ABSTRACT
Speech systems are typically deployed either over phones, e.g. IVR agents, or on embodied agents, e.g. domestic robots. Most of these systems are limited to a particular platform i.e., only accessible by phone or in situated interactions. This limits scalability and potential domain of operation. Our goal is to make speech interfaces more widely available, and we are proposing a new approach for deploying such interfaces on the internet along with traditional platforms. In this work, we describe a lightweight speech interface architecture built on top of Freeswitch, an open source softswitch platform. A softswitch enables us to provide users with access over several types of channels (phone, VOIP, etc.) as well as support multiple users at the same time. We demonstrate two dialog applications developed using this approach: 1) Virtual Chauffeur: a voice based virtual driving experience and 2) Talkie: a speech-based chat bot.

Author Keywords
Speech interfaces, Web apps, Phone apps, VOIP apps

ACM Classification Keywords
H.5.m. Information Interfaces and Presentation: Miscellaneous

INTRODUCTION
Over the past decade, speech interfaces have provided information access to large population of users. They have been used in a wide range of domains from bus schedules [5] to movies [4]. Spoken language research has benefited from the ability to collect speech corpora from real users calling a system. In the past few years, information access using other platforms, such as smart phone and web have become more common. However, deploying a system on different platforms such as web, phone and smartphone requires significant customization, although both commercial APIs (AT&T Speech API, Android Speech SDK, Microsoft SAPI etc.) and free APIs [6, 1] have emerged. While these development environments reduce the effort needed to customize an application, they restrict the developer to a limited range of acoustic and language models and do not provide an easy option for substituting different decoders or model sets. This limits the scope of implementable applications and limits the ability to experiment with different solutions. On the other hand, seamless interaction is another important factor for a speech applications and certain modes are not available. Phone and situated dialog systems use standard voice activity detection techniques to seamlessly interact with a user. Most web-based applications such as [1], Voxforge.org, Chrome voice search etc., as well as smartphone-based applications such as Siri, Android voice search etc., use a push-to-talk interface that requires explicit action to initiate an input.

Our goal is to enable the easy development of applications that are easily portable to different platforms (web, normal phone and smart phone) and that provide a seamless interface across different realizations. We introduce a spoken language system architecture based on Freeswitch1 and describe two dialog applications of different domains and accessible from different platforms. First, we discuss the underlying architecture behind the applications. Then, we discuss each application and their functionality. Finally, we make concluding remarks.

ARCHITECTURE
Freeswitch is the backbone of this architecture. It supports VOIP, SIP based calling, call routing, and simultaneous calls to a freeswitch application and provides boilerplate code to write an application in many of the popular computer languages. This allows easy integration of existing code as library calls from the application code. Figure 1 shows the flow chart of the architecture. A call is initiated via front-end such as, phone-line, SIP, web-based RTMP connection or VOIP. Then the audio from the user’s channel is streamed to a freeswitch server and recorded to a unix named pipe. A voice-activity-detection component reads the audio stream from the named-pipe to detect speech. This component uses Sphinx’s VAD algorithm (available in sphinxbase library2) to segment the audio into speech and silence chunks. When a speech segment is detected, it is sent to an automatic speech recognizer (ASR) engine via TCP socket. Then the recognition hypothesis is sent to dialog application. The dialog application processes the hypothesis and sends a response back to the user via text-to-speech engine and the application front-end (e.g., GUI). In the current configuration, two speech recognizers are plugged into the system — a local recognizer, Pocketsphinx [2] and a commercial cloud-based recognizer. Any recognition engine can be incorporated (given an appropriate interface), using a simple TCP connection, allowing the

1http://www.freeswitch.org
2http://sourceforge.net/projects/cmusphinx/files/sphinxbase
use of different recognition schemes. Additionally, developers are able to evaluate different (acoustic and language) models in the context of the same dialog application. The source code for VAD, ASR interfaces (C/C++) and example configuration for a freeswitch application is available as tar ball \(^1\). We discuss two dialog applications (written in python) built using this architecture in the following sections.

**APPLICATION1: VIRTUAL CHAUFFEUR**

Virtual Chauffeur is a voice based direction understanding system. Google streetview can be a potential web-platform to experiment with route instruction parsers for autonomous cars. Previously, streetview has been used for direction generation for a virtual pedestrian [3]. In this work, we prototyped a direction understanding system based on streetview. The Freeswitch distribution includes a web based VOIP interface that communicates with the application through RTMP protocol. We embedded streetview and added other html divs to this web page. This web-based front-end is connected to the application as shown in 2 where dotted line shows the architecture-flow discussed earlier. The application is capable of understanding navigation commands such as “turn right”, “go to next block”, “go to a location” etc., and car control commands such as “go faster”, “stop the car”. We use GeoNames.org and Google Places API\(^1\) to geo-locate the location names and point of interests around the current location of the car. The system was tested with Pocketsphinx and the cloud-based recognizer. With pocketsphinx, we used the WSJ acoustic model and a 25K-word trigram model built from AIML pattern phrases. With minor configuration additions in freeswitch server, same application can be accessed from different forms such as, from the web http://www.speech.cs.cmu.edu/apappu/demos/fs-ct, from gmail voice chat or a smart phone with voice chat client cmu.talkie@gmail.com.

**APPLICATION2: TALKIE**

Talkie is a small-talk application that chats about random topics. We used AIML library\(^6\) as the core of this application. The library was imported into the application to get responses for user’s utterances. The front-end is similar to one used in the first application with talking head (using gif images) instead of streetview. Front-end is connected to the dialog application as shown in Figure 3), where dotted line is the pipeline described earlier. The system was tested with both Pocketsphinx and the cloud recognizer. With Pocketsphinx, we used the WSJ acoustic model and a 25K-word trigram model built from AIML pattern phrases. With minor configuration additions in freeswitch server, same application can be accessed from different forms such as, from the web http://www.speech.cs.cmu.edu/apappu/demos/fs-nav.

**CONCLUSION**

In this paper, we presented an architecture that allows easy development of flexible speech interfaces for the web and for other platforms. To demonstrate its use, we developed two example dialog applications using this architecture, currently available of on the web. The web-based architecture can have many advantages. For example, it allows researchers to collect large amounts of data with relatively little investment in resources. It can also help create accessible interfaces for users for a website which is otherwise inaccessible to physically disabled users.

**REFERENCES**


\(^{1}\)http://www.speech.cs.cmu.edu/apappu/pubdl/fs-dialog-interface.tar.gz
\(^{2}\)https://developers.google.com/maps/
\(^{3}\)https://cmusphinx.svn.sourceforge.net/svnroot/cmusphinx/trunk/logios/
\(^{4}\)http://www.alicebot.org/aiml.html