large text corpora.

1 Introduction

The current EMILE 4.1 implementation ([11]) is one of the first efficient and scalable grammar induction algorithms that work on free text. EMILE 4.1 is not perfect, but enables researchers to conduct empirical grammar induction research on different types of corpora. EMILE 4.1 attempts to learn the grammatical structure of such a language from sentences of that language, without any prior knowledge of the grammar apart from the fact that the grammar is categorial (i.e., equivalent to context-free) and shallow ([2], [3], [4]). Theoretically, EMILE learns the correct grammatical structure of a language from sentences of that language in the limit. This and other theoretical concepts used in EMILE 4.1 are elaborated on in Pieter Adriaans' articles on EMILE 1.0/2.0 ([2]) and EMILE 3.0 ([3], [5], [6]).

In a *shallow* language every syntactical construction has an example sentence of a length that is logarithmic in the complexity of the grammar as a whole. We believe that natural languages are shallow in this sense. Categorial grammars are based on the assignment of syntactic types to words of the lexicon. A defining characteristic of categorial languages is that expressions of the same type can be substituted for each other in all contexts. This feature forms the basis for the inductive approach of EMILE. For any type in any valid grammar for the language, we can expect context/expression combinations to show up in a sufficiently large sample of sentences of the language. EMILE searches for such clusters of expressions and contexts in the sample, and interprets them as grammatical types. It then tries to find characteristic contexts and expressions, and uses them to extend the types. Finally, it formulates derivation rules based on the types found, in the manner of the rules of a context-free grammar.

In this paper we will focus on the practical aspects of EMILE 4.1 (cf. [11]). With EMILE 4.1 grammar induction experiments can be applied to different types of corpora. Especially Emile seems to be a valuable tool for syntactic and semantic analysis of large text corpora. The experiments show that put into practice EMILE 4.1 is efficient and scalable. The current EMILE 4.1 implementation learns a subclass of the shallow context-free languages that is sufficiently rich to be of practical interest in the context of syntactic and semantic analysis of large corpora.

In comparison to its precursors EMILE 4.1 is better able to handle incomplete samples and uses positive examples only. More information on the precursors of EMILE 4.1 may be found in the above mentioned articles, as well as in the E. Dörnenburg's Master's Thesis ([8]).

2 The Algorithms of EMILE

This section gives insight into the reasoning underlying the algorithms. Details on the algorithms (like pseudo-code) can be found in [11]. Below, we track the development stages of EMILE, and explain the purpose of each change.

2.1 1-Dimensional Clustering

Semantic types have the property that wherever some expression is used as an expression of a particular type, other expressions of that particular type can be substituted without making the sentence ungrammatical. EMILE uses this property to identify grammatical types. As such, a grammatical type in EMILE is characterized by the expressions that belong to that type, and the contexts in which expressions of that type can appear. The context/expression pair is the principle concept in EMILE. Basically, a context/expression pair is a sentence split into three pairs, for instance,

John (makes) tea .

We say that the *expression* 'makes' appears in the *context* 'John (.) tea', or in terms of formal categorial grammar rules ([10]):

$$\text{makes} \Rightarrow \text{John}\backslash\sigma/\text{tea} \ .$$

A simple clustering technique extracts all possible context/expression combinations from a given sample of sentences, and groups together expressions that appear in the same context. For instance, if we take the sample sentences 'John makes tea' and 'John likes tea', we get the context/expression *matrix* in Tab. 1 from which we can obtain the clusters given in Tab. 2.

Table 1. Context/expression matrix for 1-dimensional clustering

	(.) makes tea	John (.) tea	John makes (.)	(.) tea	John (.)	(.)	(.) likes tea	John likes (.)
John	x						x	
makes		x						
tea			x					x
John makes				x				
makes tea					x			
John makes tea						x		
likes		x						
John likes				x				
likes tea					x			
John likes tea						x		

Table 2. Clusters derived from context/expression matrix

[{'makes', 'likes'}, 'John (.) tea']
[{'John makes', 'John likes'}, '(.) tea']
[{'makes tea', 'likes tea'}, 'John (.)']
[{'John makes tea', 'John likes tea'}, '(.)']

Next, we can group contexts together if they appear with exactly the same expressions. For instance, if we add the sentences 'John makes coffee', 'John likes coffee' to the sample, the relevant part of the context/expression matrix will look like the matrix given in Tab. 3, which will yield the clusters given in Tab. 4.

As stated before, a grammatical type can be characterized by the expressions that are of that type, and the contexts in which expressions of that type appear.

Table 3. Extended context/expression matrix for 1-dimensional clustering

	John (.) tea	John (.) coffee	John makes (.)	John likes (.)
makes	x	x		
likes	x	x		
tea			x	x
coffee			x	x

Table 4. Clusters derived from extended context/expression matrix

[{'makes', 'likes'}, {'John (.) tea', 'John (.) coffee'}]
[{'tea', 'coffee'}, {'John makes (.)', 'John likes (.)'}]

Hence the clusters we find here can be interpreted as grammatical types. For instance, the clusters in Tab. 4 correspond to the grammatical types of 'verbs' and 'nouns', respectively.

2.2 2-Dimensional Clustering

The 1-dimensional clustering technique fails to properly handle contexts whose type is ambiguous. For instance, if we add the sentences 'John likes eating' and 'John is eating', the relevant part of the context/expression matrix is given in Tab. 5.

Table 5. Context/expression matrix for 2-dimensional clustering

	John (.) tea	John (.) coffee	John (.) eating	John makes (.)	John likes (.)	John is (.)
makes	x	x				
likes	x	x	x			
is			x			
tea				x	x	
coffee				x	x	
eating					x	x

Table 5 shows four distinct grammatical types: noun-phrases ('tea', 'coffee'), verb-phrases ('makes','likes'), 'ing'-phrases ('eating'), and auxiliary verbs ('is') that appear with 'ing'-phrases. The context 'John likes (.)' is ambiguous, since

both noun-phrases and 'ing'-phrases can appear in this context. The ambiguity is naturally represented as the context belong to two different types, i.e., we would like to obtain the clustering given in Tab. 6. However, if we proceed as before with the 1-dimensional clustering technique, we get the clusters given in Tab. 7.

In Tab. 7 the ambiguous context 'John likes (.)' is assigned a separate type, which results in a less natural representation. Moreover the separate type prevents us from correctly identifying the so-called *characteristic* expressions, i.e., the expressions that belong to exactly one type. The expected and more natural representation given in Tab. 6 would allow ambiguous contexts and expressions to belong to multiple types.

Table 6. Expected clusters in context/expression matrix

[{'makes', 'likes'}, {'John (.) tea', 'John (.) coffee'}]
[{'likes', 'is'}, {'John (.) eating'}]
[{'tea', 'coffee'}, {'John makes (.)', 'John likes (.)'}]
[{'eating'}, {'John is (.)', 'John likes (.)'}]

Table 7. Derived clusters in context/expression matrix

[{'makes', 'likes'}, {'John (.) tea', 'John (.) coffee'}]
[{'likes', 'is'}, {'John (.) eating'}]
[{'tea', 'coffee'}, {'John makes (.)'}]
[{'tea', 'coffee', 'eating'}, {'John likes (.)'}]
[{'eating'}, {'John is (.)'}]

In order to find the desired result, we need a different type of clustering. EMILE 4.1 does a type of *2-dimensional* clustering, namely, it searches for maximum-sized blocks in the matrix. Table 8 shows the matrix of the example, with the maximum-sized blocks indicated by rectangles for visualization purposes. The matrix ks

Table 8. Context/expression matrix for 2-dimensional clustering

	John (.) tea	John (.) coffee	John (.) eating	John makes (.)	John likes (.)	John is (.)
makes	x	x				
likes	x	x	x			
is			x			
eating					x	x
tea				x	x	
coffee				x	x	

adding contexts and expressions until the block can no longer be enlarged. This is done for each context/expression pair that is not already contained in some block. Some of the resulting blocks may be completely covered by other blocks (such as the two 1×3 blocks in Tab. 8): once all context/expression pairs have been covered, these superfluous blocks are eliminated.

The total 2-dimensional clustering is efficient (i.e., takes polynomial time in the size of the grammar) as is proven in [11].

2.3 Allowing for Imperfect Data: Using Characteristic Expressions and Contexts

In the previous section, a block entirely had to be contained within the matrix. That is, the clustering algorithm did not find a type unless every possible combination of contexts and expressions of that type had actually been encountered and stored in the matrix. This approach only works if a perfect sample is provided. (With a perfect sample we mean a sample that consists of all and only all sentences of the language that the to be learned grammar generates.) In practical use, we need to allow for imperfect samples. There are many context/expression combinations, which are grammatical but nevertheless will appear infrequently or never: for instance, 'John likes evaporating'.

To allow EMILE to be used with imperfect samples, two modifications have been made to the algorithm. First, the requirement that the block is completely contained in the matrix, is weakened to a requirement that the block is *mostly* contained in the matrix, i.e., exceeds some user-defined tresholds. Specifically, the percentage of context/expression pairs of the block that are contained in the matrix should exceed a treshold, and for each individual row or column of the block, the percentage for that row or column should also exceed a treshold. (The latter treshold should be lower than the former treshold.)

Secondly, the shallowness constraint on languages says that short expressions and contexts contain all grammatical information about a language. Thus initially we can restrict ourselves to short expressions and contexts. The short expressions and contexts that belong to exactly one type are considered to be *characteristic* for that type. EMILE uses these characteristic expressions and

contexts to find the long expressions and contexts that also belong to that type. EMILE assumes that any context or expression that appears with a characteristic expression or context, must be itself a context or expression for that type, regardless of length. Thus any (long) context or expression that appears with a known characteristic expression or context of a type, is a context or expression for that type. In the EMILE program, the (long) contexts and expressions identified by the characteristic expressions and contexts are called *secondary* contexts and expressions for that type, as opposed to the *primary* contexts and expressions that are found by the clustering algorithm.

2.4 Finding Rules

After the clustering EMILE transforms the grammatical types found into derivation rules. An expression e that belongs to type $[T]$ yields the rule

$$[T] \Rightarrow e \ .$$

EMILE finds more complex rules, by searching for characteristic expressions of one type that appear in the secondary expressions of another (or the same) type. For example, if the characteristic expressions of a type $[T]$ are

$$\{\text{dog}, \text{cat}, \text{gerbil}\}$$

and the type $[S]$ contains the secondary expressions

$$\{\text{I feed my dog}, \text{I feed my cat}, \text{I feed my gerbil}\}$$

EMILE will derive the rule

$$[S] \Rightarrow \text{I feed my}[T]$$

In certain cases, using characteristic and secondary expressions in this manner allows EMILE to find recursive rules. For instance, a characteristic expression of the type of sentences $[S]$ might be

$$\text{Mary drinks tea} \ .$$

If the maximum length for primary expressions is set to 4 or 5, the sentence

$$\text{John observes that Mary drinks tea}$$

would be a secondary expression for $[S]$, but not a primary or characteristic one. So if there are no other expressions involved, EMILE would derive the recursive rules

$$[S] \Rightarrow \text{Mary drinks tea}$$
$$[S] \Rightarrow \text{John observes that } [S]$$

which would allow the resulting grammar to generate the recursive sentence

$$\text{John observes that John observes that Mary drinks tea} \ .$$

The total rule generation step is efficient (requires polynomial time in the size of the grammar) as proven in [11].

3 Experiments

This section presents the results of experiments with the implemented EMILE 4.1 algorithms. The application runs on the Linux operating system (RedHat version 2.2.12-20) with 128MB RAM. A typical series (with different support percentages) of 8 times 8 experiments on 2000 lines of text data (approximately 100KB) takes 15.5 hours. On average, we can say that a single run of EMILE processes 100KB of text data in a quarter of an hour.

3.1 Experiments on a Small Data Set

We conducted several experiments with EMILE 4.1 on a small data set. The underlying grammar for the experiments was equivalent to the following context-free grammar (where the '|' symbol is used to separate alternatives).

$$S \Rightarrow \text{I cannot } V \text{ mail with } N$$
$$V \Rightarrow \text{read} \mid \text{write} \mid \text{open} \mid \text{send}$$
$$N \Rightarrow \text{MS-Mail} \mid \text{MS-Outlook} \mid \text{Mail} \mid \text{Outlook}$$

The purpose of the experiments was to identify the verb (V) and noun (N) categories. The experiments show (Tab. 9) that EMILE learns a correct grammar for small data set under various conditions. When applying EMILE with the default settings (support percentages 70, 70, 91) on the perfect sample of 16 sentences, EMILE induces a perfect grammar that identifies exactly the verb and noun categories. When EMILE is applied with more liberal settings on smaller data sets, EMILE induces the underlying grammar for the intended language.

The first, third and sixth experiments produce a grammar equivalent to the original. The six experiments show that the EMILE program also is able to generalize properly over a very small data sample. Even if only half of the intended sentences are given, EMILE induces the intended grammar when settings are chosen carefully.

3.2 Experiments on Large Data Sets

We experimented with three large data sets. The second series involves a data set of abstracts from the bio-medical (the Medline archive) domain. The final series of experiments is conducted on the Bible, King James version. The first series is an excercise for students.

A 2000 Sentence Sample. We conducted an experiment with a group of approximately 30 students. The purpose of the exercise was to reconstruct a specific grammar from a 2000 sentence sample.

Table 9. Experiment results on a small data set

A	B	C	D	E	F	G	H	I	J	K
16	70	70	91	229	139	17	2	5	16	yes
12	70	70	91	185	119	53	8	8	12	no
12	30	30	50	185	119	17	2	0	16	yes
8	70	70	91	141	99	53	8	0	8	no
8	40	40	60	141	99	37	5	10	15	no
8	50	50	70	141	99	54	6	15	16	yes

Legend
A = Num er of sentences read
B = Primary context support percentage
C = Primary expression support percentage
D = Total support percentage
E = Num er of different contexts identified
F = Num er of different expressions identified
G = Num er of different grammatical types identified
H = Num er of different dictionary types identified
I = Num er of different Chomsky rules identified
J = Size of generated language
K = Equivalent grammar

The Exercise. The students were given a sample of 2000 sentences randomly generated from the context-free grammar given in Tab. 10 (where the '|' symbol is used to separate alternatives). Using EMILE, and experimenting with the settings, the students were told to find a grammar for the sample.

The Results. On the negative side, the resulting grammars suffer from being oversized (3000 to 4000 rules). Many types appear to be slight variations of one another, all of which are used in the rules, where one variation would suffice. Clearly, the rules-finding algorithm of EMILE would benefit from being optimized to use fewer types or as few types as possible.

On the positive side, the resulting grammars are as powerful as the original grammar, i.e, generates the same constructions as the original grammar). In all cases the resulting grammars manage to capture the recursion of the grammar. In some cases, the grammars are actually stronger than the original grammar (i.e., generated new constructions). Sometimes sentences such as 'John likes the man with the house near the city with the shop' are produced. This is indicative of finding rules such as

$$[NP] \Rightarrow [NP][P][NP_p]$$

as can be induced from the rule for non-terminal $[VP_a]$.

The Bio-medical Domain. The second series of experiments involved bio-medical abstracts, an extremely complex sample. The bio-medical abstracts form

Table 10. Context-free grammar for 2000 sentence sample

$[S] \Rightarrow [NP]\ [V_i]\ [ADV] \mid [NP_a]\ [VP_a] \mid [NP_a]\ [V_s]$ that $[S]$
$[NP] \Rightarrow [NP_a] \mid [NP_p]$
$[VP_a] \Rightarrow [V_t]\ [NP] \mid [V_t]\ [NP]\ [P]\ [NP_p]$
$[NP_a] \Rightarrow$ John | Mary | the man | the child
$[NP_p] \Rightarrow$ the car | the city | the house | the shop
$[P] \Rightarrow$ with | near | in | from
$[V_i] \Rightarrow$ appears | is | seems | looks
$[V_s] \Rightarrow$ thinks | hopes | tells | says
$[V_t] \Rightarrow$ knows | likes | misses | sees
$[ADV] \Rightarrow$ large | small | ugly | eautiful

a more heterogeneous data set than the Bible in the next section. The abstracts form a total of approximately 150KB (3000 lines) of free text. The experiments show that each new abstract introduces new words and new sentences (see Fig. 1). The number of different contexts and expressions identified increases linearly with the number of sentences read.

Although the data set is by far not large enough to converge, the experiments already yield some interesting types. Grammatical type [16] in Tab. 11 shows a group of academic institutes; type [94] a group of languages from which the abstracts were translated; type [101] a group of journal issues and type [105] denotes a group of observation-verbs.

Table 11. Clusters from the Medline data set

[16] ⇒ School of Medicine, University of Washington, Seattle 98195, USA
[16] ⇒ University of Kitasato Hospital, Sagamihara, Kanagawa,Japan
[16] ⇒ Heinrich-Heine-University, Dusseldorf, Germany
[16] ⇒ School of Medicine, Chi a University
[94] ⇒ Chinese
[94] ⇒ Japanese
[94] ⇒ Polish
[101] ⇒ 32 : Cancer Res 1996 Oct
[101] ⇒ 35 : Genomics 1996 Aug
[101] ⇒ 44 : Cancer Res 1995 Dec
[101] ⇒ 50 : Cancer Res 1995 Fe
[101] ⇒ 54 : Eur J Biochem 1994 Sep
[101] ⇒ 58 : Cancer Res 1994 Mar
[105] ⇒ identified in 13 cases (72
[105] ⇒ detected in 9 of 87 informative cases (10
[105] ⇒ o served in 5 (55

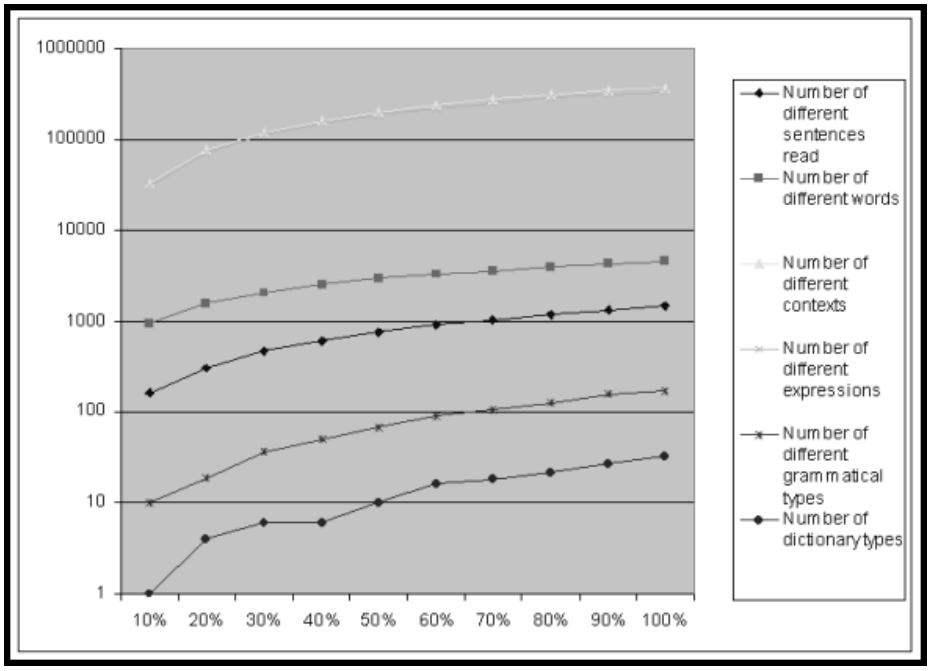

Fig. 1. The Medline experiments

The Bible Experiments. The Bible experiments show that on large homogeneous data sets (approximately 6MB of free text) the number of different sentences and words encountered starts to converge. As a consequence, also the number of different contexts and expressions starts to converge at a higher level. The experiments also identify significant shifts in style. The graph in Fig. 2 depicts a steep increase in sentences and words at Bible book 19.

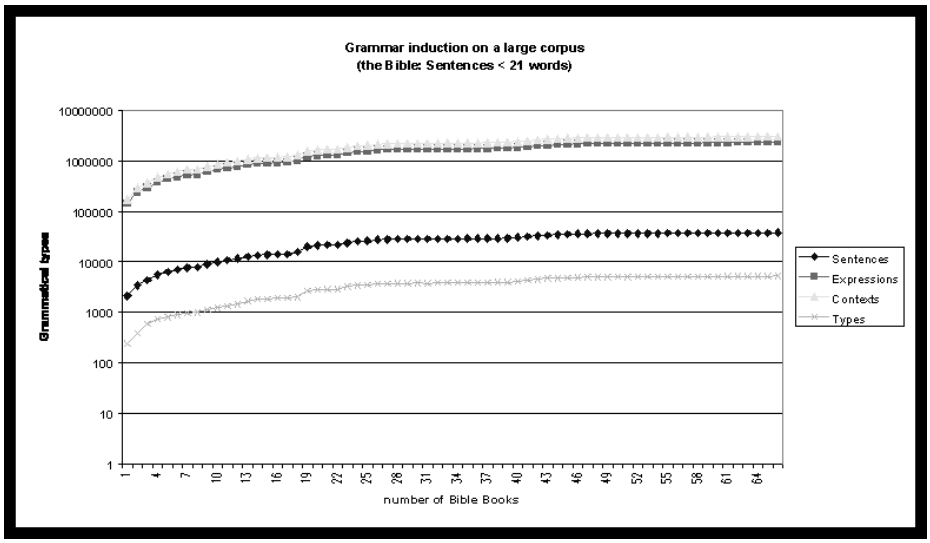

Fig. 2. The Bible experiments

The Bible experiment shows that the grammar induction starts to converge at a practical data set. This experiment falsifies the conjecture that learning natural language grammars from positive examples only is infeasible.

Table 12 presents a few types for large semantic groups of names that EMILE revealed in the Bible.

4 Future Developments

Although the initial results with EMILE 4.1 are interesting, there is still a lot of room for improvement. In particular, there should be a better algorithm to transform the grammatical types into a grammar of derivation rules. The current algorithm is a brute-force search with a few basic 'tricks' used to decrease the size of the resulting grammars. The grammars still are larger than necessary, often by a large factor. Additionally, currently the grammars constructed by EMILE are context-free: it may be possible to adapt EMILE to produce more sensible, context-sensitive, grammars.

Table 12. Clusters from the Bible data set

[76] ⇒	Esau | Isaac | Abraham | Rachel | Leah | Levi | Judah | Naphtali | Asher | Benjamin | Eliphaz | Reuel | Anah | Shobal | Ezer | Dishan | Pharez | Manasseh | Gershon | Kohath | Merari | Aaron | Amram | Mushi | Shimei | Mahli |Joel | Shemaiah | Shem | Ham | Salma | Laadan | Zophah | Elpaal | Jehieli
[414] ⇒	Simeon | Judah | Dan | Naphtali | Gad | Asher | Issachar | Zebulun | Benjamin | Gershom
[3086] ⇒	Egypt | Moab | Dumah | Tyre | Damascus

Furthermore, a better understanding of the potential application domains of EMILE is desirable. The experiments showed that EMILE can be used to extract semantic data from a text. The world wide web is an obvious application domain for EMILE, which has not yet been explored. Currently we are applying EMILE to a large body of different texts and messages: a textbook on Dutch for foreigners, bio-medical data, vocational profiles and archaeological inscriptions. EMILE might be useful as the kernel of a tool that constructs thesauri or knowledge bases from free text. EMILE can be applied to words instead of sentences and learn morphological structures. EMILE might be of help to develop mathematical models of first and second language acquisition.

5 Conclusions

Theoretically, EMILE learns the correct grammatical structure of a language from sentences of that language in the limit. The current implementation (EMILE 4.1) is one of the first efficient ([2], [11]) grammar induction algorithms that work on free text. Although EMILE 4.1 is far from perfect, it enables researchers to start empirical grammar induction research on various types of corpora. A big drawback of the current implementation is the overgeneralisation of types. EMILE almost never finds the simple basic 'sentence is noun-phrase + verb-phrase' rule when applied to real life data. In most cases it finds thousands of small variants of this rule in a large body of text. This is obviously an issue that has to be addressed.

The current EMILE 4.1 implementation learns a subclass of the shallow context-free languages that is sufficiently rich to be of practical interest in the context of syntactic and semantic analysis of large corpora. A very promising observation is that EMILE already starts to converge on data sets of moderate size like the Bible. A better understanding of the prospects of convergence on various types of text is necessary.

References

1. N. A e, *Learnability and locality of formal grammars*, in Proceedings of the 26th Annual meeting of the Association of computational linguistics, 1988.
2. P.W. Adriaans, *Language Learning from a Categorial Perspective*, PhD thesis, University of Amsterdam, 1992.
3. P.W. Adriaans, *Bias in Inductive Language Learning*, in Proceedings of the ML92 Workshop on Biases in Inductive Learning, A erdeen, 1992.
4. P.W. Adriaans, *Learning Shallow Context-Free Languages under Simple Distributions*, ILLC Research Report PP-1999-13, Institute for Logic, Language and Computation, Amsterdam, 1999.
5. P.W. Adriaans, S. Janssen, E. Nomden, *Effective identification of semantic categories in curriculum texts by means of cluster analysis*, in workshop-notes on Machine Learning Techniques for Text Analysis, Vienna, 1993.
6. P.W. Adriaans, A.K. Kno e, *EMILE: Learning Context-free Grammars from Examples*, in Proceedings of BENELEARN'96, 1996
7. W. Buszkowski, G. Penn, *Categorial Grammars Determined from Linguistic Data by Unification*, The University of Chicago, Technical Report 89-05, June 1989.
8. E. Dörnen urg, *Extension of the EMILE algorithm for inductive learning of context-free grammars for natural languages*, Master's Thesis, University of Dortmund, 1997.
9. M. Kanazawa, *Learnable Classes of Categorial Grammars*, PhD thesis, University of Stanford, 1994.
10. R. Oehrle, E. Bach, D. Wheeler (eds.), *Categorial Grammars and Natural Language Structures*, D. Reidel Pu lishing Company, Dordrecht, 1988.
11. M.R. Vervoort, *Games, Walks and Grammars: Problems I've Worked On*, PhD thesis, University of Amsterdam, 2000.