Games, walks and grammars: Problems I've worked on

Vervoort, M.R.

Publication date
2000

Chapter 13

Results: Theory and Practice

13.1 Theoretical Results

First, we give some definitions and a conjecture adapted from P. Adriaans[34]:

13.1.1. DEFINITION. Let G be a grammar (context-free or otherwise) of a language L. G has *context separability* if each type of G has a characteristic context, and *expression separability* if each type of G has a characteristic expression.

13.1.2. DEFINITION. A class of languages[37] C is *shallow* if for each language L it is possible to find a context- and expression-separable 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).

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[38], 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[39]. Hence the following conjecture seems tenable:

13.1.3. CONJECTURE. *Natural languages are (mostly) context- and expression-separable, and shallow.*

[37] Strictly speaking the shallowness property cannot be applied to single languages. However, when a language L has a grammar G and a set of sentences S as described in the definition, then the size of S relative to the logarithm of the descriptive length of L can be taken as a measure of the (un-)shallowness of S, so we can (imprecisely) speak of a language being 'very shallow' or 'not so shallow'.

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

[39] At the 1997 CSLI workshop, P. Adriaans offered a thousand dollars for a syntactical construction in any known language, that cannot be expressed in 16 words or less. Nobody has claimed this money yet. The offer is still open as of this writing.

Now, if a grammar is context- and expression-separable, then EMILE will be able to find its types given the proper settings and a sufficiently complete sample, as the following lemma shows

13.1.4. LEMMA. *Let T be a type with a characteristic context c^{ch} and a characteristic expression e^{ch}. Suppose that the maximum lengths for primary contexts and expressions are set to at least* $\text{len}(c^{ch})$ *and* $\text{len}(e^{ch})$ *and suppose that the* ***total_support%***, ***expression_support%*** *and* ***context_support%*** *settings are all set to* 100%. *Let $T_C^{\leq max}$ and $T_E^{\leq max}$ be the sets of contexts and expressions of T that are small enough to be used as primary contexts and expressions. If EMILE is given a sample containing all combinations of contexts from $T_C^{\leq max}$ and expressions from $T_E^{\leq max}$, then EMILE will find type T.*

Proof
For any type U, if c^{ch} belongs to U, then all expressions of U appear with e^{ch}, and hence are also expressions of T. Similarly, for any type U, if e^{ch} belongs to U, then all contexts of U are also contexts of T. It follows that for any type U, if U covers the context/expression pair (c^{ch}, e^{ch}), then $U_C \times U_E \subseteq T_C \times T_E$. Conversely, $T_C \times T_E$ is a type covering (c^{ch}, e^{ch}). We conclude that $T_C \times T_E$ is the unique maximal type covering (c^{ch}, e^{ch}), and hence will appear in the grammar output by EMILE.

□

Given this result, if the conjecture that natural languages are (mostly) context- and expression-separable holds, then EMILE should have the potential to learn natural languages. If natural languages are also shallow, then the required sample can be relatively small. The question whether EMILE works in practice, and what constitutes a 'small' sample, will be considered in the next two sections.

13.2 Results for a Generated Sample

The EMILE program was given 100.000 different sentences generated by the following context-free grammar:

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

(where the '|' symbol is used to separate alternatives). The EMILE program used the following settings (see appendix A for the exact meaning of all the settings):

`maximum_sentence_length` = 14	`expression_support_percentage` = 25
`maximum_primary_expr_length` = 4	`context_support_percentage` = 25
`maximum_primary_context_length` = 5	`secondary_expression_support%` = 25
`minimum_contexts_per_type` = 3	`secondary_context_support%` = 25
`minimum_expressions_per_type` = 4	`rule_support_percentage` = 25
`type_usefulness_required` = 1	`sesp_for_no_characteristics` = 34
`ruleset_increase_disallowed` = 1	`scsp_for_no_characteristics` = 26
`total_support_percentage` = 44	`rsp_for_no_characteristics` = 26

After processing 100.000 sentences, EMILE generated the following grammar:

[0] ⇒ [17] [6]
[0] ⇒ [17] [22] [17] [6]
[0] ⇒ [17] [22] [17] [22] [17] [22] [17] [6]
[6] ⇒ misses [17]
[6] ⇒ likes [17]
[6] ⇒ knows [17]
[6] ⇒ sees [17]
[6] ⇒ [22] [17] [6]
[6] ⇒ appears [34]
[6] ⇒ looks [34]
[6] ⇒ is [34]
[6] ⇒ seems [34]
[6] ⇒ [6] near [17]
[6] ⇒ [6] from [17]
[6] ⇒ [6] in [17]
[6] ⇒ [6] with [17]
[17] ⇒ the child
[17] ⇒ Mary
[17] ⇒ the city
[17] ⇒ the man
[17] ⇒ John
[17] ⇒ the car
[17] ⇒ the house
[17] ⇒ the shop
[22] ⇒ tells that
[22] ⇒ thinks that
[22] ⇒ hopes that
[22] ⇒ says that
[22] ⇒ [22] [17] [22]
[34] ⇒ small
[34] ⇒ beautiful
[34] ⇒ large
[34] ⇒ ugly

As can be seen, EMILE identifies most of the structures of the original grammar, and even manages to capture its recursive structure. Furthermore, the resulting grammar is not much larger than the original grammar. This gives hope that EMILE, or a program based on EMILE, could be used as a tool to find meaningful patterns in languages.
However, it should be noted that the grammar found by EMILE is weaker than the original grammar. For one, it does not differentiate between types of nouns, making possible sentences such as 'the car says that...'. Furthermore, in the original grammar, phrases such as 'with the car' were optional additions to certain sentences, and at most one such phrase could be appended. In the grammar found by EMILE, a second recursive structure allows any sentence to be followed by an arbitrary number of these phrases. Very likely, a higher value for 'minimum_expression_length' and higher values for the support settings will result in a grammar weakly equivalent to the original one, but for these settings, a larger sample will probably be required to achieve meaningful results.

The grammar found by EMILE contains a few superfluous rules, such as '[0] ⇒ [17] [22] [17] [22] [17] [22] [17] [6]'. This is caused by the fact that when checking which rules have been made superfluous, EMILE only checks one-step instantiations, i.e. those expressions which can be obtained from a rule by directly replacing type references with secondary expressions. To check more thoroughly, it is necessary to consider those expressions which can be obtained using successive rule substitutions, which is very expensive (in terms of computation time).[40]
To study the program's behavior as a function of the sample-size, the CFG was used to generate 1000 sentences at a time. This produced the following statistics:

number of sentences	number of types found	size of rule-set	number of types used	time used (minutes)[41]
1000	421	269	12	0
2000	647	288	19	3
3000	657	79	11	11
4000	643	82	14	30
5000	638	60	10	78
6000	566	72	12	78
7000	468	60	11	101
8000	283	98	12	98
9000	217	57	12	143
10000	195	87	12	112
11000	196	40	8	61
12000	211	68	8	45
13000	202	100	11	87
14000	214	84	11	50
15000	215	46	9	109
16000	202	41	8	157
17000	214	41	8	199
18000	211	63	10	107
19000	201	56	13	180
20000	199	66	12	169
30000	205	42	9	387
40000	180	40	6	521
50000	154	63	7	462
60000	168	38	7	773
70000	125	38	7	939
80000	112	34	6	838
90000	89	51	6	2806
100000	61	33	5	3598

[40] In fact this requires the program to solve the problem of whether two grammars are weakly equivalent, which is undecidable in general. However, in this case we can limit ourselves to expressions encountered by the program, which makes it decidable.

As can be seen, these statistics do not yield a smooth curve, although the deviations are not extravagantly large. This is probably caused by the randomizer, which is used at a number of points in the grammar deduction process, to implement nondeterministic selection. As can be seen, the number of rules found briefly increases, then drops, increases again, and then slowly drops to slightly more than 30. Something similar happens to the number of types found and the number of types used. Presumably, the increase around the 10.000 sentence-point is caused by a shift in probability distributions around then: the sentence generator is prohibited from repeating sentences, so around that point the proportion of long sentences vs. short sentences will start to change. Another observation is that, taking into account the large variations in used time caused by the changing CPU loads, the time used by EMILE does not seem to be exponential in the number of sentences, or even high-order polynomial.
We can conclude that in this experiment, the output of EMILE converges to a concise grammar, and that a sample of 30,000 sentences suffices to get good results.

Results for large real-world datasets

EMILE was given the text of the Bible (King James edition) to see if it could derive a grammar for the English language. Using the following settings, EMILE processed the 21070 different sentences of the Bible that were of length ≤ 14:

`maximum_sentence_length` = 14	`expression_support_percentage` = 40
`maximum_primary_expr_length` = 4	`context_support_percentage` = 40
`maximum_primary_context_length` = 5	`secondary_expression_support%` = 20
`minimum_contexts_per_type` = 2	`secondary_context_support%` = 20
`minimum_expressions_per_type` = 3	`rule_support_percentage` = 20
`type_usefulness_required` = 1	`sesp_for_no_characteristics` = 51
`ruleset_increase_disallowed` = 1	`scsp_for_no_characteristics` = 34
`total_support_percentage` = 64	`rsp_for_no_characteristics` = 34

The result was a grammar containing 20858 rules, only 212 less than the trivial grammar containing only the literal sentences. In fact most (20441) of the rules in the generated grammar are rules for literal sentences, such as

[0] ⇒ And the flood was forty days upon the earth :

This indicates that for most sentences, EMILE could not discern a pattern, or at least not a pattern that could be used to reduce the size of the grammar. Amongst the patterns which EMILE did manage to discover, are some which might be significant:

[41] This time should be taken as a *very* rough indication, as it is strongly influenced by the load caused by other programs running on the same computer

[0] ⇒ Thou shall not [582]	[582] ⇒ eat it :
[0] ⇒ Neither shalt thou [582]	[582] ⇒ kill .
	[582] ⇒ commit adultery .
	[582] ⇒ steal .
	[582] ⇒ bear false witness against thy neighbour .
	[582] ⇒ abhor an Edomite :

and some which are probably mere accidents:

[0] ⇒ and [72]	[72] ⇒ Er and Onan died in the land of Canaan .
[0] ⇒ but [72]	[72] ⇒ let me not fall into the hand of man .
	[72] ⇒ they could not .
	[72] ⇒ he saw :
	[72] ⇒ now murderers .
	[72] ⇒ they shall know that I am the Lord GOD .

EMILE was also run with successively larger subsets of this sample, with the following results:

number of sentences	number of types found	size of rule-set	number of types used	time used (seconds)
2107	63	2097	7	4
4214	151	4172	11	9
6321	290	6255	13	16
8428	396	8337	18	23
10535	420	10410	18	29
12642	487	12506	20	37
14749	579	14582	22	46
16856	653	16670	28	62
18963	800	18753	29	79
21070	840	20858	33	102

There is no sign of the convergence that characterized the previous experiment: presumably, the Bible simply isn't big enough as a sample of the English language. A problem with using larger samples is that the EMILE program uses a lot of memory. To analyze the Bible, EMILE needs between 100 and 250 megabytes of memory (depending on the settings used): larger samples have proportionally larger memory requirements. It may be possible to design a version of EMILE which allows for the data to be distributed over several machines. However, even if a distributed version turns out to be impractical, this is a temporary problem, given the exponential growth of available computer memory of the last few years.