A Gesture-Controlled Internet-Based Spoken Conversation Partner On A Mobile Device

Nikos Tsourakis (1), Manny Rayner (1), Matthew Fuchs (2)

(1) University of Geneva, TIM/ISSCO, 40 bvd du Pont-d'Arve
CH-1211 Geneva 4, Switzerland
{Nikolaos.Tsourakis,Emmanuel.Rayner}@unige.ch

(2) Paideia Inc, 2225 E. Bayshore Rd, Suite 200
Palo Alto, CA 94303, California
mattfuchs@paideiacomputing.com

Abstract. We describe a version of the CALL-SLT spoken conversation partner adapted for use on a mobile device. The system allows beginner/intermediate language students to practise interactive spoken language skills, using a version of MIT's "translation game" concept: the machine prompts using an abstract description of what the student is supposed to say, and the student gives a spoken response. We focus on issues concerning the mobile version, which uses a Internet-based client/server configuration in which most processing is performed on the remote server; the client is controlled using accelerometer-based gesture recognition. We present an initial evaluation addressing two central questions: comparison of performance on mobile, Web and desktop versions and accuracy of gesture recognition.

1 Introduction and motivation

CALL-SLT [1] is a generic multilingual Open Source platform based on the "spoken translation game" idea of Wang and Seneff [2]. It is designed for beginner- to intermediate-level language students who wish to improve their spoken fluency in a limited domain. The core idea is to give the student a prompt, formulated in their own (L1) language, indicating what they are supposed to say; the student then speaks in the learning (L2) language, and is scored on the quality of their response. When the student has practised sufficiently on the current prompt, they can ask for the next one. At any time, they can request help; the system responds by giving textual and/or spoken representations of a correct response to the current prompt. Help items are collected from successful logged interactions with the system carried out by previous users.

The system also offers several ways to control both the flow of prompts and the way in which the matching process is performed. For example, prompts are grouped into lessons, each of which will typically be arranged around a theme, and recognition can be adjusted so as to make it more or less forgiving of imperfect pronunciation. The student will sometimes use these features, perhaps

selecting a new lesson or making the recognition more forgiving if they are having difficulties. Most of the time, however, they will be in an interaction loop which only uses a small set of core commands. They get the next prompt, optionally ask for help, start recognition, stop it when they have finished speaking, and see whether the system accepted their spoken response. If it did, they move to the next prompt; otherwise, they try again. It is consequently very important to make the core commands ergonomically efficient.

The original version of CALL-SLT was deployed as a standalone application running on a laptop; since then, we have exploited the system's client/server architecture to develop Internet-based versions in which the server, which performs most of the processing, runs on a remote machine. In this paper, we will concentrate on the case where the client is hosted on a mobile device. This is the most natural way to package the application, since language students often want to exploit "dead time" to fit in fluency practice.

The price to pay is that the system has to work in a more challenging environment; it is particularly important that the user should be able to execute the core set of commands in a comfortable and ergonomically efficient way, something that is not easy to do using conventional modes of interaction. Here, we describe a novel solution to the problem, where concise and intuitively meaningful gestures are used to trigger the commands: for example, the user tilts the tablet right to get the next example, and lifts it to their mouth to speak. The methods used are also appropriate to users who lack fine motor skills or are visually impaired.

The rest of the paper is organised as follows. Section 2 describes the top-level architecture, and Section 3 a series of experiments designed to evaluate performance issues. Section 4 concludes.

2 Architecture

This section presents an overview of the architecture. There are three top-level runtime components: the Dialogue Server, the Speech Router, and the Client. The Dialogue Server performs most of the processing. It receives requests from the Speech Router, and passes back responses. The Speech Router handles the connection to the Internet, and mediates message traffic between the Client and the Dialogue Server. The Client is an application-specific process running on the user's device; we are currently able to run on normal web-browsers, and also under Android. Apart from the design of the CALL-SLT client, the architecture is generic, and has also been used to build other applications. The most visible of these are two speech-enabled Internet games, both of which have been deployed on a large scale. The first, Minion Dominion ([3]; `http://www.miniondominion.com`), was developed as part of the marketing for the Universal Studios movie *Despicable Me*; it allowed two animated characters to be controlled by spoken or typed commands, using a vocabulary of about 400 words. The game was visited by over one million people, and at its peak, shortly after release of the movie in

July 2010, received more than 50K hits a day. A similar game was released by Disney in December 2010.

Although the focus of this paper will be on the Client, we begin by briefly describing the other two components, which run on the server machine. The Dialogue Server is implemented on top of the commercial Nuance Toolkit and the Open Source Regulus Platform [4], encapsulating their runtime functionality in a way that makes them easy to integrate into a distributed application. Speech recognition is performed by the Nuance Toolkit, using grammar-based language models created by the Regulus platform's offline tools. Use of grammar-based models has the advantage of avoiding production of ungrammatical recognition results, which can often be confusing for students, and also allows rapid reconfigurability of the system's coverage. The Regulus Toolkit is responsible for language processing (parsing, semantic analysis, etc), and also for dialogue management.

The Speech Router is designed to connect a client to cloud-based recognition and applications. It is a multi-tiered architecture permitting any number of clients to talk to any number of applications with any number of languages and grammars. Each message arriving from a client identifies the application and session. This allows the first tier to determine which recogniser and application are appropriate. Messages to other tiers then contain the processing chain for the message. The first tier also converts the incoming audio to the WAV format required by Nuance (different clients use different formats, such as mp3 or 3gp). Speech is recorded on a client device, whether a PC or mobile device, and then recognised and processed on the server.

As already noted, the architecture is organized so that most processing happens on the server side. The client, which for CALL-SLT is implemented using a combination of Flash 10 and ActionScript 3.0, is responsible for managing the user interface and state logic, streaming audio to and from the Speech Router, and requesting services from the remote peer. The left side of Figure 1 shows a screenshot of the GUI for the mobile version of the CALLSLT system.

An important point is the way recognition is controlled. Due to the limitations of the target platform (lack of an endpointing mechanism), the user needs to signal both start and end of speech. In the initial versions of the distributed system, this was realised using a push-and-hold solution: the user pushes a button to signal start of speech, and keeps it pressed while speaking. The release of the button signifies end of user speech, terminating the recording and initiating recognition.

2.1 A Gesture-Based Interface

A button-controlled interface works reasonably well if the CALL-SLT client is a process running inside a web-browser on a laptop. The user will typically be sitting with the laptop on a table in front of them, wearing a headset and using the mouse to push buttons. Even here, many users clearly find the push-and-hold mechanism uncomfortable, as is quantitatively demonstrated by the experiments described in Section 3.1.

Fig. 1. Left: CALL-SLT application running on the Samsung Galaxy Tab. The middle pane shows the prompt; the top pane, the recognition result; the bottom pane, text help examples. Button controls are arranged along the bottom. **Right:** equivalent gestures used to control the device.

For the mobile version of the system, a button-controlled interface poses many problems. Few users will have a headset, and the majority will use the tablet's onboard microphone; this involves lifting the tablet to the user's mouth while speaking, and makes a push-and-hold interface extremely inconvenient. Another important point is that there is no tactile feedback from the touch-screen, increasing the user's uncertainty about the interaction status. All of these problems become more acute when one considers that one of the points of deployment on a mobile device is to be able to access the system in outdoor environments, where the screen is less easily visible and the user may be walking or inside a moving vehicle.

For these reasons, we have recently begun investigating the use of an interface which controls the key CALL-SLT functionalities using the intuitive gestures shown on the right side of Figure 1. The current version of the interface supports six gestures. "Get next prompt" and "Return to previous prompt" are signalled by tipping the tablet right and left. "Start recognition" is triggered by moving the tablet so that the microphone is in front of the user's mouth (this involves rotating the device by about 90 degrees, since the Galaxy Tab's microphone is on the upper left side), and "End recognition" is triggered by moving the tablet away from the mouth again. "Help" is requested by moving the device so that the speaker is next to the subject's ear, the natural position for listening to

spoken help in a noisy environment. "Abort" is signalled by shaking the device from side to side. We describe our initial experiments in Section 3.2.

3 Experiments

This section describes two simple experiments we have carried out to evaluate the framework quantitatively. The first (Section 3.1) compares recognition and task performance of the standalone desktop, Web and mobile versions of CALL-SLT. The second (Section 3.2) gives an initial estimate of performance for gesture recognition.

3.1 Comparing performance on different platforms

Our first experiment was intended to compare the performance on the different platforms. We asked eight subjects to interact with the "standalone desktop", "Web" and "mobile" versions of CALL-SLT, using the English-for-French-speakers course. The standalone desktop and Web versions were run on the same high-end laptop, and the mobile version on a Samsung Galaxy Tab, using the onboard microphone. All experiments were carried out in a quiet office environment; all the subjects were good but non-native speakers of English, and either native French speakers or strong second-language speakers. For each of the three platform, they were asked to do 10 examples for each of three different lessons, i.e. 90 examples in total. The order in which the subjects used the different platforms was randomised.

An example consisted of the following individual steps. The system gives the user a prompt in a telegraphic version of the L1 (here, French), e.g. `DEMANDER DE_MANIERE_POLIE BIÈRE`; the user optionally obtains a spoken help example, e.g. "I would like a beer"; the user speaks their response, e.g. "I would like a beer" or "A beer, please"; the system marks the response as "correct" or "incorrect". Responses were typically about 5 to 7 words long for the first two lessons ("asking for things" and "asking for things with questions"), and about 10 to 12 words long for the third one ("using time expressions to make a booking").

All data was logged, and recorded speech files transcribed, after which we computed figures for Word Error Rate (WER) and Task Error Rate (TER); TER was defined as the proportion of examples which were incorrectly rejected by the system. The main results are presented in Figure 2. It is apparent that average WER is better on the desktop than on the Web version (6.2% versus 7.5%), though TER was closer; two of the seven users in fact scored better TER on the Web version. The mobile version was marginally worse than the Web one.

Our impression is that the main factor responsible for these differences is the divergence in user interface functionality already mentioned in Section 2: the desktop version is push-to-talk, while the Web and mobile versions are push-and-hold. It is clear that subjects find push-and-hold less user-friendly; it requires considerably better coordination, and it is easy to release the button too early.

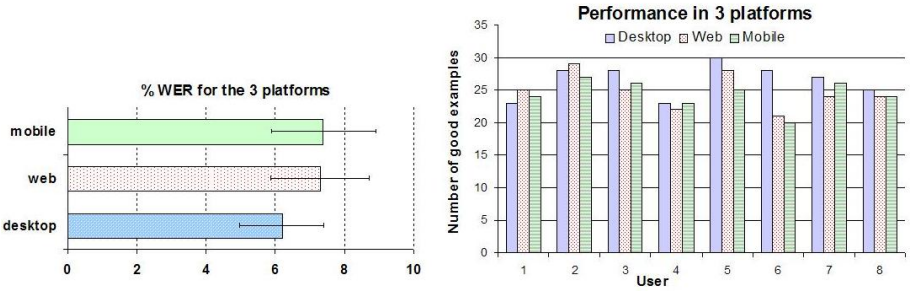

Fig. 2. WER and TER for desktop, Web and mobile versions of English CALL-SLT, for 8 non-native speakers. 95% confidence intervals are calculated using per-utterance bootstrap resampling [5].

3.2 Initial results for gesture recognition

Finally, we present some initial results for gesture recognition. In these preliminary experiments, our ambition has been limited to getting a rough estimate of the separation, in acceleration-space, between the gesture commands shown on the right-hand side of Figure 1. We used the Galaxy Tab's onboard accelerometer, which returns measurements of the G-force experienced by the device along each of the three component axes, and sampled these values every 50 ms for one second while performing examples of the six commands. We collected 20 examples of each command from five subjects. The four right-handed subjects used the device as depicted in the diagram, holding it in their left hand while seated. This configuration is the natural one for a right-handed person; they hold the tablet in their left hand, since they wish to press the buttons with the fingers of their right hand. The single left-handed subject (number 5) held the device in his right hand, and used his left hand to manipulate the controls. We also collected similar data for eight common non-gesture conditions shown in Table 1.

Lying	The device is lying on the table
Sitting, holding	The user is sitting, holding the device in front of him
Standing, holding	The user is standing, holding the device in front of him
Standing, relaxing	The user is standing, holding the device vertically
Running	The user is running
Climbing	The user is climbing a flight of stairs
Descending	The user is descending a flight of stairs
Walking	The user is walking

Table 1. Non-gesture movements used in experiment.

We extracted the mean and Root Mean Square (RMS) values for the X-, Y- and Z-axis components, and used these six values as our features. The plots in Figure 3 show the data-points for the X-Y plane, tagged by gesture, for one of the subjects.

Even with our very basic feature-space, Figure 3 suggests that the gestures should be easy to separate from each other. Experimentation with some standard machine-learning algorithms confirmed this intuitive impression, and also showed that the gestures could be separated reasonably well from the non-gesture conditions. For each subject, we used 75% of the data (both gesture and non-gesture) for training and 25% for testing. We obtained the best figures, shown in Table 2, using `svm-multiclass` [6]; we also used several versions of a simple feed-forward neural net with back-propagation and a version of k-nearest-neighbor, but these gave less convincing results. The lowest average error rate over the five subjects, obtained with a polynomial kernel, was 8.3%. It seems likely that more sophisticated methods (e.g. [7, 8]) could reduce this further.

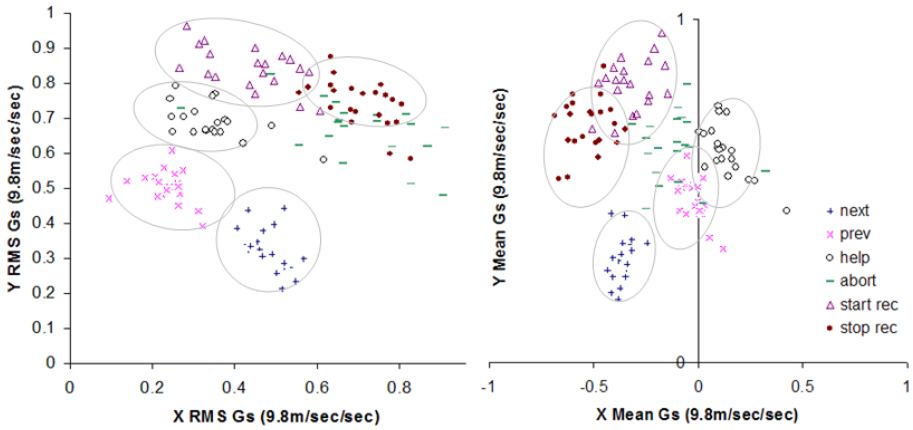

Fig. 3. Separation of gestures in acceleration-space: RMS (left) and mean (right) values of the X and Y components of acceleration for Subject 1.

Kernel	Subject 1	Subject 2	Subject 3	Subject 4	Subject 5	Mean
Radial	18.6	10.0	14.3	18.6	12.9	14.9
Polynomial	8.6	5.7	10.0	10.0	7.1	8.3

Table 2. Classification error (percentage) on gesture recognition using svm-multiclass, for five subjects and two different kernel functions.

4 Summary and further directions

We have described a prototype version of a speech-enabled conversation partner hosted on a mobile tablet computer, and presented an initial evaluation. This pointed us towards a gesture-controlled interface, which is currently in an early stage of implementation.

Although our move in this direction was motivated by feedback from normally-enabled people who have used the application, it becomes apparent that all the arguments apply even more strongly to users who are vision-impaired or lack fine motor control. The coordination required to use the normal button-controlled interface is experienced as challenging by many normally-enabled people, and would be beyond the reach of almost all users who experience problems with sight or fine motor skills. In contrast, we think it likely that the gesture-based interface could be operated by many of these people. If, for example, the device is strapped to the user's hand, it can be operated using only gross motor movements. The fact that gesture identification is trained from the user's own repertoire of movements means that it can potentially be adapted to a wide range of conditions. It would also be straightforward to add a "speech-only-output" mode which could be used even by completely blind people.

The Wii has made everyone aware of the potential of interfaces based on motion sensing; but speech-enabled applications on mobile devices have only become common within the last year or two, and connections between the two technologies have not yet been widely discussed. We are surprised to see what rich synergies are available, and plan to explore them further in the near future.

References

1. Rayner, M., Bouillon, P., Tsourakis, N., Gerlach, J., Georgescul, M., Nakao, Y., Baur, C.: A multilingual CALL game based on speech translation. In: Proceedings of LREC 2010, Valetta, Malta (2010)
2. Wang, C., Seneff, S.: Automatic assessment of student translations for foreign language tutoring. In: Proceedings of NAACL/HLT 2007, Rochester, NY (2007)
3. Chua, C., Rayner, M.: What's the Magic Word? In: Proceedings of the Thirteenth Australasian International Conference on Speech Science and Technology, Melbourne, Australia (2010)
4. Rayner, M., Hockey, B., Bouillon, P.: Putting Linguistics into Speech Recognition: The Regulus Grammar Compiler. CSLI Press, Chicago (2006)
5. Bisani, M., Ney, H.: Bootstrap estimates for confidence intervals in ASR performance evaluation. In: Proc. ICASSP 2004. (2004)
6. Joachims, T.: SVM-Multiclass (2004) `http://svmlight.joachims.org/svm_multiclass.html`.
7. Kela, J., Korpipää, P., Mäntyjärvi, J., Kallio, S., Savino, G., Jozzo, L., Marca, S.: Accelerometer-based gesture control for a design environment. Personal and Ubiquitous Computing **10**(5) (2006)
8. Choe, B., Min, J., Cho, S.: Online gesture recognition for user interface on accelerometer built-in mobile phones. Neural Information Processing. Models and Applications (2010)