A Dynamic Vocabulary Spoken Dialogue Interface

Grace Chung†, Stephanie Seneff‡, Chao Wang‡, Lee Hetherington‡

†Corporation for National Research Initiatives
1895 Preston White Drive, Suite 100, Reston, VA 22209
gchung@cnri.reston.va.us

‡MIT Computer Science and Artificial Intelligence Laboratory
The Stata Center, 32 Vassar Street, Cambridge, MA 02139
{seneff,wangc,ilh}@csail.mit.edu

Abstract

Mixed-initiative spoken dialogue systems today generally allow users to query with a fixed vocabulary and grammar that is determined prior to run-time. This paper presents a spoken dialogue interface enhanced with a dynamic vocabulary capability. One or more word classes can be made dynamic in the speech recognizer and natural language (NL) grammar so that a context-specific vocabulary subset can be incorporated on-the-fly as the context of the dialogue changes, at each dialogue turn. Described is a restaurant information domain which continually updates the restaurant name class, given the dialogue context. We examine progress made to the speech recognizer, natural language parser and dialogue manager in order to support the dynamic vocabulary capability, and present preliminary experimental results conducted from simulated dialogues.

1. Introduction

Mixed-initiative spoken dialogue systems today are generally restricted to a fixed vocabulary, determined prior to run-time. This mandates that the lexicon and grammar must anticipate in advance all entities that a user might refer to. If database contents shift, necessarily, all the linguistic structures have to be re-compiled to reflect the latest updates in the content sources.

This paper presents a spoken dialogue interface enhanced with a dynamic vocabulary capability. One or more word classes can be made dynamic in the speech recognizer and natural language (NL) grammar so that a context-specific vocabulary subset can be incorporated on-the-fly as the context of the dialogue changes, at each dialogue turn.

The underlying objective of this work is to build systems that can flexibly incorporate new words from users and from dynamic information sources across the Internet. We envision a framework where, because the dialogue interface does not have complete *a priori* knowledge of all possible data items that a user might ask, it enlists an agent to seek the data entries from the Web, given the current dialogue context. Subsequently, the system updates its vocabulary and language models with the newly retrieved data subset. Part of this vision is the enabling of an open vocabulary in which the system, through various means, attempts to provide an answer even when the queried name has never been encountered before.

†The research at CNRI is sponsored in part by SPAWAR SSC-SD. The content of this paper does not necessarily reflect the position or policy of the Government, and no official endorsement should be inferred. ‡The research at MIT is supported in part by an industrial consortium supporting the MIT Oxygen Alliance.

Here, we present a system that partially implements this vision in the context of a restaurant information domain, whereby the restaurant name word class is continually updated with a subset of restaurant names, given the context accumulated in the dialogue history. One immediate advantage is that the vocabulary size for this narrow domain task is much smaller when most of the restaurant names are excluded from the vocabulary at any one time, leading to improved recognition performance. Secondly, changes in the database content via updates, such as new restaurants, do not require re-compilation of the main finite-state transducers (FSTs) in the recognition or the natural language parser. To the knowledge of the authors, this is a first mixed-initiative spoken dialogue system to demonstrate a dynamic vocabulary capability integrated in the speech recognition and understanding components, designed to update in real-time at *every* turn as the context of the dialogue changes. The framework even has the capability of augmenting the vocabulary on-the-fly during a single user query, by utilizing context information available within that same query in a second pass through the recognition search. The vocabulary can in principle be open (i.e., determined from content sources at run-time).

The remainder of this paper will describe the procedures we used to implement the dynamic vocabulary capability, and provide results for simulated user data. We first describe a speech recognizer that integrates an out-of-vocabulary (OOV) word model with a dynamic word class that is updated before each user turn. The vocabulary items in the dynamic word class are augmented with identifier tags for special handling as dynamic sequences in the NL parser. We will elaborate on the dialogue manager's role for finding the context-specific vocabulary subset given the dialogue constraints. Then, we introduce a specialized server responsible for performing FST operations to create dynamic class FSTs to be uploaded by the recognizer. We also describe the two-pass recognition procedure where the novel restaurant name is first recognized as an unknown word, and local context supplies the dynamic vocabulary options for the second pass. Finally, we present some preliminary experimental results conducted from simulated dialogue runs.

2. Dynamic Vocabulary Technology

2.1. Speech Recognizer

The speech recognizer employs technology first described in [1], which introduced an efficient technique for addressing dynamic changes to a grammar while preserving cross-word context-dependent phonological constraints. The implementation allows arbitrary dynamic components to FSTs, while preserving the cross-

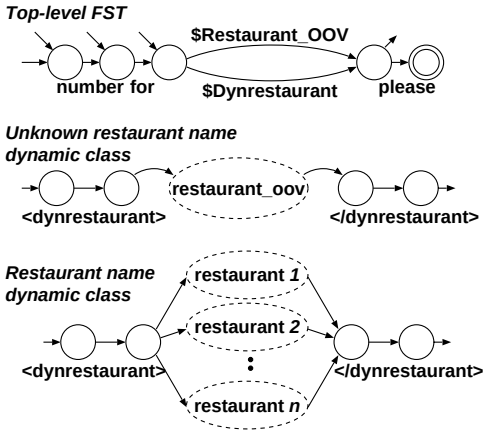

Figure 1: FSTs used in dynamic vocabulary recognition. Top: the top-level FST with dynamic splice points to the restaurant names class and unknown restaurant name model. Middle: unknown restaurant name model. Bottom: dynamic restaurant name class.

word effects. The overall FST is partially compiled prior to runtime, reducing latency.

In the restaurant information domain, the only dynamic class used thus far is a restaurant name class ($Dynrestaurant), as illustrated in Figure 1. The $Dynrestaurant class maps context-dependent phone sequences to restaurant names. The $Restaurant_OOV model contains only the dummy word "restaurant_oov," representing an unknown restaurant name, which embeds a phonetic model for generic unknown words, as described in [2]. Hence, when a user mentions a restaurant name in a query, it can be recognized as one of the n restaurants in the dynamic class or as an unknown name ("restaurant_oov").

On output, the contents of these word classes are surrounded by the labels <dynrestaurant> and </dynrestaurant>. Serving primarily as markers for the NL component (discussed later), these have no acoustic correlates, but are needed in the language model training procedure. Within the static component, the recognizer also supports a generic unknown word (<unknown>) using the same phonetic model as "restaurant_oov." Thus, the language model would give preference to the "restaurant_oov" word based on the training data.

During dialogue, the $Dynrestaurant class is continually updated by a dynamic vocabulary server (see Section 2.3). Prior to each recognition pass, the speech recognizer detects the occurrence of an updated dynamic class FST, and reloads the dynamic class before beginning the next user utterance.

2.2. NL Understanding of Open Vocabulary

The N-best output of the speech recognizer, containing the surrounding dynamic restaurant class markers, are input to TINA, the NL understanding component [3]. TINA was augmented to accommodate any sequence appearing between the markers, assigning the sequence to a "restaurant_name" semantic category. A special pre-terminal "any_sequence" category has been introduced for this purpose. Through the context-free rules, this sequence is bounded by the left and right dynamic restaurant name markers, and parses under a parent "dynamic_restaurant_name node." TINA treats the unknown word "restaurant_oov" in a similar way, passing "restaurant_name: restaurant_oov" to the dialogue manager to process accordingly.

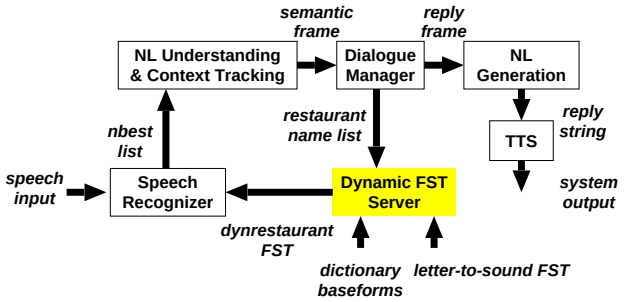

Figure 2: A schematic showing interaction of each component in the dynamic vocabulary system implementation. Highlighted is the addition of a dynamic FST server.

One issue that arose is the fact that users will often refer to restaurant names in an abbreviated form. To address this problem, aliases were generated for each restaurant using a rule-based method, proposing shortened alternatives that a user would be likely to speak instead of the restaurant's full name. Mappings from the aliases to the database entry are handled by a separate look-up in the dialogue manager.[1]

2.3. Dialogue Management and Dynamic FST Creation

A critical part of realizing the dynamic vocabulary system involves extending the dialogue management functions to trigger the switching of dynamic grammars where necessary, and the integration of operations for retrieving and updating new restaurant names and pronunciations, to be uploaded by the recognizer. These tasks are coordinated through augmentations to scripts originally implemented within the Galaxy framework [4]. A schematic of this process is illustrated in Figure 2.

A specialized dynamic FST Galaxy server has been implemented to perform the operations necessary to create the dynamic grammar (shown as the dynrestaurant FST in Figure 2) to be spliced into the static FST at recognition time. Formally, the FST structure is represented as a sequence of compositions: $R = C \circ P \circ L \circ G$, successively cascading context-dependent labels (C), phonological rules (P), phonemic baseforms (L), and language models (G). Given a list of restaurant names, the dynamic FST server will look up the dictionary for existing pronunciations or employ a letter-to-sound module [5] for unknown words. This is composed with a precomposed $C \circ P$ FST, and then saved for the recognizer to later upload. The dialogue manager, together with the hub scripting language, are central in enabling the dynamic update mechanisms to occur at the appropriate times.

At least two distinct scenarios are envisioned. One possibility is that the speaker cumulatively supplies constraints throughout the dialogue, guided by information given in the system replies, as illustrated in the first example dialogue of Table 1. For every turn, the dialogue manager filters the database on the constraints, summarizing the results in a system reply. If filtering has yielded a reasonably small set of restaurants ($n < 6$), the list of restaurant names and their plausible aliases are immediately sent to the dynamic FST server, which compiles them along with their respective baseforms into the dynamic recognition grammar. It is expected that the user is likely to query a restaurant included in the subset, particularly any that have been mentioned explicitly in the system reply.

[1] Future work will address the problem of automatic generation of such restaurant aliases.

In a second scenario, the speaker may ask about a restaurant not included in the current dynamic restaurant name grammar, as shown in the second example dialogue of Table 1. This could occur at the beginning of the dialogue, where no prior context has been given, or at any time when the user decides to introduce a new restaurant in a new location. Whenever the user has supplied sufficient context within the same utterance, a two-pass approach can be invoked, as described below:

1. The first pass recognition detects an unknown "restaurant_oov" word.
2. The attribute-value pairs, generated by the NL server, associated with the recognition output are passed to the dialogue manager, signifying an unknown restaurant query. Additional user-specified constraints from the same utterance are used to filter the database for a possible subset.
3. If a non-empty database retrieval succeeds, the dialogue manager enables the dynamic FST server to compile a new dynamic grammar with the new name subset.
4. Immediately following, Galaxy script operations trigger the speech recognizer to conduct a second Viterbi pass on the same input waveform. Because the dynamic class has been freshly updated, a reloading takes place before the second Viterbi pass. Because SUMMIT caches acoustic model scores for the whole utterance, the second pass can be very rapid, since most models have already been evaluated.
5. The second Viterbi pass may resolve the unknown restaurant into one from the existing dynamic class. The dialogue manager is then re-engaged to prepare a reply frame for answering the now fully understood query.

The above describes each step involved when a name introduced by the user can be resolved. However, under various circumstances, the unknown word may not be resolved because (1) the user has not specified other parameters to narrow the entries, (2) the user-specified constraints yield an empty database output, or (3) the second Viterbi pass persisted with selecting an unknown word instead of a known restaurant. These situations could be further complicated by recognition errors elsewhere in the utterance. In each case, our initial implementation has resorted to prompting the user for more context. Future work could address this by soliciting the names through error recovery subdialogues.

At the beginning of each dialogue, the recognizer dynamic class is empty. As a user progresses through the conversation, the dynamic grammar grows to include new restaurant names that are brought into focus over time, based on mentioned constraints (for example, neighborhoods or cuisines specified.) The restaurant names are generally flushed out between dialogues, or when a user says "clear history." In this initial implementation, the dialogue manager is connected to a static database with a fixed number of entries. However, even dynamic changes to the database itself could be supported in this framework.

3. The Restaurant Information Domain

Examples and experiments in this paper are drawn from a restaurant information system for Boston. Obtained from an on-line source, the content offers information for 982 restaurants, located in 106 cities in the Boston metropolitan area (e.g., Newton, Cambridge) and 45 neighborhoods (e.g., Back Bay, South End). Individual restaurant entries contain detailed information such as cuisine, phone number, opening hours, credit-card acceptance, price range, handicap accessibility, and menu offerings. Latitude and longitude information for each restaurant location have also been obtained. There are on average 1.23 aliases per restaurant name.

Dialogue 1	
SIM:	I would like a restaurant in The South End.
SYS:	There are 42 restaurants in The South End. Some of the options are American, and Nouvelle Cuisine. Many of them are on Columbus Avenue, and Tremont Street.
SIM:	Okay how about a cheap American restaurant on Columbus Avenue?
SYS:	There are 3 cheap American restaurants in The South End on Columbus Avenue. Your choices are Charlie's Sandwich Shoppe, Bob The Chefs, and Grill Fish.
SIM:	What is the phone number of Grill Fish?
SYS:	The phone number of Grill Fish is 617-357-1620.
Dialogue 2	
SIM:	Tell me the address for Atasca in Cambridge.
SYS:	Atasca is located at 50 Hampshire Street in Kendall Square, Cambridge.

Table 1: Two examples of simulated dialogues. The simulated inputs (SIM) are processed to produce system responses (SYS).

3.1. Experiments Using Simulation

A simulation server combined with a text generation capability, described in [6], is used to pose as a synthetic user interacting with the spoken dialogue system in real-time. The simulation framework can employ both a text-mode and a speech-mode interaction, enlisting a text-to-speech component to create synthetic user speech input. This integrated simulation module, using dialogue state information, database contents, and a rule-based recursive text generator, can produce a wide variety of user inputs. Table 1 illustrates typical simulated inputs, along with the system replies.

By running repeated iterations, thousands of dialogues have been generated both for debugging purposes, and to produce a corpus for training the recognizer language models and statistical NL grammar. To support language model training with a dynamic class, the simulator automatically generates sentences with the appropriate identifier tags. TINA will parse sequences bounded by the tags under a "dynamic_restaurant_name" category, and modify the language model training data with the dynamic $Dynrestaurant class.

In this paper, we examine system performance by creating targeted synthetic dialogues and comparing recognition error rates. In the following experiments, the performance of the dynamic vocabulary speech recognizer is compared to one which, instead of using a dynamic class, includes all the available restaurant names within a static grammar.

The speech recognizer bigram and trigram models are trained solely on simulated data. Because the test data are also generated from the simulator, they are well matched with the training data, although they are not guaranteed to have appeared in training due to the multitude of random variations generated from simulation. The recognizer was trained on over 8000 sentences, and has a static lexicon of about 1400 words. Considering both names and aliases, the dynamic word class has a maximum of around 1100 words, yielding a 2500 word vocabulary for the fixed vocabulary recognizer. The acoustic models are trained solely on telephone speech data from previously collected weather and flight information retrieval domains. For now, since the number of available restaurants is fixed, all names and aliases have been added to the dictionary, with missing baseforms pre-generated via a letter-to-sound module [5].

The synthetic input is produced by the Festival speech synthesizer [7]. To investigate system performance, two kinds of dialogues are created. In experiment I, the simulated user successively queries the system until a small subset of restaurants are provided in the system reply. The user will proceed to query an attribute (e.g., the phone number) of individual restaurants that were mentioned. The

	Full Corpus		Names Subset	
	WER	SER	WER	SER
System 1 (Full)	12.6	54.9	14.3	65.7
System 2 (Dyn)	9.5	50.8	7.5	48.3

Table 2: Recognition results for 315 utterances in experiment I.

	WER	SER	CER	Task Success Rate
System 1 (Full)	12.0	62.9	19.6	51.3
System 2 (Dyn)	14.3	68.0	24.3	52.8

Table 3: System performance for 197 utterances in experiment II.

restaurant names in the queries should have been dynamically uploaded based on the context given in preceding queries. In experiment II, we simulate scenarios where the user queries a previously unknown restaurant, while providing context within the same utterance. To simplify our experiment, the dynamic class only contains restaurant names loaded from the current context. We assume no prior contexts are given or no previously mentioned restaurant names survive in the dynamic restaurant FST.

For both experiments, the test corpus is created by running the system with the simulated user responding to the system replies on-the-fly. Results using a speech recognizer with the full vocabulary are obtained by rerunning the saved synthesized utterances.

3.2. Experiment I: Utterances with Prior Context

Table 2 displays system performance for a set of dialogues containing 315 utterances. Word error rates (WER%) and sentence error rates (SER%) are quoted for the entire corpus, and for the subset (143) which contains the proper nouns only. System 1 shows the system with a full vocabulary static grammar. System 2 shows the dynamic vocabulary results. As mentioned, the input sentences are well matched to the language model training data, and lack the effects of spontaneous speech inherent in real speech. On the other hand, some of the errors stemmed from incorrect pronunciations of the restaurant names by the synthesizer.

As expected, significant gains can be obtained by a dynamic vocabulary system because of a smaller working vocabulary at any one time during the dialogue. For the full test set, WER is reduced from 12.6% to 9.5%, and, for sentences that contain the proper names, from 14.3% to 7.5%.

3.3. Experiment II: Utterances without Prior Context

In experiment II, (Table 3), it is observed that a degradation in WER results from the two-stage procedure for eliciting the restaurant name from the user utterance. This is likely because the two passes rely on the OOV model to correctly identify the unknown word boundaries, and accurately recognize the context in the remainder of the sentence. For System 2, in 12.6% of the sentences, no OOV word is detected, and in another 13%, no restaurant name is resolved. Currently, we only add restaurants that match the context (e.g., city name) in the top-choice interpretation chosen by the NL component. In the future, we will investigate allowing all contexts in the N-best interpretations, hopefully increasing the rate at which restaurant names are correctly resolved.

Also, only when the rest of the sentence has been accurately recognized will the correct name subset be retrieved. We computed the concept error rates (CER), based on attribute-value pairs derived from the meaning representation, and the overall task success (i.e., when all the attribute-value pairs are correct). Although System 1 has better CER/WER, it does not outperform on task success. The reason is that a single mis-recognition on a context parameter generates larger errors in System 2. If the wrong city is recognized, there is no chance for recognizing the correct restaurant in the second pass, whereas in some cases System 1 ascertains the right name anyway. But as the context is wrong (i.e., wrong city) for System 1, it does not achieve task success.

4. Discussion

At a preliminary level, the simulation assessments confirm our hypothesis that dynamically switching vocabulary subsets in lieu of using the entire vocabulary set of proper names can reduce the lexical confusions in the recognizer, at least for a limited set of scenarios. We have shown improvement in recognition performance for some kinds of queries, but realize that real user data is needed for further investigations. However, our methodology, in relying heavily on simulations for development and testing, does allow assessments under controlled conditions.

The dynamic vocabulary implementation is very fast and efficient; thousands of simulated runs have demonstrated that the dynamic FST construction and updating can be performed without latency. Furthermore, we postulate that this framework would be more robust to performance degradation due to lexical confusions as the size of the restaurant database increases.

5. Future Work

The system described in this paper has recently been launched and real-user data collection is already under way. This will be followed by a full-scale system evaluation. While the current system is evaluated on a fixed database of restaurants, the framework is able to fully support a dynamically decoupled database in the back-end. A future application will be a system that searches the Web in real-time to look for data that satisfy user criteria.

In the future, the dynamic vocabulary capability will be expanded with clarification subdialogues to enable users to repeat, or speak and spell words not previously known to the recognizer [8].

Thus far, the system has been trained on simulated data alone. As more real user data accumulate, evaluation will compare systems trained on real user data with a mix of simulated and real data or simulated data alone.

6. References

[1] Schalkwyk, J., Hetherington, I. L., and Story, E., "Speech Recognition with Dynamic Grammars Using Finite State Transducers," *Proc. Eurospeech,* Geneva, Switzerland, 2003.

[2] Bazzi, I., and Glass, J., "Learning Units for Domain-Independent Out-of-Vocabulary Word Modelling," *Proc. Eurospeech,* Aalborg, Denmark, 2001.

[3] Seneff, S., Wang, C., and Hazen, T. J., "TINA: A natural language system for spoken language applications," *Computational Linguistics,* vol. 18, no. 1, pp. 61-86, March 1992.

[4] Seneff, S., et al., "Galaxy-II: A reference architecture for conversational system development," *Proc. ICSLP,* Sydney, Australia, 1998.

[5] Chug, G., Wang, C., Seneff, S., Filisko, E., and Tang, M., "Combining Linguistic Knowledge and Acoustic Information in Automatic Pronunciation Lexicon Generation," submitted to *ICSLP,* 2004.

[6] Chung, G., "Developing a Flexible Spoken Dialog System Using Simulation," submitted to *ACL,* Barcelona, Spain, 2004.

[7] The Festival Speech Synthesis System, http://www.cstr.ed.ac.uk/projects/festival/.

[8] Filisko, E. and Seneff, S., "Error Detection and Recovery in Spoken Dialogue Systems," *Workshop for Spoken Language Understanding for Conversational Systems,* Boston, MA, 2004.