# WoS – Open Source Wizard of Oz for Speech Systems

**Birgit Brüggemeier**
Fraunhofer IIS, AudioLabs
Erlangen, Germany
birgit.brueggemeier@iis.fraunhofer.de

**Philip Lalone**
Fraunhofer IIS, AME
Erlangen, Germany
philip.lalone@iis.fraunhofer.de

## ABSTRACT
Wizard of Oz (WoZ) systems are widely used in the study of human-machine interactions. In WoZ studies, users are made believe that they are interacting with artificial intelligence (AI), while – in fact – the putative AI is controlled by a human behind the scenes. This setting allows the simulation of AI and the recording of users' reactions to it. These reactions can then be used to train actual AI systems and learn about how people can interact with them. WoZ experiments are common in the field of natural language processing (NLP) and can be used to design and evaluate NLP systems. To our knowledge, currently no open WoZ systems for NLP exist. We designed a WoZ framework for NLP, that we call *Wizard of Oz for Speech Systems*, *WoS* and that is available for open-access on http://github.com/audiolabs/wos [9].

### ACM Classification Keywords
C.3 Special-purpose and application-based systems: Signal processing systems; H.5.2 User Interfaces: Natural Language

### Author Keywords
Wizard of Oz; Speech Systems; WoS; WoZ; Open Source; Natural Language Processing; NLP; Speech Assistant

## INTRODUCTION
Wizard of Oz (WoZ) is a widely-adopted method for studying human-computer interaction (HCI) [3, 15, 13]. In WoZ studies participants are made to believe that they are interacting with an intelligent machine. This machine is however controlled by human associates backstage. These associates are aware of what participants do, participants however are not aware of the associates controlling the machine.

WoZ is a popular method [3, 15, 13] for a number of reasons. One reason is that WoZ can be a solution to a chicken and egg problem of designing user-friendly systems [6]: we require tests to develop user-friendly systems, but for tests we need a running system. With WoZ researchers can simulate an intelligent system and test it with users, without having to build a system from scratch. This enables evidence-based decision making in early stages of system development.



*IUI Workshops'19,* March 20, 2019, Los Angelas, USA, Joint Proceedings of the ACM IUI 2019 Workshops



In addition, WoZ can provide valuable training data for system development. Machine learning methods are widely used to train and develop artificial intelligence (AI) [11]. This AI can then be used in intelligent machines, that interact with humans. The quality of AI depends on the quality of training data, and high-quality data needs to be representative of the use context [11]. Consider a speech assistant, which can interact with users via voice. When training such a speech assistant, one option is to strive for designing the assistant to be as human as possible, so that users can interact with the machine as if they would with fellow human beings [17]. A caveat of this approach however is that users do not interact with speech assistants as they do with people [6, 3, 10]. In fact, researchers find that users are aware that a speech assistant is a machine with limited abilities and adjust their language, for example forming shorter sentences, eliminating articles and simplifying their grammar [10]. If a system was trained on human interactions and then confronted with commands that are common in human-machine interactions, it might perform poorly. In WoZ studies data can be collected in a similar context as the eventual usage context, thus providing more representative data. Instead of simulating a system with WoZ, developers could collect data with existing systems. Interactions between humans and existing machines will be constrained by existing errors of machines [6, 3]. WoZ overcomes this constraint by enabling the simulation of potential systems, before implementing them.

WoZ studies can be differentiated based on simulation type [6, 15, 13], modalities used (Fraser and Gilbert, 1991) and whether natural language is used for HCI [6]. There exist other categories that can be used for differentiating WoZ studies, that we do not name here (e.g. whether the simulated system is embedded in a robot or not, see [8]). Here we focus on WoZ that use natural language for HCI and we call this special kind of Wizard of Oz system *WoS* for *Wizard of Oz for Speech Systems*.

## SYSTEM DESIGN GOALS
We designed WoS as an open software framework for simulating speech systems. WoS is modifiable and usable for researchers with little technical knowledge.

### Modifiability
The first paper on simulating speech systems with WoZ [6] suggested an iterative design of WoZ studies, in which WoZ systems are modified based on findings in user tests. Note, that the authors did not use the term WoS for their Wizard of Oz speech system and thus we follow their nomenclature

when referring to their work. Fraser and Gilbert suggest that in early phases of WoZ studies researchers can learn what their system should or should not say. These findings can then be used to refine the WoZ system. In addition, Fraser and Gilbert note that at each iteration improved hardware or software components can be added [6]. This process makes it possible to incrementally approach the actual speech system [13], as human-controlled software components can be replaced by computer-controlled ones. Thus, the WoZ can incrementally resemble an actual fully-automated speech system. Researchers in HCI agree that such WoZ studies produce results that are practical to implement, as the WoZ resembles the speech system closely [13]. We designed WoS such that it enables incremental addition, removal and modification of system components and implemented it using Python and NodeJS.

### Usability

System development requires evaluation and in some cases, the people who develop the system are not the same people who evaluate it [8]. In addition, evaluators may not have an engineering background, but may be trained psychologists, linguists or other human-centred fields [8]. Thus a WoS system should allow researchers with little technical knowledge to make changes to the system. In our WoS system, small changes can be made without technical knowledge, thus enabling experimenters with non-technical backgrounds to adjust the WoS independently from developers [8]. For example, adjustments to what the system can say, can be made in a text file, which experimenters can select in the Graphical User Interface (GUI) of WoS. In addition, experimenters can directly type responses in the GUI and send these text responses to a module that translates text to synthetic speech. The resulting speech gives the impression of a machine-generated response.

We present an open source WoS system that is modifiable and usable by researchers without technical background. Moreover, we implemented our system on a Raspberry Pi 3B, demonstrating that it can be implemented at a low cost.

## ARCHITECTURE

### Nomenclature

WoS consists of two systems, a *backstage* and a *frontstage* system (see Figure 1). Backstage denotes the system that is running on a computer which is controlled by an experimenter. Frontstage describes the system that is the alleged smart speaker. Frontstage implements audio streaming and text-to-speech and is what test users interact with. Backstage provides a user interface to experimenters, and enables them to monitor audio in near real-time from a frontstage microphone.

### Frontstage

The frontstage consists of a Raspberry Pi 3B running Raspbian Linux. Audio streaming is accomplished using FFmpeg and FFplay. FFmpeg sends an RTP (Real-time Transport Protocol) audio stream over UDP (User Datagram Protocol) from the frontstage to the backstage, where it is then saved, and restreamed to another port on the backstage. FFplay is then used to play the audio. In addition, the frontstage integrates a

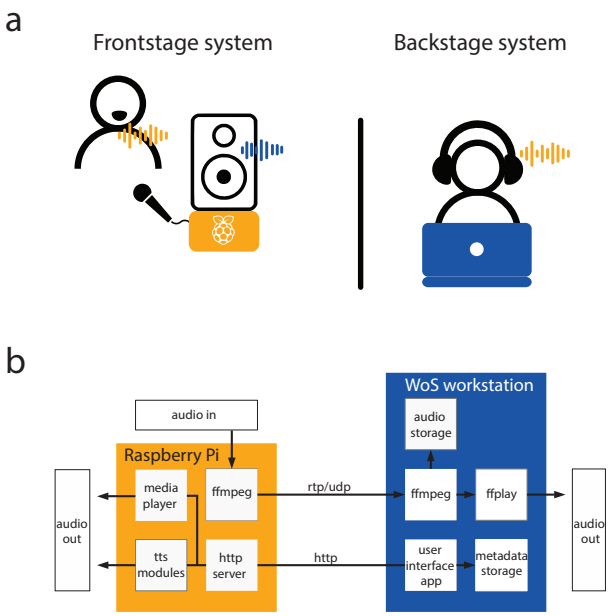


**Figure 1. Illustration of Wizard of Oz for Speech Assistants (WoS). (a) WoS consists of two systems, a frontstage and a backstage system. The frontstage system (left and in orange) is the system users get to interact with, the backstage system (right and in blue) is the system experimenters interacts with. The frontstage system captures users' utterances and sends them to the backstage system for observation (orange waveforms). In addition the frontstage system plays back commands selected by the experimenter (blue waveform). (b) A detailed overview of the modules used in WoS. Arrow heads indicate abstract information flow.**

http server that controls a media player (e.g. for remote song playback) and Text-to-Speech (TTS) modules. TTS converts text that is selected by an experimenter into artificial speech. For an illustration of the WoS system see Figure 1.

### Backstage

The backstage environment can be any laptop or desktop computer. We tested the system on Linux and MacOS but we have not tested Windows. The user interface is implemented using NodeJS and Electron. When started, the backstage software connects to the frontstage and requests the audio stream. The experimenter can then begin to monitor audio signals from the frontstage system. The user interface enables experimenters to enter text which is then spoken by the frontstage system.

The backstage system can be controlled with a simple Graphical User Interface (GUI) (see Figure 2). The GUI shows a drop-down list for language selection for speech generated by WoS. Experimenters can type custom responses in the GUI and send them to the frontstage to be synthesised into speech. In addition, the GUI features a list for selecting prepared system responses, which can reduce the duration between user request and system response, as the experimenter does not have to type prepared responses. Moreover the GUI shows a box with audio files that are saved on the frontstage system. Selecting and playing these audio files can further reduce response time. Responses can thus be prepared both in text and in audio, reducing waiting times for research subjects. What is more, the audio file box can be used to access songs that

are saved on the frontstage system, thus WoS can act as a multimedia player system.

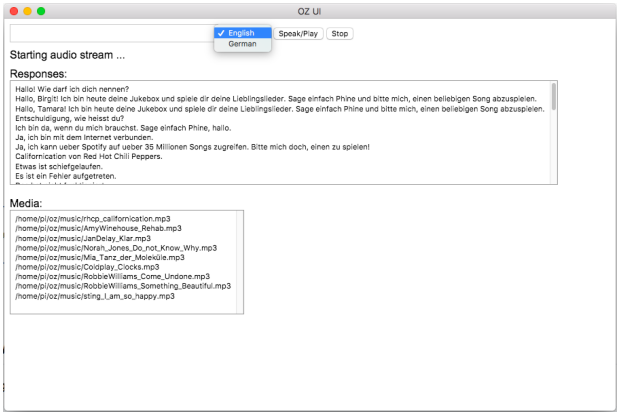


**Figure 2. Screenshot of the Graphical User Interface (GUI) of WoS. The GUI of WoS shows menus for language selection, for selection of predetermined responses, and for playback and stopping of media files.**

### Implementation

We provide an open-source, MIT-like GitHub repository for WoS: `http://github.com/audiolabs/wos` [9]. Our code is openly accessible via this repository. We implemented the system twice: on a home network and at a research institute. Both times, the frontstage system ran on a Raspberry Pi 3 Model B. The backstage system ran on a Thinkpad T61 running Ubuntu 16.04.5 LTS, an iMac, running macOS High Sierra, a MacBook Air, running macOS Sierra, and a MacBook Pro running macOS Yosemite. We tested the set-up with a HDE-X43 microphone connected to a SYBA external USB Stereo Sound Adapter with the Raspberry Pi to relay recorded speech to the backstage system and an Anker Super Sound Core Mini Mobile Bluetooth Speaker, as well as a Acemile Theatre Box speaker, to play back speech generated by the WoS TTS module.

We implemented *Google Translate's text-to-speech* API [5] in WoS, which can be used free of cost. In addition we provide support for the open source TTS solution *festival* [2].

## INTERACTION WITH WOS

We piloted our system with a naive participant (22 years, female, technical background). We wanted to test (1) if our WoS system worked in an experimental setting and (2) if our WoS system could convince people that they were interacting with a machine. Thus we told a cover story, saying that we were evaluating a prototype of a speech assistant system for music control. Regarding (1), we found that our WoS system worked as expected throughout the experiment. After the experiment the participant was debriefed and told that the speech assistant was controlled by a human experimenter. The participant reported that she had believed that she was interacting with a machine and that she was surprised to learn this was not the case, which suggests that (2) was the case.

In order to illustrate application of WoS, we present a sample interaction between a participant and experimenters using WoS. This interaction is structured similarly to a screenplay, showing elements of dialogue and describing actions in square brackets. There are two experimenters, one who introduces the participant to the frontstage set-up (*experimenter F*) and one who listens to the conversation backstage (*experimenter B*). In addition, there are two rooms, a room for the frontstage system (*room F*) and a room for the backstage system (*room B*).

[*Participant* and *experimenter F* enter *room F*.]

*Experimenter F*: [pointing to the speaker] This is the smart speaker that you agreed to test today. You can interact with the speaker using your voice. The speaker can be used to play songs. For example you can ask it: Play 'Californication' by the 'Red Hot Chilli Peppers'.

[*Experimenter B* selects the song backstage and clicks 'Speak/Play' in the GUI of WoS. The song plays in *room F*.]

*Experimenter F*: [to *participant*] I will leave you alone now, so that you can test the speaker. Please ask for at least three songs and come to see me after you are finished.

[*Experimenter F* leaves *room F*.]

*Participant*: [to speaker] Please play 'We will rock you' by the 'Rolling Stones'.

[*Experimenter B* selects the error message 'Sorry I can't help you with that' and clicks 'Speak/Play' in the GUI of WoS. The error message plays in *room F*.]

*Participant*: [reformulating request to speaker] Play the song 'We will rock you' by the band 'Rolling Stones'.

[Interaction continues until participant decides to stop.]

## CONCLUSION

WoS was built as a proof of concept of a simple Wizard of Oz system for simulating speech assistants that can be implemented cost-effectively. We wanted to make WoS usable by experimenters without a technical background. Control of the GUI requires little technical expertise, however starting the GUI requires usage of command line, which arguably may be an obstacle for non-technical experimenters. Moreover, new responses and media files have to be added in a text file and the GUI has to be restarted for them to appear. Thus usability of WoS can be improved by enabling the GUI to start without command line and building a media and response importer within the GUI. In addition, the GUI is plain and could be aesthetically improved.

While non-technical experimenters can use WoS, it is useful for the experimenter to be acquainted with responses typical of speech assistants in order to realistically mimic them. Cathy Pearl gives an overview of design principles of VUIs, that can guide response formulation [12] for experimenters with no prior knowledge of speech assistant systems. Notably, responses can be preformulated in WoS and selected in the GUI, which allows experimenters to premeditate responses before running a study. Selecting responses instead of typing them also reduces response time during experimentation. Nonetheless, it may be useful to warn participants that the VUI may have prolonged response times. This will prepare participants to expect a waiting time, if it should occur.

To improve the experience of experimenters WoS could be complemented by video streaming, so that the experimenter sees the participant. Currently WoS supports only audio streaming. Furthermore, analysis of interactions could be simplified by adding automatic transcription of recorded audio files. This could be achieved by implementing Automatic Speech Recognition (ASR) softwares.

The audio streaming protocol currently used in WoS is RTP over UDP, and a different audio streaming protocol could be used. First we tried the icecast server and we found that there was an audio delay of five to ten seconds, which was unacceptable for our purposes. FFmpeg supports different protocols. We chose FFmpeg with RTP over UDP as low-latency transmission of audio was a key requirement for us and RTP over UDP satisfies that.

Currently we provide support for Google Translate's TTS API [5] and the festival speech synthesis system [2]. In addition, *espeak* [16], *macOS's say* [14], *Google Cloud's TTS* [7][1], *Amazon Polly* [1], and *Deepspeech* [4] could be implemented in WoS. Some of these services enable researchers to select voices, thus enabling evaluation of users' preferences for specific voices, for example male or female voices or specific accents. This is possible in the festival speech synthesis framework, which is implemented in WoS: developers can build new voices and select voices developed by other users of festival [2].

Google Translate's TTS API [5] currently supports 103 languages (checked on November 23rd 2018). Thus WoS, which supports Google Translate's TTS, can be used for evaluation for a wide range of languages.

Reasons for conducting Wizard of Oz experiments include simulating a realistic human-computer interaction. In order to achieve a realistic simulation, participants have to believe that they are interacting with an actual smart speaker. In order to achieve this, some improvements have been planned. Current commercial smart speakers give visual feedback when they detect users addressing them. WoS could be modified to include a speech detection module which activates LEDs to give visual feedback for users of WoS. This could create a more realistic user experience, which could make WoS more convincing for user tests. In addition, experimenters can choose speakers and microphones in such a way as to convince users that the device they see is a smart speaker. For this the physical appearance of the frontstage system can be modified: the frontstage system of WoS runs on a Raspberry Pi and experimenters can choose to hide or display the Raspberry Pi. In addition, the Pi can be connected with any microphones and speakers that are compatible with a USB sound card that runs on a Raspberry Pi 3. Notably, commercial smart speakers conceal speaker and microphone arrays in speaker cases, thus concealing may help the illusion of interacting with a smart machine.

Our intention is to provide WoS as a framework for HCI research on speech systems. Moreover, WoS can be run as is on a Raspberry Pi and a Laptop. We want to make it easier for the community of HCI researchers to use the powerful Wizard of Oz paradigm for studies on voice user interfaces. In addition, we wish for WoS to be adopted and improved by the HCI-community, by adding new languages and features. Therefore, we make our code available under an MIT-like license at `http://github.com/audiolabs/wos` [9].

[1]Google Cloud's TTS is different from Google Translate's TTS. We used Google Translate's TTS via the Python library gTTS.

## REFERENCES


1. Inc. Amazon Web Services. unknown. Amazon Polly – Turn text into lifelike speech using deep learning. (unknown). `https://aws.amazon.com/polly/?nc1=h_ls` [Online; visited on November 24, 2018].
2. Alan W. Black and Paul A. Taylor. 1997. *The Festival Speech Synthesis System: System Documentation* (1.1 ed.). Technical Report HCRC/TR-83. Human Communciation Research Centre, University of Edinburgh, Scotland, UK. Avaliable at http://www.cstr.ed.ac.uk/projects/festival/.
3. Nils Dahlbäck, Arne Jönsson, and Lars Ahrenberg. 1992. Wizard of Oz Studies Why and How. *Intelligent User Interfaces* (1992), 193–200.
4. Kelly Davis. 2016. A TensorFlow implementation of Baidu's DeepSpeech architecture. (2016). `https://github.com/mozilla/DeepSpeech` [Online; posted on 22 Februar 2016].
5. Pierre Nicolas Durette. 2014. Python library and CLI tool to interface with Google Translate's text-to-speech API. (May 2014). `https://github.com/pndurette/gTTS` [Online; posted 15 May 2014].
6. Norman M. Fraser and G. Nigel Gilbert. 1991. Simulating speech systems. *Computer Speech and Language* 5, 1 (1991), 81–99.
7. Google. 2017. Cloud Speech-to-Text. (2017). `https://cloud.google.com/speech-to-text/` [Online; visited on November 24, 2018].
8. Guy Hoffman. 2016. OpenWoZ: A Runtime-ConfigurableWizard-of-Oz Framework for Human-Robot Interaction. *2016 AAAI Spring Symposium Series* (2016), 121–126.
9. Philip Lalone. 2018. An implementation of the Wizard of Oz experiment. (2018). `http://github.com/audiolabs/wos` [Online; posted on 8 August 2018].
10. Page Laubheimer and Raluca Budiu. 2018. Intelligent Assistants: Creepy, Childish, or a Tool? Users' Attitudes Toward Alexa, Google Assistant, and Siri. (August 2018). `https://www.nngroup.com/articles/voice-assistant-attitudes/` [Online; posted 5-August-2018].
11. Ryszard S Michalski, Jaime G Carbonell, and Tom M Mitchell. 2013. *Machine learning: An artificial intelligence approach*. Springer Science & Business Media.
12. Cathy Pearl. 2016. *Designing Voice User Interfaces: Principles of Conversational Experiences* (1st ed.). O'Reilly Media, Inc.

13. Laurel Riek. 2012. Wizard of Oz Studies in HRI: A Systematic Review and New Reporting Guidelines. *Journal of Human-Robot Interaction* 1, 1 (2012), 119–136. http://www.humanrobotinteraction.org/journal/index.php/HRI/article/view/9

14. SS64.com. unknown. say – Convert text to audible speech. (unknown). https://ss64.com/osx/say.html [Online; visited on November 24, 2018].

15. Aaron Steinfeld, Odest Chadwicke Jenkins, and Brian Scassellati. 2009. The oz of wizard. *Proceedings of the 4th ACM/IEEE international conference on Human robot interaction - HRI '09* January (2009), 101. http://portal.acm.org/citation.cfm?doid=1514095.1514115

16. Unknown. unknown. eSpeak text to speech. (unknown). http://espeak.sourceforge.net/ [Online; visited on November 24, 2018].

17. Astrid Weiss. 2010. *Validation of an Evaluation Framework for Human-Robot Interaction. The Impact of Usability, Social Acceptance, User Experience, and Societal Impact on Collaboration with Humanoid Robots.* Ph.D. Dissertation. University of Salzburg.