

# A default-and-refinement approach to pronunciation prediction

*M. Davel and E. Barnard*

Human Language Technologies Research Group  
CSIR / University of Pretoria, Pretoria, 0001

mdavel@csir.co.za, ebarnard@up.ac.za

## Abstract

We define a novel g-to-p prediction algorithm that utilises the concept of a ‘default phoneme’: a grapheme which is realised as a specific phoneme significantly more often than as any other phoneme. We find that this approach results in an algorithm that performs well across a range from very small to large data sets. We evaluate the algorithm on two benchmarked databases (*Fonilex* and *NETalk*) and find highly competitive performance in asymptotic accuracy, initial learning speed, and model compactness.

## 1. Introduction and background

The ability to predict the pronunciation of a written word accurately is an important sub-component within many speech processing systems. This task is typically accomplished through explicit pronunciation dictionaries or grapheme-to-phoneme (g-to-p) rule sets. Both of these resources can be difficult to obtain and resource-intensive to develop when creating speech technology in a new language. The dictionary creation process can be made more efficient through the use of g-to-p rule-based bootstrapping [1]: an audio-enabled process whereby g-to-p rules are extracted from the current dictionary (however small) and used to predict additional entries. Predicted entries are subsequently presented to and verified by a human verifier and the process is repeated until a dictionary of sufficient size is obtained.

The efficiency of this process is influenced by the efficiency of the g-to-p rule extraction mechanism. G-to-p rules are typically used to generalise from existing pronunciation dictionaries when handling out-of-vocabulary words; and to compress information when requiring a pronunciation model in a memory-constrained environment. Such applications require a balance between the need for small rule sets, fast computation and optimal accuracy. During bootstrapping, a key requirement is learning speed, i.e. we are specifically interested in obtaining a high level of generalisation given a very small training set.

Various approaches to g-to-p rule extraction exist. When considering data-driven approaches, formalisms that have been used successfully for this task include

neural networks [2], decision trees [3], pronunciation-by-analogy models [4]; various instance-based learning algorithms such as Dynamically Expanding Context (DEC) [5] and IB1-IG [6]; and the combination of methods and additional information sources through meta-classifiers [7].

The results when applying appropriate versions of the different formalisms mentioned above are typically comparable, with variations in performance for specific tasks. Languages with irregular spelling systems such as English and French perform well within analogy-based frameworks, while instance-based learning is well suited to languages with a more regular orthography, such as Italian or Dutch. Results for different algorithms are compared in greater detail in section 3.

In this paper, we describe a novel approach to the g-to-p rule extraction problem (section 2) and evaluate the new approach in comparison with benchmarked algorithms (section 3). We demonstrate the learning curve and asymptotic behaviour of the algorithm, and discuss the implications of our results in the concluding section.

## 2. Approach

Grapheme-to-phoneme prediction algorithms rely on the connection between the spoken and written form of a language. The more modern the language, the stronger this connection, the more regular the spelling system of the language, and the stronger the concept of a ‘default phoneme’: a grapheme that is realised as a single phoneme significantly more often than as any other phoneme. Figure 1 and Figure 2 illustrate this phenomenon for Flemish. When counting the number of times a specific grapheme is realised as a specific phoneme, most graphemes follow the trend depicted in Figure 1. For the ‘most conflicted’ phones ( $h, j, n, u$ ), the trend is less strong, but also clearly discernable, as depicted in Figure 2. Similar trends are observable for languages with less regular spelling systems, with a larger proportion of graphemes of these languages displaying the behaviour depicted in Figure 2.

We use this information to define an algorithm that uses greedy search to find the most general rule at any given stage of the rule extraction process. When applying

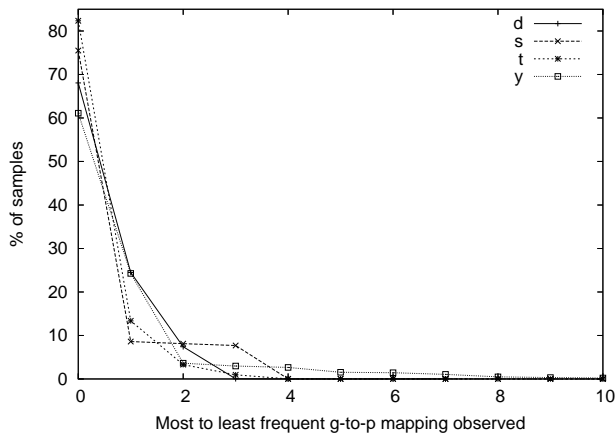


Figure 1: *Default phone behaviour of graphemes d,s,t and j in Flemish. Only the first 10 phonemic candidates are displayed.*

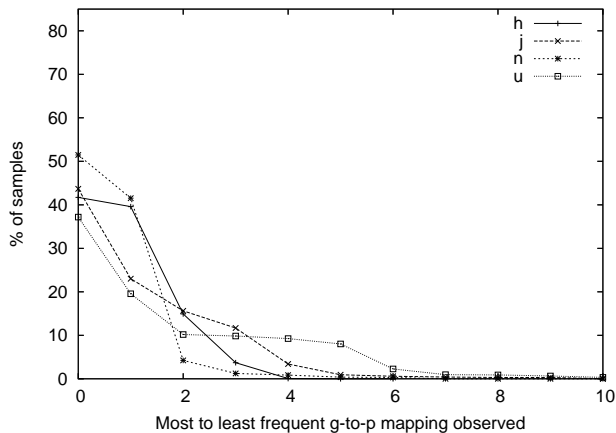


Figure 2: *Conflict phone behaviour of graphemes h,j,n,u in Flemish. Only the first 10 phonemic candidates are displayed.*

these rules during g-to-p prediction, we use the reverse rule extraction order. Explicitly ordering the rules provides flexibility during rule extraction, and ensures that the default pattern acts as a back-off for the next rule defined.

The framework we use is similar to that of most multi-level rewrite rule sets. Each g-to-p rule consists of a pattern:

$$(\textit{left context} - g - \textit{right context}) \rightarrow p \quad (1)$$

The pronunciation for a word is generated one grapheme at a time. Each grapheme and its left and right context as found in the target word are compared with each rule in the ordered rule set; and the first matching rule is applied. Interestingly, while the rule application order of DEC (the algorithm closest to ours) is ordered by context size (largest rule first), our reverse rule extraction order automatically reverts to context size ordering in the case of DEC-based rule extraction.

Prior to rule extraction, grapheme-to-phoneme alignment is performed according to the Viterbi alignment process described in [8]. During rule extraction, the rule set for each grapheme is extracted separately. For any specific grapheme, applicable words are split into two sets based on whether the current rule set predicts the pronunciation of that grapheme accurately (*Completed* words) or not (*New* words). Definition of a new rule moves words from the *New* to the *Completed* set. Any words that are currently in the *Completed* set and conflict with the new rule, are moved back to the *New* set. The rule that will cause the most net words to be moved from the *New* to the *Completed* set is chosen first, irrespective of context size. Conflict is only resolved in the *Completed* set; new rules are allowed to conflict with words still in *New*. This ensures that the rule set is built for the default pattern(s) first.

Table 1: *The relationship between a word and its sub-pattern during rule extraction for grapheme e.*

Word	test
Word pattern	#t-e-st# $\rightarrow$ E
Sub-patterns	-e- $\rightarrow$ E, -e-s $\rightarrow$ E, t-e- $\rightarrow$ E, t-e-s $\rightarrow$ E t-e-st $\rightarrow$ E, #t-e-s $\rightarrow$ E, -e-st# $\rightarrow$ E #t-e-st $\rightarrow$ E, t-e-st# $\rightarrow$ E, #t-e-st# $\rightarrow$ E

In order to implement this algorithm in a computationally efficient way, the following techniques are used:

- The large word sets are used to keep track of status, but further manipulation utilises two sets of sub-patterns: the *Possible* sub-patterns, indicating all possible new rules, and consisting of all the sub-patterns of each word pattern in *New*, excluding all for which the left-hand side is an existing rule; and the *Caught* set of sub-patterns, indicating all the sub-patterns explicitly or implicitly covered by the current rule set. The relationship between a word and its sub-patterns is illustrated in Table 1. Hashes denote word boundaries.
- Words are pre-processed and the word patterns relevant to a single grapheme extracted and written to file. All further manipulation considers a single grapheme (and set of word patterns) at a time.
- The context size of the sub-patterns considered is grown systematically: only sub-patterns up to size  $max + win$  are evaluated, where  $max$  indicates the current largest rule, and  $win$  is defined to ensure that any larger contexts that may be applicable are considered, without requiring all patterns to be searched.
- Both the *Possible* and *Caught* sets of sub-patterns count the number of times a matching word pattern is observed in the relevant word sets. The

next rule is chosen by finding the pattern for which the matching count in *Possible* minus the conflicting count in *Caught* is highest. (The conflicting count is the number of times a matching left-hand pattern is observed with a conflicting right-hand phoneme.)

- Whenever a sub-pattern in *Possible* or *Caught* reaches a count of zero, the sub-pattern is deleted and not considered further, unless re-added based on an inter-set move of a related word.

### 3. Evaluation

We use two corpora to evaluate the performance of the algorithm:

- *Fonilex*, a publicly available pronunciation dictionary of Dutch words as spoken in the Flemish part of Belgium. We use the exact pre-aligned 173,874-word dictionary as used in [7].
- *NETtalk*, a publicly available 20,008-word English pronunciation dictionary [9]. Hand-crafted grapheme-to-phoneme alignments are included in the dictionary.

In all experiments we perform 3-fold cross-validation based on a 90% training and 10% test set. We report on phoneme correctness<sup>1</sup>, phoneme accuracy<sup>2</sup> and word accuracy<sup>3</sup>. Where there is uncertainty with regard to the measure used in the benchmark result, word accuracy provides the least ambiguous comparison.

#### 3.1. Learning curve

As we aim to use this algorithm for the bootstrapping of pronunciation dictionaries, we are interested in the performance of the algorithm with very small training sets. We therefore evaluate word and phone accuracy for different training dictionaries of sizes smaller than 3,000 words, using subsets from *Fonilex*. Figures 3 and 4 demonstrates the learning curve for the algorithm *Default&Refine* in comparison with *DEC*. Each rule set is evaluated against the full 17,387-word test set.

The algorithm performs well, achieving 50% word accuracy ( $\pm 90\%$  phoneme accuracy) at 600 words. *DEC* requires an additional 1100 words before the same level of accuracy is reached. Since the correction of incorrectly predicted phonemes is the most labour-intensive aspect of bootstrapping pronunciation dictionaries, this represents a significant improvement to the process.

<sup>1</sup>Number of phonemes identified correctly

<sup>2</sup>Number of correct phonemes - insertions, divided by the total number of phonemes in correct pronunciation

<sup>3</sup>Number of words completely correct

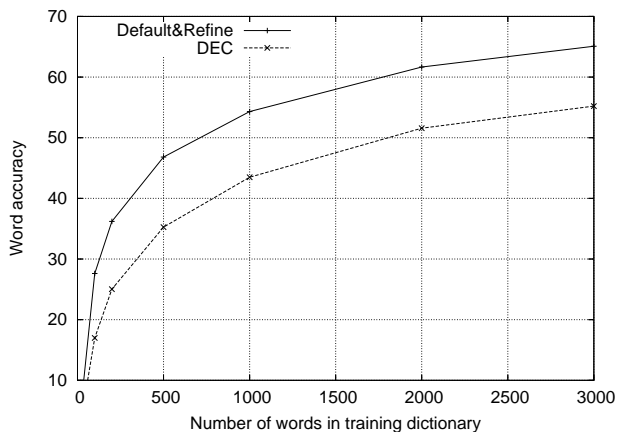


Figure 3: *Word accuracy during initial 3000 training words*

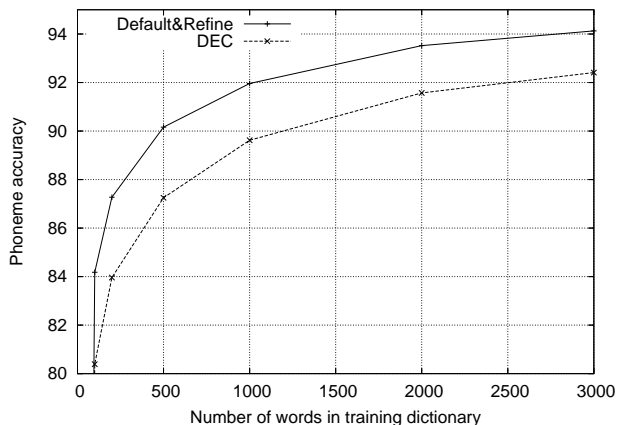


Figure 4: *Phoneme accuracy during initial 3000 training words*

#### 3.2. Asymptotic performance

A steep initial learning curve does not imply that the algorithm will continue to perform well as the training data set increases. In order to evaluate asymptotic behaviour, we evaluate the accuracy of the algorithm when trained on the full *Fonilex* training set, and use the results reported by Hoste [7] as benchmark. Hoste compared various g-to-p approaches using the *Fonilex* corpus, including:

- Instance learning based IB1-IG as a single classifier.
- Cascading two separate IB1-IG classifiers (one trained on *Fonilex* and one on *Celex* - a Dutch variant corpus).
- Combining these classifiers using various meta classifiers including C5.0 (decision tree learning) IB1-IG, IGTREE (an optimised version of IB1-IG) and MACCENT (a maximum entropy-based algorithm).

- Using IB1-IG to create a meta-meta-classifier trained on the results of the previous meta-classifiers.

The different results obtained using a 156,487-word training subset of *Fonilex* are compared in Table 2. The *Default&Refine* single classifier performs better than the single classifier and the meta-classifier variations reported on; and achieves comparable accuracy to the meta-meta-classifier, without utilising the additional *Celex* data.

Table 2: Accuracy comparison for different algorithms using the *Fonilex* corpus

	Word accuracy	Phoneme accuracy	Phoneme correct
Single			
<i>IB1-IG</i> [7]	86.37	-	98.18
<i>DEC-grow</i> [8]	89.47	98.48	98.69
<i>DEC-min</i> [8]	90.44	98.53	98.75
<i>Default&amp;Refine</i>	92.07	98.79	98.89
Meta			
<i>MACCENT</i> [7]	87.27	-	98.28
<i>C5.0</i> [7]	88.41	-	98.48
<i>IGTREE</i> [7]	91.33	-	98.85
<i>IB1-IG</i> [7]	91.55	-	98.89
Meta-Meta			
<i>IB1-IG</i> [7]	92.25	-	98.99

With 3-fold cross-validation we observe a standard deviation in phone accuracy of 0.09 for *Default&Refine*.

### 3.3. Less regular spelling systems

We were interested whether the algorithm would fail for a language with a less regular spelling system, and evaluated the asymptotic performance of the algorithm on the *NETtalk* corpus, using results obtained by Anderson [3], Torkkola [5] and Yvon [4] as benchmarks. The algorithm performed surprisingly well, as shown in Table 3. The *SMPA* algorithm employs pronunciation by analogy, and is not suitable for training on small data sets.

Table 3: Accuracy comparison for different algorithms using the *NETtalk* corpus

	Word accuracy	Phoneme accuracy	Phoneme correct
<i>Trie</i> [3]	51.7	-	89.8
<i>Decision Tree</i> [3]	53.0	-	89.9
<i>DEC</i> [5]	-	-	90.8
<i>DEC</i> [4]	56.67	-	92.21
<i>Default&amp;Refine</i>	58.45	90.39	91.31
<i>SMPA</i> [4]	63.96	-	93.19

With 3-fold cross-validation we observe a standard deviation in phone accuracy of 0.13 for *Default&Refine*. (In this table, the phoneme correctness reported in [4] for DEC seems anomalously high, in relation to our own experiments, those obtained in [5], and the reported word accuracy.)

### 3.4. Size of the rule set

While memory usage and the size of the rule set is typically not a concern during g-to-p bootstrapping, the size of the rule set does affect the speed of the g-to-p prediction algorithm. We therefore evaluate the growth in rule set size at different stages of the learning process.

In Table 4 we compare the number and size of rules for this algorithm with the rule set obtained via *DEC* and find that the rule set size is significantly smaller for *Default&Refine*. The latter algorithm provides both a more accurate and more compact prediction model.

Table 4: Number and size of rules for *DEC* and *Default&Refine* when trained on training dictionaries of various sizes. The first column lists the number of graphemes in a rule, and subsequent columns give the number of rules of that size.

	DEC			Default&Refine		
	1000	10000	156486	1000	10000	156486
1	26	26	26	26	26	26
2	94	107	132	151	197	227
3	539	1194	1429	314	775	1221
4	324	2032	3750	190	1394	4430
5	140	1882	8796	18	603	5293
6	33	740	7938	1	142	2510
7	4	276	7002	1	18	816
8	1	72	4218	-	6	288
9	-	25	2683	-	-	114
10	-	3	1574	-	-	74
11+	-	3	3165	-	-	71
	1161	6360	39270	701	3161	15070

## 4. Conclusion

The concept of a default phone suggests an interesting algorithm for g-to-p prediction, based on the extraction of a cascade of increasingly more specialized rules. This algorithm has a number of attractive properties, including rapid learning, good asymptotic accuracy, and the production of compact rule sets. We are integrating it into our bootstrapping system for dictionary creation [1], where it will be of value in our quest to develop linguistic resources for the languages of the developing world.

## 5. Acknowledgements

This work was supported by the CSIR *Information Society Technologies Centre*. We would like to thank Piet Mertens for providing us with access to the *Fonilex* database, and Veronique Hoste and Walter Daelemans for providing us with access to their experimental *Fonilex* data.

## 6. References

- [1] M. Davel and E. 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] T.J. Sejnowski and C.R. Rosenberg, “Parallel networks that learn to pronounce english text,” *Complex systems*, vol. 1, pp. 145–168, 1987.
- [3] O. Andersen, R. Kuhn, A. Lazarides, P. Dalsgaard, J. Haas, and E. Noth, “Comparison of two tree-structured approaches for grapheme-to-phoneme conversion.,” in *Proceedings of the ICSLP*, Philadelphia, 1996, vol. 3, pp. 1700–1703.
- [4] F. 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.
- [5] 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.
- [6] Walter Daelemans, Antal van den Bosch, and Jakub Zavrel, “Forgetting exceptions is harmful in language learning,” *Machine Learning*, vol. 34, no. 1-3, pp. 11–41, 1999.
- [7] Erik Tjong Kim Sang Veronique Hoste, Walter Daelemans and Steven Gillis, “Meta-learning for phonemic annotation of corpora,” in *Proceedings of the ICML-2000*, Stanford University, USA, 2000.
- [8] M. Davel and E. Barnard, “The efficient creation of pronunciation dictionaries: Machine learning factors in bootstrapping,” in *Proceedings of the ICSLP*, Jeju, Korea, 2004.
- [9] T.J. Sejnowski and C.R. Rosenberg, “Parallel networks that learn to pronounce english text,” *Complex Systems*, pp. 145–168, 1987.