Designing Language Technology Applications: A Wizard of Oz Driven Prototyping Framework

S. Schlögl
MCI Management Center Innsbruck
Management, Communication & IT
Innsbruck, AUSTRIA
schlogl@mci.edu

P. Milhorat*, G. Chollet*, J. Boudy†
Institut Mines-Télécom
*Télécom ParisTech & †Télécom SudParis
Paris, FRANCE
milhorat@telecom-paristech.fr

Abstract

Wizard of Oz (WOZ) prototyping employs a human wizard to simulate anticipated functions of a future system. In Natural Language Processing this method is usually used to obtain early feedback on dialogue designs, to collect language corpora, or to explore interaction strategies. Yet, existing tools often require complex client-server configurations and setup routines, or suffer from compatibility problems with different platforms. Integrated solutions, which may also be used by designers and researchers without technical background, are missing. In this paper we present a framework for multi-lingual dialog research, which combines speech recognition and synthesis with WOZ. All components are open source and adaptable to different application scenarios.

1 Introduction

In recent years Language Technologies (LT) such as Automatic Speech Recognition (ASR), Machine Translation (MT) and Text-to-Speech Synthesis (TTS) have found their way into an increasing number of products and services. Technological advances in the field have created new possibilities, and ubiquitous access to modern technology (i.e. smartphones, tablet computers, etc.) has inspired novel solutions in multiple application areas. Still, the technology at hand is not perfect and typically substantial engineering effort (gathering of corpora, training, tuning) is needed before prototypes involving such technologies can deliver a user experience robust enough to allow for potential applications to be evaluated with real users. For graphical interfaces, well-known prototyping methods like sketching and wire-framing allow for obtaining early impressions and initial user feedback. These low-fidelity prototyping techniques do not, however, work well with speech and natural language. The Wizard of Oz (WOZ) method can be employed to address this shortcoming. By using a human 'wizard' to mimic the functionality of a system, either completely or in part, WOZ supports the evaluation of potential user experiences and interaction strategies without the need for building a fully functional product first (Gould et al., 1983). It furthermore supports the collection of domain specific language corpora and the easy exploration of varying dialog designs (Wirén et al., 2007). WOZ tools, however, are often application dependent and built for very specific experimental setups. Rarely, are they re-used or adapted to other application scenarios. Also, when used in combination with existing technology components such as ASR or TTS, they usually require complex software installations and server-client configurations. Thus, we see a need for an easy 'out-of-the-box' type solution. A tool that does not require great technical experience and therefore may be used by researchers and designers outside the typical NLP research and development community. This demo is the result of our recent efforts aimed at building such an integrated prototyping tool.

We present a fully installed and configured server image that offers multi-lingual (i.e. English, German, French, Italian) ASR and TTS integrated with a web-based WOZ platform. All components are open-source (i.e. adaptable and extendable) and connected via a messaging server and a number of Java programs. When started the framework requires only one single script to be executed (i.e. there is a separate script for each language so that the components are started using the right parameters) in order to launch a WOZ driven system environment. With such a pre-configured setup we believe that also non-NLP experts are able to successfully conduct extended user studies for language technologies applications.

Proceedings of the Demonstrations at the 14th Conference of the European Chapter of the Association for Computational Linguistics, pages 85–88,
Gothenburg, Sweden, April 26-30 2014.

2 Existing Comparable Tools

Following the literature, existing tools and frameworks that support prototyping of language technology applications can be separated into two categories. The first category consists of so-called Dialogue Management (DM) tools, which focus on the evaluation of Language Technologies (LTs) and whose primary application lies in the areas of NLP and machine learning. Two well-known examples are the CSLU toolkit (Sutton et al., 1998) and the Olympus dialogue framework (Bohus et al., 2007). Others include the Jaspis dialogue management system (Turunen and Hakulinen, 2000) and the EPFL dialogue platform (Cenek et al., 2005). DM tools explore the language-based interaction between a human and a machine and aim at improving this dialogue. They usually provide an application development interface that integrates different LTs such as ASR and TTS, which is then used by an experimenter to specify a pre-defined dialogue flow. Once the dialogue is designed, it can be tested with human participants. The main focus of these tools lies on testing and improving the quality of the employed technology components and their interplay. Unlike DM tools, representatives from the second category, herein after referred to as WOZ tools, tend to rely entirely on human simulation. This makes them more interesting for early feedback, as they better support the aspects of low-fidelity prototyping. While these applications often offer more flexibility, they rarely integrate actual working LTs. Instead, a human mimics the functions of the machine, which allows for a less restrictive dialogue design and facilitates the testing of user experiences that are not yet supported by existing technologies. Most WOZ tools, however, should be categorized as throwaway applications i.e. they are built for one scenario and only rarely re-used in other settings. Two examples that allow for a more generic application are SUEDE (Klemmer et al., 2000) and Richard Breuer's WOZ tool[1].

While both DM and WOZ tools incorporate useful features, neither type provides a full range of support for low-fidelity prototyping of LT applications. DM tools lack the flexibility of exploring aspects that are currently not supported by technology, and pure WOZ applications often depend too much on the actions of the wizard, which can lead to unrealistic human-like behaviour and inconsistencies with its possible bias on evaluation results. A combination of both types of tools can outweigh their deficiencies and furthermore allow for supporting different stages of prototyping. That is, a wizard might complement existing technology on a continuum by first taking on the role of a 'controller' who simulates technology. Then, in a second stage one could act as a 'monitor' who approves technology output, before finally moving on to being a 'supervisor' who only overrides output in cases where it is needed (Dow et al., 2005). However, to allow for such variation an architecture is required that on the one hand supports a flexible use of technology components and on the other hand offers an interface for real-time human intervention.

[1] http://www.softdoc.de/woz/index.html

3 Integrated Prototyping Framework

In order to offer a flexible and easy to use prototyping framework for language technology applications we have integrated a number of existing technology components using an Apache ACTIVEMQ messaging server[2] and several Java programs. Our framework consists of the JULIUS Large Vocabulary Continuous Speech Recognition engine[3], an implementation of the GOOGLE SPEECH API[4], the WEBWOZ Wizard of Oz Prototyping Platform[5] and the MARY Text-to-Speech Synthesis Platform[6]. All components are fully installed and connected running on a VIRTUAL BOX server image[7] (i.e. Ubuntu 12.04 LTS Linux Server). Using this configuration we offer a platform that supports real-time speech recognition as well as speech synthesis in English, French, German and Italian. Natural Language Understanding (NLU), Dialog Management (DM), and Natural Language Generation (NLG) is currently performed by the human 'wizard'. Respective technology components may, however, be integrated in future versions of the framework. The following sections describe the different components in some more detail and elaborate on how they are connected.

[2] http://activemq.apache.org/
[3] http://julius.sourceforge.jp/en_index.php
[4] http://www.google.com/intl/en/chrome/demos/speech.html
[5] https://github.com/stephanschloegl/WebWOZ
[6] http://mary.dfki.de/
[7] https://www.virtualbox.org/

3.1 Automatic Speech Recognition

The JULIUS open-source Large Vocabulary Continuous Speech Recognition engine (LVCSR) uses n-grams and context-dependent Hidden Markov Models (HMM) to transform acoustic input into text output (Lee et al., 2008). Its recognition performance depends on the availability of language dependent resources i.e. acoustic models, language models, and language dictionaries. Our framework includes basic language resources for English, German, Italian and French. As those resources are still very limited we have also integrated online speech recognition for these four languages using the Google Speech API. This allows for conducting experiments with users while at the same time collecting the necessary data for augmenting and filling in JULIUS language resources.

3.2 Text-to-Speech Synthesis

MARY TTS is a state-of-the-art, open source speech synthesis platform supporting a variety of different languages and accents (Schröder and Trouvain, 2003). For the here presented multilingual prototyping framework we have installed synthesized voices for US English (cmu-slt-hsmm), Italian (istc-lucia-hsmm), German (dfki-pavoque-neutral) as well as French (enst-dennys-hsmm). Additional voices can be downloaded and added through the MARY component installer.

3.3 Wizard of Oz

WebWOZ is a web-based prototyping platform for WOZ experiments that allows for a flexible integration of existing LTs (Schlögl et al., 2010). It was implemented using modern web technologies (i.e. Java, HTML, CSS) and therefore runs in any current web browser. It usually uses web services to integrate a set of pre-configured LT components (i.e. ASR, MT, TTS). For the presented prototyping framework, however, we have integrated WebWOZ with our ASR solution (i.e. the combined Google/JULIUS engine) and MARY TTS. Consequently ASR output is displayed in the top area of the wizard interface. A wizard is then able to select an appropriate response from a set of previously defined utterances or use a free-text field to compose a response on the fly. In both cases the utterance is sent to the MARY TTS server and spoken out by the system.

3.4 Messaging Server and Gluing Programs

In order to achieve the above presented integration of ASR, WOZ and TTS we use an Apache ACTIVEMQ messaging server and a number of Java programs. One of these programs takes the output from our ASR component and inserts it into the WebWOZ input stream. In addition it publishes this output to a specific ASR ActiveMQ queue so that other components (e.g. potentially an NLU component) may also be able to process it. Once an ASR result is available within WebWOZ, it is up to the human wizard to respond. WebWOZ was slightly modified so that wizard responses are not only sent to the internal WebWOZ log, but also to a WIZARD ActiveMQ queue. A second Java program then takes the wizard responses from the WIZARD queue and pushes them to a separate MARY queue. While it may seem unnecessary to first take responses from one queue just to publish them to another queue, it allows for the easy integration of additional components. For example, we have also experimented with a distinct NLG component. Putting this component between the WIZARD and the MARY queue we were able to conduct experiments where a wizard instead of sending entire text utterance would rather send text-based semantic frames (i.e. a semantically unified representation of a user's input). Such shows the flexibility of using the described queue architecture. Finally we use a third Java program to take text published to the MARY queue (i.e. either directly coming from the wizard or produced by an NLG component as with one of our experimental settings) and send it to the MARY TTS server. Figure 1 illustrates the different framework components and how they are connected to each other.

4 Demo Setup

The optimal setup for the demo uses two computer stations, one for a wizard and one for a test user. The stations need to be connected via a LAN connection. The test user station runs the prototyping framework, which is a fully installed and configured Virtual Box software image (Note: any computer capable of running Virtual Box can serve as a test user station). The wizard station only requires a modern web browser to interact with the test user station. A big screen size (e.g. 17 inch) for the wizard is recommended as such eases his/her task. Both stations will be provided by the authors.

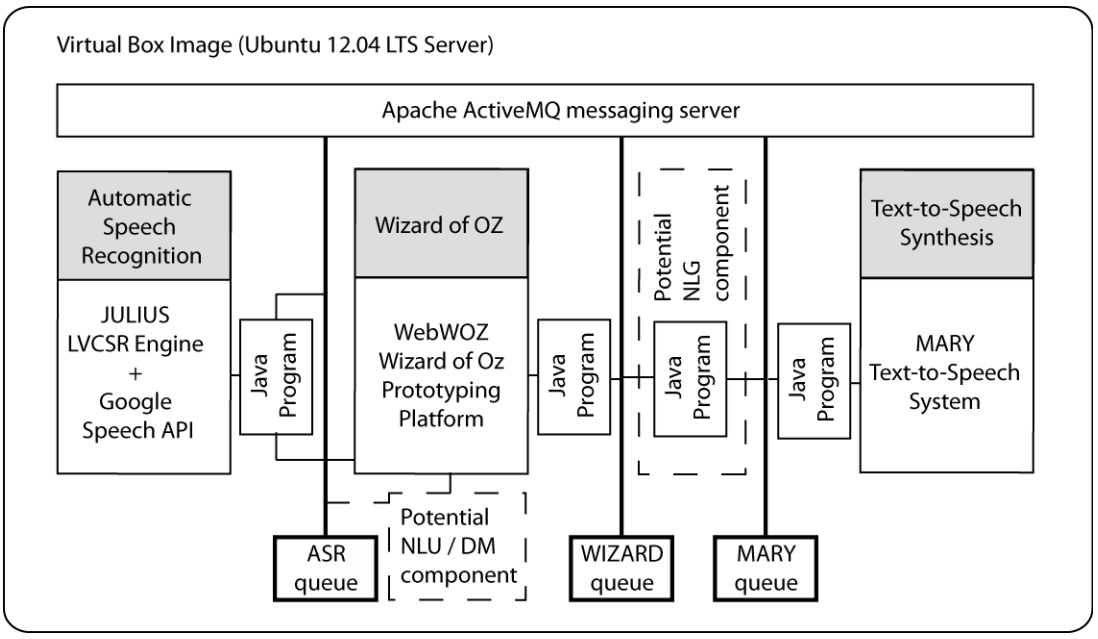

Figure 1: Prototyping Framework Components.

5 Summary and Future Work

This demo presents an integrated prototyping framework for running WOZ driven language technology application scenarios. Gluing together existing tools for ASR, WOZ and TTS we have created an easy to use environment for spoken dialog design and research. Future work will focus on adding additional language technology components (e.g. NLU, DM, NLG) and on improving the currently limited ASR language resources.

Acknowledgments

The presented research is conducted as part of the vAssist project (AAL-2010-3-106), which is partially funded by the European Ambient Assisted Living Joint Programme and the National Funding Agencies from Austria, France and Italy.

References

D. Bohus, A. Raux, T. K. Harris, M. Eskenazi, and A. I. Rudnicky. 2007. Olympus: An open-source framework for conversational spoken language interface research. In *Proc. of NAACL-HLT*, pages 32–39.

P. Cenek, M. Melichar, and M. Rajman. 2005. A framework for rapid multimodal application design. In *Proceedings of TSD*, pages 393–403.

S. Dow, B. Macintyre, J. Lee, C. Oezbek, J. D. Bolter, and M. Gandy. 2005. Wizard of oz support throughout an iterative design process. *IEEE Pervasive Computing*, 4(4):18–26.

J. D. Gould, J. Conti, and T. Hovanyecz. 1983. Composing letters with a simulated listening typewriter. *Communications of the ACM*, 26(4):295–308.

S. R. Klemmer, A. K. Sinha, J. Chen, J. A. Landay, N. Aboobaker, and A. Wang. 2000. SUEDE: A wizard of oz prototyping tool for speech user interfaces. In *Proc. of UIST*, pages 1–10.

C. Lee, S. Jung, and G. G. Lee. 2008. Robust dialog management with n-best hypotheses using dialog examples and agenda. In *Proc. of ACL-HLT*, pages 630–637.

S. Schlögl, G. Doherty, N. Karamanis, and S Luz. 2010. WebWOZ: a wizard of oz prototyping framework. In *Proc. of the ACM EICS Symposium on Engineering Interactive Systems*, pages 109–114.

M. Schröder and J. Trouvain. 2003. The German text-to-speech synthesis system MARY: A tool for research, development and teaching. *International Journal of Speech Technology*.

S. Sutton, R. Cole, J. de Vielliers, J. Schalkwyk, P. Vermeulen, M. Macon, Y. Yan, E. Kaiser, B. Rundle, K. Shobaki, P. Hosom, A. Kain, J. Wouters, D. Massaro, and M. Cohen. 1998. Universal speech tools: The CSLU toolkit.

M. Turunen and J. Hakulinen. 2000. Jaspis- a framework for multilingual adaptive speech applications. In *Proc. of ICSLP*, pages 719–722.

M. Wirén, R. Eklund, F. Engberg, and J. Westermark. 2007. Experiences of an In-Service Wizard-of-Oz Data Collection for the Deployment of a Call-Routing Application. In *Proc. of NAACL-HLT*, pages 56–63.