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 lexicons enabling the recognition of small vocabularies (up to 100 words, roughly) in the target language using an existing recognition engine designed for a high-resource source language (e.g. English). To build such lexicons, we employ an existing technique for cross-language phoneme-mapping; we describe this method and how it is implemented in *lex4all*. The application also offers a built-in audio recorder that facilitates data collection, a significantly faster implementation of the phoneme-mapping algorithm, and an evaluation module that expedites research on small-vocabulary speech recognition for low-resource languages.

1 Introduction[1]

In recent years it has been demonstrated that speech recognition interfaces can be extremely beneficial for applications in the developing world (Sherwani and Rosenfeld, 2008; Sherwani, 2009; Bali et al., 2013). Typically, such applications target low-resource languages (LRLs) for which large collections of speech data are unavailable, preventing the training or adaptation of recognition engines for these languages. However, 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, provided a pronunciation lexicon mapping each term in the target vocabulary to a sequence of phonemes in the HRL, i.e. phonemes which the recognizer can model.

This is the motivation behind *lex4all*,[2] an open-source application that allows users to automatically create a mapped pronunciation lexicon for terms in any language, using a small number of speech recordings and an out-of-the-box recognition engine for a HRL. 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 for the rapid creation and evaluation of pronunciation lexicons enabling small-vocabulary speech recognition in any language.

2 Background and related work

Several commercial speech recognition systems offer high-level Application Programming Interfaces (APIs) that make it extremely simple to add voice interfaces to an application, requiring 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, supports over 20 languages (Microsoft, 2012) – this makes it very easy to create speech-driven applications.

If, however, the target language is one of the many thousands of LRLs for which high-quality

[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] http://lex4all.github.io/lex4all/

```
<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>
```

Figure 1: Sample XML lexicon mapping the Yoruba word *beeni* ("yes") to two possible sequences of American English phonemes.

recognition engines have not yet been developed, alternative strategies for developing speech-recognition interfaces must be employed. 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, data which is by definition not available for a LRL. Furthermore, non-expert users may struggle to effectively utilize such tools, which still demand some understanding of speech/language technologies.

However, many useful applications (e.g. for accessing information or conducting basic transactions) only require small-vocabulary recognition, i.e. discrimination between a few dozen terms (words or short phrases). For example, VideoKheti (Bali et al., 2013), a smartphone application that delivers agricultural information to farmers in India using a Hindi speech interface, uses a vocabulary of 79 terms. For such small-vocabulary applications, an engine designed to recognize speech in a HRL can be used as-is to perform recognition of the LRL terms, given a grammar describing the allowable combinations/sequences of terms to be recognized, and a pronunciation lexicon mapping each target term to a sequence of phonemes in the HRL (see Figure 1 for an example).

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 discovers accurate source-language pronunciations for terms in the target language. The basic idea is to discover the best pronunciation (phoneme sequence) for a target term by using the source-language recognition engine to perform phone decoding on one or more utterances of the term. As commercial recognizers such as Microsoft's are designed for word-decoding, and their APIs do not usually allow users access to the phone-decoding mode, the Salaam approach uses a specially designed "super-wildcard" recognition grammar to mimic phone decoding and guide pronunciation discovery (Qiao et al., 2010; Chan and Rosenfeld, 2012). This grammar allows the engine to recognize 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. This allows the recognizer to identify the phoneme sequence best matching a given term, without any prior indication of how many phonemes that sequence should contain.

Given this grammar and one or more audio recordings of the target term, Qiao et al. (2010) 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 candidate(s) 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 stops improving, indicating that the sequence is complete.

Compared to pronunciations hand-written by a linguist, pronunciations generated automatically by this algorithm yield substantially higher recognition accuracy: Qiao et al. (2010) report word recognition accuracy rates in the range of 75-95% for vocabularies of 50 terms. Chan and Rosenfeld (2012) improve accuracy on larger vocabularies (up to approximately 88% for 100 terms) by applying an iterative discriminative training algorithm, identifying and removing pronunciations that cause confusion between word types.

The Salaam method is fully automatic, demanding expertise neither in speech technology nor in linguistics, and requires only a few recorded utterances of each word, an amount of data that can be collected quickly with little effort or expense. Despite these humble requirements, it can help create speech-driven applications in LRLs, as has been demonstrated in at least two projects that have successfully used the Salaam method to

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

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

add voice interfaces to real applications: an Urdu telephone-based health information system (Sherwani, 2009), and the VideoKheti application mentioned above (Bali et al., 2013). What has not existed before now is an interface to make this approach accessible to any user.

Given the established success of the Salaam method our contribution is to create a more time-efficient implementation of the pronunciation-discovery algorithm and integrate it into an easy-to-use graphical application, so that non-expert users can quickly and easily create lexicons to develop speech interfaces in LRLs using existing HRL recognizers. In the following sections, we describe the *lex4all* application and our slightly-modified implementation of the Salaam method.

3 System overview

We have developed *lex4all* as a desktop application for Microsoft Windows,[5] since it relies on the Microsoft Speech Platform (Microsoft, 2012) as explained below. The application and its source code are freely available via GitHub.[6]

The application's core feature is its lexicon-building tool, the architecture of which is illustrated in Figure 2. A simple graphical user interface (GUI) allows the user to type in the written form of each term in the target vocabulary, and select one or more audio recordings (`.wav` files) of that term. Given this input, the program uses the Salaam algorithm (Qiao et al., 2010; Chan and Rosenfeld, 2012) to find the best pronunciation(s) for each term. This algorithm requires a pre-trained recognition engine for a HRL 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). It should be noted here that our implementation of Salaam deviates slightly from the algorithm described by Qiao et al. (2010); this is explained in Section 4 below.

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 Pronunciation Lexicon Specification[7] (see Figure 1). This lexicon can then be directly included in a speech recognition application built using the Microsoft Speech Platform API or a similar toolkit.

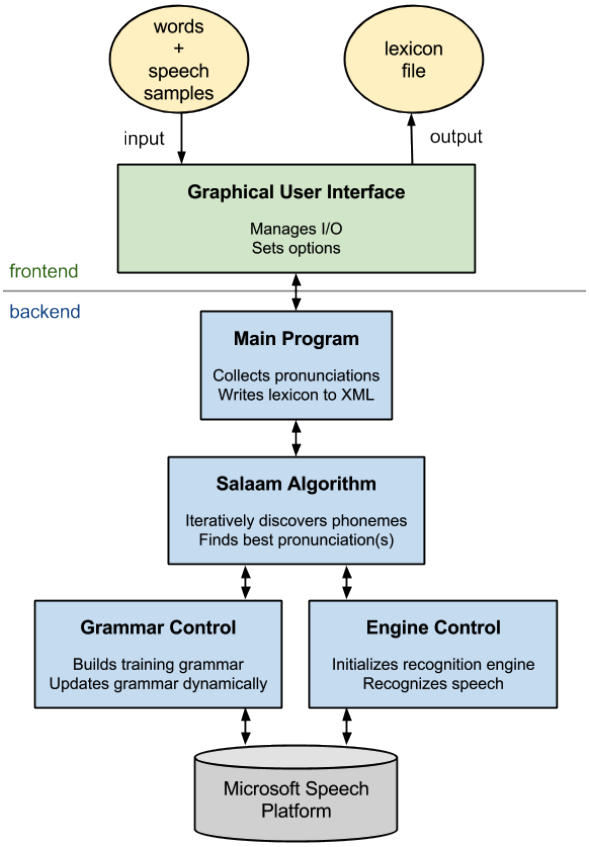

Figure 2: Overview of the core components of the *lex4all* lexicon-building application.

4 Pronunciation mapping

4.1 Recognition engine

For the HRL recognizer we use the American English recognition engine of the Microsoft Speech Platform (Microsoft, 2012). 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. Following these authors, we use an engine designed for server-side recognition of 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 telephone-quality audio or similar (see e.g. Sherwani and Rosenfeld (2008)).

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

[6] [GitHub repository link removed to preserve anonymity]

[7] http://www.w3.org/TR/pronunciation-lexicon/

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 described in Section 2. In our implementation, we stop iterations if the top-scoring sequence for a given word has not changed for three consecutive iterations (Chan and Rosenfeld, 2012), or 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) (Qiao et al., 2010). 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 Section 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. Figure 1).

4.3 Running time

A major challenge we faced in engineering a user-friendly application based on the Salaam algorithm was its long running time. As stated in Section 2, this 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 gives over 65,000 possibilities for each word, resulting in a huge training grammar and thus a long processing time. For a 25-term vocabulary with 5 training samples per term, we found the process to take approximately 1-2 hours on a standard modern laptop. For development and research, this long training time is a serious disadvantage.

To speed up training, 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. The algorithm can 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). However, the new implementation is an order of magnitude faster: constructing the same 25-term lexicon on the same hardware takes approximately 2-5 *minutes*, i.e. less than 10% of the previous training time.

		Old	New	p
Same-speaker	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	Trained on male	**70.4**	66.4	–
	Trained on female	76.8	**77.6**	–
	Average	**73.6**	72	0.63

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

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), 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. 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 implementations (all p-values are above 0.5 and thus clearly insignificant). 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 Section 5.2).

4.4 Discriminative training

As explained in Section 2, Chan and Rosenfeld (2012) achieve increased accuracy (gains of up to 5 percentage points) by applying an iterative discriminative training algorithm. This algorithm takes as input the set of mapped pronunciations generated using the Salaam algorithm (Qiao et al.,

2010); in each iteration, it 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.

We implement this accuracy-boosting algorithm in *lex4all*, and apply it by default. To enable fine-tuning and experimentation, we leave users the option to change the number of passes (4 by default) or disable discriminative training entirely, as mentioned in Section 5.2.

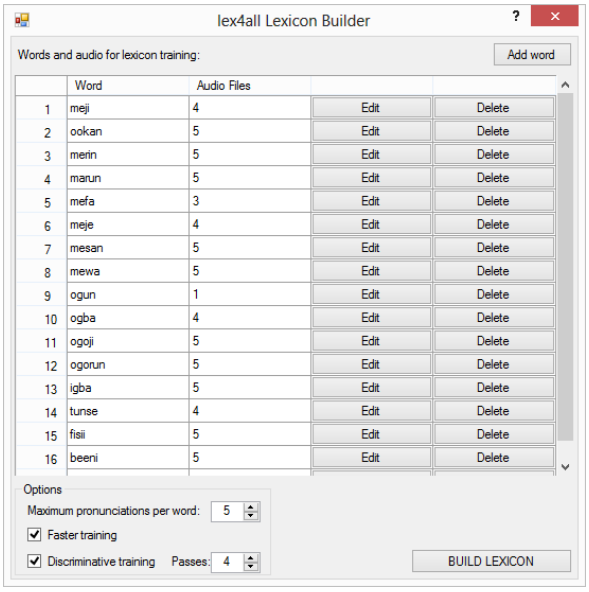

Figure 3: Screenshot of the lexicon builder.

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 3 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 Section 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, and pronunciation discovery begins. 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 Section 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 lexicons even for zero-resource languages, *lex4all* also offers a simple built-in audio recorder to record new speech samples.

The recorder, built using the open-source library NAudio,[8] takes the user's default audio input device as its data source and records one channel with a sampling rate of 8 kHz, since the recognition engine we employ is designed for low-quality audio (see Section 4.1).

5.2 Additional options

The lower left corner of the screenshot in Figure 3 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) per word that the lexicon may contain. According to Qiao et al. (2010) and Chan and Rosenfeld (2012), 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 Section 4.

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

Finally, as mentioned in Section 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 in the same format. 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.

The evaluation module thus allows users to quickly and easily assess different configurations of the lexicon-building tool, by simply changing the settings using the GUI (see Section 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 different recognition engine or pronunciation-discovery algorithm), and use the evaluation module to quickly assess the results. Therefore, *lex4all* facilitates not only application development but also further research into small-vocabulary speech recognition using mapped pronunciation lexicons.

7 Conclusion and future work

We have presented *lex4all*, an open-source 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; given the built-in audio recorder, lexicons can be constructed even for zero-resource languages. Furthermore, *lex4all*'s flexible and open design and easy-to-use evaluation module make it a valuable tool for 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 language, 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 Section 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 re-use data among languages or language families.

References

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.