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

Anjana Vakil, Max Paulus, Alexis Palmer, and Michaela Regneri
Department of Computational Linguistics, Saarland University
{anjanav,mpaulus,apalmer,regneri}@coli.uni-saarland.de

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 build lexicons enabling the recognition of small vocabularies (up to 100 terms, 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 method for cross-language phoneme-mapping. The application also offers a built-in audio recorder that facilitates data collection, a significantly faster implementation of the phoneme-mapping technique, and an evaluation module that expedites research on small-vocabulary speech recognition for low-resource languages.

1 Introduction

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, given 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*,[1] 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,[2] for example, supports over 20 languages – 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 recognition engines have not yet been developed, alternative strategies for developing speech-recognition interfaces must be employed. Though tools for quickly training recognizers for new languages exist (e.g. CMUSphinx[3]), they typically require many hours of training audio to produce effective models, data which is by definition not

[1] http://lex4all.github.io/lex4all/
[2] http://msdn.microsoft.com/en-us/library/hh361572
[3] http://www.cmusphinx.org

available for LRLs. In efforts to overcome this data scarcity problem, recent years have seen the development of techniques for rapidly adapting multilingual or language-independent acoustic and language models to new languages from relatively small amounts of data (Schultz and Waibel, 2001; Kim and Khudanpur, 2003), methods for building resources such as pronunciation dictionaries from web-crawled data (Schlippe et al., 2014), and even a web-based interface, the Rapid Language Adaptation Toolkit[4] (RLAT), which allows non-expert users to exploit these techniques to create speech recognition and synthesis tools for new languages (Vu et al., 2010). While they greatly reduce the amount of data needed to build new recognizers, these approaches still require non-trivial amounts of speech and text in the target language, which may be an obstacle for very low- or zero-resource languages. Furthermore, even high-level tools such as RLAT still demand some understanding of linguistics/language technology, and thus may not be accessible to all users.

However, many useful applications (e.g. for accessing information or conducting basic transactions by telephone) 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 text-free smartphone application that delivers agricultural information to low-literate farmers in India, recognizes 79 Hindi 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 and sequences of terms to be recognized, and a pronunciation lexicon mapping each target term to at least one pronunciation (sequence of phonemes) in the HRL (see Fig. 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

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

```
<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.

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 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 term, Qiao et al. (2010) use an iterative training algorithm to discover the best pronunciation(s) for that term, one phoneme at a time. 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. At least two projects have successfully used the Salaam method to 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 that makes 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. In the following sections, we describe this 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 (MSP) as explained in Section 4.1. 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 users 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 method to find the best pronunciation(s) for each term. This 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 MSP. We note here that our implementation of Salaam deviates slightly from that of Qiao et al. (2010), improving the time-efficiency and thus usability of the system (see Sec. 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 Pronunciation Lexicon Specification.[7] This lexicon can then be directly included in a speech recognition application built using the MSP API or a similar toolkit.

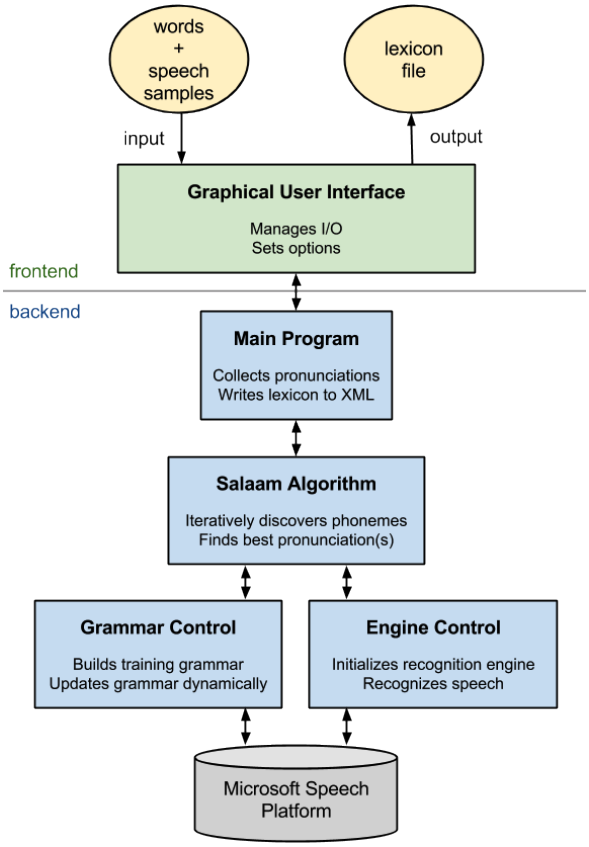

Figure 2: Overview of the core components of the *lex4all* lexicon-building application.

4 Pronunciation mapping

4.1 Recognition engine

For the HRL recognizer we use the US English recognition engine of the MSP. The engine is used as-is, with no modifications to its underlying models. We choose the MSP 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). 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 should be able to cope with telephone-quality audio or similar (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 from the audio sample(s) of that term using the iterative Salaam algorithm (Sec. 2), which employs the source-language recognizer and a special recognition grammar. In the first pass, the algorithm finds the best candidate(s) for the first phoneme of the sample(s), then the first two phonemes in the second pass, and so on until a stopping criterion is met. In our implementation, we stop iterations if the top-scoring sequence for a term has not changed for three consecutive iterations (Chan and Rosenfeld, 2012), or if the best sequence from a given pass has a lower confidence score than the best sequence from the

[5] Windows 7 or 8 (64-bit).
[6] http://github.com/lex4all/lex4all
[7] http://www.w3.org/TR/pronunciation-lexicon/

previous pass (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 users – see Sec. 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 (see Fig. 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. 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 (Qiao et al., 2010). Given the 40 phonemes of US 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, the process takes 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 the 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.

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 paired two-tailed t-tests on the results to assess the significance of the differences in mean accuracy.

		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: Word recognition accuracy for Yoruba using old (slower) and new (faster) implementations, with p-values from t-tests for significance of difference in means. Bold indicates highest 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

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; 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. 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 to disable discriminative training entirely, as mentioned in Section 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 GUI 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 that have been specified and the number of audio samples that have been 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 Sec. 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, users click the "Build Lexicon" button and specify 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 users may either proceed to the evaluation module to assess the newly created lexicon (see Sec. 6), or 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 default audio input device as its source and records one channel with a sampling rate of 8 kHz, as the recognition engine we employ is designed for low-quality audio (see Section 4.1).

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

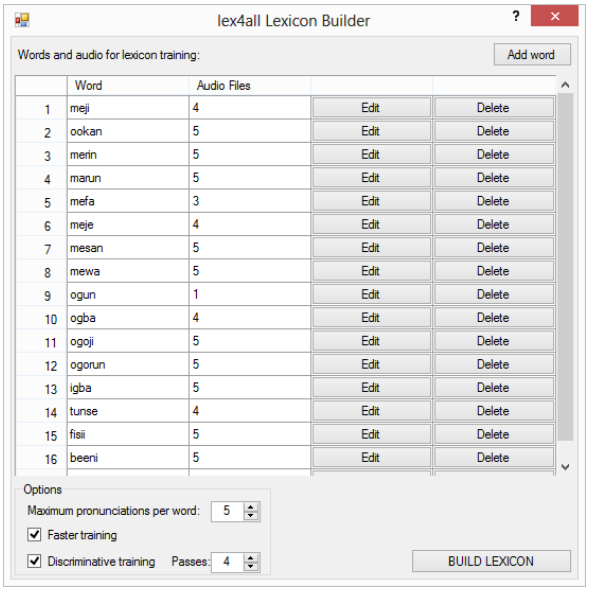

Figure 3: Screenshot of the lexicon builder.

5.2 Additional options

As seen in Figure 3, *lex4all* includes 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; allowing more pronunciations per word may improve recognition accuracy (Qiao et al., 2010; Chan and Rosenfeld, 2012). Secondly, users may train the lexicon using our modified, faster implementation of the Salaam algorithm or the original implementation. Finally, users may choose whether or not discriminative training is applied, and if so, how many passes are run (see Sec. 4.4).

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 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 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 for a high-resource language 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 only minutes of training audio are required; 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 uses US English as the source language, but any of the 20+ other HRLs supported by the MSP could be added. This would enable investigation of the target-language recognition accuracy obtained using different source languages, though our initial exploration of this issue suggests that phonetic similarity between the source and target languages might not significantly affect accuracy (Vakil and Palmer, 2014). Another future goal is to improve and extend the functionality of the audio-recording tool 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 LRL data, enabling developers and researchers to share and re-use data among languages or language families.

Acknowledgements

The first author was partially supported by a Deutschlandstipendium scholarship sponsored by IMC AG. We thank Roni Rosenfeld, Hao Yee Chan, and Mark Qiao for generously sharing their speech data and valuable advice, and Dietrich Klakow, Florian Metze, and the three anonymous reviewers for their feedback.

References

Kalika Bali, Sunayana Sitaram, Sebastien Cuendet, and Indrani Medhi. 2013. A Hindi speech recognizer for an agricultural video search application. In *ACM DEV '13*.

Hao Yee Chan and Roni Rosenfeld. 2012. Discriminative pronunciation learning for speech recognition for resource scarce languages. In *ACM DEV '12*.

Woosung Kim and Sanjeev Khudanpur. 2003. Language model adaptation using cross-lingual information. In *Eurospeech*.

Fang Qiao, Jahanzeb Sherwani, and Roni Rosenfeld. 2010. Small-vocabulary speech recognition for resource-scarce languages. In *ACM DEV '10*.

Tim Schlippe, Sebastian Ochs, and Tanja Schultz. 2014. Web-based tools and methods for rapid pronunciation dictionary creation. *Speech Communication*, 56:101–118.

Tanja Schultz and Alex Waibel. 2001. Language-independent and language-adaptive acoustic modeling for speech recognition. *Speech Communication*, 35(1-2):31 – 51.

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

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

Anjana Vakil and Alexis Palmer. 2014. Cross-language mapping for small-vocabulary ASR in under-resourced languages: Investigating the impact of source language choice. In *SLTU '14*.

Ngoc Thang Vu, Tim Schlippe, Franziska Kraus, and Tanja Schultz. 2010. Rapid bootstrapping of five Eastern European languages using the Rapid Language Adaptation Toolkit. In *Interspeech*.