SPEECH TECHNOLOGY INTEGRATION AND RESEARCH PLATFORM: A SYSTEM STUDY

Qiru Zhou, Chin-Hui Lee, Wu Chou and Andrew Pargellis

Multimedia Communication Research Laboratory
Bell Laboratories, Lucent Technologies
600-700 Mountain Avenue
Murray Hill, NJ 07974, USA
{qzhou, chl, wuchou, anp}@research.bell-labs.com

ABSTRACT

We present a generic speech technology integration platform for application development and research across different domains. The goal of the design is two-fold: On the application development side, the system provides an intuitive developer's interface defined by a high level application definition language and a set of convenient speech application building tools. It allows a novice developer to rapidly deploy and modify a spoken language dialogue application. On the system research and development side, the system uses a thin, 'broker' layer to separate the system application programming interface from the service provider interface. It makes the system easy to incorporate new technologies and new functional components. We also use a domain independent acoustic model set to cover US English phones for general speech applications. The system grammar and lexicon engine creates grammars and lexicon dictionaries on the fly to enable a practically unrestricted vocabulary for many recognition and synthesis applications.

1. INTRODUCTION

In recent years, high performance speech processing technologies, such as automatic speech recognition (ASR) and text-to-speech synthesis (TTS)[4], are becoming available for many applications, thanks to many years of research and the exponentially increasing performance-to-price ratio of general computing power. Now a software based speech application sub-system may reside in a commonly available general purpose workstation or personal computer with real-time speech application capability, without any special-purpose hardware support.

While there is a continuous progress in speech processing core technologies (such as speech recognition, speech synthesis, natural language understanding, and dialogue processing[1], [3], [5]), it is still nontrivial to design a speech-enabled prototype system for experimenting and studying a wide range of real world applications. For most of the available speech processing platforms, a team of experts on software, speech technologies, audio and telephony interface is usually needed to develop a domain-specific speech application. This shortcoming severely limits the deployment of speech enabled applications. On the other hand for the research side, we need a generic, flexible speech technology integration platform to study research and practical issues in real-world speech processing problems, such as speech recognition robustness, speech synthesis naturalness in discourse, human-machine spoken dialogue design, etc.

To address these issues, we propose a technology integration platform with the following set of system design goals:

- Enable rapid application development
- Create a research tool set to study human-machine speech interface and spoken dialogue design
- Integrate high performance speech processing
- Interface to real world applications, such as transaction control and information access
- Applications be both domain dependent and independent
- Scaleable multi-channel processing
- Open architecture to allow third party system component to integrate into the system
- Reusable component design
- Inter-operable across heterogeneous systems

2. SYSTEM ARCHITECTURE

From the system design goals, we conclude that the platform should be based on a distributed, client-server architecture. This not only makes the system scaleable and easy to add new function components, but also makes it possible for efficient application development and integration. Figure 1 illustrates the functional block diagram of the system.

Our current system comprises of the following components:

Server side:

- ASR server: The real-time speech recognition engine.
- TTS server: The text-to-speech synthesis engine. It also provides a lexicon interface to output phone-based lexical transcriptions for a given vocabulary on the fly for speech recognition task composition.
- Audio server: The audio server provides fine granularity control on speech audio I/O streams for advanced speech processing functions, including echo cancellation, barge-in, etc.
- Database server: The application databases and database management tools.
- GDC server: The grammar/dictionary factory for recognition task composition[9].
- Telephone/audio interface: The full duplex speech audio interface. The system audio interface includes desktop audio interface and telephony interface.

Interface and broker layer:

- SPI: The service provider's interface. It is the system server side interface to connect server components.
- Resource manager/proxy server: The system resource management layer connects client service requests to proper servers and manages the system resource distribution.

- API: The system application programming interface for speech technology research and advanced application prototyping.

Client side:

- Application (client): A speech dialogue application defined by the system application definition language (ADL).
- Dialogue/application manager: The system dialogue manager that interprets the application actions described by a ADL script and request system services and actions. Each of the client applications creates a dialogue manager instance to handle its own needs.

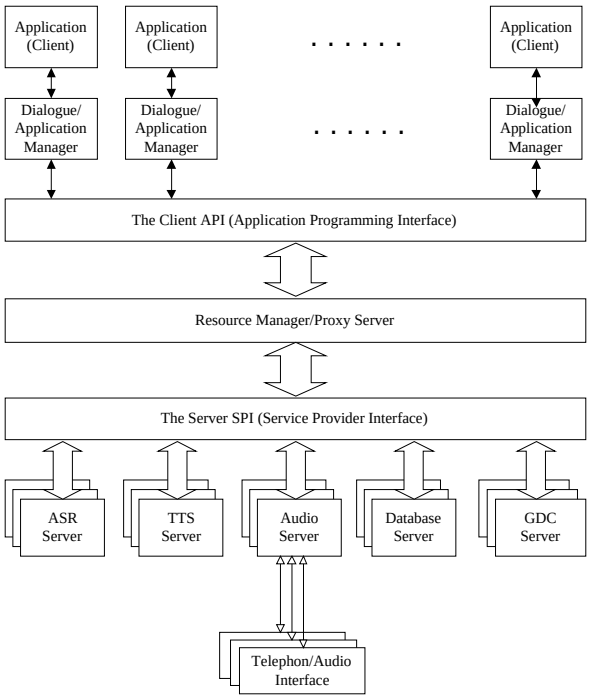

Figure 1. Speech Technology Integration Platform

In order to make the system inter-operable across heterogeneous computer systems, we designed a simple server SPI protocol on top of the TCP/IP protocol for the system server components which is machine independent (i. e., the servers talk to any one who communicates based on the system SPI protocol). Since the middle layer between applications and servers are usually thin, we use a high level interpreted language (Perl, in our implementation) to write the resource management/proxy middle layer and the application dialogue engine. This approach make it easy for rapid system prototyping and to be portable to a large range of computer systems. A new component may be integrated to the system by just implementing the TCP/IP SPI protocol and adding new API commands without rebuilding the entire system.

3. SPEECH RECOGNITION SERVER

The real-time speech recognition engine integrated into the system is a server developed at Bell Labs. The main features of the recognizer are:

1. Runtime configurable: A client may initialize a private instance of the recognition engine by using the ASR server initialization/configuration commands to load acoustic models, language models, and recognition parameters to configure the ASR server as needed. We also used the system to construct applications on languages other than English, such as Spanish, Mandarin Chinese, etc.
2. High accuracy speech decoder: The ASR engine uses a set of context dependent cross-word phone models trained on a corpus of general US English phrases as the default[8] model set. It covers common English sounds with high resolution. An optimal N-best algorithm[6] is used to produce multiple recognition hypotheses. The engine may be configured with the Bell Labs WAVE decoder[2] to save up to 95% of memory usage for large vocabulary recognition tasks. Probabilistic *n-gram* and deterministic finite state grammar language models may be used for most large vocabulary applications.
3. Unrestricted vocabulary: We use the TTS server – the Bell Labs TTS system[4] to transcribe recognition task vocabulary on the fly to compose a dictionary. Current test results shows that this transcription system produces reasonable word lexicons for most of our experiments. For best results, fine tuning of some words, such as adding alternative pronunciations, may be needed. If the grammatical specification is also given, a grammar compiler[9] allows composition of the required recognition grammar on the fly. These features enable the system to be used directly in applications where the language constraints can not be pre-built, such as speech enabled web browsing[11] in which the web pages are not seen by the recognizer before runtime.
4. Recognition task caching: multiple acoustic model sets and language models may be pre-loaded for rapid run-time recognition *context switching*. The model set may be shared among tasks to save memory. This feature is useful for spoken dialogue transaction systems where multiple sub-tasks are repeatedly used and fast task switching is necessary.
5. Robust speech recognition: various channel equalization and smoothing techniques are used to make the decoding more robust in noisy environment.

Our standard English sub-word model set is comprised of 1117 right context dependent cross-word phone units[8]. The model set was trained on a telephone speech corpus of phonetically balanced English phrases using a minimum error discriminative training algorithm[8]. The feature vector dimension is 39, which includes 12 LPC-derived cepstral coefficients, 12 delta LPC cepstral coefficients, 12 delta-delta LPC cepstral coefficients, a normalized log energy element and its time derivatives.

Since the system is designed to be used under a wide signal conditions, the ASR server has the following options to perform channel equalization:

1. Real-time signal bias removal[10] (RT-SBR): The ASR server creates initial, RT-SBR code books on the fly based on a given acoustic model set, then adapts the code book parameters to speech channel environment.

2. Real-time cepstral mean subtraction (RT-CMS): This is a CMS algorithm[10] which estimates the cepstral mean vector by using a stochastic approximation method.
3. Look-ahead sliding window signal smoothing: This algorithm reduces signal burst noise and spikes by averaging the signal feature vector in a look-ahead time window. This simple technique is especially useful for short utterances and it effectively reduces ASR error rate by 50% on an American company name recognition task (about 7000 company names).
4. DC bias removal: This adaptive algorithm removes DC biases introduced by some electrode microphones on certain system audio inputs.

We found items 3 and 4 are necessary for desktop open microphone input under noisy environments.

4. APPLICATION DEFINITION LANGUAGE

Most of the spoken dialogue tasks can be described as finite state machines of dialogue states and transitions between states. In each of the dialogue states, the system and the user conduct a specific dialogue. Based on the dialogue output, the system performs a specific semantic action or actions (such as database search or update), then transits to the next state. If a terminal state is reached, the application transaction is completed. Exceptional termination states have to be designed to handle system errors, user termination, dialogue interrupts, voice repair, system reprompts, etc.

An ADL state is defined with the following information:

```
state: {
       $StateName;
       $ttsStrings;
       $sndFiles;
       $recTask;
       %recMappingArray;
       @nextStateList;
       &actionFuncs;
       &amInterpreter;
}
```

where each state is identified by its name string. A terminal state has a process termination function as its action function. If we want the computer to say something at this dialogue state, we may concatenate text strings to a variable $ttsStrings, or specify sound files in the $sndFiles list to play back. $recTask specifies the ASR task to let the computer start speech recognition. We use an associative array %recMappingArray to remap the recognition result strings to the key word or key concept for this node. @nextStateList gives the destination state list from this node. An application may define a list of action functions if entering this state. Finally, we start the application dialogue engine, &amInterpreter, to process this state.

One of the advantages of using an interpreter engine to control the dialogue is that the application may modify the state contents at run time. This is useful for commercial transaction systems where a customer may has his/her own profile to specify personal processing needs.

The dialogue interpreter engine takes the information of a state and processes it when the transaction enters the state.

5. EXAMPLE: SPOKEN BANKING DIALOGUE

As an illustration, we present a prototype banking system (Figure 2) as an example.

This application demonstrates a simple banking system. A customer may access this system if there is a user profile in the database for access and the user passes the authentication. There are three services available in the system: *get account balance*, *transfer funds*, and *list transactions*. If *transfer funds* is selected, we need an extra confirmation step before processing the transfer.

This banking application is defined by using the following states:

Start: initialization, and load shared data.
Login: ask user id, identify the user and get the user profile.
Authentication: check user pin/password.
ServiceSelect: select a service.
GetBalance: check account balance.
TransferFunds: transfer money against user balance.
ListTransactions: list user transaction history.
AccessAcount: access account database.
ConfirmTransfer: verify transfer process.
ProcessTransaction: apply transfer transaction.
AnotherTransaction: prompts if the user wants another transaction.
End/CustomerService: terminal states.

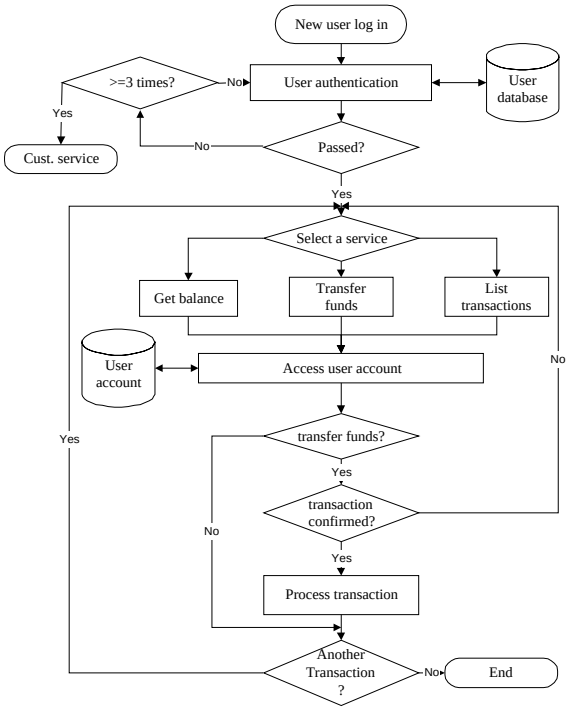

Figure 2. A bank system example

The following shows an ADL specification in Perl script for the ServiceSelect state. It demonstrates the ease of constructing a dialogue session with a high level script language like Perl.

```
ServiceSelect:{
  $StateName =" ServiceSelect";
  $ttsStrings ="Say either 'balance', \
       'transfer fund', or 'list \
       transactions'\n";
  $recTask = "gr_bank1.dat";
  %recMappingArray ={
       "balance", "balance",
       "transfer fund", "transfer",
       "list transactions", "list",
       };

  @nextStateList = ("GetBalance",
                    "TransferFunds",
                    "ListTransactions");
  &amInterpreter;
}
```

6. CONCLUSION

By integrating major speech processing components (ASR and TTS, etc.) into a network-based platform, it gives application builders a new level of convenience and flexibility to create advanced speech processing applications. A high level application definition language enables researchers and developers to quickly prototype a speech application and make it easy to modify.

This system has been successfully tested on various heterogeneous distributed systems to build speech applications such as telephony network intelligent agent, voice user interface, speech dialogue banking transaction, speech enabled web browsing[11], spoken language command control, etc.

7. ACKNOWLEDGMENTS

This work was based on years of collaborative research and development in Bell Labs and AT&T/Lucent speech product development units. In particular, the authors would like to thank E. Pinson, R. Ritenour, B. Stern of AT&T Research, and M. Brown, F. Soong, M. Baldwin, G. Erhart, A. Saad, R. Sproat of Lucent Bell Labs, for their invaluable contributions and fruitful discussions.

8. REFERENCES.

[1] Sutton, S., Novick, D., Cole, R., and Fanty, M., "*Building spoken-dialogue systems*," *Proc. ICSLP-96*, Philadelphia, Oct. 1996.

[2] Burhke, E., Chou, W., and Zhou, Q., "*A Wave Decoder for Continuous Speech Recognition*," *Proc. ICSLP-96*, Philadelphia, Oct. 1996.

[3] Colton, D. et. al., "*A Laboratory Course for Designing and Testing Spoken Dialogue Systems*," *Proc. ICASSP-96*, Atlanta, May 1996.

[4] Sproat, R. W., and Olive J. P., "*Text-to-Speech Synthesis*," *AT&T Technical Journal* **74**(2), 35-44, 1995.

[5] Goddeau, D. et. al., "*GALAXY: A Human-Language Interface to On-line Travel Information*," *Proc. ICSLP-94*, Yokohama, Sept. 1994.

[6] Chou, W. et. al., "*An Algorithm of High Resolution and Efficient Multiple String Hypothesization for Continuous Speech Recognition Using Inter-Word Models*," *Proc. ICASSP-94*, 1994.

[7] Lee, C-H. et. al., "*Improved Acoustic Modeling for Large Vocabulary Continuous Speech Recognition*," *Compute Speech and Language* 6, 103-127, 1992.

[8] Lee, C-H. et. al., "*A Study on Task-Independent Subword Selection and Modeling for Speech Recognition,*" *Proc. ICSLP-96*, Philadelphia, Oct. 1996.

[9] Brown, M. K. and Wilpon, J. G., "*A Grammar Compiler for Connected Speech Recognition*," *IEEE Trans. ASSP*, Vol. 39, No. 1, 1991.

[10] Rahim, M. et. al., "*Signal Conditioning Techniques for Robust Speech Recognition,*" *IEEE Signal Processing Letters*, Vol. 3, No. 4, 1996.

[11] Jayant, N. "*Human Machine Interaction by Voice and Gesture,*" *ICASSP-97*, Munich, April 1997.