A Multimodal Home Entertainment Interface via a Mobile Device

Alexander Gruenstein Bo-June (Paul) Hsu James Glass Stephanie Seneff
Lee Hetherington Scott Cyphers Ibrahim Badr Chao Wang Sean Liu
MIT Computer Science and Artificial Intelligence Laboratory
32 Vassar St, Cambridge, MA 02139 USA
http://www.sls.csail.mit.edu/

Abstract

We describe a multimodal dialogue system for interacting with a home entertainment center via a mobile device. In our working prototype, users may utilize both a graphical and speech user interface to search TV listings, record and play television programs, and listen to music. The developed framework is quite generic, potentially supporting a wide variety of applications, as we demonstrate by integrating a weather forecast application. In the prototype, the mobile device serves as the locus of interaction, providing both a small touchscreen display, and speech input and output; while the TV screen features a larger, richer GUI. The system architecture is agnostic to the location of the natural language processing components: a consistent user experience is maintained regardless of whether they run on a remote server or on the device itself.

1 Introduction

People have access to large libraries of digital content both in their living rooms and on their mobile devices. Digital video recorders (DVRs) allow people to record TV programs from hundreds of channels for subsequent viewing at home—or, increasingly, on their mobile devices. Similarly, having accumulated vast libraries of digital music, people yearn for an easy way to sift through them from the comfort of their couches, in their cars, and on the go.

Mobile devices are already central to *accessing* digital media libraries while users are away from home: people listen to music or watch video recordings. Mobile devices also play an increasingly important role in *managing* digital media libraries. For instance, a web-enabled mobile phone can be used to remotely schedule TV recordings through a web site or via a custom application. Such management tasks often prove cumbersome, however, as it is challenging to browse through listings for hundreds of TV channels on a small display. Indeed, even on a large screen in the living room, browsing alphabetically, or by time and channel, for a particular show using the remote control quickly becomes unwieldy.

Speech and multimodal interfaces provide a natural means of addressing many of these challenges. It is effortless for people to say the name of a program, for instance, in order to search for existing recordings. Moreover, such a speech browsing capability is useful both in the living room and away from home. Thus, a natural way to provide speech-based control of a media library is through the user's mobile device itself.

In this paper we describe just such a prototype system. A mobile phone plays a central role in providing a multimodal, natural language interface to both a digital video recorder and a music library. Users can interact with the system—presented as a dynamic web page on the mobile browser—using the navigation keys, the stylus, or spoken natural language. In front of the TV, a much richer GUI is also available, along with support for playing video recordings and music.

In the prototype described herein, the mobile device serves as the locus of natural language interaction, whether a user is in the living room or walking down the street. Since these environments may be very different in terms of computational re-

sources and network bandwidth, it is important that the architecture allows for multiple configurations in terms of the *location* of the natural language processing components. For instance, when a device is connected to a Wi-Fi network at home, recognition latency may be reduced by performing speech and natural language processing on the home media server. Moreover, a powerful server may enable more sophisticated processing techniques, such as multipass speech recognition (Hetherington, 2005; Chung et al., 2004), for improved accuracy. In situations with reduced network connectivity, latency may be improved by performing speech recognition and natural language processing tasks on the mobile device itself. Given resource constraints, however, less detailed acoustic and language models may be required. We have developed just such a flexible architecture, with many of the natural language processing components able to run on either a server or the mobile device itself. Regardless of the configuration, a consistent user experience is maintained.

2 Related Work

Various academic researchers and commercial businesses have demonstrated speech-enabled interfaces to entertainment centers. Commercial work, such as (Fujita et al., 2003), tends to focus on constrained grammar systems, where speech input is limited to a small set of templates corresponding to menu choices. (Oh et al., 2007), however, present a dialogue system for TV control that makes use of concept spotting and statistical dialogue management to understand queries. A version of their system can run independently on low-resource devices such as PDAs; however, it has a smaller vocabulary and supports a limited set of user utterance templates.

We were also inspired by previous prototypes in which mobile devices have been used in conjunction with larger, shared displays. For instance, (Paek et al., 2004) demonstrate a framework for building such applications. The prototype we demonstrate here fits into their "Jukebox" model of interaction. Interactive workspaces, such as the one described in (Johanson et al., 2002), also demonstrate the utility of integrating mobile and large screen displays. Our prototype is a departure from both systems, however, in that it provides for spoken interactions.

3 User Experience

Our current prototype system implements the basic functionalities that one expects from a home entertainment center. Users can navigate through and record programs from the television's electronic program guide, manage recording settings, and play recorded videos. They can also browse and listen to selections from their music libraries. However, unlike existing prototypes, ours employs a smartphone with a navigation pad, touch-sensitive screen, and built-in microphone as the remote control. Figure 1 provides an overview of the graphical user interface on both the TV and mobile device.

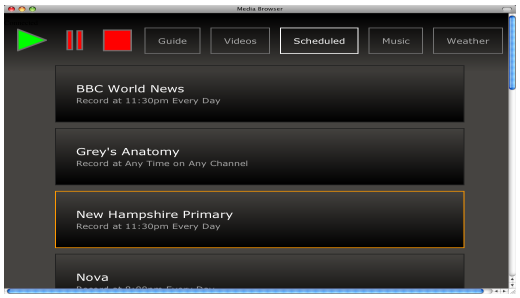
Mirroring the TV's on-screen display, the prototype system presents a reduced view on the mobile device with synchronized cursors. Users can navigate the hierarchical menu structure using the arrow keys or directly click on the target item with the stylus. While away from the living room, or when a recording is playing full screen, users can browse and manage their media libraries using only the mobile device.

While the navigation pad and stylus are great for basic navigation and control, searching for media with specific attributes, such as title, remains cumbersome. To facilitate such interactions, the current system supports spoken natural language interactions. For example, the user can press the hold-to-talk button located on the side of the mobile device and ask "What's on the *National Geographic Channel* this afternoon?" to retrieve a list of shows with the specified channel and time. The system responds with a short verbal summary "I found six entries on January seventh" and presents the resulting list on both the TV and mobile displays. The user can then browse the list using the navigation pad or press the hold-to-talk button to barge in with another command, *e.g.* "Please record the second one." Depressing the hold-to-talk button not only terminates any current spoken response, but also mutes the TV to minimize interference with speech recognition. As the previous example demonstrates, contextual information is used to resolve list position references and disambiguate commands.

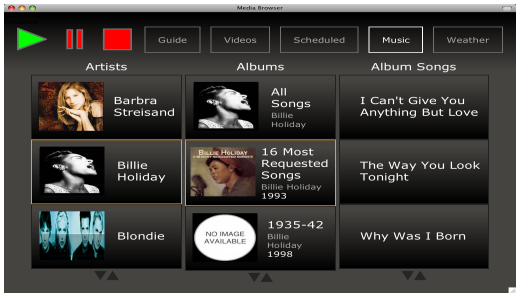
The speech interface to the user's music library works in a similar fashion. Users can search by artist, album, and song name, and then play the

Figure 1: User interface overview.

(a) Program Guide



(b) Scheduled Recordings



(c) Music Library

U: What's on tonight at seven o'clock?
S: I found seventy entries. [*Displays program guide.*]
U: Are there any *action movies* on *HBO* tomorrow?
S: I found two entries. [*Displays results.*]
U: When is *American Idol* on?
S: I found three entries. [*Displays results.*]
U: Record the first one.
S: I have added *American Idol* to the list of programs to be recorded. [*Displays recording settings.*]
U: Please record *Nova* on Wednesday night.
S: I have added *Nova* to the list of programs to be recorded. [*Displays recording settings.*]
U: Show all my scheduled recordings.
S: There are seven scheduled recordings. [*Shows list.*]
U: Delete *BBC World News*.
S: I have deleted the scheduled recording: *BBC World News*. [*Updates List.*]
U: Show my videos.
S: [*Displays list of videos.*]
U: Delete *Frasier*.
S: I have deleted the recording *Frasier*. [*Updates list.*]
U: Play *Grey's Anatomy*.
S: Playing *Grey's Anatomy*. [*Plays recording.*]
U: Stop Playing.
S: Stopping. [*Stops playback.*]
U: [*Clicks on Music tab.*]
S: [*Displays artist list.*]
U: Show albums by *Billie Holiday*.
S: I found three albums by *Billie Holiday*. [*Shows albums.*]
U: Please play *A Hard Rain's A-Gonna Fall* by *Bob Dylan*.
S: Playing *A Hard Rain's A-Gonna Fall*. [*Plays song.*]

Figure 2: Screenshots and an example interaction. Utterances are labeled with *U* for user and *S* for system.

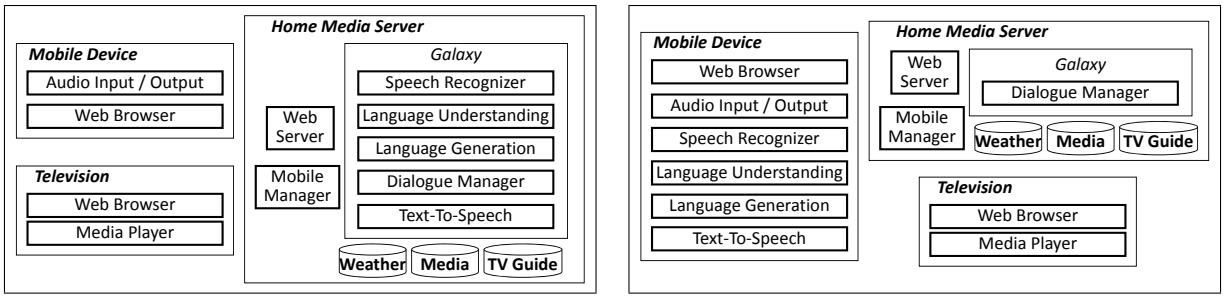


Figure 3: Two architecture diagrams. In (a) speech recognition and natural language processing occur on the server, while in (b) processing is primarily performed on the device.

songs found. To demonstrate the extensibility of the architecture, we have also integrated an existing weather information system (Zue et al., 2000), which has been previously deployed as a telephony application. Users simply click on the *Weather* tab to switch to this domain, allowing them to ask a wide range of weather queries. The system responds verbally and with a simple graphical forecast.

To create a natural user experience, we designed the multimodal interface to allow users to seamlessly switch among the different input modalities available on the mobile device. Figure 2 demonstrates an example interaction with the prototype, as well as several screenshots of the user interface. A video demonstration is available online at: `http://www.sls.csail.mit.edu/research/mobile.shtml`.

4 System Architecture

The system architecture is quite flexible with regards to the placement of the natural language processing components. Figure 3 presents two possible configurations of the system components distributed across the mobile device, home media server, and TV display. In 3(a), all speech recognition and natural language processing components reside on the server, with the mobile device acting as the microphone, speaker, display, and remote control. In 3(b), the speech recognizer, language understanding component, language generation component, and text-to-speech (TTS) synthesizer run on the mobile device. Depending on the capabilities of the mobile device and network connection, different configurations may be optimal. For instance, on a powerful device with slow network connection, recognition latency may be reduced by performing speech recognition and natural language processing on the device. On the other hand, streaming audio via a fast wireless network to the server for processing may result in improved accuracy.

In the prototype system, flexible and reusable speech recognition and natural language processing capabilities are provided via generic components developed and deployed in numerous spoken dialogue systems by our group, with the exception of an off-the-shelf speech synthesizer. Speech input from the mobile device is recognized using the landmark-based SUMMIT system (Glass, 2003). The resulting N-best hypotheses are processed by the TINA language understanding component (Seneff, 1992). Based on the resulting meaning representation, the dialogue manager (Polifroni et al., 2003) incorporates contextual information (Filisko and Seneff, 2003), and then determines an appropriate response. The response consists of an update to the graphical display, and a spoken system response which is realized via the GENESIS (Baptist and Seneff, 2000) language generation module. To support on-device processing, all the components are linked via the GALAXY framework (Seneff et al., 1998) with an additional *Mobile Manager* component responsible for coordinating the communication between the mobile device and the home media server.

In the currently deployed system, we use a mobile phone with a 624 MHz ARM processor running the Windows Mobile operating system and Opera Mobile web browser. The TV program and music databases reside on the home media server running GNU/Linux. The TV program guide data and recording capabilities are provided via MythTV,

a full-featured, open-source digital video recorder software package.[1] Daily updates to the program guide information typically contain hundreds of unique channel names and thousands of unique program names. The music library is comprised of 5,000 songs from over 80 artists and 13 major genres, indexed using the open-source text search engine Lucene.[2] Lastly, the TV display can be driven by a web browser on either the home media server or a separate computer connected to the server via a fast Ethernet connection, for high quality video streaming.

While the focus of this paper is on the natural language processing and user interface aspects of the system, our work is actually situated within a larger collaborative project at MIT that also includes simplified device configuration (Mazzola Paluska et al., 2008; Mazzola Paluska et al., 2006), transparent access to remote servers (Ford et al., 2006), and improved security.

5 Mobile Natural Language Components

Porting the implementation of the various speech recognizer and natural language processing components to mobile devices with limited computation and memory presents both a research and engineering challenge. Instead of creating a small vocabulary, fixed phrase dialogue system, we aim to support—on the mobile device—the same flexible and natural language interactions currently available on our desktop, tablet, and telephony systems; see *e.g.*, (Gruenstein et al., 2006; Seneff, 2002; Zue et al., 2000). In this section, we summarize our efforts thus far in implementing the SUMMIT speech recognizer and TINA natural language parser. Porting of the GENESIS language generation system is nearing completion, with work on the dialogue manager expected to begin shortly.

5.1 PocketSUMMIT

To significantly reduce the memory footprint and overall computation, we chose to reimplement our segment-based speech recognizer from scratch, utilizing fixed-point arithmetic, parameter quantization, and bit-packing in the binary model files. The resulting PocketSUMMIT recognizer (Hetherington, 2007) utilizes only the landmark features, initially forgoing segment features such as phonetic duration, as they introduce algorithmic complexities for relatively small word error rate (WER) improvements.

In the current system, we quantize the mean and variance of each Gaussian mixture model dimension to 5 and 3 bits, respectively. Such quantization not only results in an 8-fold reduction in model size, but also yields about a 50% speedup by enabling table lookups for Gaussian evaluations. Likewise, in the finite-state transducers (FSTs) used to represent the language model, lexical, phonological, and class diphone constraints, quantizing the FST weights and bit-packing not only compress the resulting binary model files, but also reduce the processing time with improved processor cache locality.

In the aforementioned TV, music, and weather domains with a moderate vocabulary of a few thousand words, the resulting PocketSUMMIT recognizer performs in approximately real-time on 400-600 MHz ARM processors, using a total of 2-4 MB of memory, including 1-2 MB for memory-mapped model files. Compared with equivalent non-quantized models, PocketSUMMIT achieves dramatic improvements in speed and memory while maintaining comparable WER performance.

5.2 PocketTINA

Porting the TINA natural language parser to mobile devices involved significant software engineering to reduce the memory and computational requirements of the core data structures and algorithms. TINA utilizes a best-first search that explores thousands of partial parses when processing an input utterance. To efficiently manage memory allocation given the unpredictability of pruning invalid parses (*e.g.* due to subject-verb agreement), we implemented a mark and sweep garbage collection mechanism. Combined with a more efficient implementation of the priority queue and the use of aggressive "beam" pruning, the resulting PocketTINA system can parse a 10-best recognition hypothesis list into the corresponding meaning representation in under 0.1 seconds, using about 2 MB of memory.

[1] `http://www.mythtv.org/`

[2] `http://lucene.apache.org/`

6 Rapid Dialogue System Development

Over the course of developing dialogue systems for many domains, we have built generic natural language understanding components that enable the rapid development of flexible and natural spoken dialogue systems for novel domains. Creating such prototype systems typically involves customizing the following to the target domain: recognizer language model, language understanding parser grammar, context resolution rules, dialogue management control script, and language generation rules.

Recognizer Language Model Given a new domain, we first identify a set of semantic classes which correspond to the back-end application's database, such as *artist*, *album*, and *genre*. Ideally, we would have a corpus of tagged utterances collected from real users. However, when building prototypes such as the one described here, little or no training data is usually available. Thus, we create a domain-specific context-free grammar to generate a supplemental corpus of synthetic utterances. The corpus is used to train probabilities for the natural language parsing grammar (described immediately below), which in turn is used to derive a class n-gram language model (Seneff et al., 2003).

Classes in the language model which correspond to contents of the database are marked as dynamic, and are populated at runtime from the database (Chung et al., 2004; Hetherington, 2005). Database entries are heuristically normalized into spoken forms. Pronunciations not in our 150,000 word lexicon are automatically generated (Seneff, 2007).

Parser Grammar The TINA parser uses a probabilistic context-free grammar enhanced with support for wh-movement and grammatical agreement constraints. We have developed a generic syntactic grammar by examining hundreds of thousands of utterances collected from real user interactions with various existing dialogue systems. In addition, we have developed libraries which parse and interpret common semantic classes like dates, times, and numbers. The grammar and semantic libraries provide good coverage for spoken dialogue systems in database-query domains.

To build a grammar for a new domain, a developer extends the generic syntactic grammar by augmenting it with domain-specific semantic categories and their lexical entries. A probability model which conditions each node category on its left sibling and parent is then estimated from a training corpus of utterances (Seneff et al., 2003).

At runtime, the recognizer tags the hypothesized dynamic class expansions with their class names, allowing the parser grammar to be independent of the database contents. Furthermore, each semantic class is designated either as a semantic *entity*, or as an *attribute* associated with a particular *entity*. This enables the generation of a semantic representation from the parse tree.

Dialogue Management & Language Generation Once an utterance is recognized and parsed, the meaning representation is passed to the context resolution and dialogue manager component. The context resolution module (Filisko and Seneff, 2003) applies generic and domain-specific rules to resolve anaphora and deixis, and to interpret fragments and ellipsis in context. The dialogue manager then interacts with the application back-end and database, controlled by a script customized for the domain (Polifroni et al., 2003). Finally, the GENESIS module (Baptist and Seneff, 2000) applies domain-specific rules to generate a natural language representation of the dialogue manager's response, which is sent to a speech synthesizer. The dialogue manager also sends an update to the GUI, so that, for example, the appropriate database search results are displayed.

7 Mobile Design Challenges

Dialogue systems for mobile devices present a unique set of design challenges not found in telephony and desktop applications. Here we describe some of the design choices made while developing this prototype, and discuss their tradeoffs.

7.1 Client/Server Tradeoffs

Towards supporting network-less scenarios, we have begun porting various natural language processing components to mobile platforms, as discussed in Section 5. Having efficient mobile implementations further allows the natural language processing tasks to be performed on either the mobile device or the

server. While building the prototype, we observed that the Wi-Fi network performance can often be unpredictable, resulting in erratic recognition latency that occasionally exceeds on-device recognition latency. However, utilizing the mobile processor for computationally intensive tasks rapidly drains the battery. Currently, the component architecture in the prototype system is pre-configured. A more robust implementation would dynamically adjust the configuration to optimize the tradeoffs among network use, CPU utilization, power consumption, and user-perceived latency/accuracy.

7.2 Speech User Interface

As neither open-mic nor push-to-talk with automatic endpoint detection is practical on mobile devices with limited battery life, our prototype system employs a hold-to-talk hardware button for microphone control. To guide users to speak commands only while the button is depressed, a short beep is played as an earcon both when the button is pushed and released. Since users are less likely to talk over short audio clips, the use of earcons mitigates the tendency for users to start speaking before pushing down the microphone button.

In the current system, media audio is played over the TV speakers, whereas TTS output is sent to the mobile device speakers. To reduce background noise captured from the mobile device's far-field microphone, the TV is muted while the microphone button is depressed. Unlike telephony spoken dialogue systems where the recognizer has to constantly monitor for barge-in, the use of a hold-to-talk button significantly simplifies barge-in support, while reducing power consumption.

7.3 Graphical User Interface

In addition to supporting interactive natural language dialogues via the spoken user interface, the prototype system implements a graphical user interface (GUI) on the mobile device to supplement the TV's on-screen interface. To faciliate rapid prototyping, we chose to implement both the mobile and TV GUI using web pages with AJAX (Asynchronous Javascript and XML) techniques, an approach we have leveraged in several existing multimodal dialogue systems, *e.g.* (Gruenstein et al., 2006; McGraw and Seneff, 2007). The resulting interface is largely platform-independent and allows display updates to be "pushed" to the client browser.

As many users are already familiar with the TV's on-screen interface, we chose to mirror the same interface on the mobile device and synchronize the selection cursor. However, unlike desktop GUIs, mobile devices are constrained by a small display, limited computational power, and reduced network bandwidth. Thus, both the page layout and information detail were adjusted for the mobile browser. Although AJAX is more responsive than traditional web technology, rendering large formatted pages—such as the program guide grid—is often still unacceptably slow. In the current implementation, we addressed this problem by displaying only the first section of the content and providing a "Show More" button that downloads and renders the full content. While browser-based GUIs expedite rapid prototyping, deployed systems may want to take advantage of native interfaces specific to the device for more responsive user interactions. Instead of limiting the mobile interface to reflect the TV GUI, improved usability may be obtained by designing the interface for the mobile device first and then expanding the visual content to the TV display.

7.4 Client/Server Communication

In the current prototype, communication between the mobile device and the media server consists of AJAX HTTP and XML-RPC requests. To enable server-side "push" updates, the client periodically pings the server for messages. While such an implementation provides a responsive user interface, it quickly drains the battery and is not robust to network outages resulting from the device being moved or switching to power-saving mode. Reestablishing connection with the server further introduces latency. In future implementations, we would like to examine the use of Bluetooth for lower power consumption, and infrared for immediate response to common controls and basic navigation.

8 Conclusions & Future Work

We have presented a prototype system that demonstrates the feasibility of deploying a multimodal, natural language interface on a mobile device for browsing and managing one's home media library.

In developing the prototype, we have experimented with a novel role for a mobile device—that of a speech-enabled remote control. We have demonstrated a flexible natural language understanding architecture, in which various processing stages may be performed on either the server or mobile device, as networking and processing power considerations require.

While the mobile platform presents many challenges, it also provides unique opportunities. Whereas desktop computers and TV remote controls tend to be shared by multiple users, a mobile device is typically used by a single individual. By collecting and adapting to the usage data, the system can personalize the recognition and understanding models to improve the system accuracy. In future systems, we hope to not only explore such adaptation possibilities, but also study how real users interact with the system to further improve the user interface.

Acknowledgments

This research is sponsored by the TParty Project, a joint research program between MIT and Quanta Computer, Inc.; and by Nokia, as part of a joint MIT-Nokia collaboration.

References

L. Baptist and S. Seneff. 2000. Genesis-II: A versatile system for language generation in conversational system applications. In *Proc. of ICSLP*.

G. Chung, S. Seneff, C. Wang, and L. Hetherington. 2004. A dynamic vocabulary spoken dialogue interface. In *Proc. of INTERSPEECH*, pages 327–330.

E. Filisko and S. Seneff. 2003. A context resolution server for the GALAXY conversational systems. In *Proc. of EUROSPEECH*.

B. Ford, J. Strauss, C. Lesniewski-Laas, S. Rhea, F. Kaashoek, and R. Morris. 2006. Persistent personal names for globally connected mobile devices. In *Proceedings of the 7th USENIX Symposium on Operating Systems Design and Implementation (OSDI '06)*.

K. Fujita, H. Kuwano, T. Tsuzuki, and Y. Ono. 2003. A new digital TV interface employing speech recognition. *IEEE Transactions on Consumer Electronics*, 49(3):765–769.

J. Glass. 2003. A probabilistic framework for segment-based speech recognition. *Computer Speech and Language*, 17:137–152.

A. Gruenstein, S. Seneff, and C. Wang. 2006. Scalable and portable web-based multimodal dialogue interaction with geographical databases. In *Proc. of INTERSPEECH*.

I. L. Hetherington. 2005. A multi-pass, dynamic-vocabulary approach to real-time, large-vocabulary speech recognition. In *Proc. of INTERSPEECH*.

I. L. Hetherington. 2007. PocketSUMMIT: Small-footprint continuous speech recognition. In *Proc. of INTERSPEECH*, pages 1465–1468.

B. Johanson, A. Fox, and T. Winograd. 2002. The interactive workspaces project: Experiences with ubiquitous computing rooms. *IEEE Pervasive Computing*, 1(2):67–74.

J. Mazzola Paluska, H. Pham, U. Saif, C. Terman, and S. Ward. 2006. Reducing configuration overhead with goal-oriented programming. In *PerCom Workshops*, pages 596–599. IEEE Computer Society.

J. Mazzola Paluska, H. Pham, U. Saif, G. Chau, C. Terman, and S. Ward. 2008. Structured decomposition of adapative applications. In *Proc. of 6th IEEE Conference on Pervasive Computing and Communications*.

I. McGraw and S. Seneff. 2007. Immersive second language acquisition in narrow domains: A prototype ISLAND dialogue system. In *Proc. of the Speech and Language Technology in Education Workshop*.

H.-J. Oh, C.-H. Lee, M.-G. Jang, and Y. K. Lee. 2007. An intelligent TV interface based on statistical dialogue management. *IEEE Transactions on Consumer Electronics*, 53(4).

T. Paek, M. Agrawala, S. Basu, S. Drucker, T. Kristjansson, R. Logan, K. Toyama, and A. Wilson. 2004. Toward universal mobile interaction for shared displays. In *Proc. of Computer Supported Cooperative Work*.

J. Polifroni, G. Chung, and S. Seneff. 2003. Towards the automatic generation of mixed-initiative dialogue systems from web content. In *Proc. EUROSPEECH*, pages 193–196.

S. Seneff, E. Hurley, R. Lau, C. Pao, P. Schmid, and V. Zue. 1998. GALAXY-II: A reference architecture for conversational system development. In *Proc. ICSLP*.

S. Seneff, C. Wang, and T. J. Hazen. 2003. Automatic induction of n-gram language models from a natural language grammar. In *Proceedings of EUROSPEECH*.

S. Seneff. 1992. TINA: A natural language system for spoken language applications. *Computational Linguistics*, 18(1):61–86.

S. Seneff. 2002. Response planning and generation in the MERCURY flight reservation system. *Computer Speech and Language*, 16:283–312.

S. Seneff. 2007. Reversible sound-to-letter/letter-to-sound modeling based on syllable structure. In *Proc. of HLT-NAACL*.

V. Zue, S. Seneff, J. Glass, J. Polifroni, C. Pao, T. J. Hazen, and L. Hetherington. 2000. JUPITER: A telephone-based conversational interface for weather information. *IEEE Transactions on Speech and Audio Processing*, 8(1), January.