A Web-Based Tool for Developing Multilingual Pronunciation Lexicons

Samantha Ainsley[1,2], *Linne Ha*[3], *Martin Jansche*[3], *Ara Kim*[2], *Masayuki Nanzawa*[3]

[1] Department of Computer Science, Columbia University, USA

[2] Work done at Google Inc., USA

[3] Google Inc., USA

{sainsley, linne, mjansche, mnanzawa}@google.com, ara@googlealumni.com

Abstract

We present a web-based tool for generating and editing pronunciation lexicons in multiple languages. The tool is implemented as a web application on Google App Engine and can be accessed remotely from a web browser. The client application displays to users a textual prompt and interface that reconfigures based on language and task. It lets users generate pronunciations via constrained phoneme selection, which allows users with no special training to provide phonemic transcriptions efficiently and accurately.

Index Terms: pronunciation lexicons, speech recognition, text to speech, internationalization

1. Introduction

A high quality pronunciation lexicon is at the heart of any robust speech recognition system. However, commercially available pronunciation lexicons are often expensive and lack modern non-dictionary terms that occur frequently in speech recognition applications. Most existing pronunciation dictionaries do not have sufficient coverage for the enormous variety of proper names one encounters in spoken web search (our domain of interest), including many unusual names of people, organizations, places, websites, etc. On the other hand, custom lexicons are labor intensive to generate, requiring specialized linguistic training and are prone to errors and inconsistencies. However, for resource-scarce languages as well as for very broad coverage or specialized applications in resource-rich languages, developing new pronunciation lexicons cannot easily be avoided.

Related work by Davel and colleagues [1, 2] has shown that it is possible for "developers with limited linguistic experience [to] develop accurate pronunciation models" [1] using a development process with appropriate tool support. We follow an approach similar to the one advocated by Davel et al. for bootstrapping pronunciation lexicons. In this paper, we describe the design of a web-based tool for entering pronunciation information about words. Our tool differs from Davel and Peche's *DictionaryMaker* [3] in several ways: *DictionaryMaker* is a stand-alone Java application that users have to download and install, whereas our tool is a web application that only requires a browser and working internet connection;[1] our web-based client-server application takes care of downloading and uploading data without user intervention; and our tool offers additional transcription or annotation modes described below.

Like *DictionaryMaker* our system solves the issues faced in generating phonemic transcriptions for customized pronunciation lexicons by constraining the transcribers input to a fixed phoneme inventory and providing audio feedback on input to minimize errors. The user interface maps individual phonemes to relevant keyboard letters to provide transcribers with an intuitive and efficient means of input. Transcribers are additionally provided with audio previews and example words illustrating individual phonemes, eliminating the need for specialized knowledge of the phoneme set for a given language. The phonemic transcription interface is used both for entering a small seed dictionary of pronunciations and for correcting pronunciations predicted by a model trained on the seed dictionary.

The client-server architecture of our system additionally decreases the time required to generate new lexicons by allowing multiple users to collectively build the same lexicon. Pronunciation data entered in the client side of the application is continuously written to a centralized storage location for later processing. The system allows for grouping of word lists into smaller tasks that can be assigned to transcribers on a rolling basis for an efficient work flow by letting faster transcribers take on more tasks. Additionally, the same word can be assigned to multiple transcribers for increased accuracy and to check agreement between transcribers. These features have greatly improved efficiency and accuracy in our experience.

The lexicon-editing tool is implemented as a web application on top of Google App Engine [4], or GAE. This provides a scalable framework that allows us to concentrate on application development without having to worry about maintaining production servers. GAE also provides components for user authentication as well as a distributed datastore. The User Interface (UI) is implemented with the help of the Google Web Toolkit [5], or GWT. This lets us focus on expressing the UI and the application logic as a Java program. GWT then compiles the Java code into HTML, CSS, and JavaScript to be interpreted by the browser on the client side. For cross-browser audio support, we use the gwt-voices sound library [6].

In general the client retrieves pronunciation prompts through asynchronous calls to the server, which looks up available tasks in the GAE datastore. The UI for working with these prompts runs entirely in the browser. The client then passes the pronunciation or annotation generated by the user back to the server for persistence in the datastore. From the user's point of view, these transactions and most of the other inner workings of the system are invisible. Users only need to point a supported web browser (Chrome, Firefox, or Safari) at the URL for our tool and sign in with a Google Account. They can then retrieve assigned tasks and work on them inside the browser. There is no need to install extra software or to explicitly download data for their work.

2. User interface

The UI is designed to support two primary user experiences: An efficient means for expert users to transcribe words quickly, and an intuitive means for naïve users without special training in phonetics or phonology to provide accurate transcriptions.

For a transcriber the interface is composed of two components: a pronunciation editor (e.g. Figure 1) and a task manager sidebar (not shown). The pronunciation editor displays the current prompt (the word or phrase to be transcribed) above a phoneme display box (show-

[1] This requirement is not without problems in the developing world.

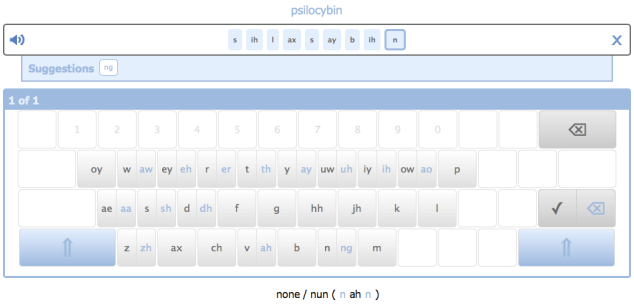

Figure 1: *Pronunciation editor: phoneme display, virtual keyboard, and example words.*

ing the phonemic transcription in progress) and a virtual phoneme keyboard. The user is asked to transcribe the prompt into a string of phonemes taken from a pre-defined phoneme inventory. We provide the user with a virtual phoneme keyboard (Figure 1), which maps physical keys to individual phonemes. The keyboard configures its display based on the task language, and handles key substitutions for language-specific keyboard layouts. The situation for US English is shown in Figure 1: It uses a standard QWERTY keyboard onto which phoneme symbols in ArpaBet notation have been projected. For example, the (unshifted) "E" key yields the tense /ey/ phoneme and Shift + "E" yields the lax /eh/ phoneme.

The entire phoneme inventory is thus always visible and the phoneme-to-key mapping is visually reinforced. Phoneme keys in the on-screen virtual keyboard respond both to mouse clicks as well as physical key presses. Novice users can pick phonemes with the mouse, while advanced users can simply type once they've internalized the phoneme-to-key mapping. We provide both audio feedback and example words for individual phonemes. When the user hovers their mouse over a phoneme key in the virtual keyboard, an example word for that phoneme appears together with its transcription. In Figure 1 the example *none* or *nun* and its phonemic transcription /n ah n/ appears as an illustration of /n/. If available, we also provide audio recordings of the example word and the isolated phoneme, which play when the corresponding virtual or physical key are touched. This further reinforces the phoneme-to-key mapping acoustically.

Additionally, the transcriber can review their entry and listen to the entire phoneme sequence by selecting the speaker button on the left side of the editor. The resulting concatenation of isolated phonemes is far from ideal; however, it has a precedent in *DictionaryMaker* [3], where it was found to be helpful [1].

The selected phonemes display in a text editor box with a cursor for insertion and deletion control. The phoneme preceding the cursor is highlighted and one or more phoneme suggestions – typically easily confused alternatives – are displayed below the editor. In Figure 1 the phoneme /n/ in the text editor box is highlighted and the alternative /ng/ is suggested in a shaded box below. The transcriber can swap the highlighted phoneme with one of its suggestions by pressing the corresponding numeric key (i.e. "1" to take the first suggestion). Suggestions also respond to mouse clicks. When the transcriber has completed the pronunciation, they can hit the "Enter" key to save their work and they are automatically advanced to the next prompt.

When a user first logs into the application, the client requests a list of tasks currently assigned to that user. Each task is a collection of prompts the user has to transcribe. The user selects a task from a drop down list in the task manager sidebar and the server sends the prompts associated with the task – along with any pronunciations already supplied by the user – back to the client and the task displays in the user interface as previously described.

As soon as the user marks a phoneme sequence as complete, the client sends the pronunciation to the server to be written to the GAE datastore along with a unique identifier of the transcriber. This allows us to collect multiple pronunciations for the same prompt and track disparities between transcribers. Pronunciation data is sent back to the server with each completed task item in order to minimize the amount of data held on the client side in the event of a lost connection or expired user log-in. The user is advanced to the next prompt and a check mark is placed by the current prompt only if the client receives a successful response from the server, in order to minimize data loss.

As previously mentioned, the tool is also intended to allow users with no linguistic training to intuitively generate phonemic pronunciations. Volunteers enter their own prompt in a text field and generate its pronunciation via the phonemic keyboard. Whereas the key bindings provide efficiency for transcribers, they also provide an intuitive display for the untrained user who naturally associates phonemes with letters. With meaningful key mappings, the naive user can retype the original prompt to yield a rough estimate of its pronunciation. They can then refine the pronunciation by listening to the phoneme previews and replacing ambiguous with phones with pre-populated suggestions.

Early versions of our tool were focused on simplifying the work of transcribers, especially through the use of phonemic keyboards. However, the creation of these keyboards used to happen outside the tool and was itself quite error-prone. We found it helpful to implement a graphical editor for phoneme keyboards within recent versions of the tool. It allows administrators to design keyboard layouts, define phoneme inventories, and graphically edit phoneme-to-key mappings.

3. Conclusions

We have presented a simple web-based tool for lexicographic work that arises in many speech projects. Our goal has been to enable language experts in remote locations to provide information about the pronunciation of words quickly and accurately. A virtual on-screen keyboard displays and continually reinforces a mapping from keys to phonemes. As the experts become more and more familiar with this task, they can increasingly rely on their physical keyboards to enter phonemes, thus speeding up their work. Thus far, we have used our tool to gather lexicons of several tens of thousands of words in four languages from language experts who have little to no prior experience with building pronunciation lexicons.

4. Acknowledgments

We would like to thank our colleagues from the Google Speech Group for helpful discussions. Special thanks to Kaisuke Nakajima for detailed feedback on design and coding. Many thanks also to our language experts for their detailed feedback on our tool, which has shaped its current look and functionality.

5. References

[1] Davel, M. and Barnard, E., "The efficient generation of pronunciation dictionaries: Human factors during bootstrapping", Interspeech/ICSLP 2004.

[2] Davel, M. and Martirosian, O., "Pronunciation dictionary development in resource-scarce environments", Interspeech 2009.

[3] Davel, M. and Peche, M., "DictionaryMaker user manual, version 2.0 (i)", 2006. Online: `http://dictionarymaker.sourceforge.net/doc/dictmaker_manual_v2.0.pdf`, accessed on 29 March 2011.

[4] "Google App Engine", Google, Mountain View, CA. Online: `http://code.google.com/appengine/`, accessed on 19 Sept 2010.

[5] "Google Web Toolkit", Google, Mountain View, CA. Online: `http://code.google.com/webtoolkit/`, accessed on 19 Sept 2010.

[6] Sauer, F., "gwt-voices: Sound library for Google-Web-Toolkit (GWT)", 2009. Online: `http://code.google.com/p/gwt-voices/`, accessed on 29 March 2011.