Learning unification-based grammars using the Spoken English Corpus

Miles Osborne and Derek Bridge
Department of Computer Science, University of York, Heslington,
York YO1 5DD, U. K.
{miles,dgb}@minster.york.ac.uk

July 12, 1994

Abstract

This paper describes a grammar learning system that combines model-based and data-driven learning within a single framework. Our results from learning grammars using the Spoken English Corpus (SEC) suggest that combined model-based and data-driven learning can produce a more plausible grammar than is the case when using either learning style in isolation.

1 Introduction

In this paper, we present some results of our grammar learning system acquiring unification-based grammars using the Spoken English Corpus (SEC). The SEC is a collection of monologues for public broadcast and is small (*circa* 50,000 words) in comparison to other corpora, such as the Lancaster-Oslo-Bergen Corpus [JLG78], but sufficiently large to demonstrate the capabilities of the learning system. Furthermore, the SEC is tagged and parsed, thus side-stepping the problems of constructing a suitable lexicon and of creating an evaluation corpus to determine the plausibility of the learnt grammars.

In contrast to other researchers (for example [BMMS92, GLS87, Bak79, LY90, VB87]), we try to learn competence grammars and not performance grammars. We also try to learn grammars that assign linguistically plausible parses to sentences. Learning competence grammars that assign plausible parses is achieved by combining model-based and data-driven learning within a single framework [OB93b, OB93a]. The system is implemented to make use of the Grammar Development Environment (GDE) [CGBB88] and it augments the GDE with 3300 lines of Common Lisp.

Our aim in this paper is to show that combining both learning styles produces a grammar that assigns more plausible parses than is the case for grammars learnt using either learning style in isolation. Plausibility is important in Natural Language Processing as it is very rare that applications need just to determine whether a sentence is grammatical: applications need also to determine the internal structure of sentences (a plausible parse). A grammar that assigns plausible parses is therefore preferable over one that does not assign plausible parses.

The structure of this paper is as follows. Section 2 gives an overview of the combined model-based and data-driven learner. Section 3 then describes the method used to generate the results, which are then presented in section 4. Section 5 discusses these results and points the way forward.

2 System overview

2.1 Architecture

We assume that the system has some initial grammar fragment, G, from the outset. Presented with an input string, W, an attempt is made to parse W using G. If this fails, the learning system is invoked. Learning takes place through the interleaved operation of a parse completion process and a parse rejection process.

In the parse completion process, the learning system tries to generate rules that, had they been members of G, would have enabled a derivation sequence for W to be found. This is done by trying to extend incomplete derivations using what we call *super rules*. Super rules are the following unification-based grammar rules:

$$[\,] \rightarrow [\,]\,[\,] \quad \text{(binary)}$$
$$[\,] \rightarrow [\,] \quad \text{(unary)}$$

The binary rule says (roughly) that any category rewrites as any two other categories, and the unary rule says (roughly) that any category rewrites as any other category. The categories in unification grammars are expressed by sets of feature-value pairs; as the three categories in the binary super rule and two categories in the unary super rule specify no values for any of the grammar's features, these rules are the most general (or vacuous) binary and unary rules possible. These rules thus enable constituents found in an incomplete analysis of W to be formed into a larger constituent. In unifying with these constituents, the categories on the right-hand side of the super rules become partially instantiated with feature-value pairs. Hence, these rules ensure that at least one derivation sequence will be found for W.

Many instantiations of the super rules may be produced by the parse completion process described above. Linguistically implausible instantiations must

be rejected and we interleave this rejection process with the parse completion process. Rejection of rules is carried out by the model-driven and data-driven learning processes described below. Note that both of these processes are modular in design, and it would be straightforward to add other constraints, such as lexical co-occurrence statistics or a theory of textuality, to help select correct analyses.

If all instantiations are rejected, then the input string W is deemed ungrammatical. Otherwise, surviving instantiations of the super rules used to create the parse for W are regarded as being linguistically plausible and may be added to G for future use.

2.2 Model-driven learning

A grammatical *model* is a high-level theory of syntax. In principle, if the model is complete, an 'object' grammar could be produced by computing the 'deductive closure' of the model (e.g. a 'meta'-rule can be applied to those 'object' rules that account for active sentences to produce 'object' rules for passive sentences). An example of purely model-based language learning is given by Berwick [Ber85]. More usually, though, the model is incomplete and this leads us to give it a different rôle in our architecture.

Our model currently consists of GPSG Linear Precedence (LP) rules [GKPS85], semantic types [Cas88], a Head Feature Convention [GKPS85] and X-bar syntax [Jac77].

- *LP rules* are restrictions upon *local trees*. A local tree is a (sub)tree of depth one. An example of an LP rule might be [GKPS85, p.50]:

  $$[\text{SUBCAT}] \prec \sim [\text{SUBCAT}]$$

  This rule should be read as 'if the SUBCAT feature is instantiated (in a category of a local tree) then the SUBCAT feature of the linearly preceding category should not be instantiated'. The SUBCAT feature is used to help indicate minor lexical categories, and so this rule states that verbs will be initial in VPs, determiners will be initial in NPs, and so on. In our learning system, any putative rule that violates an LP rule is rejected.

- We construct our syntax and semantics in tandem, adhering to the *principle of compositionality*, and pair a semantic rule to each syntactic rule [DWP81]. Our semantics uses the typed λ-calculus with extensional typing. For example, the syntactic rule:

  $$S \rightarrow NP\ VP$$

  is paired with the following semantic rule:

VP(NP)

which should be read as 'the functor **VP** takes the argument **NP**'[1]. The functor **VP** is of type[2]:

$$<<<e,t>,t>,t>$$

and the argument **NP** is of type:

$$<<e,t>,t>$$

The result of functionally applying **VP(NP)** has the type:

$$t$$

For many newly-learnt rules, we are able to check whether the semantic types of the categories can be functionally applied. If they cannot, then the syntactic rule can be rejected. For example, the syntactic rule:

$$VP \rightarrow VP\ VP$$

has the semantic rule **VP(VP)**, which is ill-formed because the type

$$<<<e,t>,t>,t>$$

cannot be functionally applied with itself.

- Head Feature Conventions (HFCs) help instantiate the mother of a local tree with respect to immediately dominated daughters. For example, the verb phrase dominating a third person verb is itself third person.
- X-bar syntax specifies a restriction upon the space of possible grammar rules. Roughly speaking, the RHS of a rule contains a distinguished category called the *head* that characterises the rule. The LHS of the rule is then a *projection* of the head. Projecting the head category results in a phrasal category of the same syntactic class as that of the head. For example, the rule $NP \rightarrow Det\ N1$ has a nominal head and a NP projection.

Model-based learning consists of filtering out instantiations of the super rules that violate any aspect of the model, or refining instantiation of a super rule such that they comply with some aspect of the model. LP rules and semantic types filter instantiations, whilst the Head Feature Convention and X-bar syntax refine instantiations.

[1]Syntactic categories are written in a normal font and semantic functors and arguments are written in a **bold** font.

[2]The exact details of these types is not important to understanding the thrust of this section and so they are not given any detailed justification.

2.3 Data-driven learning

Our data-driven component can prefer learnt rules that are 'similar' to rules previously seen by the parser. For this to work at all well, the system will need some prior training using a pre-training corpus. This can then be used in subsequent learning to score instantiations of the super rules.

In pre-training the frequencies of mother-daughter pairs (MDPs) found in parses of sentences taken from the pre-training corpus are recorded [LG91]. For example, the tree (S (NP Sam) (VP (V laughs))) has the following MDPs:

<S,NP>
<S,VP>
<VP,V>

The frequencies of these MDPs in the parse trees of the pre-training corpus are noted. From these frequencies, the score of each distinct MDP can be computed: if pair <A, B> occurs with frequency n out of a total number of N MDPs, then the MDP's score, f, is:

$$f(< A, B >) = n/N$$

The set of MDP frequencies is computed in advance of using our system for learning. During learning, after parse completion by the super rules, local trees in completed parses can be scored. The score is computed recursively, as follows:

- For local trees of the form (A (B C)) whose daughters are leaves, the score of the local tree is:

$$score(A) = gm(f(< A, B >), \\ f(< A, C >))$$

  where gm is the geometric mean. We take the geometric mean, rather than the product, to avoid penalising local trees that have more daughters over local trees that have fewer daughters [MM91].

- For interior trees of the form (B (C D)), the score of the local tree is:

$$score(B) = gm(score(C) \times f(< B, C >), \\ score(D) \times f(< B, D >))$$

(This does leave the problem of dealing with MDPs that arise in completed parses but which did not arise in the pre-training corpus. These can be given a low score. Giving them a score ensures that all trees can be scored, and thus the data-driven learner is 'complete', i.e. it can always make a decision.)

After scoring, instantiations of the super rule that have daughters whose scores exceed some threshold can be accepted. Other instantiations can be rejected. The higher the threshold, the fewer the number of rules accepted[3].

The approach we have described is a generalisation of the work of Leech, who uses a simple phrase structure grammar, whereas we use a unification-based grammar [Lee87].

3 Method

We predicted that the plausibility of grammars learnt using both model-based and data-driven learning would be better than the plausibility of grammars obtained by using either learning style in isolation. Plausibility is determined as how 'close', for the same sentence, a test parse is to a benchmark parse, taken (in our case) from the SEC. The following algorithm defines closeness between the test tree (T) and the benchmark tree (B):

- Each tree is normalised to use the same labelling scheme.
- The list L_T is a preorder walk of T and the list L_B is a preorder walk of B.
- Construct the set of lists M as follows. Find β, the longest list that is common to both L_T and L_B and add β to M. Remove β from L_T. Repeat removing lists until either L_T is the empty list or no list can be found that is both in L_T and L_B.
- Closeness is then the arithmetic mean of the list lengths of M divided by the list length of L_B. The nearer this figure is to unity, the better the match. A figure of 0 indicates no match at all.

For example, if L_T was the list (a b c d) and L_B the list (c a b c), then β would initially be (a b c). Removing β from L_T results in L_T becoming the list (d). As there are no lists common to both L_T and L_B, matching halts, with M being {(a b c)}. The closeness score would then be 3/4.

The matching algorithm is designed to allow a certain degree of fuzziness in matching. For example, it is the case that manually produced trees in the SEC are relatively shallow, whilst those generated using the learnt grammars are steep. However, taking a preorder of the trees and searching for longest common lists helps overcome this in-built mismatch. The matching algorithm is our attempt to strike a pragmatic balance between computational efficiency and achieving reliable matches.

To test the prediction, the following steps were taken:

[3]We have not investigated the effect of varying the threshold. Clearly, this would be interesting future work.

- Three disjoint sets of sentences were arbitrarily selected from the SEC. These were *pretrain* (less than 20 sentences), *train* (60 sentences) and *test* (60 sentences).

- A grammar, G, was used as the initial grammar. This was manually constructed and consisted of 97 unification-based rules with a terminal set of the CLAWS2 tagset [BGL93].

- The Model was configured to consist of 4 LP rules, 32 pairings of semantic types and corresponding syntactic categories, and a Head Feature Convention.

- *Pretrain* was used to calculate scores of MDPs, thus providing an initial estimate of grammaticality for the data-driven learner.

- *Train* was then processed using interleaved parsing and learning with the following configurations of the learner:

Configuration	Grammar produced
(A) No learning	G
(B) Data-driven learning only	G1
(C) Model-based learning only	G2
(D) Both learning styles together	G3

  Note that X-bar syntax is such a vital aspect of acquiring plausible grammars that it is not optional and hence all configurations used this aspect of the model. Configuration A is the base case for comparison with the other configurations.

- *Test* was then parsed, without learning, using each of these grammars and the number of sentences successfully parsed was recorded.

- The set of sentences *plausible* was created as being 15 sentences in *test* that could be generated by grammars G1, G2 and G3. *Plausible* contained no sentence that could be generated by grammar G and hence guaranteed that each sentence needed at least one learnt rule in order to be generated. As a yardstick, 15 other sentences (*yardstick*) that could be generated using G were selected from *test*.

- *Plausible* was then parsed using grammars G1, G2 and G3 and the first 10 parses produced for each sentence was sampled. Out of these 10 parses, the score of the most plausible parse was noted.

- *Yardstick* was parsed using grammar G and the same process was carried out to derive 10 plausibility scores.

Learning grammars in the manner outlined previously is computationally intractable. For example, using the binary super rule may lead to a number of parses equal (at least) to the Catalan series with respect to sentence length. This is because, as a worst case, the binary super rule will create all possible binary branching parses for some sentence [CP82]. In order to generate results therefore, steps were taken to place resource bounds upon the learning process. These bounds were to halt when n parses or m edges had been generated (n=1, m=3000) for some sentence. Increasing n leads to more ambiguous attachments being learnt. The motivation for the m limit follows from Magerman and Weir who suggest that large numbers of edges being generated might correlate with ungrammaticality [MW92]. In effect, the parser spends a lot of time searching unsuccessfully for a parse and this is reflected in the large number of edges generated. The other constraint upon the system was that we only used the binary super rule during interleaved parsing and learning. This is because use of the unary rule greatly increases the search space that needs to be explored. The effect of only learning binary rules, however, will be to decrease the plausibility of the parses produced.

4 Results

In the following table, showing some characteristics of the various grammars, the size column is the number of rules in the grammar, coverage is the percentage of sentences in *test* generated by each grammar, and plausibility is the arithmetic mean of the closeness scores of *yardstick* using G and *plausible* with G1, G2 and G3.

Configuration	Size	Coverage	Plausibility
A	97	26.7	0.103
B	129	75.0	0.086
C	128	65.0	0.095
D	129	75.0	0.098

5 Discussion

From the previous table, it is clear that extending the initial grammar G using learning reduced G's undergeneration considerably. For example, G could only parse 26.7% of the sentences in *test*, but G3 could parse 75.0% of these sentences. As predicted, combining model-based and data-driven learning produces a grammar that assigns more plausible parses than do grammars learnt using either approach in isolation (as shown by the plausibility score for configuration D being higher than the score for configuration B or C). Learnt grammars are less plausible than the original manually constructed grammar (again, as shown

by comparing the plausibility score for configuration A with that of the other configurations). The low score given to grammar plausibility is due to difficulties in matching the fine-grained, steep parses produced by the unification-based grammar with the coarse-grained, shallow parses that were manually constructed for the SEC sentences. The uneven quality of the SEC parses does not help in plausibility determination. However, the plausibility results are encouraging and suggest that using both learning styles together is a viable way of allowing formal grammars to be used for corpus parsing.

Future work will evaluate how much the learnt grammars overgenerate. We also intend investigating other constraints upon grammaticality, such as to be found in Government and Binding Theory [Cho81], punctuation [Num90], or textuality [HH76, dBD81]. Furthermore, we intend to consider using a lexically-based formalism in place of the current rule-orientated formalism currently used.

6 Acknowledgements

We would like to thank Eric Atwell (Leeds University) for allowing access to the SEC, the anonymous referee for providing comments upon this paper, and Ted Briscoe (Cambridge University) for supplying the grammar G. The first author is supported by a Science and Engineering Research Council grant.

References

[Bak79] J. K. Baker. Trainable grammars for speech recognition. In D. H. Klatt and J. J. Wolf, editors, *Speech Communication Papers for the* 97^{th} *Meeting of the Acoustical Society of America*, pages 547–550. 1979.

[Ber85] Robert C. Berwick. *The acquisition of syntactic knowledge*. MIT Press, 1985.

[BGL93] Ezra Black, Roger Garside, and Geoffrey Leech, editors. *Statistically driven computer grammars of English the IBM-Lancaster approach*. Rodopi, 1993.

[BMMS92] Eric Brill, David Magerman, Mitchell Marcus, and Beatrice Santorini. Deducing Linguistic Structure from the Statistics of Large Corpora. In *AAAI-92 Workshop Program: Statistically-Based NLP Techniques, San Jose, California*, 1992.

[Cas88] Claudia Casadio. Semantic Categories and the Development of Categorial Grammars. In Richard T. Oehrle, editor, *Categorial Grammars and Natural Language Structures*, pages 95–123. D. Reidel, 1988.

[CGBB88] John Carroll, Claire Grover, Ted Briscoe, and Bran Boguraev. A Development Environment for Large Natural Language Grammars. Technical report number 127, University of Cambridge Computer Laboratory, 1988.

[Cho81] Noam Chomsky. *Lectures on Government and Binding.* Dordrecht: Foris, 1981.

[CP82] K. Church and R. Patil. Coping with syntactic ambiguity or how to put the block in the box on the table. *Computational Linguistics*, 8:139–49, 1982.

[dBD81] Robert de Beaugrande and Wolfgang Dressler. *Introduction to Text Linguistics.* Longman, 1981.

[DWP81] D.R. Dowty, R.E. Wall, and S. Peters. *Introduction to Montague Semantics.* D. Reidel Publishing Company, 1981.

[GKPS85] G. Gadzar, E. Klein, G.K. Pullum, and I.A. Sag. *Generalized Phrase Structure Grammar.* Harvard University Press, 1985.

[GLS87] R. Garside, G. Leech, and G. Sampson, editors. *The Computational Analysis of English: A Corpus-based Approach.* Longman, 1987.

[HH76] M. A. K. Halliday and Ruqaiya Hasan. *Coherence in English.* Longman, 1976.

[Jac77] Ray S. Jackendoff. *X-Bar Syntax: A Study of Phrase Structure.* The M.I.T Press, 1977.

[JLG78] S. Johansson, G. Leech, and H. Goodluck. Manual of Information to Accompany the Lancaster-Oslo/Bergen Corpus of British English, for Use with Digital Computers. Technical report, Department of English, University of Oslo, 1978.

[Lee87] Fanny Leech. *An approach to probabilistic parsing.* MPhil Dissertation, 1987. University of Lancaster.

[LG91] Geoffrey Leech and Roger Garside. Running a grammar factory: The production of syntactically analysed corpora or "treebanks". In Stig Johansson and Anna-Brita Stenström, editors, *English Computer Corpora: Selected Papers and Research Guide.* Mouten de Gruyter, 1991.

[LY90] K. Lari and S. J. Young. The estimation of stochastic context-free grammars using the Inside-Outside Algorithm. *Computer Speech and Language*, 4:35–56, 1990.

[MM91] D. Magerman and M. Marcus. Pearl: a probabilistic chart parser. In *Proceedings of the* 2^{nd} *International Workshop on Parsing Technologies, Cancun, Mexico*, pages 193–199, 1991.

[MW92] David Magerman and Carl Weir. Efficiency, Robustness and Accuracy in Picky Chart Parsing. In *Proceedings of the* 30^{th} *ACL, University of Delaware, Newark, Delaware*, pages 40–47, 1992.

[Num90] G. Numberg. *The linguistics of punctuation.* Center for the Study of Language and Information, 1990.

[OB93a] Miles Osborne and Derek Bridge. Inductive and deductive grammar learning: dealing with incomplete theories. In *Grammatical Inference Colloquim, Essex University*, 1993.

[OB93b] Miles Osborne and Derek Bridge. Learning unification-based grammars and the treatment of undergeneration. In *Workshop on Machine Learning Techniques and Text Analysis, Vienna, Austria*, 1993.

[VB87] Kurt Vanlehn and William Ball. A Version Space Approach to Learning Context-free Grammars. *Machine Learning*, 2.1:39–74, 1987.