The Cohort and Speechify Libraries for Rapid Construction of Speech Enabled Applications for Android

Tejaswi Kasturi, Haojian Jin, Aasish Pappu, Sungjin Lee
Beverly Harrison, Ramana Murthy, Amanda Stent
Yahoo Labs
{kasturit,haojian,aasishkp,junion,rmurthy,stent}@yahoo-inc.com

Abstract

Despite the prevalence of libraries that provide speech recognition and text-to-speech synthesis "in the cloud", it remains difficult for developers to create user-friendly, consistent spoken language interfaces to their mobile applications. In this paper, we present the Speechify / Cohort libraries for rapid speech enabling of Android applications. The Speechify library wraps several publicly available speech recognition and synthesis APIs, incorporates state-of-the-art voice activity detection and simple and flexible hybrid speech recognition, and allows developers to experiment with different modes of user interaction. The Cohort library, built on a stripped-down version of OpenDial, facilitates flexible interaction between and within "Speechified" mobile applications.

1 Introduction

There are now numerous libraries that provide access to cloud-based ASR and NLU for mobile applications, including offerings from Microsoft[1], AT&T[2] and Nuance[3]. However, speech does not yet live up to its promise on mobile devices. Partly, this is because developers who are not expert speech technologists may make suboptimal decisions regarding interaction management, choice of speech API, and consistency across apps. Also, individual speech-enabled apps are less user-friendly than an app *ecosystem* within which a user may move fluidly from GUI interaction to hands/eyes-free interaction and from one app to another as interest and attention demand. In this paper we present the Speechify/Cohort libraries for development of speech-enabled Android apps. Speechify enables rapid development of usable speech-enabled applications; Cohort allows the user of a suite of Speechified applications to be hands-free/eyes-free when they need it, to use the rich multimodality of the applications themselves when they want to, and to move naturally and fluidly within and between applications. The Speechify and Cohort libraries will be made available on github[4].

2 Speechify

The Speechify library is designed to solve the following problem: when an organization is speech enabling a suite of mobile applications, it is easy to end up with a poor user experience because of inconsistencies in implementing features like:

- choice of speech API - Speechify wraps several publicly available speech recognition and speech synthesis APIs, including the Google Android[5] and Pocketsphinx (Huggins-Daines et al., 2006) speech recognition engines and the Google Android and Ivona speech synthesizers.

- mode of user interaction - Speechify supports push-to-talk, wait-to-talk, and always-on speech recognition (see Section 2.1). In addition, it can detect when the user is moving and switch to speech interaction.

- hybrid speech recognition - Speechify includes a module for tunable hybrid embedded/cloud-based speech recognition, to permit the speed of embedded recognition for command and control, with the flexibility of cloud-based recognition for open-vocabulary input (see Section 2.2).

[1] https://www.projectoxford.ai/
[2] http://developer.att.com/sdks-plugins
[3] http://dragonmobile.nuancemobiledeveloper.com
[4] https://github.com/yahoo
[5] http://developer.android.com/reference/android/speech/

Proceedings of the SIGDIAL 2015 Conference, pages 441–443,
Prague, Czech Republic, 2-4 September 2015. ©2015 Association for Computational Linguistics

- voice activity detection and acoustic echo cancellation - Speechify includes a state-of-the-art voice activity detection/acoustic echo cancellation module, allowing more accurate speech input in noisy environments (see Section 2.3).

2.1 Interaction Management

Speechify is built for applications that may or may not require hands-free interaction, depending on the user's other activities. Therefore, Speechify supports three modes for interaction:

- push-to-talk - In this mode, when the user taps any non-interactive part of the application's display, a microphone icon is presented in a transparent overlay on the application, and the user may speak. We do not use a push-to-talk button because a user who is driving, running or walking may not have attention to spare to find a push-to-talk button.

- wait-to-talk - In this mode, when the application is not itself talking, a microphone icon is presented in a transparent overlay on the application to indicate that the application is listening. In this mode, the user cannot "barge in" on the system.

- always-on - In this mode, the application is always listening, even when it is itself talking. We use state-of-the-art voice activity detection and acoustic echo cancellation to minimize recognition errors in this mode (see Section 2.3).

Speechify supports one additional feature for interaction management: it incorporates movement detection, so that when the user starts moving it can switch to always-on mode.

In addition to providing the developer with flexibility to experiment with different modes for speech interaction, the microphone overlay and speech control menu provided by Speechify enable a consistent interface and interaction for the user across multiple "Speechified" applications.

2.2 Hybrid Recognition

Cloud-based speech recognition offers unparalleled ease of access to high-accuracy, large vocabulary speech recognition. However, even on fast networks the latency introduced by cloud-based recognition may negatively impact ease of use for speech-enabled applications. For many applications, the majority of speech input is aimed at command and control (requiring only a small, fixed vocabulary), while a minority requires a very large open vocabulary (especially for search). A hybrid recognition approach may offer a good trade-off of accuracy and speed.

Recognizer	WER	RTF
Google	18.16	0.67
PocketSphinx	38	0.15
Hybrid (threshold=47000)	16.45	0.57

Table 1: Hybrid recognition can give simultaneous improvements in recognition accuracy and recognition speed

There are three general approaches to hybrid recognition: a voted combination of multiple recognizers run in parallel (Fiscus, 1997); lattice rescoring of the outputs of multiple recognizers (Richardson et al., 1995; Mangu et al., 1999); or heuristic selection of recognition output. Only the third is currently an on-device option. Speechify supports tunable on-device heuristic selection between (a) the output of any wrapped cloud-based recognizer, and (b) the output of PocketSphinx, an embedded recognizer.

To assess the tradeoffs for hybrid recognition, we ran an experiment using the Google Android cloud-based recognizer and PocketSphinx. For PocketSphinx we used an off-the-shelf acoustic model trained on broadcast news speech, with a grammar based on the prompts recorded by the speakers. We used 38 prompts each recorded by 7 speakers (from both genders, and with a variety of accents) in a noisy environment, for a total of 266 utterances. The results in terms of word error rate (WER) and real time factor (RTF; processing time / utterance length) are shown in Table 1. We get a small decrease in real time factor, along with a useful increase in recognition accuracy, through the use of hybrid recognition.

2.3 Voice Activity Detection

In a noisy environment or when the phone is in speaker mode, background noise or system speech may cause high rates of recognition errors for speech-enabled mobile apps. Speechify includes a state-of-the-art, on-device module for voice activity detection and acoustic echo cancellation. The module uses a three-stage process: feature extrac-

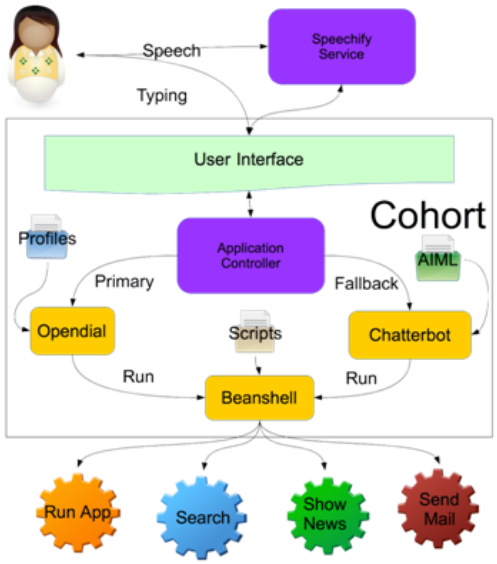

Figure 1: Cohort Architecture

tion, classification of short regions of voice activity, and smoothing. Only low-level acoustic features (such as MFCCs) are used. The classifier for both classification and smoothing is a lightweight random forest implementation. On the data from the recent IEEE AASP voice activity detection challenge (Giannoulis et al., 2013), our approach obtained very good F scores of 98.7% (balanced test data) and 97.6% (unbalanced test data).

3 Cohort

The Cohort library is a wrapper around a stripped-down version of OpenDial (Lison, 2013), and is designed to allow the user to navigate hands-free within and between Speechified applications. Cohort frees the application developer to think mainly about the functionality of the application itself, while Cohort deals with aspects related to speech interaction.

The architecture of Cohort and Speechify is presented in Figure 1. Participating apps must be Speechified, and must supply an application profile (as an OpenDial rule set). Cohort:

- automatically registers all participating apps on each user's phone
- passes control to each app as requested by the user through speech input
- handles contextual disambiguation of commands like "list" that apply to multiple apps, through the Opendial machinery
- supports basic speech commands like "pause" or "start" for all apps

The Cohort library also wraps a simple AIML-based chatbot, suitable for jokey conversations or backoff in case of errors. Cohort comes with a simple text input interface for debugging.

4 Demo

In general it takes less than 25 lines of code to Speechify an app, and about the same number of lines of OpenDial rules to subscribe an app to Cohort. We have "Speechified" and "Cohort-subscribed" three mobile applications: an email client, a news reader, and an Android "home page" app. We will demonstrate user interactions through and between these apps, and present the code necessary to make each app work in the Speechify/Cohort framework.

References

Jonathan G Fiscus. 1997. A post-processing system to yield reduced word error rates: Recognizer output voting error reduction (rover). In *Proceedings of ASRU*.

Dimitrios Giannoulis, Emmanouil Benetos, Dan Stowell, Mathias Rossignol, Mathieu Lagrange, and Mark D Plumbley. 2013. Detection and classification of acoustic scenes and events: an ieee aasp challenge. In *Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*.

David Huggins-Daines, Mohit Kumar, Arthur Chan, Alan W Black, Mosur Ravishankar, and Alex Rudnicky. 2006. Pocketsphinx: A free, real-time continuous speech recognition system for hand-held devices. In *Proceedings of ICASSP*.

Pierre Lison. 2013. Ph.D. thesis, University of Oslo.

Lidia Mangu, Eric Brill, and Andreas Stolcke. 1999. Finding consensus among words: lattice-based word error minimization. In *Proceedings of Eurospeech*.

Frederick Richardson, Mari Ostendorf, and JR Rohlicek. 1995. Lattice-based search strategies for large vocabulary speech recognition. In *Proceedings of ICASSP*.