lex4all: A language-independent tool for building and evaluating pronunciation lexicons for small-vocabulary speech recognition

Abstract

This paper describes *lex4all*, an open-source PC application for the generation and evaluation of pronunciation lexicons in any language. With just a few minutes of recorded audio and no expert knowledge of linguistics or speech technology, individuals or organizations seeking to create speech-driven applications in low-resource languages can use this tool to build pronunciation lexicons enabling small-vocabulary speech recognition in the target language using a high-quality commercial recognition engine designed for a high-resource source language (e.g. English). This is possible thanks to an existing algorithm for cross-language phoneme-mapping; we give an overview of this method and describe its implementation in *lex4all*. Beyond the core functionality of building new lexicons, the tool also offers a built-in audio recorder that facilitates data collection, and an evaluation module that simplifies and expedites research on small-vocabulary speech recognition using cross-language mapping.

1 Introduction[1]

In recent years it has been demonstrated that speech recognition interfaces can be extremely beneficial for applications in the developing world, particularly in communities where literacy rates are low (Sherwani and Rosenfeld, 2008; Bali et al., 2013; Sherwani, 2009). Typically, the languages spoken in such communities are under-resourced, such that the large audio corpora typically needed to train or adapt recognition engines are unavailable. However, in the absence of a recognition engine trained for the target low-resource language (LRL), an existing recognizer for a completely unrelated high-resource language (HRL), such as English, can be used to perform small-vocabulary recognition tasks in the LRL. All that is needed is a pronunciation lexicon mapping each term in the target vocabulary to one or more sequences of phonemes in the HRL, i.e. phonemes which the recognizer can model.

This is the motivation behind *lex4all*,[2] an open-source desktop application for Windows that allows users to automatically create a mapped pronunciation lexicon for words in any language, using a small number of audio recordings and a pre-existing recognition engine in a HRL such as English. The resulting lexicon can then be used with the HRL recognizer to add small-vocabulary speech recognition functionality to applications in the LRL, without the need for the large amounts of data and expertise in speech technologies required to train a new recognizer. This paper describes the *lex4all* application and its utility as a tool for rapid creation and evaluation of mapped pronunciation lexicons for small-vocabulary speech recognition in any language.

2 Background and related work

Many commercial speech recognition systems offer high-level Application Programming Interfaces (APIs) that make adding voice recognition capabilities to an application extremely simple, and require very little general technical expertise and virtually no knowledge of the inner workings of the recognition engine. If the target language is supported by the system – the Microsoft Speech Platform, for example, currently supports 26 lan-

[1] Parts of this paper (Sections 1 and 2) overlap with a paper submitted to the 4th Workshop on Spoken Language Technologies for Under-resourced languages (SLTU '14, `http://www.mica.edu.vn/sltu2014`). That paper, which is currently under review, concerns related research not reported here, and makes no mention of the lex4all application.

[2] [Link removed to preserve submission anonymity]

guages/dialects (Microsoft, 2012) – this makes it very easy for small-scale software developers (i.e. individuals or small organizations without much funding) to create speech-driven applications.

While many such individuals or organizations in the developing world may be interested in using such platforms to build applications for use in their communities, the LRLs typically spoken in these areas are obviously not supported by such commercial systems. And though tools for quickly training or adapting recognizers for new languages exist (e.g. CMUSphinx[3] or the Rapid Language Adaptation Toolkit[4]), these typically require hours of training audio to produce effective models, and even the highest-level tools still require a nontrivial amount of expertise with speech technologies. This data and expertise may not be available to the small-scale developers in question.

However, many useful development-oriented applications (e.g. for accessing information or conducting basic transactions) require only small-vocabulary recognition tasks, by which we mean tasks requiring recognition of a few dozen terms (words or short phrases). For such tasks, an engine designed to recognize speech in a HRL can be used as-is to perform recognition of the LRL terms, given an application-specific grammar describing the allowable combinations and sequences of terms to be recognized, and a pronunciation lexicon which maps each target term to a sequence of phonemes in the HRL for which the recognizer has been trained.

This is the thinking behind Speech-based Automated Learning of Accent and Articulation Mapping, or "Salaam" (Sherwani, 2009; Qiao et al., 2010; Chan and Rosenfeld, 2012), a method of cross-language phoneme-mapping that enables the automatic discovery of source-language pronunciations (phoneme sequences) for words or phrases in the (unrelated) target language.

The basic idea of phoneme-mapping is to discover the best pronunciation sequence for a given term in the target language by using the source language recognizer to perform phone decoding on one or more audio samples of the target term. However, the APIs for commercial recognizers such as Microsoft's are designed for word-decoding, and do not usually enable the use of the phone-decoding mode. The insight of the Salaam approach is to use a specially designed grammar to mimic phone decoding and guide pronunciation discovery (Qiao et al., 2010, §4.1; Chan and Rosenfeld, 2012, §3.2). This so-called "super-wildcard" grammar allows the recognizer to treat each sample of the target term as a "phrase" made up of 0-10 "words", where each "word" can be matched to any possible sequence of 1, 2, or 3 source language phonemes (Qiao et al., 2010, §4.1).

Given this super-wildcard grammar and one or more audio recordings of the target term, Qiao et al. (2010, §4.1) use an iterative training algorithm to discover the best pronunciation(s) for that term, one phoneme at a time. Essentially, the recognizer makes one pass to find the best match(es) for the first phoneme, then a second pass to find the first two phonemes, and so on until a stopping criterion is met, e.g. the recognition confidence score assigned to the resulting "phrase" stops improving (Qiao et al., 2010, p. 4).

Compared to pronunciations hand-written by a linguist, pronunciations generated automatically by this algorithm yield substantially higher recognition accuracy (Qiao et al., 2010, §5.2). Qiao et al. further improve accuracy by training on samples from multiple speakers, and by creating a lexicon with multiple pronunciations for each word (i.e. the n-best results of the training algorithm instead of the single best result). Chan and Rosenfeld (2012) achieve still higher accuracy by applying an iterative discriminative training algorithm, identifying and removing pronunciations that cause confusion between word types.

In sum, the Salaam method is fully automatic, requiring expertise neither in speech technology nor in linguistics, and for each new target language it requires only a few recorded utterances of each word from one or more speakers, an amount that can be collected in a short time with little effort or expense. Despite these humble requirements, it enables the construction of pronunciation lexicons that can help bring speech recognition applications to LRLs. This has been demonstrated in at least two projects that have successfully used the Salaam method to add voice interfaces to real applications: an Urdu telephone-based health information system in Pakistan (Sherwani, 2009), and a Hindi text-free smartphone application to deliver agricultural information to farmers in India (Bali et al., 2013).

[3] http://www.cmusphinx.org

[4] http://i19pc5.ira.uka.de/rlat-dev

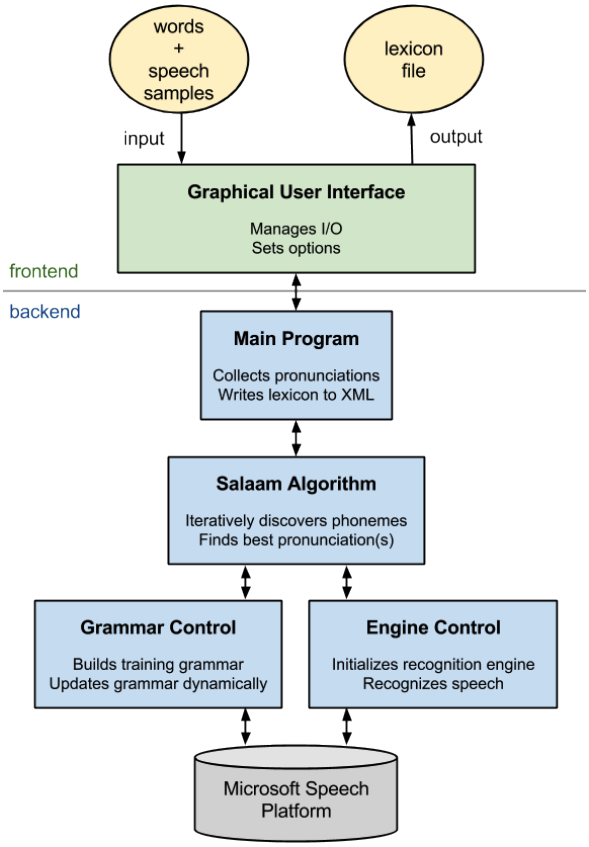

Figure 1: Overview of the core components of the lex4all lexicon-building application.

Given the established success of the Salaam method for pronunciation discovery, our contribution is to build an easy-to-use graphical application around these algorithms, so that non-expert users can quickly and easily create the pronunciation lexicons necessary for developing speech interfaces in LRLs using existing HRL recognizers. In the following sections, we describe the lex4all application and the implementation of the Salaam method which is at its core.

3 System overview

lex4all is a free and open-source desktop application for the Microsoft Windows operating system.[5] The application and its source code are available for download via GitHub.[6]

As stated above, the primary functionality of lex4all is the fast and easy construction of pronunciation lexicons; Figure 1 illustrates the architecture of the lexicon-building system.

A simple graphical user interface (GUI) allows the user to easily specify one written form (text string) and and one or more audio samples (`.wav` files) for each term in the target vocabulary. Given this input, the program uses the Salaam phoneme-mapping algorithm (Qiao et al., 2010; Chan and Rosenfeld, 2012) to find the best pronunciation(s) for each word in the LRL vocabulary. This algorithm requires a pre-trained recognition engine for a HRL (we use American English) as well as a series of dynamically-created recognition grammars. The engine and grammars are constructed and managed using the Microsoft Speech Platform speech recognition libraries (Microsoft, 2012), which are therefore crucial prerequisites for the lex4all application. It should be noted here that in order to make the system more time-efficient, our implementation of Salaam deviates somewhat from the algorithm described by Qiao et al. (2010); this is discussed further in §4.

Once pronunciations for all terms in the vocabulary have been generated, the application outputs a pronunciation lexicon for the given terms as an XML file conforming to the standard pronunciation lexicon specification (Baggia, 2008), an example of which is given in Listing 1. This lexicon can then be directly included in a speech recognition application built using the Microsoft Speech Platform API or a similar toolkit.

```
<?xml version="1.0" encoding="utf-8"?>

<lexicon version="1.0" xmlns="http://www.w3.org
     /2005/01/pronunciation-lexicon" xml:lang="en-US
     " alphabet="x-microsoft-ups">

  <lexeme>
    <grapheme>beeni</grapheme>
    <phoneme>B E NG I</phoneme>
    <phoneme>B EI N I I</phoneme>
  </lexeme>

</lexicon>
```

Listing 1: Sample lexicon mapping the Yoruba word *beeni* (“yes”) to two possible sequences of American English phonemes.

4 Pronunciation mapping

4.1 Recognition engine

As described above, lex4all uses a recognition engine trained for a HRL to map audio in the target LRL to pronunciations (phoneme sequences) in the source HRL. In the current implementation, we use the English (US) speech recognition engine developed by Microsoft for server-side recog-

[5] Windows 7 or 8 (64-bit).

[6] [Link removed to preserve submission anonymity]

nition of telephone-quality audio, accessed via the Microsoft Speech Platform SDK 11 (Microsoft, 2012). In keeping with the overall objective of the Salaam approach, the engine is used as-is, with no modifications to its underlying models. We choose the Microsoft Speech Platform for its robustness and ease of use, as well as to maintain comparability with the work of Qiao et al. (2010) and Chan and Rosenfeld (2012), who also used Microsoft's server-side American English recognizer. We use an engine designed to recognize low-quality audio since we aim to enable the creation of useful applications for LRLs, including those spoken in developing-world communities, and such applications will most likely need to cope with low-quality audio, e.g. for telephone-based deployment (see e.g. Sherwani and Rosenfeld (2008)).

4.2 Implementation of the Salaam method

Pronunciations (sequences of source-language phonemes) for each term in the target vocabulary are generated using the iterative Salaam algorithm (Qiao et al., 2010, §4.1), described in §2 above. In our implementation, we stop iterations if the top-scoring sequence for a given word has not changed for three consecutive iterations, or – following Qiao et al. (2010, p. 4) – if the best sequence from the i^{th} pass has a lower score than the best sequence of the $i-1^{th}$ pass (with the $i-1^{th}$ results returned as the best pronunciations). In both cases, at least three passes are required.

After the iterative training has completed, the n-best pronunciation sequences (with n specified by the user – see §5.2) for each term are written to the lexicon, each in a `phoneme` element corresponding to the `grapheme` element containing the term's orthographic form (e.g. Listing 1).

4.3 Running time

A major challenge we faced in engineering a user-friendly application based on the Salaam pronunciation-mapping algorithm (Qiao et al., 2010) was the long running time of the algorithm. As stated in §2, the algorithm depends on a "super-wildcard" grammar that allows the recognizer to match each sample of a given term to a "phrase" of 0-10 "words", each word comprising any possible sequence of 1, 2, or 3 source-language phonemes. Given the 40 phonemes of American English, this results in over 65,000 possibilities for each word, which results in a huge training grammar and thus a long processing time. For a vocabulary of 25 terms with 5 training samples per term, we found the process to take approximately 1-2 hours. This is well beyond the amount of time that a developer will wish to spend creating just one component (the lexicon) of their speech-driven application, especially for a meager 25-term lexicon.

Therefore, we limit the length of each "word" in the grammar to only one phoneme, instead of up to 3, giving e.g. 40 possibilities instead of tens of thousands. Despite the shorter word sequences, this implementation of the algorithm can nonetheless still discover pronunciation sequences of an arbitrary length, since, in each iteration, the phonemes discovered so far are prepended to the super-wildcard grammar, such that phoneme sequence of the first "word" in the phrase grows longer with each pass (Qiao et al., 2010, p. 4). However, the new implementation is an order of magnitude faster: constructing the same 25-term lexicon on the same hardware takes only approximately 2-5 *minutes*.

To ensure that the new implementation's vastly improved running time does not come at the cost of reduced recognition accuracy, we evaluate and compare word recognition accuracy rates using lexicons built with the old and new implementations. The data we use for this evaluation is a subset of the Yoruba data collected by Qiao et al. (2010, §5.1), comprising 25 Yoruba terms (words) uttered by 2 speakers (1 male, 1 female), with 5 samples of each term per speaker. To determine same-speaker accuracy for each of the two speakers, we perform a leave-one-out evaluation on the five samples recorded per term per speaker. Cross-speaker accuracy is evaluated by training the system on all five samples of each term recorded by one speaker, and testing on all five samples from the other speaker. Using R[7] we perform a paired two-tailed t-test on the results to assess the significance of the differences in accuracy.

The results of our evaluation, given in Table 1, indicate no statistically significant difference in accuracy between the two conditions (all p-values are well above any reasonable significance threshold). As our new, modified implementation of the Salaam algorithm is much faster than the original, yet equally accurate, lex4all uses the new implementation by default, although for research purposes we leave users the option of using the original (slower) implementation (see §5.2).

[7] `http://www.r-project.org/`

	Old	New	p-value
Same-speaker results			
Female average	72.8	**73.6**	0.75
Male average	**90.4**	**90.4**	1.00
Overall average	81.6	**82**	0.81
Cross-speaker results			
Train male, test female	**70.4**	66.4	
Train female, test male	76.8	**77.6**	
Average	**73.6**	72	0.63

Table 1: Difference in word recognition accuracy in Yoruba using old (slower) and new (faster) implementations, with p-values from t-tests.

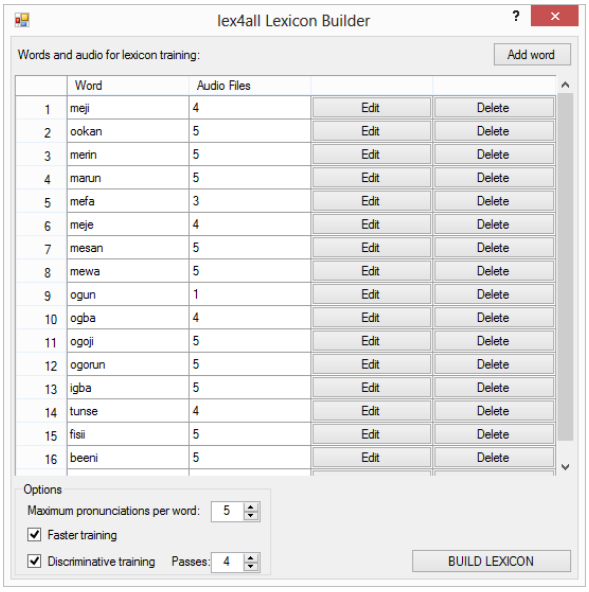

Figure 2: Screenshot of the lexicon builder.

4.4 Discriminative training

We implement the iterative discriminative training algorithm of Chan and Rosenfeld (2012) and offer it as an optional step in the lexicon-building process. In each iteration, this algorithm takes as input the set of mapped pronunciations generated using the Salaam algorithm (Qiao et al., 2010), simulates recognition of the training audio samples using these pronunciations, and outputs a ranked list of the pronunciations in the lexicon that best match each sample according to the recognizer. Pronunciations that cause "confusion" between words in the vocabulary, i.e. pronunciations that the recognizer matches to samples of the wrong word type, are thus identified and removed from the lexicon, and the process is repeated in the next iteration. Chan and Rosenfeld (2012, p. 5) obtain a significant increases in accuracy with up to 8 iterations of the algorithm. In lex4all, discriminative training is applied by default, though users may change the number of passes (4 by default) or disable discriminative training entirely, as mentioned in §5.2.

5 User interface

As mentioned above, we aim to make the creation and evaluation of lexicons simple, fast, and above all accessible to users with no expertise in speech or language technologies. Therefore, the application makes use of a simple graphical user interface that allows users to quickly and easily specify input and output file paths, and to control the parameters of the lexicon-building algorithms.

Figure 2 shows the main interface of the lex4all lexicon builder. This window displays the terms the user has specified and the number of audio samples that the user has selected for each word. Another form, accessed via the "Add word" or "Edit" buttons, allows users to add to or edit the vocabulary by simply typing in the desired orthographic form of the word and selecting the audio sample(s) to be used for pronunciation discovery (see §5.1 for more details on audio input).

Once the target vocabulary and training audio have been specified, and the additional options have been set if desired, the user clicks the "Build Lexicon" button and specifies the desired name and target directory of the lexicon file to be saved. In case of errors (e.g. there are terms in the vocabulary for which no audio files have been specified), a message pops up instructing the user on how to correct the error. Once any errors have been resolved, pronunciation discovery begins, and a progress bar appears above the "Build Lexicon" button to keep the user informed of the program's status. When all pronunciations have been generated, a success message displaying the elapsed training time is displayed, and the user may either proceed to the evaluation module to assess the newly created lexicon (see §6), or may return to the main interface to build another lexicon.

5.1 Audio input and recording

The GUI allows users to easily browse their file system for pre-recorded audio samples (`.wav` files) to be used for lexicon training. To simplify data collection and enable the development of ap-

plications even for zero-resource languages, users also have the option of using the simple built-in audio recorder to record new speech samples.

As the backend for the recording feature we use the open-source library NAudio.[8] The recorder takes the user's default audio input device as its data source and records one channel with a sampling rate of 8 kHz. We use this low sampling rate because the recognition engine we employ is for low-quality audio input (see §4.1).

5.2 Additional options

The lower left corner of the screenshot in Figure 2 shows optional controls for quick and easy fine-tuning of the lexicon-building process (the default settings are pictured).

First of all, users can specify the maximum number of pronunciations (`phoneme` elements) that each word in the lexicon may contain. According to Qiao et al. (2010, §5.2.3) and Chan and Rosenfeld (2012, §4.2.1), more pronunciations per word in the lexicon may improve recognition accuracy. Secondly, users may train the lexicon using our modified, much faster implementation of the Salaam algorithm or the original implementation following Qiao et al. (2010), as explained in §4. Finally, as mentioned in §4.4, users may choose whether or not discriminative training is applied, and if so, how many passes are run.

6 Evaluation module for research

In addition to its primary utility as a lexicon-building tool, lex4all is also a valuable research aide thanks to an evaluation module that allows users to quickly and easily evaluate the lexicons they have created. The evaluation tool allows users to browse their file system for an XML lexicon file that they wish to evaluate; this may be a lexicon created using lex4all, or any other lexicon conforming to same format (Baggia, 2008). As in the main interface, users then select one or more audio samples (`.wav` files) for each term they wish to evaluate. The system then attempts to recognize each sample using the given lexicon, and reports the counts and percentages of correct, incorrect, and failed recognitions. Users may optionally save this report, along with a confusion matrix of word types, as a comma-separated values (`.csv`) file.

This evaluation tool thus allows users to quickly and easily assess different configurations of the lexicon-building tool, by simply changing the settings using the GUI (see §5.2) and evaluating the resulting lexicons. Furthermore, as the application's source code is freely available and modifiable, researchers may even replace entire modules of the system (e.g. use a recognition engine for a different source language, or a different pronunciation-discovery algorithm), and use this evaluation module to quickly assess the results. This tool therefore greatly facilitates further research into language-independent small-vocabulary speech recognition.

7 Conclusion and future work

We have presented lex4all, an open-source Windows desktop application that enables the rapid automatic creation of pronunciation lexicons in any (low-resource) language, using an out-of-the-box commercial recognizer (Microsoft, 2012) for a high-resource language (English) and the Salaam method for cross-language pronunciation mapping (Qiao et al., 2010; Chan and Rosenfeld, 2012). The application thus makes small-vocabulary speech recognition interfaces feasible in any language, since the algorithm requires only minutes of training audio; combined with the built-in audio recorder, lexicons can be constructed even for zero-resource languages. We hope that this tool will help developers create speech-driven applications in LRLs, as well as facilitate research in language-independent small-vocabulary recognition.

In future work, we plan to expand the selection of source-language recognizers; at the moment, lex4all only offers American English as the source languge, but any of the 20+ other HRLs supported by the Microsoft Speech Platform could theoretically be used, and it would be interesting to investigate different pairings of source and target languages. Another future goal is to improve and extend the functionality of the audio-recording tool (see §5.1), to make it more flexible and user-friendly. Finally, as a complement to the application, it would be beneficial to create a central online data repository where users can upload the lexicons they have built and the speech samples they have recorded Over time, this could become a valuable collection of data for LRLs, enabling developers and researchers to share and reuse data among languages or language families.

[8] `http://naudio.codeplex.com/`

References

Paolo Baggia. 2008. Pronunciation lexicon specification (PLS) version 1.0. W3C recommendation, W3C, October. http://www.w3.org/TR/2008/REC-pronunciation-lexicon-20081014/.

Kalika Bali, Sunayana Sitaram, Sebastien Cuendet, and Indrani Medhi. 2013. A Hindi speech recognizer for an agricultural video search application. In *Proceedings of the 3rd ACM Symposium on Computing for Development*, ACM DEV '13, pages 5:1–5:8, New York, NY, USA. ACM.

Hao Yee Chan and Roni Rosenfeld. 2012. Discriminative pronunciation learning for speech recognition for resource scarce languages. In *Proceedings of the 2nd ACM Symposium on Computing for Development*, ACM DEV '12, pages 12:1–12:6, New York, NY, USA. ACM.

Microsoft, 2012. *Microsoft Speech Platform SDK 11 Documentation*. http://msdn.microsoft.com/en-us/library/dd266409.

Fang Qiao, Jahanzeb Sherwani, and Roni Rosenfeld. 2010. Small-vocabulary speech recognition for resource-scarce languages. In *Proceedings of the First ACM Symposium on Computing for Development*, ACM DEV '10, pages 3:1–3:8, New York, NY, USA. ACM.

Jahanzeb Sherwani and Roni Rosenfeld. 2008. The case for speech technology for developing regions. In *Proc. HCI for Community and International Development, Florence, Italy*. ACM.

Jahanzeb Sherwani. 2009. *Speech interfaces for information access by low literate users*. Ph.D. thesis, Carnegie Mellon University, Pittsburgh, PA, USA.