Implementation of Speech Engineering in Android Application using Chess Game

Nur Hadisukmana[1], Lupita Citra Astari[2]
[1,2]*Faculty of computing, President University*
Jl. Ki Hajar Dewantara, Cikarang Baru, Bekasi, 17550
[1]anursu2002@yahoo.com

Abstract— **Nowadays, almost all people have smartphone as one of their needs for their social life. Although the main function of smartphone is to communicate easier with others, it doesn't close the possibility for the phone to be 'smart', which is have another functions beside that. From smartphone, human can develop their skills, mostly soft skills anytime and anywhere.**

***Keywords*— speech engineering, android, chess game**

I. Introduction

Chess, is a two player strategic board game played on a chessboard, a checkers game board with 64 squares arranged in an eight-by-eight grid. Chess is played by millions of people worldwide in homes, urban parks, clubs, online, correspondence, and in tournaments.

Since the second half of the 20th century, computers have been programmed to play chess with increasing success, to the point where the strongest home computers play chess at a higher level than the best human players. Since the 1990s, computer analysis has contributed significantly to chess theory, particularly in the endgame.

Voice Recognition is the field of computer science that deals with designing computer systems that can recognize spoken words. Note that voice recognition implies only that the computer can take dictation, not that it understands what is being said. Comprehending human languages falls under a different field of computer science called natural language processing.

The speech recognition has several limitations such as: need more capability to handle infinite vocabulary, and any grammars that contain non-speech loops can cause speech recognizer to hang.

II. Methodology

The methodology used for developing *ChessCo* follows the main phase of the Software Engineering methodology.

Speech recognition systems provide computers with the ability to listen to user speech and determine what is said. Current technology does not yet support unconstrained speech recognition: the ability to listen to any speech in any context and transcribe it accurately. To achieve reasonable recognition accuracy and response time, current speech recognizers constrain what they listen for by using grammars.

The JSpeech Grammar Format (JSGF) defines a platform-independent, vendor-independent way of describing one type of grammar, a rule grammar (also known as a command and control grammar or regular grammar). It uses a textual representation that is readable and editable by both developers and computers, and can be included in source code.

A rule grammar specifies the types of utterances a user might say (a spoken utterance is similar to a written sentence). For example, a simple window control grammar might listen for "open a file", "close the window", and similar commands.

What the user can say depends upon the context: is the user controlling an email application, reading a credit card number, or selecting a font. Applications know the context, so applications are responsible for providing a speech recognizer with appropriate grammars.

Java Speech Grammar Format (JSGF) will handle every word to be include in the application. Words that will be included inside game will be define with their own function. The words will be shown in Table 2.1

TABLE 2.1.
GAME INSTRUCTION

Words		**Function**
English	**Japanese**	
Apple	Aka	Coordinate A
Beta	Biru	Coordinate B
Charlie	Chaba	Coordinate C
Delta	Dango	Coordinate D
Echo	Edo	Coordinate E
Foxtrot	Furi	Coordinate F
Golf	Gifu	Coordinate G
Hotel	Hana	Coordinate H
One	Ichi	Coordinate 1
Two	Ni	Coordinate 2
Three	San	Coordinate 3
Four	Shi / Yon	Coordinate 4
Five	Go	Coordinate 5
Six	Roku	Coordinate 6
Seven	Shichi/ Nana	Coordinate 7
Eight	Hachi	Coordinate 8
Exit	Deru（出る）	Exit Game from Game Screen

III. Results

This application will accept input in a form of speech, provided a temporary and final result of speech-to-text on the screen. After receiving any kind of the input, System will process it and give the output. This application output is also varies and they are in a form of speech, printed out information, movement and toast message. Figure 3.1 will show this application system overview figure of its input and output.

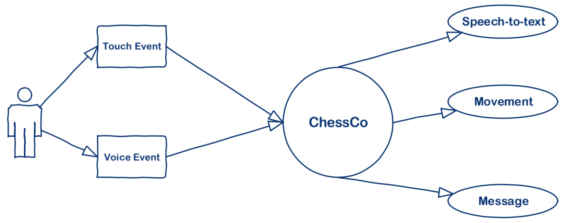

Figure 3.1 Application System Overview

Figure 3.2 will shows the use case diagram of the application. The use case includes "Choose Language", "White Side", "Black Side", Speech Recognizer", "Conversion", "Tokenizer", "Game State", and "Exit Game".

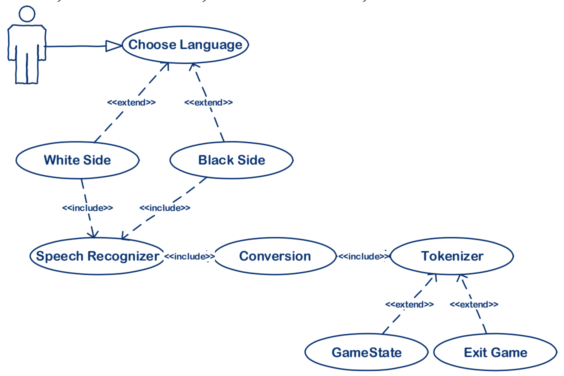

Figure 3.2 Use Case Diagram

If user chooses start at the menu screen, then the choose language screen will appear as shown is Figure 3.3. In this screen, it will provide 2 options of languages, which are English and Japanese. After choosing, Figure 3.4 will shows the game screen. In this screen, speech recognizer will be triggered by recognizer button in the top/bottom of the screen and if valid command is issued then the piece will move.

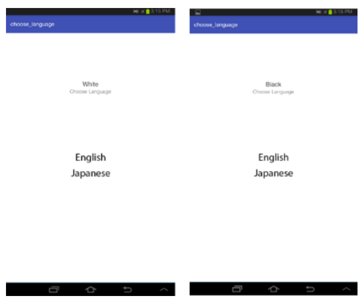

Figure 3.3 Choose Language Screen

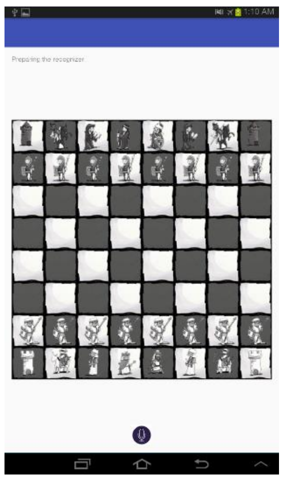

Figure 34 Game Screen

From the result above, factors that can influence probability of success rate in recognizing input are two important factor, which are noise and the surrounding environment. Based on the result, probabilities will be divide into three section based on certain value, which are:

Low probability (0 – 40%)

The low probability usually make character move not defined correctly. Although it is not making the application crash, but the condition of low probability will make the application so hard to play on.

Normal Probability (41% – 80%)

Normal probability sometimes make character move not defined correctly. Although it will not happen very often, some noise came from the surrounding area.

High Probability (81% – 100%)

The high probability is best condition to get the desirable command for the application. This condition is less distraction which make the application will run smoothly.

IV. Conclusions

The application which using user's voice has been successfully made. This application named 'ChessCo' made from 3 elements which by using Java for Android and method provided by PocketSphinx as Speech Recognition Library and Java Speech Grammar Format (JSGF). ChessCo controller is focused on voice recognizing but if the recognizer does not run properly, it can be played as standard way of playing by touching. In order to recognize specific command, it will be described inside JSGF itself and will be checked by activity of the appointed screen. ChessCo can run in any mobile device with Android platform version 2.2 or higher. It provides some points of user interactive such as controlling application using voice. It will give the users new and unique experience. Another advantage of using voice control is it will give simplicity and efficiency to users while they're playing it, make it playable for designated age.

References

[1] Anusuya, M.A. Katti, S.K. International Journal of Computer Science and Information Security: Speech Recognition by Machine: A review. India: 2009.

[2] Hunt, Andrew. JSpeech Grammar Format. 1998

[3] Booch, Grady. Rumbaugh, James. Jacobson, Ivan. *Unified Modeling Language User Guide, The*. Addison Wesley: Massachusetts, 1998.

[4] *International Code of Signals*, United States Edition, 1969 Edition (Revised 2003), Chapter 1, pages 18–19, 148.

[5] Klampfl, Stefan. Jack, Kris. Kern, Roman. A Comparison of Two Unsupervised Table Recognition Methods from Digital Scientific Articles. Austria: 2014.

[6] Picone, Dr. Joseph. Fundamentals of Speech Recognition: A Short Course. Mississippi: 1996.

[7] Rouse, Margaret. Rapid Application Development (RAD) http://searchsoftwarequality.techtarget.com/definition/rapid-application-development