Building transcribed speech corpora quickly and cheaply for many languages

Thad Hughes, Kaisuke Nakajima, Linne Ha, Atul Vasu, Pedro Moreno, Mike LeBeau

Google Research, USA

{thadh,kaisuke,linne,atulvasu,pedro,mlebeau}@google.com

Abstract

We present a system for quickly and cheaply building transcribed speech corpora containing utterances from many speakers in a variety of acoustic conditions. The system consists of a client application running on an Android mobile device with an intermittent Internet connection to a server. The client application collects demographic information about the speaker, fetches textual prompts from the server for the speaker to read, records the speaker's voice, and uploads the audio and associated metadata to the server. The system has so far been used to collect over 3000 hours of transcribed audio in 17 languages around the world.

Index Terms: speech corpora, speech recognition, internationalization

1. Introduction

Transcribed speech corpora are the lifeblood of systems used to construct acoustic models, but recording and transcribing new speech corpora can be time consuming and expensive. When training acoustic models, researchers often rely on well-known corpora such as those developed by the Linguistic Data Consortium, such as Switchboard[1]. However, pre-existing corpora have certain disadvantages; some were recorded over telephony channels with low quality audio, some lack a diverse vocabulary, and some are expensive or have restrictive licenses. Pre-existing corpora often don't match real-world usage conditions, from the utterances, to the microphones, to the background noise. Additionally, no large, transcribed speech corpus exists for many of the world's languages. It is therefore sometimes necessary to construct a new transcribed speech corpus.

In the past, constructing a large transcribed speech corpus was time consuming and expensive. If new speech is recorded, speakers must be recruited and somehow brought into the vicinity of audio recording equipment. Often telephones are used as relatively ubiquitous recording devices conveniently located near the speakers, but telephony channels have acoustic limitations and telephones are not always available in remote areas. Using portable voice recorders is another option, but these devices can't easily associate metadata with each utterance, so bookkeeping must be done by hand. In many cases, the recorded speech must also be transcribed by human listeners, which is labor intensive and often error prone.

Our system solves many of these problems and makes collecting large, transcribed speech corpora relatively quick and easy. Like [2], we use commodity mobile phones running the Android platform to record and store many hours of high-quality read speech in various recording environments, which provides a better match to the acoustic conditions encountered in real-life usage. The phones also associate metadata with each recorded utterance, such as information about the speaker and recording environment, and the prompt the speaker was asked to read, thus automatically associating a transcript with each utterance. An arbitrarily large number of phones can be used simultaneously to parallelize the data collection effort while easily keeping track of associated metadata. Finally, the phones can use their networking capabilities to automatically send the utterances and metadata stored on each phone to a centralized storage location for later use. These advantages have drastically reduced the cost of speech data collection and corpus construction, and thereby assisted with our efforts to internationalize speech technology.

The remainder of this paper will describe in detail the system and our experiences using it to build speech corpora. Section 2 describes the architecture of the client and server components of the system, section 3 describes how we use it to collect data, and section 4 describes some characteristics of the corpora we have constructed.

2. System architecture

The system has a client/server architecture, where the client application is designed for mobile devices running Google's Android platform [4] and the server handles HTTP requests from the client. To enable usage in places without Internet connectivity, such as remote areas or developing countries, the client only requires intermittent connections to the server to download prompts and upload recorded utterances.

2.1. Client system

The client is what speakers use to read prompts and record their voices. It is a standard Android application, which means that it can be installed and run on any mobile phone device that uses the Android platform, such as the Android G1 and the Nexus One.

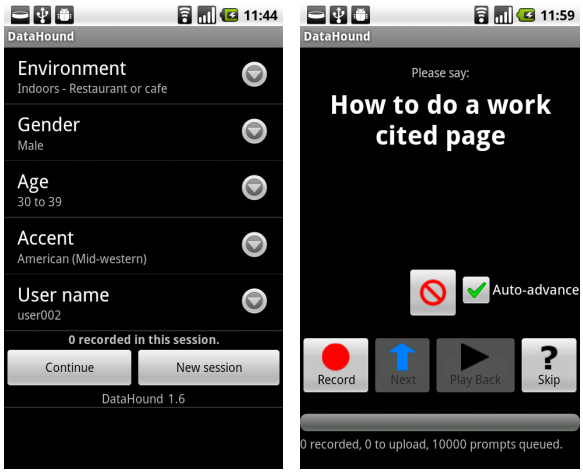
2.1.1. User interface

The user interface (UI) is session-based. When a speaker begins a session, they are presented with an initial screen to collect metadata relevant to the session, which the user may decline to provide. The UI, shown in figure 1a, currently collects metadata containing these fields:

- The acoustic environment (indoors, outdoors, in an automobile, level of background noise)
- Speaker's gender
- Speaker's age, grouped into decades
- Speaker's accent
- A user name to identify the session

Additionally, the application also automatically tags each session with this information:

- Current date and time



(a) The initial screen, presented to the user at the beginning of each session to collect information about demographics and acoustic conditions.

(b) The recording screen, which gives realtime feedback about audio levels and allows the user to skip prompts or mark them as offensive.

Figure 1: The client UI running on the Google Nexus One.

- Telephone's hardware version and Android OS version
- Telephone's IMEI number (a globally unique identifier for the mobile device)
- Geographic location as determined by the phone's GPS

This information is associated with every utterance recorded during the current session.

Once the speaker presses the *Continue* button on the session metadata screen, they see the main recording screen, shown in figure 1b. The top portion of the screen prompts the user with a short phrase of text to read. (These textual prompts have been downloaded from the server as described in section 2.1.2.) At the bottom of the screen, there is a horizontal bar that displays the current audio volume level sampled by the device, which is updated in real-time to help the speaker position the device and speak at an appropriate volume level. When the speaker is ready, they click the *Record* button to begin recording, speak the prompt text, and then click the *Record* button again to terminate recording. The speaker can then play back the recorded utterance, and when satisfied, continue to the next prompt.

Sometimes, the speaker might not understand or prefer not to say the text presented in the prompt. The text might be in a language that the speaker doesn't understand, it may use words or URLs that the speaker doesn't know, or it might contain offensive content. For example, many English speakers might not feel comfortable reading the text "nvidia geforce fx 5200." Whenever the speaker wants to skip an unknown prompt, they can do so by pressing the button marked with a question mark (?). If the speaker feels the prompt is offensive, they can mark it as such by pressing the red crossed-out circle button (⃠). In either case, the prompt is skipped, the server is notified of the user's action, and the user is presented with the next prompt.

The UI is optimized to allow a speaker to rapidly record many prompts in succession, which is important for maximizing the speaker's time before they become fatigued or bored. In particular, once the speaker is comfortable using the program, they can activate the *Auto-advance* checkbox, which causes the application to automatically continue to the next prompt and begin recording when the speaker presses the button to terminate the current utterance. This means that the speaker is only required to press a single button per utterance, making it possible to advance through the utterances rapidly.

The client application keeps track of how many utterances have been recorded during the session, and when the required number, typically several hundred, has been recorded, the user is notified and the session is ended.

2.1.2. *Client implementation*

The client, diagrammed in figure 2, is a standard Android application written in Java that uses Android's raw audio recording and playback APIs to collect audio and standard Java APIs to communicate with the server and read and write the SD card. The client requires Android platform release 1.5 (Cupcake) [4] or later, because this release added the raw audio recording and playback APIs used by the application.

As soon as the recording screen has been shown to the speaker, the program immediately begins recording audio and never stops, whether or not the user has actually initiated recording. This allows the onscreen audio volume display to be updated in real-time, showing the current ambient volume level before actual recording begins. When the speaker presses the *Record* button to initiate and terminate an utterance, the program stores the recorded audio to a file, including audio recorded 0.5 seconds before initiation and after termination of recording. This enables the device to record the ambient noise surrounding the speaker's utterance, and to ensure that the speaker's utterance is not accidentally truncated. The sampling format and rate default to 16-bit linear, 16kHz monaural audio but are fully configurable.

The client makes use of the phone's SD card to store information and minimize the necessity of server communication. When the program is first run in a particular language, the client downloads a large list of textual prompts for the speaker to read and stores them in a text file on the SD card for later use.

The SD card is also used to store the audio recordings made by a speaker until they can be uploaded to the server. When a speaker finishes recording a prompt, the audio is written to a file, along with the relevant metadata for that utterance, including the prompt which the speaker read. Android SD cards are typically at least 1 GB and hold over 8 hours of 16-bit, 16kHz monaural audio, meaning that the program can be used for an entire day without communicating with the server.

The Android version of the client is actually the third generation client we developed to collect these kinds of speech corpora, as earlier versions were written for iPhone and BlackBerry. Key features of the Android client that made it more successful than earlier versions are its ability to operate offline and the ability for a speaker to quickly move through many utterances using the *Auto-advance* mode.

2.2. Server application

The server supplies the client with text prompts and receives and aggregates recorded utterances and associated metadata from the client through an HTTP interface. The server is implemented in Java, and also supports authentication to prevent unauthorized users from accessing prompts or uploading recordings.

Whenever the client application needs more text prompts for the user to read, it sends the server an HTTP request, that includes a parameter for the language of the prompts. The server responds with text containing several prompts, one on each line. The server constructs the list of prompts sent to the client by

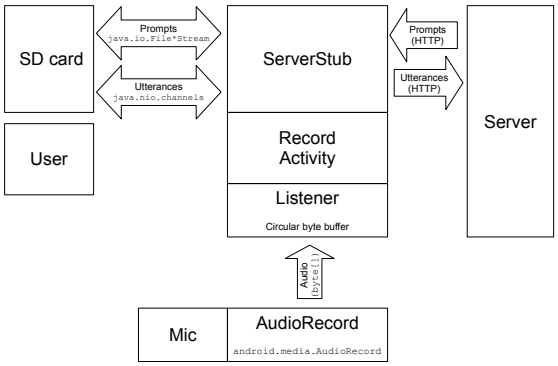


Figure 2: The client implementation.

randomly sampling from a large pre-defined list of prompts in the specified language. There are no provisions to eliminate duplicates prompts, so they sometimes occur.

The server receives recordings from the client via HTTP POST requests. The utterance metadata is sent using parameters in the HTTP request, and the accompanying audio is sent in the body of the request. When the server has received the entire utterance, it stores the utterance and the associated metadata into a central repository and informs the client that the utterance can safely be deleted from its SD card.

3. Data collection

3.1. Prompt preparation

The server must be configured with a large list of text prompts for the speakers to read. To generate these lists we used a sample of common typed Google search queries, and in some instances an additional set of phrases in the target language. For each target language, we used a statistical language classifier and information about the user's locale settings to find common Google search queries in that language. We then removed misspelled words and pornographic terms from the list. Finally, the remaining queries were sampled uniformly and become the prompts given to speakers.

There are significant reasons for using web queries to prompt speakers. Most importantly, we used these corpora to train initial acoustic models for Google search by voice[3], and we expect that typed and spoken web queries would have similar distributions. A related advantage is that the web query data contains many modern words, such as "pokemon" and "openvpn," that are not present in standard pronunciation lexicons. Having users read these words can give us an idea about how they are actually pronounced, as discussed in section 4.3.

One drawback of using web queries as prompts is that speakers will sometimes be given prompts they don't understand and make speaking errors when reading them. A common issue was foreign words in the prompt list. Sometimes this was desirable; English speakers, for example, often say foreign phrases such as "bon appétit." But the prompt lists also contained lesser known foreign words which were either mispronounced or skipped by speakers. The distribution of typed and spoken search queries can also be different; for example, English words comprise about 15% of typed Chinese web queries, but a much smaller percentage of spoken queries. It seems that users in some locales are more comfortable typing English than speaking it.

3.2. Audio recording in the field

The system is ideal for research purposes because it can capture rich data in a variety of environments. Before a phone is ready for field use, it must be configured for a particular language and environment. We use Android's locale settings to set the phone's UI language, and a *Preferences* screen in the client application controls the prompt language, the number of recordings for each session, and the number of cached prompts. Sometimes it is useful to lower the number of recordings per session to capture a more diverse group of speakers and accents while using less of each speaker's time. When recording in environments without Internet connectivity, it is important to pre-fetch a larger number of prompts onto the SD card.

Once the phones are configured, a single session of 500 utterances takes an average of 30 minutes of recording time in *Auto-advance* mode, and can be completed in as few as 20 minutes if the speaker reads quickly. Educating the speaker ahead of time can speed up the process. We walk the speaker through recording the first prompt manually, and then play back the recording to make sure the speaker is engaged in each step. The speaker then activates *Auto-advance* mode once they feel comfortable with the process.

Using one device, it is possible to record 8,000 utterances per day. To record sessions in real-life operating conditions, such as outdoors, on the street or even in a public transportation station, it is essential to have extra batteries and a fully-loaded SD card because the G1's battery lasts about 3 hours under constant use. Disengaging the device's radio and exiting unnecessary applications helps extend the battery life.

3.3. Crowd-sourcing

The system's ease of use and comparatively inexpensive setup make it possible for a large number of unskilled people to collect speech data in parallel. In many locales, we engaged university students, working relatively independently in the tradition of census takers and surveyors, to execute the data collection. The students set a target number of speakers with certain demographic and recording environment distributions and were able to leverage their social networking communities to recruit speakers. Hiring university students can also be efficient in that they are often just entering the work force and many are edge technology users requiring less technical support.

4. Corpus characteristics

The speech corpora collected so far using the system have several common characteristics. The utterances are fairly short, typically just a few words, and they are surrounded by a margin of background noise at the beginning and end. Each utterance is annotated with metadata about the speaker as described in section 1a. And finally, each utterance is also annotated with the prompt given to the speaker to read, which we use as a transcript when training acoustic models.

4.1. Speaker errors

Speakers do not always read the prompt text perfectly during a recording session. There are occasional speaking errors and repetitions, extraneous comments, and errors using the UI. This means that the prompt text is only an approximation of the ac-

Error category	Rate	German example
Misread	3.5%	“fa**san**erie” → ”fa**nas**ierie”
Side-speech/noise	2.5%	Coughing, extra commentary “scheffler” → “quatsch scheffler”
Restart	2.0%	“fm09” → ”f m oh- f m null neun”
Truncation	1.5%	“kinoprogram” → “-gram”
Empty	0.5%	“konstanz” → “ ”

Table 1: Categories of errors made by German speakers. Around 10% of utterances contain a speaking error, which compares favorably with the error rate achieved using human transcribers.

tual transcript. To estimate the rate of these kinds of errors, we hand-transcribed a random sample of German recordings, with the results shown in table 1. Approximately 10% of the utterances contain some sort of discrepancy between the prompt text and what was actually spoken, which is less than the typical error rate of human transcribers. Therefore, we made no special provisions to deal with the erroneously pronounced utterances when training our acoustic models, although about 2% of them are discarded automatically from the training process because the Viterbi forced alignment algorithm fails to find an appropriate alignment.

4.2. Evaluating recognition performance

We collect the speech corpora to train acoustic models, so it is natural try to understand the quality of these corpora for that purpose. However, in many cases, we don't have another transcribed corpus against which we can evaluate the acoustic models trained with the new corpus. To better understand the quality of the data collected, we split the new corpus into training and testing sets containing 80% and 20% of the data, respectively. We group the utterances by session prior to splitting them to ensure that no speaker is present in both the training and test sets.

Figure 3 shows a normalized histogram of the sentence accuracy of the trained acoustic model on both the training and test data. Naturally, the model performs better on the training data, where sessions with higher accuracy are more common. However, the histogram also reveals that the acoustic model performs very poorly even on some training sessions. Deeper investigation reveals that these sessions had very low signal-to-noise ratios or suffered from systematic user errors.

The metadata associated with each utterance allows us to analyze how various factors affect recognition performance. The most significant factor affecting performance is the acoustic environment in which the recordings were made. Sentence accuracy on the test set can be as much as 50% worse for recording environments with persistent sources of non-stationary noise, such as inside noisy restaurants or with music or TV in the background.

4.3. Identifying incorrect lexicon entries

The prompts our speakers recorded contain many modern words that are not present in standard lexicons. Our recognizer generates pronunciations for these words using letter-to-sound rules, which do not always work properly for words such as “pokemon” or “openvpn.” We do not have tools to automatically fix incorrect pronunciations, but we were able to semi-

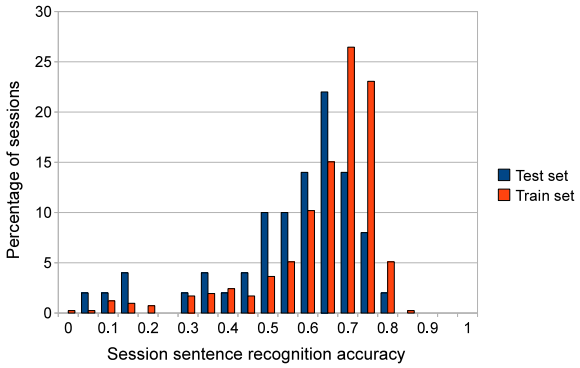


Figure 3: Histogram showing the per-session sentence recognition accuracy of the training and test data.

automatically identify them by running a maximum likelihood trained recognizer on the training data and searching for words that were consistently misrecognized; we found that MMI was able to compensate for a significant percentage of the incorrect pronunciations in the lexicon, making it harder to identify the incorrect pronunciations. More work remains to automate the process of learning pronunciations from spoken examples of words.

5. Conclusions

Our system has proven to be a very efficient tool for building corpora in a variety of languages. We have already used it for several languages, and have only just begun to tap into the various ways that the data can be utilized and analyzed.

6. Acknowledgements

The authors would like to thank Vivek Kumar, Martin Jansche, Amir Mane, Gummi Hafsteinsson, Dave Burke, Bill Byrne, Mike Schuster, Trausti Kristjansson, Pankaj Risbood, Etienne Barnard, and Alta de Waal for contributing to this project.

7. References

[1] Godfrey, John J., and Hollman, Edward, “Switchboard-1 Release 2”, Linguistic Data Consortium, Philadelphia, 1997.

[2] T. Hazen, E. Weinstein, R. Kabir, A. Park, and B. Heisele. Multi-Modal Face and Speaker Identification on a Handheld Device. In Proceedings, Workshop on Multimodal User Authentication (MMUA) 2003, pp. 113-120, December 2003, Santa Barbara, CA, USA.

[3] J. Schalkwyk, D. Beeferman, F. Beaufays, B. Byrne, C. Chelba, M. Cohen, M. Kamvar, and B. Strope, “Google Search by Voice: A case study,” in Visions of Speech: Exploring New Voice Apps in Mobile Environments, Call Centers and Clinics, A. Neustein, Ed. Springer, 2010 (in press).

[4] ”Android 1.5 Platform”, Google, Mountain View, CA. Online: http://developer.android.com/sdk/android-1.5.html, accessed on 26 Mar 2010.