A Speech-In List-Out Approach to Spoken User Interfaces

Vijay Divi[1], Clifton Forlines[2], Jan Van Gemert[2], Bhiksha Raj[2], Bent Schmidt-Nielsen[2], Kent Wittenburg[2], Joseph Woelfel[2], Peter Wolf[2], Fang-Fang Zhang[2]

1. Massachussetts Institute of Tehcnology, Cambridge, MA, USA
2. Mitsubishi Electric Research Laboratories, Cambridge, MA, USA

vdivi@mit.edu, {forlines, gemert, bent, wittenburg, woelfel, wolf, fzhang}@merl.com

ABSTRACT

Spoken user interfaces are conventionally either dialogue-based or menu-based. In this paper we propose a third approach, in which the task of invoking responses from the system is treated as one of retrieval from the set of all possible responses. Unlike conventional spoken user interfaces that return a unique response to the user, the proposed interface returns a shortlist of possible responses, from which the user must make the final selection. We refer to such interfaces as Speech-In List-Out or SILO interfaces. Experiments show that SILO interfaces can be very effective, are highly robust to degraded speech recognition performance, and can impose significantly lower cognitive load on the user as compared to menu-based interfaces.

Keywords

Speech interfaces, information retrieval, spoken query

1. INTRODUCTION

Spoken input based user interfaces can be broadly categorized as dialogue-based interfaces and menu-selection interfaces. In dialogue-based interfaces, the system engages in a dialogue with the user in an attempt to determine the user's intention. In menu-based interfaces users traverse a tree of menus, each node of which presents a list of possible choices for the user. In both kinds of interfaces, the speech recognizer must typically convert the user's speech to an unambiguous text string, which is then used by the UI to determine the action it must take next. Both kinds of interfaces eventually respond to the user with a unique output.

In this paper we advocate a third, and different approach to spoken user interfaces. We note that in a majority of applications for which speech interfaces may be used, the goal of the interaction between the user and the system is to evoke a specific response from a limited set of possible responses. In our approach, we view the set of possible responses as documents in an *index*, and the task of obtaining a specific response as that of retrieval from the set. Spoken input from the user is treated as a query, which is used to retrieve a list of potentially valid responses that are displayed to the user. The user must then make the final selection from the returned list. We call spoken user interfaces based on this approach "Speech-In List-Out" or SILO interfaces.

While much has been written on text-based retrieval of spoken or multimedia documents, the topic of information retrieval (IR) using spoken queries has not been addressed much. The usual approach to spoken query based IR has been to use the recognizer as a speech-to-text convertor that generates a text string (Chang et. al, 2002; Chen et. al., 2000) or a phoneme sequence (Kupiec et. al. 1994) which is used to query the index. This approach is critically dependent on accurate recognition of the spoken query.

In our system, however, we do not require direct conversion of the spoken input to an unambiguous text string. Instead, the spoken query is converted to a probabilistically scored "bag of words" derived from the entire hypothesis space of the recognizer, that serves as the query to the index. This system is able to perform effectively even when the actual text string output by the recognizer is erroneous.

The proposed SILO approach has several advantages over dialogue-based or menu-based interfaces. The latter require users to know or guess what to say at each stage in an interaction. Interactions typically follow a sequence of steps, and the allowed responses from the user vary with the state of the interaction. On the other hand, since the SILO interface essentially performs information retrieval based on the query, restrictions on the allowed language are few, if any. Additionally, the SILO interface responds in a single step to the user, without requiring repeated refinement of the request. This simplicity of operation makes the SILO interface measurably simpler to use than menu-based interfaces.

Speech recognition systems often make mistakes, especially under noisy recording conditions. Recognition errors can result in incorrect responses from user interfaces. To improve the robustness of the UI to recognition errors, dialogue and menu-based systems use various techniques such as rejection, confirmatory responses and error-correcting dialogues. SILO interfaces do not use such techniques, and are more reliant on getting the responses right in the first place. This is possible in SILO interfaces because the spoken query based IR technique used in them is inherently robust to recognition errors.

The rest of this paper is arranged as follows: in Section 2 we describe the basic operation of the SILO interface. In Section 3 we describe the spoken query based IR algorithm used by the SILO interface. In Section 4 we describe some example applications that use the SILO interface and present experiments evidence of the effectiveness of SILO. Finally in Section 5 we present our conclusions.

2. THE OPERATION OF THE SILO INTERFACE

Figure 1. demonstrates the difference between the conventional spoken user interfaces and the SILO interface.

Figure 1a. shows the typical operation of a conventional dialog or menu-based interface. The user initiates the interaction typically using a push-to-talk or press-to-talk button. Thereafter the system goes through a cycle of processing

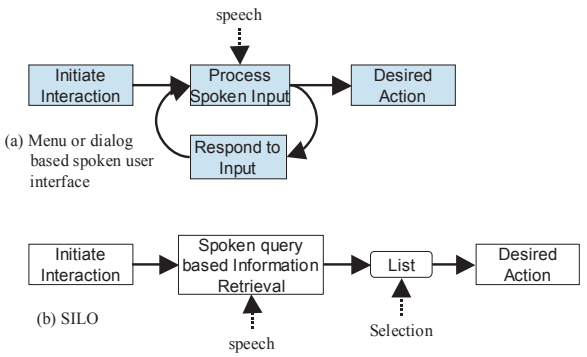

Figure 1. Operation of spoken user interfaces. (a) Conventional dialog or menu-based interfaces. (b) SILO.

the user's spoken input, and responding to it in some manner, either with a menu, or some intermediate response of a dialogue. After a number of cycles through this loop, the system eventually responds with the desired action.

Conventional interfaces are predicated on the theory that it is advantageous to obtain information about the desired final response in an incremental manner, through controlled exchanges in which the number of possible interpretations of the user's input is limited. In addition, dialogue-based interfaces also attempt to make the interaction between the user and the system conversational.

The design of the SILO interface, on the other hand, is based on the following premises: a) when the set of possible responses by the system is limited and can be enumerated, simple information retrieval techniques are sufficient to shortlist the possible final responses to the user. The shortlist may bear no resemblance to the structured lists that are derived by a menu or dialogue-based system, but will nevertheless contain the desired response provided the IR technique used is sufficiently robust. b) It is best to push the final choice of response from such a list back to the user.

Figure 1b. shows the operation of the SILO interface. The user initiates the interaction using a push-to-talk button. The system then records the user's spoken input and returns a ranked list of plausible responses based on the input. The user finally selects the desired response through a secondary selection, which may be performed using buttons, or even by voice. The entire operation is performed in two steps, one in which the system retrieves a list of choices, and the other in which the user selects from the list.

3. INFORMATION RETRIEVAL IN SILO USING SPOKEN QUERIES

At the heart of the SILO interface is the MERL SpokenQuery information retrieval engine (Wolf and Raj, 2002).

The standard approach to spoken query based IR requires the unambiguous conversion of the spoken input to text by the recognizer, and is likely to fail if the recognizer makes errors, especially in recognizing keywords that identify the desired documents. The SpokenQuery IR engine used in the SILO interface, however, does not require unambiguous speech-to-text conversion. Spoken queries need not conform to any grammar and are permitted to be free form.

Textual descriptions of the responses required from the system are stored as documents in a document index. The descriptions must uniquely identify the response. For example, in a digital music retrieval system, the documents might be the descriptive meta data associated with the music. For a command and control system, they might be a text description of possible actions to be taken by the application.

Documents are represented in this index as *word-count vectors*. Each entry in the vector represents the frequency of occurrence of a specific word in the document. This representation ignores the order in which words occur in the documents, retaining information only about their frequency of occurrence. This particular representation is related to the absence of linguistic constraints applied to spoken queries. Since queries are permitted to be free form, word order cannot be a consideration in the formation of the query, and consequently in the document index.

Spoken queries are converted to a data structure that is similar to the structures used to represent documents in the index. The query structure is a vector of normalized *expected* word counts in the spoken query. The expected count for a word is simply the statistical expectation of the number of times the word occurred in the spoken query and is computed from the entire set of word sequence hypotheses considered by the recognizer, not just the single text string finally output by the recognizer.

The recognizer represents the set of all considered hypotheses as a graph that is commonly known as a word lattice. Nodes in the graph represent words and arcs represent valid transitions between words. The scores associated with nodes and arcs are the probabilities of the corresponding words and transitions. The best-choice transcript generated by the recognizer is the most likely path through this lattice (i.e. the path with the best score).

The *a posteriori* probability of any node in the word lattice is the ratio of the total probability scores of all paths through the lattice that pass through that node, to the total probability of all paths through the lattice and can be computed through a forward-backward algorithm. Multiple nodes in the word lattice may represent the same word. The normalized expected count for a word is simply the total of the *a posteriori* probabilities of all instances of that word in the lattice. We call these "normalized" counts since the expected counts of all words sum to 1.0.

A useful feature of speech recognition systems is that the actual words spoken by the user are usually included in the recognition lattice for the utterance, often with high probability, even when they are not included in the recognizer's final output. As a result, the true query word usually have non-zero, and often relatively high, expected word counts, regardless of their inclusion in the final recognizer output.

For retrieval, the dot product of the document word-count vectors and the expected word count vectors derived from the spoken query is computed. The output of the

retrieval system is a list of the documents sorted by this value.

Figure 2 shows the overall system architecture for the SpokenQuery information retrieval system in SILO.

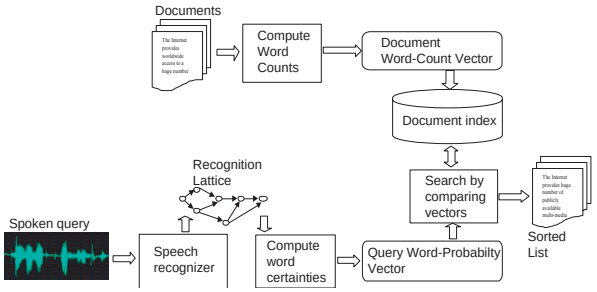

Figure 2. A block diagram representation of the SpokenQuery IR engine. The document indexer computes word count vectors for documents and stores them in an index. For retrieval, spoken queries are converted to a recognition lattice by a recognizer. Normalized expected word count vectors are computed from the lattice. Documents are ranked by the dot product of their word count vectors and the query vector, and retrieved in the order of their ranking.

The vocabulary of the speech recognizer used for query construction minimally includes all keywords in the documents. In addition to these, the recognizer may include all the rest of the words in the documents as well, or may replace them with a smaller set of garbage words. Since queries may be free form, the recognizer cannot use constrained grammars that impose strict restrictions on word order. Instead, it uses an N-gram language model that highlights the keywords and key phrases, but does not strictly disallow any particular sequence of words.

4. EXAMPLE SILO APPLICATIONS AND EXPERIMENTS

In this section we describe some applications for which we have implemented SILO interfaces, and report three experiments that evaluate different aspects of the SILO interface. The targeted aspects are: a) the effectiveness of the spoken input based IR technique used in the SILO interface, b) the effectiveness of the SILO interface and c) whether the SILO interface provides any advantage over conventional UIs.

Document retrieval with spoken queries

Since the main component of the SILO interface is the proposed SpokenQuery IR engine, our first application demonstrates the effectiveness of the proposed IR method on a conventional document retrieval task. This application is exactly the same as normal IR (e.g. Google, AltaVista, etc.) except that the user speaks instead of types. As with normal IR, the user may say any word that he/she judges to specify the information. There is no grammar to memorize. It is intended that the returned list of documents contain appropriate documents near the top, however, as with any IR engine, there is no guarantee that all returned documents will be pertinent to the query.

For the experimental evaluation, a database of 262 technical reports from our laboratory formed the document

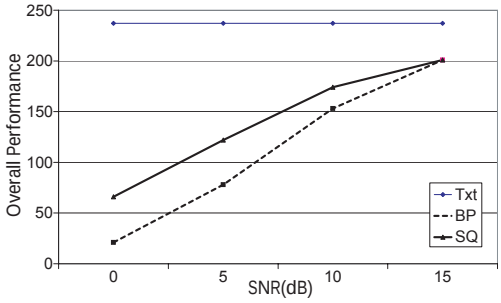

Figure 3. Average ranking of documents retrieved using i) text transcriptions of spoken queries, ii) the text output of a recognizer and iii) the proposed spoken query IR method.

index. A total of 75 spoken queries were recorded from a number of users. In order to establish ground truth, the true relevance of each of the 262 documents to each of the queries was judged by a team of humans. For each query, documents were assigned a relevance score of 0 (irrelevant), 1 (somewhat relevant), or 2 (definitely relevant). All queries were evaluated by every evaluator and their average score, scaled to lie between 1 and 10, was deemed to be the ground truth.

Figure 3 shows the average relevance of the first 30 documents retrieved, for retrieval using a text transcription of the queries, with the recognizer's best output, and the SpokenQuery IR engine respectively. The text transcription is not affected by noise. For retrieval based on spoken input, however, SpokenQuery is found to be significantly better than retrieval based on the recognizer's single best output. While performance on noisy speech is generally poor, the proposed method is observed to result in an equivalent of 5dB improvement in noise level over retrieval based on the recognizer's output.

MediaFinder: retrieving music with spoken queries

We now evaluate the effectiveness of SILO as a user interface. A UI is effective if the user is able to obtain the desired response from the system in a small number of steps. Since the SILO based UI returns lists of possible responses, the interaction may be considered successful, if the returned list contains the desired response. Further, we deem it more effective if the desired response is ranked higher in the list, since the user has to spend less time scanning the list. If the returned list does not contain the desired response, the user must repeat the interaction, and the exchange is considered a failure.

The MediaFinder application is a spoken interface for retrieving music from digital collections, and represents a good example of an application where SILO can make a significant difference in the effectiveness of the UI. Handheld mp3 players can hold thousands of songs, yet the interface provided on these devices is usually minimal, consisting of a small screen and some buttons. Users must access music by navigating a hierarchy of menus with the buttons. An effective spoken user interface can improve the usability of such devices greatly.

MP3 files contain extensive metadata (ID3) that describe their contents. In the case of music files, these

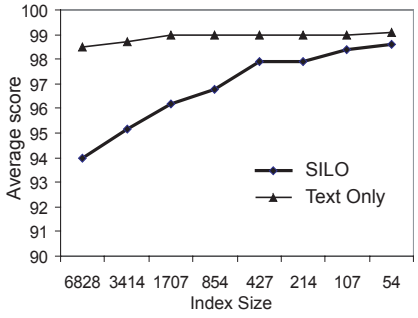

Figure 4. Average ranking of the correct response in a list of responses returned by SILO, as a function of the size of the index. A score of 0 indicates that the desired response was not returned. 100 indicates that it was returned at the top of the list.

would include the title of the album, the name of the singer or composer, and often other details such as the musical genre. This meta-data text is used to index MP3 files. To search for a song, the user speaks a description of the desired music. The description may include a combination of words from the name of the song, the artist, the album, or the genre in any order. A list of songs that match the spoken query most closely are displayed on the screen. Using up, down and select buttons, the user scrolls through the returned list, and selects the desired song. If the requested song is not in the displayed list, the speaker must repeat the query, perhaps trying different words.

We conducted two different experiments on MediaFinder: one to evaluate the ability of users to successfully obtain the desired response from the application - in this case the playback of a desired piece of music, and a second to determine if there is any advantage to the SILO interface over conventional interfaces.

In the first experiment users attempted to retrieve a desired piece of music from collections of different sizes. MediaFinder returns a list of up to 10 songs in response to a query. A score of 100 is given to the interaction if the desired song is at the top of the list. The score decreases linearly if the required response is lower in the list. If the required response is not in the returned list, the score for the interaction is 0. Figure 4 shows the average score for an interaction, as a function of the size of the collection from which songs are to be retrieved. Each point in the plot represents an average score across 100 queries from two users.

We note that the average score is greater than 90 in all cases, indicating that the desired music is not only retrieved, but is typically near the very top of the list in a majority of the cases. This shows that the SILO interface can indeed be used effectively for such tasks, and may even effectively substitute other more complex interfaces.

In the second experiment we compared the cognitive load imposed by the MediaFinder SILO interface to that imposed by a conventional graphical menu-based interface for the same task - selection of music from a digital collection in an automobile. We note here that digital music players are becoming increasingly common in automobiles, and having an effective UI that imposes minimal cognitive load is of tantamount importance in such devices.

Experiments were conducted using a simple driving simulator that mimicked two important facets of driving: steering and braking. Subjects steered, braked, and controlled the searching interfaces with a steering wheel and its gas and brake pedals. Steering was measured with a pursuit tracking task in which the subject used the wheel to closely frame a moving target. Braking was measured by recording subjects' reaction time to circles that appeared on screen at random intervals. The tests were conducted on fourteen subjects (8 male, 6 female, 18-37 years old). Four were non-native speakers and all but one were regular automobile drivers.

The study shows that a) subjects made an average of 20.7% less steering error when using the SILO interface, and b) Subjects took an average of 28.6% less time to retrieve songs using the SILO interface. Both interfaces had the same effect on braking response. The results indicate that the SILO interface does indeed impose a significantly lower cognitive load on the user, at least for such tasks.

It must be mentioned, however, that subjects were much better at both steering and braking when they did not attempt to retrieve music at all (suggesting, perhaps, that when driving an automobile, people must simply just drive).

5. DISCUSSIONS AND CONCLUSION

The SILO interface presents a different approach to user interfaces, that treats the problem of obtaining a particular *final* response from a system as one of *retrieving* one of the elements from the set of all possible responses from the system. The experiments demonstrate that the SILO interface can be effectively used in applications where the responses of the system can be enumerated, and textually described, and that it can actually be *easier* to use in some situations.

REFERENCES

1. Chang, E., Seide, F., Meng, H., Chen, Z. Shi, Y., Li, Y. A system for spoken query information retrieval on mobile devices. *IEEE Trans. on Speech and Audio Processing*, 10:8, pp. 531-541. November 2002.
2. Chen, B., Want, H.M., Lee, L.S. Retrieval of broadcast news speech in Mandarin Chinese collected in Taiwan using syllable-level statistical characteristics. *Proc. IEEE Intl. Conf. on Acoustics Speech and Signal Processing* (*ICASSP2000*). Istanbul, Turkey. 2000.
3. Kupiec, J., Kimber, D., Balasubramanian, V. Speech-based retrieval using semantic co-occurrence filtering, *Proc. Human Language Technology Conf*. 1994.
4. Wolf, P. and Raj, B. The MERL SpokenQuery Information Retrieval System: A System for Retrieving Pertinent Documents from a Spoken Query. *Proc. IEEE Conference and Multimedia Expo* (*ICME2003*). Lausanne, Switzerland. August 2002.