The Efficient Generation of Pronunciation Dictionaries: Machine Learning Factors during Bootstrapping

Marelie Davel and Etienne Barnard

CSIR / University of Pretoria
Pretoria, South Africa
mdavel@csir.co.za ebarnard@up.ac.za

Abstract

Several factors affect the efficiency of bootstrapping approaches to the generation of pronunciation dictionaries. We focus on factors related to the underlying rule-extraction algorithms, and demonstrate variants of the Dynamically Expanding Context algorithm, which are beneficial for this application. In particular, we show that continuous updating of the learned rules, coupled with a new approach to grapheme-to-phoneme alignment and a sliding-window approach to choosing the context window, leads to an efficient and accurate bootstrapping mechanism.

1. Introduction

An accurate pronunciation dictionary or letter-to-sound conversion model is an important resource when developing speech technology for a new language. The development of either of these resources typically requires significant effort and linguistic expertise. This can be an obstacle in language environments where prior linguistic resources do not exist and skilled computational linguists are not readily available. In prior work we therefore proposed audio-enabled bootstrapping [1] as an effective approach to the development of pronunciation dictionaries and/or rule sets. The aim of this approach is to combine machine learning and human intervention during the dictionary creation process in a way that minimizes the amount and sophistication of human effort required. This is achieved by (a) optimizing the speed and accuracy with which the system learns from the human input, and (b) minimizing the effort required by the human verifier to identify errors accurately.

Our initial explorations focused on the feasibility of the proposed process and predicted that significant acceleration could be achieved via this approach. In [2] we explored ways in which the process can be optimised by analysing user-system interaction. In this paper we describe the techniques implemented to optimise the process from a machine learning perspective, and report on the results achieved. Section 2 provides background with regard to the use of bootstrapping in language resource generation. Section 3 describes the specific machine learning techniques implemented and section 4 describes the experimental results obtained.

2. Background: Bootstrapping

Bootstrapping techniques, including cross-language bootstrapping, have proven useful for the cost-effective development of language resources in new languages. For example, when acoustic models are developed for a new target language, an automatic speech recognition system can be initialised with models from an acoustically similar source language, and these initial models improved through an iterative process during which audio data in the target language is automatically segmented and used to retrain the target language acoustic models. The potential saving in effort achieved through such a process has been well demonstrated[3].

Bootstrapping approaches are applicable to various language resource development tasks, specifically where an automated mechanism can be defined to convert between various representations of the data considered. In the above example, two representations are utilised: annotated audio data and acoustic models, and the mechanisms to move from one representation to the other are well defined through the phoneme-alignment and acoustic modelling tasks respectively.

This general approach can be applied to the task of creating a pronunciation dictionary, using word/pronunciation pairs and word-to-pronunciation rules (also referred to as letter-to-sound, grapheme-to-phoneme or G2P rules) as alternative representations of the same information, as we described in [1].

Here the system is initialised with a large word list (containing no pronunciation information). 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. 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.

3. Optimizing system learning efficiency

We now discuss a number of algorithmic factors that are important for the efficiency of the bootstrapping approach.

3.1. System continuity

The faster the system learns, the fewer corrections are required of the human verifier, and the more efficient the bootstrapping process becomes. The most important aspect that influences the speed at which the system learns relates to the continuity with which the system updates its knowledge base. A continuous process was chosen, whereby the system regenerates its predic-

tion models after every single word verified. This has a significant effect on system training responsiveness, especially during the initial stages of dictionary development when the system has access to very little information on which to base its predictions.

3.2. Word-to-pronunciation prediction model

A second aspect that influences system performance significantly relates to the type of word-to-pronunciation prediction model used. The ideal formalism would be able to represent the word/pronunciation data exactly, have high predictive ability and achieve an acceptably high level of performance at a low computational cost for model training – an important consideration for our system, given that continuous model updating is required.

Various formalisms have been used for the generic task of pronunciation prediction, including explicit hand-crafted grapheme-to-phoneme mapping rules, neural networks, decision trees and various forms of instance-based learning. The results when applying appropriate versions of the different formalisms mentioned above are comparable, with different algorithms outperforming others in different conditions. Kohonen's Dynamically Expanding Context (DEC), initially applied by Torkkola to the G2P problem [4], is a popular instance-based learning algorithm that predicts phoneme realisation based solely on graphemic context. As DEC meets the specific requirements for our bootstrapping system (as mentioned above), a variation of DEC was chosen as rule extraction mechanism.

In DEC, each rule specifies a mapping of a single grapheme to a single phoneme for a given left and right graphemic context, i.e is of the form: *(left context, grapheme, right context)* $\rightarrow$ *phoneme*. Each word in the training dictionary is first aligned with its pronunciation on a per-grapheme basis. For each grapheme-to-phoneme alignment pair, rules are extracted by finding the smallest graphemic context that provides a unique mapping to a specific phoneme. In DEC, if an $n-$letter context is not sufficient, the context is expanded to either the right or the left. This 'specificity order' influences the performance of the algorithm. In the initial implementation, the set of rules was then ordered in an efficient tree structure.

Two variations on traditional DEC are implemented: (1) Using a sliding window for each context size, and (2) Overgrowing the set of rules for each context size, optionally removing redundant rules. DEC, as applied by Torkkola [4] expands the context one letter at a time, either favouring the right- or left-hand side explicitly. We use a sliding window that first considers all possible contexts of size *n*, before continuing to consider contexts of size *n+1*. This prevents a rule with unnecessarily large context from being extracted, when a more general rule would suffice. Since multiple rules of the same context size may apply to a single grapheme-to-phoneme mapping (such as *re,s,ti* $\rightarrow$ *s* and *ere,s,t* $\rightarrow$ *s*), contexts that are already served by existing rules can be removed to prevent over-specialisation. Because all contexts of each size are considered, the order in which contexts are expanded (for a specific context-level) becomes insignificant.

Generating a new pronunciation is a simple procedure: each grapheme in the word is considered in turn, and the rule describing the largest matching context is used to predict the phoneme to be generated. When a shifting window is used, more than one conflicting rule of the same size may apply to a word. Various conflict resolution strategies can be implemented: in the set of experiments reported below, the most frequently observed rule is selected.

3.3. Grapheme-to-phoneme alignment

In order to extract DEC G2P rules, each word/pronunciation pair in the current pronunciation dictionary must be aligned on a grapheme-to-phoneme basis, inserting graphemic and phonemic nulls as required. (A phonemic null is inserted where a single phoneme is produced from more than one grapheme, and a graphemic null where a single grapheme is realised as more than one phoneme.)

Errors in grapheme-to-phoneme alignment do not affect different rule extraction techniques to the same extent. The DEC-based rule extraction mechanisms used by this system is sensitive to alignment accuracy. For example, the correct DEC extraction rule for the grapheme-pair 'aa' in Afrikaans is $a\ a -> a$:0 where 0 indicates the null phoneme. If the system incorrectly aligns the words "daar" and "waar" as follows: $d\ a\ a\ r -> d\ a : 0\ r$ and $w\ a\ a\ r -> v\ 0\ a : r$, DEC will not be able to extract the fairly simple rule specified above, as the two words provide conflicting evidence with regard to the pronunciation of 'aa'. For this reason, the grapheme-to-phoneme alignment process is optimised. Where the alignment process is typically based on the set of probabilities that a grapheme is realised as a specific phoneme, we add an additional set of probabilities: the probability that a grapheme is realised as a null phoneme, given the identity of the preceding non-null phoneme observed.

3.4. Word validation order

The final system design aspect considered that has a significant influence on the speed at which the system learns, relates to the mechanism whereby the next 'best' word to add to the knowledge base is predicted. As user verification speed relates first and foremost to the number of incorrect phonemes to correct (see [2]) shorter words are chosen first. The system grows its understanding of pronunciations-in-context systematically. Contexts of varying sizes are ordered according to occurrence frequency in general text, creating a list of 'contexts in question'. A continuous process predicts the next best word to verify based on the current state of the system: the shortest word is chosen that contains the next context in question.

4. Experimental results

In order to measure the accuracy of the dictionary development process from a system perspective, the process was tested on three existing pronunciation dictionaries:

- *NETtalk*, a publicly available 20,008-word English pronunciation dictionary[5]. Hand-crafted grapheme-to-phoneme alignments are included in the dictionary.
- *Fonilex*, a publicly available 200,000-word pronunciation dictionary of Dutch words as spoken in the Flemish part of Belgium[6].
- a 3,439-word Afrikaans pronunciation dictionary, built using the bootstrapping system.

4.1. G2P alignment accuracy

Iterative forced Viterbi alignment (*Align v1*) is used to align each grapheme-to-phoneme pair, inserting phonemic nulls where required, as used by Anderson[7] and others. Graphemic nulls do not occur in the *NETtalk* corpus. For other corpora, our algorithm inserts graphemic nulls during a pre-processing step: graphemic null generator pairs (two graphemes that result in more than two phonemes) are identified by Viterbi-alignment of

all word-pairs where pronunciation length is longer than word length. The initial probabilities for both types of Viterbi alignment are obtained from words and pronunciations that have equal length, and the alignment process is repeated until no further likelihood improvement is observed. Alignment accuracy on the *NETtalk* corpus using this implementation is higher than the results reported by Anderson [7], as compared in Table 1. The basic algorithms are similar, apart from the different handling of graphemic nulls.

An additional improvement can be obtained if the transcription convention used by *NETtalk* is adapted. In *NETtalk*, null phonemes are used to identify graphemes that are "deleted" during pronunciation, for example the word *writer* is transcribed as $w\ r\ i\ t\ e\ r\ ->\ 0\ r\ A\ t\ 0\ R$. An alternative convention would be to use null phonemes simply to identify instances where two or more graphemes give rise to a single phoneme (without identifying a particular grapheme as deleted), by aligning the first grapheme in such a group with a non-null phoneme, and subsequent graphemes with nulls. Using this convention, the word *writer* is transcribed as $w\ r\ i\ t\ e\ r\ ->\ r\ 0\ A\ t\ R\ 0$

Using a set of about 40 rewrite rules, the *NETtalk* dictionary can be rewritten using either the one convention or the other. Using the second convention, the dictionary responds better to data-driven alignment and the second version of our Viterbi algorithm (*Align v2*). This algorithm explicitly calculates the probability that a specific grapheme is realised as a null phoneme, given the previous non-null phonemic realisation of the preceding grapheme or graphemes, and provides a significant performance improvement.

Table 1: *Phone and word alignment accuracy obtained on the full 20,008-word NETtalk corpus*

Database	Type	Phone	Word
NETtalk-original	Iterative Viterbi [7]	93.2	83.7
NETtalk-original	Align v1	96.5	87.3
NETtalk-rewritten	Align v2	98.7	95.4

The effect of the improvement in alignment accuracy on rule extraction accuracy is depicted in Fig. 1. The *Align v1* and *Align v2* algorithms are used prior to *DEC-min* rule extraction on the *Fonilex* database, and grapheme-to-phoneme prediction accuracy measures against a 5000-word test set.

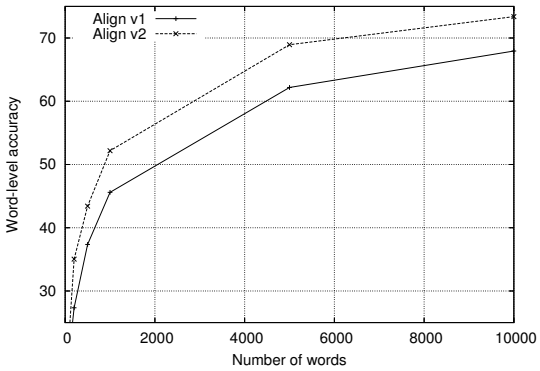

Figure 1: *Effect of different alignment algorithms on word-level pronunciation prediction accuracy of DEC-min*

4.2. G2P rule extraction

4.2.1. Accuracy during bootstrapping

The accuracy of the different DEC variations tested differ based on the stage of bootstrapping. In this section we describe the trends observed during the critical first 5000 words of bootstrapping. The bootstrapping process is simulated by testing the algorithms on an existing dictionary (the Flemish *Fonilex* dictionary). In all experiments the size of the maximum context allowed when extracting rules is not restricted and the same word training order (even selection from corpus) is used. A 5000 word test set is used. Figures 2 and 3 compare the performance [1] of different DEC variations.

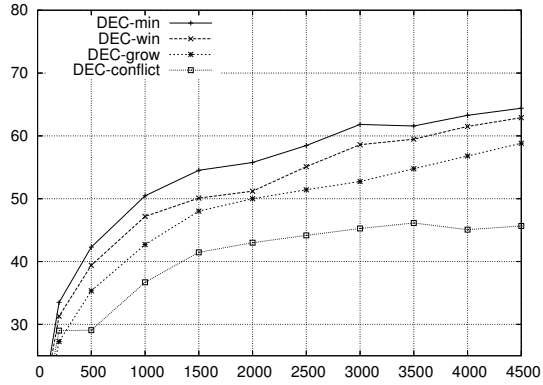

Figure 2: *Word accuracy of different DEC variations*

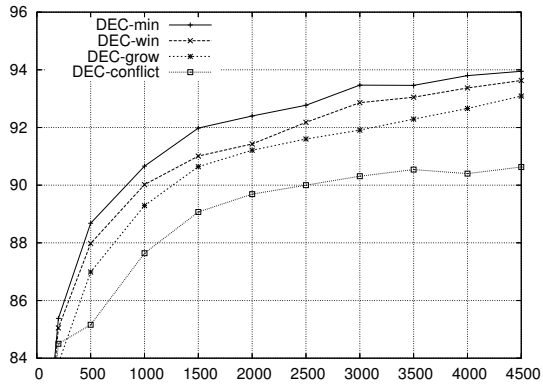

Figure 3: *Phoneme accuracy of different DEC variations*

The lowest accuracy is observed if DEC is not allowed to grow a context beyond word boundaries and conflicting words are ignored (*DEC-conflict*). In order to remove this boundary effect, the version of DEC (*DEC-grow*) that was implemented allows a context to grow towards the opposite side if a word boundary is encountered. Three shifted window versions of DEC are implemented: extracting the first valid rule (*DEC-win*), extracting the maximum number of valid rules (*DEC-max*), and pruning this system to obtain the minimum number of rules that still provide full coverage for the training corpus (*DEC-min*). Since *DEC-max* (not shown) over generates rules, performance is lower than in the case of *DEC-win* but once

[1] Word accuracy measures the number of words that are completely correct, while phoneme accuracy is measured as the number of correct grapheme-to-phoneme mappings minus deletions, divided by the total number of grapheme-to-phoneme mappings considered.

pruned back (*DEC-min*), performance again increases. For the training set sizes analysed, the pruned, shifted window version of DEC (*DEC-min*) provides a consistent improvement in accuracy, as illustrated in Figures 2 and 3.

Asymptotic performance is only achieved for larger training sets, yet the system performs well for the small data sets that are available during the initial stages of the dictionary creation process: after a 1000 words have been observed, a verifier would be correcting less than 1 out of 10 phonemes.

While our goal was to find an algorithm that performs well during the initial stages of bootstrapping, accuracy on larger corpora compare with previously published results. On the *NETtalk* corpus we achieve an accuracy similar to Torkkola's DEC [4] and Hakkinen's trie and decision-tree implementations [8]; and slightly worse than Yvon's chunk-based implementation [9]. For larger corpora, the relative performance of the DEC variations studied becomes less distinct; for example, at 40,000 training words from the *Fonilex* corpus, *DEC-grow* obtains a phoneme prediction accuracy of 97.10% compared to the 97.18% phoneme prediction accuracy of *DEC-min*.

4.2.2. Rule set analysis

While algorithmic analysis is possible using pre-existing dictionaries, the aim of the bootstrapping process is the development of new dictionaries. The system described here was used to generate a bootstrapped dictionary for Afrikaans, as described in [2]. Using *DEC-min*, a phoneme accuracy of 93.1% and a word accuracy of 68.57% were obtained after training on 2,580-words (and testing on the remainder). An analysis of the rules obtained from the bootstrapped Afrikaans dictionary is shown in Table 2. The numbers of rules of each size (the size of the context that specifies the rule), as extracted from different-sized training dictionaries, using *DEC-grow*, *DEC-max* and *DEC-min* are compared. Note that *DEC-max* tends to extract more rules than *DEC-grow* but that these rules tend to be shorter, as expected. *DEC-min* reduces the number of rules significantly (without compromising on performance.)

Table 2: *Number of rules of each size generated by different rule extraction techniques*

	DEC grow			DEC max			DEC min		
	100	1,000	2,580	100	1,000	2,580	100	1,000	2,580
1	26	26	26	26	26	26	26	26	26
2	35	50	46	74	89	85	53	75	70
3	63	364	526	74	589	922	32	310	507
4	6	200	626	3	274	1078	2	111	414
5	0	85	305	0	8	41	0	4	22
6	1	7	39	0	0	0	0	0	0
7	0	0	1	0	0	0	0	0	0
	131	732	1569	177	986	2152	113	526	1039

5. Conclusion

By carefully optimizing a number of factors, such as selection of the context window, alignment algorithm, and the order of word validation, it is possible to achieve significant gains in the efficiency of bootstrapping for the generation of pronunciation dictionaries. Our initial experiments, as reported in [2], confirm that this enables dictionary developers with limited formal linguistic training to create accurate pronunciation dictionaries efficiently.

The current system provides an effective platform for the development of such dictionaries, but further gains are likely to arise from future improvements. Most importantly, the current algorithm treats each grapheme as a unique entity, whereas similar graphemes (e.g. those that code for vowels) tend to behave similarly. A class-based algorithm should therefore be able to improve learning speed significantly. Further work relates to exploring the ways in which the algorithmic requirements change for different phases of the bootstrapping process.

6. Acknowledgements

This work was supported by the CSIR *Information Society Technologies Centre*, South Africa, as well as the *Local Language Speech Technology Initiative* (LLSTI).

7. References

[1] Marelie Davel and Etienne Barnard, "Bootstrapping for language resource generation," in *Proceedings of the 14th Symposium of the Pattern Recognition Association of South Africa*, South Africa, 2003, pp. 97–100.

[2] Marelie Davel and Etienne Barnard, "The efficient creation of pronunication dictionaries: Human factors in bootstrapping," in *Proceedings of the ICSLP*, Jeju, Korea, 2004.

[3] T. Schultz and A. Waibel, "Language-independent and language-adaptive acoustic modeling for speech recognition," *Speech Communication*, vol. 35, pp. 31–51, Aug. 2001.

[4] K. Torkkola, "An efficient way to learn english grapheme-to-phoneme rules automatically," in *Proceedings of the International Conference on Acoustics and Speech Signal Processing (ICASSP)*, Minneapolis, 1993, vol. 2, pp. 199–202.

[5] T.J. Sejnowski and C.R. Rosenberg, "Parallel networks that learn to pronounce english text," *Complex Systems*, pp. 145–168, 1987.

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

[7] Ove Andersen, Roland Kuhn, Ariane Lazarides, Paul Dalsgaard, Jurgen Haas, and Elmar Noth, "Comparison of two tree-structured approaches for grapheme-to-phoneme conversion," in *Proceedings of the ICSLP*, Delaware, 1996, pp. 1808–11.

[8] J. Hakkinen, J. Suontausta, S. Riis, and K Jensen, "Accessing text-to-phoneme mapping strategies in speaker independent isolated word recognition," *Speech Communication*, vol. 41, pp. 455–467, 2003.

[9] Francois Yvon, "Grapheme-to-phoneme conversion using multiple unbounded overlapping chunks," in *Proceedings of Conference on New Methods in Natural Language Processing (NeMLaP)*, Ankara, Turkey, 1996, pp. 218–228.