# Grammar Specialisation Meets Language Modelling

**Manny Rayner**
**Netdecisions Technology Centre**
**Westbrook Centre, Milton Road**
**Cambridge CB4 1YG**
**United Kingdom**
**mannyrayner@yahoo.com**

**Beth Ann Hockey, John Dowding**
**RIACS, Mail Stop 19–39**
**NASA Ames Research Center**
**Moffett Field, CA 94035-1000**
**USA**
**{bahockey, jdowding}@riacs.edu**

## 1 Introduction

Constructing language models that appropriately constrain recognition is a key task in building spoken language applications. There are two ways to do this: either to induce a statistical language model from corpus data, or to hand-code the language model explicitly as a grammar. Although the academic community has paid more attention to statistical language models (Rosenfeld and Huang, 1992; Ward and Issar, 1995), there are many circumstances in which a grammar based model offers clear advantages:

- Expert users who are familiar with the intended coverage of the system tend to get better recognition performance from grammar based models than from statistical models (Knight et al., 2001).
- There may be no training data, or insufficient training data, to train a domain-specific statistical language model, and the performance of domain-independent statistical language models is inadequate for most spoken dialogue applications.
- Vocabulary can change unpredictably in ways that require corresponding adjustment of the LM, for example if the user needs to refer to a member of a set of objects which is only known at run-time. Again, this makes it difficult to construct a high-quality domain-specific statistical language model.

For both commercial and research systems in novel domains, the above is the rule rather than the exception. Consequently, commercial spoken language implementation platforms like Nuance (Nuance, 2002) and SpeechWorks (SpeechWorks, 2002) that cater to commercial developers have over the last few years focussed almost exclusively on hand-coded grammar-based language models, typically realised as annotated CFGs written in formalisms formalisms like Nuance's Grammar Specification Language (GSL).

Increasing interest in mixed-initiative systems, which require more elaborate grammars, has motivated a corresponding interest in development of tools that allow specification of CFG grammars using more expressive linguistic formalisms such as unification grammar (Moore, 1998; Kiefer and Krieger, 2000; Dowding et al., 2001; Rayner et al., 2001; Bos, 2002). From a software engineering point of view, there are good reasons to want to use a high-level formalism: grammars can be written in a compact and modular form, making it easier to maintain and reuse them. However, there is a clear downside. Very general grammars are good from the point of view of reusability, but by their nature will not be closely tuned to a given domain. Also, the increase in expressive power is as usual accompanied by a decrease in tractability; small changes in the source unification grammar can have large effects on compilation times and run-time performance (Rayner et al., 2000d).

In this paper, we will show how we can attack the problems we have just named using the tool of grammar specialisation via explanation-based learning (Rayner, 1988; Samuelsson, 1994; Rayner and Carter, 1996). We start with a general unification grammar, and a domain corpus. We use the corpus

to create a *specialised* version of the unification grammar; we do this by chunking together rules in ways suggested by the corpus examples so as to form macro-rules. The specialised grammar has a coverage that is strictly less than that of the original grammar, but with a suitable corpus the practical impact on coverage need not be large. The structure of the grammar is sufficiently simplified by the specialisation process that compilation into a language model and the run-time behaviour of that language model both become tractable. The grammar is in effect shoe-horned into a standard shape with known properties, making it possible to tune the language model systematically while still keeping a perspicuous representation of it.

Experiments were carried out using the Gemini platform (Dowding et al., 1993) and a substantial unification grammar developed for a simulated robotics domain. Section 2 describes the experimental framework in more detail. Section 3 briefly presents background on grammar specialisation, and Section 4 describes how it was applied to the task of generating CFG language models. Section 5 describes experiments in which a domain corpus was used to create CFG language models derived from several different specialised versions of the original grammar. Section 6 concludes.

## 2 Basic framework

The experiments described here have been carried out in the context of a spoken language interface to a simulated version of the Personal Satellite Assistant (PSA; (PSA, 2002)). The real PSA is a robot currently being developed at NASA Ames Research Center, which is intended for deployment on the Space Shuttle and/or International Space Station. The PSA spoken dialogue interface demo supports interaction with a simple simulation of the Shuttle and of the robot's movement and sensor functions.

Speech recognition is performed using a version of the Nuance recognizer (Nuance, 2002). Initial language processing is carried out by the SRI Gemini system (Dowding et al., 1993). The language processing grammar is compiled into a recognition grammar using the methods of (Moore, 1998); the net result is that only grammatically well-formed utterances can be recognized. Dialogue management and other downstream processing is described in (Rayner et al., 2000c). The dialogue model has two states: the "main" state accepts a wide range of user utterances, and the "confirmation" state accepts only confirmation/disconfirmation utterances ("yes"/"no"/"affirmative" etc).

| |
|---|
| Start scenario three |
| What is the temperature? |
| How about carbon dioxide? |
| Measure the pressure at flight deck |
| Go to the crew hatch and close it |
| Close all three doors |
| What were temperature and pressure at fifteen oh five? |
| Is the temperature going up? |
| Is the door at crew hatch open? |
| Do the fixed sensors say that the pressure is decreasing? |
| Was the fan at mid deck switched on three minutes ago? |

Figure 1: Examples of utterances covered by the PSA application grammar

The language model for the "main" dialogue state was compiled from a general unification grammar essentially consisting of a scaled-down and adapted version of the Core Language Engine grammar for English (Pulman, 1992), combined with a lexicon specific to the PSA domain comprising 334 uninflected entries. This grammar is described in detail in (Rayner et al., 2000d). The examples in Table 1 illustrate current coverage.

The training corpus used in the experiments we describe below is the systems's "recognition log". Every spoken utterance input to the system since the start of the project has been stored, which has to date given us a corpus consisting of 20944 logged utterances (71934 words) of data. Specifically, what is stored is the text output by the recognizer in

those cases where anything was actually produced. The obvious drawback is that not all the logged speech data is correct, given that recognition is less than perfectly accurate. On the other hand, we were interested to see what could be achieved without incurring any sizable overheads related to corpus collection; the results suggest that this data is in fact quite useful. Of the 20944 logged utterances, we actually only made use of the 14271 utterances produced in the "main" dialogue state that were inside the coverage of the grammar, discarding the 6436 confirmation-state utterances and the 237 out-of-grammar utterances.

In addition to the low-grade untranscribed training data described in the last paragraph, we also recorded and transcribed a further 6264 utterances (29008 words) of data for use in development and testing. This data was collected from 24 subjects not previously involved in the development of the system; each subject was first given a uniform short introductory session, and then asked to solve a number of tasks which involved using the speech interface to the simulated robot. We discarded the 2087 confirmation-state utterances and the 576 out-of-grammar utterances, leaving 3601 main-state in-coverage utterances totalling 20985 words.

## 3 Grammar Specialization Using Explanation Based Learning

The idea of using Explanation Based Learning (Mitchell et al., 1986) to specialise a unification grammar was originally suggested in (Rayner, 1988) and has since been explored in a number of papers (Samuelsson and Rayner, 1991; Samuelsson, 1994; Neumann, 1994; Srinivas and Joshi, 1995; Rayner and Carter, 1996); it is most easily conceptualized as a kind of chunking or macro-rule learning method (Fikes and Nilson, 1971). We start with a unification grammar $G$ and a parsed corpus of correct derivation trees $D_i$ for a set of utterances within the coverage of $G$. Each derivation tree $D_i$ is decomposed into one or more subtrees $D_{ij}$; each $D_{ij}$ is then converted into a "chunked" grammar rule $R_{ij}$, by recursively unifying together every daughter of every rule in $D_{ij}$ with the mother of the rule immediately below it. The mother of $R_{ij}$ will then be the mother of the rule at the root of $D_{ij}$, and its daughters will be the concatenation of the daughters of the rules at the fringe of $D_{ij}$. The definition implies that all the constraints present in the derivation tree $D_{ij}$ are included in the corresponding rule $R_{ij}$[1]. The intent is to replace $G$ with a new grammar $G'$ consisting of the union of all the $R_{ij}$. By construction, $G'$ has strictly less coverage than $G$, but is more closely tuned to the corpus.

All of the above is very abstract: we illustrate with a concrete example. Suppose $G$ is the toy unification grammar

```
SIGMA:[] --> S:[]
S:[] -->
   NP:[num=N, pers=P], VP:[num=N, pers=P]
VP:[num=N, pers=P] -->
   V:[type=trans, num=N, pers=P], NP:[]
NP:[num=sing, pers=3] -->
   NAME:[]
NP:[num=N, pers=3] -->
   DET:[num=N], NBAR:[num=N]
NBAR:[num=N] -->
   ADJ:[], NBAR:[num=N]

NAME:[] --> john
V:[type=trans, num=sing, pers=3] --> has
DET:[num=sing] --> a
ADJ:[] --> red
NBAR:[num=sing] --> car
```

and our training example $D_1$ is the single possible derivation tree for the sentence "John has a red car". We can combine together all the rules in $D_1$ to yield the single chunked rule

```
SIGMA:[] -->
   NAME:[],
   V:[type=trans, num=N, pers=3],
   DET:[num=N],
   ADJ:[],
   NBAR:[num=N]
```

Alternately, we can extract two rules from $D_1$ by first chunking together the SIGMA-level rules to make the macro-rule

```
SIGMA:[] -->
```

[1] This result is stated in a more precise form and formally proved in (Rayner et al., 2000b).

```
NP:[num=N, pers=P],
V:[type=trans, num=N, pers=P],
NP:[]
```

and then combining the NP-level rules to make the rule

```
NP:[num=N, pers=3] -->
   DET:[num=N], ADJ:[], NBAR:[num=N]
```

This shows a typical strategy when performing EBL-based grammar specialisation: we "flatten" the grammar so that only a small number of non-pre-terminal categories are left. In the first example, there are no non-pre-terminal categories left except SIGMA; in the second one, the non-pre-terminals in the specialised grammar are SIGMA and NP. A set of meta-rules is thus needed to determine which chunks of derivation are turned into macro-rules. We will call meta-rules of this type "cutting-up criteria"[2].

Experiments from the papers quoted above demonstrate that a specialised grammar can often in practice be much better that an unspecialised one for parsing tasks; the smaller search space means that parsing times and ambiguity are decreased, compensating for the loss of coverage. There are however some important caveats. In particular, since a specialised grammar has a very different structure compared to a normal grammar, it is by no means guaranteed that performance will improve if it is used in conjunction with a standard parsing strategy.

For example (Samuelsson, 1994) found that a specialised version of the SRI Core Language Engine grammar actually parsed more slowly than the unspecialised one, if the CLE's left-corner parser was used; however, a specially designed LR parser produced dramatically improved parsing times, outperforming the CLE parser by more than an order of magnitude. In the next section, we will see that similar considerations apply to the task of compiling grammars into language models.

[2]Another common name is "operationality criteria".

## 4 Applying grammar specialisation to language model compilation

This section describes how we have applied grammar specialisation to the task of building language models from unification grammars. We focus on three specific issues: defining cutting-up criteria, filtering training material, and post-processing of generated specialised grammars for purposes of language model compilation.

### 4.1 Cutting-up criteria

In order to investigate the importance of specific choice of cutting-up criteria we have used two cutting-up criteria in our experiments. The first ("2L"), follows the strategy from (Samuelsson and Rayner, 1991), and generates a two-level grammar in which the only non-pre-terminals are SIGMA and NP. This is illustrated in the second example in Section 3. The second ("3L") generates a three-level grammar in which the possible non-terminals are SIGMA, PP and NP. SIGMAs may dominate PPs, NPs and words; PPs may dominate NPs and words; and NPs may dominate PPs and words. Thus for example the training example "Are the fans at crew hatch and flight deck on[3]?" will give us four chunked rules, schematically of the forms

```
SIGMA --> V NP ADJ
NP --> DET N PP
PP --> P NP
NP --> N CONJ N
```

### 4.2 Filtering training material

Since the training data we are using is very noisy (cf. Section 2), we expected that it would be desirable to filter it in some way. During the EBL training phase, we parse all the corpus utterances and in each case extract the most preferred derivation; we then decompose each derivation according to the cutting-up criteria, and from each subderivation save the chunked EBL rule together with a) the fringe of the tree used to create it, b) the full utterance. Each chunked rule is thus tagged

[3]"Crew hatch" and "flight deck" are both lexical items in the grammar.

with the set of corpus examples that could have been used to create it.

At the end of the training run, an expert judge familiar with the original grammar manually filters the set of chunked rules, using a tool which displays the rules with examples. Rules derived from bad training examples, which are typically produced by incorrect speech recognition, are eliminated.

Data saved by the rule-filtering tool is stored in a form that allows judgements to be reused across runs. For the application investigated here, the initial EBL training run typically results in about 150 to 750 derived rules, depending on choice of cutting-up criteria. Manual filtering can then be carried out at a rate of about 5 to 10 rules per minute.

### 4.3 Post-processing specialised grammars for LM compilation

Our initial plan was simply to apply the Gemini UG-to-CFG compiler (Moore, 1998) to the specialised grammars produced by EBL learning. A little experimentation however revealed that raw specialised grammars caused the compiler severe problems. Compilation times were very high, and for large specialised grammars they exceeded reasonable resource limits. We consequently investigated the factors which were responsible for this behaviour.

Similarly to the results in (Samuelsson, 1994), it turned out that the problem was essentially that the compiler had not been optimised for specialised grammars, which typically have a flat structure with many long rules. Since the UG-to-CFG compiler essentially works by non-deterministically expanding out unification grammar rules to all their possible instantiations, long rules can for obvious reasons result in combinatoric explosion. It was however easy to solve the problem: we eliminated the long rules by transforming the grammar into a binarised form, so that no production has more than two daughters. The experiments in the next section contrast compilation behaviour with and without the binarisation transform.

## 5 Experiments

This section reports a series of experiments in which we investigated the idea of using grammar specialisation as an aid to compiling language models from unification grammars. We were interested in two main questions. Most obviously, we wanted to know whether grammar specialisation could be used to improve the quality of a language model. Thus our first series of experiments (Section 5.1) takes a number of language models derived from specialised grammars, and contrasts their coverage and performance against the language model derived from the initial unspecialised grammar.

The second, and arguably more important question is whether grammar specialisation can be used to make the process of deriving a language model from a general unification grammar more *scalable*. The initial unification grammar's structure is complex and idiosyncratic, and small changes can have large effects on both compilation times and runtime performance. The specialised grammar's structure is in contrast very uniform, and its performance can be scaled in a straightforward way by choosing how many rules to retain or discard. For example, an obvious strategy is to retain only rules derived from patterns that occur in $N$ or more training examples for some theshold value $N$. Discarding rules derived from low-frequency examples loses coverage, but once again tightens the language model. Our second series of experiments (Section 5.2) consequently examines the performance of language models derived from different-sized subsets of both the original and the specialised grammars.

In all these experiments, we trained specialised grammars on the 14271-utterance training set described in Section 2, and where appropriate tested performance on the unseen wave-file test data described in the same section, using the Nuance Toolkit `batchrec` tool. We were interested in the following parameters: the time taken to compile the language model using the Gemini UG-to-CFG compiler (**CmpT**), the coverage of the grammar, de-

| Version | CmpT (secs) | Cov (%) | WER (%) | SER (%) | ×RT |
|---|---|---|---|---|---|
| Unspec. | 4418 | 100 | 12.79 | 30.41 | .720 |
| 2L, F– | 2051 | 99.75 | 11.48 | 32.19 | .300 |
| 2L, F+ | 536 | 98.28 | 11.43 | 32.24 | .244 |
| 3L, F– | 234 | 99.44 | 12.13 | 29.29 | .287 |
| 3L, F+ | 148 | 99.44 | 11.17 | 29.07 | .229 |

Table 1: Performance of language models and recognisers derived from original unspecialised grammar and four specialised grammars

| Version | CmpT (secs) | Cov (%) | WER (%) | SER (%) | ×RT |
|---|---|---|---|---|---|
| 25% | 18 | 75.06 | 32.37 | 41.93 | .176 |
| 50% | 37 | 98.22 | 11.29 | 28.99 | .209 |
| 75% | 83 | 99.25 | 11.06 | 28.94 | .212 |
| 100% | 148 | 99.44 | 11.17 | 29.07 | .229 |

Table 2: Performance of language models and recognisers derived from 3-level specialised grammars of four different sizes

| Version | CmpT (secs) | Cov (%) | WER (%) | SER (%) | ×RT |
|---|---|---|---|---|---|
| 25% | 11 | 48.60 | 60.07 | 62.40 | .158 |
| 50% | 36 | 82.84 | 25.11 | 39.24 | .234 |
| 75% | 369 | 99.94 | 11.70 | 30.10 | .481 |
| 100% | 4418 | 100 | 12.79 | 30.41 | .720 |

Table 3: Performance of language models and recognisers derived from unspecialised grammars of four different sizes

fined as the proportion of utterances in the test set that were within grammar coverage (**Cov**), the accuracy of the recognition package derived from the language model in terms of Word Error Rate (**WER**) and Sentence Error Rate (**SER**), and the average recognition speed expressed as a multiple of real-time (×**RT**). Tests were run on a 360MHz SUN UltraSparc60 with 1.5GB of RAM, using Version 3.8.5 of SICStus Prolog and Version 7.0.2 of Nuance.

### 5.1 Can specialisation improve the language model?

We created specialised grammars using the cutting-up criteria **2L** and **3L** defined in Section 4.1. In order to investigate the extent to which rule filtering (cf. Section 4.2) affected the quality of the language model, we produced two versions of each language model, differing with respect as to whether the rules were filtered (**F+**) or unfiltered (**F−**). Table 1 summarises the results. For comparison, the first line shows the corresponding values for the original unspecialised grammar.

### 5.2 Are specialised grammars scalable?

We investigated the scalability of the 3-level filtered version of the specialised grammar ("3L, F+") by constructing language models from four versions of the grammar containing different numbers of rules. The full grammar has 80 rules. These rules were first ordered by the number of times the example they were derived from occurred in the training corpus; the four grammars were then built, respectively, from the first 25% of the rules, the first 50%, the first 75% and the whole set. The results are presented in Table 2.

In order to construct a corresponding test of scalability for the original 59-rule unspecialised grammar, we ordered its rules by their frequency of occurrence in the parsed training corpus. We then constructed three proper subsets of the grammar, which respectively consisted of the most frequent 25%, 50% and 75% of the rules. We derived language models from these subsets and from the full grammar, and evaluated them similarly. The results are in Table 3.

Finally, we tested the effect of the binarisation transform on the UG-to-CFG compilation process by compiling six sample grammars (two unspecialised and four specialised) with the switch controlling binarisation set in turn on (**BIN+**) and off (**BIN−**). Table 4 presents the results. It is apparent that binarisation is substantially irrelevant to unspecialised grammars, but crucial when specialised grammars are used.

| Version | BIN+ | BIN– |
|---|---|---|
| Unspec, 50% | 36 | 37 |
| Unspec, 100% | 4418 | 4537 |
| 2L, F+, 50% | 174 | - |
| 2L, F+, 100% | 516 | - |
| 3L, F+, 50% | 37 | 48 |
| 3L, F+, 100% | 148 | 2924 |

Table 4: Compilation times in seconds for two unspecialised and four specialised grammars, with binarisation respectively on and off. Compilation of both "2L" grammars exceeded resource bounds with binarisation off.

## 6 Conclusions and further directions

Our original goal was to use grammar specialisation to tune the language model by associating it more closely with the domain. Comparing the best specialised grammar (the "75%" grammar from Table 2) and the best unspecialised grammar (the "75%" grammar from Table 3), we see that the recogniser produced from the specialised grammar is more than twice as fast as the one produced from the unspecialised grammar (0.212×RT versus 0.481×RT). Specialisation has lost 0.7% of coverage (99.25% versus 99.94%), but the tighter model means that the specialised recogniser actually has slightly *better* WER (11.06% versus 11.70%) and SER (28.94% versus 30.10%). Only a couple of hours of human expert time were needed to do the rule-filtering described in Section 4.2. We regard this as a clear success.

To our thinking, however, the really exciting data are the scalability results from Section 5.2. Looking at Table 2 and Table 3, we see utterly different behaviours. The unspecialised grammars in Table 3 becomes increasingly intractable as more rules are added, with compilation times going up approximately exponentially and processing times approximately linearly. In contrast, the relationship between compilation times and number of rules for the specialised 3-level grammar of Table 2 is approximately quadratic, and the relationship between processing times and numbers of rules is clearly sub-linear. As far as generation of CFG language models is concerned, the bottom line is that the unspecialised grammar has already reached its maximum size. Specialised grammars, on the other hand, can almost certainly be made much larger than the ones shown here and still be practically useful.

The next step seems clear. The existing unspecialised grammar is, as noted, essentially an adapted and scaled-down subset of the SRI Core Language Engine grammar for English; the reason why it is only a subset is that anything larger fails to compile. Given the above results, it however seems quite feasible to expand it so that it becomes a fully general broad-coverage grammar. A grammar of this kind would certainly not compile into a useful CFG language model; it would, however, support parsing of text corpora, and hence EBL-based generation of specialised grammars. The indications are that these specialised grammars could then be compiled into language models. Putting the pieces together, the result would be a practical method for rapid derivation of CFG language models from small example corpora. We are in the process of implementing this plan, and hope to be able to report soon as to whether it yields the desired results.

## References


H. Alshawi, editor. 1992. *The Core Language Engine.* MIT Press, Cambridge, Massachusetts.

J. Bos, 2002. *UNIANCE: A compiler that translates unification grammars into GSL.* http://www.iccs.informatics.ed.ac.uk/ jbos/-systems.html. As of 28 February 2002.

J. Dowding, M. Gawron, D. Appelt, L. Cherny, R. Moore, and D. Moran. 1993. Gemini: A natural language system for spoken language understanding. In *Proceedings of the Thirty-First Annual Meeting of the Association for Computational Linguistics.*

J. Dowding, B.A. Hockey, J.M. Gawron, and C. Culy. 2001. Practical issues in compiling typed unification grammars for speech recognition. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*, Toulouse, France.

R.E. Fikes and N.J. Nilson. 1971. Strips: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 3:251–288.

B. Kiefer and H. Krieger. 2000. A context-free approximation of head-driven phrase structure grammar. In *Proceedings of the 6th International Workshop on Parsing Technologies*, pages 135–146.

S. Knight, G. Gorrell, M. Rayner, D. Milward, R. Koeling, and I. Lewin. 2001. Comparing grammar-based and robust approaches to speech understanding: a case study. In *Proceedings of Eurospeech 2001*, pages 1779–1782, Aalborg, Denmark.

T.S. Mitchell, S. Kedar-Cabelli, and R. Keller. 1986. Explanation-based generalization: a unifying view. *Machine Learning*, 1(1):47–80.

R. Moore. 1998. Using natural language knowledge sources in speech recognition. In *Proceedings of the NATO Advanced Studies Institute.*

G. Neumann. 1994. Application of explanation-based learning for efficient processing of constraint-based grammars. In *Proceedings of the Tenth IEEE Conference on Artifical Intelligence for Applications*, pages 208–215, San Antonio, TX.

Nuance, 2002. http://www.nuance.com. As of 1 Feb 2002.

PSA, 2002. *Personal Satellite Assistant (PSA) Project.* http://ic.arc.nasa.gov/ic/projects/psa/. As of 1 Feb 2002.

S.G. Pulman. 1992. Syntactic and semantic processing. In Alshawi (Alshawi, 1992), pages 129–148.

M. Rayner and D.M. Carter. 1996. Fast parsing using pruning and grammar specialization. In *Proceedings of the Thirty-Fourth Annual Meeting of the Association for Computational Linguistics*, pages 223–230, Santa Cruz, California.

M. Rayner, D. Carter, P. Bouillon, V. Digalakis, and M. Wirén, editors. 2000a. *The Spoken Language Translator.* Cambridge University Press.

M. Rayner, D. Carter, and C. Samuelsson. 2000b. Grammar specialization. In Rayner et al. (Rayner et al., 2000a).

M. Rayner, B.A. Hockey, and F. James. 2000c. A compact architecture for dialogue management based on scripts and meta-outputs. In *Proceedings of ANLP 2000.*

M. Rayner, B.A. Hockey, and F. James. 2000d. Compiling language models from a linguistically motivated unification grammar. In *Proceedings of the Eighteenth International Conference on Computational Linguistics.*

M. Rayner, J. Dowding, and B.A. Hockey. 2001. A baseline method for compiling typed unification grammars into context free language models. In *Proceedings of Eurospeech 2001*, pages 729–732, Aalborg, Denmark.

M. Rayner. 1988. Applying explanation-based generalization to natural-language processing. In *Proceedings of the International Conference on Fifth Generation Computer Systems*, pages 1267–1274, Kyoto, Japan.

R. Rosenfeld and X. Huang. 1992. Improvements in stochastic language modeling. In *Speech and Natural Language Workshop*, pages 107–111.

C. Samuelsson and M. Rayner. 1991. Quantitative evaluation of explanation-based learning as an optimization tool for a large-scale natural language system. In *Proceedings of the Twelfth IJCAI*, pages 609–615, Sydney, Australia.

C. Samuelsson. 1994. *Fast Natural-Language Parsing Using Explanation-Based Learning.* Ph.D. thesis, The Royal Institute of Technology and Stockholm University.

SpeechWorks, 2002. http://www.speechworks.com. As of 1 Feb 2002.

B. Srinivas and A. Joshi. 1995. Some novel applications of explanation-based learning to parsing lexicalized tree-adjoining grammars. In *Proceedings of the 33th Annual Meeting of the Association for Computational Linguistics.*

W. Ward and S. Issar. 1995. The cmu atis system. In *Spoken Language System Technology Workshop*, pages 249–251.