Bootstrapping Pronunciation Dictionaries: Practical Issues

M. Davel and E. Barnard

Human Language Technologies Research Group
AAIICT / University of Pretoria, Pretoria, 0001, South Africa.
mdavel@csir.co.za, ebarnard@up.ac.za

Abstract

Bootstrapping techniques are an efficient way to develop electronic pronunciation dictionaries [1, 2], but require fast system response to be practical for medium-to-large lexicons. In addition, user errors are inevitable during this process, and it is useful if automatic means can be used to assist in the search for such errors. We describe how the Default&Refine grapheme-to-phoneme rule extraction algorithm [3] can be adapted to meet both of these goals. Experimental results demonstrate the utility of these methods.

1. Introduction

Electronic pronunciation dictionaries can be created efficiently through the use of bootstrapping [1, 2]. In prior work, we have developed an audio-enabled bootstrapping approach that combines machine learning and human intervention during the dictionary creation process in a way that minimizes and simplifies the human effort required. This approach is the basis for an open-source system that has been used for the development of several pronunciation dictionaries [1, 4, 5].

During use of the system, two important practical issues have arisen repeatedly, namely (a) the need for rapid system response, and (b) the requirement to validate user inputs carefully, in order to minimize the number of erroneous words used in training the system. In this paper, we report on adaptations to a previously reported algorithm that allow us to address these requirements.

In Section 2 we review the general bootstrapping approach that is used in our system. Section 3 describes the incremental variant of this approach which provides the necessary acceleration of the process, and reports on a number of experiments which reveal its advantages and limitations. Section 4 shows how the evidence from the extracted rules can be used to flag potential user errors.

2. Background

The core of the dictionary bootstrapping process relies on an efficient grapheme-to-phoneme pronunciation prediction algorithm. This algorithm is used to generalise from the existing dictionary in order to predict additional entries, increasing the size of the dictionary in an incremental fashion.

2.1. The bootstrapping process

The bootstrapping system is initialised with a large word list (containing no pronunciation information), or with a pre-existing pronunciation dictionary, if such a resource is available. The system chooses the next 'best' word to consider, predicts a pronunciation for this word and presents a human dictionary developer with an audio version of the predicted pronunciation. The human acts as a 'verifier' and provides a verdict with regard to the accuracy of the word-pronunciation pair: whether the pronunciation is *correct* as predicted, or not. The verifier can also indicate that the word itself is *invalid*, *ambiguous* depending on context, or that he or she is *uncertain* about the status. If the word is wrong, the verifier specifies the correct pronunciation by removing, adding or replacing phonemes in the presented pronunciation. A new audio version is generated, for which the verifier can specify a new verdict. At this stage, the learning algorithm updates the word-to-pronunciation model in order to account for the corrected pronunciation. The process is repeated (with increasingly accurate predictions) until a pronunciation dictionary of sufficient size is obtained.

2.2. The Default&Refine algorithm

The predictive ability of the rule extraction technique substantially influences the speed at which the system learns, and therefore the efficiency of the bootstrapping process. In the bootstrapping system described here, we utilise the Default&Refine algorithm [6] for rule extraction. Iterative Viterbi alignment is used to obtain grapheme-to-phoneme mappings, after which a hierarchy of rewrite rules is extracted. The rule set is extracted in a straightforward fashion: for every letter (grapheme), a default phoneme is derived as the phoneme to which the letter is most likely to map. "Exceptional" cases – words for which the expected phoneme is not correct – are handled as refinements. The smallest possible context of letters that can be associated with the correct phoneme is extracted as a refined rule. Exceptions to this refined rule are similarly represented by further refinements, and so forth, leading to a rule set that describes the training set with complete accuracy. Although this approach is very straightforward, it is quite competitive in terms of learning efficiency (that is, the accuracy achieved with a limited number of training examples) and asymptotic accuracy, when compared to alternative approaches [3]. Further detail with regard to our implementation can be found in [3].

3. Accelerating the process by incremental learning

Since the bootstrapping system updates the rule set after every correction by the verifier, the time taken for such updates is of crucial importance. The update speed is influenced by two factors: the alignment speed and the rule extraction speed. If n represents the number of words in the training dictionary and m the number of conflicting patterns in the training dictionary,

then the complexity of the alignment process is $O(n)$, and that of the rule extraction process approximately $O(m) \leq O(n)$, if it is assumed for the sake of simplicity that all words are more or less of equal length. This is typical of various of the rule extraction techniques that are appropriate for g-to-p bootstrapping.

If the entire set of training words is processed after every correction, the update time becomes a limiting factor as the dictionary grows. In our implementation, continuous updating becomes unwieldy when the number of words with known pronunciations exceeds approximately 2000. On the other hand, by performing batch updates at specific times that suit the verifier (e.g. at the end of a verification session), the update time does not become a constraint, but the learning obtained during the session is not utilised to refine models until after the end of the session. In order to obtain an algorithm that allows for continuous model updating while keeping the update time within acceptable limits, an incremental version of the Default&Refine algorithm was developed.

While the original algorithm creates a set of graphemic rule trees (one tree per grapheme) from the training set by considering all the training words simultaneously, the incremental version utilises the trees constructed during the previous (batch mode) update, and adds the new refinements as leaves to these trees: for each grapheme in the new word, if the realised phoneme is predicted accurately by the current graphemic tree, no update occurs; otherwise the smallest rule is extracted that will describe the new word without affecting any of the existing predictions. This version has $O(d)$ complexity where d represents the average depth of the various graphemic rule trees (which is approximately equivalent to the average context size of the graphemic rule set). Using this incremental process, additional learning can be obtained from the new words added without causing discernible delay, even for large training dictionaries.

In practice, the bootstrapping process operates in two phases: during the first phase a batch update occurs for every word; during the second phase a batch update occurs at synchronisation events only, and incremental updates are performed in between synchronisation events. The interval between synchronisation events is based on a set number of "update words", i.e. words that have been corrected by the verifier (words that were correctly predicted prior to verification do not contribute to this count). At the end of this interval, a synchronisation event occurs: the complete training dictionary is re-aligned, and new rules are extracted in batch mode. During the update interval, the Viterbi probabilities calculated at the previous synchronisation event are used per word to perform a fast alignment (the probabilities are used in the standard way, but not updated) and incremental Default&Refine is used to extract additional rules from the single aligned word-pronunciation pair. Phase 2 is initiated well before the time required by the full update event becomes noticeable. (For our current system we progress from phase 1 to phase 2 when 1500 valid words have been processed.)

As can be expected, the new algorithm is an approximation of standard Default&Refine, and therefore somewhat less accurate than the original. We evaluate the performance of the system using an existing pronunciation dictionary: *Fonilex*, a pronunciation dictionary of Dutch words as spoken in the Flemish part of Belgium [7]. In order to determine the efficiency of the incremental approach, we first compare the two rule extraction processes (incremental mode and batch or standard mode) without taking changes in alignment into account. We utilise

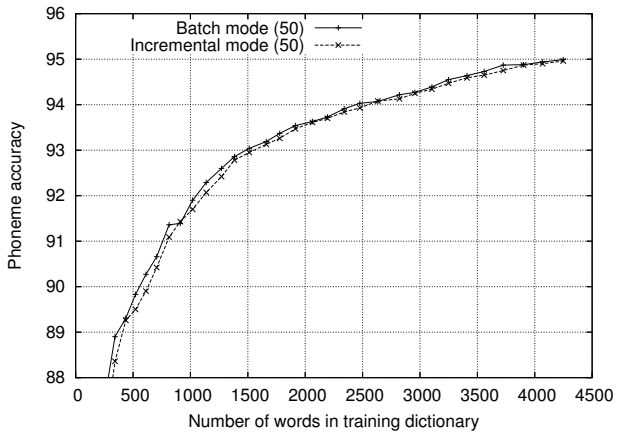

Figure 1: *Phoneme accuracy comparison for incremental and batch mode at an update interval of 50.*

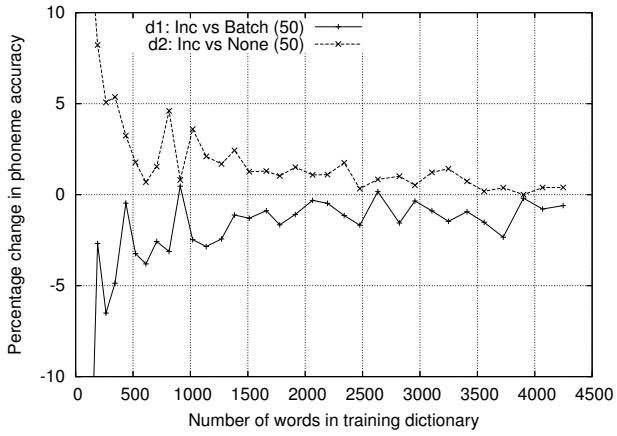

Figure 2: *Relative change in phoneme accuracy when comparing incremental with batch mode (δ_1), and incremental mode vs no updating between synchronisation events (δ_2); both at update interval 50.*

the same set of alignments[1] for both types of rule extraction, and measure phoneme accuracy on the same training set using the two different algorithms. We find that the decrease in accuracy is slight once the graphemic trees are of sufficient size, as demonstrated in Fig. 1 for a synchronisation interval of 50. The difference in accuracy can be analysed in further detail by calculating two values: δ_1, the relative increase in phoneme error rate when utilising the incremental mode compared to the batch mode, and δ_2, the relative decrease in phoneme error rate when utilising the incremental mode, in comparison with only performing updates at synchronisation events and not updating the models in between; that is,

$$\delta_1(x) = \frac{inc(x) - batch(x)}{1 - batch(x)} * 100 \quad (1)$$

$$\delta_2(x) = \frac{inc(x) - batch(x-1)}{1 - batch(x-1)} * 100 \quad (2)$$

and where $batch(x)$ indicates the phoneme accuracy using batch rule extraction, and $inc(x)$ the phoneme accuracy using incremental rule extraction, both at synchronisation point

[1]The alignments used were obtained from a 173,873-word training dictionary.

x. Fig. 2 illustrates the trends for the δ_1 and δ_2 values for an update interval of 50 (still utilising ideal alignments), providing an additional perspective on the same data as displayed in Fig. 1.

The effect on rule set accuracy is strongly influenced by the length of the update interval. We therefore compare the performance of the two algorithms for different update intervals, and find that the average δ_1 and δ_2 values are both fairly linear in relation to the update interval: the longer the interval, the less accurate incremental updating becomes when compared with batch updating, and the more value is provided by incremental updating vs performing no updates in between synchronisation events. In Fig 3 we plot the δ_1 and δ_2 values for update intervals of length $50, 100, 150$ and 200 during the first 4500 words of bootstrapping. These trends continue for larger update intervals.

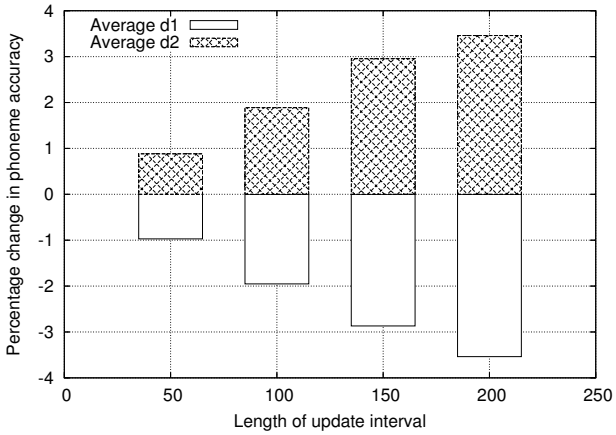

Figure 3: *Average δ_1 and δ_2 values for update intervals of length 50,100,150 and 200.*

Finally, in order to ensure that the fast alignment process does not introduce a noticeable loss in accuracy, we compare the two algorithms (batch and incremental rule extraction), applying the alignment process as it would be used in practise: performing a full alignment during synchronisation events and using the fast alignment process in between. We find that while there is a greater variance in the effect on phoneme accuracy when using the fast alignment process during the first phase of bootstrapping, this effect becomes negligible during the second phase of bootstrapping. (In practice, fast alignment is only used during the second stage of bootstrapping.) In Fig 4 we plot the δ_1 values for an update interval of 50, when using ideal alignments and actual alignments.

The above results indicate that incremental Default&Refine provides an effective way of increasing system responsiveness. As there is a clear trade-off between the length of an update interval and learning efficiency, the update interval can be chosen in a way that is suitable for the specific dictionary developer: longer continuous sessions (requiring slightly more corrections), or shorter sessions with frequent breaks. As the dictionary size increases and the rule set approaches asymptotic accuracy, the number of words considered between synchronisation events increases automatically[2]. For large dictionaries, the batch update process can become a daily event, rather than an hourly event, as would be the case for relatively small dictionaries.

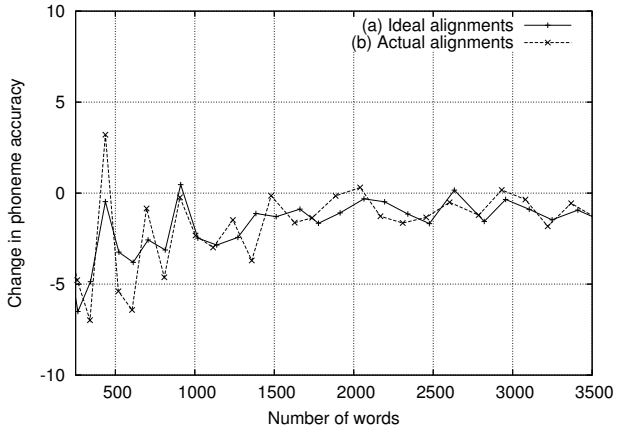

Figure 4: *Change in phoneme accuracy (δ_1) when comparing incremental with batch mode when (a) ideal alignments are used, and (b) when actual alignments are used.*

4. Semi-automatic detection of verifier errors

Dictionary developers are typically required to enter phonemic predictions for several thousand words in order to develop dictionaries of sufficient accuracy. Although our interface attempts to assist developers in this task (e.g. by audibly sounding out the chosen pronunciations and by providing automatic predictions for every word), it is inevitable that errors will arise from time to time.

Fortunately, the Default&Refine approach is well suited to assist in the detection of such errors. Since every rule in the hierarchy is selected to describe a particular set of words, and errors are likely to result in rules that are applicable to few words besides the erroneous one, one expects that erroneous transcriptions will tend to show up as rules which support few words. Of course, there may also be valid pronunciation rules which are not supported by many examples; it therefore is an experimental issue to determine how useful this guideline is in practically detecting transcription errors. Different languages will differ in this regard – a highly "regular" language such as Spanish [3] will generally have many examples of each valid rule, whereas the idiosyncrasies of English pronunciation will produce a large number of valid special cases. As a consequence, our approach is expected to be more successful for languages such as Spanish.

To investigate the utility of the proposed method for detecting transcription errors, we have conducted a number of simulation experiments with Afrikaans, which is a Germanic language with a fairly regular grapheme-to-phoneme relationship. Heuristically, we expect Afrikaans to lie somewhere in the middle of the continuum between regular and irregular languages. Our experiments used a dictionary with 5 013 words, which were transcribed by a linguistically sophisticated first-language Afrikaans speaker and verified by the authors. Based on earlier experience with dictionary developers who are more error prone [5], we artificially corrupted a fraction of these transcriptions and then measured the efficiency of the number-of-words

[2]For example, using an update interval of 50, approximately 200 training words are considered per session when just past the 4000-word mark. (See Fig. 2.)

[3]That is, a language with a very regular mapping between phonemes and graphemes.

guideline to indicate the words with corrupted transcriptions. We introduced two types of corruptions into the transcriptions:

- *Systematic corruptions* reflect the fact that users are prone to making certain transcription errors - for example, in the DARPA phone set, *ay* is often used where *ey* is intended. We allowed a number of such substitutions, to reflect observed confusions by Afrikaans transcribers.
- *Random corruptions* simulate the less systematic errors that also occur in practice; in our simulations, random insertions, substitutions and deletions of phonemes were introduced.

We generated four corrupted data sets (systematic substitutions; random insertions, substitutions and deletions), where 1% of the words were randomly selected for corruption. Default&Refine rule sets were then generated for each case [4], and the percentage of erroneous words that are matched by the most specific rules was determined. (Since Default&Refine always applies rules in the order most to least specific, the rule ordering used for prediction was used as measure of specificity. The specificity of a word was taken as the specificity of its most specific grapheme, since a transcription error may result in one or more rules becoming highly specific to that word.) In Fig. 5 we show the fraction of errors that remain undetected against the fraction of words examined, as this threshold of specificity is adjusted. (Note that this depiction is closely related, but not identical, to that in the well-known Detection Error Tradeoff (DET) curves [8].)

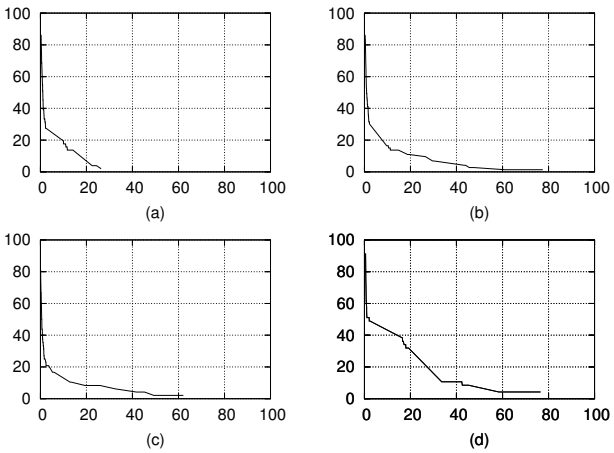

Figure 5: *Fraction of erroneous words that are not detected as a function of the fraction of all words examined, when words are examined in the order of their most specific rules, for various types of corruptions: (a) random substitutions (b) random insertions (c) random deletions and (d) systematic substitutions.*

These results suggest that this method has significant use in accelerating the process of error detection. For all three types of random errors, more than 90% of the errors can be identified after inspecting fewer than 20% of the transcriptions. As far as the systematic errors are concerned, about half the errors occur in the first 5% of the words inspected; by that time, the systematic patterns are obvious, and can be used to select other candidate words where these same errors may have occurred.

In practice, the error-detection process can be combined with the synchronisation event, with possible errors flagged by the bootstrapping system and corrected where necessary by a human verifier, prior to continuing with the next session. This then becomes a simple and efficient way of identifying errors during bootstrapping. Alternatively, the error-detection process can be used as a stand-alone technique, in order to identify possible errors in a pronunciation dictionary developed via different means.

[4] The full algorithm was used, not the incremental version described in Section 3.

5. Conclusion

Bootstrapping is an important approach to the development of resources in human language technologies. In the specific case of developing pronunciation dictionaries, the past few years have seen significant progress in terms of available algorithms and systems. The techniques discussed here make it possible to maintain acceptable system response times for growing vocabulary sizes, and also to detect errors in such dictionaries in an efficient manner. It is hoped that these and related developments will accelerate the development of high-quality open resources for many of the languages of the world.

6. Acknowledgements

This work was supported by the *African Advanced Institute for Information and Communication Technologies (AAIICT)*.

7. References

[1] M. Davel and E. Barnard, "Bootstrapping for language resource generation," in *Proceedings of the Symposium of the Pattern Recognition Association of South Africa*, South Africa, 2003, pp. 97–100.

[2] S. Maskey, L. Tomokiyo, and A.Black, "Bootstrapping phonetic lexicons for new languages," in *Proceedings of Interspeech*, Jeju, Korea, October 2004, pp. 69–72.

[3] M. Davel and E. Barnard, "Prediction pronunciations with default&refine: analysis and results," in *Proceedings of Interspeech (submitted for publication)*, Lisboa, Portugal, October 2005.

[4] M. Davel and E. Barnard, "The efficient creation of pronunciation dictionaries: Machine learning factors in bootstrapping," in *Proceedings of Interspeech*, Jeju, Korea, October 2004, pp. 2781–2784.

[5] M. Davel and E. Barnard, "The efficient creation of pronunication dictionaries: Human factors in bootstrapping," in *Proceedings of Interspeech*, Jeju, Korea, October 2004, pp. 2797–2800.

[6] M. Davel and E.Barnard, "A default-and-refinement approach to pronunciation prediction," in *Proceedings of the Symposium of the Pattern Recognition Association of South Africa*, South Africa, November 2004, pp. 119–123.

[7] P. Mertens and F. Vercammen, "Fonilex manual," Tech. Rep., K.U.Leuven CCL, 1998.

[8] A. Martin, G. Doddington, T. Kamm, M. Ordowski, and M. Przybocki, "The det curve in assessment of detection task performance," in *Proceedings of the European Conference on Speech Communication and Technology*, 1997, pp. 1895–1898.