Mobile Embodied Conversational Agent for Task Specific Applications

Marcos Santos-Pérez, Eva González-Parada, and José M. Cano-García, Member, IEEE

Abstract — In the recent years major efforts have been made to promote the use of virtual assistants on mobile devices. But in most cases these assistants can only manage a limited and predetermined set of applications. Other limitation that makes difficult the adoption of this technology is the need for a data connection. Thus, this paper introduces the use of an Embodied Conversational Agent (ECA) as an assistant for task-specific applications. The proposed ECA is based on open-source components and runs entirely on the mobile device. The system design was validated by conducting tests to measure the latency and energy consumption.1

Index Terms — Embodied Conversational Agents, Natural Language Interaction, Home Automation, Control Systems.

I. INTRODUCTION

The recent emergence of voice-driven virtual assistants is called to change the way users interact with mobile devices. But in order to achieve a massive adoption of this technology some barriers must be overcome. First, the majority of today’s virtual assistants only can perform certain tasks such as search the web, get directions or schedule reminders and meetings. The tasks addressed by the conversational agents must expand and include many other aspects of the day-by-day life. It can be noted as a second limitation the need of a data connection to run the personal assistants present in the market. This can lead to a negative experience for the user if big latencies occur and can also cause the device to have a much greater energy consumption and then run out of battery.

To address these limitations this paper proposes a mobile Embodied Conversational Agent (ECA) for task specific applications. The ECA possesses all the components needed to maintain a conversation with the user in natural language and is capable of showing some basic emotions. To demonstrate the new range of applications that can be covered by the ECA it was developed an initial prototype for controlling a home automation system. An extra effort was made to get rid of the connectivity need and thus the ECA runs entirely offline on the mobile device. For means of validation some test were conducted. These tests compare the latency and energy consumption of the main elements forming the ECA against other publicly available.

II. RELATED WORK

Most of the conversational agents with presence in the mobile market do not have an animated personification. This lack of embodiment may be one of the multiple reasons why such systems are not yet in widespread use. In contrast, previous studies noted the importance of the ECAs for engaging the user when performing a complex or tedious task. Thus, face to face conversation with an embodied agent has been proven to increase the participation with the system [1]. There have been made previous attempts to describe a complete ECA architecture but they were desktop oriented or network dependent due to the computational requirements that these kind of systems have [2] [3]. Two of the most promising applications of this type of intelligent agents are the following: their use as sociable and emotionally intelligent companions for the users [4] and to control the intelligent devices present in smart environments that have been often addressed separately [5] [6] [7]. This work tries to offer to some extent a common solution to these needs by the use of an emotion-capable ECA.

III. ECA PLATFORM DESIGN

To maximize the future usage and evolution of the platform for other researchers and developers, the ECA software platform was built as a full mobile application that can easily be compiled, packaged and installed on a wide range of devices.

The architecture of the platform shown in Figure 1 follows a modular design so that each component can be modified without affecting others [12].

In order to build a high performance platform for mobile devices all the elements except the conversational engine and the control interface are implemented as shared libraries in C/C++ that only make use of the operating system’s native libraries. But at the same time another objective was not to lose the ability to quickly develop flexible applications with it. Thus, the final application script that defines the ECA behaviour, the conversational engine and the control interface are implemented as Python modules and the rest of the elements reach the same abstraction level as Python façade-interfaces that connect the Python world with the native code via the Java Native Interface.
A. Voice Activity Detector

The Voice Activity Detector’s (VAD) role is to discriminate the user's voice frames from those containing noise. That way, the VAD allows the segmentation of the user's speech into utterances. This module reads the digitized audio samples acquired from a microphone and sends the filtered raw audio to the ASR. The actual implementation of the VAD module is based on the SphinxBase library, which was modified so it could work with the native audio libraries present in the device’s operating system.

B. Automatic Speech Recognition

The Automatic Speech Recognition (ASR) module performs speech to text conversion. It takes as input the utterance with the user's speech that come from the VAD and sends the resultant text to the CE. In the proposed platform, the ASR module is based on the PocketSphinx speech recognition library. Some changes were made to the original code in order to improve the response time on embedded devices by starting the recognition phase once the first speech frame is detected by the VAD [8] and by choosing the most appropriate language model for the topic of the conversation [9].

C. Conversational Engine

The Conversational Engine (CE) extracts the meaning of the utterance, manages the dialog flow and produces the actions appropriate for the target domain. It generates a response based on the input, the current state of the conversation and the dialog history. The CE module is based on PyAIML, an AIML chatbot. AIML (Artificial Intelligence Markup Language) is an extension to XML that provides symbolic reduction, recursion, context-awareness and history management in order to understand the user's utterance and generate an appropriate response. The original code was improved with an optional lemmatizer submodule that reduces both the response time and the memory usage of the CE module when dealing with inflectional languages. It was also added support for an object-oriented database that can decrease the dynamic memory usage when needed at the expense of an increment of the response time [10].

D. Control Interface

The Control Interface translates the commands said by the user to a format that can be understood by the target applications or services running on the same device or accessible remotely. This module is domain-specific and has to be re-implemented or adapted for every new target application.

E. Text-To-Speech

The Text-To-Speech (TTS) subsystem carries out the generation of the synthetic output voice from the text that comes as a response from the CE. For the sake of getting a realistic ECA, it sends to the VHA module a list of the phonemes with their duration so animation and artificial speech match up. The TTS module implementation is based on the Flite TTS library.

F. Virtual Head Animation

The virtual head is the embodiment of the conversational agent and the visual counterpart of the TTS module. This module receives as inputs both the mood information from the CE and the list of the phonemes' durations from the TTS module. By processing the inputs, it generates the visemes (the visual representation of the phonemes) and the facial expression that will be rendered along with the synthetic voice. The purpose of the phonemes' timing list is to modulate the animation speed to achieve perfect lip synchronization. OGRE 3D was used as the rendering engine of the software platform.

IV. Prototype Details

The home automation system was implemented in a room and can control the door lock, the lighting intensity, the blinds position and the air-conditioning system. All the elements under control are connected to form an intelligent wireless network. Every device to be controlled must be connected to a control and communication module like shown in the Figure 2. These control modules are powered by batteries and can last up to ten years due to their ultra-low energy consumption. By being powered by batteries, they also may detect power failures at any node of the network.

The distributed intelligence network is also able to re-configure itself in order to find the best internal topology under faulty conditions. The whole network is connected to an external embedded server. Its purpose is threefold: first, to act as the central node of the wireless network, second, to monitor the electricity consumption of the home automation system, and third, to provide access to different control interfaces.

The home automation system has a manual control interface shown in Figure 3. It needs a tactile display connected directly to the server.

The proposed ECA application plays the role of an alternative control interface that can run on a hand-held device. This way, users can control any device from anywhere in the house with only carrying on a conversation with the virtual assistant.
Likewise the manual control interface, the ECA can control the door lock, the lighting, the blinds and the temperature. It also can give a report with the state of one or several of the intelligent elements that comprises the home automation wireless network.

For testing purposes the ECA application was installed on a mobile device equipped with a 1.2 GHz dual-core CPU and a single-core GPU system-on-chip. Figure 4 shows the ECA running on one of the testing devices.

A. VAD + ASR

The customized PocketSphinx ASR module (CPASR) was tested against the device’s OS built-in ASR system (BIASR). The BIASR can run online and connect to external servers or run offline on the device. Regardless of the online or the offline mode, the users cannot customize the acoustic model or the language model. The CPASR module needs both an acoustic and a language model to run. It was trained a semi-continuous acoustic model with the Voxforge Spanish corpus sampled at 16kHz [8]. For training the trigram language model a few hundreds of user utterances of the home automation control system were gathered and transcripted.

B. CE+CI

This module is domain-specific and there is no public application to compare with but its latency and energy consumption were also measured. The AIML files loaded for the home automation control system comprises of about 250 different entries including recursions.

C. TTS + VHA

The Flite TTS (FLTTS) was compared with the built-in offline TTS system (BITTS) and with an online commercial TTS system (CMTTS). It is important to remark that neither the BITTS nor the CMTTS produce the phoneme timing information needed for lip synchronization and the virtual head got useless when some of them are selected. For this reason, the virtual head animation module was disabled for the tests discussed within this document.

VI. RESULTS

As mentioned in the previous section, both the latency and energy consumption were measured for each component that comprises the ECA. Each test was repeated 10 times to reduce the variability of the results and get an accurate mean value. When an online component that requires a data connection was selected its latency and energy consumption were tested three times depending on the type of data connection selected (WLAN, HSPA or EDGE). That makes 11 different test configurations: 5 for the VAD+ASR components, 1 for the CE+CI and 5 for the TTS+VHA components.
A. VAD + ASR

It can be seen in the Table I that the BIASR system gets the lowest latency value, 190ms, when connected to a WLAN. The next best results are achieved by the offline BIASR and the CPASR module with 278ms and 279ms respectively. The worst latencies are obtained by the BIASR when connected to either an HSPA connection, 337ms, or an EDGE connection, 453ms. These can make oneself to make the wrong decision and substitute the CPASR module with the BIASR system and work only connected to a WLAN. But now if one look at the results of energy consumption in Table II then the right choice can be made and it is to keep the ECA with the CPASR module. It gets an energy consumption of only 320mJ, being the lowest by far. The BIASR obtains more than three times that value, 1120mJ when offline, and gets even bigger figures when connected to WLAN, HSDPA or EDGE networks, 1990mJ, 6570mJ and 6950mJ respectively.

<table>
<thead>
<tr>
<th>Type / Connection</th>
<th>Mean Latency</th>
</tr>
</thead>
<tbody>
<tr>
<td>BIASR / WLAN</td>
<td>190 ms</td>
</tr>
<tr>
<td>BIASR / HSPA</td>
<td>337 ms</td>
</tr>
<tr>
<td>BIASR / EDGE</td>
<td>453 ms</td>
</tr>
<tr>
<td>BIASR / Offline</td>
<td>278 ms</td>
</tr>
<tr>
<td>CPASR / Offline</td>
<td>279 ms</td>
</tr>
</tbody>
</table>

TABLE II

<table>
<thead>
<tr>
<th>Type / Connection</th>
<th>Mean Energy Consumption</th>
</tr>
</thead>
<tbody>
<tr>
<td>BITTS / WLAN</td>
<td>1990 mJ</td>
</tr>
<tr>
<td>BITTS / HSPA</td>
<td>6570 mJ</td>
</tr>
<tr>
<td>BITTS / EDGE</td>
<td>6950 mJ</td>
</tr>
<tr>
<td>CMTTS / Offline</td>
<td>1120 mJ</td>
</tr>
<tr>
<td>FLTTS / Offline</td>
<td>320 mJ</td>
</tr>
</tbody>
</table>

B. CE+CI

The PyAIML interpreter gets a mean latency value of 166ms while only consuming 40mJ on average. These values do not take into account the latency or energy consumed in the communication with the home automation system.

C. TTS + VHA

Table III shows the latencies for the TTS modules. The BITTS system gets an impressive result of only 10ms, being followed by the FLTTS with 166ms. In this case, all the online CMTTS configurations (WLAN, HSPA and EDGE) obtain extremely large delays of more than 1 second that makes this module impossible to use on a real application. Looking at the results of energy consumption of the Table IV, it can be find out that the FLTTS gets again the lowest value, 159mJ. The BITTS consumes more than twice that value, 411mJ and the CMTTS values are affected again by the use of the radio interfaces to transmit the data, 340mJ, 3683mJ and 4663mJ respectively.

<table>
<thead>
<tr>
<th>Type / Connection</th>
<th>Mean Latency</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMTTS / WLAN</td>
<td>1379 ms</td>
</tr>
<tr>
<td>CMTTS / HSPA</td>
<td>1634 ms</td>
</tr>
<tr>
<td>CMTTS / EDGE</td>
<td>2212 ms</td>
</tr>
<tr>
<td>BITTS / Offline</td>
<td>10 ms</td>
</tr>
<tr>
<td>FLTTS / Offline</td>
<td>166 ms</td>
</tr>
</tbody>
</table>

TABLE III

<table>
<thead>
<tr>
<th>Type / Connection</th>
<th>Mean Energy Consumption</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMTTS / WLAN</td>
<td>340 mJ</td>
</tr>
<tr>
<td>CMTTS / HSPA</td>
<td>3683 mJ</td>
</tr>
<tr>
<td>CMTTS / EDGE</td>
<td>4663 mJ</td>
</tr>
<tr>
<td>BITTS / Offline</td>
<td>411 mJ</td>
</tr>
<tr>
<td>FLTTS / Offline</td>
<td>159 mJ</td>
</tr>
</tbody>
</table>

VII. DISCUSSION

Taking into account the results exposed above, it can be concluded that the proposed ECA architecture gets the best tradeoff between latency and energy consumption. This fact responds to multiple factors, being some of the most relevant the implementation of the complex and CPU-intensive tasks in native code and the ability to customize all the components of the ECA platform to the domain-specific application.

A quick estimation of the latency and energy consumption of the entire ECA can also be made by putting all its elements together. For the former, it would be the sum of 279ms + 166ms + 166ms, resulting in a total latency of 611ms. And for the latter, it would be the sum of 320mJ + 40mJ + 159mJ, resulting in a total energy consumption of 519mJ. At this time, it must be noted again that in this result is not addressed the energy consumption of the virtual head because the lip synchronization can only be achieved with the FLTTS module.

VIII. CONCLUSION

This paper proposed a mobile ECA platform for developing task-specific applications on hand-held devices. Thus, the ECA architecture was described and also some brief implementation details for such platform were given. In addition, the whole platform is based on free and open source libraries and a first prototype was built for controlling a home automation system. Then, several tests were conducted to measure the performance of the proposed platform in terms of latency and energy consumption. Finally, the results validated the design and showed that it is ready for wide experimentation with human users.

The future work includes conveying some experiments with real users to measure the usefulness, usability and performance of the platform. Also we plan to extend the functionality of the platform by adding a speaker recognition module and developing new specific applications such as a virtual caregiver for monitoring the health of elderly people.
REFERENCES


IEEE Transactions on Consumer Electronics, Vol. 59, No. 3, August 2013


BIOGRAPHIES

Marcos Santos-Pérez received the MSc degree in Telecommunications Engineering from the University of Vigo in 2008. He was awarded a Ph.D. Fellowship from the Andalusian Regional Government in 2009. His research interests include Embodied Conversational Agents with particular interest on their implementation and performance on embedded devices.

Eva González-Parada received the MSc and PhD degrees in Telecommunications Engineering in 1996 and 2005 from the University of Malaga. She is currently an Associate Professor at the Department of Electronic Technology. Her research interests include embedded systems, wireless sensor networks and ambient intelligence.

José M. Cano-García (M’02) is an Associate Professor of the Electronic Technology Department at the University of Malaga and became a Member of IEEE in 2002. Currently, his main research interests are wireless sensor networks, ambient intelligence, and hardware and software for embedded systems.