Bootstrapping Phonetic Lexicons for New Languages

Sameer R. Maskey[1], *Alan W Black*[2,3], *Laura M. Tomokiyo*[3]

Dept. of Computer Science, Columbia University, New York, NY[1]
Language Technologies Institute, Carnegie Mellon University, Pittsburgh, PA[2]
Cepstral LLC, Pittsburgh, PA[3]

smaskey@cs.columbia.edu, awb@cs.cmu.edu, laura@cepstral.com

Abstract

Although phonetic lexicons are critical for many speech applications, the process of building one for a new language can take a significant amount of time and effort. We present a bootstrapping algorithm to build phonetic lexicons for new languages. Our method relies on a large amount of unlabeled text, a small set of 'seed words' with their phonetic transcription, and the proficiency of a native speaker in correctly inspecting the generated pronunciations of the words. The method proceeds by automatically building Letter-to-Sound (LTS) rules from a small set of the most commonly occurring words in a large corpus of a given language. These LTS rules are retrained as new words are added to the lexicon in an Active Learning step. This procedure is repeated until we have a lexicon that can predict the pronunciation of any word in the target language with the accuracy desired. We tested our approach for three languages: English, German and Nepali.

1. Introduction

Most speech applications utilize language-specific phonetic lexicons to derive word pronunciation. Unfortunately, such lexicons are not readily available for new languages; this creates a significant barrier for applications in speech synthesis and recognition. One way of addressing this problem is to list all pronunciations in the lexicon. This approach, however, is costly in terms of time and errors. Stüker [1] has proposed an alternate strategy for building such lexicons by voting among phoneme recognizers in nine different languages; however, they conclude that, as yet, their method is not suitable for high quality lexicons. Bellegarda [2] has proposed a data-driven grapheme-to-phoneme conversion procedure by computing the distance between new words and words whose pronunciation are known. Other methods have been proposed by [3] and [4] to build or adapt available pronunciation dictionaries, but these methods assume an existing base phonetic lexicon.

Here we present a novel method to build such a base lexicon for a new language. Our approach is based on access to a substantial amount of text in the target language and a small set of seed words with their phonetic transcriptions. In some languages (e.g. Spanish), the relationship between orthography and pronunciation is trivial while in other languages (e.g. English, German), the relationship is more opaque. The more opaque the language, the more difficult it is to build its phonetic lexicon. We have used our method successfully for both trivial (Nepali) and opaque languages (English).

In our approach, we begin with a seed lexicon of hand-annotated pronunciations. We build an initial set of LTS rules from this base lexicon. We add new entries to the lexicon in a bootstrapping process designed to enhance the accuracy of the LTS rules. We iteratively add entries to the lexicon until we reach a desired pronunciation accuracy threshold. We demonstrate that, as we enlarge the lexicon, the information provided by words already in the lexicon help to predict words that are not yet in the lexicon. The important issues we must address in this process are i) what set of words should be added in each iteration, and ii) how do we verify that the predicted pronunciation of words in the lexicon is correct.

To address (i), we regard our process as a form of Active Learning. An active learner always begins with a small set of labeled data. It then looks for instances in a pool of large unlabeled data that increase the classification accuracy most rapidly [5] [6], and then asks a human annotator to label those instances. In our case, we have a large amount of text (unlabeled for pronunciation) and a few seed words labeled with their pronunciations. Since the accuracy of phonetic lexicons can be measured by the number of tokens in any given text that are pronounced correctly, a basic function to select the words which might increase the accuracy of the lexicon would be to add the N most frequently occurring words in the corpus that are not yet in the lexicon. The fact that the 250 most frequent words in English account for more than 50% of the tokens in any random sample of Wall Street Journal (WSJ) [12] text intuitively supports the hypothesis that incrementally adding the most frequent words not yet in the lexicon at each iteration should boost its accuracy. The second issue (ii) is not an easy task in actual practice for a new language. First, we predict the pronunciation of each new word selected for addition to the lexicon, together with a confidence score for that pronunciation. If the confidence score is greater than a certain threshold, the word and its predicted pronunciation is immediately added to the lexicon. Otherwise we ask a native speaker to correct the pronunciation, if necessary.

In this paper, we first describe the bootstrapping algorithm we employ (Section 2). In Section 3 we describe the corpus we used. In Section 4 we describe the experimental setup and accuracy results, and in Section 5 we present our conclusions.

2. The Bootstrapping Algorithm

1. We begin by collecting a substantial amount of text in the target language. We build a set of the most frequent words, which we call set L_i. We select the top 200–500 most frequent words from set L_i depending on the percentage of tokens these words account for in random segment of the corpus as the SEED WORDS for our procedure, which we term S_i. In addition, we also add enough words to ensure that every letter in the language's orthography occurs at least once.

2. We next ask a native speaker to assign pronunciations to S_i. We use a generic data-driven LTS rule system [7] to build a set of initial pronunciation rules from our seed lexicon. We term this set of rules plus lexicon built at this stage P_i.

3. We next select the N most frequent words from the new set L_{i+1} $(L_i - S_i)$, which we term S_{i+1}. The value of N is defined by the user, however, we suggest that N should be small in the first few iterations and larger in later iterations. Since the earlier LTS rules will be less accurate, a large N would mean that more words are mis-pronounced, and thus must be corrected in the active learning stage.

4. We then predict pronunciations for each word in S_{i+1} using our previously derived LTS rules and lexicon the pronunciation module P_i.

5. We compute confidence scores C_o for all the pronunciations generated by the system for these words. We first compute C_o using an orthography distance measure between the test word **w** and all words in the lexicon.

$$C_o = \frac{2 * |(lcs(w, w_k)|}{|w| + avg(|w_k|)} \tag{1}$$

$k =$ *1..m (Num. of words in lexicon with LCS)*

The computation of C_o assumes that similar orthography correlates with similar pronunciation. So, if there are words in the lexicon that are orthographically similar to **w**, our confidence in the pronunciation predicted by P_i for **w** will be higher. C_o is computed by finding all words w_k in the lexicon with the longest common subsequence (LCS) 'q' compared with **w**; e.g., if the LCS between **w** and any word in the lexicon has length 'q', then we find all words in the lexicon with LCS of length 'q', whether they are the **same** strings or not. We term this set S_w. Then we divide 2 times 'q' (the length of the LCS) by the sum of the length of **w** and the average length of S_w. Thus, if there is a close to exact match between **w** and any current lexical entries, the confidence score for the pronunciation of **w** will be close to 1; otherwise it will be less. If the C_o is higher than some threshold t_o, we compute an additional confidence score, C_p.

6. After finding the orthographically similar words to **w**, we next want to compute how closely related the pronunciation of the words in S_w with the pronunciation of **w** proposed by P_i. There may be many words orthographically similar to **w** which have quite different pronunciations; in such cases we should not be confident about the pronunciation P_i has predicted for **w**. We compute C_p for the words in S_w using a variant of the phonetic distance measure DICE [8].

$$C_p = \frac{2}{m}\sum_{k=1}^{m} \frac{\frac{|trigrams(w) \cap trigram(w_k)|}{(1+(pos(w)-pos(w_k))^2)}}{(|trigram(w)| + |trigram(w_k)|)} \tag{2}$$

$k =$ *1..m (Num of words in* S_w

For each word in S_w, we compute the trigram overlap of the phones of its pronunciation with the phones of the test word and scale it by the position of the overlap. That is, we weight overlap occurring in similar position in the phone sequence more highly. The denominator in equation 2 takes account of the length of the phone sequences in the pronunciations of **w** and w_k. We compute average of these scores for all M words in the set S_w to obtain the confidence score C_p for **w**. If C_p is greater than a threshold t_p, we assume that the pronunciation proposed for **w** by P_i is correct, and we add **w** and its pronunciation to the lexicon.

The following is an example from one of iteration of this procedure:

TEST WORD: rune TEST PHONES: /r uw n/
prunes : /p r uw n z/
pruned : /p r uw n d/
brunei : /b r uw n ay/
prune : /p r uw n/
brunet : /b r uw n eh t/
gruneich : /g r uw n ay k/
where C_o =0.78688 C_p =0.24722

We can safely keep the second threshold t_p low, as we have already filtered candidate words with C_o. In our example above, if the $t_o < 0.78$ and $t_p < 0.24$ then the pronunciation of the test word *rune* would be considered highly likely to be correct and added to the lexicon. In the early iterations, t_o is set fairly high, so that incorrect pronunciations are unlikely to be added to a small lexicon.

7. On the remaining words of S_i — for which the pronunciation hypothesized by P_i has been deemed

below threshold by the above procedure — we ask a human annotator to evaluate their pronunciations and correct them by hand if needed. We then add these words to the lexicon as well. From this new lexicon, we build a new set of LTS rules and thus produce a new pronunciation module P_{i+1}.

We iterate from 2 to 7 until the lexicon correctly predicts pronunciations for some percentage of new words deemed "good enough" for the purposes of the application.

3. Data and Languages

We did experiments using this procedure on three languages: English, German and Nepali.

3.1. Nepali

Nepali is an Indic language spoken by less than 25 million people, primarily Nepalese. No phonetic lexicon was available for Nepali until the current research produced one. Written Nepali uses Devanagari script similar to Hindi; there is a close relationship between orthography and pronunciation. However the language is still complex enough that the mapping is not trivial. Some characters in the Devanagari script, such as that corresponding to the phone /iy/ may appear orthographically either before or after certain consonants but they are always pronounced at the end of the consonant. We identify such characters explicitly in the LTS rules.

We collected Nepali text comprising more than 1 million words of text from approximately three years of daily newspapers in Nepali [9]. We pre-processed the text to remove English words that are sometimes embedded within Nepali text. We further normalized the text by removing punctuation and numbers. Then we obtained the unique words and sorted them by frequency to obtain the top 350 words as our initial seed set.

3.2. German

For our second test language, German, we obtained a large volume of text from online newspaper sources. German already has a phonetic lexicon, CELEX [10] so we used this lexicon as an oracle, in place of a human annotator, to evaluate and 'correct' our hypothesized pronunciations as described in Section 2, step 7. For the purposes of this paper, we assume these pronunciations to be correct.

3.3. English

For English, we used 50 million words of WSJ text [12] as a source of data. As for German, we used an existing lexicon, the Carnegie Mellon Pronouncing Dictionary (CMUDICT) [11] as an oracle to evaluate and correct pronunciations hypothesized by our method. We again assume these pronunciations to be correct.

4. Experiments and Results

For each language, we first identified the J most frequent words, where J varied between 118K for English, 225K for German, and 100K for Nepali. We then divided J into **p** development sets. We made the first few such sets small – 100 words each for the sets used in the first 10 iterations of the algorithm. We increased this size for later iterations to 250, 500, 750, 1000, 2500 and 5000 words, decreasing the number of iterations used with each set to avoid fatigue for the Nepali annotators.

We also created 10 test sets of 5000 word (tokens) of held out data for English from the news corpora described in Section 3. We term these the TOKENS test sets. From each TOKENS set, we created two other test sets: A set consisting of all the unique words in the tokens set – the TYPES test sets – and a set of all the types occurring more than once in the TYPES set, which we term TYPES_PLUS.

We tested on each of these test sets at each iteration of lexicon creation – each in two different scenarios. i) We check for pronunciations of all test words manually or by oracle and add these to the lexicon; ii) We use the metrics C_o and C_p to identify pronunciations we are confident in and automatically add these to the lexicon, and then check the pronunciation of the remaining words manually or by oracle and them as well.

We tested our algorithm on English, German and Nepali. The results for English are shown in the Figure 1. Out initial 250 word seed set produced a lexicon of more

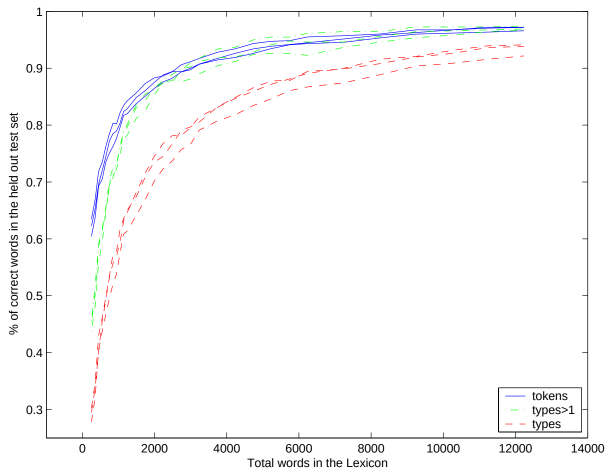

Figure 1: *Accuracy of bootstrapped English lexicon on each iteration for each of the 3 types of tests for 3 held out test sets*

than 55% accuracy on the Tokens test set. Lexicon accuracy on held out test data quickly grew to above 90% in just 18 iterations with only 4000 entries in the lexicon; after the twentieth iteration, accuracy improved only slightly. On average, confidence scores added 9.92% of test words automatically. Average precision for the confidence metric was 87.79% with the threshold C_o and C_p set at 0.8 and 0.4, respectively after iteration 12.

After only 23 iterations we obtained a lexicon for English with 16,000 entries with a pronunciation accuracy of 94.31% on the held-out TOKENS test set and 87.23% on TYPES set. Furthermore, it is approximately one-seventh the size of CMUDICT. The accuracy achieved for the TYPES test set was lower than for the TOKENS test set by 7.08%, as we had expected.

For German with a lexicon of only 350 words, we obtain a pronunciation accuracy of 35% on the TOKENS test set. In subsequent iterations, accuracy increased in the same way as for English. After 13 iterations, our procedure achieved an accuracy of 90%. For the TYPES test set, we obtained a maximum of 80.67% accuracy, and for TYPES_PLUS our maximum accuracy was 87.04%. After ten iterations accuracy on TYPES_PLUS increased very little, perhaps due to the high variation in the morphological structure of words in German.

Finally, we tested our procedure on Nepali, for which no phonetic lexicon was previously available. The lexicon had 350 most frequent words in the first iteration. The lexicon and its LTS rules at this stage had accuracy rate of 60%, when manually checked. We obtained a precision of 91.47% for our confidence metrics, C_o and C_p. No held-out test data was available for Nepali, so instead we manually tested about 100 word types in the development set for each iteration. In average 12 words were added to the lexicon automatically using our confidence scores; the rest had to be checked by hand. Thus, our confidence scoring reduced the words that had to be manually checked by 12%.

Lexicon accuracy for word types occurring with frequency > 3 had an accuracy of 80.9% initially. After 16 iterations, we obtained a pronunciation accuracy of 92.9% on new data. For tokens that were not in the lexicon, the initial accuracy was 69.8%. After bootstrapping was complete we obtained an accuracy of 94.6%.

We believe that fewer words need to be verified in Nepali compared to the other two languages (English and German) to achieve more than 90% accuracy rate because Nepali has the advantage of having orthography and pronunciation more closely related.

5. Conclusion

We presented a bootstrapping algorithm that can be used to build phonetic lexicons for new languages without existing phonetic lexicons but where large amounts of unlabeled text are available. Indeed, new, accurate, and efficient lexicons can be built in as short a duration as a couple of weeks, as we did for Nepali. While it may be easier to produce a lexicon for a language like Nepali, whose orthography and pronunciation are closely linked, we have also demonstrated that a similar process can be followed for more opaque languages like English and German, where there is a looser relationship between written and oral forms. Our technique achieves significant coverage for a language with a significant saving in manual labor, since, by using active learning techniques and effective confidence metrics, only some data must be labeled by hand. We also show that a lexicon built using our method is likely to need fewer entries than existing lexicons for English and German, because of our incremental approach; for example, our bootstrapped English lexicon is five times smaller than the CMU lexicon but achieves high accuracy nonetheless.

6. Acknowledgments

This work was partially supported by DARPA/TIDES grant N66001-00-1-8919. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of DARPA/TIDES.

7. References

[1] Stüker, S., “Automatic Generation of Pronunciation Dictionaries - For New, Unseen Languages by Voting Among Phoneme Recognizer in Nine Different Language”, Studienarbeit, Institut für Logik, University of Karlsruhe.

[2] Bellegarda, J.R, “A Novel Approach to Unsupervised Grapheme to Phoneme Conversion” Pronunciation Modeling and Lexicon Adaptation for Spoken Language, 2002

[3] Riley, D. M., and Ljolje A., “Automatic Speech and Speaker Recognition: Advanced Topics”, Kluwer Publishers, 1995

[4] Lamel, L., and Adda, G, “On Designing Pronunciation Lexicons for Large Vocabulary, Continuous Speech Recognition”, Proceedings of ICSLP-96, pp 6-9

[5] Sarawag, S., Bhmidipaty, A., “InterActive Deduplication using Active Learning”, SIGKDD 02 Edmonton, Alberta, Canada, 2002

[6] Tong, S., Koller, D., “Support Vector Machine Active Learning with Applications to Text Classification”, Journal of Machine Learning Research, pages 45-46, 2002

[7] Black, A. Lenzo, K., and Pagel, V., “Issues in building general letter to sound rules”, Proc. ESCA Workshop on Speech Synthesis, (Australia, 1998), pp. 77–80.

[8] Brew, C. and D. McKelvie, “Word-pair extraction for lexicography.”, In Proceedings of International Conference on New Methods in Natural Language Processing, Bilkent, Turkey, pp. 45-55

[9] Mercantile Communications Pvt. Ltd, Nepal news, http://www.nepalnews.com, 2001.

[10] Linguistic Data Consortium, Celex2, LCD96L14, 1996.

[11] CMU, Carnegie Mellon Pronouncing Dictionary, 1998.

[12] Linguistic Data Consortium, Wall Street Journal, 1995.