CASSANDRA: A multipurpose configurable voice-enabled human-computer-interface

Tiberiu Boros, Stefan Daniel Dumitrescu and Sonia Pipa
Research Center for Artificial Intelligence
Romanian Academy
Bucharest, Romania
tibi@racai.ro, sdumitrescu@racai.ro, sonia@racai.ro

Abstract

Voice enabled human computer interfaces (HCI) that integrate automatic speech recognition, text-to-speech synthesis and natural language understanding have become a commodity, introduced by the immersion of smart phones and other gadgets in our daily lives. Smart assistants are able to respond to simple queries (similar to text-based question-answering systems), perform simple tasks (call a number, reject a call etc.) and help organizing appointments. With this paper we introduce a newly created process automation platform that enables the user to control applications and home appliances and to query the system for information using a natural voice interface. We offer an overview of the technologies that enabled us to construct our system and we present different usage scenarios in home and office environments.

1 Introduction

Major mobile developers are currently including some form of personal assistants in their operating systems, which enable users to control their device using natural voice queries (see Google Now in Android, Siri in Apple's iOS, Microsoft's Cortana, etc.). While these assistants offer powerful integration with the device, they are restricted to that device only - the user (a) is able to control its device, (b) has access to information from simple queries (QA systems included by major competitors in the mobile OS world are able to understand and automatically summarize answers to queries such as: how is the weather today?, who is the president of the United States or who was Michael Jackson) and (c) can organize his/her agenda according to the documents stored in his/her own cloud-hosted storage (Google is able to parse and obtain information from plane tickets and bookings and automatically provides calendar entries as well as general tips such as your plane leaves tomorrow at 11 AM and you should be at the airport before 10 AM due to traffic). We introduce our natural-voice assistive system for process automation that enables control of external devices/gadgets as well as allowing the user to interact with the system in the form of information queries, much like a personal assistant existing now on mobile devices. The user interacts with the system using his/her voice, the system understands and acts accordingly by controlling external devices or by responding to the user with the requested information. All the work presented in this paper was done during the implementation of the Assistive Natural-language, Voice-controlled System for Intelligent Buildings (ANVSIB) national project. We cover aspects related to system architecture, challenges involved by individual tasks, performance figures of the sub-modules and technical decisions related to the development of a working prototype. The tools and technologies are divided in 3 main topics: (a) automatic speech recognition (ASR); (b) text-to-speech synthesis (TTS) and (c) integration with home automation services.

2 System architecture

From a logical point of view, the system is divided in three components (each presented in this paper): ASR, TTS, and integration with external services using natural language understanding (NLU). From an procedural point of view, the system has two distinct entities: one entity acts as an endpoint(client) and is implemented as an android application that is responsible for (a) acquisition of

Proceedings of the EACL 2017 Software Demonstrations, Valencia, Spain, April 3-7 2017, pages 33–36

speech data from the user, processing and recognition (all using external services), as well as the (b) presentation of data to the user. The second entity acts as a server and is responsible for receiving text-input from the endpoint, identifying a scenario from a limited set, extracting parameters, performing the necessary operations and returning information to the endpoint in the form of text, images and sound.

Because we wanted to keep our system as scalable as possible, Automatic Speech Recognition and Text-to-speech synthesis operations are carried out on external servers. The endpoint can be configured to access both Cassandra's own ASR and TTS services (described here) as well as the ASR and TTS servers provided by Google Speech API.

2.1 Cassandra's TTS synthesis system

Text-to-speech synthesis refers to the conversion of any arbitrary (unrestricted) text into audio signal. The unrestricted requirement makes this task difficult and, while state-of-the-art systems produce remarkable results in terms of intelligibility and naturalness, the recipe for producing synthetic voices which are indistinguishable from natural ones has not yet been found. This limitation is caused by the fact that natural language understanding still poses serious challenges for machines and the fact that the surface form of the text does not provide sufficient cues for the prosodic realization of a spontaneous and expressive voice (Taylor, 2009).

TTS synthesis involves two major steps: (a) extraction of features from text and (b) conversion of symbolic representations into actual speech. The text processing step (step a) is usually composed of low-level text-processing tasks such as part-of-speech tagging, lemmatization, chunking, letter-to-sound conversion, syllabification etc. The signal processing task (step b) consists of selecting an optimal set of speech parameters (given the features provided by step a) and generating an acoustic signal that best fits these parameters. Text processing is performed by our in-house developed natural language processing pipeline called Modular Language Processing for Lightweight Applications (MLPLA) (Zafiu et al., 2015). Most of the individual modules have been thoroughly described in our previous work (Boros, 2013; Boros and Dumitrescu, 2015), so we only briefly list them here: the part-of-speech tagger is a neural inspired approach (Boros et al., 2013b) achieving 98.21% accuracy on the "1984" novel by G. Orwell using morphosyntactic descriptors (MSDs) (Erjavec, 2004) in the tag-set; all the lexical processing modules were described in (Boros, 2013)

For syllabification we used the onset-nucleus-coda (ONC) tagging strategy proposed in (Bartlett et al., 2009) and chunking is performed using a POS-based grammar described in (Ion, 2007).

Our TTS implements both unit-selection (Boros et al., 2013a) and statistical parametric speech synthesis. For Cassandra we chose to use parametric synthesis with our implementation of the STRAIGHT filter (Kawahara et al., 1999).

Our TTS system primarily supports English, German, French and Romanian and can be accessed for demonstration purposes from our web portal(link will be provided after evaluation). For other languages Cassandra uses Google TTS, which provides statistical parametric speech synthesis for a large number of languages.

2.2 Cassandra's speech recognition

Cassandra's Automatic Speech Recognition (ASR) service was created by our partners, the Speech and Dialogue Research Laboratory (http://speed.pub.ro) and it is a scalable and extensible on-line Speech-to-Text (S2T) solution. A demo version of the Speech-to-Text (S2T) system resides in the cloud or in SpeeDs IT infrastructure and can be accessed on-line using the Web API or a proprietary protocol. Client applications can be developed using any technology that is able to communicate through TCP-IP sockets with the server application. The server application can communicate with several clients, serving them either simultaneously or sequentially.

The speech-to-text system can be configured to transcribe different types of speech, from a number of domains and languages. It can be configured to instantiate multiple speech recognition engines (S2T Transcribers), each of these engines being responsible for transcribing speech from a specific domain (for example, TV news in Romanian, medical-related speech in Romanian, country names in English, etc.). The speech recognition engines are based on the open-source CMU Sphinx speech recognition toolkit. The ASR systems with small vocabulary and grammar language models were evaluated in depth in (Cucu

et al., 2015), while the ASR system with large vocabulary and statistical language model was evaluated in depth in (Cucu et al., 2014). The first ones have word error rates between 0 and 5%, while the large vocabulary ASR has a word error rate of about 16%.

2.3 Natural language processing

The core of Cassandra is driven by a natural language processing system that is responsible for receiving a text and, after analysis and parameter extraction, acting based on a predefined scenario. A scenario is the equivalent of a frame in frame-based dialogue systems. At any time, the end-user is able to alter Cassandra's configuration by creating scenarios or editing existing ones. A scenario is defined by its name and 3 components:

Component 1 contains a list of example sentences with parameters preceded by a special character '$' which is used to identify them. (e.g. "Set the temperature on $value degrees.")

Component 2 tells the system what to actions to take, depending on the scenario. Actions are described in a JSON with the following structure: (a) gadget - a unique identifier telling the system what module must be used for processing (e.g. a light, an A/C unit, the security system, etc.); (b) gadget_parameters - a free-form JSON structure that tells the system what parameters to pass on to the gadget (e.g. turn the lights on=1 or off=0, set A/C to X degrees, etc.). The values of these parameters can be either constants, predefined system variables or actual values extracted from the text. A gadget can return a text response.

Component 3 is used for feedback. After processing, this JSON is returned to the end-point which initiated the session and it is used to relay information back to the user. This is also a JSON with 2 attributes: (a) friendly_response - a text response that if present will be synthesized and played back to the user; the friendly response can include system variables, parameters extracted from the text or the response returned by the gadget; (b) launch_intent - a structure that informs the end-point that it must launch an external Android Intent with a given set of parameters; Cassandra comes with a predefined set of Intents for image preview, audio playback or video playback.

The methodology for scenario identification and parameter extraction is performed in three sequential steps: (a) the scenario is identified using Long-Short Term Memory neural networks and word embeddings; (b) parameter start/stop markers are then added using a deep neural network classifier trained on a window of 4 words; (c) next, parameter types are added using a window of six words, in which the parameters are ignored. Theoretically, parameter identification (c) and boundary detection (b) could be carried out simultaneously. However, we found that splitting this task in two steps works significantly better, mainly because it mitigates the data sparsity issue. An interesting observation is that using word embeddings in the scenario identification step, allowed the system to identify the query "It's too hot" as an air conditioning activity, though the training data only contained examples such as: "Set AC to $value", "Set the temperature to $value degrees" etc.

Because we wanted to keep the system as language independent as possible, the only external resources required for language adaptation is a word embeddings file extracted using word2vec (Mikolov et al., 2014). The process is simple and given a large enough corpus (e.g. a Wikipedia dump) one can obtain good embeddings by running the word2vec tool on the tokenized text.

3 Standard gadgets and usage scenarios

At submission time we have already implemented 4 demo scenarios:

Scenario 1 is a standard query system, in which the user can ask Cassandra various questions (for example "how is the weather") to which the system will respond using a Knowledge Base (KB). The KB was built using Wikipedia for Romanian and English.

Scenario 2 is a home automation system that allows the user to control home appliances using a natural voice interface. There are several predefined tasks such as multimedia, lighting, climate and security system control. Communication with these devices is performed through KNX (EN 50090, ISO/IEC 14543), which is a purpose-built standardized network communication protocol that is simple and scalable.

Scenario 3 is a set of hard-coded short questions/answers that increase Cassandra's appeal. For example, this Q/A set contains a game called "shrink". It enables the system to perform word associations in a similar manner in which a psychologist would ask a patient to do. This particular game proved to be very appealing to the users in

our test group primarily because it made the system seem more "life-like".

Scenario 4 is a business oriented demo. We integrated Cassandra with a Document Management System (DMS) and created an application that filters documents based on associated meta-data. The interaction is described using a large JSON in which all parameters are optional, missing parameters being ignored by the system. By doing so we are able to respond to queries such as: "Give me all invoices", "Give me all invoices issued to Acme Computers", "Give me all invoices issued to Acme computers that are due in two weeks" etc. We find this scenario particularly interesting from a commercial point-of-view, especially because it enables users access to the DMS without being forced to master any specific search skills. In a similar way, an application could be built to give users access to databases and enable them to construct complex queries and reports with no SQL knowledge whatsoever: "Give me a list of customers that bought tablets over the last 6 months and group them by age".

4 Conclusions and future development

Cassandra is an open-source, freely available personal assistant and, from our knowledge, the only system that is extensible to several languages, not only English, using minimal effort. We intend to further develop this system and extend the basic set of applications to suit most common usage scenarios, as well as to offer more complex NLP-powered business scenarios that integrate with various existing software and hardware implementations. A necessary next step in the near future is to create an open source repository that will enable the creation of a community of developers around it.

Acknowledgements: This work was supported by UEFISCDI, under grant PN-II-PT-PCCA-2013-4-0789, project Assistive Natural-language, Voice-controlled System for Intelligent Buildings (2013-2017).

References

Susan Bartlett, Grzegorz Kondrak, and Colin Cherry. 2009. On the syllabification of phonemes. In *Proceedings of Human Language Technologies: North American Chapter of the Association for Computational Linguistics*, pages 308–316. Association for Computational Linguistics.

Tiberiu Boros and Stefan Daniel Dumitrescu. 2015. Robust deep-learning models for text-to-speech synthesis support on embedded devices. In *Proceedings of the 7th International Conference on Management of computational and collective intElligence in Digital EcoSystems*, pages 98–102. ACM.

Tiberiu Boros, Radu Ion, and tefan Daniel Dumitrescu. 2013a. The racai text-to-speech synthesis system.

Tiberiu Boros, Radu Ion, and Dan Tufis. 2013b. Large tagset labeling using feed forward neural networks. case study on romanian language. In *ACL (1)*, pages 692–700.

Tiberiu Boros. 2013. A unified lexical processing framework based on the margin infused relaxed algorithm. a case study on the romanian language. In *RANLP*, pages 91–97.

Horia Cucu, Andi Buzo, Lucian Petrică, Dragoş Burileanu, and Corneliu Burileanu. 2014. Recent improvements of the speed romanian lvcsr system. In *Communications (COMM), 2014 10th International Conference on*, pages 1–4. IEEE.

Horia Cucu, Andi Buzo, and Corneliu Burileanu. 2015. The speed grammar-based asr system for the romanian language. *ROMANIAN JOURNAL OF INFORMATION SCIENCE AND TECHNOLOGY*, 18(1):33–53.

Tomaz Erjavec. 2004. Multext-east version 3: Multilingual morphosyntactic specifications, lexicons and corpora. In *LREC*.

Radu Ion. 2007. Word sense disambiguation methods applied to english and romanian. *PhD thesis. Romanian Academy, Bucharest*.

Hideki Kawahara, Ikuyo Masuda-Katsuse, and Alain De Cheveigne. 1999. Restructuring speech representations using a pitch-adaptive time–frequency smoothing and an instantaneous-frequency-based f0 extraction: Possible role of a repetitive structure in sounds. *Speech communication*, 27(3):187–207.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2014. word2vec.

Paul Taylor. 2009. *Text-to-speech synthesis*. Cambridge university press.

Adrian Zafiu, Tiberiu Boros, and Stefan Daniel Dumitrescu. 2015. Modular language processing for lightweight applications. In *Proceedings of Language & Technology Conference*.