Rapidly Developing Spoken Chinese Dialogue Systems with the d-Ear SDS SDK

Xiaojun Wu[†], *Thomas Fang Zheng*[†], *Michael Brasser*[‡], *Zhanjiang Song*[‡]

[†]Center for Speech Technology (CST), State Key Laboratory of Intelligent Technology and Systems, Tsinghua University, Beijing, 100084, China
[‡]Beijing d-Ear Technologies Co. Ltd., Beijing, 100085, China
[†][wuxj, fzheng]@cst.cs.tsinghua.edu.cn, [‡][mbrasser, zjsong]@d-Ear.com

Abstract

Developing a spoken dialog system is typically time-consuming, and must often be accomplished using difficult-to-learn professional technologies. Most existing toolkits use statistical semantic parsers and model a dialogue interaction as a finite-state network. However, for developing flexible spoken Chinese dialogue systems, these toolkits have several problems. A new toolkit named the d-Ear SDS SDK is introduced here. The SDK suggests a multi-session dialogue system framework with a powerful semantic parser specially designed for spoken Chinese understanding, and a powerful dialogue manager providing non-finite-state dialogue control. To set up a new dialogue system, the developer can customize all the system modules with domain-specific information and operations, using the d-Ear SDS Studio to save time. Using the SDK, we have built several dialogue systems with excellent performance in a very short time.

1. Introduction

Spoken dialogue systems provide information in a natural and convenient way, and thus are becoming more and more useful in daily life. However, developing a spoken dialog system can take months or years of work, and must often be accomplished using difficult-to-learn professional technologies. To solve these two problems, there have been quite a few toolkits introduced for prototyping dialogue systems [1-4].

A typical toolkit for dialogue system development includes a preferred dialogue system framework and several tools for 1) customizing a task-specific semantic parser and 2) implementing the desired dialogue control. Helper tools are often included as well, such as tools for data collection and processing, system debugging, and so on.

Most toolkits use a statistical semantic parser/speech recognizer, whose performance greatly depends on the availability and size of tagged corpora, to extract information from the user's utterances [2, 3, 5]. To train an acceptable parser/recognizer for special domain tasks, sufficient data should be collected and tagged carefully, which often requires much time and manual work. Furthermore, in many cases such a parser/recognizer still cannot correctly handle pauses or corrections by the user. Even in systems with a pure text interface, many users won't strictly follow the required input format.

Spoken Chinese dialogue systems also have to solve language-specific problems, such as those related to the monosyllabic structure, open vocabulary nature, and flexible wording structure of Chinese [6]. These characteristics cause difficulties in segmentation and syntactic parsing. In text-interfaced dialogue systems, the user's utterances can involve incorrect Chinese characters, English letters, Arabic numerals and various symbols of single- or double-byte length.

In regards to dialogue control (also known as dialogue management) most toolkits model a dialogue interaction as a finite-state network [2, 3, 7]. In such a system, the users must follow the system's initiative, and speak in a way the system expects, which is typically very strict in sentence pattern and word order. Unfortunately most real-world users don't know what utterances the system can handle or how they can best cooperate with the system. To give more user freedom, the developer might prepare for all possible dialogue transitions at every dialogue state. However, for those tasks with complex relationships between items or with multiple dialogue topics, there will be too many transitions to take into consideration. In such complicated cases, a customizable, dynamic dialogue control is more desirable.

In this paper, we will present our work on a new toolkit named the d-Ear SDS SDK. It has been specially designed for rapidly developing spoken Chinese dialogue systems. It provides a customizable, rule-based semantic parser for spoken Chinese and a customizable, non-finite-state dialogue manager within a suggested multi-session dialogue system framework. It also includes an integrated development environment to help with domain customization. Using the d-Ear SDS SDK, a trained developer can finish building a flexible information inquiry system with a text interface in only a few days.

The d-Ear SDS SDK is briefly described in the next section. Then we introduce the customization steps and give more details. At last there is a summary of several dialogue systems we have developed with the d-Ear SDS SDK.

2. The d-Ear SDS SDK

The d-Ear SDS SDK was developed by Beijing d-Ear Technologies Co., Ltd. to enable rapid development of spoken Chinese dialogue systems. The first version was released in February 2003, with basic code and tool support. Since the release of the second version, the toolkit has also included an integrated development environment called the d-Ear SDS Studio, which packages several useful tools together in an easy-to-use graphical interface. The latest version was released in June 2004 with most tools strengthened.

The toolkit's suggested dialogue system framework supports multiple concurrent dialogues, with a semantic parser for spoken Chinese and a dialogue manager. Using the d-Ear SDS Studio, various help documents, and an example project, it is relatively easy for a developer to customize a

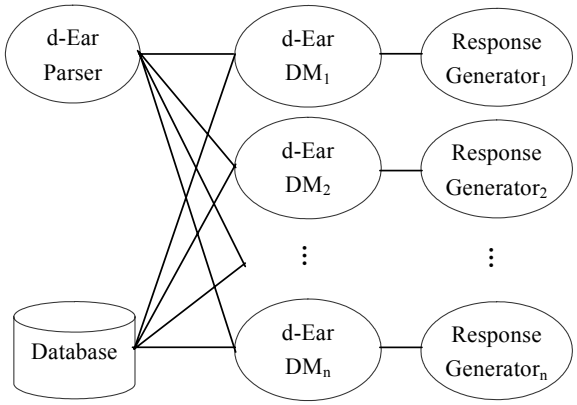

Figure 1: The dialogue system framework

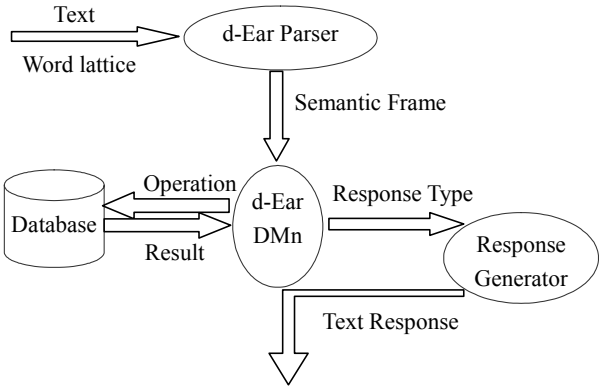

Figure 2: Data flow in a dialogue turn

spoken Chinese dialogue system based on the dialogue system framework.

2.1. The dialogue system framework

The dialogue system framework is depicted in Figure 1. It suggests that during runtime there is a single semantic parser (the d-Ear Parser) with multiple dialogue managers (the d-Ear DM) - one manager instance for each dialogue interaction. The application database (or more broadly, any structured knowledgebase) can be shared among different dialogues. Response generation is currently associated with an individual dialogue manager, while future versions of the SDK will allow for a separate, sharable response generator.

The d-Ear Parser and the d-Ear DM can be customized using configuration files, while more complex domain-dependant operations can be realized by coding using the provided library and program files. Interfaces for the database and response generation are also carefully predefined in the framework, so that the developer is only required to work on the domain-specific (or project-specific) aspects.

The input to the framework may be either pure text or a word lattice (i.e. speech recognition results). The output is a text response, which can then be converted to speech by a text-to-speech engine if desired. Thus the suggested usage is text in, text out or speech in, text out.

The data flow for a dialogue turn is shown in Figure 2. The framework has predefined the structural conventions of the interfaces between modules, while still allowing the developer some freedom as to the format of individual information items. For example, the database operation conditions are constructed from information contained in the semantic frames used during the course of the dialogue. The response type for response generation is determined from the information given by the semantic frames and the database operation results. More details about the interface conventions will be described in the next section.

The dialogue system framework is suggested rather than rigid. For example, the developer could replace the d-Ear Parser with a different semantic parser, as long as the parser's output was in the format needed by the d-Ear DM. Similarly, each d-Ear DM could have its own Database, rather than sharing a single one.

2.2. The d-Ear Parser

The d-Ear Parser employs an extended context free grammar based on semantic classes proposed by Yan [8]. The grammar terminals are keyword classes grouped according to semantics. There are five types of grammar rules: up-tying (strict), up-messing (unordered), by-passing (jumping), long-spanning, and over-crossing. The rules can be grouped in ten priorities, and each priority can be designated lexical or non-lexical.

The extended grammar deals with spoken Chinese phenomena, especially flexible wording structure, hesitation, repeats, and correction. In practice the grammar rules are often over-generated. Thus the developer doesn't need to think over all possible sentences, but only define rules for a few paraphrases. This helps developers inexperienced in grammar definition to save much time during customization.

The d-Ear Parser applies several optimized strategies to achieve quick and robust performance in Chinese word segmentation, and syntactic parsing to conquer the over-generation problem.

2.3. The d-Ear DM

Domain knowledge and task schema representation is the most important aspect of a dialogue system, because it determines the system's interaction capability and the overall flow of the dialogue. The d-Ear DM implements a hybrid dialogue management method based on the Topic Forest structure proposed by Wu [9, 10].

The Topic Forest representation is powerful enough to differentiate items with different importance to the topic and to share information between topics. Because the representation of domain notions is combined together with the representation of domain schema, and the Topic Forest is directly used to represent the dialogue knowledge and dialogue history, it is easy for the dialogue manager to decide how to control the dialogue according to the dialogue knowledge and the current turn information.

Based on the Topic Forest, the dialogue management method adopts algorithms for management of interaction knowledge, management of dialogue history, and response reasoning with a dynamic and mixed-initiative dialogue control. The developer is free to focus on the representation of the real task, leaving the d-Ear DM to automatically control other aspects.

2.4. The d-Ear SDS Studio

The d-Ear SDS Studio is an integrated development environment useful to build and manage a *d-Ear SDS project*, which includes (1) all necessary configuration files, (2) program files, and (3) other resources needed for a dialogue system.

The d-Ear SDS Studio integrates several tools for reducing the amount of customization work, such as a keyword tool, a Topic Forest tool, a "semantic wizard" tool, and so on. There are also diagnostic tools such as a grammar assistant, a sentence parser, a sentence generator, and a system log viewer.

3. Developing New Dialogue Systems

There are typically 7 steps to develop a new dialogue system with the d-Ear SDS SDK.

1. Analyze the domain task and the domain database to find the semantic information items to deal with.
2. Represent the domain knowledge and task schema in a topic forest.
3. Customize the semantic parser.
4. Customize the task-specific turn information and operations.
5. Implement the database operations.
6. Customize the response generator.
7. Choose necessary modules to assemble the system, optionally with a speech recognition module and text-to-speech module for a speech interface.

However, an expert developer doesn't need to strictly follow these steps.

3.1. Parser customization

Customizing the semantic parser can be broken down into three areas of definition: keyword classes, grammar rules, and semantic translation.

Keyword classes are defined in a configuration file. Because of the abundance of homophones and homographs in Chinese, the developer is required to provide both the Chinese characters and Chinese pinyin for all keywords. The keyword's text is used for pure text input and it's pinyin for word lattice input. A keyword tool integrated into the d-Ear SDS Studio can automatically assign pinyin to keywords and give warnings for keywords with identical pinyin.

Grammar rules are also defined in a configuration file. Since defining a grammar is not very easy for beginners -- because of a grammar's complexity -- the d-Ear SDS Studio provides some predefined sets of commonly used keyword classes and rules. In addition, a grammar assistant will semantically check the rules and give errors for undefined terminals or non-terminals, or for ill-formatted rules, and give warnings for unused terminals, unused non-terminals, multiple initial symbols, and rules that won't be applied because of inadequate order of priorities. The Studio's sentence parser is useful for tracing the syntactic parsing results, and the sentence generator for showing the coverage of the grammar.

To customize the semantic translation for different dialogue tasks, the d-Ear Parser associates each grammar rule with a semantic translation function. A "semantic wizard" tool is provided to automatically generate the associations, so that the parser can recursively calls the correct translation function according to the tree-like parsing result. The auto-generated code for translation includes all necessary parameters and sub-constituent calls. Thus, the developer only needs to design the semantic translation for each rule and terminals. The keyword configuration file includes the information how each class of keywords should be translated, which may be (1) an empty representation, (2) the same as keyword text, or (3) in some customized format. In the last case, the developer can either configure a keyword translation file or provide a translation function.

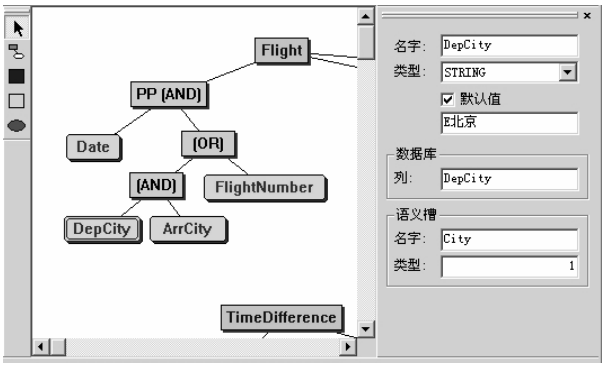

Figure 3: The tool for building the Topic Forest

The semantic translation results are represented in a conventional semantic frame, which defines a representation method for common meta-dialogue acts, the sentence topic (if it can be decided), and information items. There is also an interface for the developer to define new meta-dialogue acts. The format of information content is not predefined so as to provide the developer with increased flexibility.

3.2. Dialogue management customization

The d-Ear SDS Studio includes a tool for building Topic Forests (Figure 3). All the topics and semantic items in a dialogue task are represented, along with their relationships and the corresponding elements in the database.

Once given the domain knowledge and task schema represented in a topic forest, the d-Ear DM is capable of managing dialogue history and interaction knowledge, and reasoning for response focuses automatically. The d-Ear DM can intelligently decide the current turn topic and update the dialogue information according to the context.

Although the d-Ear DM maintains much turn information during the dialogue, the developer can also decide to define some task-specific turn information through the reserved interface. Other task-specific information, such as the default topic and default information items can also be customized.

3.3. Database operation and response generation

An operation interface is abstractly defined between the d-Ear DM and the domain database. It is the developer who decides the actual database management software and implements the real operations.

Dialogue responses have been classified into several types according to the dialogue status or response acts as shown in Table 1. The detail types are for common use and the developer can define new detail types and assemble task-specific responses for each type. After generating a response for the current turn, the developer can also set turn expectations for the next turn.

Table 1: Response types

Type	Sub-type	Detail type
System	Parsing error	Error type
	Convention	Greeting, farewell, thanks
	Meta-dialogue	Repeat, turn scratch, dialogue reset
	Database error	Lost connection, parameter error, operation failed
Topic	User confirmation	Yes, no
	Answer question	User question type
	Give Information	
Item	User confirmation	Yes, no, change
	User Navigation	Navigation type and suggested position
	Raise question	Multiple conditions, multiple results,
	Answer question	User question type
	Give Information	

3.4. Dialogue system assembly

After all the modules have been customized, the developer can integrate them according to the dialogue system framework. According to the task requirements, the developer can dynamically start or end a dialogue transaction by API function calls. If the database operation is relatively slow, e.g. using a remote database, the API function which returns the response to a dialogue turn, should be called in a separate thread.

4. Dialogue systems Achievements

We developed a spoken Chinese dialogue system for flight information inquiry over the course of two years. During the development we read many related materials, and found that building such a prototype system is always time-consuming work, and furthermore that existing toolkits were not flexible enough for our needs, especially when confronted with spoken Chinese.

To test the d-Ear SDS SDK, we tried and spent only one week to successfully replant our flight system into the new, toolkit-preferred framework. We also developed several spoken Chinese dialogue systems with a pure text interface (Table 2) using the d-Ear SDS SDK.

- *Soccer Guess* is a small application to compute the stake for a soccer game from mobile short messages sent from users.
- *Green Food* is a system that provides information on fresh and unpolluted foods as certificated by the Chinese government.
- *JP Restaurant* is a mobile short messaging service that provides information on restaurants in Beijing, opened to the public in June 2004.

Besides this work, an undergraduate student without any linguistics experience has set up a spoken Chinese dialogue system for train information inquiry using the d-Ear SDS SDK, accomplished in only two months. This system is now on *http://www.d-Ear.com/Demo/EasyTrain/EasyTrain.aspx* as a demo.

In regards to response correct rates, *Soccer Guess* and *Green Food* are both above 97%. The *JP Restaurant* system has not undergone formal testing, but overall impressions are in line with the results of the previous systems.

Table 2: Dialogue systems achievements

System name	Soccer Guess	Green Food	JP Restaurant
Number of keyword classes	11	42	66
Number of keywords	47	2789	6058
Database soft-ware and location	In memory	Microsoft Access, local	SQL Server, remote
Number of rules	42	228	133
Efficiency* (sentences/sec)	2500	70	N/A
Development time**	3 days	2 weeks	1 month

*The efficiency is tested on a desktop computer (windows 2 000 Professional, single 1.4GHz Pentium-IV CPU, 256MB memory).
**The time for database construction is not calculated in.

5. References

[1] McTear M.F., "Modelling spoken dialogues with state transition diagrams: experiences with the CSLU toolkit," *Proceedings of ICSLP'98*. pp. 1223~1226

[2] Pargellis A., Kuo J., and Lee C.H., "Automatic Dialogue Generator Creates User Defined Applications," *Proceedings of EuroSpeech'99*. v3, pp. 1175~1178.

[3] Nouza T., and Nouza. J., "Graphic Platform for Designing and Developing Practical Voice Interaction Systems," *Proceedings of EuroSpeech2001*. v2, pp. 1287~1290

[4] http://foto.hut.fi/research/TargetJr/manpages/Viewing/IVAdditions/DialogBuilder.html

[5] Wang Y.Y., and Acero A., "Concept Acquisition in Example-based Grammar Authoring," *Proceedings of ICASSP2003*. v1, pp. 284~287

[6] Lee L.S., Ho Y., Chen J.F., and Chen S.C., "Why is the Special Structure of the Language Important for Chinese Spoken Language Processing? - Examples on Spoken Document Retrieval, Segmentation and Summarization," *Proceedings of EuroSpeech2003*. v1, pp. 49~52

[7] Brndsted T., Bai B., Olsen J. "The REWARD Service Creation Environment. An Overview," *Proceedings of ICSLP'98*. pp. 1175~1178

[8] Yan P.J., Zheng F., Sun H. et al., "Spontaneous Speech Parsing in Travel Information Inquiring and Booking Systems," *Journal of Computer Science and Technology, 2002, 17(6): 924~932*

[9] Wu X.J., Zheng F., and Xu M.X., "Topic Forest: A Plan-based Dialog Management Structure," *Proceedings of ICASSP2001*. v1, pp. 617~620

[10] Wu X.J., Zheng F., and Wu W.H., "A Hybrid Dialogue Management Approach for a Flight Spoken Dialogue System," *Proceedings of ICMLC2002*. pp. 824~829