Inductive and deductive learning of grammar: dealing with incomplete theories

Miles Osborne and Derek Bridge[1]

1 Introduction

In this paper, we present a framework for learning plausible unification-based natural language grammars. We assume that the system will have some initial unification-based grammar [Shi86] [OB93], and we use learning to overcome the incompleteness of this grammar (its *undergeneration*, i.e. where it fails to generate strings that humans would regard as grammatical).

In contrast to other researchers, we do not believe that formal grammars need be abandoned in favour of stochastic grammars [LF79] [GLS87] [SO90] [LY90] [O'D91] [MM91] [BW92] [Sam87]. We believe that undergeneration can be treated by learning the missing rules. Our approach is to use both model-driven (deductive) and data-driven (inductive) learning. Our framework requires no *a priori* decision to be made about the balance between being model-driven or data-driven. We can experiment using anything from being purely model-driven to being purely data-driven. We expect the data-driven learning to compensate for weaknesses of the model-driven learning (most notably the problems of incompleteness of the model), and we expect the model-driven learning to compensate for weaknesses of the data-driven learning (most notably the likelihood that data-driven learning will learn a linguistically implausible grammar).

The structure of this paper is as follows. Section 2 gives an overview of our learning framework. Section 3 then describes the model-driven learning, and Section 4 describes data-driven learning. Finally, Section 5 gives some tentative conclusions.

2 System Overview

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. First, 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 the *super rule*. The super rule is the following unification grammar rule:

$$[\,] \rightarrow [\,]\,[\,]$$

This rule says (roughly) that any category rewrites as any two other categories. The categories in unification grammars are expressed by sets of feature-value pairs; as the three categories in the super rule specify no values for any of the grammar's features, this rule is the most general (or vacuous) binary rule possible. This rule thus enables any two adjacent constituents found in an incomplete analysis of W to be formed into a larger constituent. (Note that in unifying with these

[1] Authors' address:
Department of Computer Science,
University of York,
York YO1 5DD,
U.K.

two constituents, the categories on the right-hand side of the super rule become partially instantiated with feature-value pairs.) Hence, this rule ensures that at least one derivation sequence will be found for W.[2]

Many instantiations of the super rule may be produced by the parse completion process that we have described above. Linguistically implausible instantiations must be rejected. (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 that we describe below.

If no instantiation is plausible, then the input string W is deemed ungrammatical. Otherwise, the surviving instantiations of the super rule are linguistically plausible and may be added to G for future use.

The system is implemented to make use of the chart parser of the Grammar Development Environment (GDE) [CGBB88]. The chart parser gives us all possible substrings for a sentence and, from these substrings, rules are learnt. Our implementation is not fully tested as yet. It augments the GDE with 2000 lines of AKCL Common Lisp and runs on a Sun 3/50 workstation.

3 Model-driven learning

Here we explain what we mean by a 'model', and we illustrate how this can reject linguistically implausible instantiations of the super rule.

3.1 Rule deduction

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].

Our model currently consists of GPSG Linear Precedence (LP) rules [GKPS85] and semantic types [Cas88].

- *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 are 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:

  $$S \rightarrow NP\ VP$$

[2] In using a *binary* super rule, we are prepared to accept the position that what, on other analyses, might be an n-ary, $n > 2$, analysis (e.g. a ditransitive verb and its two nominal complements) will, for us, have to be broken down into a binary branching structure. Note also that we have as yet to fully work out the details of the role of *unary* rules in this work.

is paired with the following semantic rule:

$$\mathbf{VP(NP)}$$

which should be read as 'the functor **VP** takes the argument **NP**'[3]. The functor **VP** is of type[4]:

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

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

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

The result of composing **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 composed. 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 composed with itself).

These are the two components of our model that we have investigated so far. We intend to consider some of the many other grammatical theories when determining how to augment our model.

3.2 An example of model-driven learning

Here is an example of our model in action. We shall learn the rule:

$$N1 \rightarrow Adj\ N1$$

which is (explicitly) missing from the following grammar[5]:

$S \rightarrow NP\ VP$	Sam : NP
$NP \rightarrow Det\ N1$	chases : V
$VP \rightarrow V\ NP$	the : Det
$N1 \rightarrow N0$	happy : Adj
$NP \rightarrow N1$	cat : $N0$

Note that the lexical entry for the adjective is present. The trace is taken using the Grammar Development Environment's chart parser [CGBB88].

The following input string:

Sam chases the happy cat

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

[4] 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.

[5] From now one we shall use a phrase structure grammar as notational abbreviation of a unification-based grammar.

cannot be parsed by the initial grammar as there is no rule that accounts for adjectives:

```
4 Parse>> Sam chases the happy cat
350 msec CPU, 567 msec elapsed
13 edges generated
No parses
```

Suppose we had no model. The super rule, [ ] → [ ] [ ], would be added to the grammar and all possible parses would be generated:

```
6 Parse>> Sam chases the happy cat
2809467 msec CPU, 167781083 msec elapsed
10870 edges generated
8619 parses
```

Somewhere within this set of 8619 parses may be the correct parse(s). Now we will show how having a model restricts this.

Suppose our model contains just the LP rule that we mentioned earlier. Instantiations of the super rule that violate this LP rule will be rejected as they are generated:

```
10 Parse>> Sam chases the happy cat
559150 msec CPU, 15616100 msec elapsed
4118 edges generated
3375 parses
```

The number of parses has been halved and, because bad rules are rejected immediately, the search space is approximately a sixth of the space when using no model.

Suppose instead that our model contains only the semantic type checking described earlier:

```
16 Parse>> Sam chases the happy cat
4367 msec CPU, 30233 msec elapsed
31 edges generated
1 parse
```

Only a single parse is generated. The search space is also drastically smaller than the previous examples.

If our model contains *both* the LP rule and semantic type checking, then a single parse is generated but after exploring even less of the search space:

```
22 Parse>> Sam chases the happy cat
1333 msec CPU, 2733 msec elapsed
31 edges generated
1 parse
```

From this successful parse, we would find the successful instantiation of the super rule:

$$N1 \rightarrow Adj\ N1$$

and we can add this to the grammar for future use. (Remember that we are using a unification grammar formalism, so the rule will not have atomic categories as shown, but sets of feature-value pairs as its categories.)

4 Data-driven learning

Here we explain what we mean by data-driven learning, and we illustrate how this can help to compensate for incompleteness of the model. Our implementation of data-driven learning is not fully tested. The ideas reported here are provisional.

4.1 Rule induction

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 training corpus. This can then be used to score instantiations of the super rule. (Note that the training set is initially the training corpus but is updated as the system encounters more texts.)

The learner is trained by recording the frequencies of *bigrams* (mother-daughter pairs) found in parses of sentences taken from the training corpus [LG91]. For example, the tree in figure 1a has the following bigrams:

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

The frequencies of these bigrams in the parses of the corpus are noted. From these frequencies, the probability of each distinct bigram can be computed: if pair $< A, B >$ occurs with frequency n out of a total number of N bigrams, then the bigram's probability, f, is:

$$f(< A, B >) = n/N \tag{1}$$

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

- For local trees whose daughters are leaves, as in figure 1b, the score of the local tree is:

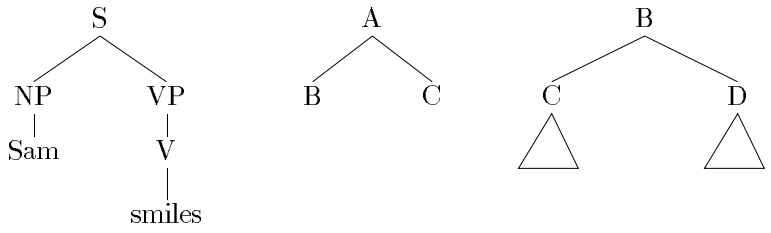
$$score(A) = gm(f(< A, B >), f(< A, C >)) \tag{2}$$

  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, as in figure 1c, the score of the local tree whose root is B is:

$$score(B) = gm(score(C) \times f(< B, C >), score(D) \times f(< B, D >)) \tag{3}$$

(This does leave the problem of dealing with bigrams that arise in completed parses but which did not arise in the 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.)



Figures 1a, 1b and 1c

After scoring, instantiations of the super rule that have daughters whose scores exceed some threshold can be accepted. Other instantiations can be rejected.

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].

Before we show the data-driven component in operation, we describe a problem which it will help to overcome.

4.2 Determining the left-hand side of a rule

Suppose we have found two adjacent constituents having the following categories:

$$A1\ N1$$

and we have invoked use of the super rule for parse completion. The learnt rule, being an instantiation of the super rule, will have as its right-hand side (RHS) the result of unifying [ ] [ ] (RHS of the super rule) and A1 N1, this result being A1 N1. However, this unification does not tell us what the left-hand side (LHS) of the learnt rule should be.

In this example, there are four possible LHS categories according to whether the rule is recursive or not, and according to whether the head of the rule is taken to be nominal or adjectival. Thus, the LHS could be either a finite nominal category (NP), a recursive nominal category (N1), a finite adjectival category (AP), or a recursive adjectival category (A1).[6]

In our system, we capture these possibilities by using disjunctive categories. The learnt rule will be:

$$\{A1 \vee N1 \vee AP \vee NP\} \rightarrow A1\ N1$$

It is now clear that our model-driven and data-driven components need both to reject implausible instantiations of the super rule, and to strike out implausible disjuncts from disjunctive categories.

In our example of using the model-driven component in Section 3.2, we glossed over this problem. In fact, in that case, we should have shown the system learning a rule with a disjunctive LHS category. But, then the system can determine that, of the four possibilities, the LHS of the learnt rule should be N1 as this can be used by the already existing rule NP $\rightarrow$ Det N1 to form an NP, which itself can be used to complete the parse.

However, the case when the grammar and grammar principles (model) are *not* sufficient to make this choice gives a good illustration of the use of the data-driven component.

4.3 An example of data-driven learning

We will show how the data-driven component might refine a disjunctive rule that has been learnt by the model-driven component. Given that part of this is the task of deciding whether the learnt rule should be recursive or not, it is worth mentioning Gold's theorem, which states that a text alone is insufficient to determine when a rule should be recursive [Gol67]. It is our hope that the combined contributions of the model-driven and data-driven components might lead to more reliable decisions.

We will use an example for which the model and initial grammar are not able to determine the LHS of the learnt rule. The example involves learning a rule to handle sentential adverbs.[7]

Faced with the sentence:

[6] This limitation to four possibilities is founded upon assumptions made in almost all modern grammar theories.

[7] Note that this example is contrived and for illustrative purposes only.

Unfortunately Sam died

the system parses *Unfortunately* as an Adv and *Sam died* as S, and, in combining these two constituents into a single constituent, the super rule is instantiated to the following disjunctive learnt rule:

$$\{S \ \vee \ S1 \ \vee \ Adv1 \ \vee \ Adv\} \rightarrow Adv \ S$$

If we assume that the initial training set was empty, the scores we would compute for the four rules encoded disjunctively above would be based solely on the tree for the sentence *Unfortunately Sam died* and would be as follows:

Rule	Score
$S \rightarrow Adv \ S$	1/11
$S1 \rightarrow Adv \ S$	1/11
$Adv \rightarrow Adv \ S$	1/11
$Adv1 \rightarrow Adv \ S$	1/11

At this stage there is no preference for any of the possibilities.

If the next sentence encountered is:

Briefly therefore Sam died [8]

This sentence *can* be parsed given that the disjunctive rule has been added to the grammar. The bigram in the resulting tree are used to update the rule's scores:

Rule	Score
$S \rightarrow Adv \ S$	1/8
$S1 \rightarrow Adv \ S$	1/12
$Adv \rightarrow Adv \ S$	1/12
$Adv1 \rightarrow Adv \ S$	1/12

The second application of the disjunctive rule that enables *briefly* to be combined with *therefore Sam died*, constrains the category of *therefore Sam died* to be S (the second daughter in the disjunctive rule). Hence, the score for $S \rightarrow Adv \ S$ is higher than the scores for the other possibilities. Thus, at this stage, data-driven learning has revealed a *preference* for thr first of our four disjuncts.

5 Conclusion

We have presented a framework in which the incompleteness of an initial grammar is overcome by a learning process. The learning process uses model-driven and data-driven components to reject linguistically implausible learnt rules. The hope is that the data-driven learning will help to overcome the incompleteness of the model.

Our ideas can easily be related to other work in machine learning. What we call 'model-driven' learning can be seen as *Explanation-Based Learning* (EBL) [Ell89] and what we call 'data-driven' learning can be seen as *Similarity-Based Learning* (SBL) [Leb90] [OM90] [FD89]. Our framework is an attempt to deal with the *incomplete domain theory problem* [MKKC86]: we are investigating the relationship of EBL to SBL and dealing with incompleteness in two theories (the grammar and the model).

[8] Some readers might feel that this sentence is unlikely to occur naturally. We stress, again, that our example is being used only to illustrate the contribution that data-driven learning can make.

The components of an implementation of this framework are in place. The data-driven component needs further work, and then the whole system needs testing. Once these tasks have been carried out, we can begin to evaluate the framework. (We have already given quite some thought to the design of suitable experiments.)

As part of this future evaluation work, we must take a critical look at three assumptions that we have made:

- We assume that every word the system will encounter has an entry in the lexicon. This may be a wrong assumption: Briscoe and Waegner suggest that the major cause of undergeneration will be lexical and not syntactic [BW92]. On the other hand, we believe that lexical acquisition is the problem of lesser difficulty: the Core Language Engine [Als92], to take an example system, has a lexical acquisition component, which uses machine readable dictionaries, lexical inference, lexical correction and a battery of other techniques to deal with lexical incompleteness.

- We assume a relatively simple representation of lexical information. We note that in certain lexically-orientated grammar formalisms, such as Head-driven Phrase Structure Grammar, information that is traditionally encoded in phrase structure rules is encoded within the lexical entries [PS87]. If we were to adopt such a richly specified lexicon, then we would not need to learn many rules. We have not adopted such a formalism, partly because the lexical acquisition systems that we mentioned in the previous bullet point cannot use such formalisms.

- We assume that the feature space is complete, i.e. we assume that the system does not need to learn new features. In support of this we note that Briscoe and Waegner's experience of extending a grammar was that their feature space did not need to be extended [BW92].

These assumptions are contentious and need further justification. However, we believe that our approach has much intrinsic interest both to those interested in learning grammar in particular and to those interested in machine learning in general.

References

[Als92] Hiyan Alshawi, editor. *The CORE Language Engine*. The MIT Press, 1992.

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

[BW92] Ted Briscoe and Nick Waegner. Robust Stochastic Parsing Using the Inside-Outside Algorithm. In *Proceedings of the workshop on statistically-based techniques in Natural Language Processing, 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.

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

[Ell89] T. Ellman. Explanation-based learning: a survey of programs and perspectives. *ACM Computing Surveys*, 21:163–222, 1989.

[FD89] Nick Flann and Tom Dietterich. A Study of Explanation-based methods for Inductive learning. *Machine Learning*, 4:187–226, 1989.

[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.

[Gol67] E. M. Gold. Language Identification to the Limit. *Information and Control*, 10:447–474, 1967.

[Leb90] Michael Lebowitz. The Utility of Similarity-Based Learning in a World needing Explanation. In Yves Kodratoff and Ryszard Michalski., editors, *Machine Learning: An Artifical-Intelligence Approach*, volume III, pages 399–422. Morgan Kaufmann, 1990.

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

[LF79] S. Y. Lu and K. S. Fu. Stochastic tree grammar inference for texture synthesis and discrimination. *CGIP*, 9:234–245, 1979.

[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.

[MKKC86] T. Mitchell, R. Keller, and S. Kedar-Cabelli. Explanation-based generalization: A unifying view. *Machine Learning*, 1.1:47–80, 1986.

[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.

[OB93] 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.

[O'D91] Tim O'Donoghue. *The Vertical Strip Parser: A Lazy Approach to Parsing*. Report 91.15, University of Leeds School of Computer Studies, 1991.

[OM90] Dirk Ourston and Raymond Mooney. Changing the Rules: A Comprehensive Approach to Theory Refinement. In *Proceedings of the 8^{th} National Conference on Artifical Intelligence,*, pages 815–820, 1990.

[PS87] Carl Pollard and Ivan A. Sag. *Information-Based Syntax and Semantics: Volume 1: Fundamentals*. Center for the Study of Language and Information, 1987.

[Sam87] G. Sampson. Evidence against the 'Grammatical/Ungrammatical' Distinction. In W. Meijs, editor, *Corpus Linguistics and Beyond*. Rodopi, 1987.

[Shi86] Stuart M. Shieber. *An Introduction to Unification-Based Approaches to Grammar.* Center for the Study of Language and Information, 1986.

[SO90] Clive Souter and Tim O'Donoghue. *Probabilistic Parsing in the COMMUNAL Project.* Report 90.2, University of Leeds School of Computer Studies, 1990.