Speech and Language Processing for Multimodal Human-Computer Interaction [1]

L. Deng, Y. Wang, K. Wang, A.Acero, H. Hon, J. Droppo,

C. Boulis, , M. Mahajan, and X.D.Huang

Microsoft ResearchOne Microsoft Way, Redmond, WA 98052, USA.

Phone: 425-706-2719; fax: 425-936-7329; contact author : Li Deng, e-mail: deng@microsoft.com

Abstract

In this paper, we describe our recent work at Microsoft Research, in the project codenamed *Dr. Who*, aimed at the development of enabling technologies for speech-centric multimodal human-computer interaction. In particular, we present in detail MiPad as the first *Dr. Who's* application that addresses specifically the mobile user interaction scenario. MiPad is a wireless mobile PDA prototype that enables users to accomplish many common tasks using a multimodal spoken language interface and wireless-data technologies. It fully integrates continuous speech recognition and spoken language understanding, and provides a novel solution to the current prevailing problem of pecking with tiny styluses or typing on minuscule keyboards in today's PDAs or smart phones. Despite its current incomplete implementation, we have observed that speech and pen have the potential to significantly improve user experience in our user study reported in this paper. We describe in this system-oriented paper the main components of MiPad, with a focus on the robust speech processing and spoken language understanding aspects. The detailed MiPad components discussed include: distributed speech recognition considerations for the speech processing algorithm design; a stereo-based speech feature enhancement algorithm used for noise-robust front-end speech processing; Aurora2 evaluation results for this front-end processing; speech feature compression (source coding) and error protection (channel coding) for distributed speech recognition in MiPad; HMM-based acoustic modeling for

[1] Various portions of this article have been presented at ICSLP-2000 [1][8][10] in Beijing, China, ICASSP-2001 [3][9][12] in Salt Lake City, US, Eurospeech-1999 [13] in Hungary, Eurospeech-2001 [6] in Denmark, and at ASRU-2001 Workshop [4][14] in Italy.

continuous speech recognition decoding; a unified language model integrating context-free grammar and N-gram model for the speech decoding; schema-based knowledge representation for the MiPad's personal information management task; a unified statistical framework that integrates speech recognition, spoken language understanding and dialogue management; the robust natural language parser used in MiPad to process the speech recognizer's output; a machine-aided grammar learning and development used for spoken language understanding for the MiPad task; *Tap & Talk* multimodal interaction and user interface design; back channel communication and MiPad's error repair strategy; and finally, user study results that demonstrate the superior throughput achieved by the *Tap & Talk* multimodal interaction over the existing pen-only PDA interface. These user study results highlight the crucial role played by speech in enhancing the overall user experience in MiPad-like human-computer interaction devices.

Keywords: Speech-centric multimodal interface, human-computer interaction, robust speech recognition, SPLICE algorithm, denoising, online noise estimation, distributed speech processing, speech feature encoding, error protection, spoken language understanding, automatic grammar learning, semantic schema, user study.

Contact author: Li Deng, Microsoft Research, One Microsoft Way, Redmond, WA 98052, USA;
deng@microsoft.com

I. INTRODUCTION

When humans converse and interact with one another, we often utilize a wide range of communicative devices and effectively exploit these devices to clearly and concisely express our communicative intentions. In contrast, in human-computer interaction, humans have typically been limited to interactions through the use of the graphical user interface (GUI) including a graphical display, a mouse, and a keyboard. While GUI-based technology has significantly improved human computer interface by using intuitive real-world metaphors, it nevertheless still makes human's interaction with computers far from natural and flexible. Intensive research under the general name of multimodal interfaces has been actively pursued in recent years. These interfaces provide the potential of allowing users to interact with computers through several different modalities such as speech, graphical displays, keypads, and pointing devices, and offer significant advantages over traditional unimodal interfaces. The purpose of this paper is to present some of our group's recent research activities, under the project's codename *Dr. Who*, on speech and language processing as applied to speech-centric multimodal human-computer interaction. In particular, we will describe the first showcase project for *Dr. Who*, the Multimodal Intelligent Personal Assistant Device, or MiPad, which is a next generation Personal Digital Assistant (PDA) prototype embedding several key sub-areas of spoken language technology. These sub-areas include speech analysis, environment-robust speech processing, spoken language understanding, and multimodal interaction --- all directly pertinent to the major themes in this special issue on real-world speech processing.

One particular limitation of the traditional GUI which we specifically address in our research is the heavy reliance of GUI on a sizeable screen, keyboard and pointing device, whereas such a screen or device is not always available. With more and more computers being designed for mobile usages and hence being subject to the physical size and hands-busy or eyes-busy constraints, the traditional GUI faces an even greater challenge. Multimodal interface enabled by spoken language is widely believed to be capable of dramatically enhancing the usability of computers because GUI and speech have complementary strengths. While spoken language has the potential to provide a natural interaction model, the ambiguity of spoken language and the memory burden of using speech as output modality on the user have so far prevented it from becoming the choice of a mainstream interface. MiPad is one of our attempts in developing enabling technologies for speech-centric multimodal interface. MiPad is a prototype of wireless PDA that enables users to accomplish many common tasks using a multimodal spoken language interface (speech + pen + display). One of MiPad's hardware design concepts is illustrated in Figure 1.

MiPad intends to alleviate the current prevailing problem of pecking with tiny styluses or typing on minuscule keyboards in today's PDAs by introducing the speech capability using a built-in microphone. However, resembling more a PDA and less a telephone, MiPad intentionally avoids speech-only interactions. MiPad is designed to support a variety of tasks such as E-mail, voice-mail, calendar, contact list, notes, web browsing, mobile phone, and

document reading and annotation. This collection of functions unifies the various devices that people carry around today into a single, comprehensive communication and productivity tool. While the entire functionality of MiPad can be accessed by pen alone, it is preferred to be accessed by speech and pen combined. The user can dictate to an input field by holding the pen down in it. Alternatively, the user can also select the speech field by using the roller to navigate and by holding it down while speaking. The field selection, called *Tap & Talk*, not only indicates where the recognized text should go but also serves as a push to talk control. *Tap & Talk* narrows down the number of possible instructions for spoken language processing. For example, selecting the "*To:"* field on an e-mail application display indicates that the user is about to enter a name. This dramatically reduces the complexity of spoken language processing and cuts down the speech recognition and understanding errors to the extent that MiPad can be made practically usable despite the current well known limitations of speech recognition and natural language processing technology.

One unique feature of MiPad is a general purpose "Command" field to which a user can issue naturally spoken commands such as "Schedule a meeting with Bill tomorrow at two o'clock." From the user's perspective, MiPad recognizes the commands and follows with necessary actions. In response, it pops up a "meeting arrangement" page with related fields (such as date, time, attendees, etc.) filled appropriately based on the user's utterance. Up to this date, MiPad fully implements PIM (Personal Information Management) functions including email, calendar, notes, task, and contact list with a hardware prototype based on Compaq's iPaq PDA. All MiPad applications are configured in a client-server architecture as shown in Figure 2. The client on the left side of Figure 2 is MiPad powered by Microsoft Windows CE operating system that supports (1) sound capturing ability, (2) front-end acoustic processing including noise reduction, channel normalization, feature compression, and error protection, (3) GUI processing module, and (4) a fault-tolerant communication layer that allows the system to recover gracefully from the network connection failures. Specifically, to reduce bandwidth requirements, the client compresses the wideband speech parameters down to a maximal 4.8 Kbps bandwidth. Between 1.6 and 4.8 Kbps, we observed virtually no increase in the recognition error on some tasks tested. A wireless local area network (WLAN), which is currently used to simulate a third generation (3G) wireless network, connects MiPad to a host machine (server) where the continuous speech recognition (CSR) and spoken language understanding (SLU) take place. The client takes approximately 450 KB of program space and an additional 200 KB of runtime heap, and utilizes approximately 35% of CPU time with iPAQ's 206 MHz StrongARM processor. At the server side, as shown on the right side of Figure 2, MiPad applications communicate with the continuous speech recognition (CSR) and spoken language understanding (SLU) engines for coordinated context-sensitive *Tap & Talk* interaction.

Given the above brief overview of MiPad which exemplifies our speech-centric multimodal human-computer interaction research, the remaining of this paper will describe details of MiPad with an emphasis on speech processing, spoken language understanding, and the UI design considerations. Various portions of this paper have been presented at the several conferences (ICSLP-2000 [1][8][10], ICASSP-2001 [3][9][12], Eurospeech-1999 [13],

Eurospeech-2001 [6] and ASRU-2001 Workshop [4][14]). The purpose of this paper is to synthesize these earlier presentations on the largely isolated MiPad components into a single coherent one so as to highlight the important roles of speech and language processing in the MiPad design, and report some most recent research results. The organization of this paper is as follows. In Sec. II and Sec. III, we describe our recent work on front-end speech processing, including noise robustness and source/channel coding, which underlie MiPad's distributed speech recognition capabilities. Acoustic models and language models used for the decoding phase of MiPad continuous speech recognition are presented in Sec. IV. Spoken language understanding using the domains related to MiPad and other applications is described in Sec. IV, together with dialogue management directly linked to the application logic and user interface processing. Finally, MiPad user interface and user study results are outlined in Sec. VI, and a summary is provided in Sec. VII.

II. ROBUST SPEECH PROCESSING

Robustness to acoustic environment or immunity to noise and channel distortion is one most important aspect of MiPad design considerations. For MiPad to be acceptable to the general public, it is desirable to remove the need for a close-talking microphone in capturing speech. Although close-talking microphones pick up relatively little background noise and allow speech recognizers to achieve high accuracy for the MiPad-domain tasks, it is found that users much prefer built-in microphones even if there is minor accuracy degradation. With the convenience of using built-in microphones, noise robustness becomes a key challenge to maintaining desirable speech recognition and understanding performance. Our recent work on acoustic modeling aspects of MiPad has focused mainly on this noise-robustness challenge. In this section we will present most recent results in the framework of distributed speech recognition (DSR) that MiPad design has adopted.

A. Distributed speech recognition considerations for algorithm design

There has been a great deal of interest recently in standardizing DSR applications for a plain phone, PDA, or a smart phone where speech recognition is carried out at a remote server. To overcome bandwidth and infrastructure cost limitations, one possibility is to use a standard codec on the device to transmit the speech to the server where it is subsequently decompressed and recognized. However, since speech recognizers such as the one in MiPad only need some features of the speech signal (e.g., Mel-cepstrum), bandwidth can be further saved by transmitting only those features. Much of the recent research programs in the area of DSR have concentrated on the Aurora task [7], representing a concerted effort to standardize a DSR front-end and to address the issues surrounding robustness to noise and channel distortions at a low bit rate. Our recent work on noise robustness in DSR has been concentrated on the Aurora task [6].

In DSR applications, it is easier to update software on the server because one cannot assume that the client is always running the latest version of the algorithm. With this consideration in mind, while designing noise-robust algorithms for MiPad, we strive to make the algorithms front-end agnostic. That is, the algorithms should make no

assumptions on the structure and processing of the front end and merely try to undo whatever acoustic corruption that has been shown during training. This consideration also favors approaches in the feature rather than the model domain.

Here we describe one particular algorithm that has so far given the best performance on the Aurora2 task and other Microsoft internal tasks with much larger vocabularies. We called the algorithm SPLICE, short for Stereo-based Piecewise Linear Compensation for Environments. In a DSR system, the SPLICE may be applied either within the front end on the client device, or on the server, or on both with collaboration. Certainly a server side implementation has some advantages as computational complexity and memory requirements become less of an issue and continuing improvements can be made to benefit even devices already deployed in the field. Another useful property of SPLICE in the serve implementation is that new noise conditions can be added as they are identified once by a server. This can make SPLICE quickly adapt to any new acoustic environment with minimum additional resource.

B. Basic version of the SPLICE algorithm

SPLICE is a frame-based, bias removal algorithm for cepstrum enhancement under additive noise, channel distortion or a combination of the two. In [1] we reported the approximate MAP (Maximum A Posteriori) formulation of the algorithm, and more recently in [3][6] we described the MMSE (Minimum Mean Square Error) formulation of the algorithm with a much wider range of naturally recorded noise, including both artificially mixed speech and noise, and naturally recorded noisy speech.

SPLICE assumes no explicit noise model, and the noise characteristics are embedded in the piecewise linear mapping between the "stereo'' clean and distorted speech cepstral vectors. The piecewise linearity is intended to approximate the true nonlinear relationship between the two. The nonlinearity between the clean and distorted (including additive noise) cepstral vectors arises due to the use of the logarithm in computing the cepstra. Because of the use of the stereo training data that provides accurate estimates of the bias or correction vectors without the need for an explicit noise model, SPLICE is potentially able to handle a wide range of distortions, including nonstationary distortion, joint additive and convolutional distortion, and nonlinear distortion (in time-domain). One key requirement for the success of the basic version of SPLICE described here is that the distortion conditions under which the correction vectors are learned from the stereo data must be similar to those corrupting the test data. Enhanced versions of the algorithm described later in this section will relax this requirement by employing a noise estimation and normalization procedure. Also, while the collection of a large amount of stereo data may be inconvenient for implementation, the collection is needed only in the training phase. Further, our experience suggests that a relatively small amount of stereo data is already sufficient for the SPLICE training, especially for the noise normalization version of the SPLICE.

We assume a general nonlinear distortion of a clean cepstral vector, **x**, into a noisy one, **y**. This distortion is approximated in SPLICE by a set of linear distortions. The probabilistic formulation of the basic version of SPLICE is provided below.

1) Basic assumptions

The first assumption is that the noisy speech cepstral vector follows a mixture distribution of Gaussians

$$p(\mathbf{y}) = \sum_s p(\mathbf{y} \mid s)\, p(s), \qquad with \qquad p(\mathbf{y} \mid s) = N(\mathbf{y}; \boldsymbol{\mu}_s, \boldsymbol{\Sigma}_s). \tag{1}$$

where s denotes the discrete random variable taking the values 1, 2, …, N, one for each region over which the piecewise linear approximation between the clean cepstral vector **x** and distorted cepstral vector is made. This distribution, one for each separate distortion condition (not indexed for clarity), can be thought as a "codebook" with a total of N codewords (Gaussian means) and their variances.

The second assumption made by SPLICE is that the conditional probability density function (PDF) for the clean vector **x** given the noisy speech vector, **y**, and the region index, s, is a Gaussian with the mean vector to be a linear combination of the noisy speech vector **y**. In this paper, we take a simplified form of this (piecewise) function by making the rotation matrix to be identity one, leaving only the bias or correction vector. Thus, the conditional PDF has the form

$$p(\mathbf{x} \mid \mathbf{y}, s) = N(\mathbf{x}; \mathbf{y} + \mathbf{r}_s, \boldsymbol{\Gamma}_s) \tag{2}$$

2) SPLICE training

Since the noisy speech PDF $p(\mathbf{y})$ obeys mixture-of-Gaussian distribution, the standard EM (Expectation and Maximization) algorithm is used to train $\boldsymbol{\mu}_s$ and $\boldsymbol{\Sigma}_s$. Initial values of the parameters can be determined by a VQ (Vector Quantization) clustering algorithm.

The parameters $\mathbf{r}_s$ and $\boldsymbol{\Gamma}_s$ of the conditional PDF $p(\mathbf{x} \mid \mathbf{y}, s)$ can be trained using the maximum likelihood criterion. Since the variance of the distribution is not used in cepstral enhancement, we only give the ML estimate of the correction vector below:

$$\mathbf{r}_s = \frac{\sum_n p(s \mid \mathbf{y}_n)(\mathbf{x}_n - \mathbf{y}_n)}{\sum_n p(s \mid \mathbf{y}_n)}, \qquad \text{where} \tag{3}$$

$$p(s \mid \mathbf{y}_n) = \frac{p(\mathbf{y}_n \mid s)\, p(s)}{\sum_s p(\mathbf{y}_n \mid s)}. \tag{4}$$

This training procedure requires a set of stereo (two channel) data. One channel contains the clean utterance, and the other channel contains the same utterance with distortion, where the distortion represented by the correction vectors is estimated above. The two-channel data can be collected, for example, by simultaneously recording

utterances with one close-talk and one far-field microphone. Alternatively, it has been shown in our research that a large amount of synthetic data can be bootstrapped from a small amount of real data with virtually no loss of speech recognition accuracy.

3) SPLICE for cepstral enhancement

One significant advantage of the above two basic assumptions made in SPLICE is the inherent simplicity in deriving and implementing the rigorous MMSE estimate of clean speech cepstral vectors from their distorted counterparts. Unlike the FCDCN (Fixed Code-Dependent Cepstral Normalization) algorithm [1], no approximations are made in deriving the optimal enhancement rule. The derivation is outlined below.

The MMSE is the following conditional expectation of clean speech vector given the observed noisy speech:

$$E_x[\mathbf{x}|\mathbf{y}] = \sum_s p(s|\mathbf{y}) E_{\mathbf{x}}[\mathbf{x}|\mathbf{y}, s]. \tag{5}$$

Due to the second assumption of SPLICE, the above codeword-dependent conditional expectation of **x** (given **y** and *s)* is simply the bias-added noisy speech vector:

$$E_x[\mathbf{x}|\mathbf{y}, s] = \mathbf{y} + \mathbf{r}_s \tag{6}$$

where bias $\mathbf{r}_s$ has been estimated from the stereo training data according to Eq.(3). This gives the simple form of the MMSE estimate as the noisy speech vector corrected by a linear weighted sum of all codeword-dependent bias vectors already trained:

$$\hat{\mathbf{x}} = E_x[\mathbf{x}|\mathbf{y}] = \mathbf{y} + \sum_s p(s|\mathbf{y}) \mathbf{r}_s \tag{7}$$

While this is already efficient to compute, more efficiency can be achieved by approximating the weights according to

$$\hat{p}(s|\mathbf{y}) \cong \begin{cases} 1 & s = \arg\max_s p(s|\mathbf{y}) \\ 0 & \text{otherwise} \end{cases}. \tag{8}$$

This approximation turns the MMSE estimate to the approximate MAP estimate that consists of two sequential steps of operation. First, finding optimal codeword *s* using the VQ codebook based on the parameters $(\mathbf{r}_s, \mathbf{\Gamma}_s)$, and then adding the codeword-dependent vector $\mathbf{r}_s$ to the noisy speech vector. We have found empirically that the above VQ approximation does not appreciably affect recognition accuracy.

C. Enhancing SPLICE by temporal smoothing

In this enhanced version of SPLICE, we not only minimize the static deviation from the clean to noisy cepstral vectors (as in the basic version of SPLICE), but also seeks to minimize the dynamic deviation.

The basic SPLICE optimally processes each frame of noisy speech independently. An obvious extension is to jointly process a segment of frames. In this way, although the deviation from the clean to noisy speech cepstra for an

individual frame may be undesirably greater than that achieved by the basic, static SPLICE, the global deviation that takes into account the differential frames and the whole segment of frames will be reduced compared with the basic SPLICE.

We have implemented the above idea of "dynamic SPLICE" through temporally smoothing the bias vectors obtained from the basic, static SPLICE. This is an empirical way of implementing the rigorous solution via the use of a more realistic model for the time-evolution of the clean speech dynamics, either in the discrete state, $p(\mathbf{x}_n \mid \mathbf{y}_n, s_n, s_{n-1})$, or in the continuous clean speech vector estimate, $p(\mathbf{x}_n \mid \mathbf{y}_n, s_n, \mathbf{x}_{n-1})$.

An efficient way of implementing an approximate dynamic SPLICE is to time-filter each component of the cepstral bias vector $\mathbf{r}_{s_n}$. We have achieved significant performance gains using this efficient heuristic implementation. In our specific implementation, we used a simple zero-phase, non-causal, IIR filter to smooth the cepstral bias vectors. This filter has a low-pass characteristic, with the system transfer function of

$$H(z) = \frac{-0.5}{\left(z^{-1} - 0.5\right)(z - 2)}. \tag{9}$$

This transfer function is the result of defining an objective function as the summation of the static and dynamic deviations from clean speech to noisy speech vectors. The optimal solution that minimizes this objective function is of the form of Eq. (9), where the constants are functions of the variances in the speech model. In practice, we found in experiments that using Eq. (9) instead of the exact solution produces similar results at a lower computational cost.

D. Enhancing SPLICE by noise estimation and noise normalization

One well recognized deficiency of the SPLICE algorithm discussed so far is that the noise condition is often unseen in the collected stereo data used to train the SPLICE codebooks. In the new, noise-normalized version of SPLICE, different noise conditions between the SPLICE training set and test set are normalized. The procedure for noise normalization and for denoising is as follows. Instead of building codebooks for noisy speech feature vector **y** from the training set, they are built from **y** - **n**, where **n** is an estimated noise from **y**. Then the correction vectors are estimated from the training set using the noise-normalized stereo data (**y – n**) and (**x** – **n**). The correction vectors trained in this new SPLICE will be different from those in the basic version of SPLICE. This is because the codebook selection will be different since p(s|**y**) is changed to p(s|**y-n**). For denoising in the test data, the noise-normalized noisy cepstra **y – n** are used to obtain the noise-normalized MMSE estimate, and then the noise normalization is undone by adding the estimated noise **n** back to the MMSE estimate. This noise normalization procedure is intended to eliminate possible mismatch between the environment where SPLICE stereo training takes place and the environment where SPLICE is deployed to remove the noise in the test data.

Our research showed that the effectiveness of the above noise-normalized SPLICE is highly dependent on the accuracy of the noise estimate **n** [4][6]. We have carried out research on several ways of automatically estimating nonstationary noise in the Aurora2 database. We describe below one algorithm that has given by far the highest accuracy in noise estimation and at the same time by far the best noise-robust speech recognition results evaluated on the Aurora2 task.

E. Nonstationary noise estimation by iterative stochastic approximation

In [4], a novel algorithm is proposed, implemented, and evaluated for recursive estimation of parameters in a nonlinear model involving incomplete data. The algorithm is applied specifically to time-varying deterministic parameters of additive noise in a mildly nonlinear model that accounts for the generation of the cepstral data of noisy speech from the cepstral data of the noise and clean speech. For computer recognition of the speech that is corrupted by highly nonstationary noise, different observation data segments correspond to very different noise parameter values. It is thus strongly desirable to develop recursive estimation algorithms, since they can be designed to adaptively track the changing noise parameters. One such design based on the novel technique of iterative stochastic approximation within the recursive-EM framework is developed and evaluated. It jointly adapts time-varying noise parameters and the auxiliary parameters introduced to piecewise linearly approximate the nonlinear model of the acoustic environment. The accuracy of approximation is shown to have improved progressively with more iterations.

The essence the algorithm is the use of iterations to achieve close approximations to a nonlinear model of the acoustic environment while at the same time employing the "forgetting" mechanism to effectively track nonstationary noise. Using a number of empirically verified assumptions associated with the implementation simplification, the efficiency of this algorithm has been improved close to real time for noise tracking. The mathematical theory, algorithm, and implementation detail of this iterative stochastic approximation technique can be found in [4][5].

Figures 3, 4, and 5 show the results of noise-normalized SPLICE denoising using the iterative stochastic algorithm for tracking nonstationary noise **n** in an utterance of the Aurora2 data, where the SNR is 10dB, 5dB, and 0dB, respectively. From top to bottom panels are noisy speech, clean speech, and denoised speech, all in the same spectrogram format. Most of the noise has been effectively removed, except for some strong noise burst located around 150-158 frames in Figure 5 where the instantaneous SNR is significantly lower than zero.

F. Aurora2 evaluation results

Noise-robust connected digit recognition results obtained using the best version of SPLICE are shown in Figure 6 for the full Aurora2 evaluation test data. Sets-A and -B each consists of 1101 digit sequences for each of four noise conditions and for each of the 0dB, 5 dB, 10dB, 15dB, and 20dB SNRs. The same is for Set-C except there are only two noise conditions. All the results in Figure 6 are obtained with the use of cepstral mean normalization (CMN) for all data after applying noise-normalized, dynamic SPLICE to cepstral enhancement. The use of CMN has substantially improved the recognition rate for Set-C. For simplicity purposes, we have assumed no channel

distortion in the implementation of the iterative stochastic approximation algorithm for noise estimation. This assumption would not be appropriate for Set-C which contains unknown but fixed channel distortion. This deficiency has been, at least partially, offset by the use of CMN.

The word error rate reduction achieved as shown in Figure 6 is 27.9 % for the multi-condition training mode, and 67.4% for the clean-only training mode, respectively, compared with the results using the standard Mel cepstra with no speech enhancement. In the multi-condition training mode, the denoising algorithm is applied to the training data set and the resulting denoised Mel-cepstral features are used to train the HMMs. In the clean-training mode, the HMMs are trained using clean speech Mel-cepstra and the denoising algorithm is applied only to the test set. The results in Figure 6 represent the best performance in the September-2001 AURORA2 evaluation. The experimental results also demonstrated the crucial importance of using the newly introduced iterations in improving the earlier stochastic approximation technique, and showed a varying degree of sensitivity, depending on the degree of noise nonstationarity, of the noise estimation algorithm's performance to the forgetting factor embedded in the algorithm [5].

III. COMPRESSION AND ERROR PROTECTION

In addition to noise robustness, we recently also started the work on the feature compression (source coding) and error protection (channel coding) aspects of distributed speech recognition that is required by the client-server architecture for MiPad. This work is intended to address the three key requirements for successful deployment of distributed speech recognition associated with the client-server approach: 1) Compression of cepstral features (via quantization) must not degrade the speech recognition performance; 2) The algorithm for source and channel coding must be robust to packet losses, bursty losses or otherwise; and 3) The total time delay due to the coding, which results from a combined quantization delay, error-correction coding delay, and transmission delay, must be kept within an acceptable level. In this section, we outline the basic approach and preliminary results of this work.

A. Feature compression

A new source coding algorithm has been developed that consists of two sequential stages. After the standard Mel-cepstra are extracted, each speech frame is first classified to a phonetic category (e.g., phoneme) and then is vector quantized (VQ) using the split-VQ approach. The motivation behind this new source coder is that the speech signal can be considered piecewise-stationary segments, and therefore can be most efficiently coded using one of many small codebooks that is tuned into that particular segment. Also, the purpose of the source coding considered here is to reduce the effect of coding on the speech recognizer’s word error rate on the server-side of MiPad, which is very different from the usual goal of source coding aiming at maintaining perceptual quality of speech. Therefore, the use of phone-dependent codebooks is deemed most appropriate since phone distinction can be enhanced by using

separate codebooks for distinct phones. Phone distinction often leads to word distinction, which is the goal of speech recognition and also the ultimate goal of feature compression designed for this purpose in the MiPad design.

One specific issue to be addressed in the coder design is bit allocation, or the number of bits that must be assigned to the subvector codebooks. In our coder, C0, C1-6, and C7-12 form three separate sets of subvectors that are quantized independently (i.e., M=3, where M is the total number of independent codebooks). Starting from 0 bit for each subvector codebook of each phone we can evaluate every possible combination of bits to subvectors and select the best according to a certain criterion. To better match the training procedure to the speech recognition task we use the criterion of minimal word error rate (WER). That is, big assignment is the result of following constrained optimization:

$$\hat{B} = \arg\min_{B}\{WER(B)\},$$

under the constraint of $\sum_i b_i = N$, where b_i is the number of bits assigned to the i-th subvector, N is the total number of bits to be assigned, and *WER(B)* is the WER by using B as the assignment of the bits to subvectors. For the full search case, because for each one of the possible combinations we must run a separate WER experiment and the total number of combinations is prohibitively high we use a greedy bit allocation technique. At each stage we add a bit at each one of the subvectors and we keep the combination with the minimal WER. We repeat the procedure for the next stage by starting at the best combination of the previous step. By having M subvectors and N total bits to assign the total number of combinations is reduced from M^N to $M \times N$.

The experiments carried out to evaluate the above phone-dependent coder use the baseline system with a version of the Microsoft continuous-density HMMs (Whisper). The system uses 6000 tied HMM states (senones), 20 Gaussians per state, Mel-cepstrum, delta cepstrum, and delta-delta cepstrum. The recognition task is 5000-word vocabulary, continuous speech recognition from Wall Street Journal data sources. A fixed, bigram language model is used in all the experiments. The training set consists of a total of 16,000 female sentences, and the test set of 167 female sentences (2708 words). The word accuracy with no coding for this test set was 95.7%. With use of a perfect phone classifier, the coding using the bit allocation of (4, 4, 4) for the three subvectors gives word accuracy of 95.6%. Using a very simple phone classifier with Mahalanobis distance measure, the recognition accuracy drops only to 95.0%. For this high-performance coder, the bandwidth has been reduced to 1.6 Kbps with the use of coder memory of 64 Kbytes.

B. Error protection

In the recent work, a novel channel coder has also been developed to protect the Mel-cepstral features for MiPad speech recognition based on the client-server architecture. The channel coder assigns unequal amounts of redundancy among the different source code bits, giving a greater amount of protection to the most important bits. The greater contributions the bits make to reducing the word error rate in speech recognition, the more important these bits are.

A quantifiable procedure to assess the importance of each bit is developed, and the channel coder exploits this utility function for the optimal forward error correction (FEC) assignment. The FEC assignment algorithm assumes that packets are lost according to a Poisson process. Simulation experiments are performed where the bursty nature of loss patterns are taken into account. When combined with the new source coder, the new channel coder is shown to provide considerable robustness to packet losses even under extremely adverse conditions.

Some alternatives to FEC coding are also explored, including the use of multiple transmission, interleaving , and interpolation. We conclude from this preliminary work that the final choice of channel coder should depend on the relative importance among such factors as delay, bandwidth, and tolerance to the burstiness of noise.

Our preliminary work on the compression and error protection aspects of distributed speech recognition has provided clear insight into the tradeoffs we need to make between source coding, delay, computational complexity and resilience to packet loss. Most significantly, the new algorithms developed have been able to bring down the Mel-cepstra compression rate to as low as 1.6 Kbps with virtually no degradation in word error rate compared with no compression. These results are currently being incorporated into the next version of the MiPad design.

IV. CONTINUOUS SPEECH RECOGNITION AND LANGUAGE MODELING

While the compressed and error-protected Mel-cepstral features are computed in the MiPad client, major computation for continuous speech recognition (decoding) resides in the host computer as the server. The entire set of the statistical language model, the acoustic model in the form of hidden Markov models (HMMs), and the lexicon that are used for speech decoding all reside in the server, processing the Mel-cepstral features transmitted from the client. Denoising operations such as SPLICE that extract noise-robust Mel-cepstra can reside either on the server or on the client, or on both working in collaboration.

The MiPad is designed to be a *personal* device. As a result, the recognition uses speaker-adaptive acoustic models (HMMs) and a user-adapted lexicon to improve recognition accuracy. The HMMs and the continuous speech decoding engine are both derived from an improved version of the Microsoft's Whisper speech recognition system and of the HTK, which combines the best features of these earlier two separate systems. Both MLLR (Maximum Likelihood Linear Regression) and MAP adaptation are used to adapt the speaker-independent acoustic model for each individual speaker. There are 6000 senones, each with 20-component mixture Gaussian densities. The context-sensitive language model is used for relevant semantic objects driven by the user's pen tapping action, as will be described in the MiPad's Tap and Talk interface design in Sec. VI. As speech recognition accuracy remains as a major challenge for MiPad usability, most of our recent work on MiPad's acoustic modeling has focused on noise robustness as described in Sec.II. The work on language modeling for improving speech recognition accuracy has focused on language model portability, which is described in this section below.

The speech recognition engine in MiPad uses the unified language model [12] that takes advantage of both rule-based and data-driven approaches. Consider two training sentences:

- *"Meeting at three with Zhou Li"*. vs.
- *"Meeting at four PM with Derek"*.

Within a pure n-gram framework, we will need to estimate

P(Zhou|three with) and *P(Derek|PM with)*

individually. This makes it very difficult to capture the obviously needed long-span semantic information in the training data. To overcome this difficulty, the unified model uses a set of CFGs that captures some of the common named entities. For the example listed here, we may have CFG's for <NAME> and <TIME> respectively, which can be derived from the factoid grammars of smaller sizes. The training sentences now look like:

- *"Meeting <at three:TIME> with <Zhou Li:NAME>"*, and
- *"Meeting <at four PM:TIME> with <Derek: NAME>"*.

With parsed training data, we can now estimate the n-gram probabilities as usual. For example, the replacement of

P(Zhou|three with) ← P(<NAME>|<TIME> with)

makes such "n-gram" representation more meaningful and more accurate.

Inside each CFG, however, we can still derive

P("Zhou Li"|<NAME>) and P("four PM"|<TIME>)

from the existing n-gram (n-gram probability inheritance) so that they are appropriately normalized [12]. This unified approach can be regarded as a generalized n-gram in which the vocabulary consists of words and structured classes. The structured class can be simple, such as <DATE>, <TIME>, and <NAME>, if there is no need to capture deep structural information. It can be made complicated also in order to contain deep structured information. The key advantage of the unified language model is that we can author limited CFGs for each new domain and embed them into the domain-independent n-grams. In short, CFGs capture domain-specific structural information that facilitates language model portability, while the use of n-grams makes the speech decoding system robust against catastrophic errors.

Most decoders can only support either CFGs or word n-grams. These two ways of representing sentence probabilities were mutually exclusive. We have modified the decoder so that we can embed CFGs in the n-gram search framework to take advantage of the unified language model. An evaluation of the use of the unified language model is shown in Table 1. The speech recognition error rate with the use of the unified language model is demonstrated to be significantly lower than that with the use of the domain-independent trigram. That is, incorporating the CFG into the language model drastically improves cross-domain portability. The test data shown in Table 1 are based on MiPad's PIM *conversational speech*. The domain-independent trigram language model is based

on Microsoft Dictation trigram models used in Microsoft Speech SDK 4.0. In Table 1, we also observe that using the unified language model directly in the decoding stage produces about 10% fewer recognition errors than doing N-best re-scoring using the identical language model. This demonstrates the importance of using the unified model in the early stage of speech decoding.

Systems	Perplexity	Word Error	Relative Decoding Time
Domain-independent Trigram	593	35.6%	1.0
Unified decoder with the unified LM	141	22.5%	0.77
N-best re-scoring with the unified LM	-	24.2%	-

Table 1: Cross-domain speaker-independent speech recognition performance with the unified language model and its corresponding decoder.

V. SPOKEN LANGUAGE UNDERSTANDING AND DIALOGUE MANAGEMENT

The spoken language understanding (SLU) engine used in our speech-centric multimodal human-computer interaction research, in MiPad research in particular, is based on a robust chart parser [13] and a plan-based dialog manager [11]. Each semantic object defined and used for SLU is either associated with a real-world entity or an action that the application takes on a real-entity. Each semantic object has slots that are linked with their corresponding CFG. In contrast to the sophisticated prompting response in voice-only conversational interface, the response is a direct graphic rendering of the semantic object on MiPad's display. After a semantic object is updated, the dialog manager fulfills the plan by executing application logic and error repair strategy.

One of the critical tasks in SLU is semantic grammar authoring. Manual development of domain-specific grammars is time-consuming, error-prone, and it requires a significant amount of expertise. We have been working on semi-automatic grammar learning tools that take advantage of multiple information sources to help developers author domain-specific semantic grammars.

In this section, we will describe in detail our recent research activities outlined above.

A. Semantic schema and knowledge representation

MiPad adopts a semantic based robust understanding technology for spoken language understanding. At the center of the technology is *semantic schema* defined in the Semantic Description Language (SDL). The semantic schema is a domain model; it defines the entity relations of a specific domain. The semantic schema is used for many different purposes. It serves as the specification for a language-enabled application: once a semantic schema is defined, grammar and application logic development can proceed simultaneously according to the semantic schema. It also plays a critical role in dialogue management. Further, semantic schema is language and expression independent in the sense that it does not specify the linguistic expressions used to express the concepts. Because of this, it is used not only for language-enabled applications, but also for integrating inputs from multi-modalities, such as mouse click events. Below is an example of concept definitions in a semantic schema:

```
<command type="AppointmentUpdate" name="AddAttendee">
    <slot type="People"/>
    <slot type="ExistingAppt"/>
</command>
<command type="AppointmentUpdate" name="ScheduleAppointment">
    <slot type="People"/>
    <slot type="Time" name="StartTime"/>
    <slot type="Time" name="EndTime"/>
</command>
<entity type="ExistingAppt" name="ApptByAttributes">
    <slot type="People"/>
    <slot type="Time" name="StartTime"/>
    <slot type="Time" name="EndTime"/>
</entity>
<entity type="People" name="ByName">
    <slot class="FirstName"/>
    <slot class="LastName"/>
</entity>
<verbatim type="Name" name="Firstname"/>
<verbatim type="Name" name="Lastname"/>
```

A semantic schema consists of a list of definitions for semantic classes. A semantic class corresponds to a concept in the application domain. The example above shows three different kinds of semantic classes: *command*, *entity* and *verbatim*. Command and entity are semantic classes that contain component slots, while verbatim is a semantic terminal. Each semantic class has a type, and multiple semantic classes may share the same type. For example, the type "People" can be shared by semantic classes "ByName" (e.g., Peter Johnson), "ByReportingRelation" (e.g., my manager), "ByAnaphora" (e.g. him), or "ByClick" (e.g. Mouse Clicking on a person's picture). Slots of a semantic class are specified with either a type or a semantic class. The former constrains that the slot must be filled with a semantic object (an instantiation of a semantic class) that has the specified type, and the latter requires that the slot be filled with an instantiation of that specific semantic class. In case two slots are specified with the same type or semantic class, additional names are used to differentiate them (e.g., StartTime and EndTime in the above example.)

In a human-machine conversation, the computer system responds to the semantics (denoted by S) of a user's utterance (word sequence w) with an appropriate action (A). It does so with the help of the discourse structure (D), which accumulates over all the relevant semantic information from the beginning of the conversation up to the current utterance. Both the utterance semantics and the discourse information are represented in an XML structure that maps words in the utterances to the semantic classes (including all their slots). The following is a concrete example of a dialogue, together with the utterance semantics and discourse structures after each of the user's turns in the dialogue.

w1: Schedule a meeting with Peter.
A1: Peter who?
w2: Peter Johnson.
A2: When do you want to start the meeting?
w3: Tuesday at 2 pm.

After the user utters w1, its meaning can be represented as the following (partial) semantic object:

```
S1: <ScheduleAppointment type = "AppointmentUpdate">
        <ByName  type= "Person">
            <FirstName type= "Name"> Peter </FirstName>
        </ByName>
    </ScheduleAppointment>
```

Since this is the first utterance of the dialogue, the discourse structure D1 = S1.

By comparing D1 with the semantic class definition of ByName, the system knows that it cannot resolve the ambiguity without the last name for Peter. Therefore, it takes the appropriate action (A1) to ask the user for the last name of Peter. After the user responds with the reply w2, the SLU system "understands" w2 by generating the following utterance semantics (via robust parsing):

```
S2: <ByName  type= "Person">
        <FirstName type= "Name"> Peter </FirstName>
        <LastName type= "Name"> Johnson </LastName>
    </ByName>
```

And the new discourse structure is obtained by adjoining S2 to D1:

```
D2: <ScheduleAppointment type = "AppointmentUpdate">
        <ByName type="People">
            <FirstName type= "Name"> Peter </FirstName>
            <LastName type= "Name"> Johnson </LastName>
        </ByName>
    </ScheduleAppointment>
```

After w2, the system resolves the ByName semantic object. It then tries to resolve the parent semantic object "ScheduleAppointment". By comparing D2 with the semantic class definition for "ScheduleAppointment", it knows that the time information for the appointment is missing. Therefore, it prompts the user with A2. After the user

replies with w3, it generates the following utterance semantic object and augments the discourse structure accordingly:

S3: <Time type = “Time” name = “StartTime”> Tuesday 2 pm </Time>

D3: <ScheduleAppointment type = “AppointmentUpdate”>
<ByName type= “People”>
<FirstName type= “Name”> Peter </FirstName>
<LastName type= “Name”> Johnson </LastName>
</ByName>
<Time type = “Time” name = “StartTime”> Tuesday 2 pm </Time>
</ScheduleAppointment>

After the system resolves all ambiguities about the meeting by completing all the slots in the semantic class specified in the schema, it will take the appropriate action to set the meeting in the calendar and inform the attendees about it.

The example above illustrates the underlying working principle of the language processing component in the MiPad design. In the next section, we will provide a more rigorous mathematical framework for such a component.

B. A unified framework for speech recognition, understanding, and dialogue management

The SLU and dialogue management components in our multimodal interaction system’s architecture exemplified by MiPad can be integrated with the speech recognition component, and, together, be placed into a unified pattern recognition framework employing powerful optimization techniques in system engineering. Using the same notation as the previous example, we let S_n and D_n denote the utterance semantic object and discourse structure, respectively, and A_n the system’s action, all after the n-th dialog turn. Given a new acoustic signal of speech $\mathbf{y}$, the SLU problem can be formulated as the following optimization one:

$$\hat{D}_n = \underset{D_n}{\operatorname{argmax}}\, P(D_n \mid \mathbf{y}, D_{n-1}, A_1^{n-1}) = \underset{D_n}{\operatorname{argmax}} \sum_w P(D_n \mid w, \mathbf{y}, D_{n-1}, A_1^{n-1}) P(w \mid \mathbf{y}, D_{n-1}, A_1^{n-1}), \quad (10)$$

where w is the word sequence (in the current dialogue turn) corresponding to the speech utterance $\mathbf{y}$, and is marginalized out on the right hand side of Eq. (10) using the weighting function $P(w \mid \mathbf{y}, D_{n-1}, A_1^{n-1})$ provided by the speech recognizer. The speech recognizer computes this posterior probability according to

$$P(w \mid \mathbf{y}, D_{n-1}, A_1^{n-1}) = \frac{P(\mathbf{y} \mid w, D_{n-1}, A_1^{n-1}) P(w \mid D_{n-1}, A_1^{n-1})}{\sum_w P(\mathbf{y} \mid w, D_{n-1}, A_1^{n-1}) P(w \mid D_{n-1}, A_1^{n-1})} \approx \frac{P(\mathbf{y} \mid w) P(w \mid D_{n-1}, A_{n-1})}{\sum_w P(\mathbf{y} \mid w) P(w \mid D_{n-1}, A_{n-1})}, \quad (11)$$

where $P(\mathbf{y} \mid w, D_{n-1}, A_1^{n-1}) = P(\mathbf{y} \mid w)$ is the acoustic score from the recognizer,[2] and $P(w \mid D_{n-1}, A_1^{n-1}) = P(w \mid D_{n-1}, A_{n-1})$ is the dialogue-state dependent language model score. For the discourse model

[2] Here we assume conditional independence between the speech acoustics and the semantic object given the word sequence.

$P(D_n \mid \mathbf{y}, w, D_{n-1}, A_1^{n-1})$, since D_n can be deterministically obtained from D_{n-1} and S_n, as their adjoin, we can simplify it to

$$P(D_n \mid \mathbf{y}, w, D_{n-1}, A_1^{n-1}) = P(S_n \mid \mathbf{y}, w, D_{n-1}, A_1^{n-1}).$$

In practice, we further assume conditional independence between the speech acoustics **y** and the semantic object S_n given word sequence *w*, and make a Markov assumption on the action sequence A_1^{n-1}. This thus further simplifies the discourse model component in Eq. (10) to

$$P(S_n \mid \mathbf{y}, w, D_{n-1}, A_1^{n-1}) = P(S_n \mid w, D_{n-1}, A_{n-1}).$$

We now take into account the acoustic denoising operation that nonlinearly transforms the generally distorted speech signal **y** into its estimated undistorted version $\hat{\mathbf{x}}$. Using the example of the SPLICE algorithm as described in Sec.II, we have the relationship between the distorted and estimated undistorted speech features:

$$\hat{\mathbf{x}} = g(\mathbf{y}) = \mathbf{y} + \sum_s p(s \mid \mathbf{y}) \mathbf{r}_s(\mathbf{y}).$$

Further, we use Viterbi approximation to reduce the computational load in the overall optimization operation shown in the framework of Eq. (10) for the SLU problem. Taking account all the above considerations and approximations, the SLU problem formulated in Eq.(10) becomes drastically simplified to

$$\begin{aligned} \hat{S}_n &\approx \underset{S_n, w}{\text{argmax}}\ P(S_n \mid w, D_{n-1}, A_{n-1}) P(w \mid \hat{\mathrm{x}}, D_{n-1}, A_{n-1}) \\ &= \underset{S_n, w}{\text{argmax}}\ P(S_n \mid w, D_{n-1}, A_{n-1}) \frac{P(\hat{\mathrm{x}} \mid w, D_{n-1}, A_{n-1}) P(w \mid D_{n-1}, A_{n-1})}{\sum_w P(\hat{\mathrm{x}} \mid w, D_{n-1}, A_{n-1}) P(w \mid D_{n-1}, A_{n-1})} \end{aligned} \quad (12)$$

We note that even with all these simplifications, significant challenges remain due to the data sparseness problem in training the semantic model $P(S_n \mid w, D_{n-1}, A_{n-1})$ and due to the large computational requirement for evaluating the denominator in the above equation.

Based on the formulation of the SLU solution discussed above, which seeks the maximal probability $P(S_n \mid \hat{\mathrm{x}}, D_{n-1}, A_{n-1})$, or equivalently the maximal product of the two probabilities: $P(S_n \mid w, D_{n-1}, A_{n-1}) P(w \mid \hat{\mathrm{x}}, D_{n-1}, A_{n-1})$, the dialog management problem can also be formulated as an optimization problem as follows: For the *n*-th turn, find an action *A* such that the averaged cost function $C(A, D_n)$ is minimized over the conditional probability measure of $P(S_n \mid \hat{\mathrm{x}}, D_{n-1}, A_{n-1})$. That is,

$$\hat{A} = \underset{A}{\text{argmax}}\, E\left[C(A, D_n) \mid \hat{\mathrm{x}}, D_{n-1}, A_{n-1}\right] = \underset{A}{\text{argmax}} \sum_{D_n} C(A, D_n) P(S_n \mid \hat{\mathrm{x}}, D_{n-1}, A_{n-1}).$$

The principal challenge in implementing this dialog manager is again the data sparseness problem in determining the cost function $C(A, D_n)$ and the discourse model $P(S_n \mid w, D_{n-1}, A_{n-1})$. In our current implementation for

multimodal interaction such as MiPad, the data sparseness problem in both SLU and dialog management is partially by-passed using empirical rules in approximating the probabilities formulated above. One key element in this implementation is the (deterministic) semantic grammar-based robust parser that is used to map from w to S_n with the help of the dialogue state (D_{n-1}, A_{n-1}). The grammar will need to be generalized to a probabilistic one in order to compute the necessary probability terms in the unified framework discussed above. We now describe our recent research on the semantic based robust understanding that is in the current implementation of the SLU system in MiPad design.

C. Robust parser and SLU in MiPad

Since the *Tap & Talk* interface in MiPad explicitly provides dialog state (tapped field) information already, dialogue management plays relatively minor role, compared with the SLU, in the overall MiPad functionality. The major SLU component in MiPad is a robust chart parser, which accepts the output of the continuous speech recognizer using field-specific language models and employs field-specific grammars. In the typical MiPad usage scenario, users use the built-in MiPad microphone that is very sensitive to environment noise. With the iPaq device from Compaq as one of our prototypes, the word recognition error rate increased by a factor of two in comparison to a close-talking microphone in the normal office environment. This highlights the need not only for noise-robust speech recognition but also for robust SLU.

The MiPad SLU is modeled with domain-specific semantic grammars. Normally, semantic grammars are CFGs with non-terminals representing semantic concepts instead of syntactic categories. Our grammar introduces a specific type of non-terminals called semantic classes to describe the schema of an application. The semantic classes define the conceptual structures of the application that are independent of linguistic structures. The linguistic structures are modeled with context free grammars. In doing so, it makes the linguistic realization of semantic concepts transparent to an application; therefore the application logic can be implemented according to the semantic class structure, in parallel with the development of linguistic context free grammar. We in the past few years have developed a robust spoken language parser that analyzes input sentences according to the linguistic grammar and maps the linguistic structure to the semantic conceptual structure. Recently, we have made substantial modifications to the parser to take full advantage of the form factor of MiPad and to better support the semantic based analysis.

1) Robust Chart Parser

The robust parsing algorithm used in MiPad is an extension of the bottom-up chart-parsing algorithm. The robustness to ungrammaticality and noise can be attributed to its ability of skipping minimum unparsable segments in the input. The algorithm uses dotted rules, which are standard CFG rules plus a dot in front of a right-hand-side symbol. The dot separates the symbols that already have matched with the input words from the symbols that are yet to be matched. Each constituent constructed in the parsing process is associated with a dotted rule. If the dot appears at the end of a rule like in $\mathbf{A}\rightarrow\alpha\bullet$, , we call it a complete parse with symbol **A**. If the dot appears in the middle of a

rule like in **A→B•CD**, we call it a partial parse (or hypothesis) for A that is expecting a complete parse with root symbol **C**.

The algorithm maintains two major data structures --- A chart holds hypotheses that are expecting a complete constituent parse to finish the application of the CFG rules associated with those hypotheses; an agenda holds the complete constituent parses that are yet to be used to expand the hypotheses in the chart. Initially the agenda is empty. When the agenda is empty, the parser takes a word (from left to right) from the input and puts it into the agenda. It then takes a constituent **A**[*i*,*j*] from the agenda, where A is the root symbol of the constituent and [*i*,*j*] specifies the span of the constituent. The order by which the constituents are taken out of the agenda was discussed in [5]. The parser then activates applicable rules and extends appropriate partial parses in the chart. A rule is applicable with respect to a symbol **A** if either A starts the rule or all symbols before **A** are marked optional. The activation of an applicable rule may result in multiple constituents that have the same root symbol (the left-hand-side of the rule) but different dot positions, reflecting the skip of different number of optional rule symbols after **A.** If the resulting constituent is a complete parse, namely with the dot positioned at the end of the rule, the complete constituent is added into the agenda. Otherwise partial constituents are added into the chart; To extend the partial parses with the complete parse **A**[*i*,*j*], the parser exams the chart for incomplete constituent with dotted rule **B**[*l*,*k*] → α•**A**β for $k < i$, and constructs new constituents **B**[*l*,*j*] → α**A**β with various dot positions in β, as long as all the symbols between **A** and the new dot position are optional. The complete constituent **B**[*l*,*j*] → α**A**β• is added into the agenda. Other constituents are put into the chart. The parser continues the above procedure until the agenda is empty and there are no more words in the input sentence. By then it outputs top complete constituents according to some heuristic scores.

In [13], we distinguished three different types of rule symbols: optional symbols that can be skipped without penalty; normal symbols that can be skipped with penalty; and mandatory symbols that cannot be skipped. We found the skip of normal symbols is very problematic, because grammar authors are generally very forgetful to mark a symbol mandatory. This is also because skipping normal rule symbols often adversely increases the constituent space. This dramatically slows down the parser and results in a great number of bogus ambiguities. Therefore, in our current parser implementation, we do not skip rule symbols unless the symbols are explicitly marked as optional.

2) Special features of the robust parser

To take advantage of the MiPad form factor and better support semantic analysis, we have enhanced the parser with the four new features that are described below.

Dynamic grammar support

Dynamic grammar support provides the parser with the capability of modifying the grammar it is using on the fly. It is necessary because different users may have different data; therefore the parser should be able to customize the

grammar online. For example, users may have different and changing contact list, therefore the parser should dynamically modify the rule for the contacts of different users. Dynamic grammar support is necessary also because different dialog states need different grammars too. If MiPad is showing a New-Email card, then the ByName semantic class should only contain those names in the user's contact list, since they are the only names that the user can specify as recipients; otherwise the user has to specify an e-mail address. On the other hand, if MiPad is showing a New-Contact card, then we should use a name set with a much greater coverage, perhaps even introducing a spelling grammar.

We have devised an API for application developers to dynamically and efficiently change the grammar used by the parser. The change can be made at different granularity, from the entire grammar to every single rule. This enables the parser to adapt to different users and dialog states as appropriate.

Wildcard parsing

In a semantic grammar, some semantic concepts are free-form texts and can hardly be modeled with semantic rules. For instance, meeting subjects can hardly be predicated and modeled with semantic grammar rules. To model this type of semantic units, the parser is augmented to handle rules with wildcards like the following one:

<Meeting-Property>::=<about> <Subject:Wildcard>
<about> ::= about | regarding | to discuss

Here "<Subject:Wildcard>" represents the non-terminal semantic class "<Subject>" that can match a free-form text. "<about>" serves as a context cue that triggers the wildcard rule for "<Subject>". Anything that does not match other appropriate rules in the context and that matches the wildcard rules with an appropriate trigger will be parsed as a wildcard semantic component in the appropriate context.

Parsing with focus

The parser approximates the statistical discourse model $P(S_n \mid w, D_{n-1}, A_{n-1})$ by taking advantage of the dialogue state information (D_{n-1}, A_{n-1}) to reduce parsing ambiguities and the search space, hence to improve its performance. For example, if the parser knows that the system is in the middle of a dialog with the user, talking about the attendee of a meeting, then the parser will only parse "John Doe" as Attendee, although it could be E-mail Recipient or new Contact according to the grammar. The parser can get the context information (focus) either from the dialog manager in the general *Dr. Who* architecture or directly from the applications with the *Tap & Talk* interface. The focus is specified as a path from the root to the semantic class that the system is expecting an input for, such as Root/ScheduleMeeting/MeetingProperty/StartTime.

The parser can override the focus mode in case the user volunteers more information in mixed initiative dialogs. In this case, if the system is expecting an input of the aforementioned focus and the user speaks "from 3 to 5pm", the parser will be smart enough to identify both Start-Time and End-Time, and return the semantic class that is the closest common ancestor of the identified semantic classes, which in this case is Meeting-Property.

N-best parsing

The parser can take n-best hypotheses from the speech recognizer, together with their scores, as the input. The parser analyzes all the n-best hypotheses and ranks them with a heuristic score that combines the speech recognition score and the parsing score. The best parse will be forwarded to the application.

3) Practical issues in the parser implementation

In the parser implementation, it has been discovered that the parser often slowed down with the support of wildcard parsing. To overcome this weakness, we have redesigned the data structure, which leads to dramatic improvement in the parser efficiency. We briefly describe this and some related implementation issues below.

Lexicon representation

The lexicon contains both terminals and non-terminals. The non-terminals include names for semantic class, groups and productions. Each entry in the lexicon is represented with an ID. The lexicon can map a terminal or non-terminal string to its ID. An ID is an integer with the least signification 3 bits devoted to the type of the ID (word, semantic classes, types, productions, groups etc.) The rest bits can be used as index for the direct access to the definition of the grammar components of the specific type. Each lexical entry points to a set Φ that contains the information of the IDs of the context free rules it can activate, as well as the position of the lexical item in these rules. Set Φ is used to locate the applicable rules after a complete is taken out of the agenda.

Each non-terminal entry A has a Boolean element, specifying if it can derive a wildcard as its left-most descendant in a parse, or more formally, if $\mathbf{A} \rightarrow \text{wildcard } \alpha$. The value of this element has to be pre-computed at grammar load time, and recomputed every time after dynamic grammar modification. The speed to set this Boolean value is hence very important. Fortunately we already have the information of rules that can be activated by a symbol in set Φ. With that information, we can define the relation $\Re = \{(x,b) \mid b \Rightarrow \alpha x \beta\}$, where α is empty or a sequence of optional symbols. Then a non-terminal can derive a wildcard on its leftmost branch if and only if it is in the transitive closure of the wildcard with respect to relation $\Re$. This transitive closure can be computed in time linear to the number of non-terminal symbols.

Chart and agenda

The chart consists of a dynamic array of n elements and a dynamic programming structure of $n*(n+1)/2$ (n is the length of an input sentence) cells that corresponds to the $n*(n+1)/2$ different span of constituents. Each array element corresponds to a position in the input sentence, and it contains a heap that maps from a symbol **A** to a list of partial parses. The partial parses cover the input sentence to the position that the element represents for, and they expect a complete parse of a constituent with root symbol **A**. With this the parser can quickly find the partial parses to extend when a complete parse with root **A** is popped from the agenda; Each cell of the dynamic programming structure contains a heap that maps a grammar symbol **A** to a pointer to the complete constituent tree with root **A** and the span that the cell represents for. This enables the parser to quickly find out if a new constituent has the same root name and span as an existing constituent. If so, the parser will safely prune the constituent with lower score.

The agenda is implemented as a priority queue. An element of the queue has a higher priority if it has a smaller span and higher heuristic score. This guarantees that the parser does not miss any parses with high heuristic score.

Wildcard support

Since wildcard match is expensive, we would like to treat input words as wildcard only when it fits in the context. Therefore we added some top down guidance for the creation of a wildcard constituent. With the wildcard derivation information available for non-terminal symbols as described in 0, this can be implemented efficiently: during the parsing process, after a partial constituent with dotted rule $\mathbf{A} \rightarrow \alpha \bullet \mathbf{B} \beta$ is added to the chart, if **B** can derive a wildcard on its leftmost branch, we then set a flag that allows the next input word to be introduced as a wildcard.

After we introduce a wildcard to the parser, theoretically it can build *m* different wildcard constituents with different coverage, where *m* is the number of remaining words in input. This adversely increases the search space drastically, since each of these wildcard constituents can be combined with other constituents to form much more constituents. Instead of generating these m constituents, we assume that the wildcard only covers a single word. After the parser has built the parse for the complete sentence, it expands the wildcard coverage to all the skipped words adjacent to the word covered by a wildcard in the initial parse. The parser always prefers non-wildcard coverage to wildcard coverage. So wildcard will be used only when there is no no-wildcard parse of the input segment that fits the context.

D. Machine-aided grammar learning and development for SLU

The SLU component implemented in MiPad is a variation of the general semantic-based robust understanding technology. The technology has been widely used in human/machine and human/human conversational systems. For example, many research labs have used it in the DARPA-sponsored Airline Travel Information System (ATIS) evaluations. Such implementations have relied on manual development of a domain-specific grammar, a task that is time-consuming, error-prone and requires a significant amount of expertise. If conversational systems are to be a

mainstream, it becomes apparent that writing domain-specific grammars is a major obstacle for a typical application developer. Recently researchers have been working on tools for rapid development of mixed-initiative systems, but without addressing the problem of grammar authoring per se. Also, other researchers have developed tools that let an end user refine an existing grammar, which still relies on an initial grammar and also assumes that the developer has a good knowledge of language structures.

On the other hand, automatic grammar inference has also attracted the attention of researchers for many years, though most of that work has focused on toy problems. Applications of such approaches on grammar structure learning for natural language have not been satisfactory for natural language understanding application. This has been due to the complexity of the problem. That is, the available data will typically be sparse relative to the complexity of the target grammar, and there has not been a good generalization mechanism developed to correctly cover a large variety of language constructions.

Therefore, instead of aiming at an ambitious empirical automatic grammar inference, we focus in our research on an engineering approach that could greatly ease grammar development by taking advantage of many different sources of prior information. In doing so, a good quality semantic grammar can be derived semi-automatically with a small amount of data.

1) Multiple information sources used for constructing semantic grammar

A semantic CFG, like a syntactic CFG, defines the legal combination of individual words into constituents and constituents into sentences. In addition, it also has to define the concepts and their relations in a specific domain. It is this additional dimension of variation that makes it necessary to develop a grammar for every new domain. While it is not realistic to empirically learn structures from a large corpus due to technical and resource constraints, we can greatly facilitate grammar development by integrating different information sources to semi-automatically induce language structures. These various information sources are described below.

Domain-specific semantic information

As we mentioned earlier at the beginning of this section, we use the semantic schema to define the entity relations of a specific domain in our multi-modal human-computer interaction research. The domain specific semantic information in the schema can be incorporated into a CFG to model the semantic constraints. Since the semantic schema is used for many different purposes, from dialogue modeling to multi-modal input integration, it has to be developed in the first place in a multi-modal application; therefore it is not an extra burden to use it in grammar learning. Due to its language- or expression-independency, semantic schema is easy to author by a developer with good knowledge of an application. For the MiPad's calendar domain described in [9], the schema contains 16 concepts (semantic object constituents) with fewer than 60 slots. This is two orders of magnitude lower than the ~3000 CFG rules for ~1000 nonterminals. Such a semantic schema can be easily developed within a couple of hours.

Grammar library

Some low level semantic entities, such as date, time, duration, postal address, currency, numbers, percentage, etc, are not domain-specific. They are isolated universal building blocks that can be written once and then shared by many applications. Grammar libraries can greatly save development time, and we have used them extensively in developing MiPad grammars.

Annotation

We can also get developers involved to annotate the data against the schema. For example, the sentence "invite Ed to the meeting with Alex" is annotated against the schema as follows:

```
<AddAttendee text="invite Ed to the meeting with Alex">
    <ApptByAttributes text="the meeting with Alex">
        <People text="Alex"/>
    </ApptByAttributes>
    <People text="Ed"/>
</AddAttendee>
```

Here, each XML tag is the name of a semantic class in the schema, and the sub-structures are the members of semantic objects that fill in the slots of a semantic class. The annotation is surface-structure (i.e., linguistic expression) independent --- different sentences that convey the same meaning would have the same annotation. The use of the grammar library also eases the annotation process since we do not have to annotate to the very bottom level of concepts.

Syntactic constraints

The final source of information for semantic grammar development is the syntactic constraints, as domain specific language must follow the syntactic constraints of the language. Some simple syntactic clues, for example, part-of-speech constraints, are used to reduce the search space in grammar development.

2) Growing semantic grammar

Inherit semantic constraints from schema

An assumption we made is that the linguistic constraints that guide the integration of smaller units into a larger chunk is an invariant for the subset of the natural language used in human-computer interaction. It is only the domain-specific semantics and linguistic expressions for concepts that can vary. This allows us to create a template CFG that inherits the semantic constraints from the semantic schema. For example, the two concepts in the previous example can be automatically translated to the following template CFG:

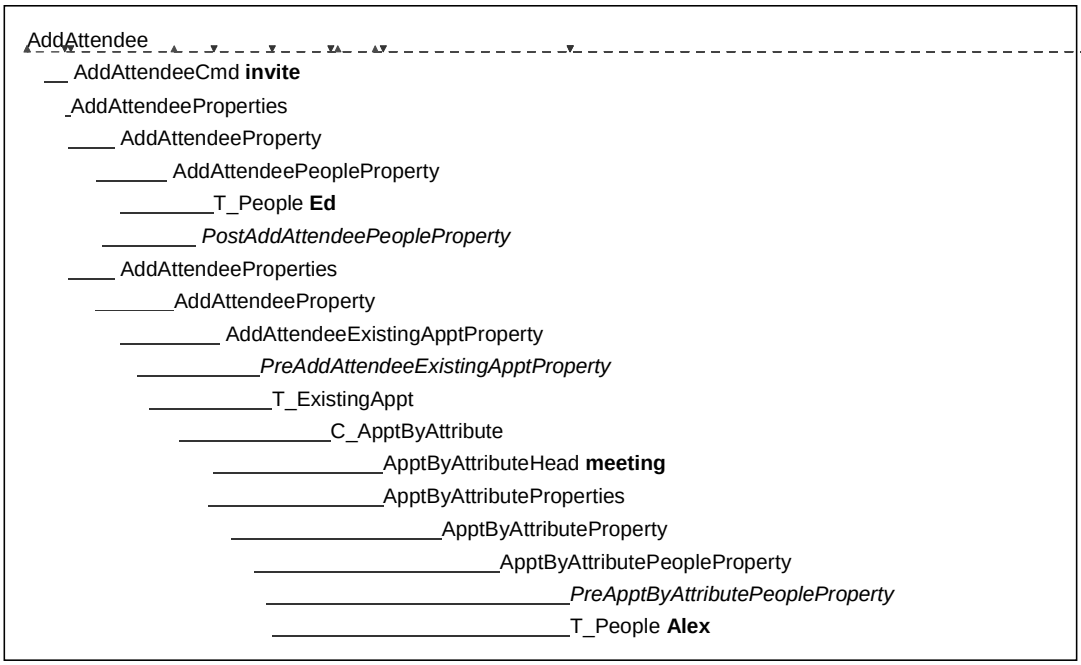
<T_ExistingAppt> ➔ <C_ApptByAttributes> (1)

<C_ApptByAttributes> ➔ {<ApptByAttributeMods>} <ApptByAttributeHead> {<ApptByAttributeProperties>} (2)
<ApptByAttributeProperties> ➔ <ApptByAttributeProperty> {<ApptByAttributeProperties>} (3)
<ApptByAttributeProperty> ➔ <ApptByAttributePeopleProperty> |
<ApptByAttributeStartTimeProperty> |
<ApptByAttributeEndTimeProperty> (4)
<ApptByAttributePeopleProperty> ➔ {<PreApptByAttributePeoplePropertyp>} <T_People> {<PostApptByAttributePeoplePropertyp>} (5)
<ApptByAttributeHead> ➔ NN (6)
<PreApptByAttributePeopleProperty> ➔ .* <T_UpdateAppt> ➔ <C_AddAttendee>
<C_AddAttendee> ➔ <AddAttendeeCmd> {<AddAttendeeProperties>}
<AddAttendeeCmd> ➔ .*
<AddAttendeeProperties> ➔ <AddAttendeeProperty> {<AddAttendeeProperties>}
<AddAttendeeProperty> ➔ <AddAttendeePeopleProperty> | <AddAttendeeExistingApptProperty>
<AddAttendeeExistingApptProperty>➔ {<PreAddAttendeeExistingApptProperty>} <T_ExistingAppt> {<PostAddAttendeeExistingApptProperty>
<AddAttendeePeopleProperty> ➔ {<PreAddAttendeePeopletProperty>} <T_People> {<PostAddAttendeePeopletProperty>}
<PreAddAttendeeExistingApptProperty ➔ .*
<PostAddAttendeeExistingApptProperty ➔ .*
<PreAddAttendeePeopleProperty ➔ .*
<PostAddAttendeePeopleProperty ➔ .*

Here an *entity*, like ApptByAttributes, consists of a head, optional (in braces) modifiers that appear in front of the head (e.g. "*Alex's* meeting"), and optional properties that follow the head (e.g. "meeting *with Alex*") (rule 2). Both modifiers and properties are defined recursively, so that they finally incorporate a sequence of different slots (rules 3-4). Each slot is bracketed by an optional preamble and postamble (rule 5). The heads, slot preambles and postambles are originally placeholders (.*). Some placeholders are specified with part-of-speech constraints --- e.g., head must be a NN (noun). For a *command* like AddAttendee, the template starts with a command part <AddAttendeeCmd>, followed by <AddAttendeeProperties>. The rest is very similar to that of the template rules for an entity. The template sets up the structural skeleton of a grammar. Hence the task of grammar learning becomes to learn the expressions for the pre-terminals like heads, commands, preambles, etc. The placeholders, without any learning, can match anything and result in ambiguities. When the learned grammar is used in our experiments, the placeholders were allowed to match any input string with a large penalty. The non-terminal <T_People> is application dependent and therefore will be provided by the developer (in the form of a name list in this example).

Annotation: Reducing the Search Space

The annotation reduces the search space for the rewriting rules for the pre-terminals: the annotated slots serve as the divider that localizes the learning space. For example, with the semantic annotation just described and the template CFG, our robust parser can obtain the partial parse shown below:



Formatted: Font: 10 pt
Deleted:
Deleted:
Deleted:
Formatted: Font: 10 pt
Deleted:
Deleted:
Formatted: Font: 10 pt
Deleted:
Deleted:
Formatted: Font: 10 pt
Deleted:
Formatted: Font: 10 pt

where the pre-terminals in *italic* are place-holders that are not matched with any word in the sentence, because neither the template grammar nor the annotation provides sufficient information for the correct decision. The terminals in bold face are matched to the pre-terminals according to the template grammar or the annotation. For example, *Ed* and *Alex* are respectively attached to the two T_People positions due to the constraints from the annotation. *Meeting* is associated with ApptByAttributeHead because it is the only remaining NN found by POS tagger and the template grammar requires that the head be a NN. *Invite*, which appears in front of *Ed*, can match both AddAttendeeCmd and PreAddAttendeePeopleProperty. Since the latter is optional, it is matched against the former. The remaining words, *to*, *the*, and *with*, cannot be deterministically aligned to any pre-terminals by the parser. However, given the partial parse tree, they can only be aligned with those pre-terminals in italic. This effectively reduces the search space for possible alignment.

Specializing the inherited template CFG by alignment of pre-terminal and text

After obtaining the parse tree with the template grammar and the annotation for the above example sentence, syntactic clues are then used to align the remaining words that are not covered in the parse tree. Prepositions and determiners can only combine with the word behind them, hence "to the" cannot align with the pre-terminal *PostAddAttendeePeopleProperty*. This leaves *PreAddAttendeeExistingApptProperty* the only choice. For the same

reason, *PreApptByAttributeStartTimeProperty* is the only pre-terminal that *with* has to be aligned with. Therefore we can induce the following rules (which will be added to the existing template CFG rules):

PreAddAttendeeExistingApptProperty ➔ *to the*
PreApptByAttributePeopleProperty.➔ *with*

Sometimes syntactic clues are not enough to resolve all the ambiguities. In this case, the system prompts the developer for the right decision. We are currently working on an alignment model based on the Expectation-Maximization algorithm to do this automatically.

VI. MIPAD USER INTERFACE DESIGN AND EVALUATION

MiPad takes advantage of the graphical display in the UI design. The graphical display simplifies dramatically the dialog management. For instance, MiPad is able to considerably streamline the confirmation and error repair strategy as all the inferred user intentions are confirmed *implicitly* on the screen. Whenever an error occurs, the user can correct it in different modalities, either by soft keyboard or speech. The user is not obligated to correct errors immediately after they occur. The display also allows MiPad to confirm and ask the user many questions in a single turn. Perhaps the most interesting usage of the display, however, is the *Tap & Talk* interface.

A. Tap & Talk interface

Because of MiPad's small form-factor, the present pen-based methods for getting text into a PDA (Graffiti, Jot, soft keyboard) are potential barriers to broad market acceptance. Speech is generally not as precise as mouse or pen to perform position-related operations. Speech interaction can also be adversely affected by the unexpected ambient noise, despite the use of denoising algorithms in MiPad. Moreover, speech interaction could be ambiguous without appropriate context information. Despite these disadvantages, speech communication is not only natural but also provides a powerful complementary modality to enhance the pen-based interface if the strengths of using speech can be appropriately leveraged and the technology limitations be overcome. In Table 2, we elaborate several cases which show that pen and speech can be complementary and used effectively for handheld devices. The advantage of pen is typically the weakness of speech and vice versa.

Pen	**Speech**
Direct manipulation	Hands/eyes free manipulation
Simple actions	Complex actions
Visual feedback	No Visual feedback
No reference ambiguity	Reference ambiguity

Table 2 Complementary strengths of pen and speech as input modalities

Through usability studies, we also observe that users tend to use speech to enter data and pen for corrections and pointing. Three examples in Table 3 illustrate that MiPad's *Tap and Talk* interface can offer a number of benefits. MiPad has a field that is always present on the screen as illustrated in MiPad's start page in Figure 7 (a) (the bottom gray window is always on the screen).

Actions	**Benefits**
Ed uses MiPad to read an e-mail, which reminds him to schedule a meeting. Ed taps to activate microphone and says *Meet with Peter on Friday.*	Using speech, information can be accessed directly, even if not visible. Tap and talk also provides increased reliability for ASR.
Ed taps Time field and says *Noon to one thirty*	Field values can be easily changed using field-specific language models
Ed taps Subject field dictates and corrects the text about the purpose of the meeting.	Bulk text can be entered easily and faster.

Table 3: Three examples showing benefits to combine speech and pen for MiPad user interface

Tap & Talk is a key feature of the MiPad's user interface design. The user can give commands by tapping the *Tap & Talk* field and talking to it. *Tap & Talk* avoids speech detection problem that are critical to the noisy environment deployment for MiPad. The appointment form shown on MiPad's display is similar to the underlying semantic objects. By tapping to the *attendees* field in the calendar card shown in Figure 7 (b), for example, the semantic information related to potential attendees is used to constrain both CSR and SLU, leading to a significantly reduced error rate and dramatically improved throughput. This is because the perplexity is much smaller for each slot-dependent language and semantic model. In addition, *Tap & Talk* functions as a user-initiative dialog-state specification. The dialog focus that leads to the language model is entirely determined by the field tapped by the user. As a result, even though a user can navigate freely using the stylus in a pure GUI mode, there is no need for MiPad to include any special mechanism to handle spoken dialog focus and digression.

B. Back Channel Communications

MiPad handles back-channel communications on the device. As a user speaks, it displays a graphical meter reflecting the volume of the recording. When the utterance is beyond the normal dynamic range, red bars are shown to instruct the user to tone down. As the host computer processes the user's utterance, a running status bar is shown. The user can click a cancel button next to the status bar to stop the processing at the host computer. If the status bar vanishes without changing the display, it indicates the utterance has been rejected either by the recognizer or by the

understanding system. MiPad's error repair strategy is entirely user initiative: the user can decide to try again or do something else.

C. User study results

Our ultimate goal is to make MiPad produce real value to users. It is necessary to have a rigorous evaluation to measure the usability of the prototype. Our major concerns are:

- "*Is the task completion time much better*?" and
- "*Is it easier to get the job done*?"

For our user studies, we set out to assess the performance of the current version of MiPad (with PIM features only) in terms of task-completion time, text throughput, and user satisfaction. In this evaluation, computer-savvy participants who had little experience with PDAs or speech recognition software used the partially implemented MiPad prototype. The tasks we evaluated include creating a new appointment and creating a new email. Each participant completed half the tasks using the tap and talk interface and half the tasks using the regular pen-only iPaq interface. The ordering of tap and talk and pen-only tasks is statistically balanced.

1) Is the task completion time much better?

Twenty subjects were included in the experiment to evaluate the tasks of creating a new email, and creating a new appointment. Task order was randomized. We alternated tasks for different user groups using either pen-only or *Tap & Talk* interfaces. The text throughput is calculated during e-mail paragraph transcription tasks. On average it took the participants 50 seconds to create a new appointment with the *Tap & Talk* interface and 70 seconds with the pen-only interface. This result is statistically significant with $t(15) = 3.29$, $p < .001$. The saving of time is about 30%. For transcribing an email it took 2 minutes and 10 seconds with *Tap & Talk* and 4 minutes and 21 seconds with pen-only. This difference is also statistically significant, $t(15) = 8.17$, $p < .001$. The saving of time is about 50%. Error correction for the *Tap & Talk* interface remains as one of the most unsatisfactory features. In our user studies, calendar access time using the *Tap & Talk* methods is about the same as pen-only methods, which suggests that pen-based interaction is suitable for simple tasks.

2) Is it easier to get the job done?

Fifteen out of the 16 participants in the evaluation stated that they preferred using the *Tap & Talk* interface for creating new appointments and all 16 said they preferred it for writing longer emails. The preference data is consistent with the task completion times. Error correction for the *Tap & Talk* interface remains as one of the most unsatisfactory features. On a seven point Likert scale, with 1 being "disagree" and 7 being "agree", participants responded with a 4.75 that it was easy to recover from mistakes.

Figure 8 summarizes the quantitative user study results on task completion times of email transcription and of making appointment, showing comparisons of the pen-only interface with the *Tap & Talk* interface. The standard deviation is shown above the bar of each performed task.

VII. SUMMARY AND FUTURE WORK

Speech is a necessary modality to enable a pervasive and consistent user interaction with computers across different devices --- large or small, fixed or mobile, and it has the potential to provide a natural user interaction model. However, the ambiguity of spoken language, the memory burden of using speech as output modality on the user, and the limitations of current speech technology have prevented speech from becoming the choice of mainstream interface. Multimodality is capable of dramatically enhancing the usability of speech interface because GUI and speech have complementary strengths as we have shown in this paper. Multimodal access will enable users to interact with an application in a variety of ways --- including input with speech, keyboard, mouse and/or pen, and output with graphical display, plain text, motion video, audio, and/or synthesized speech. Each of these modalities can be used independently or simultaneously.

Dr. Who is Microsoft's attempt to develop a speech-centric multimodal user interface framework and its enabling technologies. MiPad as we have focused on in this paper is the first *Dr. Who's* application that addresses specifically the mobile interaction scenario and it aims at the development of a consistent human-computer interaction model and component technologies for multimodal applications. Our current applications comprise mainly the PIM functions. Despite its current incomplete implementation, we have observed that speech and pen have the potential to significantly improve user experience in our preliminary user study. Thanks to the multimodal interaction, MiPad also offers a far more compelling user experience than standard voice-only telephony interaction.

The success of MiPad depends on spoken language technology and an always-on wireless connection. With upcoming 3G wireless deployments in sight, the critical challenge for MiPad remains the accuracy and efficiency of our spoken language systems since likely MiPad will be used in the noisy environment with no availability of a close-talking microphone, and the server also needs to support a large number of MiPad clients.

To meet this challenge, much of our recent work has focused on: 1) noise-robustness and transmission efficiency aspects of the MiPad system in the distributed speech processing environment, and 2) SLU with specific attention paid also to robustness as well as to automatic and high-quality application grammar development. We report our new front-end speech processing algorithm developments and some related evaluation results in this paper. Various other MiPad system components are also presented, including HMM-based speech modeling, a unified language model that integrates CFGs and N-grams for speech decoding, several key aspects of SLU (schema-based knowledge representation for the MiPad's PIM functionality, a unified statistical framework for ASR/SLU/dialogue, the robust

chart parser, and semi-automatic MiPad grammar learning), *Tap & Talk* multimodal user interface, error repair strategy, and user study results comparing the multimodal interaction and the pen-only PDA interface.

The prototype of MiPad as the first *Dr. Who* application discussed in this paper has recently been successfully transferred from the research lab to the Microsoft .NET speech product division as a client browser component in the grand Kokanee architecture. This new architecture is aimed at speech-enabling the web applications based on the Speech-Application-Language-Tag standard for multimodal (speech and GUI) interactions between end users and either mobile or fixed devices. Future research work will be focused on the next version of *Dr. Who* and its new applications aimed to provide a greater degree of intelligence and automaton to larger domains and tasks than the limited PIM task of the MiPad developed so far.

REFERENCES

[1] A. Acero and R. Stern. "Robust speech recognition by normalization of the acoustic space," Proc. ICASSP-1991, Toronto.

[2] L. Deng, A. Acero, M. Plumpe, and X.D. Huang. "Large-vocabulary speech recognition under adverse acoustic environments," Proc-ICSLP2000, Beijing, China, October 2000, Vol. 3, p. 806-809.

[3] L. Deng, A. Acero, L. Jiang, J. Droppo, and XD Huang. "High-performance robust speech recognition using stereo training data," Proc. ICASSP-2000, Vol. I, Salt Lake City, Utah, April 2001, pp. 301-304.

[4] L. Deng, J. Droppo, and A. Acero. "Recursive estimation of nonstationary noise using a nonlinear model with iterative stochastic approximation," Proc IEEE Workshop on ASRU-2001, Italy, Dec. 2001.

[5] L. Deng, J. Droppo, and A. Acero. "Robust speech recognition using iterative stochastic approximation and recursive EM for estimation of nonstationary noise," submitted to IEEE Trans. Speech and Audio Proc., 2001.

[6] J. Droppo, L. Deng, and A. Acero, "Evaluation of the SPLICE algorithm on the Aurora2 database," *Proc. EuroSpeech-2001 (web update version)*, Aalborg, Demark, 2001.

[7] H. G. Hirsch and D. Pearce, "The AURORA experimental framework for the performance evaluations of speech recognition systems under noisy conditions," *ISCA ITRW ASR2000 "Automatic Speech Recognition: Challenges for the Next Millennium*," Paris, France, September 18-20, 2000.

[8] X. D. Huang et al., "MiPad: A next generation PDA prototype", *Proc. ICSLP-2000*, Beijing China, October 2000.

[9] X. D. Huang et al. ''MiPad: A multimodal interaction prototype," *Proc. ICASSP-2001*, Vol. I, Salt Lake City, Utah, April 2001, p. 9-12.

[10] K. Wang, "Implementation of a multimodal dialog system using extended markup language," *Proc. ICSLP-2000*, Beijing, China, 2000.

[11] K. Wang, H. Hon, A. Acero, "A distributed understanding and dialog environment using Web infrastructure," submitted to IEEE Trans. Speech and Audio Proc., 2001.

[12] Y. Wang, M. Mahajan, X. Huang, “A unified context-free grammar and N-gram model for spoken language processing”, *Proc. ICASSP-2000*, Istanbul, Turkey, 2000.

[13] Y. Wang, “A robust parser for spoken language understanding,” *Proc. Eurospeech-1999*, Budapest, Hungary, 1999.

[14] Y. Wang and A. Acero, “Grammar learning for spoken language understanding,” *Proc. IEEE Workshop on ASRU-2001*, Madonna di Campiglio, Italy, Dec. 2001

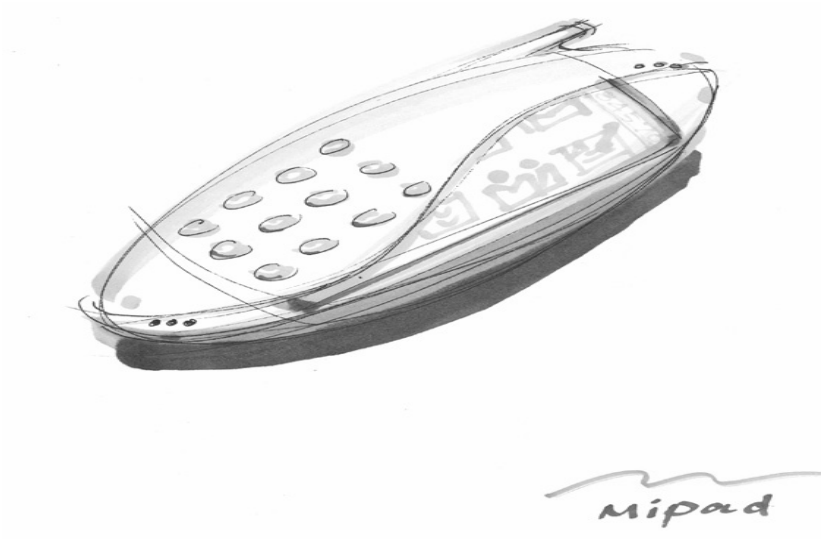


Figure 1: One of MiPad's industrial design template

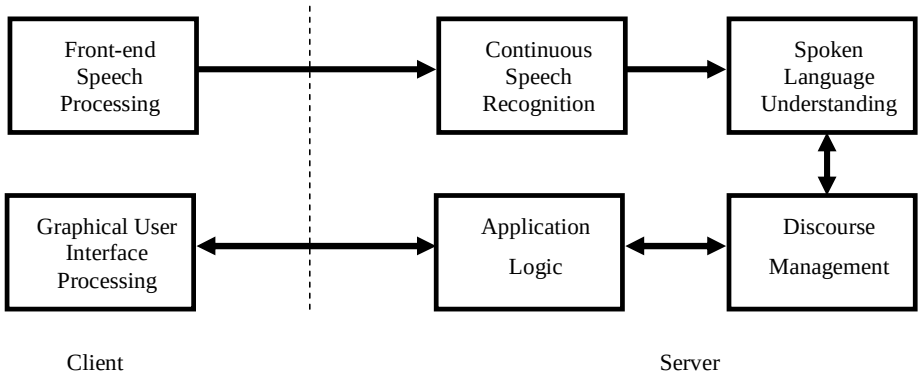


Figure 2: MiPad's client-server (peer-to-peer) architecture. The client is based on a Windows CE iPAQ, and the server is based on a Windows server. The client-server communication is currently based on the wireless LAN.

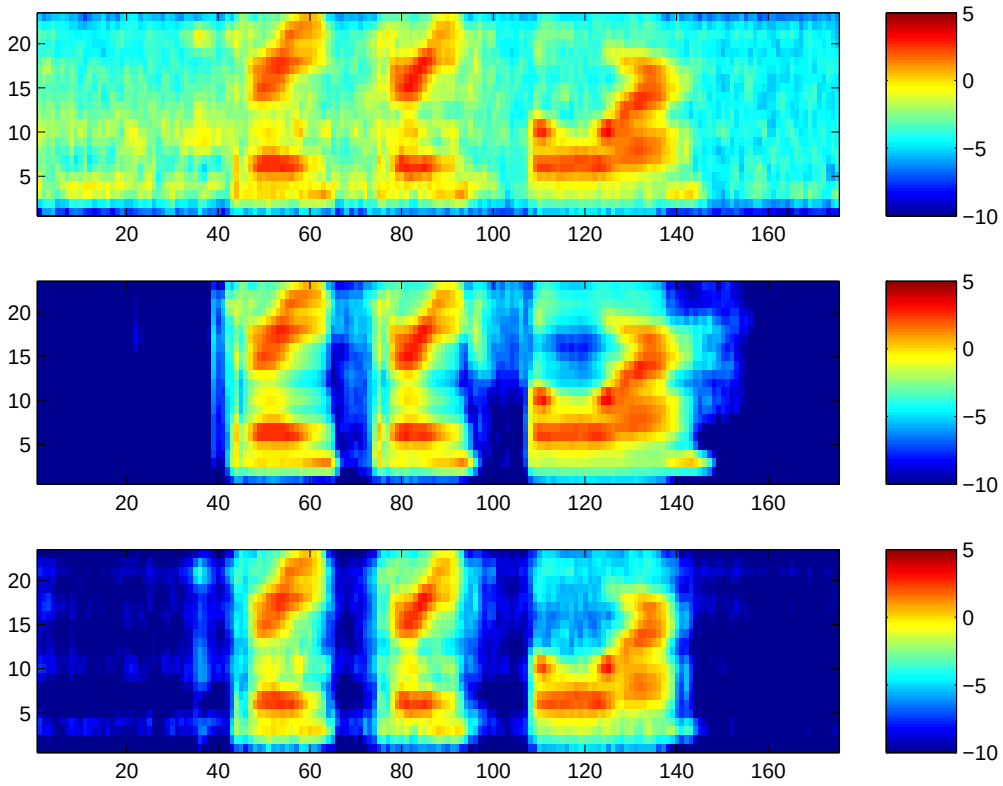


Figure 3: Noise-normalized SPLICE denoising using the iterative stochastic algorithm for tracking nonstationary noise in an utterance of the Aurora2 data with an average SNR=10dB. From top to bottom panels are noisy speech, clean speech, and denoised speech, all in the same spectrogram format.

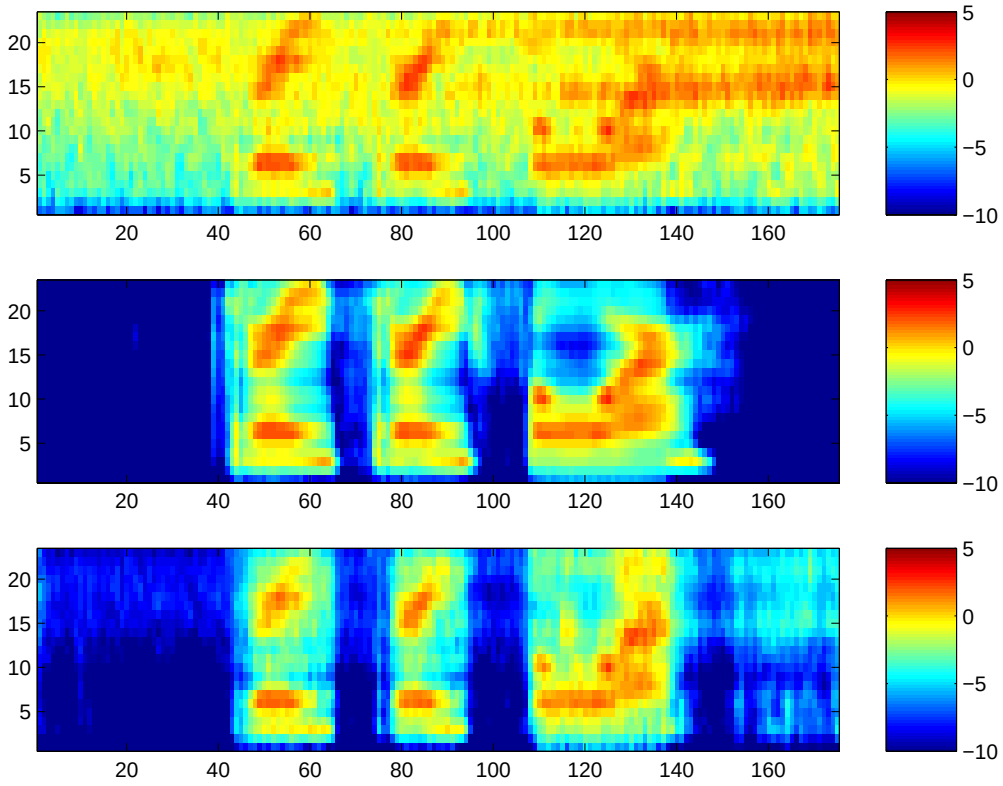


Figure 4: Noise-normalized SPLICE denoising using the iterative stochastic algorithm for tracking nonstationary noise in an utterance of the Aurora2 data with an average SNR=5dB. From top to bottom panels are noisy speech, clean speech, and denoised speech, all in the same spectrogram format.

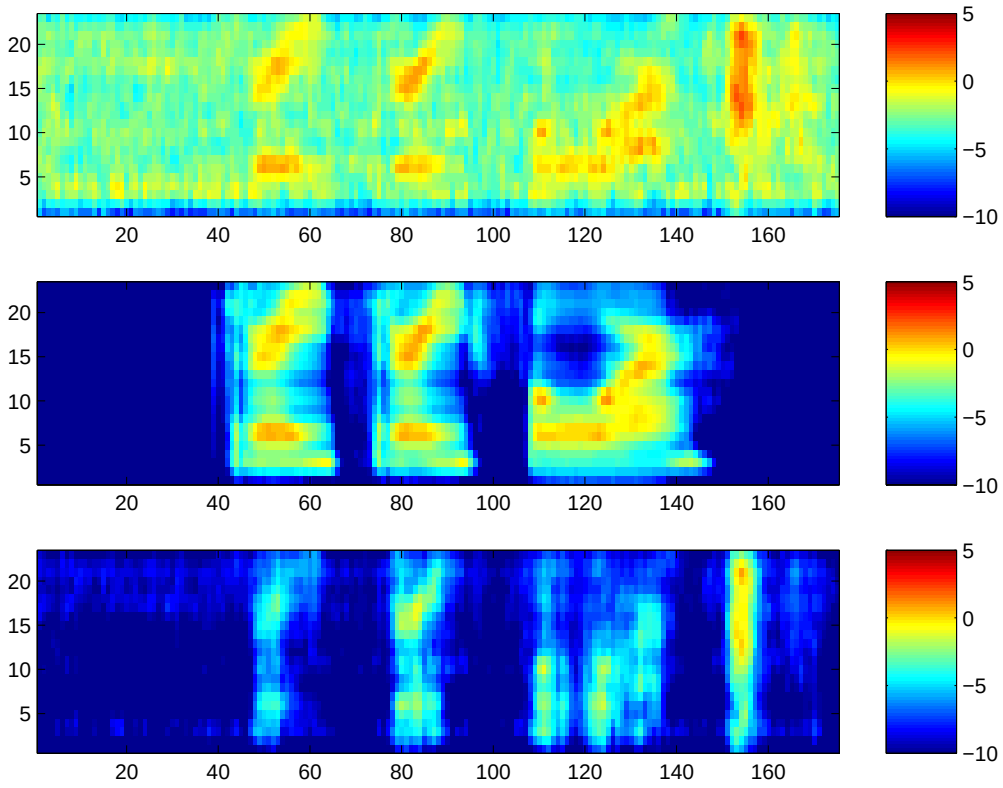
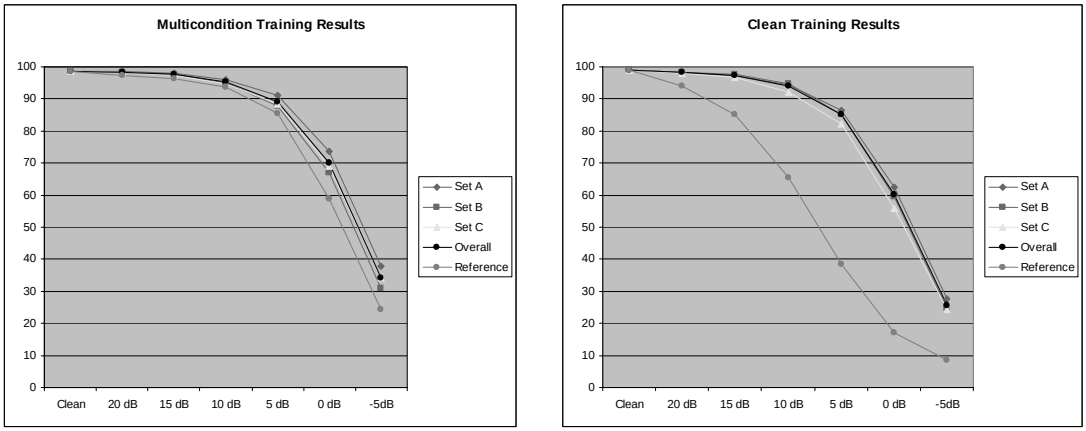
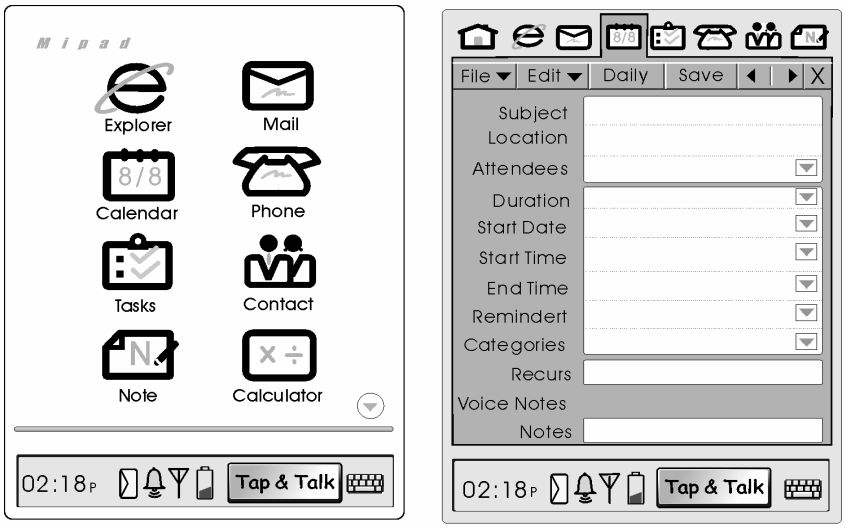


Figure 5: Noise-normalized SPLICE denoising using the iterative stochastic algorithm for tracking nonstationary noise in an utterance of the Aurora2 data with an average SNR=0dB. From top to bottom panels are noisy speech, clean speech, and denoised speech, all in the same spectrogram format.



(a) (b)

Figure 6: Full set of noise-robust speech recognition results in the September-2001 Aurora2 evaluation, using the dynamic and noise-normalized SPLICE with the noise estimation obtained from iterative stochastic approximation; Sets A, B, and C are separate test sets with different noise and channel distortion conditions. In (a) are the recognition rates using multi-condition training mode where the denoising algorithm is applied to the training data set and the resulting denoised Mel-cepstral features are used to train the HMMs. In (b) are the recognition rates using the "clean" training mode where the HMMs are trained using clean speech Mel-cepstra and the denoising algorithm is applied only to the test set. Reference curves in both (a) and (b) refer to the recognition rates obtained with no denoising processing.



(a) (b)

Figure 7: Concept design for (a) MiPad's first card and (b) MiPad's calendar card

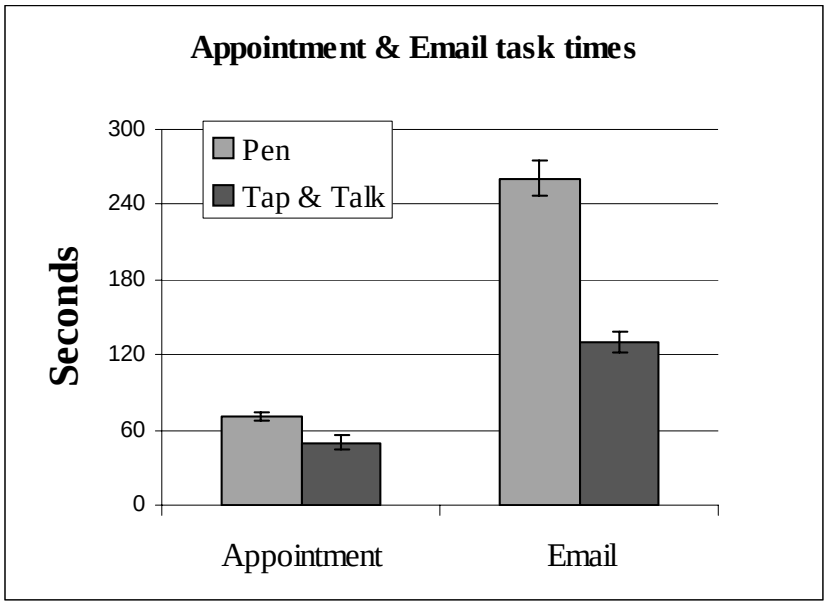


Figure 8: User Study on task completion times of email transcription and of making appointment, showing comparisons of the pen-only interface with the Tap and Talk interface. The standard deviation is shown above the bar of each performed task.