Comparison of Grammar-Based and Statistical Language Models Trained on the Same Data

Beth Ann Hockey and Manny Rayner
Mail Stop T27A-2
NASA Ames Research Center
Moffett Field, CA 94035
bahockey@email.arc.nasa.gov, mrayner@riacs.edu

Abstract

This paper presents a methodologically sound comparison of the performance of grammar-based (GLM) and statistical-based (SLM) recognizer architectures using data from the Clarissa procedure navigator domain. The Regulus open source packages make this possible with a method for constructing a grammar-based language model by training on a corpus. We construct grammar-based and statistical language models from the same corpus for comparison, and find that the grammar-based language models provide better performance in this domain. The best SLM version has a semantic error rate of 9.6%, while the best GLM version has an error rate of 6.0%. Part of this advantage is accounted for by the superior WER and Sentence Error Rate (SER) of the GLM (WER 7.42% versus 6.27%, and SER 12.41% versus 9.79%). The rest is most likely accounted for by the fact that the GLM architecture is able to use logical-form-based features, which permit tighter integration of recognition and semantic interpretation.

Introduction

The Clarissa procedure navigator is a spoken dialogue system designed to assist astronauts on the International Space Station (ISS) in executing the type of detailed procedures that constitute much of their workload. This system, installed on the ISS in January 2005, provides hands- and eyes-free navigation of procedures through a mixed-initiative dialogue interface.

Clarissa uses a grammar-based language model (GLM) recognition architecture. At the start of the project, we had several reasons for choosing this approach over the more popular statistical language model (SLM) approach. First, we had no available training data. Second, the system was to be designed for experts who would have time to learn its coverage; although there is not a great deal to be found in the literature, an earlier study in which we had been involved (Knight *et al.* 2001) suggested that grammar-based systems outperformed statistical ones for this kind of user. A third consideration that affected our choice was the importance of "open mic" recognition; there was again little published research, but folklore results suggested that grammar-based recognition methods had an edge over statistical ones here too.

While these factors biased us in favor of the grammar-based architecture, we wanted to implement a framework which would allow us to compare grammar-based methods with statistical ones in a methodologically sound way, and also retain the option of switching from a grammar-based framework to a statistical one if that later appeared justified. The Regulus (Rayner, Hockey, & Dowding 2003; Regulus 2005) and Alterf (Rayner & Hockey 2003) platforms, which we have developed under Clarissa and other earlier projects, are designed to meet these requirements.

The basic idea behind the Regulus platform is to construct grammar-based language models using example-based methods driven by small corpora. Since under this approach, grammar construction is a corpus-driven process, the same corpora can also be used to build normal statistical language models, facilitating a direct comparison between the two methodologies. On its own, however, Regulus only permits comparison at the level of recognition strings. Alterf extends the paradigm to the semantic level, by providing a uniform trainable semantic interpretation framework which can work on either surface strings or logical forms.

Exploiting the special properties of Regulus and Alterf, this paper presents an evaluation of the Clarissa system's speech understanding component, which provides a methodologically sound comparison between the implemented grammar-based architecture and a plausible statistical/robust counterpart. In terms of speech understanding on in-domain utterances, our bottom-line conclusion is that, at least for this application, grammar-based methods significantly outperform statistical ones.

Regulus

The core functionality provided by the Regulus Open Source platform is compilation of typed unification grammars into annotated context-free grammar language models expressed in Nuance Grammar Specification Language (GSL) notation (Nuance 2003). GSL language models can be converted into runnable speech recognizers by invoking the Nuance Toolkit compiler utility, so the net result is the ability to compile a unification grammar into a speech recognizer.

Experience with grammar-based spoken dialogue systems shows that there is usually a substantial overlap between the

structures of grammars for different domains. This is hardly surprising, since they all ultimately have to model general facts about the linguistic structure of English and other natural languages. It is consequently natural to consider strategies which attempt to exploit the overlap between domains by building a single, general grammar valid for a wide variety of applications. A grammar of this kind will probably offer more coverage (and hence lower accuracy) than is desirable for any given specific application. It is however feasible to address the problem using corpus-based techniques which extract a specialized version of the original general grammar.

Regulus implements a version of the grammar specialization scheme which extends the Explanation Based Learning method described in (Rayner, Hockey, & Dowding 2002). There is a general unification grammar, loosely based on the Core Language Engine grammar for English (Pulman 1992), which has been developed over the course of about ten individual projects. The semantic representations produced by the grammar are in a simplified version of the Core Language Engine's Quasi Logical Form notation (van Eijck & Moore 1992).

A specialized Nuance grammar is created from the general grammar with the following processing stages:

1. The training corpus is converted into a "treebank" of parsed representations. This is done using a left-corner parser representation of the grammar.
2. The treebank is used to produce a specialized grammar in Regulus format, using the EBL algorithm (van Harmelen & Bundy 1988; Rayner 1988).
3. The final specialized grammar is compiled into a Nuance GSL grammar.

Alterf

Alterf is another Open Source toolkit, whose purpose is to allow a clean combination of rule-based and corpus-driven processing in the semantic interpretation phase. Alterf characterises semantic analysis as a task slightly extending the "decision-list" classification algorithm (Yarowsky 1994; Carter 2000). We start with a set of *semantic atoms*, each representing a primitive domain concept, and define a semantic representation to be a non-empty set of semantic atoms. For example, in Clarissa we represent the utterances

please speak up
set an alarm for five minutes from now
no i said go to the next step

respectively as

{increase_volume}
{set_alarm, 5, minutes}
{correction, next_step}

where increase_volume, set_alarm, 5, minutes, correction and next_step are semantic atoms. As well as specifying the permitted semantic atoms themselves, we also define a *target model* which for each atom specifies the other atoms with which it may legitimately combine. Thus here, for example, correction may legitimately combine with any atom, but minutes may only combine with correction, set_alarm or a number.

Training data consists of a set of utterances, in either text or speech form, each tagged with its intended semantic representation. We define a set of *feature extraction rules*, each of which associates an utterance with zero or more features. Feature extraction rules can carry out any type of processing. In particular, they may involve performing speech recognition on speech data, parsing on text data, application of hand-coded rules to the results of parsing, or some combination of these. Statistics are then compiled to estimate the probability $p(a \mid f)$ of each semantic atom a given each separate feature f, using the standard formula

$$p(a \mid f) = (N_f^a + 1)/(N_f + 2)$$

where N_f is the number of occurrences in the training data of utterances with feature f, and N_f^a is the number of occurrences of utterances with both feature f and semantic atom a.

The decoding process follows (Yarowsky 1994) in assuming complete dependence between the features. Note that this is in sharp contrast with the Naive Bayes classifier (Duda, Hart, & Stork 2000), which assumes complete *independence*. Of course, neither assumption can be true in practice; however, as argued in (Carter 2000), there are good reasons for preferring the dependence alternative as the better option in a situation where there are many features extracted in ways that are likely to overlap.

We are given an utterance u, to which we wish to assign a representation $R(u)$ consisting of a set of semantic atoms, together with a target model comprising a set of rules defining which sets of semantic atoms are consistent. The decoding process proceeds as follows:

1. Initialise $R(u)$ to the empty set.
2. Use the feature extraction rules and the statistics compiled during training to find the set of all triples $\langle f, a, p \rangle$ where f is a feature associated with u, a is a semantic atom, and p is the probability $p(a \mid f)$ estimated by the training process.
3. Order the set of triples by the value of p, with the largest probabilities first. Call the ordered set T.
4. Remove the highest-ranked triple $\langle f, a, p \rangle$ from T. Add a to $R(u)$ iff the following conditions are fulfilled:
   - $p \geq p_t$ for some pre-specified threshold value p_t.
   - Addition of a to $R(u)$ results in a set which is consistent with the target model.
5. Repeat step (4) until T is empty.

Intuitively, the process is very simple. We just walk down the list of possible semantic atoms, starting with the most probable ones, and add them to the semantic representation we are building up when this does not conflict with the consistency rules in the target model. We stop when the atoms suggested are too improbable, that is, have probabilities below a specified threshold.

Using Regulus and Alterf in Clarissa

We now describe the details of how we have used the Regulus and Alterf platforms in Clarissa. The Clarissa Regulus grammar is composed of the general Regulus grammar and the general function-word lexicon, together with a Clarissa-specific domain lexicon containing 313 lemmas, which realise 857 surface lexical rules. (Most of the difference is accounted for by verbs, which in the Regulus grammar have five surface forms). Table 1 summarizes the data for the domain-specific lexicon.

The general Regulus grammar, the Clarissa lexicon and the Clarissa training examples are used to create a grammar tailored to the Clarissa domain through the grammar specialization process already outlined. This domain specific grammar is then compiled into a Nuance compatible grammar which is used to build the runtime recognition component. The training corpus we use for the current version of the system contains 3297 examples; of these, 1349 have been hand-constructed to represent specific words and constructions required for the application, while the remaining 1953 are transcribed examples of utterances recorded during system development.

The parameters guiding the grammar specialization process have been chosen to produce a fairly "flat" grammar, in which many noun-phrases become lexical items. This reflects the generally stereotypical nature of language in the Clarissa domain. The specialized unification grammar contains 491 lexical and 162 non-lexical rules; Table 2 shows examples of specialized grammar rules, together with associated frequencies of occurrence in the training corpus. The specialized grammar is compiled into a CFG language model containing 427 non-terminal symbols and 1999 context-free productions. Finally, the training corpus is used a second time to perform probabilistic training of the CFG language model using the Nuance `compute-grammar-probs` utility, and the resulting probabilistic version of the language model is compiled into a recognition package using the `nuance-compile` utility.

On a 1.7GHz P4 laptop running SICStus 3.8.5, the whole compilation process takes approximately 55 minutes. This divides up as about 30 minutes to parse the training corpus, 10 minutes to extract the specialized grammar from the parsed treebank, 10 minutes to perform unification grammar to CFG language model compilation, and 5 minutes for probabilistic grammar training and Nuance recognizer compilation.

Semantic representations produced by the Clarissa grammar are general domain-independent logical forms. By construction, the same representations are produced by the specialized grammar and the derived recognizer. The Alterf package is used to convert these general representations into unordered lists of semantic atoms; a final post-processing stage transforms Alterf output into the "dialogue moves" used as input by the dialogue manager. Figure 1 shows examples of these different levels of representation.

Recall that the Alterf algorithm requires definition of feature extraction rules, so that it can then trained to acquire associations between extracted features and cooccurring semantic atoms. We have experimented with three different kinds of feature extraction rules: surface N-grams, hand-coded surface patterns and hand-coded logical form patterns.

Surface
"no i said go to step five point three"

Logical form

```
[[interjection, correction],
 [imp,
  form(imperative,
       [[go,
         term(pro, you, []),
         [to, term(null, step,
           [[number,
             [decimal,5,3]]]
               )]]])]]
```

Alterf output

```
[correction, go_to, [decimal,5,3]]
```

Dialogue move

```
correction(go_to([decimal,5,3]))
```

Figure 1: Examples showing different levels of representation for a Clarissa utterance. We show the surface words, the general logical form produced by the Regulus grammar and derived recognizer, the list of semantic atoms produced by Alterf, and the dialogue move.

Surface N-gram features are the simplest type. First, the recognized string is tokenised using a phrase-spotting grammar that identifies numbers, times, durations in minutes and seconds, and a few other similar types of expression. Bigrams and trigrams are then extracted from the tokenised string. Thus for example the string "go to step five point three" would be tokenised as

```
[*start*, go, to, step,
        [decimal,5,3], *end*]
```

and yields the trigrams

```
[*start*, go, to]
[go, to, step],
[to, step, [decimal,5,3]]
[step, [decimal,5,3], *end*]
```

When calculating associations, Alterf replaces numbers and similar constructs with generic tokens: after this replacement, the trigrams above become

```
[*start*, go, to]
[go, to, step],
[to, step, *decimal_number*]
[step, *decimal_number*, *end*]
```

Thus in this case Alterf will not derive an association between the trigram `[to, step, [decimal,5,3]]` and the semantic atom `[decimal,5,3]`, but rather between the generic trigram `[to, step,`

POS	#Entries		Examples	Example contexts
	Lemmas	Words		
Verb	129	645	continue go put	"**continue**" "**go** to step three" "**put** a voice note on step six"
Noun	99	127	caution alarm mode	"read **caution** before step eleven" "list **alarms**" "enter challenge verify **mode**"
Number	25	25	zero five	"set alarm for ten **zero fiver**"
Interjection	20	20	copy	"**copy** go to step three"
Preposition	15	15	on	"give me help **on** navigation"
Adjective	15	15	skipped	"list **skipped** steps"
Adverb	10	10	louder	"speak **louder**"
Total	313	857		

Table 1: Summary information for Clarissa lexicon. For each part of speech, we give the number of lexicon entries, example words, and example contexts.

`*decimal_number*]` and the generic semantic atom `*decimal_number*`.

Unsurprisingly, we discovered that surface N-gram features were not particularly reliable. We then implemented two more sets of feature extraction rules, which defined different types of hand-coded patterns. The first set consists of conventional phrasal patterns over the tokenised recognition string, written in a simple string-matching language; the second set encodes structural patterns in the logical form. Examples of string-based and logical-form-based patterns are shown in Figure 2. The version of Clarissa described here has 216 string-based patterns and 305 logical-form-based patterns. The patterns have been developed and debugged using the 3297 utterance training corpus: on this corpus, each set of patterns has a classification error rate of about 0.5%.

Evaluating speech understanding performance

Having described the speech understanding architecture, we now present an evaluation. There are a large number of types of experiment which we could potentially have carried out. Given limited resources, we decided to focus on two main questions:

- How does the Regulus grammar-based framework compare against a more conventional framework using a class N-gram language model and a set of phrase-spotting rules?
- How do different types of features compare against each other? In particular, are logical-form-based patterns more effective than string-based or N-gram patterns, and is it useful to combine several types of pattern?

The next issue to resolve is the choice of appropriate performance metrics and test data. Given that we are essentially interested in speech understanding performance, the obvious metric is semantic error rate, by which we will specifically mean the proportion of utterances which fail to receive a correct semantic interpretation after Alterf processing. For completeness, we will also present figures for Word Error Rate (WER) and Sentence Error Rate (SER).

The choice of appropriate test data raises interesting questions. A data collection was done using naive subjects who had had no previous exposure to the system. Comparison of the results of that data collection with the small amount of data we had from prospective astronaut users revealed a serious mismatch. Astronauts are exceptionally intelligent and capable individuals, who had a strong motivation to learn to use the system; the naive subjects were normal people, with no particular reason to want to acquire the relevant skills. The performance figures reflected this imbalance, with the astronauts scoring enormously better than nearly all of the naive subjects.

Our conclusion was that data collected from naive subjects was not useful, and that a much better fit came from the data recorded by system developers during the course of the project. Although the developers know the system a little better than the astronaut users, our intuitive observation was that the difference was not large, and that the astronauts would probably catch up after only a relatively short period of use[1]. The developers' superior knowledge of the system was also counterbalanced to some extent by the fact that several of them were not native American speakers, whereas all the relevant astronauts were.

We are of course aware that this is not an ideal situation from the point of view of experimental design, but the virtual impossibility of getting significant quantities of astronaut time forced us to adopt a compromise: this appeared to be the best one. The figures below are thus based on a sample of 8158 in-domain utterances (23369 words) collected and transcribed during the course of the project. By "in-domain", we mean here that the utterances expressed commands meaningful in the context of the Clarissa task, and that the system should ideally have responded to them; we do not necessarily imply that the utterances were all syntac-

[1] None of the astronauts used the system for more than a total of one hour. We had originally expressed concern that this would not give them adequate time to learn to use it effectively; after watching a sample training session, we were forced to admit that we had underestimated their capabilities.

Rule	Freq	Example
S --> V NP	606	“[delete] [the voice note]”
NP --> NUMBER	511	“[one hundred thirty seven]”
ADJ --> next	508	“no i said [next]”
NP --> step NUMBER	481	“play voice note on [step four]”
SIGMA --> INTERJECTION NP	344	“[no i meant] [four point one]”
S --> V P NP	321	“[go] [to] [row one]”
S --> V NP POST_MODS	295	“[set] [timer] [for two minutes]”
NUMBER --> NUMBER point NUMBER	278	“[twenty five] [point] [nine]”
S --> V	259	“[speak up]”
POST_MODS --> P NP	228	“set alarm [at] [three zero six]”
V --> go back	108	“[go back]”
S --> V NP NP	78	“[show] [me] [figure one]”
NP --> the voice note	40	“cancel [the voice note]”
S --> V P NP POST_MODS	28	“[go] [to] [the note] [before step one]”
NP --> what procedure	12	“[what procedure] are we on”
S --> NP V NP	11	“[what] [is] [the current step]”
V --> exit	6	“[exit] review mode”
NP --> value	3	“[value] is undefined”

Table 2: Examples of rules in specialized version of Clarissa grammar. For each rule we give the context-free skeleton, the frequency of occurrence in the training corpus, and an example.

tically well-formed or within the coverage of the grammar. The data had not previously been used for development purposes, and can reasonably be considered as unseen.

In order to compare the Regulus-constructed GLM-based recognizer with an SLM-based recognizer architecture, we used the Nuance SayAnything© tool and the same 3297 utterance training set to build a standard class N-gram model. We defined three classes, for numbers, times, and objects that could be displayed using the “show” command. Raw recognition performance figures for the two recognizers, measured in terms of WER and SER, are shown in Table 3.

Recognizer	WER	SER
GLM	6.27%	9.79%
SLM	7.42%	12.41%

Table 3: WER and SER for the Regulus-based recognizer (GLM) and a conventional class N-gram recognizer (SLM) trained on the same data.

The main experimental results are presented in Table 4. Here, we contrast speech understanding performance for the Regulus-based recognizer and the class N-gram recognizer, using several different sets of Alterf features. For completeness, we also present results for simulated perfect recognition, i.e. using the reference transcriptions. We used six different sets of Alterf features:

N-grams: N-gram features only.

LF: Logical-form-based patterns only.

String: String-based patterns only.

String + LF: Both string-based and logical-form based patterns.

String + N-grams: Both string-based and N-gram features.

String + LF + N-grams: All types of features.

As can be seen, the GLM performs very considerably better than the SLM. The best SLM version, S-3, has a semantic error rate of 9.6%, while the best GLM version, G-4 has an error rate of 6.0%, a relative improvement of 37%. Part of this is clearly due to the fact that the GLM has better WER and SER than the SLM. However, Table 3 shows that the relative improvement in WER is only 15% (7.42% versus 6.27%), and that in SER is 21% (12.41% versus 9.79%). The larger improvement by the GLM version at the level of semantic understanding is most likely accounted for by the fact that it is able to use logical-form-based features, which are not accessible to the SLM version. Although logical-form-based features do not appear to be intrinsically more accurate than string-based features (contrast rows T-2 and T-3), the fact that they allow tighter integration between semantic understanding and language modelling is intuitively advantageous.

It is interesting to note that the combination of logical-form-based features and string-based features outperforms logical-form-based features alone (rows G-4 and G-5). Although the difference is small (6.0% versus 6.3%), a pairwise comparison shows that it is significant at the 1% level according to the McNemar sign test. There is no clear evidence that N-gram features are very useful. This supports the standard folk-lore result that semantic understanding components for command and control applications are more appropriately implemented using hand-coded phrase-spotting patterns than general associational learning techniques.

Finally, Table 5 presents a breakdown of speech understanding performance, by utterance length, for the best GLM-based and SLM-based versions of the system. There are two main points we want to make here. First, speech un-

String-based patterns

```
% ‘‘decrease’’ or ‘‘reduce’’ followed by ‘‘volume’’
% indicates the atom decrease_volume
surface_pattern([decrease/reduce,’...’,volume], decrease_volume).

% ‘‘back’’ not following ‘‘go’’ and at the end
% indicates the atom previous_line
surface_pattern([not_word(go),back,’*end*’], previous_line).

% ’’put’’ followed by ‘‘voice note’’
% indicates the atom record_voice_note
surface_pattern([put,’...’,voice,note], record_voice_note).
```

Logical-form-based patterns

```
% ‘‘decrease’’ or ‘‘reduce’’ with object an NP whose head is ‘‘volume’’
% indicates the atom decrease_volume
lf_pattern([decrease,_,term(_,volume,_)],decrease_volume).
lf_pattern([reduce,_,term(_,volume,_)],decrease_volume).

% ‘‘back’’ used as an interjection
% indicates the atom previous_line
lf_pattern([interjection,back],previous_line,back).

% ‘‘put’’ with object an NP whose head is ‘‘voice_note’’
% indicates the atom record_voice_note
lf_pattern([put,_,term(_,voice_note,_),_],record_voice_note).
```

Figure 2: Examples of string-based and logical-form-based patterns used in Clarissa.

derstanding performance remains respectable even for the longer utterances; second, the performance of the GLM-based version is consistently better than that of the SLM-based version for all utterance lengths.

Conclusion

For this kind of task, there is reasonable evidence that grammar-based recognition methods work better than statistical ones. The extra robustness of statistical methods does not appear to outweigh the fact that the grammar-based approach permits tighter integration of recognition and semantic interpretation. Speech understanding performance was very substantially better with the grammar-based method. We were able to make a clear comparison between the two methods because we used a carefully constructed methodology which built the grammar-based recognizer from a corpus, but there is no reason to believe that other ways of building the grammar-based recogniser would have led to inferior results.

References

Carter, D. 2000. Choosing between interpretations. In Rayner, M.; Carter, D.; Bouillon, P.; Digalakis, V.; and Wirén, M., eds., *The Spoken Language Translator*. Cambridge University Press.

Duda, R.; Hart, P.; and Stork, H. 2000. *Pattern Classification*. New York: Wiley.

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

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

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

Rayner, M., and Hockey, B. 2003. Transparent combination of rule-based and data-driven approaches in a speech understanding architecture. In *Proceedings of the 10th Conference of the European Chapter of the Association for Computational Linguistics*.

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

Rayner, M.; Hockey, B.; and Dowding, J. 2003. An open source environment for compiling typed unification grammars into speech recognisers. In *Proceedings of the 10th Conference of the European Chapter of the Association for Computational Linguistics (interactive poster and demo track)*.

Rayner, M. 1988. Applying explanation-based generalization to natural-language processing. In *Proceedings of the*

International Conference on Fifth Generation Computer Systems, 1267–1274.

Regulus. 2005. http://sourceforge.net/projects/regulus/. As of 8 January 2005.

van Eijck, J., and Moore, R. 1992. Semantic rules for English. In Alshawi, H., ed., *The Core Language Engine*. MIT Press. 83–116.

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

Yarowsky, D. 1994. Decision lists for lexical ambiguity resolution. In *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics*, 88–95.

Version	Rec	Features	Errors		
			Rejected	Incorrect	Total
T-1	Text	N-grams	7.3%	5.9%	13.2%
T-2	Text	LF	3.1%	0.5%	3.6%
T-3	Text	String	2.2%	0.8%	3.0%
T-4	Text	String + LF	0.8%	0.8%	1.6%
T-5	Text	String + LF + N-grams	0.4%	0.8%	1.2%
G-1	GLM	N-grams	7.4%	9.7%	17.1%
G-2	GLM	LF	1.4%	4.9%	6.3%
G-3	GLM	String	2.9%	4.8%	7.7%
G-4	**GLM**	**String + LF**	**1.0%**	**5.0%**	**6.0%**
G-5	GLM	String + LF + N-grams	0.7%	5.4%	6.1%
S-1	SLM	N-grams	9.6%	11.9%	21.5%
S-2	SLM	String	2.8%	7.4%	10.2%
S-3	SLM	String + N-grams	1.6%	8.0%	9.6%

Table 4: Speech understanding performance for 8158 test sentences recorded during development, on 13 different confi gurations of the system. The "Rec" column indicates either simulated perfect recognition ("Text"), recognition using the Regulus-derived grammar-based language model ("GLM") or recognition using a class N-gram language model ("SLM"). The "Features" column indicates the Alterf features used.

Length	#Utts	Best GLM (G-4)			Best SLM (S-3)		
		WER	SER	SemER	WER	SER	SemER
1	3049	5.7%	3.5%	2.5%	6.3%	4.2%	3.5%
2	1589	12.0%	12.0%	8.7%	14.6%	18.4%	14.6%
3	950	7.2%	12.8%	7.2%	10.4%	15.2%	15.4%
4	1046	7.6%	14.8%	9.9%	7.7%	15.6%	14.7%
5	354	5.7%	14.4%	9.0%	6.1%	19.8%	10.8%
6	543	2.8%	11.1%	7.2%	4.1%	15.3%	9.8%
7	231	3.0%	16.0%	3.5%	4.6%	19.5%	6.5%
8	178	4.4%	14.6%	4.5%	3.6%	16.3%	5.7%
9	174	3.9%	20.1%	9.2%	4.0%	20.7%	10.3%

Table 5: Speech understanding performance, broken down by utterance length, for the best GLM-based and SLM-based versions of the system (cf. Table 4). Results are omitted for the small group of utterances of length 10 or more.