IDIAP COMMUNICATION REPORT

WHO WANTS TO BE A MILLIONAIRE? (II)

Huseyn Gasimov[a] Petr Motlicek Hervé Bourlard

Idiap-Com-02-2013

FEBRUARY 2013

[a] EPFL

Centre du Parc, Rue Marconi 19, P.O. Box 592, CH - 1920 Martigny
T +41 27 721 77 11 F +41 27 721 77 12 info@idiap.ch www.idiap.ch

"Who Wants To Be A Millionaire? (II)"

Quiz Game with Automatic Speech Recognition and Test-To-Speech Synthesis

Optional Project

EPFL/Idiap

version from 13/02/2013

Huseyn Gasimov (EPFL student) (huseyn.gasimov@epfl.ch)
Petr Motlicek (petr.motlicek@idiap.ch)
Hervé Bourlard (bourlard@idiap.ch)

Lausanne, December 2012

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

idiap
RESEARCH INSTITUTE

Contents

Introduction

Game Description

The developed and implemented game exploiting state-of-the-art speech processing technologies is called “Who wants to be a millionaire?”. This game simulates well-known TV-based quiz game spread in many countries. This optional project, partially made at EPFL and Idiap, is a follow-up of the previous work done in the summer semester 2012. In this project, we improve capabilities of the speech control of the game which makes the game more comfortable and fun for players. Further, the game application was extended to support an open-architecture distributed system enabling to be played on different platforms.

The game application has 4 screens: Main, Game, Add a Question, Best Scores.

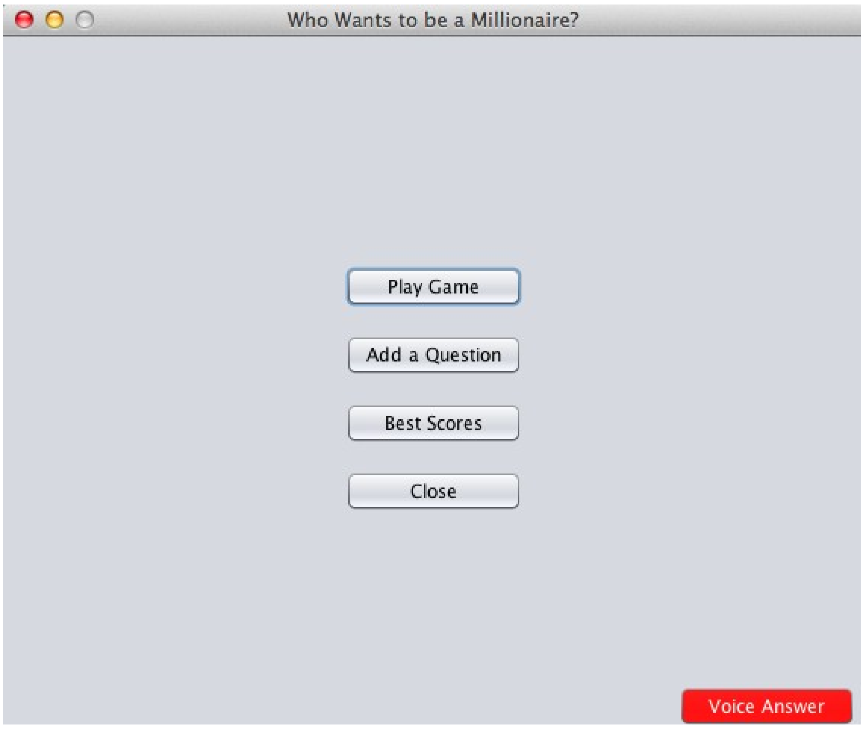


Figure 1: Main Screen.

The main screen is the first screen that is opened when the game is started. From this screen, the other screens can be opened (Figure 1). “Add a Question” screen is intended to give the users an opportunity to add new questions to the questions-archive (database) of the game (Figure 2). The third screen is “Best Scores” which visualizes the best scores achieved by different players (Figure 3). However, this function is not implemented yet in neither Java nor in Android versions of the game. The actual game is played on the “Game” screen (Figure 4). On the left side of the screen, the amount of money won by the player is displayed. In the right top corner, particular helps available during the game are shown to the player.

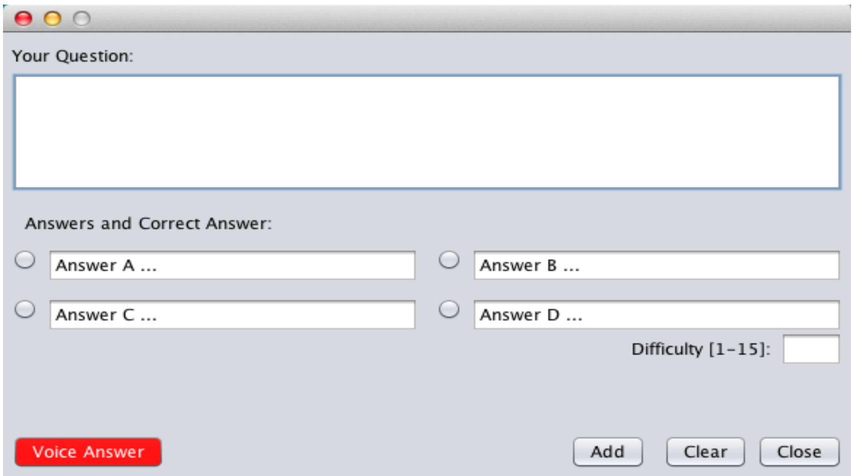


Figure 2: “Add a Question” screen.

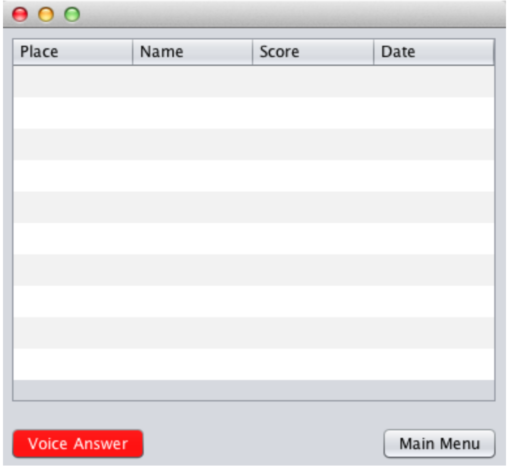


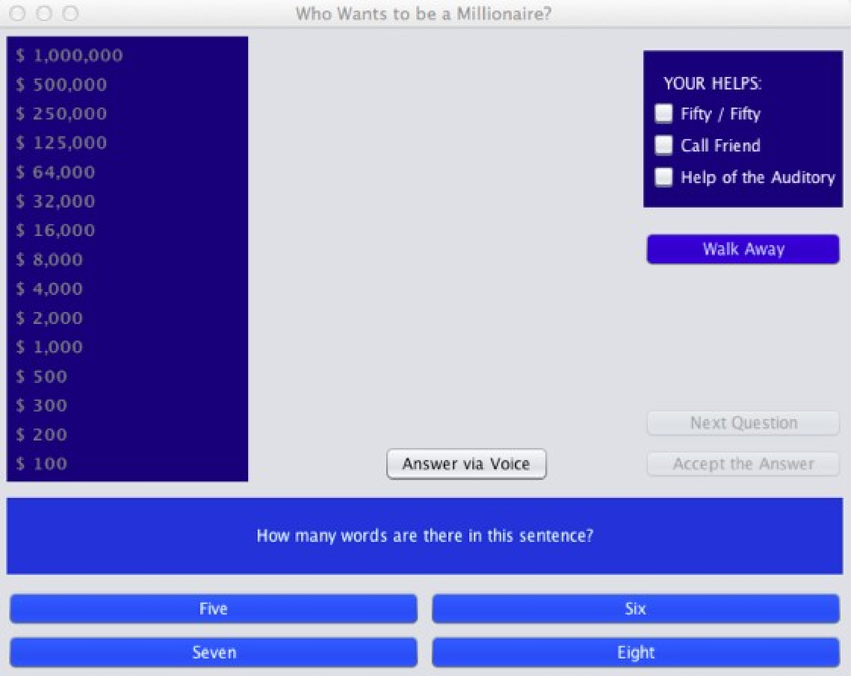
Figure 3: “Best Scores” screen.

Keyword spotting for the game

The developed application allows players to control the game via speech commands (i.e., without traditional devices such as keyboard or mouse). The application automatically recognizes the speech commands. To do so, a dedicated remote Speech Server has been implemented (i.e., communicating with the game in server-client architecture through a communication protocol). More particularly, automatic Key-Word Spotting (KWS) system is developed and integrated instead of full automatic speech recognition tool. KWS receives, as an input, a speech recorded using a built-in microphone (i.e., from remote devices, touchscreens or smart phone) placed often in a noisy environment. Users' speech commands can be answers of the questions selected from the database, or it can be a demand for the available helps or “Walk Away” command. For example, as visualized in Figure 4, users can pronounce one of the following keywords: “Five”, “Six”, “Seven”, “Eight”, “Fifty Fifty”, “Call Friend”, “Help of the Auditory” or “Walk Away”. Since the question and answers can be easily added to the game-database by any user, the KWS needs to be developed as an “open vocabulary” system.

What did we do in the previous semester (summer 2012)?

During last academic semester (summer 2012), we (with help of Aleksey Tryastsyn) designed an architecture of the game and implemented the first version of the game using Swing framework of Java. This version of the game was already capable of accepting speech commands. The speech server was developed as a local application (i.e., running on a device where the game itself resided). The game was also integrated with a Text-To-Speech (TTS) engine (based on Festival TTS **[9]**) which allowed to utter questions to user. We also spent lot of time to import the speech processing tools for KWS (namely HTK and STK applications such as Hcopy, SfeaCat, and SLRatio) from Linux to Mac OS in order to run the game on Mac devices. Besides some small remaining bugs, we were able to run the speech processing tools on Mac devices. For more information, please see our report of the last semester. The report can be found in the public domain of Idiap Research Institute **[10]** or in Google Docs using this link **[1]**.



How many words are there in this sentence?

Picture 4: "Game" screen

New Challenges

During this semester, most of the work was done only by the first author. The main goal of the work in this optional project was to distribute the game application among different platforms (i.e., to enable users utilizing different devices to play the game). To do so, it required to extend the KWS system onto different platforms. Since we were not able to completely import the speech processing tools from Linux to Mac OS, we decided to use Client-Server architecture to implement a cross-platform application. In this hierarchy, KWS was implemented on a dedicated server (operating in Linux), i.e., it performs as a Speech Server communicating with different game clients. After separating the KWS system from the game, few other tasks have remained, such as "voice recording", "sending "wav" file to the speech server via HTTP protocol and receiving a response. These tasks realized over client seemed to be feasible and not too complex to be implemented in different operating systems.

Technologies

Java Swing Framework

Swing is one of the main Java Graphical User Interface (GUI) toolkits and is a part of Java Foundation Classes (JCF) API **[2].** This framework was used to create a cross-platform GUI for the game. Since it is implemented in Java framework, it easily operates over all devices where Java Virtual Machine (JVM) can be installed. Therefore, it enables the game to be played on devices running on Windows, Linux, as well as Mac OS X.

Java Servlet Technology

A Servlet is a Java-based server-side web technology **[3].** Servlet was used to implement Speech Server from the given set of speech tools (HTK **[7]**, STK **[8]**). When a new request arrives at the server, Servlet starts KWS by executing binary files (HCopy, SFeaCat, SLRatio) in the 'ASR' folder and parsing the result from the 'output.mlf' file. Android – MySQL integration (described later) also required Servlet as an intermediate server because direct MySQL connection is not well-supported in Android yet.

Glassfish Application Server

GlassFish is an open-source application server project for the Java EE platform **[4].** It was used as an application container to run the Servlets at the server-side.

Android OS

Android is a Linux-based operating system designed primarily for touchscreen mobile devices such as smartphones and tablet computers **[5].** A separate version of the game was developed for Android OS to demonstrate integration of ASR services with mobile devices.

MySQL Relational Database Management System (RDMS)

MySQL is an open-source RDMS operating as a server and provides multi-user access to a number of databases **[6].** It was exploited to create a database of questions for the game.

HTK and STK Speech tools

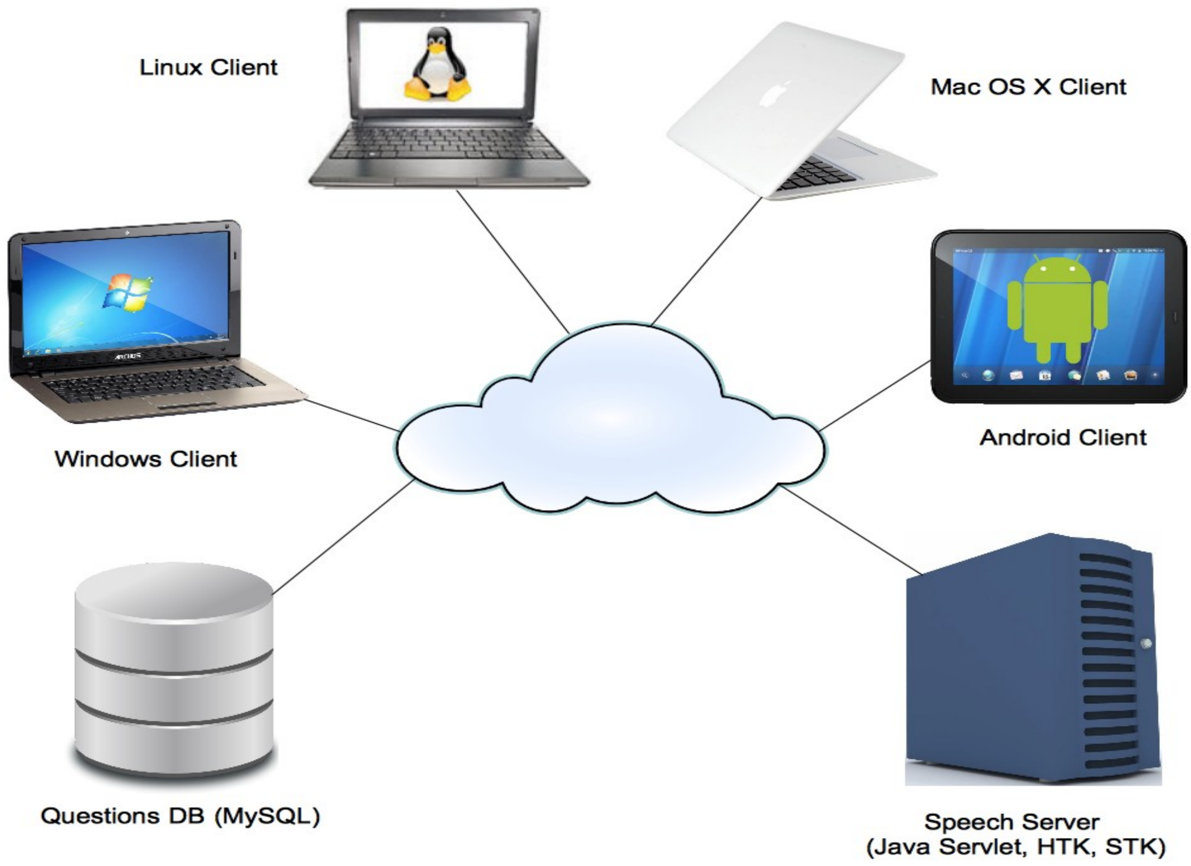
The Hidden Markov Model (HMM) Toolkit (HTK) is a portable toolkit for building and manipulating hidden Markov models.**[7].** STK (HMM Toolkit, STK compatible with HTK) version is a set of tools (for training and ASR using HMMs and Neural Nets (NN)) **[8].** They both were used to perform KWS at the server-side. HCopy tool (from HTK) was used to perform feature extraction, SFeaCat and SLRatio (from STK) were used to perform phone-posteriors estimation (based on a NN trained on large amounts of English meeting data) and decoding correspondingly.

Festival Speech Synthesis System

Festival offers a general framework for building speech synthesis systems as well as includes examples of various modules **[9]**. As a complex module, it offers full text-to-speech through a number APIs: from shell level, through a Scheme command interpreter, as a C++ library, and Java**.** In this project, Festival was used to perform Speech Synthesis (exploited in the Java version of the game). For every question, the application uses Festival to generate the speech content of question and all the answers. Festival was not integrated with the Android version of the game.

Distributed Architecture

The game application is built on a client-server architecture. The client is implemented as a Java Swing or Android application which runs in user's device/machine. The Java Swing version of the game can run in Windows, Linux and Mac OS X. The Questions-database is implemented as a MySQL server which runs in Linux and stores set of questions and their answers (we refer to the previous report of this project for information about the structure and content of the Questions Database **[1])**. The Speech Server is implemented as a web server which was developed using Java Servlet technology. It also operates in Linux (please refer to the Speech Server section of this report for more information).



Picture 5: Client-Server architecture of the game

Speech Server and Keyword Spotting

Investigation and programming in the last semester (summer 2012) revealed that importing the speech tools (HTK, STK) from Linux to another operating systems is not an easy task. Since the tools were written in C/C++, slight change in the C compiler or processor version makes the tools in-compatible over different platforms. Therefore implementing the speech tools together with the game application and to subsequently run the whole – as one application – over different operating systems is not feasible. Therefore, we decided to implement a client-server architecture (where server is operating under Linux and runs speech tools, i.e., the complete KWS and returns the result via HTTP to the requester (client)). In such an architecture, the game application is independent of the speech processing tasks which was the only obstacle for cross-platform implementation.

Having the server-client architecture in mind, we started to develop a Speech Server. We decided to implement it as a Web Server which accepts KWS requests and responds to them via HTTP protocol because HTTP is well-supported by all well-known operating systems. We used Java Servlet technology for this purpose. It works as follows:

1. HTTP multi-part request arrives to the server in the following format:
   - wav (recorded) file to be recognized
   - KeyWord_1 to be recognized
   - KeyWord _2 to be recognized
   - ...
   - KeyWord_n to be recognized
2. The server saves the wav file into the ASR folder
3. The server creates a new dictionary file in the following format using the key-words (that are obtained from the HTTP request) and a common English (CMU version exploited in AMI/DA projects) lexicon which contains pronunciation of English words:
   KeyWord_1 [pronunciation of KeyWord_1]
   ...
   KeyWord_n [pronunciation of KeyWord_n]
4. A new network file (i.e., of the 3-state mono-phones to build a key-word to be detected) is created using the lexicon to be exploited in the decoding. This network contains only the words appearing in new lexicon.
5. Speech features are extracted from the wav file. The input wav file is processed by the HCopy tool and the corresponding feature vectors are retrieved. The input wav file must fulfill certain conditions: recorded speech is sampled with 16 kHz and encoded as 16 bit PCM.
6. Phone posteriors are estimated using a Multilayer Perceptron (MLP). SFeaCat tool from the STK toolkit is used for this purpose to forward-pass input speech features and to obtain vectors of phone-posteriors. In our case, the MLP is trained to estimate the posterior probabilities of 45 English phonemes. The training of the MLP was performed on 150 hours of AMI and ICSI data. After the training, parameters of the MLP are stored in a file and loaded before each KWS detection cycle.
7. Viterbi decoding is done using the Key-word network file created in the step 4. Achieved results are stored to an output file “output.mlf”. SLRatio from the STK toolkit is used for this purpose to perform log-likelihood ratio between a key-word and a garbage model.
8. The output file is parsed and the word with maximum log-likelihood value is chosen. If this maximum value is higher than a threshold (value of negative log-likelihood equal to -30 was used as a threshold for demonstration purposes), the corresponding key-word is sent back to the client as an HTTP response. More precisely, a number assigned to this word,

based on a lexicon, is sent back instead of the word itself. For example, in order to send back a Keyord_2 as a response, '2' is sent back via HTTP to indicate that the second key word was most probably uttered in the retrieved wav file. '-1' is sent back if there are no key-words with log-likelihood higher than the threshold.

More detailed information about KWS system can be found in the previous report **[1]**.

Cross-platform game for Windows, Linux , Mac and Android

Java version of the game

We decided to develop the first version of the game in Java because of the cross-platform compatibility of the Java engine. Current version of the game runs in 3 most popular operating systems: Windows, Linux, Mac OS X. The game application consists of the following parts:

1. *GUI* which was written using Java Swing framework. For more information, see the "Game Description" part of this report.
2. *Voice Recorder for KWS:* It uses standard audio recording functions of Java, see the "Capture.java in" the 'helper" package for the source code.
3. *KWS Client for connecting to the Speech Server.* It uses Apache HTTP client to connect to the Speech Server. The latest recorded file is sent to the Speech Server for KWS and the results are passed to the GUI to select the answer generated by TTS module, see the "ASRClient.java" in the 'helper" package for the source code.
4. *Database Client for connecting to the Questions Database:* It uses the standard MySQL client of Java to connect to the Questions database. At the beginning of the game and after each correct answer, the GUI connects with the database to retrieve a new questions from the database, see the "QuestionsDB.java" in the 'helper" package for the source code.

Android version of the game

As a mobile version of the game, we decided to support Android OS. It is similar to the Java version of the game from the functionality point of view, but most of the code was created from scratch to support Android OS. It also consists of 4 major parts:

1. *GUI* which was designed using the standard Android UI designer. It has 4 different screens: "Main", "Game", "Add a Question", and "Best Scores" (see the following Figures 6 and 7).
2. *Voice Recorder KWS:* It uses standard audio recording features of Android OS, see the "ExtAudioRecorder.java" in the "helper" package for the source code.
3. *KWS Client for connecting to the Speech Server.* It also uses Apache HTTP Client to connect to the Speech Server as its Java counterpart. The latest recorded file is sent to the Speech Server for KWS and the results are passed to the GUI for choosing the answer uttered by the user, see the "ASRClient.java" in the "helper" package for the source code.
4. *Database Client for connecting to the Questions Database:* This part is almost identical to the database client for the Java version of the game. It uses the standard MySQL client of Java to connect to the Questions Database. After every correct answer, the application uses the database client to retrieve a new questions from the database and update the GUI, see the "QuestionsDB.java" in the 'helper" package for the source code.

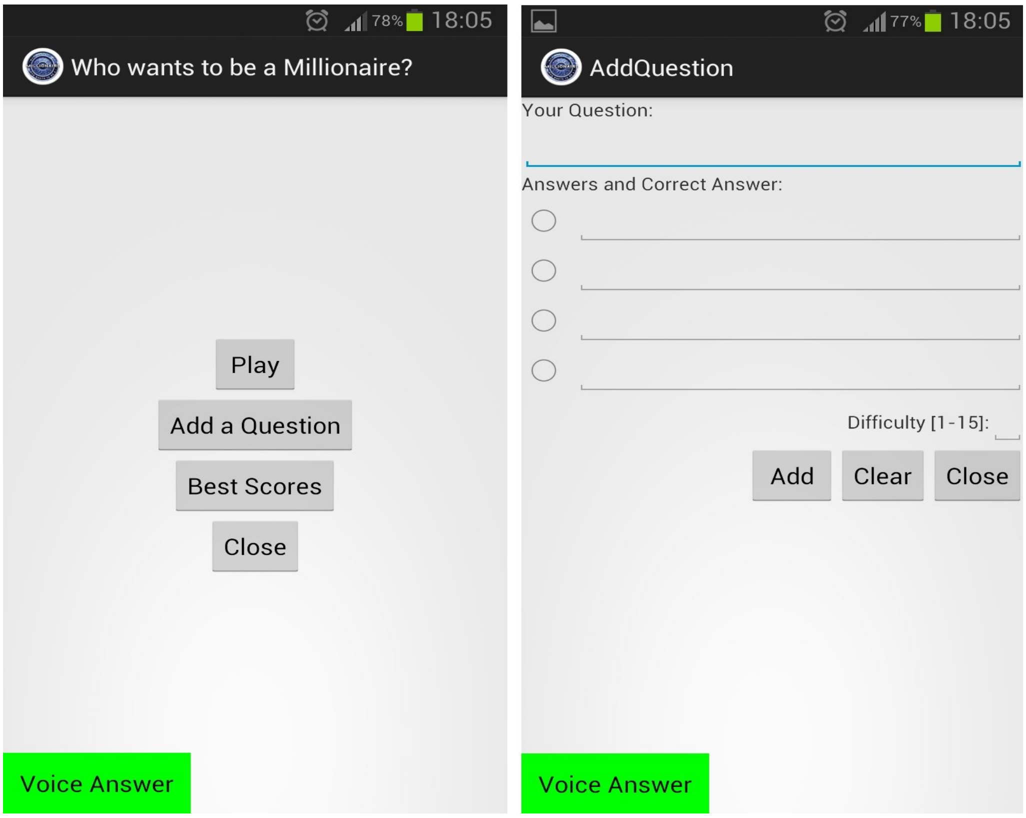
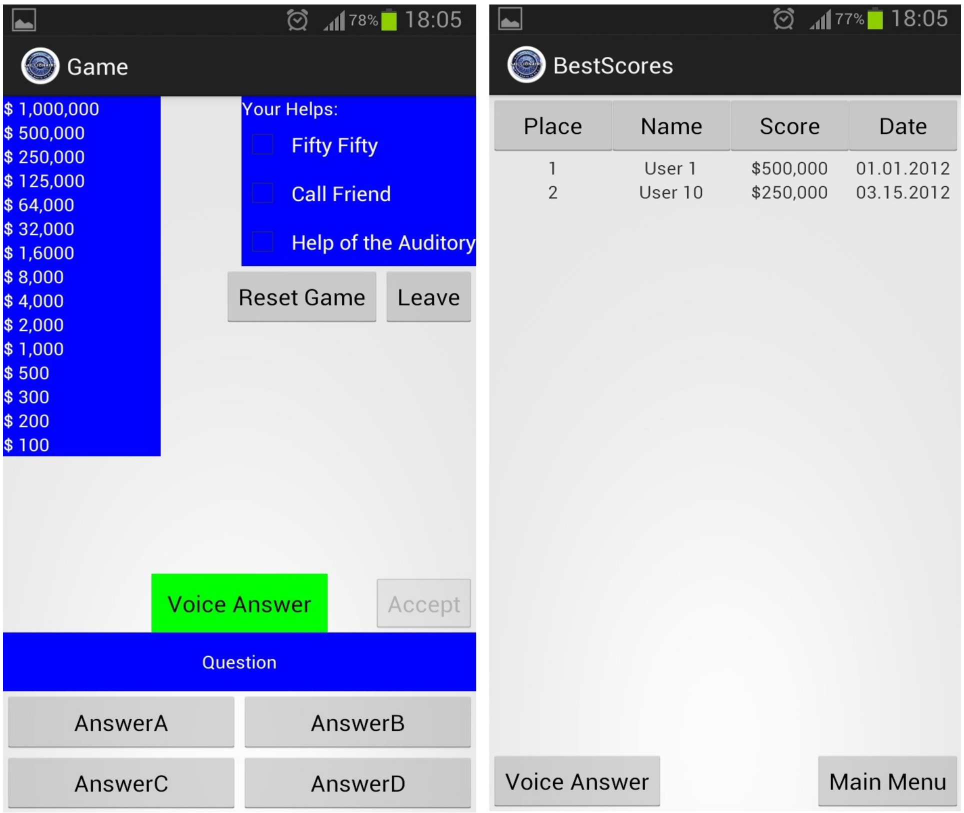


Figure 6: Android version of the game: 'Main' screen (left) and 'Add a Question' scree (right).



Place	Name	Score	Date
1	User 1	$500,000	01.01.2012
2	User 10	$250,000	03.15.2012

Figure 7: Android version of the game: 'Game' screen (left) and 'Best Score' scree (right).

Where we are now and what can be improved?

This project already spanned over two semesters. First, we assembled different tools to perform keyword spotting. Then we developed a Java-based version of the game that was also able to accept speech commands. Second, in this semester, we implemented a Speech Server that helps the game to run KWS independently of the game application. Third, we created Android version of the game, which was mostly built from scratch.

What is therefore available in this stage of the project:

1. Speech Server which can be accessed from anywhere in the network.
2. Cross-platform Java-based game application operating under Windows, Linux and Mac OS X.
3. Android version of the game.

Nevertheless, many other useful things can be added or improved in the game. Some of them are listed below:

1. Concurrency issue can be solved in the Speech Server, because now two requests can not be handled by the Speech Server at the same time.
2. Voice Activity Detection (VAD) can be built into the game that filters out recordings that do not carry speech commands. In this way, we can optimize the usage of the Speech server by accepting only those recordings that potentially carry relevant speech commands (this also means that the game could be played completely without the screen, which is currently nowadays to visualize status of the game).
3. Security modules can be built-in into the game, as well as into the Speech Server. Currently, there is a database for neither remembering users, nor saving sessions. No requests to the Speech Server are authenticated.

References

[1] Report of the last semester, docs.google.com/open?id=0B-4ldv0C5MXvSnNBUkR5TWVQZ2c
[2] Swing Framework, http://en.wikipedia.org/wiki/Swing_(Java)
[3] Java Servlet Technology, http://en.wikipedia.org/wiki/Java_Servlet
[4] Glassfish Application Server, http://en.wikipedia.org/wiki/GlassFish
[5] Android OS, http://en.wikipedia.org/wiki/Android_(operating_system)
[6] MySQL RDMS, http://www.mysql.com/
[7] Hidden Markov Model Toolkit, http://htk.eng.cam.ac.uk/
[8] HMM Toolkit STK, http://speech.fit.vutbr.cz/software/hmm-toolkit-stk
[9] Festival Speech Synthesis System, **http://www.cstr.ed.ac.uk/projects/festival/**
[10] Gasimov H., Triastcyn, A., Motlicek P. and Bourlard H., "Who Wants To Be A Millionaire?", Idiap-Com-03-2012, 2012, http://publications.idiap.ch/index.php/publications/show/2363