Handling Out-of-Grammar Commands in Mobile Speech Interaction Using Backoff Filler Models

Tim Paek[1], Sudeep Gandhe[2], David Maxwell Chickering[1], Yun Cheng Ju[1]
[1] Microsoft Research, One Microsoft Way, Redmond, WA 98052 USA
[2] USC Institute for Creative Technologies, 13274 Fiji Way, Marina del Rey, CA 90292, USA
{timpaek|dmax|yuncj}@microsoft.com, gandhe@usc.edu

Abstract

In command and control (C&C) speech interaction, users interact by speaking commands or asking questions typically specified in a context-free grammar (CFG). Unfortunately, users often produce out-of-grammar (OOG) commands, which can result in misunderstanding or non-understanding. We explore a simple approach to handling OOG commands that involves generating a backoff grammar from any CFG using filler models, and utilizing that grammar for recognition whenever the CFG fails. Working within the memory footprint requirements of a mobile C&C product, applying the approach yielded a 35% relative reduction in semantic error rate for OOG commands. It also improved partial recognitions for enabling clarification dialogue.

1 Introduction

In command and control (C&C) speech interaction, users interact with a system by speaking commands or asking questions. By defining a rigid syntax of possible phrases, C&C reduces the complexity of having to recognize unconstrained natural language. As such, it generally affords higher recognition accuracy, though at the cost of requiring users to learn the syntax of the interaction (Rosenfeld et al., 2001). To lessen the burden on users, C&C grammars are authored in an iterative fashion so as to broaden the coverage of likely expressions for commands, while remaining relatively simple for faster performance. Nevertheless, users can, and often still do, produce OOG commands. They may neglect to read the instructions, or forget the valid expressions. They may mistakenly believe that recognition is more robust than it really is, or take too long to articulate the right words. Whatever the reason, OOG commands can engender misunderstanding (i.e., recognition of the wrong command) or non-understanding (i.e., no recognition), and aggravate users who otherwise might not realize that their commands were OOG.

In this paper, we explore a simple approach to handling OOG commands, designed specifically to meet the memory footprint requirements of a C&C product for mobile devices. This paper is divided into three sections. First, we provide background on the C&C product and discuss the different types of OOG commands that occur with personal mobile devices. Second, we explain the details of the approach and how we applied it to the product domain. Finally, we evaluate the approach on data collected from real users, and discuss possible drawbacks.

2 Mobile C&C

With the introduction of voice dialing on mobile devices, C&C speech interaction hit the wider consumer market, albeit with rudimentary pattern recognition. Although C&C has been commonplace in telephony and accessibility for many years, only recently have mobile devices have the memory and processing capacity to support not only automatic speech recognition (ASR), but a whole range of multimedia functionalities that can be controlled with speech. Leveraging this newfound computational capacity is *Voice Command*, a C&C application for high-end mobile devices that allows users to look up contacts, place phone calls, retrieve appointments, obtain device status information, control multimedia and launch applications. It uses an embedded, speaker-independent recognizer and operates on 16 bit, 16 kHz, Mono audio.

Proceedings of SPEECHGRAM 2007, pages 33–40,
Prague, Czech Republic, June 2007.

OOG commands pose a serious threat to the usability of *Voice Command*. Many mobile users expect the product to "just work" without having to read the manual. So, if they should say "*Dial Bob*", when the proper syntax for making a phone call is *Call {Name}*, the utterance will likely be misrecognized or dropped as a false recognition. If this happens enough, users may abandon the product, concluding that it or ASR in general, does not work.

2.1 OOG frequency

Given that C&C speech interaction is typically geared towards a relatively small number of words per utterance, an important question is, how often do OOG commands really occur in C&C? In *Project54* (Kun & Turner, 2005), a C&C application for retrieving police information in patrol cars, voice commands failed on average 15% of the time, roughly 63% of which were due to human error. Of that amount, roughly 54% were from extraneous words not found in the grammar, 12% from segmentation errors, and the rest from speaking commands that were not active.

To examine whether OOG commands might be as frequent on personal mobile devices, we collected over 9700 commands of roughly 1 to 3 seconds each from 204 real users of *Voice Command*, which were recorded as sound (.wav) files. We also logged all device data such as contact entries and media items. All sound files were transcribed by a paid professional transcription service. We ignored all transcriptions that did not have an associated command; the majority of such cases came from accidental pressing of the push-to-talk button. Furthermore, we focused on user-initiated commands, during which time the active grammar had the highest perplexity, instead of yes-no responses and clarification dialogue. This left 5061 transcribed utterances.

2.2 Emulation method

With the data transcribed, we first needed a method to distinguish between In-Grammar (ING) and OOG utterances. We developed a simulation environment built around a desktop version of the embedded recognizer which could load the same *Voice Command* grammars and update them with user device data, such as contact entries, for each sound file. It is important to note the desktop version was not the engine that is commercially shipped and optimized for particular devices, but rather one that serves testing and research purposes. The environment could not only recognize sound files, but also parse string input using the dynamically updated grammars as if that were the recognized result. We utilized the latter to emulate recognition of all transcribed utterances for *Voice Command*. If the parse succeeded, we labeled the utterance *ING*, otherwise it was labeled *OOG*.

Overall, we found that slightly more than one out of every four (1361 or 26.9%) transcribed utterances were OOG. We provide a complete breakdown of OOG types, including extraneous words and segmentation errors similar to Project54, in the next section. It is important to keep in mind that being OOG by emulation does not necessarily entail that the recognizer will fail on the actual sound file. For example, if a user states *"Call Bob at mobile phone"* when the word "phone" is OOG, the recognizer will still perform well. The OOG percentage for *Voice Command* may also reflect the high perplexity of the name-dialing task. Users had anywhere from 5 to over 2000 contacts, each of which could be expressed in multiple ways (e.g., first name, first name + last name, prefix + last name, etc.). In summary, our empirical analysis of the data suggests that OOG utterances for mobile C&C on personal devices can indeed occur on a frequent basis, and as such, are worth handling.

2.3 OOG type

In order to explore how we might handle different types of OOG commands, we classified them according to functional anatomy and basic edit operations. With respect to the former, a C&C utterance consists of three functional components:

1. **Slot**: A dynamically adjustable list representing a semantic argument, such as *{Contact}* or *{Date}*, where the value of the argument is typically one of the list members.
2. **Keyword**: A word or phrase that uniquely identifies a semantic predicate, such as *Call* or *Battery*, where the predicate corresponds in a one-to-one mapping to a type of command.
3. **Carrier Text**: A word or phrase that is designed to facilitate naturalistic expression of commands and carries no attached semantic content, such as "What is" or "Tell me".

OOGType	% Total	Description	Examples
Insertion	14.2%	adding a non-keyword, non-slot word	call britney porter on mobile phone ["phone" is superfluous]
Deletion	3.1%	deleting a non-keyword, non-slot word	my next appointments ["what are" missing]
Substitution	2.5%	replacing a non-keyword, non-slot word	where is my next appointment ["where" is not supported]
Segmentation	8.2%	incomplete utterance	show, call, start
Keyword Substitution	4.6%	replacing a keyword	call 8 8 2 8 0 8 0 ["dial" is keyword] , dial john horton ["call" is keyword]
Keyword Segmentation	0.1%	incomplete keyword	what are my appoint
Keyword Deletion	2.2%	deleting the keyword	marsha porter at home ["call" missing]
Slot Substitution	0.4%	replacing slot words	call executive 5 on desk ["desk" is not slot value]
Slot Segmentation	0.9%	incomplete slot	call alexander woods on mob
Slot Deletion	1.0%	deleted slot	call tracy morey at
Disfluencies	1.8%	disfluencies - mostly repetitions	start expense start microsoft excel
Order Rearrangement	0.6%	changing the order of words within a keyword	what meeting is next [Should be "what is my next meeting"]
Noise	0.7%	non primary speaker	oregon state home coming call brandon jones on mobile phone
OOG Slot	59.8%	The slot associated with this utterance is out of domain	Show Rich Lowry ["Richard" is contact entry] , dial 0 2 1 6 [Needs > 7 digits]

Table 1. Different OOG command types and their relative frequencies for the Voice Command product. The bracketed text in the "Examples" column explicates the cause of the error

For example, in the command *"Call Bob at mobile"*, the word "Call" is the keyword, "Bob" and "mobile" are slots, and "at" is a carrier word.

If we were to convert an ING command to match an OOG command, we could perform a series of edit operations: *substitution*, *deletion*, and *insertion*. For classifying OOG commands, substitution implies the use of an unexpected word, deletion implies the absence of an expected word, and insertion implies the addition of a superfluous word.

Starting with both functional anatomy and edit operations for classification, Table 1 displays the different types of OOG commands we labeled along with their relative frequencies. Because more than one label might apply to an utterance, we first looked to the slot for an OOG type label, then keyword, then everything else.

The most frequent OOG type, at about 60%, was *OOG Slot*, which referred to slot values that did not exist in the grammar. The majority of these cases came from two sources: 1) contact entries that users thought existed but did not – sometimes they did exist, but not in any normalized form (e.g., "Rich" for "Richard"), and 2) mislabeling of mostly foreign names by transcribers. Although we tried to correct as many names as we could, given the large contact lists that many users had, this proved to be quite challenging.

The second most frequent OOG type was *Insertion* at about 14%. The majority of these insertions were single words. Note that similar to Project54, segmentation errors occurred quite often at about 9%, when the different segmentation types are added together.

3 Backoff Approach

Having identified the different types of OOG commands, we needed to devise an approach for handling them that satisfied the requirements of *Voice Command* for supporting C&C on mobile devices.

3.1 Mobile requirements

For memory footprint, the *Voice Command* team specified that our approach should operate with less than 100KB of ROM and 1MB of RAM. Furthermore, the approach could not require changes

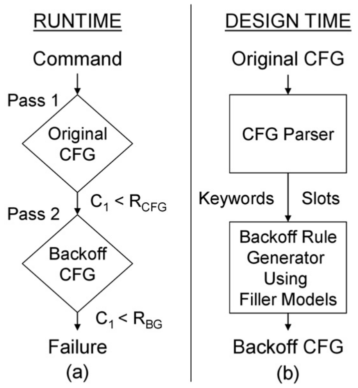

Figure 1. (a) A two-pass approach which leverages a base CFG for ING recognition and a backoff grammar for failed utterances. (b) Design time procedure for generating a backoff grammar

to the existing embedded Speech API (SAPI). Because the team also wanted to extend the functionality of *Voice Command* to new domains, we could not assume that we would have any data for training models. Although statistical language models (SLM) offer greater robustness to variations in phrasing than fixed grammars (Rosenfeld, 2000), the above requirements essentially prohibited them. So, we instead focused on extending the use of the base grammar, which for *Voice Command* was a context-free grammar (CFG): a formal specification of rules allowing for embedded recursion that defines the set of possible phrases (Manning & Schűtze, 1999).

Despite the manual effort that CFGs often require, they are widely prevalent in industry (Knight et al., 2001) for several reasons. First, they are easy for designers to understand and author. Second, they are easy to modify; new phrases can be added and immediately recognized with little effort. And third, they produce transparent semantics without requiring a separate natural language understanding component; semantic properties can be attached to CFG rules and assigned during recognition. By focusing on CFGs, our approach allows industry designers who are more accustomed to fixed grammars to continue using their skill set, while hopefully improving the handling of utterances that fall outside of their grammar.

3.2 Leveraging a backoff grammar

As long as utterances remain ING, a CFG affords fast and accurate recognition, especially because engines are often tuned to optimize C&C recognition. For example, in comparing recognition performance in a statistical and a CFG-based recognizer for the same domain, Knight et al. (2001) found that the CFG outperformed the SLM. In order to exploit the optimization of the engine for C&C utterances that are ING, we decided to utilize a two-pass approach where each command is initially submitted to the base CFG. If the confidence score of the top recognition C_1 falls below a rejection threshold R_{CFG}, or if the recognizer declares a false recognition (based on internal engine features), then the audio stream is passed to a backoff grammar which then attempts to recognize the command. If the backoff grammar fails to recognize the command, or the top recognition falls again below a rejection threshold R_{BG}, then users experience the same outcome as they normally would otherwise, except with a longer delay. Figure 1(a) summarizes the approach.

In order to generate the backoff grammar and still stay within the required memory bounds of *Voice Command*, we explored the use of the built-in filler or garbage model, which is a context-independent, acoustic phone loop. Expressed in the syntax as "...", filler models capture phones in whatever context they are placed. The functional anatomy of a C&C utterance, as explained in Section 2.3, sheds light on where to place them: before and/or after keywords and/or slots. As shown Figure 1(b), to construct a backoff grammar from a CFG during design time, we simply parse each CFG rule for keywords and slots, remove all carrier phrases, and insert filler models before and/or after the keywords and/or slots. Although it is straightforward to automatically identify keywords (words that uniquely map to a CFG rule) and slots (lists with semantic properties), developers may want to edit the generated backoff grammar for any keywords and slots they wish to exclude; for example, in cases where more than one keyword is found for a CFG rule.

For both slots and keywords, we could employ any number of different patterns for placing the filler models, if any. Table 2 displays some of the patterns in SAPI 5 format, which is an XML format where question marks indicate optional use. Although the Table is for keywords, the same patterns apply for slots. As shown in k_4, even the functional constituent itself can be optional. Furthermore, alternate lists of patterns can be composed, as in k_n. Depending on the number and type

Scheme	Keyword Pattern
k_1	<keyword/>
k_2	(…)? <keyword>
k_3	(…)? <keyword/> (…)?
k_4	(…)? <keyword/>? (…)?
k_n	<list> (…)? <keyword/>? (…)? (…) </list>

Table 2. Possible patterns in SAPI 5 XML format for placing the filler model, which appears as "...".Question marks indicate optional use.

of functional constituents for a CFG rule, backoff rules can be constructed by adjoining patterns for each constituent. We address the situation when a backoff rule corresponds to multiple CFG rules in Section 3.4.

3.3 Domain feasibility

Because every C&C utterance can be characterized by its functional constituents, the backoff filler approach generically applies to C&C domains, regardless of the actual keywords and slots. But the question remains, is this generic approach feasible for handling the different OOG types for *Voice Command* discussed in Section 2.3?

The filler model is clearly suited for *Insertions*, which are the second most frequent OOG type, because it would capture the additional phones. However, the most frequent OOG type, *OOG Slot*, cannot be handled by the backoff approach. That requires the developer to write better code for proper name normalization (e.g, "Rich" from "Richard") as well as breaking down the slot value into further components for better partial matching of names. Because new C&C domains may not utilize name slots, we decided to treat improving name recognition as separate research. Fortunately, opportunity for applying the backoff filler approach to *OOG Slot* types still exists.

3.4 Clarification of partial recognitions

As researchers have observed, OOG words contribute to increased word-error rates (Bazzi & Glass, 2000) and degrade the recognition performance of surrounding ING words (Gorrell, 2003). Hence, even if a keyword surrounding an OOG slot is recognized, its confidence score and the overall phrase confidence score will often be degraded. This is in some ways an unfortunate by-product of confidence annotation, which might be circumvented if SAPI exposed word lattice probabilities. Because SAPI does not, we can instead generate *partial backoff rules* that comprise only a subset of the functional constituents of a CFG rule. For example, if a CFG rule contains both a keyword and slot, then we can generate a partial backoff rule with just one or the other surrounded by filler models. Using partial backoff rules prevents degradation of confidence scores for ING constituents and improves partial recognitions, as we show in Section 4. Partial backoff rules not only handle *OOG Slot* commands where, for example, the name slot is not recognized, but also many types of segmentation, deletion and substitution commands as well.

Following prior research (Gorrell et al., 2002; Hockey et al., 2003), we sought to improve partial recognitions so that the system could provide feedback to users on what was recognized, and to encourage them to stay within the C&C syntax. Clarification dialogue with implicit instruction of the syntax might proceed as follows: If a partial recognition only corresponded to one CFG rule, then the system could assume the semantics of that rule and remind the user of the proper syntax. On the other hand, if a partial recognition corresponded to more than one rule, then a disambiguation dialogue could relate the proper syntax for the choices. For example, suppose a user says *"Telephone Bob"*, using the OOG word "Telephone". Although the original CFG would most likely misrecognize or even drop this command, our approach would obtain a partial recognition with higher confidence score for the contact slot. If only one CFG rule contained the slot, then the system could engage in the confirmation, *"Did you mean to say, call Bob?"* On the other hand, if more than one CFG rule contained the slot, then the system could engage in a disambiguation dialogue, such as *"I heard 'Bob'. You can either call or show Bob"*. Either way, the user is exposed to and implicitly taught the proper C&C syntax.

3.5 Related research

In related research, several researchers have investigated using both a CFG and a domain-trained SLM simultaneously for recognition (Gorrell et al., 2002; Hockey et al., 2003). To finesse the performance of a CFG, Gorrell (2003) advocated a two-pass approach where an SLM trained on CFG data (and slightly augmented) is utilized as a backoff grammar. However, only the performance of the SLM on a binary OOG classification task was evaluated and not the two-pass approach itself. In designing a multimodal language acquisition system, Dusan & Flanagan (2002) developed a two-pass approach where they utilized a dictation n-gram as a backoff grammar and added words recognized in the second pass into the base CFG. Unfortunately, they only evaluated the general usability of their architecture.

Because of the requirements outlined in Section 3.1, we have focused our efforts on generating a backoff grammar from the original CFG, taking advantage of functional anatomy and filler models. The approach is agnostic about what the actual filler model is, and as such, the built-in phone loop can easily be replaced by word-level (e.g., Yu et al., 2006) and sub-word level filler models (e.g., Liu et al., 2005). In fact, we did explore the word-level filler model, though so far we have not been able to meet the footprint requirements. We are currently investigating phone-based filler models.

Outside of recognition with a CFG, researchers have pursued methods that directly model OOG words as sub-word units in the recognition search space of a finite state transducer (FST) (Bazzi & Glass, 2000). OOG words can also be dynamically incorporated into the FST (Chung et al., 2004). Because this line of research depends on entirely different engine architecture, we could not apply the techniques.

4 Evaluation

In C&C speech interaction, what matters most is not word-error rate, but semantic accuracy and task completion. Because task completion is difficult to evaluate without collecting new data, we evaluated the semantic accuracy of the two-pass approach against the baseline of using just the CFG on the data we collected from real users, as discussed in Section 2.1. Furthermore, because partial recognitions can ultimately result in a successful dialogue, we carried out separate evaluations for the functional constituents of a command (i.e., keyword and slot) as well as the complete command (keyword + slot). For *Voice Command*, no command contained more than one slot, and because the vast majority of single slot commands were commands to either call or show a contact entry, we focused on those two commands as a proof of concept.

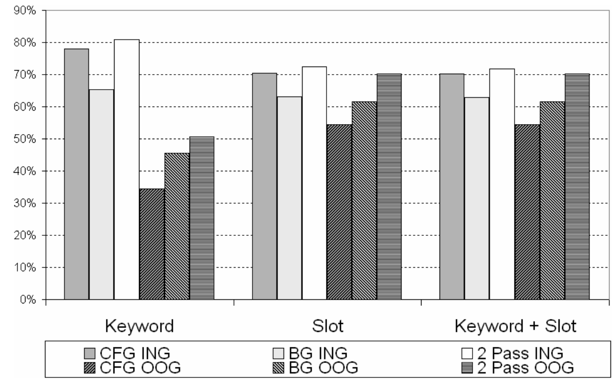

Figure 2. The semantic accuracies comparing the baseline CFG against both the BG (backoff grammar alone) and the two-pass approach (CFG + Backoff) separated into functional constituent groups and further separated by ING and OOG commands.

For any utterance, the recognizer can either accept or reject it. If it is accepted, then the semantics of the utterance can either be correct (i.e., it matches what the user intended) or incorrect. The following metrics can now be defined:

$$precision = CA / (CA + IA) \quad (1)$$
$$recall = CA / (CA + R) \quad (2)$$
$$accuracy = CA / (CA + IA + R) \quad (3)$$

where CA denotes accepted commands that are correct, IA denotes accepted commands that are incorrect, and R denotes the number of rejected commands. Although R could be decomposed into correct and incorrect rejections, for C&C, recognition failure is essentially perceived the same way by users: that is, as a non-understanding.

4.1 Results

For every C&C command in *Voice Command*, the embedded recognizer returns either a false recognition (based on internal engine parameters) or a recognition event with a confidence score. As described in Section 3.2, if the confidence score falls below a rejection threshold R_{CFG}, then the audio stream is processed by the backoff grammar which also enforces its own threshold R_{BG}. The R_{CFG} for *Voice Command* was set to 45% by a proprietary tuning procedure for optimizing acoustic word-error rate. For utterances that exceeded R_{CFG}, 84.2% of them were ING and 15.8% OOG. For utterances below R_{CFG}, 48.5%

		CFG	2-PASS	RER
Keyword	Prec	85.0%	79.0%	-39.7%
	Recall	36.8%	58.6%	34.5%
	Acc	**34.5%**	**50.7%**	**24.7%**
Slot	Prec	89.3%	88.2%	-10.3%
	Recall	58.1%	77.6%	46.5%
	Acc	**54.4%**	**70.3%**	**34.9%**
Keyword + Slot	Prec	89.3%	88.2%	-10.3%
	Recall	58.1%	77.6%	46.5%
	Acc	**54.4%**	**70.3%**	**34.9%**

Table 3. Relative reductions in semantic error rate, or Relative Error Reduction (RER) for OOG commands grouped by keyword, slot and keyword + slot evaluations. "2-PASS" denotes the two-pass approach.

were ING and 51.5% OOG. Because a considerable number of utterances may be ING in the second pass, as it was in our case, R_{BG} requires tuning as well. Instead of using a development dataset to tune R_{BG}, we decided to evaluate our approach on the entire data with R_{BG} set to the same proprietary threshold as R_{CFG}. In post-hoc analyses, this policy of setting the two thresholds equal and reverting to the CFG recognition if the backoff confidence score falls below R_{BG} achieved results comparable to optimizing the thresholds.

Figure 2 displays semantic accuracies separated by ING and OOG commands. Keyword evaluations comprised 3700 ING and 1361 OOG commands. Slot and keyword + slot evaluations comprised 2111 ING and 138 OOG commands. Overall, the two-pass approach was significantly higher in semantic accuracy than the baseline CFG, using McNemar's test (*$p<0.001$*). Not surprisingly, the largest gains were with OOG commands. Notice that for partial recognitions (i.e., keyword or slot only), the approach was able to improve accuracy, which with further clarification dialogue, could result in task completions. Interestingly, the approach performed the same for keyword + slot as it did for slot, which suggests that getting the slot correct is crucial to recognizing surrounding keywords. Despite the high percentage of OOG Slots, slot accuracy still increased due to better handling of other OOG types such as deletions, insertions and substitutions.

Finally, as a comparison, for the keyword + slot task, an upper bound of 74.3% ± 1.1% (10-fold cross-validated standard error) overall semantic accuracy was achieved using a small footprint statistical language modeling technique that re-ranked CFG results (Paek & Chickering, 2007), though the comparison is not completely fair given that the technique was focused on predictive language modeling and not on explicitly handling OOG utterances. Also note that in all cases, the backoff grammar alone performed worse than either the CFG or the two-pass approach.

Table 3 provides a more detailed view of the results for the just OOG commands as well as the relative reductions in semantic error rate (RER). Notice that the approach increases recall, which signifies less non-understandings. However, this comes at the price of a small increase in misunderstandings, as seen in the decrease in precision. Overall, the best reduction in semantic error rate achieved by the approach was about 35%.

Decomposing RER by OOG types, we found that for keyword evaluations, the biggest improvement (52% RER), came about for *Deletion* types, or commands with missing carrier words. This makes sense because the backoff grammar only cares about the keyword. For slot and keyword + slot evaluations, *Insertion* types maintained the biggest improvement at 38% RER.

Note that the results presented are those obtained without tuning. If application developers wanted to find an optimal operating point, they would need to decide what is more important for their application: precision or recall, and adjust the thresholds until they reach acceptable levels of performance. Ideally, these levels should accord with what real users of the application would accept.

4.2 Efficiency

Given that the approach was aimed at satisfying the mobile requirements stated in Section 3.1, which it did, we also compared the processing time it takes to arrive at a recognition or false recognition between the CFG alone and the two-pass approach. Because of the filler models, the backoff grammar is a more relaxed version of CFG with a larger search space, and as such, takes slightly more processing time. The average processing time for the CFG in our simulation environment was about 395 milliseconds, whereas the average processing time for the two passes was about 986 milliseconds. Hence, when the backoff grammar is used, the total computation time is approximately 2.5 times that of a single pass alone. In our experiments, a total of 1570 commands (i.e. 31%) required the two passes, while 3491 of them were accepted after a single CFG pass.

4.3 Drawbacks

In exploring the backoff filler approach, we encountered a few drawbacks that are worth considering when applying this approach to other domains. The first issue dealt with false positives. In the data collection for *Voice Command*, a total of 288 utterances contained no discernable speech. If these were included in the data set, they would amount to about 5% of all utterances. As mentioned previously, these were mostly cases when the push-to-talk button was accidentally triggered. When we evaluated the approach on these utterances, we found that the CFG accepted 36 or roughly 13% of them, while the proposed approach accepted 115 or roughly 40% of them. For our domain, this problem can be avoided by instructing users to lock their devices when not in use to prevent spurious initiations. For other C&C domains where unintentional command initiations occur frequently, this may be a serious concern, though we suspect that users will be more forgiving of accidental errors than real errors.

Another drawback dealt with generating the backoff grammar. As we discussed in Section 3.2, various patterns for placing filler models can be utilized. Although we did explore the possibility that perhaps certain patterns might generalize across domains, we found that it was better to hand-craft patterns to the application. For *Voice Command*, we used the k_n pattern specified in Table 2 for keywords, and the identical s_n pattern for slots because they proved to be best suited to the product grammars in pre-trial experiments.

5 Conclusion & Future Direction

In this paper, we classified the different types of OOG commands that might occur in a mobile C&C application, and presented a simple two-pass approach for handling them that leverages the base CFG for ING recognition and a backoff grammar OOG recognition. The backoff grammar is generated from the original CFG by surrounding keywords and/or slots with filler models. Operating within the memory footprint requirements of a mobile C&C product, the approach yielded a 35% relative reduction in semantic error rate for OOG commands, and improved partial recognitions, which can facilitate clarification dialogue.

We are now exploring small footprint, phone-based filler models. Another avenue for future research is to further investigate optimal policies for deciding when to pass to the backoff grammar and when to use the backoff grammar recognition.

References

I. Bazzi & J. Glass. 2000. *Modeling out-of-vocabulary words for robust speech recognition.* In Proc. ICSLP.

G. Chung, S. Seneff, C.Wang, & I. Hetherington. 2004. *A dynamic vocabulary spoken dialogue interface.* In Proc. ICSLP.

S. Dusan & J. Flanagan. 2002. *Adaptive dialog based upon multimodal language acquisition.* In Proc. ICMI.

G. Gorrell, I. Lewin, & M. Rayner. 2002. *Adding intelligent help to mixed initiative spoken dialogue systems.* In Proc. ICSLP.

G. Gorrell. 2003. *Using statistical language modeling to identify new vocabulary in a grammar-based speech recognition system.* In Proc. Eurospeech.

B. Hockey, O. Lemon, E. Campana, L. Hiatt, G. Aist, J. Hieronymus, A. Gruenstein, & J. Dowding. 2003. *Targeted help for spoken dialogue systems: intelligent feedback improves naive users' performance.* In Proc. EACL, pp. 147–154.

S. Knight, G. Gorrell, M. Rayner, D. Milward, R. Koeling, & I. Lewin. 2001. *Comparing grammar-based and robust approaches to speech understanding: A case study.* In Proc. Eurospeech.

A. Kun & L. Turner. 2005. *Evaluating the project54 speech user interface.* In Proc. Pervasive.

P. Liu, Y. Tian, J. Zhou, & F. Soong. 2005. *Background model based posterior probability for measuring confidence.* In Proc. Interspeech.

C.D. Manning & H. Sch¨utze. 1999. *Foundations of Statistical Natural Language Processing.* MIT Press, Cambridge,Massachusetts.

Paek, T. & Chickering, D. 2007. *Improving command and control speech recognition: Using predictive user models for language modeling*. UMUAI, 17(1):93-117.

Rosenfeld, R. 2000. Two decades of statistical language modeling: Where do we go from here? In Proc. of the IEEE, 88(8): 1270–1278.

R. Rosenfeld, D. Olsen, & A. Rudnicky. 2001. *Universal speech interfaces.* Interactions, 8(6):34–44.

D. Yu, Y.C. Ju, Y. Wang, & A. Acero. 2006. *N-gram based filler model for robust grammar authoring.* In Proc. ICASSP.