EasyCmd: Navigation by Voice Commands

Yinfei HUANG, Fang ZHENG, Wenhu WU
Center of Speech Technology, State Key Laboratory of Intelligent Technology and Systems
Department of Computer Science & Technology, Tsinghua University, Beijing, 100084
hifi@sp.cs.tsinghua.edu.cn

ABSTRACT

In this paper we present a system named EasyCmd that provides voice navigation on the desktop of Microsoft Windows 9x system. Speech recognition engine for EasyCmd is much similar to that for dictation machine. Statistical Knowledge Based Frame Synchronous Search algorithm (SKBFSS) and Word Search Tree (WST) technologies are applied for acoustic decoding. Recognition Score Gap (RSG) is used for rejection. We also describe the techniques of monitoring the system, collecting vocabulary and simulating system operations, which are essential to enhance the desktop with voice commands.

1. INTRODUCTION

In this paper we present a system named EasyCmd that provides voice navigation on the desktop of Microsoft Window 9x system. EasyCmd can accept user’s voice commands and then invoke corresponding operations so that user can control the operating system (OS) by speech. EasyCmd is developed to provide users a more convenient and friendly environment.

EasyCmd is a vocabulary-independent voice command system. Speech recognition technology is improved from small-vocabulary, isolated word recognition to large-vocabulary, continuous speech recognition in these years. The typical applications for the former and the latter are voice command system and dictation machine respectively. Previous applications on voice command system are mainly vocabulary-dependent, which implies that lots of speech data need to be collected to train word models. However, speech engine for EasyCmd is vocabulary-independent and shares many technologies with Chinese Dictation Machine (CDM). Thus it needs no training data and uses the same models to CDM. In section 2, we shall describe the technologies used in the speech recognition engine in detail.

For the ease of system development, an Application Programming Interface (API) is defined so that the developers are separated from the technical details of speech recognition engine. Similar to Microsoft SAPI and IBM VoiceType Developers Toolkit[1,2], our API is provided to transfer the efforts of developing speech applications to software manufacturers who have actual demands. Many kinds of speech-based applications, such as CDM, desktop voice navigation, Internet voice navigation and spoken dialogue system, can be built on this API. As the first application based on our API, EasyCmd proves that it is easy and effective to build system on API. This will be introduced in section 3.

As a desktop voice navigation system, EasyCmd has some particular problems to deal with. First, it should organize a vocabulary that captures all meaningful commands. Second, it should follow the changes on the desktop and then inform the engine. Third, when the voice commands are recognized, EasyCmd should simulate the operations. In section 4, we shall reveal the working manner of EasyCmd and show the methods to solve the above problems.

In section 5, we give some experiment results to show the performance of the system. In section 6, we shall give a discussion.

2. SPEECH RECOGNITION ENGINE

The speech engine that EasyCmd system relies on is rather the same one for CDM. But the two have different settings. The unique engine reduces developing cost and running cost. Also the various products with shared engine are easy to maintain and upgrade. The process of speech recognition in engine will be described in the following.

Speech segmentation is the first work in the engine. Speech segmentation separates continuous signals into several segments. Segmentation is accomplished using features such as energy, zero crossing rate and pitch. In voice command system, the purpose of segmentation is to indicate the start and the end of user’s speech. In CDM, more sophisticated segmentation is applied. The engine detects not only sentence level pause but also word level pause, which would reduce the speech decoding cost because the decoding may be carried on in each segment instead of the whole speech. Speech segmentation also provides a prediction on the range of syllable numbers in each segment.

Speech signal is digitized at 16k Hz sampling rate and blocked into frames of 32 ms in length spaced every 16 ms. Each frame is processed to extract 16th-order Mel-Frequency Cepstral Coefficients (MFCC) and 16th-order Auto Regression Coefficients (ARC). The acoustic model is described by a left-to-right non-skipping HMM. The base unit of the acoustic model is syllable. The syllable model is described by Mixture Gaussian Density (MGD) of 6 states and 16 mixtures. Totally 418 toneless syllable models are constructed. The model is user independent.

Acoustic decoding applies frame-synchronous algorithm, with Word Search Tree (WST) to reduce the search space [5]. As shown in Figure 1, WST connects syllable models into word model. When decoding at each frame, all the partial paths in the stack are extended to possible successor states. Since the search space grows rapidly with the increase of the time, we set the maximum number of the paths as a threshold to lighten the burden of the search. Compared with linear search, WST can retain more word hypothesis within the same number of the paths. Thus when the number of paths is limited, it can greatly
increase the efficiency of the decoding.

\[ \sum_{i=1}^{n} \frac{R(i-1) - R(i)}{R(0)} = \frac{R(k) - R(0)}{R(0)} \]  

\( R(0) \) is the recognition score of the first candidate. \( R(k) \) is the recognition score of the \( k \)-th candidate. The candidate is regarded as wrong candidate when its RSG is greater than a given threshold. The word is regarded as wrong command when only more than one syllables that construct the word are rejected. Moreover, when the segmentation predicts that only one syllable is detected in speech signal, the syllable recognition is applied. We use RSG rejection to retain the valid candidates. Then the word is achieved by matching one-syllable word in the vocabulary.

3. API INTERFACE

The speech API is designed after seeking some features from IBM and Microsoft’s developing interface [1,2]. Currently only isolated word recognition function is supported. Although many aspects, such as multiple modes of recognition, focus switching between multiple tasks, vocabulary, grammar and speech database management, are included in design document, they are not realized at this time.

The speech API benefits the development greatly. Before API is developed, the common way to construct a system is to copy algorithm source code from other existed systems and draw out the code relevant to call procedure for speech processing from user interfaces of these systems. The work is quite exhausting for one that is not expert at the speech algorithm and procedure. The debugging and modifying efforts are often made for some details of the algorithm or some changes of the settings. However when someone constructs similar systems later, the work will be done again. When speech API is provided, developers need only to know several functions to clearly grasp the whole working procedure. Since the developers are not responding to the details and the settings of the algorithm, they are greatly released. When the algorithm changes or even call procedure changes, the application can remain untouched, which in the past would be a great trouble when the source code is from both old system and new algorithm.

Many details about the engine are transparent to the developers of the actual system. The API interface realizes some call procedure to simplify the developers’ work. While the engine realizes the technologies, the interface organizes the procedures to call the engine so as to provide functions for the application.

The engine and the application are running simultaneously, while the API interface acts as the connector of the two. After initializing and setting the vocabulary, the engine runs in the background, collecting sound signals continuously. When a word is detected, the engine notifies the interface and the interface calls the recognition procedure to get the recognition result. When the interface gets the recognition result, it calls the rejection procedure to restrict the result. So the application does nothing but waits for the message sent by the interface to retrieve the recognized command. That simplifies the design of the application’s flow.

Vocabulary can be changed at any time. The application provides the interface with a list of words. The interface organizes the procedures to call the engine so as to provide functions for the application.

Desktop voice navigation system EasyCmd is an application designed for user to handle Win9x OS more freely. It receives user’s speech input and recognizes the input to get a command. Then the recognized command is translated into messages which invoke corresponding operation on the desktop. EasyCmd enhances the user interface so that user can control the OS by speech besides keyboard and mouse.

4. NAVIGATION SYSTEM

Desktop voice navigation system EasyCmd is an application designed for user to handle Win9x OS more freely. It receives user’s speech input and recognizes the input to get a command. Then the recognized command is translated into messages which invoke corresponding operation on the desktop. EasyCmd enhances the user interface so that user can control the OS by speech besides keyboard and mouse.
The running appearance of EasyCmd looks like a floating bar as shown in Figure 2. The button on the left is the “run/pause” switch. Center place on the bar displays recognized command or other information. When user utters the helper command “列出词表” (list the vocabulary), the dialog expands to show current commands that user can use.

Vocabulary that EasyCmd collects includes menu items, system menu items, buttons, start button, control keys, program links and base commands. The first three groups of commands are gained from the current, or frontmost window. After recursively exploring the sub-windows of the current window, every level of information is collected. Start button is the left button on the toolbar of the desktop. When user utters “开始” (start), the start button is pressed and start menu is brought out. User may continue to utter the direction keys to select items in the menu. Besides direction keys, other useful keys like enter, escape are also included in control keys. Program link supports any application that can be run in the OS. It even supports system applications like “My Computer” and “Window Explorer”. Base commands include helper commands and “中止导航” (pause navigation). Since the engine doesn’t recognize commands when paused, “start navigation” command can’t be supported.

EasyCmd follows the changes of the current window in order to update the vocabulary in time. Hook is a powerful mechanism provided by the OS [4]. Before a message reaches its destination, it will be caught by the hook user installed, no matter which window the destination is. EasyCmd utilizes this mechanism to monitor the window activation message so that whenever a window is activated or created, EasyCmd will be notified to retrieve new vocabulary from this window. This ensures the engine always recognizes commands for the current window.

When speech input is successfully recognized, the gained command needs to be translated into messages that can drive the OS to do the corresponding operation. Menu items are invoked by WM_COMMAND message, while buttons are simulated by mouse click. Other commands can also be simulated in various ways.

5. EXPERIMENT

We make some experiments to see the performance of EasyCmd system. The vocabulary contains 46 words, which is the commands available when the current window is the main window of Notepad. Totally 5 person (all males) utter all words in the vocabulary once in order to see the recognition performance of the engine. The rejection performance is also tested by uttering 17 out-of-vocabulary (OOV) words. 14 of OOV words are gotten from start menu. Other 3 words (cough, en, ah) simulate disturbance that may be introduced. Table 1 and 2 show the experiment result.

Table 1: Performance Test of EasyCmd for word in the vocabulary (46/person)

<table>
<thead>
<tr>
<th>Rejection</th>
<th>Error</th>
<th>Zero Candidate</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>Total</td>
<td>9</td>
<td>9</td>
</tr>
</tbody>
</table>

Table 2: Performance Test of EasyCmd for OOV (17/person)

<table>
<thead>
<tr>
<th>Error</th>
<th>Rejection</th>
<th>Zero Candidate</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>7</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>9</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>9</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>7</td>
<td>1</td>
</tr>
<tr>
<td>Total</td>
<td>38</td>
<td>7</td>
</tr>
</tbody>
</table>

Error count is the number of wrong recognition result. Rejection count is the number of wrong rejection result. Zero candidate count is the number of result of nothing recognized. Main reason of zero candidate result is the mismatch of speech input and syllable models, which leads to search failure. Since the effect of zero candidate result is similar to rejection result when the word in the vocabulary is uttered, we count the two together to calculate rejection error rate.

Table 3: Statistics on performance of EasyCmd

<table>
<thead>
<tr>
<th>Word in the vocabulary (230)</th>
<th>OOV(85)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error Rejection</td>
<td>Error Recognition</td>
</tr>
<tr>
<td>Count</td>
<td>9</td>
</tr>
<tr>
<td>Rate</td>
<td>3.91%</td>
</tr>
</tbody>
</table>

As shown in table 3, many words in the vocabulary are wrongly rejected, which is the cost of applying rejection for decreasing error rate. In voice command system, the risk of error recognition is much greater. It is quite unsatisfying if wrong operation is invoked, while speaking again may be easy to be accepted by user. Many OOV words are not rejected but wrongly recognized. One reason is that the rejection condition is loose. The word is rejected only when two or more syllables in it are rejected, so that some words can’t be rejected in theory. Further improvement of the rejection algorithm may be made according to the feature of the actual application.

6. DISCUSSION

Desktop voice navigation system EasyCmd achieves good effects as a demo to show our technology level. As a sample utilizing the speech API, EasyCmd implies that we are ready to cooperate with software manufacturers to transfer our
technologies to products. The system also shows that isolated word recognition technology is mature. Software manufacturers need not to collect large quantities of voice data to train word models. By applying the speech API, only a little cost needs to be paid to construct a fairly good speaker-independent speech recognition system. The successful construction of EasyCmd also suggests other applications like Internet voice navigation and spoken dialogue system can be built based on the current speech technologies.

There is much space to improve in EasyCmd’s functions. Menu items in Internet Explorer, Word and start menu could not be retrieved now. Hook mechanism does some harm to the engine, which sometimes may cause the response speed not regular. Moreover, custom commands may be provided as a supplement to current command set for user to add their own favorite.

7. References


