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 or where PCs and internet connections are not always available (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, simple 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 as simple as specifying (in text) the words/phrases that should be recognized; this requires 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 –

[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]

the Microsoft Speech Platform, for example, currently supports recognition and synthesis for 26 languages/dialects (Microsoft, 2012) – this makes it very easy for small-scale software developers (i.e. individuals or small organizations without much funding) to create new speech-driven applications.

While many such individuals or organizations in the developing world may be interested in using such platforms to create speech-driven applications for use in their communities, the low-resource languages typically spoken in these areas are obviously not supported by such commercial systems. And though many effective techniques for 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 for building new models still require a nontrivial amount of expertise with speech technologies; such 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 those requiring discrimination between a few dozen terms. For such tasks, an unmodified HRL recognizer can be used as-is to perform recognition of the LRL terms; as Figure 1 illustrates, we simply need an application-specific grammar describing the allowable combinations and sequences of words/phrases to be recognized, and a pronunciation lexicon which maps each of the target words/phrases to a sequence of phonemes in the source language 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). This method of cross-language phoneme-mapping enables the automatic discovery of source-language pronunciation sequences for words or phrases in the (unrelated) target language, and thus constitutes the foundation on which the lex4all tool is built.

The basic idea of phoneme-mapping is to discover the best pronunciation sequence for a given

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

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

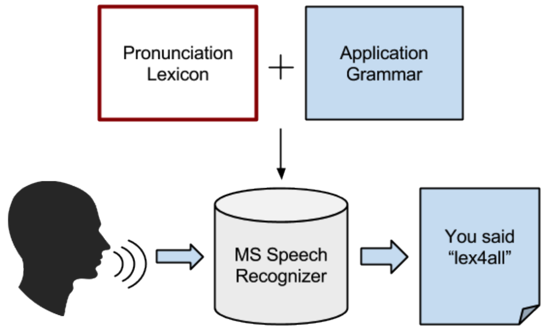

Figure 1: Given a pronunciation lexicon and application-specific grammar, an existing recognizer for a HRL can be used to recognize speech in a LRL.

word in the target language by using the source language recognizer to perform phone decoding on one or more audio samples of the target word. 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 this phone decoding (Chan and Rosenfeld, 2012, §3.2). Specifically, Qiao et al. (Qiao et al., 2010, 4.1) create a recognition grammar representing a phoneme "super-wildcard" that guides pronunciation discovery. This grammar allows the recognizer to treat an audio sample of the target word 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 word, Qiao et al. (Qiao et al., 2010, 4.1) use an iterative training algorithm to discover the best pronunciation(s) for that word, one phoneme at a time. In the first pass, the recognizer finds the best match(es) for the first phoneme, then for the first two phonemes in the second pass, 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 expert-crafted pronunciations, using pronunciations generated automatically by this algorithm improves recognition accuracy substantially (Qiao et al., 2010, 5.2). By training on samples from two speakers instead of one, and by using a pronunciation lexicon containing multiple pronunciations for each word (i.e. the n-best results of the training algorithm instead of the single best result), Qiao et al. are able to further improve accuracy. Chan and Rosenfeld (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 (to modify acoustic models) nor in linguistics (to manually generate seed pronunciations), and for each new target language it requires only a few minutes' worth of training data from one or more speakers, an amount that can be collected in a short time with little effort or expense. At the same time, it enables the construction of pronunciation lexicons that can help bring speech recognition applications to LRLs. This has been demonstrated in at least two developing-world 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 text-free Hindi smartphone application to deliver agricultural information to farmers in rural India (Bali et al., 2013).

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 modified implementation of the Salaam method which is at its core.

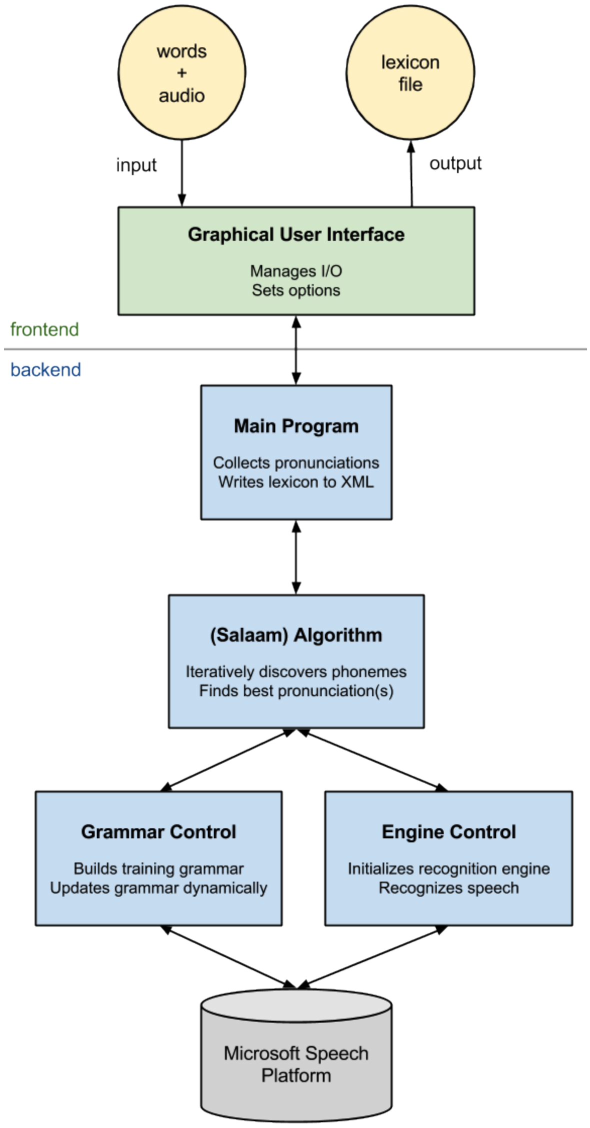

Figure 2: Overview of the core components of the lex4all lexicon-building application.

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 2 illustrates the architecture of the lexicon-building system, explained below. In addition to this core system, lex4all also features an evaluation module that simulates recognition using a given lexicon and can be used as a research tool; this module is explained in detail in Section 6.

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, and to set other options (e.g. number of pronunciations per word, name/save location of lexicon file, etc.).

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 (e.g. American English) as well as a series of dynamically-created recogni-

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

[6][Link removed to preserve submission anonymity]

```xml
<?xml version="1.0" encoding="utf-8"
    standalone="no"?>

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

  <lexeme>
    <grapheme>mefa</grapheme>
    <phoneme>M EH F AX</phoneme>
  </lexeme>

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

</lexicon>
```

Listing 1: TODO: MORE PRONUNCIATIONS Sample lexicon file mapping Yoruba words to American English pronunciations.

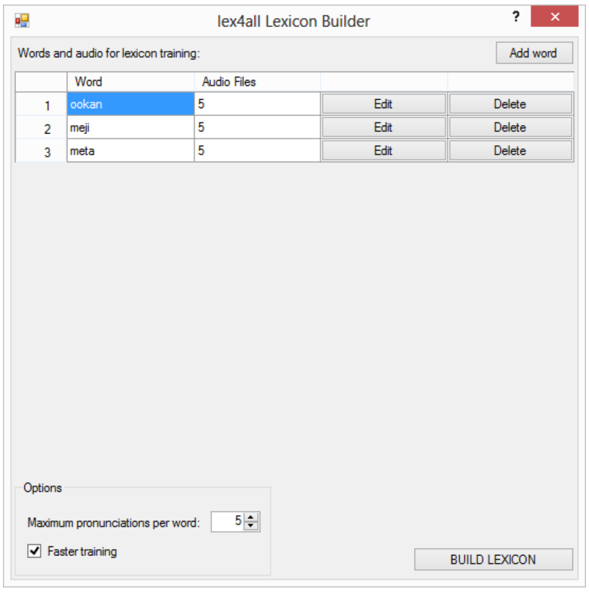

Figure 3: TODO: UPDATE SCREENSHOT Screenshot of the lexicon builder interface.

tion 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 Section 5 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 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.

4 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 graphically selecting the audio sample(s) to be used for pronunciation discovery (see Section 4.1 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), an informative message pops up to instruct the user how to correct the error. Once any errors have been resolved, the pronunciation discovery process begins to run, 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 Section 6), or may return to the main interface to build another lexicon.

4.1 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. However, as lex4all has been designed as a language-independent tool, it should enable the development of applications even in zero-resource languages for which no recorded audio is yet available; to this end, the application includes a built-in audio recorder, to simplify the process of collecting audio samples from native speakers.

As the backend for the recording feature we use the open-source library NAudio.[7] 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. This low sampling rate was chosen for two reasons: (a) the Microsoft Speech Platform recognition engine we use is designed for server-side recognition of low-quality audio, and (b) the ultimate goal of lex4all is to enable the creation of useful applications for low-resource languages, 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).

A simple GUI form allows users to record audio samples for a given word that has been or is being added to the lexicon vocabulary. Recorded files can be used immediately for lexicon training and/or saved for future use, and can be combined with pre-recorded audio files for training.

4.2 Additional options

The lower left corner of the screenshot in Figure 3 shows the parameters and additional options which users can easily modify, if they so choose:

- Users can specify 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), a higher number of pronunciations per word in the lexicon may improve the resulting recognition accuracy.

- Users have the choice of training the lexicon using a modified, much faster implementation of the Salaam algorithm instead of an implementation more closely following Qiao et al. (2010). See Section 5 for a detailed explanation of the alternative implementations.

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

- Users may choose whether or not the discriminative training algorithm of Chan and Rosenfeld (2012) is applied, and if so, how many passes are run. See Section 5.4 for more information.

We make these options easily modifiable via the graphical interface for users who wish to fine-tune the lexicon-building process, possibly to experiment with different settings to achieve the highest possible recognition accuracy. In conjunction with the application's evaluation module (see Section 6), this expedites further research into language-independent small-vocabulary recognition. However, users who do not wish to change the default settings (shown in Figure 3) may simply ignore these controls.

5 Pronunciation mapping

5.1 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 recognition of telephone-quality audio, accessed via the Microsoft Speech Platform SDK 11 (Microsoft, 2012). This system was chosen for its robustness and 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. In keeping with the overall objective of the Salaam approach, the recognition engine is used as-is, with no modifications to underlying models.

5.2 Implementation of the Salaam method

Pronunciations for each term in the target vocabulary are generated using the iterative Salaam algorithm of Qiao et al. (2010, 4.1), described in Section 2 above. In our implementation, we stop iterations if the top-scoring phoneme sequence for a given word has not changed for three consecutive iterations, or – following Qiao et al. (2010, p. 4) – if the best result from the i^{th} pass has a lower score than the best result 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.

To determine the best pronunciation sequences for a given term from each pass, the results for all training samples of that term are naively combined into a set, and sorted by the confidence score assigned to each pronunciation (phoneme sequence). If a given pronunciation matches more than one sample, the overall score for that sequence in that pass is simply the sum of confidence scores for all samples it was associated with. In future work, it may be worth exploring more intelligent heuristics for ranking the set of sequences (see Section 7).

After the iterative training has completed, the n-best pronunciation sequences (with n specified by the user as described in Section 4.2) for each term are written to the lexicon, each comprising a `phoneme` element corresponding to the single `grapheme` element containing the word's orthographic form (e.g. Listing 1).

5.3 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 described in Section 2, the algorithm described by Qiao et al. (2010) depends on a "super-wildcard" grammar that allows the recognizer to match the training audio sample to a "phrase" of 0-10 "sub-words", and each sub-word to any possible sequence of 1, 2, or 3 source-language phonemes. Given the 40 phonemes of American English, this results in 64,000 possibilities for each sub-word, which results in a huge training grammar and thus a long processing time. For a 25-word vocabulary with 5 training audio samples per word, we found the lexicon-building 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-word lexicon.

Therefore, we limit the length of each sub-word in the grammar to only one phoneme, instead of up to 3, giving e.g. 40 possibilities for each sub-word instead of the previous 64,000. Despite the shorter sub-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, just as before (Qiao et al., 2010, p. 4). However, using this new implementation, constructing the same 25-word vocabulary with the same 5 samples per word takes approximately 2-5 *minutes*. In other words, the new implementation runs some 20+ times faster than the old implementation.

To ensure that the new implementation's vastly improved running time does not come at the cost of reduced recognition accuracy using the resulting lexicon, we evaluate and compare word recognition accuracy rates using lexicons built using 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 words uttered by 2 speakers (1 male, 1 female), with 5 samples of each word 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 word per speaker (this amounts to a five-fold evaluation, reserving one sample per word per speaker for testing, and training on the other four). Cross-speaker accuracy is evaluated by training the system on all five samples of each word recorded by one speaker, and testing on all five samples from the other speaker. We use the R software environment[8] for statistical analysis of the difference in accuracy in each condition.

The results of our evaluation, given in Table 1, indicate no statistically significant difference between the accuracy obtained using the old and new implementations (all p-values are well above any reasonable significance threshold). Therefore, by making this small modification to the wildcard grammar used for pronunciation discovery, we achieve an implementation of the Salaam algorithm that is much faster than the original, yet yields just as high recognition accuracy. The lex4all application therefore uses the new implementation by default, although (as mentioned in Section 4.2) we leave users the option of using the original (slower) implementation, so that they may perform their own assessment.

5.4 Discriminative training

As mentioned above, 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,

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

		Word Recognition Accuracy (%)		Difference	p-value
		Old wildcard	New wildcard		
Same-speaker (leave-one-out)	Female (avg.)	72.8	**73.6**	+0.8	0.75
	Male (avg.)	**90.4**	**90.4**	0.0	1.00
	Overall average	81.6	**82**	+0.4	0.81
Cross-speaker (train → test)	Male → Female	**70.4**	66.4	-4.0	
	Female → male	76.8	**77.6**	+0.8	
	Average	**73.6**	72	-1.6	0.63

Table 1: Word recognition accuracy using American English recognizer on Yoruba audio (25 words, 2 speakers, 5 samples per word per speaker), with results of paired two-tailed t-test for significance.

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) obtain a significant increases in accuracy with up to 8 iterations of the algorithm. lex4all therefore applies this discriminative training (TODO: DEFAULT NUMBER OF PASSES) by default, though users may change the number of passes or disable discriminative training entirely, as mentioned in Section 4.2.

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). The application reads this lexicon and automatically populates a list of its words (`grapheme` elements), displaying them in a window similar to that of Figure 3; users may remove words from the evaluation set, but may not add any words that are not included in the lexicon file. As in the main interface, users then select one or more audio samples (`.wav` files) for each word, which will be used for testing. At the click of a button, the system attempts to recognize each of the given audio samples using the given lexicon, and then reports the number of samples correctly and incorrectly recognized, and the number (if any) that could not be recognized. Users may optionally save this report, along with a confusion matrix of word types, as a comma-separated values (`.csv`) file, which can be easily imported into standard spreadsheet or statistical analysis software for analysis of the results.

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 Section 4.2) 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.

It would be advantageous to display the volume while recording which is at the moment not possible because data is redirected either way. Unfortunately we had issues when doing both simultaneously which resulted in disturbances in the recorded audio.

A major advantage would it be to play back what has been recorded, so the user does not have to save the file first in order to check its quality and content.

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.