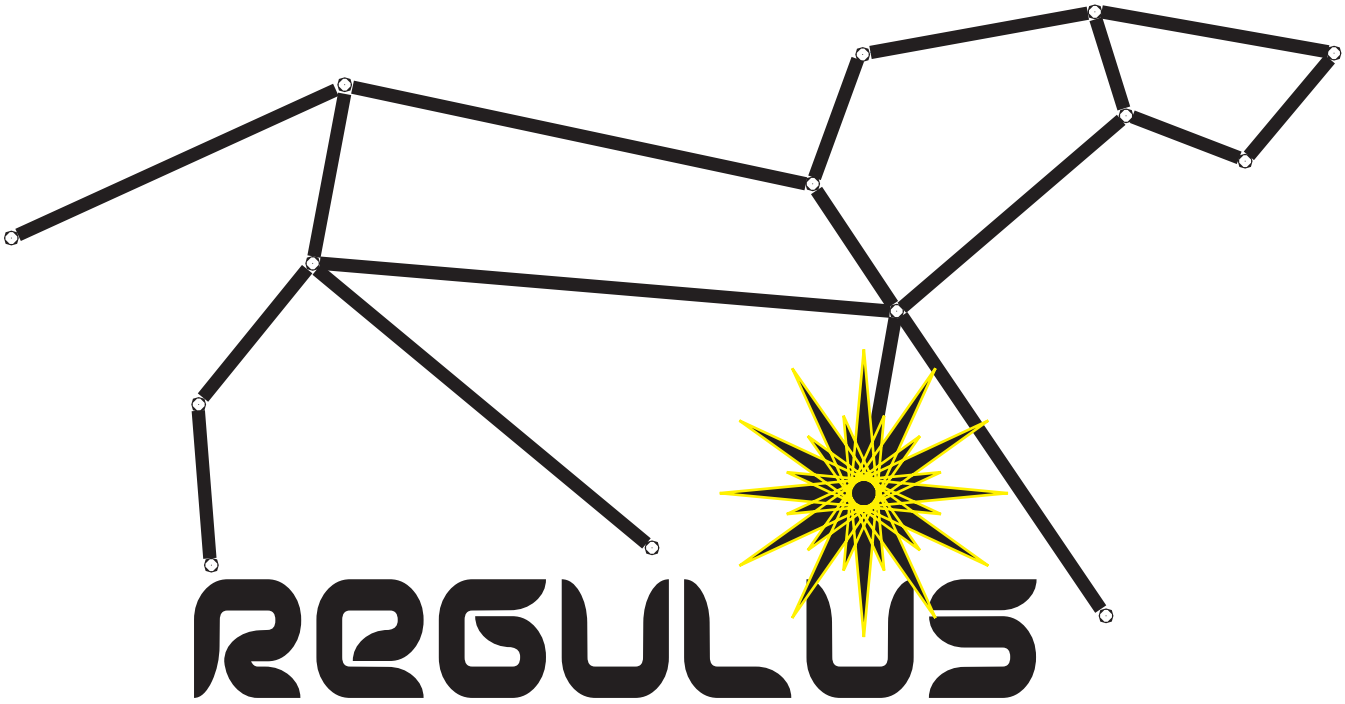

Technical Report 2003-001

Example Based Derivation of Efficient Domain-Specific Speech Recognisers from a General Linguistically Motivated Unification Grammar

Manny Rayner, Beth Ann Hockey
and John Dowding

Example Based Derivation of Efficient Domain-Specific Speech Recognisers from a General Linguistically Motivated Unification Grammar

Manny Rayner, Beth Ann Hockey and John Dowding
Mail Stop T-27A
NASA Ames Research Center
Moffett Field, CA 94035-1000
{mrayner,bahockey,jdowding}@riacs.edu

Abstract

We describe an approach to portable grammar-based language modelling in which all models are derived from a single linguistically motivated unification grammar. Domain-specific CFG language models are produced by first specialising the grammar using an automatic corpus-based method, and then compiling the resulting specialised grammars into CFG form.

We present results showing that recognisers for multiple, fairly different, domains can be derived from a single general grammar. The process remains tractable as the size of the general grammar increases, and also scales well with the size of the training corpus used. We also show that a simple method for compiling unification grammars into CFG form can be successfully applied to grammars containing large numbers of features, if it is enhanced by suitable interleaving of the expansion and filtering stages.

1 Introduction

Grammar based language models for constraining speech recognition are particularly attractive as an alternative to statistical models in domains that lack extensive speech corpora. For commercial dialogue systems, the case in which there is not enough speech data to train effective statistical models is the norm. This lack of data also impacts research domains that are relatively novel, such as dialogue interfaces to robots. Given the difficulties involved in using statistical modeling with limited speech data, we think it is important to investigate ways in which grammar based models can be efficiently and effectively produced.

Commercial speech platform vendors like Nuance (Nuance, 2003) and SpeechWorks (SpeechWorks, 2003) have primarily focussed on grammar-based language models, typically implemented in some variant of context free grammar (CFG). However, even moderate sized CFGs are tedious to write and difficult to maintain, compared to grammars written in higher level formalisms such as unification based grammars. For each rule in the higher level grammar there are likely to be many rules, very similar to each other, in a comparable CFG. The higher-level formalism provides a more compact representation and expresses linguistic dependencies and relations more transparently and explicitly than a corresponding CFG.

Compiling the CFG language model from a grammar written in the higher level formalism is one step toward producing a CFG language model efficiently while taking advantage of the attractive properties of the higher-level formalism. Techniques for compilation from unification grammars to CFGs have been discussed in (Moore, 1998; Chappelier et al., 1999; Kiefer and Krieger, 2000; Dowding et al., 2001; Rayner et al., 2001a; Bos, 2002).

A further step in making the production of grammar based language models efficient is to avoid the overhead of creating a new unification based gram-

mars from scratch for each new domain. Grammars for different domains can be sensibly viewed as subsets of the larger grammar of a language. From the point of view of portability, having a single general unification based grammar for a language and deriving domain-specific grammars from it is clearly preferable to building individual grammars for each domain by hand. The obvious advantage is that grammar rules are reusable and need only be written once. Consequently, effort that would have gone into developing the grammar for each domain can instead be put into refining the general grammar, thereby benefiting all subsequent domains.

In summary, the overall approach we are advocating is to develop a general domain-independent unification grammar for each language, semi-automatically derive specialised versions of this grammar for each new domain, and compile these specialised grammars down first into CFG language models and then into concrete recognisers. It is clear that this is an ambitious programme, and that there are many potential problems.

In the current paper, we describe a series of experiments, carried out using the XXX system[1], that provide surprisingly positive answers to several of the obvious questions. Our main claims are as follows:

- It is possible to derive multiple domain-specific recognisers from a single linguistically motivated unification grammar, using example-based methods driven by small corpora.
- These recognisers are competitive with ones built from hand-coded grammars.
- The methods have good scalability properties, both with respect to the number of domain-specific training examples and with respect to the size of the general grammar.

2 The XXX system

This section presents an overview of XXX. XXX is a suite of software modules, which extends the Open Source REGULUS system (Rayner et al., 2001a). It consists of the following main components:

1. An environment for developing and debugging typed unification grammars, whose main component is a compiler that converts grammars of this form into left-corner parsers, using a version of the algorithm described in (Moore, 2000).
2. A general linguistically motivated unification grammar for a substantial fragment of English, including an accompanying core lexicon.
3. Tools to support corpus-based specialisation of the general grammar.
4. A compiler that translates typed unification grammars into CFG language models expressed in Nuance Grammar Specification Language (GSL) formalism. These language models can then be compiled into Nuance recognition packages using the Nuance Toolkit `nuance-compile` utility.

All these components are implemented on top of Sicstus Prolog and the Nuance Toolkit. In the rest of the section, we will describe the general grammar, the specialisation tools and the unification grammar to CFG compiler.

2.1 The general grammar

The general grammar currently contains 145 phrase-structure rules and 72 features, and is a greatly expanded version of the grammar described in (Rayner et al., 2000c), which in turn is loosely based on the Core Language Engine grammar (Pulman, 1992). It is built according to reasonably standard linguistic principles and covers a large proportion of the basic constructions of English, including the following: declarative clauses; Y-N questions; WH-questions with movement of NPs, PPs, ADJPs and ADVPs; embedded Y-N and WH-questions; imperatives; elliptical NPs, PPs, ADVPs and sequences of these constituents; impersonal subjects; passives; verbal and sentential adverbs; a wide variety of subcategorisation types of verb, including intransitives, transitives, ditransitives, verbs taking PPs, verbs taking particles, equative and predicative “be”, auxiliaries, modals, verbs taking “-ing” complements, infinitives and “to” VP complements, verbs taking propositional and embedded question complements

[1]The name of the system has been suppressed in the interests of preserving anonymity.

Personal Satellite Assistant (PSA)
“affirmative”
“go to flight deck”
“mid deck and lower deck”
“measure pressure”
“what were oxygen and c o two one minute ago”
“when did the temperature reach twenty degrees”
“go to the crew hatch and close it”
“close all three doors”
Home Automation (HA)
“is there a tv in the living room”
“which devices are switched on”
“turn on the kitchen light and the stove”
“dim the light to fifty percent”
“thank you”
Travel Deals (TD)
“holidays in paris under two hundred pounds”
“i want something leaving from stansted”
“in spain during may or june from gatwick”
“is there anything in italy before may tenth”
“give me a winter brochure”
“do you have three star or four star”
Medical Speech Translator (MST)
“do you often have headaches in the morning”
“is the pain in the front of your chest”
“does the pain spread to the left arm”
“have you had chest pains for more than a week”
“are the headaches relieved by stress removal”
“how severe are the symptoms”
“is the frequency of the attacks increasing”
Intelligent Procedure Assistant (IPA)
“next step”
“go back”
“go to step three point two”
“no i said go to step five”
“set alarm for twelve minutes from now”
“record a voice note on step seven”
“delete voice note on step four point one”
“increase volume”
“say that again”
Mobile Agents (MA)
“take a picture of me”
“boudreaux follow me now”
“return to the hab”
“start tracking my physiological sensors”

Table 1: Example utterances for domains currently covered by the general grammar

and verbs taking adjectives; postmodification of NPs and VPs by PPs, ADJPs, ADVPs, relative and reduced relative clauses, numbers and names; use of NPs as temporal adverbials; postpositions; date and time expressions; conjunction of NPs, PPs, ADJPs, DETs and clauses; possessives; pronouns; bare determiners; prenominal adjectives and compound nominals; measure phrases; complex DETs; numbers; correction utterances; interjections; and politeness phrases. The grammar includes a flexible mechanism for specifying domain-specific lexical sortal constraints, which for example makes it possible for a transitive verb to constrain the sortal type of its direct object, or for a noun to constrain the sortal type of a pre-modifying adjective.

The original grammar was developed for a robotic command and control domain (Rayner et al., 2000b). Further development work has been systematically carried out by extending it to cover five more domains: a home automation application (Rayner et al., 2001b); an ATIS-like travel planning application; a medical speech translator (Rayner and Bouillon, 2002); an intelligent procedure assistant (Aist et al., 2002); and a mobile geology assistant. Examples of typical utterances for each of these domains are shown in Table 1. Transcribed corpora of at least several hundred utterances exist for all the domains, and have been used to debug the grammar coverage.

2.2 The grammar specialisation tools

XXX implements a version of the grammar specialisation scheme which extends the Explanation Based Learning method described in (Rayner et al., 2002). A grammar built on top of the general grammar is transformed into a specialised unification grammar in the following processing stages:

1. The training corpus is converted into a “treebank” of parsed representations, using the left-corner parser representation of the grammar.

2. Each parsed representation in the treebank is processed into one or more specialised unification grammar rules, using the EBL algorithm (van Harmelen and Bundy, 1988; Rayner, 1988).

3. Duplicate specialised rules are merged, so that each unique rule is tagged with the number

of training examples from which it could have been derived.

4. Each rule is subjected to a binarisation transform, to ensure that no rule in the final set has more than two daughters.

The EBL algorithm performs *grammatical* specialisation; it creates new grammar rules by merging the constraints present in existing rules. Thus, in an air travel domain, an example like "show me flights to Boston" might induce a rule schematically of the form

```
S -> V, NP, NP
```

The features on this derived rule will contain various constraints, the exact nature of which will depend on the original general grammar. With the grammar of Section 3.2, the derived rule will for example constrain the first daughter NP's `case` feature to unify with the value `nonsubj`, and its `np_type` (sortal) feature to unify with the value of the corresponding `indobj_type` feature on the V.

In general, the coverage of a specialised grammar derived using the EBL method is a strict subset of the coverage of the original grammar, in a sense made precise in (Rayner et al., 2000a). The loss of coverage is compensated by the specialised grammar's simpler structure, which typically produces faster processing and decreased ambiguity. In practice, the net result is often that the specialised grammar yields more accurate results than the original one, both for text-based parsing (Samuelsson and Rayner, 1991; Samuelsson, 1994) and for spoken language processing (Rayner et al., 2002).

2.3 The UG to CFG compiler

The basic compilation mechanism for the REGULUS UG to CFG compiler, *enumerative expansion*, is described in (Rayner et al., 2001a). The compiler essentially performs non-deterministic expansion of the unification grammar to yield a context-free grammar, and then filters the result to remove unreachable rules. Although a completely naive implementation of enumerative expansion is insufficient for any but the very smallest grammars, (Rayner et al., 2001a) showed how some simple enhancements can greatly increase its power. In particular, it is possible to effect a major reduction in the size of the expanded rule-space by performing the *singular variable elimination* transform, and it is also possible to filter the space of expanded rules in time linear in the number of rules.

For grammars as large as the general grammar described in Section 2.1, the methods used by the REGULUS compiler turn out to be inadequate; the space of expanded rules is too large to generate in its entirety. The REGULUS compiler breaks down not only on the general grammar itself, but also on specialised grammars derived from it. The problem is not the number of rules in the grammar, but rather the number of features, which is unaffected by the specialisation process; the size of the expanded rule space increases exponentially with the number of features. Other experiments showed that the GEMINI compiler (Moore, 1998) was similarly incapabable of compiling these grammars.

It does however turn out to be possible to compile grammars with large numbers of features by adding a further refinement to the enumerative expansion algorithm, which interleaves the expansion and filtering phases. In XXX, the set of features is divided into an ordered list of subsets; each subset is in turn expanded non-deterministically, and the result is then filtered using the algorithms of (Rayner et al., 2001a) before proceeding to the next subset. The experiments in Section 3.2 show that compile times still increase as the feature set expands, but only slowly.

3 Experiments

This section describes a series of experiments which empirically investigate the claims made in Section 2. The experiments are divided into four groups, as follows:

1. Comparison of recognisers derived from general grammars and recognisers derived from hand-coded grammars (Section 3.1).

2. Scalability of specialised grammars with respect to size of the original general grammar (Section 3.2).

3. Scalability of specialised grammars with respect to size of training corpora (Section 3.3).

4. Effect of interleaved expansion and filtering on efficiency of UG to CFG compilation (Section 3.4).

Corpora from two domains were used in these experiments[2]. The first corpus, used in Section 3.1, was collected in the September 2002 field test for the Mobile Agents robotic geologist's assistant at Meteor Crater, AZ. The speech data in this corpus was collected in an "open-mic" configuration, where everything spoken by the subject was recorded, whether the speech was intended for the robotic assistant or not, and is acoustically noisy, since the data was collected either from a subject inside a space suit, or outdoors in high wind. The Mobile Agents corpus used in our experiment contains 608 in domain utterances (3535 words) from 8 speakers, which we divided into a 485 utterance training set and a 123 utterance test set.

The remaining experiments were run on the Personal Satellite Assistant (PSA) corpus, which was collected in user tests of the PSA system. The PSA corpus has 10513 utterances (38943 words) from 27 speakers, which we divided into a training set of 5394 utterances and a test set of 5119 utterances. It is worth mentioning that the length distribution on this corpus is extremely skewed. A little more than half the corpus (5344 utterances) consists of one-word yes/no responses; the remaining 5169 utterances have a mean length of 6.5 words. The test sets for both corpora were unseen data for the purposes of these experiments.

We used the XXX system to compile a variety of language models and associated recognisers, using the general grammar of Section 2.1 and the methods of Section 2.2 and 2.3. We evaluated the language models and recognisers both in terms of compile-time and run-time performance. With reference to the compilation process, we were interested in the time taken to perform EBL-based grammar specialisation (**EBL_T**), time taken to compile the specialised unification grammar into a CFG language model (**UG_CFG_T**), and the number of nodes in the Nuance recognition package's node array (**#Nodes**).

[2]We have concentrated on these two domains here, since they are the ones for which we have best data. Two of the other systems built using XXX are in parallel being submitted to the demo track at this conference.

At run-time, we were interested in the accuracy of the resulting recogniser and on its speed, measured on both training and test data. Grammar-based recognisers reject a certain proportion of their input: utterances are typically rejected if they are either well outside grammar coverage, or acoustically too noisy. We consequently present accuracy in terms of three numbers. **WER** represents the standard word error rate. **REJ** measures the proportion of utterances rejected by the recogniser, and **AWER** ("adjusted word error rate") measures the word error rate on the utterances that were not rejected. Speed is as usual measured as a multiple of real-time. Experiments were run on a 1.9 GHz Pentium 4, using Nuance 8 and SICStus 3.8.5.

3.1 Specialised versus hand-coded grammars

The data collected in the Mobile Agents field test used a language model derived from a hand-optimised unification grammar. Due to the existence of this hand-optimized grammar, the Mobile Agents domain provided the best comparison of hand-built to automatically generated. After the field test, the 485 utterance training set was used to build a specialised grammar from the general grammar. The specialised version was built in one day, and underwent no additional tuning. Table 2 compares the performance of this specialised grammar with the hand-coded grammar on the training and test sets respectively. The specialised grammar performs quite well, generally out-performing the hand-coded grammar. The results also demonstrate that it is possible to build effective specialised grammars in this domain from relatively small amounts of training data.

Version	WER (%)	REJ (%)	AWER (%)	xCPUrt (%)
Measured on training set				
Hand-Coded	12.56	7.72	4.54	58.79
Specialised	5.29	1.86	3.21	11.78
Measured on test set				
Hand-Coded	9.50	7.50	3.25	57.50
Specialised	5.49	2.44	2.91	13.74

Table 2: Comparing an automatically specialised grammar with a hand-coded grammar: WER, proportion rejected, adjusted WER and speed.

3.2 Scalability of general grammars

The experiments reported in this section investigate scalability with respect to the size of the general unification grammar. We trained versions of the domain-specific PSA grammar on reconstructed versions of the general unification grammar corresponding to merges of increasingly large numbers of domains, ending with a grammar that merged all six domains. Table 3 presents statistics on the sizes of the various versions, measured in terms of the numbers of rules and features.

Version	Domains included	#Rules	#Feats
1	PSA	70	42
2	PSA,HA	74	46
3	PSA,HA TD	106	56
4	PSA,HA TD,MST	127	64
5	PSA,HA TD,MST IPA	139	68
6	PSA,HA TD,MST IPA,MA	145	68

Table 3: Versions of the general grammar used for grammar scaling experiments. Domain abbreviations as in Table 1.

It seemed to us that increasing the size of the general grammar could potentially cause two undesirable effects. Firstly, compile times could increase unmanageably; secondly, runtime performance of the resulting recognisers could be adversely affected. In fact, the experiments suggest that neither of these things happens. Table 4 shows a fairly modest increase in compile times and node arrays as the size of the general grammar increases. A large part of the increase occurred in the move from Version 1 to Version 2; this appears to be due to the fact that noun compounding rules appeared at this point, significantly increasing grammar coverage and correspondingly decreasing WER. Table 5 shows that word error rates and rejection rates remain stable, and processing speed decreases only slightly as the size of the general grammar increases.

Version	EBL_T (secs)	UG_CFG_T (secs)	#Nodes
1	110	28	2590
2	155	51	10035
3	205	55	12768
4	300	65	12760
5	341	85	14540
6	409	84	14540

Table 4: Compile-time behaviour with increasingly large general grammars: time for EBL processing, time for compilation to CFG form, and number of nodes in resulting recogniser. Grammar versions as in Table 3.

3.3 Scalability of training corpora

We next investigated the effects on compile-time and run-time performance as we increased the number of training examples used in the EBL specialisation process. We again used the PSA domain for the experiments. Table 6 suggests that compile times grow at most linearly with the size of the corpus, and that the size of the node array grows sub-linearly.

The runtime performance figures in Table 7 suggests that recogniser performance tops out fairly quickly; the error rates for 5000 training examples are only marginally better than those for 2500.

3.4 Interleaved expansion and filtering

Finally, we investigated the value of interleaving expansion and filtering in the UG to CFG compilation stage (cf. Section 2.3). By suppressing features, we created a series of versions of the specialised PSA grammar with the same number of rules, but increasing numbers of features. We then attempted to compile these grammars using a version of the compiler which carried out a single expansion step which expanded all features simultaneously, followed by a single filtering step.

In Table 8 we see that the number of expanded rules and the compilation time both increase rapidly, and exceeded resource bounds when the grammar reached 40 features. These results contrast sharply with those in Table 4, where compilation times increase only slowly as the grammar grows from 42 features (Version 1) to 68 features (Version 6).

Version	WER (%)	REJ (%)	AWER (%)	xCPUrt (%)
Measured on training set				
1	20.64	5.66	10.99	6.43
2	10.32	2.02	7.01	9.57
3	10.27	1.73	7.42	10.29
4	10.25	1.65	7.52	10.83
5	10.26	1.69	7.46	11.69
6	10.26	1.69	7.46	11.63
Measured on test set				
1	22.61	6.22	12.46	6.60
2	13.88	2.43	9.57	10.11
3	14.17	2.23	10.34	10.93
4	14.08	2.11	10.39	11.48
5	14.13	2.17	10.38	12.26
6	14.13	2.17	10.38	12.24

Table 5: Runtime behavior with increasingly large general grammars: WER, proportion rejected, adjusted WER and speed. Grammar versions as in Table 3.

4 Summary and conclusions

This paper has described an approach to grammar-based language modelling, in which all models are derived from a single linguistically motivated unification grammar. Domain-specific CFG language models are produced by first specialising the grammar using an automatic corpus-based method, and then compiling the resulting specialised grammars into CFG form. We have presented results showing that recognisers for multiple, fairly different, domains can be derived from a single general grammar. The process remains tractable as the size of the general grammar increases, and also scales well with the size of the training corpus used. We have also shown that a simple method for compiling unification grammars into CFG form can be successfully applied to grammars containing large numbers of features, if it is enhanced by suitable interleaving of the expansion and filtering stages.

Based on experience with other broad-coverage grammars, we believe that our current general grammar will only need to become moderately larger in order to achieve very reasonable cross-domain coverage for a large variety of domains. The grammar, and the tools used to perform the specialisation and compilation operations, are all Open Source, and will soon be made generally available to the community. The final version of the paper will contain instructions for accessing and downloading them.

#Examples	EBL_T (secs)	UG_CFG_T (secs)	#Nodes
250	21	22	4067
500	40	27	9236
1000	79	40	12545
2500	194	63	12827
5000	385	84	14462

Table 6: Compile-time behaviour with increasingly large training corpora: time for EBL processing, time for compilation to CFG form, and number of nodes in resulting recogniser.

#Examples	WER (%)	REJ (%)	AWER (%)	xCPUrt (%)
Measured on training set				
250	16.12	3.01	11.33	5.25
500	13.33	2.46	9.45	6.03
1000	12.09	2.09	8.79	7.64
2500	11.19	1.95	7.71	9.39
5000	10.39	1.76	7.45	11.90
Measured on test set				
250	21.04	5.07	12.73	5.36
500	17.30	3.84	11.23	6.17
1000	16.63	3.58	10.86	7.67
2500	14.91	2.35	10.72	9.70
5000	14.25	2.20	10.47	12.29

Table 7: Runtime behavior with increasing size of training corpus: WER, proportion rejected, adjusted WER and speed.

References

G. Aist, J. Dowding, B.A. Hockey, and J. Hieronymus. 2002. An intelligent procedure assistant for astronaut training and support. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (demo track)*, Philadelphia, PA.

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.

#Features	Rules before filtering	Rules after filtering	Time (secs)
5	378	342	0.2
10	412	364	0.1
15	735	367	0.2
20	771	388	0.2
25	1189	457	0.3
30	2027	468	0.7
35	9245	1052	5.1
36	56849	1082	5.3
37	56849	1082	19.1
38	210933	1086	99.9
39	210933	1086	75.3
40	exceeded resource limits		

Table 8: Effect on compilation performance of increasing number of features in the grammar, when all features are expanded simultaneously and then filtered.

J.-C. Chappelier, M. Rajman, R. Aragues, and A. Rozenknop. 1999. Lattice parsing for speech recognition. In *Proceedings of the 6th TALN*, Carges, Corsica.

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.

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.

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

R. Moore. 2000. Improved left-corner chart parsing for large context-free grammars. In *Proceedings of the 6th International Workshop on Parsing Technologies*, pages 171–182.

Nuance, 2003. http://www.nuance.com. As of 25 February 2003.

S.G. Pulman. 1992. Syntactic and semantic processing. In H. Alshawi, editor, *The Core Language Engine*, pages 129–148. MIT Press, Cambridge, Massachusetts.

M. Rayner and P. Bouillon. 2002. A phrasebook style medical speech translator. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (demo track)*, Philadelphia, PA.

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

M. Rayner, B.A. Hockey, and F. James. 2000b. A compact architecture for dialogue management based on scripts and meta-outputs. In *Proceedings of the 6th Applied Natural Language Processing Conference*, Seattle, WA.

M. Rayner, B.A. Hockey, and F. James. 2000c. 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. 2001a. 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, I. Lewin, G. Gorrell, and J. Boye. 2001b. Plug and play spoken language understanding. In *Proceedings of the 2nd ACL SIGDIAL Workshop on Discourse and Dialogue*, Aalborg, Denmark.

M. Rayner, B.A. Hockey, and J. Dowding. 2002. Grammar specialisation meets language modelling. In *Proceedings of the 7th International Conference on Spoken Language Processing (ICSLP)*, Denver, CO.

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, Tokyo, Japan.

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, 2003. http://www.speechworks.com. As of 25 Feb 2003.

T. van Harmelen and A. Bundy. 1988. Explanation-based generalization = partial evaluation (research note). *Artificial Intelligence*, 36:401–412.