Learning unification-based grammars and the treatment of undergeneration

Miles Osborne
Department of Computer Science
University of York
Heslington
York 5DD
England

Derek Bridge
Department of Computer Science
University of York
Heslington
York 5DD
England

Abstract

We present a framework for learning plausible unification-based natural language grammars. Our framework uses both model-based and data-driven learning without being committed to any particular configuration of these two learning schemes. We use learning to overcome the problem of undergeneration in natural language grammars. This paper presents work that is still in progress: the model-based learning component has been built but the data-driven learning component has not. Full evaluation of the framework awaits a complete implementation.

1 Introduction

1.1 Undergeneration

An application of learning natural language grammars is the treatment of undergeneration. A grammar *undergenerates* when it fails to generate some sentence which human informants judge to be grammatical. Undergeneration undermines the successful processing of natural language.

Consider the grammar:

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

This grammar does not contain the rule:

$N1 \rightarrow Adj\ N1$

and so it would generate the sentence:

Sam chases the cat

but not:

Sam chases the happy cat

This is an example of undergeneration. Any attempt to deal computationally with unrestricted natural language must face this problem.

Undergeneration has many causes:

- The engineering task of writing a grammar is made difficult by the syntactic complexity of natural languages. Deliberate or accidental omissions are therefore commonplace.
- Empirical work on building large grammars by Sampson et al. led to the conclusion that finite grammars do not exist [Sam87]. This is because people will always generate sentences containing syntactic novelties. Sampson's position is controversial (pace Briscoe et al. [Bri87]).
- Grammars are static theories which do not reflect the dynamics of language change.

Reactions towards undergeneration vary [Osb92]:

- Formal grammars can be abandoned in favour of stochastic grammars. A stochastic grammar will assign any sentence, no matter how ungrammatical, a measure of grammaticality. In this way undergeneration is replaced by overgeneration. Examples of such an approach include [LF79] [GLS87] [SO90] [LY90] [O'D91] [MM91].
- Formal grammars can be manually extended to cope with each individual case of undergeneration. An example of manual extension occurred when the Alvey Natural Language Tools Grammar [GBCB92] was used to parse dictionary definitions [Bri87]. Manual extension is labour intensive, tedious, skilled work, and may be a never ending task.

- The grammar can be dynamically extended. An example of this is Weischedel's *Meta-rules*. The meta-rules extend the grammar in an unprincipled manner. As a result the extended grammar will admit ungrammatical sentences [Wei83].
- Procedures can be written which can map input strings that are not covered by the grammar into sentences that are covered by the grammar. This assumes we can determine the intended syntactic form of the sentence (which is a strong assumption to make). Examples of this approach are NOMAD and error-correcting parsers for computer languages [Gra83] [HRS84].

In our approach we overcome undergeneration by learning grammatical rules during parsing. It is useful to make a distinction between *performance* and *competence grammars* [Lyo68]. A good competence grammar does not overgenerate (in the sense that all sentences that humans judge to be grammatical should be generated by the grammar), it does not overgenerate (meaning that sentences judged to be ungrammatical are not generated), and it assigns 'appropriate' constituency analyses to well-formed strings (i. e. the way it groups words into constituents should produce constituents that are judged to be 'natural').

A performance grammar, on the other hand, is usually designed to generate all possible utterances (not necessarily only those judged to be grammatical, but possibly also those understandable but ungrammatical utterances that can arise because of factors such as nervousness, tiredness, boredom, and so on.). Thus, by competence grammar criteria, performance grammars often overgenerate. Arguably, there is also less concern over assignment of 'appropriate' constituency analyses in performance grammars.

Our position is that it is appropriate to use competence grammars, as these are theories of syntactic well-formedness. The performance factors identified above (nervousness, etc.) are best explained by theories of psycholinguistics.

1.2 Unification-based grammars

To be useful a grammar formalism should *be computationally tractable.* A computer should be able to process the grammar efficiently. This requirement favours explicit formalisms over implicit formalisms. Examples of implicit formalisms include stochastic grammars and grammars couched in terms of a domain theory.

A grammatical formalism should also be parsimonious. A context free grammar (CFG) does not capture important linguistic generalisations and consequently a CFG will be larger than an equivalent grammar in a more parsimonious formalism. A parsimonious formalism will help reduce the computational cost of parsing with large grammars [Ber87, p.251] [Shi86].

Unification-based formalisms meet these conditions. In a unification-based grammar each symbol in a CFG is replaced by a *category.* A category consists of *features* and the feature's *values.* A value can be either atomic or another category. An example category might be:

$$[\ cat\ V\ ]$$

This category has a single feature *cat* which has the value *V*.

A unification-based rule might be:

$$[\ cat\ S\ ] \rightarrow [\ cat\ NP\ ] \quad [\ cat\ VP\ ]$$

A CFG might express such a rule as:

$$S \rightarrow NP\ VP^{1}$$

If we now introduce a variable (shown as *?1*) into the unification-based rule we might have:

$$\begin{bmatrix} cat\ S \\ person\ ?1 \end{bmatrix} \rightarrow \begin{bmatrix} cat\ NP \\ person\ ?1 \end{bmatrix} \quad \begin{bmatrix} cat\ VP \\ person\ ?1 \end{bmatrix}$$

For this rule to be used during parsing the *person* feature must be bound with the same value in all three categories. This parsimoniously enforces subject-verb agreement. For example such a rule can be used to generate:

Toby likes Early Music

and not:

**Toby like Early Music*[2]

Unification is a form of pattern matching and is best explained with examples. The unification C of categories A and B is shown as:

$$C = A \sqcup B$$

For example,

$$\begin{bmatrix} cat\ NP \\ person\ 3 \end{bmatrix} \sqcup \begin{bmatrix} cat\ NP \\ person\ ?1 \end{bmatrix} = \begin{bmatrix} cat\ NP \\ person\ 3 \end{bmatrix}$$

An example of unification failing might be:

$$\begin{bmatrix} cat\ NP \\ person\ 3 \end{bmatrix} \not\sqcup \begin{bmatrix} cat\ NP \\ person\ 2 \end{bmatrix}$$

Unification is undefined in this last example as there is no generalisation of the feature *person* that is consistent with the values *2* and *3*. A feature can only have one binding.

[1] From now onwards we shall sometimes use a context free formalism as a notational variant for a unification-based formalism. This is intended to aid in legibility.

[2] By convention, sentences that are prefixed with asterisks are ungrammatical.

The parsimony of a unification-based formalism is shown by comparison with how we would express agreement within a context free grammar. A CFG would need three rules as opposed to the single unification-based rule.

Therefore our approach to undergeneration is to learn *unification-based* rules as opposed to simple context free grammar rules.

2 Review of grammar learning approaches

In this section we shall briefly look at four contrasting grammar learning systems. The four approaches can be classified as being either *model-driven* or *data-driven*.

A *model-driven* approach learns grammar by constructing a 'proof' of the grammaticality of a sentence from a model of grammaticality. Model-driven 'deduction' does not require exposure to data to learn grammar. The deductive closure of the model could be computed, yielding a grammar accounting for undergeneration. Therefore a model-driven approach can learn a competence grammar because the model defines grammaticality. Model-based learning relies upon the model being complete: its incompleteness can prevent the system from coping with undergeneration.

A *data-driven* approach relies upon finding regularities within the sentences as they are processed. Data-driven learning can only acquire a performance grammar, as there is no model to guide learning. Unlike model-driven learning this style of learning is not reliant upon a model.

2.1 Model-driven approaches

In Berwick's approach [Ber85], when faced with a failure to parse, the system tries to construct a rule using a set of simple actions. These actions are the model of grammaticality and inspired by Government and Binding Theory (GB) [Hae91]. GB has a set of principles that are parameterised to a particular language and these principles interact to account for sentential grammaticality. Co-ordination cannot be learnt as Berwick's model is incomplete. Rules learnt are context free.

Liu and Soo use a model that is based on the feature instantiation principles of Generalised Phrase Structure Grammar (GPSG) [LS92] [GKPS85]. GPSG is a unification-based formalism that specifies local trees, partly in terms of unification-based rules and partly in terms of feature instantiation principles. When a case of undergeneration is encountered, Liu and Soo's system prompts the user for a parse tree to be given manually. The model then helps to fill out uninstantiated features in this given parse tree. It could be argued that Liu and Soo use a model of grammaticality that is part GPSG-based and part human. The rules learnt are unification-based.

2.2 Data-driven approaches

Wolf presents an approach that is entirely data-driven [Wol87] (which Hutchinson further develops [Hut88]). Wolf's system takes a stretch of text and treats it as if it were generating a single production. For example the sentence:

Toby had a bread basket

would be generated by the rule:

$$S \rightarrow NP\ V\ Det\ N0\ N0$$

This rule would then be refined by looking for regularities:

$$S \rightarrow NP\ V\ Det\ \alpha$$
$$\alpha \rightarrow N0\ N0$$

As there is no model of grammaticality, the rules learnt are not necessarily linguistically plausible. Rules learnt are context free.

Vanlehn and Ball learn syntax by searching through a space of grammars [VB87]. The grammars generate the positive example sentences and reject the negative examples that are presented to it during learning. As soon as a new positive example is encountered, a new set of grammars is created. These grammars are extensions of the old grammars capable of parsing the new sentence. The new grammars are pruned by removing any grammar that either generates one of the negative examples or fails to generate a new positive example. In order to manage the search process, Vanlehn and Ball use a version space [Mit78]. Each node in the space is a grammar. The grammars are ordered by language inclusion. To make this ordering computable, Vanlehn and Ball restrict the form of the rules within each grammar. An example restriction is to disallow rules with an empty right hand side. This particular restriction means that long-distance dependencies cannot be generated easily. Long-distance dependencies typically involve a sentence having some constituent displaced. The place where the constituent originated will have an empty category (known as a *gap*). An example long-distance dependency is:

Who does Sam chase.

Here the the object of *chase* is missing. The usual analysis would be to introduce an NP gap at the end of the sentence. This cannot be done without rules with empty right hand sides.

The formalism of the rules learnt is context free.

2.3 Analysis

The two model-driven approaches that we discussed learn competence grammars. Both of these approaches have no mechanism for dealing with incompleteness within the model.

The data-driven approaches learn performance grammars. Performance grammars are inadequate theories of language as they generate ungrammatical sentences. Vanlehn and Ball cannot learn plausible natural language grammars given their restriction on rules learnt. Wolf's approach will also generate ungrammatical sentences because he tries to learn recursion when given only finite sequences of constructs. A finite instance of a recursive construct cannot be distinguished from a finite instance of a finite construct.

Whilst Berwick's approach is unsupervised, Liu and Soo require human intervention and is therefore supervised. Supervision prevents natural language processing being autonomous. Vanlehn and Ball's approach uses negative examples which constitutes supervision.

Liu and Soo's approach is the only one to learn a parsimonious (unification-based) formalism.

These approaches can be summarised as follows:

Approach	Competence	Unsupervised	Unification-based
Berwick	Yes	Yes	No
Vanlehn and Ball	No	No	No
Wolf	No	Yes	No
Liu and Soo	Yes	No	Yes

2.4 Criteria for successful learning

None of the approaches reviewed meets the criteria we believe are required of a system for learning grammar for the treatment of undergeneration. Our criteria are:

- A competence grammar should be learnt. In part this is a philosophical decision regarding our view of what a grammar should be. We take the view that a grammar should be a theory of grammaticality. Performance data is accounted for by a competence grammar in conjunction with theories of human language processing.
- A parsimonious formalism should be used. We explained why grammars for natural language need to be expressed succinctly.
- Linguistically plausible rules should be capable of being learnt. The learning strategy should not place restrictions upon the formal complexity of the rules.
- There should be little or no supervision. Supervision is largely a method of dealing with inadequacies in an approach and prevents natural language processing being successful.

3 Model and Data-driven grammar learning

3.1 Introduction

To deal with undergeneration we need to learn a competence grammar expressed in a parsimonious formalism. We need to learn grammar without supervision. Our approach is to use both model-driven and data-driven learning. We use sentences to guide the model-based learning when considering which rules to learn and data-driven learning both to overcome problems of model incompleteness and to provide an alternative learning scheme. This guidance is similar to the use of training examples that Mitchell et al. give in their formulation of model-based learning [MKKC86]. Our framework requires no a priori decisions 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. This freedom allows us to empirically determine the 'optimal' balance between the two extremes.

3.2 Overview

Learning begins as soon as parsing fails for some input string. The grammar is augmented with the *super rule*:

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

which will match any unmatched pair of mothers in the parse tree to be completed. The super rule intensionally expresses all binary rules capable of being expressed in our grammar, and so its usage will always lead to a derivation sequence begin completed. The next task is to reject linguistically implausible instantiations of the super rule as soon as they are proposed for use in the parse. This is currently carried out by the model, and in future will be also carried out by the data-driven learner. Surviving instantiations of the super rule are linguistically plausible and can be used to complete the parse of the input string and added to the grammar for future use. After parsing the input string, the super rule is removed from the grammar. If no instantiation is plausible, then the input string is ungrammatical. Our approach assumes a complete lexicon and possibly some sort of initial grammar.

We use the chart parser of the Grammar Development Environment (GDE) as a basis for learning. A chart parser gives us all possible substrings for a sentence and from these substrings rules are learnt[3]. Our scheme is only partially implemented so far. It augments the GDE with 2000 lines of AKCL Common Lisp and runs on a Sun 3/50 workstation.

[3]This is similar to Hall's work on learning by failing to explain [Hal86]

3.3 Model-driven

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

LP rules are restrictions upon *local trees*. A local tree is a 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. 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:

$$S \rightarrow NP\ VP$$

has a semantic rule of:

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

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

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

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

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

The result of composing **VP(NP)** is the type

$$t$$

For many newly-learnt rules, we are able to construct the corresponding semantic rules and types. These types are checked and a type clash results in the associated syntactic rule being rejected. For example, the syntactic rule:

$$VP \rightarrow VP\ VP$$

will have the ill-formed semantics of **VP(VP)**. Type checking constrains the space of syntactic rules by rejecting those syntactic rules that are semantically implausible.

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

To illustrate semantic types we shall give the types and semantics of the grammar introduced earlier:

$S \rightarrow NP\ VP$: **VP(NP)**
$VP \rightarrow V\ NP$: **V(NP)**
$NP \rightarrow Det\ N1$: **Det(N1)**
$NP \rightarrow N1$: **N1**
$N1 \rightarrow N0$: **N0**
chases : V : $\lambda\mathbf{P}[\lambda\mathbf{Q}[\mathbf{Q}(\lambda\mathbf{x}[\mathbf{P}(\lambda\mathbf{y}[\mathbf{chases(x,y)}])])]]$
Sam : NP : $\lambda\mathbf{P}[\mathbf{P(sam1)}]$
the : Det : $\lambda\mathbf{P}[\lambda\mathbf{Q}[\exists\mathbf{x}(\mathbf{P(x)} \wedge \mathbf{Q(x)})]]$
happy : A : $\lambda\mathbf{P}[\lambda\mathbf{x}[\mathbf{happy(x)} \wedge \mathbf{P(x)}]]$
cat : $N0$: $\lambda\mathbf{y}[\mathbf{cat1(y)}]$

The types of the syntactic categories are:

S : t
VP : $<<< e,t >,t >,t >$
V : $<<< e,t >,t >,<<< e,t >,t >,t >>$
NP : $<< e,t >,t >$
N1 : $< e,t >$
N0 : $< e,t >$
A : $<< e,t >,< e,t >>$
Det : $<< e,t >,<< e,t >,t >>$

From the types it follows (for example) that a syntactic rule's right hand side can contain an adjectival category and a nominal category. The left hand side of such a rule will be the result of this composition and thus nominal. Note that these types are only a partial filter upon the space of possible rules. For example the types predict that a rule such as:

$$S \rightarrow N1\ NP$$

is well-formed. This is clearly (syntactically) wrong.

Here is an example of our model in action. We shall use the grammar, types and LP rules previously mentioned to learn the rule:

$$N1 \rightarrow Adj\ N1$$

which is is (explicitly) missing from our grammar. Note that the lexical entry for the adjective is present. The trace is taken using the Grammar Development Environment's chart parser [CGBB88].

If the sentence:

Sam chases the happy cat

is parsed then undergeneration will occur. 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
```

Now suppose that the super rule [ ] → [ ] [ ] is added to the grammar. This will generate all possible parses:

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

Somewhere within this set of parses may be the correct parse(s). We need to restrict ourselves a little and reject bad rules.

In the next example we reject instantiations of the super rule that violate the LP rules:

```
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 constraints.

The effect of just using type checking, as opposed to LP rules, to reject implausible instantiations of the super rule gives the following result:

```
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 both LP rules and type checking are used 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
```

The resulting parse tree is shown in figure 1. From the tree (figure 1) the rule:

$$N1 \rightarrow Adj\ N1$$

can be extracted. Remember that in reality this rule is unification-based.

3.4 Data-driven learning

Our data-driven theory is still provisional. It has not been implemented. However we can motivate the choice of theory:

- The theory should prefer rules that are similar to rules that have been previously used.
- The theory should take into account the local syntactic context when deciding if a rule is to be accepted or not.

We are considering training our data-driven learner with a *treebank* and, from this treebank, to calculate *dominance probabilities*. A *treebank* is a set of parsed sentences [LG91]. A *dominance probability* is the probability of some category A dominating category B within the treebank. Leech describes dominance probabilities in detail [Lee87]. We shall generalise her approach from using a CFG to using a unification-based grammar. Dominance probabilities help relate the left hand side of a rule to the right hand side and help determine the plausibility of local trees (and hence rules). The size of the treebank required for successful data-driven learning will need to be determined empirically.

Other ideas that we are investigating include:

- Rule monotonicity. What happens if we lose confidence in a rule?
- Probationary rules and first-class rules. When does a newly induced rule prove its value? How do probationary rules relate to first-class rules when parsing?
- Determining the left hand side of a rule. From Gold's theorem we know that we cannot learn recursion from just a text [Gol67]. This means that we cannot determine the left hand side properly. We therefore intend to consider disjunctive features [Shi86] to express this non-determinism and let the usage of this rule refine the selection.
- When do we re-estimate dominance probabilities? If we re-estimate the probabilities too soon then we shall be biased towards rules that we have just induced. If we re-estimate the probabilities too late then we may swamp some rule that is heavily used within a small stretch of a text.

4 Discussion

We introduced undergeneration and said why it is a problem. We gave a number of solutions for this problem and found them all to be inadequate. Our solution was to learn grammar when undergeneration occurs. We then motivated unification grammars as the formalism to use. After considering four schemes for learning grammar we determined the conditions for successful grammar learning. Our theory meet these condition in a flexible way.

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) (e. g. [Leb90]). 'Undergeneration' can be called the *Incomplete domain theory problem* [MKKC86] [OM90] [FD89]. We therefore can be said to be investigating the relationship of EBL to SBL and dealing with incompleteness both within the grammar and in the model.

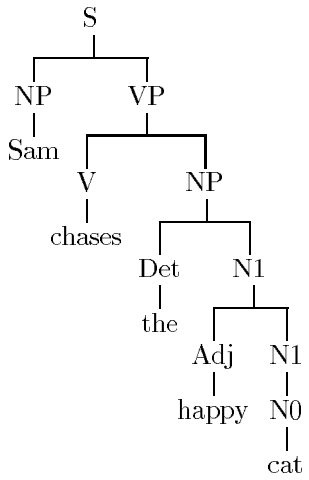

Figure 1: The parse tree after inducing the rule *N1* → *Adj N1*

There are a number of assumptions (or problems) with our approach:

- The lexicon is complete. We assume that each word encountered has a lexical entry present in the lexicon. This is arguably a strong assumption. Briscoe and Waegner suggest that contrary to Sampson the major cause of undergeneration will be lexical and not syntactic [BW92]. This suggests that we have finessed the learning problem. However we believe that we can assume a (reasonably) complete lexicon. The Core Language Engine is an example of a system that has a lexical acquisition component [Als92]. This component uses machine readable dictionaries, lexical inference, lexical correction and a battery of other techniques to deal with lexical incompleteness. We cannot comment as to whether undergeneration is lexically or syntactically determined as of yet but from the arguments about the causes of undergeneration our intuition is that the cause is syntactic.

  An allied problem is the amount of information present in the lexicon. Lexically-orientated formalisms such as Head-driven Phrase Structure Grammar encode much of their grammar within the lexical entries [PS87]. If we have a full richly specified lexicon then we will not need to learn grammar. The lexical acquisition mentioned previously does not use such a rich lexical formalism and so for the general case we will have to rely upon a relatively poor lexical formalism.
- The feature space is complete. In the same paper Briscoe and Waegner comment that their feature space did not need to be extended. If this is true then we can use their features and not have to consider learning new features.

Our discussion of future work was mainly in the section on data-driven learning. Once we have developed the data-driven scheme we shall empirically determine the plausibility of the grammars we have learnt. Grammar evaluation is difficult and there is no widely-accepted evaluation metric. We will therefore have to consider how we can evaluate grammars. We shall also consider enriching the model-based learning mechanism, possibly by using principles from GB.

Obvious applications of our work include:

- Corpus parsing.
- Machine translation.
- Text summarising.
- Any other system that attempts to process extensive quantities of natural language.

Acknowledgements

We should like to thank Sara Climpson for reading drafts of this paper, John Carroll for technical help with the chart parser, and Andy Fisher for helping typographically. The first author is supported by a Science and Engineering Research Council grant.

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.

[Ber87] Robert C. Berwick. *Computational Complexity and Natural Language*. MIT Press, 1987.

[Bri87] Ted Briscoe. Noun Phrases are Regular: a Reply to Professor Sampson. In

W. Meijs, editor, *Corpus Linguistics and Beyond*. Rodopi, 1987.

[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.

[GBCB92] Claire Grover, Ted Briscoe, John Carroll, and Bran Boguraev. *The Alvey Natural Language Tools Grammar (Third Release)*. Technical report, University of Cambridge Computer Laboratory, 1992.

[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.

[Gra83] Richard H. Granger. The NOMAD System: Expectation-Based Detection and Correction of Errors during Understanding of Syntactically and Semantically Ill-formed Text. *Computational Linguistics*, 9:188–196, 1983.

[Hae91] Liliane Haegerman. *Introduction to Government and Binding Theory*. Basil Blackwell, 1991.

[Hal86] R. J. Hall. Learning by failing to explain. In *Proceedings of the 5^{th} International Conference on Artificial Intelligence*, pages 568–572, 1986.

[HRS84] K Hammond and V.J. Rayward-Smith. A Survey on Syntactic Error Recovery and Repair. *Computer Language*, 9:51–67, 1984.

[Hut88] Alan Hutchinson. Building Grammars from Natural Text. In *Proceedings of the 3^{rd} European Working Session on Learning, Turing Institute, Glasgow*, pages 45–52, 1988.

[Leb90] Michael Lebowitz. The Utility of Similarity-Based Learning in a Wprld 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.

[LS92] Rey-Long Liu and Von-Wun Soo. Augmenting and Efficiently Utilizing Domain Theory in Explanation-Based Natural Language Aquisition. In *Proceedings of the 9^{th} International Conference on Machine Learning, Aberdeen University, Scotland*, pages 282–289, 1992.

[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.

[Lyo68] John Lyons. *Introduction to Theoretical Linguistics*. Cambridge University Press, 1968.

[Mit78] T. M. Mitchell. *Version Spaces: An Approach to Concept Learning*. PhD thesis, Stanford University, 1978.

[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.

[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.

[Osb92] Miles Osborne. *Text Parsing and the treatment of undergeneration.* First Year DPhil Report, 1992. University of York.

[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.

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

[Wei83] Ralph M. Weischedel. Meta-rules as a Basis for Processing Ill-formed Input. *Computational Linguistics*, 9:161–177, 1983.

[Wol87] J. G. Wolff. Cognitive Development as Optimisation. In Leonard Bolc, editor, *Computational Models of Learning*, pages 161–205. Springer-Verlag, 1987.