

Pronunciation prediction with Default&Refine

Marelie Davel*, Etienne Barnard

Human Language Technologies Research Group, CSIR Meraka Institute, P.O. Box 395, Pretoria 0001, South Africa

Received 20 December 2006; received in revised form 9 January 2008; accepted 10 January 2008

Available online 20 January 2008

Abstract

The Default&Refine algorithm is a new rule-based learning algorithm that was developed as an accurate and efficient pronunciation prediction mechanism for speech processing systems. The algorithm exhibits a number of attractive properties including rapid generalisation from small training sets, good asymptotic accuracy, robustness to noise in the training data, and the production of compact rule sets. We describe the Default&Refine algorithm in detail and demonstrate its performance on two benchmarked pronunciation databases (the English OALD and Flemish FONILEX pronunciation dictionaries) as well as a newly-developed Afrikaans pronunciation dictionary. We find that the algorithm learns more efficiently (achieves higher accuracy on smaller data sets) than any of the alternative pronunciation prediction algorithms considered. In addition, we demonstrate the ability of the algorithm to generate an arbitrarily small rule set in such a way that the trade-off between rule set size and accuracy is well controlled. A conceptual comparison with alternative algorithms (including Dynamically Expanding Context, Transformation-Based Learning and Pronunciation by Analogy) clarifies the competitive performance obtained with Default&Refine.

© 2008 Elsevier Ltd. All rights reserved.

Keywords: Pronunciation prediction; Grapheme-to-phoneme prediction; Default&Refine; Rule extraction

1. Introduction and background

Predicting the pronunciation of a written word is an important sub-task within many speech processing systems. This task is typically accomplished either through explicit pronunciation lexicons or through grapheme-to-phoneme rule sets. Both of these resources can be difficult to obtain and resource-intensive to develop when creating speech technology in a resource-scarce language. The pronunciation lexicon creation process can be made more efficient through the use of grapheme-to-phoneme rule-based bootstrapping: an audio-enabled process whereby grapheme-to-phoneme rules are extracted from the current lexicon, however small, and used to predict additional entries (Davel and Barnard, 2003; Maskey et al., 2004). Predicted entries are subsequently presented to and verified by a human verifier, and this process repeated until a pronunciation lexicon of sufficient size is obtained. The efficiency of this bootstrapping process is strongly influenced by the efficiency of the grapheme-to-phoneme rule extraction mechanism used.

* Corresponding author. Tel.: +27 12 841 2466; fax: +27 12 841 4829.

E-mail addresses: mdavel@csir.co.za (M. Davel), ebarnard@csir.co.za (E. Barnard).

Apart from its application during bootstrapping of pronunciation lexicons, grapheme-to-phoneme rule extraction can be used to generalise from existing pronunciation lexicons when handling out-of-vocabulary words and to compress information for use in a memory-constrained environment. These applications require a balance between the need for small rule sets, fast computation and optimal predictive accuracy. During bootstrapping, a key requirement is initial learning speed, i.e. apart from the typical requirements mentioned above we are also interested in obtaining a high level of generalisation given a very small training set.

Various data-driven approaches to grapheme-to-phoneme rule extraction exist. Formalisms that have been used successfully for this task include neural networks (Sejnowski and Rosenberg, 1987), rule induction techniques (van Coile, 1990), Dynamically Expanding Context (DEC) (Torkkola, 1993), decision trees (Andersen et al., 1996; Black et al., 1998), k -nearest neighbour classifiers such as IB1-IG (Daelemans et al., 1999), pronunciation by analogy (PbA) approaches (Dedina and Nusbaum, 1991; Yvon, 1996; Marchand and Damper, 2000) and the combination of methods and additional information sources through meta-classifiers (Hoste et al., 2000).

A comparative analysis of the performance of different approaches for English pronunciation prediction in Damper et al. (1999) found PbA to outperform both a neural network and the IB1-IG nearest neighbour classifier. Indeed the highest reported accuracies for English on various databases have been achieved with variations of PbA (Yvon, 1996; Damper et al., 1999, 2005). Interestingly, it seems that languages with irregular spelling systems such as English perform well within analogy-based frameworks, while for languages with more regular (or shallow) spelling systems, such as Italian, higher accuracies have been reported using DEC than the analogy-based SMPA (Yvon, 1996).

In prior work (Davel and Barnard, 2004a; Davel, 2005) the concept of a default-and-refinement approach to pronunciation prediction was described and initial results obtained using this approach were discussed. In this paper we describe the Default&Refine algorithm in detail and perform a thorough evaluation of the performance of the algorithm with regard to asymptotic behaviour, learning efficiency and model compactness. When evaluating asymptotic accuracy, we use 10-fold cross-validation to benchmark the Default&Refine algorithm against existing pronunciation prediction algorithms using the publicly available Flemish FONILEX (Mertens and Vercammen, 1998) and English OALD (The Oxford Advanced Learners Dictionary) (Mitten, 1992) pronunciation dictionaries, as well as a newly-developed Afrikaans database. We report on comparative results obtained using decision tree learning, IB1-IG, DEC and PbA.

We find that the Default&Refine algorithm learns more efficiently (achieves higher accuracy on smaller data sets) than any of the other pronunciation prediction algorithms considered. We also find that the algorithm produces good asymptotic accuracy, outperforming most algorithms compared with. (Only PbA achieves higher asymptotic accuracy when evaluated on the English OALD corpus.) We demonstrate both the compactness of the standard rule set extracted, as well as the ability of the algorithm to generate an arbitrarily small rule set in such a way that the trade-off between rule set size and accuracy is well controlled.

The paper is structured as follows: in Section 2, we describe the Default&Refine algorithm and discuss the rationale for this approach. In Section 3, we evaluate the performance of the algorithm and report on the results obtained. In Section 4, we discuss the conceptual differences when comparing the new algorithm to alternative grapheme-to-phoneme prediction algorithms including DEC, Transformation-Based Learning (TBL), decision trees, van Coile's rule induction technique and pronunciation by analogy. We conclude the paper by discussing the implications of our results in Section 5.

2. The Default&Refine algorithm

We first discuss the rationale for approaching the pronunciation prediction task in the way we did, before discussing the algorithm in more detail. We differentiate between the data preparation phase, the learning phase (the actual heart of the algorithm) and the classification phase. We end this section by mentioning some practical considerations to be considered during implementation.

2.1. Rationale for approach

Grapheme-to-phoneme prediction algorithms rely on the connection between the spoken and written form of a language. The more regular the spelling system of the language, the stronger this connection, and the

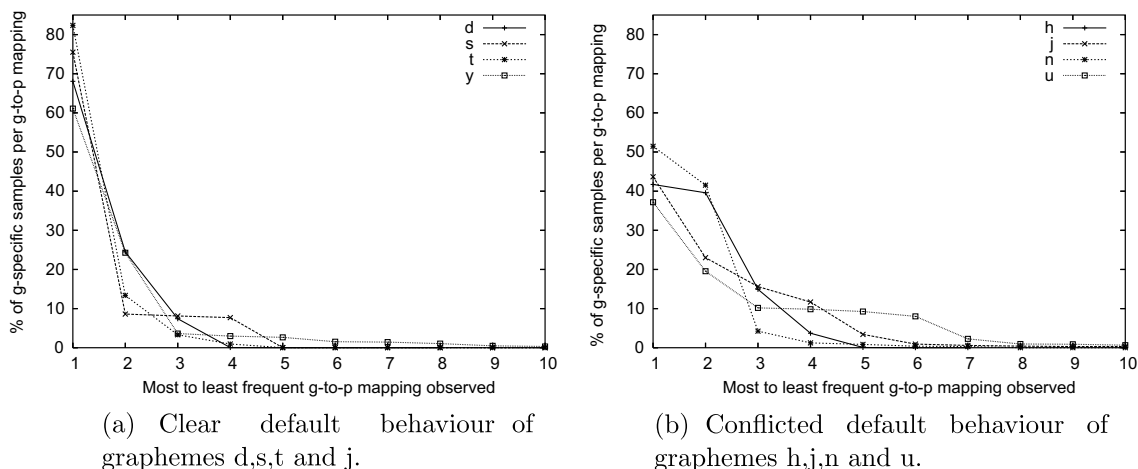


Fig. 1. Demonstrating clear and conflicted default behaviour of Flemish graphemes (according to the FONILEX corpus). Only the first 10 phonemic candidates are displayed.

stronger the concept of a ‘default phoneme’: a grapheme¹ that is realised as a single phoneme significantly more often than any other phoneme. Fig. 1a and b illustrates this phenomenon for Flemish. To draw these figures, we align words and their pronunciations so that every grapheme is associated with a particular phoneme (details are provided below).

When counting the number of times a given grapheme is realised as a specific phoneme, most graphemes follow a similar trend to that depicted in Fig. 1a: the Flemish grapheme *y*, for example, is realised as a single phoneme more than 60% of the time, with the next two phonemic candidates occurring only 24% and 4% of the time, respectively. We refer to this trend as ‘clear default behaviour’. For graphemes that exhibit ‘conflicted default behaviour’, such as *h*, the trend is less strong, but also clearly discernible, as can be seen in Fig. 1b. Similar trends are observable for languages with less regular spelling systems than Flemish, with a larger proportion of graphemes of these languages displaying conflicted default behaviour.

Given the strong tendency towards default behaviour that is observed, it seems probable that the pronunciation prediction task would be well modelled by a rule set that captures hierarchical default behaviour: the last rule associated with a specific grapheme defines the default behaviour for that grapheme, and acts as a back-off rule to the second-last (more refined) rule, which would capture deviations from the default behaviour. The second-last rule would then act as back-off to the third-last rule, and so forth. In order to obtain such a rule set, we use a greedy search algorithm to find the most general rule predicting the phonemic outcome of a single grapheme at any given stage of the rule extraction process, as described in more detail in Section 2.3.

Each extracted rule consists of a pattern

$$g_{\text{left}}-g-g_{\text{right}} \rightarrow p \quad (1)$$

where ‘*g*’ indicates the grapheme being considered, ‘*g*_{left}’ and ‘*g*_{right}’ indicate the graphemic left and right contexts of the rule, and ‘*p*’ the specific phonemic realisation of grapheme *g*.

When predicting the pronunciation of a word, rules are applied one grapheme at a time. Each grapheme and its left and right contexts as found in the target word are compared with each rule in the rule set and the first matching rule applied. During rule extraction, rules are explicitly ordered according to the reverse rule extraction order. (The rule extracted first is matched last.) Explicitly ordering the rules provides flexibility during rule extraction, and ensures that the default pattern acts as a back-off mechanism for the more specialized rules.

¹ In this paper we use grapheme to mean strictly ‘letter’, rather than the alternative meaning of ‘functional spelling unit’ (Henderson, 1985).

As rule extraction progresses, the number of training instances to process become fewer and fewer. At each stage of rule extraction we try to determine what the *default* rule would be for the current state of the dictionary, and then *refine* the rule set further by repeating the process with a smaller set of unprocessed instances.

In the following sections, we describe how this general idea is realised as the Default&Refine algorithm. Specifically, we address data preparation and algorithm initialisation (Section 2.2), learning (Section 2.3), classification (Section 2.4) and practical considerations (Section 2.5). Although our primary interest is in pronunciation prediction, we do present a somewhat more general classification formalism which we have found useful in other applications.

2.2. Data preparation and algorithm initialisation

Data preparation and algorithm initialisation consist of four steps: data alignment, removal of inconsistent training instances, definition of appropriate feature generation rules and selection of a tie resolution strategy.

2.2.1. Data alignment

Prior to rule extraction, the two-level graphemic and phonemic data are aligned on a per data item basis. During grapheme-to-phoneme alignment each grapheme in a word is associated with a specific phoneme in the pronunciation of that word, with graphemic or phonemic nulls being inserted as required. A graphemic null ϕ_g is inserted where one grapheme is realised as more than one phoneme, e.g. *tax* ϕ_g *i* realised as /T AE K S IY/ (using ARPAbet notation), while a phonemic null ϕ_p is inserted where more than one grapheme is realised as a single phoneme, e.g. *when* realised as /WH ϕ_p EH N/.

Grapheme-to-phoneme mappings can be obtained using iterative Viterbi alignment (Viterbi, 1967) with estimated grapheme-to-phoneme probabilities used to specify path costs, where initial probabilities are obtained from words and pronunciations of equal length. (While words and pronunciations of equal length are not necessarily correctly aligned, the obtained probabilities provide a good initialisation for the iterative Viterbi process.) As the quality of alignments influences the predictive accuracy of extracted rules, the alignment process can be further optimised by utilising the phonemic character of nulls during alignment (keeping separate grapheme-to-phoneme probabilities for null phonemes to the right of different non-null phonemes) and pre-processing graphemic nulls (Davel and Barnard, 2004b).

When graphemic nulls are pre-processed, alignment consists of two separate phases: (1) first identifying possible graphemic null generator contexts based on words with longer orthography than pronunciation, and (2) applying all these null generator contexts – whether accurate or not – prior to insertion of phonemic nulls during a second separate alignment phase. If an unnecessary graphemic null was inserted, this is typically automatically corrected during the second alignment phase through the addition of another phonemic null inserted to match the unnecessary graphemic null. This technique has the additional advantage that the graphemic null generator contexts can similarly be applied during the prediction of an unknown test work. (For a test word it may otherwise be difficult to add the appropriate graphemic nulls prior to prediction, as its pronunciation and therefore word-pronunciation alignment is unknown.)

2.2.2. Removing inconsistent training instances

The standard rule extraction algorithm assumes that all training instances are unique, and that no conflicting outcomes occur. For the pronunciation modelling task this translates to the assumption that pronunciation variants do not occur, i.e. that no word has more than one possible pronunciation. Pronunciation variants must either be removed from the training data, or incorporated in the training data through pre-processing, as described by Davel and Barnard (2006). (This is an effective techniques for encoding the type of variants typically occurring in training dictionaries.) While this state of affairs is not ideal, it is typical of many data-driven grapheme-to-phoneme rule extraction techniques.

2.2.3. Defining the feature template generation rules

The most important step during data preparation is the definition of the vector of available features to be used during the learning phase. In the general case, feature vectors are generated according to feature templates that, in turn, are generated from feature template generation rules. For pronunciation prediction, only

a small subset of the possible feature templates is required. The results reported on in Section 3 use two possible feature template generation rules. The first is defined as

$$\text{template}(j_k) \rightarrow x_{i-k} \cdots x_{i-1} \wedge x_i \wedge x_{i+1} \cdots x_{i+m}, \quad i \geq 0, \quad k + m = j \quad (2)$$

where x_i is the focal grapheme, $x_{i-k}, \dots, x_{i-2}, x_{i-1}$ the k -length left graphemic context of x_i , and $x_{i+1}, x_{i+2}, \dots, x_m$ the m -length right graphemic context of x_i . The second template generation rule, defined as

$$\begin{aligned} \text{template}(j_k) &\rightarrow f_{i-k} \cdots f_{i-1} \wedge f_i \wedge f_{i+1} \cdots f_{i+m}, \quad i \geq 0, \quad k + m = j \\ f_i &= x_i \quad \text{or} \quad S_n \quad \text{where} \quad x_i \in S_n, \quad n = 1, \dots, 3 \\ S_1 &= \{a, e, i, o, u, y\} \\ S_2 &= \{b, c, d, f, g, h, j, k, l, m, n, p, q, r, s, t, v, w, x, z,\} \\ S_3 &= S_1 \cup S_2 \end{aligned} \quad (3)$$

introduces the concept of graphemic groups S_n , where each element of the feature vector can either refer to a specific grapheme or to a pre-defined set of graphemes S_n . While we have constrained the S_n to three simple predefined groups consisting of vowels, consonants and the space delimiter (#), much additional flexibility is possible here.

For illustration, the feature template generation rule in (2) will create the feature templates listed in Table 1, with j the order of the template. A dot (‘.’) is used to denote any feature, while a dash (‘-’) is used to separate the three sections of the template. Applying the feature templates of Table 1 to the single pattern #wh-e-n# \rightarrow /EH/ (where # denotes word boundaries), will result in the feature vectors listed in Table 2, all associated with the class /EH/.

2.2.4. Selecting a tie resolution strategy

Tie resolution does not play a key role in Default&Refine learning. When the algorithm is undecided about which of a small set of candidate patterns to select as the next rule, the algorithm can be guided in its choice by defining explicit tie resolution strategies. (Tie resolution – selecting one from a small set of equally possible rules identified by the algorithm – should not be confused with conflict resolution, which is performed automatically by the algorithm.)

Experimental results show that the tie resolution strategy has only limited effect on the performance of the algorithm for pronunciation prediction. (An appropriate tie resolution strategy typically results in very small improvements only – this may be different for other tasks.) If not specified explicitly by the user, the default strategy is to select the first candidate rule obtained with the smallest feature vector. For the pronunciation

Table 1
Feature vector templates of order j generated according to Eq. (2)

Template order	Feature templates
$j = 0$	-
$j = 1$	-. .-
$j = 2$	-. .- .- .-
$j = 3$	-. .- .- .- .- .-

Table 2
Feature vectors of order j associated with the feature templates of Table 1

Template order	Feature vectors	Class
$j = 0$	-e-	/EH/
$j = 1$	-e-n, h-e-	/EH/
$j = 2$	h-e-n, -e-n#, wh-e-	/EH/
$j = 3$	h-e-n#, wh-e-n, #wh-e-	/EH/

prediction results reported in the following section, simple heuristic tie resolution strategies were selected that matched those used in DEC: (1) From the candidate list, select the rule with the smallest feature vector (the most general rule). (2) If still tied (more than one candidate with the same feature vector size), select the most symmetrical rule from the remaining tied candidates. (3) If still tied, select the rule that has a right-first expansion above a left-first expansion. Once the data have been prepared and the algorithm initialised, the learning phase can be initiated, as discussed in the following section.

2.3. Learning

The learning phase is guided by the feature template generation rules created during data preparation. These rules specify the various feature vectors that need to be considered during rule extraction. A greedy search algorithm selects the best rule from this set of feature vectors one rule at a time, as described below.

2.3.1. Learning algorithm

During initialisation of the learning phase, the variables listed in Table 3 are created.

Using these variables, we define two additional sets: the *match_set* of all instances in *new* matching both the feature vector v and the class c of a specific pattern (v, c) , and the *conflict_set* of all patterns in *done* matching the feature vector v but conflicting with the class c of a specific pattern (v, c) :

$$\text{match_set}(v, c) = \text{match}(v, c, \text{new}) \quad (4)$$

$$\text{conflict_set}(v, c) = \cup_{c' \in \text{classes}, c' \neq c} \text{match}(v, c', \text{done}) \quad (5)$$

From these two sets we can define the net number of instances that will move from *new* to *done*, if pattern (v, c) is selected as the next rule:

$$\text{net_move}(v, c) = |\text{match_set}(v, c)| - |\text{conflict_set}(v, c)| \quad (6)$$

where $|X|$ indicates the number of members of set X .

Once initialised, the following steps are repeated until the *new* set is empty and, consequently, the *done* set consists of the entire set of training instances:

- (1) Select as the set of candidate patterns $\{(v_{cand}, c_{cand})\}$ all patterns that will – if selected as the next rule – result in the largest net move of patterns from *new* to *done*, that is

$$\{(v_{cand}, c_{cand})\} = \cup_{(v,c) \in \text{training}} \max \text{net_move}(v, c) \quad (7)$$

Table 3
Variables initialised as part of the learning phase

Variable	Description
<i>templates</i>	Set of allowed feature templates, generated according to the specified feature template generation rules, and ordered accordingly
<i>training</i>	Set of training instances. Duplicate entries are ignored: if duplicates occur, only a single instance is used during the learning phase. Each training instance consists of a feature vector and an associated class. During pronunciation prediction the feature vector consists of the focus grapheme and the full left and right graphemic contexts (e.g. the instance #wh-e-n# →/EH/)
<i>classes</i>	Set of possible classifications
<i>match(v, c, set)</i>	The subset of instances in <i>set</i> of class c matching the feature vector v . An instance matches the feature vector v if v is included in the features associated with the training instance. For example, all the feature vectors in Table 2 will match the training instance #wh-e-n# →/EH/
<i>rules</i>	List of extracted rules, ordered according to the reverse rule extraction order. Initially this list is empty
<i>new</i>	The subset of <i>training</i> predicted inaccurately by the current rule set. Initially this consists of the full set of training instances
<i>done</i>	The subset of <i>training</i> predicted accurately by the current rule set. Initially this set is empty

- (2) Select $(v', c') \in \{(v_{cand}, c_{cand})\}$, according to the prescribed tie resolution strategy, as discussed in Section 2.2.
- (3) Add (v', c') to the beginning of the list of *rules*.
- (4) Move the set of corrected instances from *new* to *done*, resulting in an updated *new''* and *done''*:

$$done'' = done \cup match_set(v', c') \quad (8)$$

$$new'' = new - match_set(v', c') \quad (9)$$

- (5) Move the set of instances now incorrect while previously correct from *done'* to *new'*, again resulting in an updated *new''* and *done''*:

$$done' = done' - conflict_set(v', c') \quad (10)$$

$$new'' = new' \cup conflict_set(v', c') \quad (11)$$

2.3.2. Discussion

The algorithm ensures that the next rule selected is always the one that will cause the most net training instances to be moved from the *new* to the *done* set, irrespective of the characteristics of the specific feature vector. (Specific characteristics only influence resolution among candidate rules, all of which will result in the same net number of instances moved.) Conflict is only resolved in the *done* set: new rules are allowed to conflict with training instances still in the *new* set.

It is worth noting that the decision with regard to the next rule to add is not guided by any heuristics with regard to context size or appropriate balance between left and right context lengths (as is the case with DEC). The number of net training instances that will be moved is the sole aspect taken into consideration – this then dictates the best context to select.

If a rule may consist of the full feature vector associated with an instance – as is typically the case – and as there are no conflicting instances (see Section 2.2), it is always possible to select, as a possible rule, a feature vector and class that specifies a single instance in the *new* set. For example, for the feature template generation rule (2) used in pronunciation prediction, the full feature vector is simply the full left and right context of the grapheme being predicted. If the grapheme being predicted is ‘e’ and the feature vector being considered is #wh-e-n#, then a rule that consists of the full feature vector would be the rule #wh-e-n# → /EH/. Selection of such a rule will result in a *net_move* of one, as no other words in the training dictionary would match this rule. As the algorithm selects the rule with the maximum *net_move* value, this maximum value will always be larger than or equal to one, which ensures that the algorithm will always converge. In the worst case, the algorithm will converge at a speed of one training instance per rule generated, but much faster convergence is typical.

While this is a conceptually simple algorithm, care is required during implementation to ensure computational efficiency, as discussed further in Section 2.5.

2.4. Classification

When using non-recursive features (as used in the version of the algorithm described in this paper), classification is a straightforward process: a value is associated with each feature in the test instance, one feature at a time, by applying the first matching rule found in the ordered rule list. Only a single rule is applied per feature predicted and no additional conflict resolution is required. (The ordering of the rule list ensures that only one rule will ever be triggered.)

In the case of pronunciation prediction this means that the graphemes are processed from left to right, and for each grapheme the first matching rule is applied.

2.5. Practical considerations

In order to ensure that the algorithm remains computationally tractable, even for large data sets, the following techniques are used.

- Instances are pre-processed and the training instances relevant to a single feature being predicted are extracted and written to file. All further manipulation considers a single file and therefore a single feature (and associated set of training instances) at a time. When the task is pronunciation prediction, the individual files are created per grapheme and each file contains all the word patterns associated with a specific grapheme.
- The context size of the feature vectors considered is grown systematically according to the order specified by the feature template generation rules: only feature vectors up to order $max + win$ are evaluated, where max indicates the current largest rule, and win is defined to ensure that any larger feature contexts that may be applicable are considered, without requiring all feature vectors to be searched. The win variable can be made arbitrarily large, depending on computational resources.
- While the two large sets *new* and *done* are used to keep track of the status of training instances, further manipulation utilises two sets of sub-vectors: the *possible* and *caught* feature vector sets, generated by applying the feature templates up to order $max + win$ to the training instances in *new* and *done*, respectively, removing all existing rules from the *possible* set, and counting the number of times per possible outcome that each feature vector is observed. The *net_move* value can now be calculated efficiently for each feature vector and class by subtracting the conflicting count in *caught* from the *possible* count associated with that feature vector and class, where the conflicting count is the number of times a matching feature vector is observed with a conflicting class.
- As, for every feature pattern, the class corresponding to the most observed training instances associated with that pattern remains the same, irrespective of the number of words that have been processed, only counts for the winning class need to be kept per pattern.
- Whenever a feature vector in the *possible* or *caught* set reaches a count of zero, the feature vector is deleted and not considered further, unless reactivated based on an inter-set move of a related training instance.
- Structuring of all training instances in an efficient tree structure results in improved computational and storage efficiency.

3. Evaluation

In this section, we evaluate the performance of the Default&Refine algorithm in comparison with pronunciation prediction algorithms described in the literature. In Section 3.1 we describe our experimental approach and the specific data used during experimentation. We then report on results as we evaluate the performance of the algorithm with regard to asymptotic accuracy (Section 3.2), ability to generalise from small data sets (Section 3.3) and compactness of the rule set (Section 3.4).

3.1. Experimental setup

We use three corpora in our evaluation:

- FONILEX (Mertens and Vercammen, 1998), a pronunciation lexicon of Dutch words as spoken in the Flemish part of Belgium. We use the same pre-aligned 173 874-word lexicon as used by Hoste et al. (2000).
- OALD, the Oxford Advanced Learners Dictionary of Contemporary English (Mitten, 1992), a British English pronunciation lexicon. We use the same pre-aligned 60 399-word lexicon as used by Black et al. (1998).
- Meraka Afrikaans 2004, a 7782-word Afrikaans pronunciation dictionary benchmarked in Davel (2005).

We initially considered comparing results on NETtalk (Sejnowski and Rosenberg, 1987) as well. This is a publicly available 20 008-word English pronunciation lexicon that includes hand-crafted grapheme-to-phoneme alignments, and is popular for testing automated pronunciation prediction algorithms. However, it seems that results are reported on various versions of the dictionary that are available (typically as errors are found and corrected) making this a less trustworthy resource to use for benchmarking.

The difficulty in comparing pronunciation prediction results does not only stem from the various versions of the training dictionaries available, but also from the different measures reported. ‘Word correctness’ is a

straightforward measure, namely the percentage of words that are predicted 100% correctly, typically ignoring the placement of any nulls. Phoneme-based measures are more tricky. Once a pronunciation prediction has been generated, the aligned word-pronunciation prediction pair can be compared directly with the aligned reference dictionary and the number of correct grapheme-to-phoneme mappings counted. When expressed as a percentage of all phonemes in either of the pronunciations (as these are equal) we refer to this as *pre-aligned phoneme correctness*.

Alternatively, since the positions of the nulls are irrelevant when the pronunciation is used in a speech processing system, a more accurate measure can be obtained by removing all nulls from both the predicted pronunciation and the reference pronunciation, aligning the two pronunciations and comparing the aligned versions. Since the two pronunciations are no longer of equal length, it is now necessary to report on two measures: *phoneme correctness* (the number of phonemes identified correctly) and *phoneme accuracy* (number of correct phonemes minus number of insertions, divided by the total number of phonemes in the correct pronunciation). The relationship between phoneme correctness, phoneme accuracy and pre-aligned phoneme correctness for one specific experiment (Default&Refine rule extraction on OALD) is illustrated in Fig. 2. As can be seen, the differences are large enough that care should be taken when comparing results. (The specific relationships among these measures depend on the data being analysed – the specific trends shown here are for illustration only.)

While some authors prefer word correctness as the more trustworthy measure to be used in comparisons, it does not always provide a complete picture, as elaborated below.

Except where stated otherwise, we perform 10-fold cross-validation, based on a 90% training and 10% test set. We indicate clearly the specific measure we use in reporting, and also report on the standard deviation of the mean of each of these measurements, indicated by σ_{10} . (If the mean of a random variable is estimated from n independent measurements, and the standard deviation of those measurements is σ , the standard deviation of the mean is $\sigma_n = \frac{\sigma}{\sqrt{n}}$.)

3.2. Asymptotic performance

We first evaluate the effectiveness of the algorithm when trained using a large dictionary for a language with a fairly regular spelling system, where we expect the algorithm to perform well. We then consider the more difficult task of pronunciation prediction for English, a language with an irregular spelling system, in Section 3.2.2.

3.2.1. Regular spelling systems

In this section, we select Flemish as our target language and evaluate the accuracy of the Default&Refine algorithm when trained on 90% of the FONILEX training set.

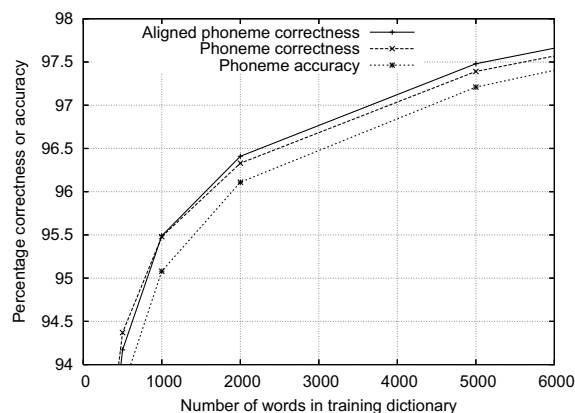


Fig. 2. The relationship between pre-aligned phoneme correctness, phoneme correctness and phoneme accuracy reporting the same results (learning curve on OALD using Default&Refine).

We compare the performance of the algorithm with that of three alternative algorithms previously reported on in the literature. All experiments utilise the same alignments and overall data set, but partitioning into cross-validation training and test sets differ, as exact partitions from previous experiments were not available. Results are listed in Table 4: the *IB1-IG-Hoste* result utilises IB1-IG, an instance-based learning algorithm, and is as reported by Hoste et al. (2000); the *DEC-grow* and *DEC-min* results are calculated using variants of Dynamically Expanding Context, as described by Davel and Barnard (2004b).

In addition to these previously published results, we use the exact partitions used to generate the new Default&Refine values and perform our own comparison using the Tilburg Memory-Based Learner (TiMBL) developed by Daelemans et al. (2004). The best TiMBL results were obtained when using the IB1 algorithm with 3 nearest neighbours ($k = 3$), Modified Value Difference Metric (MVDM) as similarity measure throughout training, information gain as feature weighting mechanism and class voting based on exponential decay (with $\alpha = 1$). These parameters were selected with the assistance of a parameter searching utility (*paramsearch*, van den Bosch (1990)), as well as own experimentation. Results are listed as *IB1-IG-Timbl* in Table 4, while the *D&R* result reports the Default&Refine values.

The focus of Hoste et al. (2000) was to investigate the effect of cascading two classifiers – one trained on FONILEX and one on CELEX (a Dutch variant corpus), and creating meta-classifiers using C5.0 (decision tree learning), IB1-IG (instance-based learning), IGTREE (a computationally more effective but less accurate version of IB1-IG) and MACCENT (a maximum entropy-based algorithm). The highest accuracy reported was for such a meta-classifier system: 91.55% word correctness for a single meta-classifier; and 92.25% word correctness for a meta-meta-classifier of all meta-classifiers. It should be noted that these systems all utilised the CELEX data as an additional data source.

We find that Default&Refine has good asymptotic accuracy, and performs better than any of the other classifiers evaluated.

3.2.2. Less regular spelling systems

As the Default&Refine algorithm is motivated by ‘default behaviour’ we were interested in the extent in which the algorithm would fail for a language with a significantly less regular spelling system, such as English. We therefore evaluate the asymptotic performance of the algorithm against benchmark results available for the OALD corpus.

In Table 5, we compare the performance of Default&Refine (*D&R*) with the results obtained by Black et al. (1998) using Classification and Regression Trees (*CART*) for two data sets: one including stress assignment (SA) and one without. We use the exact alignments, training set and test set as used by Black et al. (1998).

In addition, we perform a full 10-fold cross-validation experiment, comparing Default&Refine and the version of PbA as described by Marchand and Damper (2000) using the same training data. It should be noted that the *CART* trees were generated taking part-of-speech information into account – additional information that neither Default&Refine nor PbA use in this experiment.

It is reassuring to find that the performance of the algorithm does not degrade unacceptably for this less regular pronunciation prediction task. While PbA does perform better (10% improvement on phoneme error rate over Default&Refine), the Default&Refine results compare well, providing a 40% improvement on the phoneme error rate achieved using *CART*. The implications of the difference in word error rate becomes clearer in Section 3.3.3.

Table 4
Phoneme correctness, phoneme accuracy and word correctness comparison for different algorithms using the FONILEX corpus

	Phoneme correct $\pm\sigma_{10}$		Phoneme accuracy $\pm\sigma_{10}$		Word correct $\pm\sigma_{10}$	
<i>IB1-IG-Hoste</i>	98.18	0.01	–	–	86.37	–
<i>DEC-grow</i>	98.50	0.01	98.32	0.04	88.60	0.07
<i>DEC-min</i>	98.58	0.01	98.41	0.01	89.58	0.06
<i>IB1-IG-Timbl</i>	98.67	0.01	98.55	0.01	90.24	0.07
<i>D&R</i>	99.43	0.01	99.36	0.01	95.64	0.05

Table 5

Aligned phoneme correctness, phoneme correctness, phoneme accuracy and word correctness comparison for CART, PbA and Default&Refine using the OALD corpus

	Pre-aligned phoneme correct	Phoneme correct	Phoneme accuracy	Word correct
Incl. SA, one set only				
<i>CART</i> (without POS)	–	95.32	–	71.28
<i>CART</i> (with POS)	–	95.80	–	74.56
<i>D&R</i> (without POS)	–	97.12	96.87	83.76
Excl. SA, one set only				
<i>CART</i> (with POS)	–	96.36	–	76.92
<i>D&R</i> (without POS)	–	97.80	97.56	87.40
Excl. SA, cross-validated				
<i>D&R</i>	97.77	97.74	97.47	86.76
<i>D&R-groups</i>	97.82	97.81	97.54	87.00
<i>PbA</i>	98.00	–	–	89.37

SA indicates that stress assignment is also required, while POS indicates that part-of-speech information was used.

In a separate set of experiments by [Damper et al. \(2005\)](#), PbA was tested on the even larger British English Example Pronunciation (BEEP) dictionary. A 178090-word subset of BEEP was used consisting of those words that do not contain graphemic nulls. (The PbA algorithm as implemented by [Marchand and Damper \(2000\)](#) and [Damper et al. \(2005\)](#) is unable to deal with words containing graphemic nulls.) The best results obtained by PbA were 97.89% pre-aligned phone correctness and 87.48% word correctness using leave-one-out testing. With the same data set Default&Refine obtains 97.75% phoneme correctness and 84.41% word correctness using 10-fold cross-validation and 10 distinct 1000-word test sets.

On closer evaluation it is not clear whether this comparison is a valid one: the BEEP dictionary as reported on by [Damper et al. \(2005\)](#) contains a large number of obvious errors (such as /EY P R IH V AE R IH K EY SH N/ for the word ‘aprication’ and /D IH Z AA S T R EH P AX R AX S N AX S/ for the word ‘disastrousness’). Indeed it seems that a process analogous to PbA was used to construct the database, which would explain why, given a phonemic transcription such as /N AH M B AX Z/ for the word ‘noms’, words such as ‘americanos’ and ‘pannos’ are included in the dictionary as /AX M EH R IH K AX N AH M B AX Z/ and /P AE N N AH M B AX Z/. BEEP may therefore not be a reliable resource to use for the comparison of pronunciation prediction algorithms.

3.3. Learning curve

In order to use this algorithm for the bootstrapping of pronunciation lexicons, we are interested in its performance with very small training sets. Since the correction of incorrectly predicted phonemes is the most labour-intensive aspect of bootstrapping pronunciation lexicons ([Davel and Barnard, 2004c](#)), improving learning efficiency represents a significant improvement to the overall bootstrapping process.

We therefore evaluate word and phoneme accuracy for different training lexicons of sizes smaller than 5000 words. As these results are sensitive to training and test set selection, even when cross validating, we use the exact training and test sets to obtain comparative results for the different algorithms. We report on three different experiments comparing different algorithms and data sets. While the three experiments are not directly comparable, the combined result is conclusive with regard to Default&Refine’s performance when trained using small data sets.

3.3.1. Comparison with DEC and IBI-IG using FONILEX

In this experiment we select training subsets of different sizes from the FONILEX dictionary. [Fig. 3](#) demonstrates the phoneme accuracy learning curve for Default&Refine in comparison with two alternative algorithms: DEC-min, the most accurate DEC variant tested by [Davel and Barnard \(2004b\)](#), and the same IBI-IG implementation as reported in [Section 3.2.1](#). Cross-validation is performed and each rule set is evaluated against the full 17387-word test set.

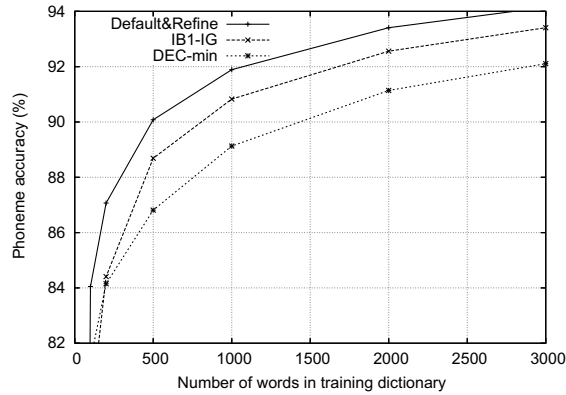


Fig. 3. Phoneme accuracy during training with initial 3000 words of FONILEX dictionary.

The algorithm performs well, achieving 90% phoneme accuracy prior to the 500-word mark. IB1-IG requires an additional 300 words and DEC-min another 800 before the same level of accuracy is reached.

3.3.2. Comparison with PbA using Afrikaans 2004

PbA provided the best results when comparing the asymptotic accuracy of different algorithms. How does this result relate to the initial learning curves of the algorithms? First, we compare learning curves using the Afrikaans 7782-word dictionary. Afrikaans has a fairly regular spelling system (comparable with Flemish) and should achieve a fairly similar learning curve.

Fig. 4 and 5a demonstrates the aligned phoneme correctness learning curves for the two algorithms. Two different results are provided for Default&Refine, learning with or without grouped graphemes (*D&R* and *D&R-groups*). It is clear that Default&Refine learns significantly more quickly during the initial learning phase. (As the database is fairly small, it is not possible to comment conclusively on comparative asymptotic performance on this data set.)

It is very interesting to note the difference in performance observed when evaluating either aligned phoneme correctness or word correctness. According to the aligned phoneme correctness measure, Default&Refine (either with or without grouped graphemes) is the clear winner. However with regard to word correctness, *D&R-groups* is the better performer, with *D&R* and PbA demonstrating similar learning curves from 2000 words onward. It therefore seems that at the same level of word correctness, Default&Refine tends to make fewer mistakes per word (achieving higher phoneme correctness) for the same level of word correctness. This trend is evaluated further in Section 3.3.3.

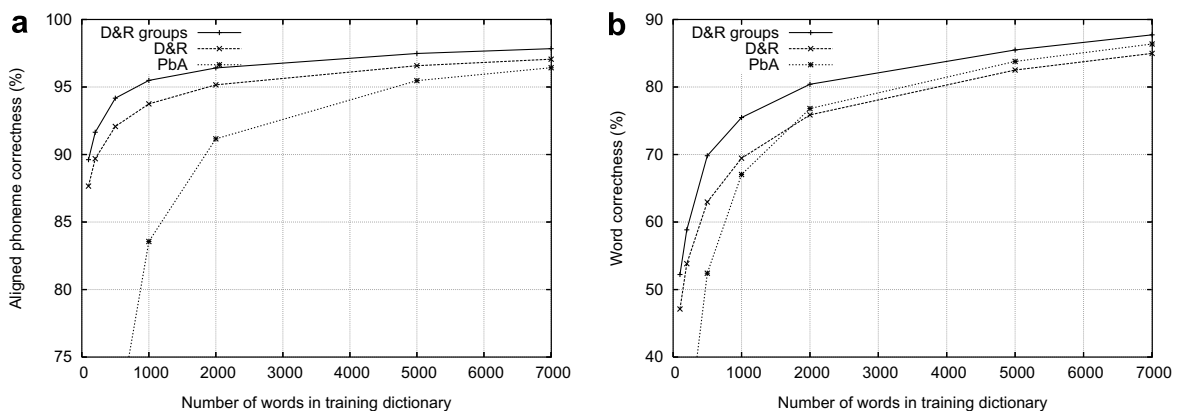


Fig. 4. Comparing PbA and Default&Refine using the full Afrikaans dictionary.

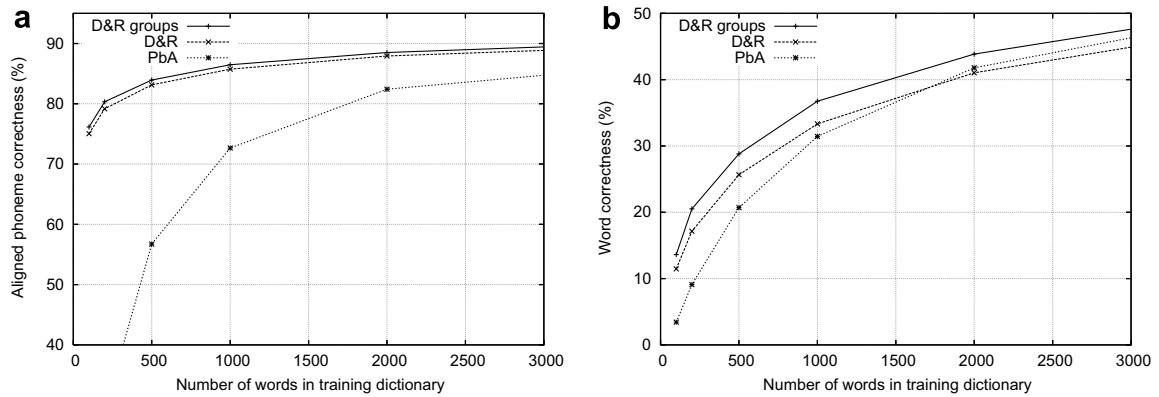


Fig. 5. Comparing PbA and Default&Refine using subsets of up to 3000 words from OALD.

3.3.3. Comparison with pronunciation by analogy using OALD

In the final learning efficiency experiment, we evaluate the learning curves for English using both PbA and Default&Refine. As PbA’s asymptotic performance on this data set is better than that of Default&Refine (see Section 3.2.2) we are interested in whether this difference in performance holds from the start. Again, we perform cross-validation and evaluate each rule set against the full 5900-word test set.

Fig. 4a contains the aligned phoneme correctness learning curves for the different algorithms: PbA and Default&Refine with or without grouped graphemes. It is clear that Default&Refine learns significantly more quickly. In fact, PbA only overtakes Default&Refine learning with and without grouped graphemes, at 15000 and 20000 words, respectively. Again it is clear from the difference in word correctness and aligned phoneme correctness that Default&Refine tends to make smaller mistakes in more words, while PbA tends to make bigger mistakes per word, but in fewer words.

3.4. Size of the rule set

While the size of the rule set is typically not a concern during grapheme-to-phoneme rule-based bootstrapping, it can be important for other applications such as pronunciation lexicon compression. (The size of the rule set is also generally a useful heuristic indication of the generalisation that can be expected.) We therefore analyse the size of the rule set using the FONILEX corpus, and find that the rule set extracted by Default&Refine is significantly smaller than that extracted by DEC-min (the same DEC variant as reported on above), as shown in Fig. 6a and b. Default&Refine provides both a more accurate and more compact pre-

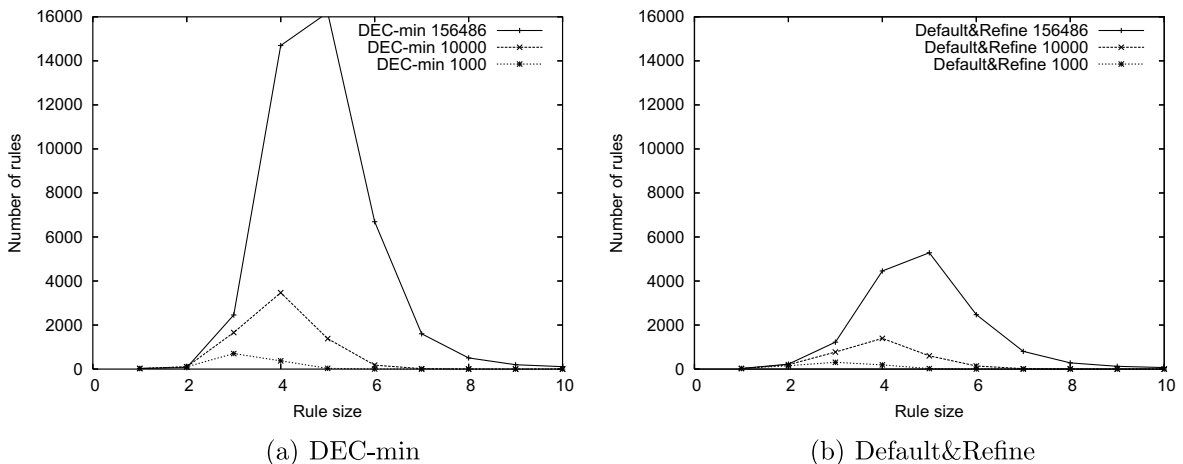


Fig. 6. Number of rules per context size extracted by DEC-min and Default&Refine from training lexicons of three different sizes.

diction model: the 156486-word training lexicon is represented with 100% accuracy by 15053 rules. It is also interesting to see how the most common rule size increases in similar fashion for both algorithms as the training dictionary grows.

As IB1-IG is an instance-based learning algorithm, no explicit rules are generated during training – all training instances encountered are stored. It is therefore not possible to compare rule set size directly. One relevant measure is that the IB1-IG algorithm trained on the full training set with a context window of five generates 581 347 nodes, which requires 175 Megabytes of storage. This increases to 68 909 369 nodes which requires approximately 2.6 Gigabytes of storage if a full context window is allowed. It should be noted that these nodes are generated for fast computation and were never intended for dictionary compression. The full training dictionary itself utilises 3.8 Megabytes of storage (1.1 Megabytes if compressed).

These storage requirements can be compared to the approximately 15000 rules extracted by Default&Refine using the full context window and all training instances. The space requirement for this rule set is only 132 Kilobytes (56 Kilobytes if compressed). When grouped graphemes are utilised during rule extraction, a further 19% decrease in rule set size is achieved.

3.5. Controlling the rule set size

Default&Refine has another useful characteristic: it is possible to control the trade-off between rule set accuracy and rule set size in a predictable way. Since rules are ordered according to generality, only a percentage of the more general rules can be retained if a smaller rule set is required, and the accuracy of the rule set will degrade gracefully. Fig. 7a demonstrates how the observed degradation slowly increases as a decreasing percentage of rules is retained (again using the FONILEX corpus). As indicated in Fig. 7b, degradation occurs at a linear rate in the logarithmic domain until fewer than approximately 40% of the rules remain, after which the speed of degradation accelerates. Interestingly, when only 1% of rules (the 150 most general rules) remain, the phoneme accuracy is still fairly high at 88.24%.

3.6. Utilising contextual information

It is interesting to consider the extent in which the graphemic context is utilised during training. In Fig. 8 we display the phoneme accuracy obtained if the context size is limited to x graphemes to the right and left of the focal grapheme. (A context window of x therefore signifies a largest graphemic context of $2x + 1$.) Both Default&Refine and the IB1-IG classifier follow similar trajectories with limited improvement once a context window of 3 is reached, and almost no improvement beyond a context window of 7 (when trained on 90% of the 156486-word FONILEX corpus).

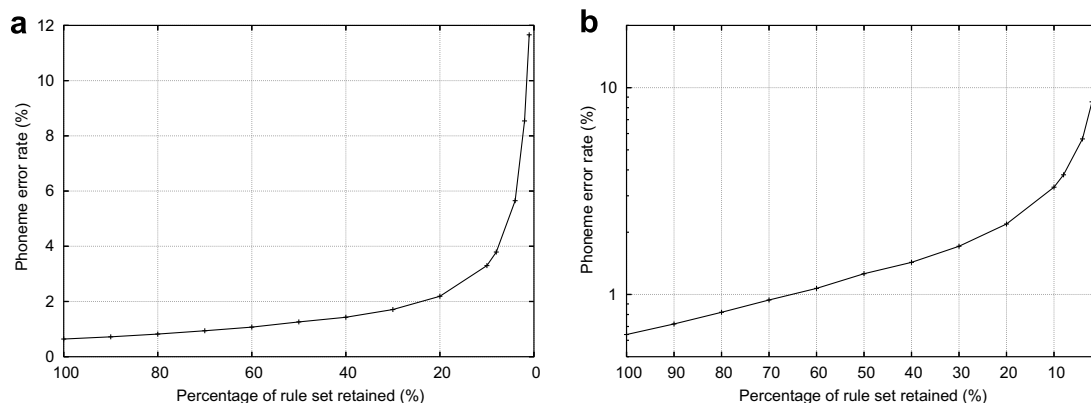


Fig. 7. The effect of decreasing rule set size on phoneme error rate. Until only 30% of rules remain, degradation is linear in the logarithmic domain.

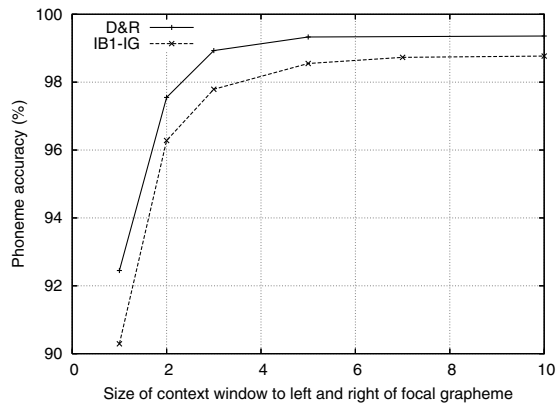


Fig. 8. Effect of different context sizes on phoneme accuracy.

4. Comparing Default&Refine with alternative algorithms

In this section, we describe the conceptual differences between Default&Refine and a number of alternative algorithms, including Dynamically Expanding Context (DEC), Transformation-based learning (TBL), Pronunciation by analogy (PBA), decision tree learning and Van Coile's inductive learning of grapheme-to-phoneme rules.

4.1. Dynamically expanding context

Dynamically Expanding Context (DEC) as defined by Kohonen (1986) was first applied to the pronunciation prediction task by Torkkola (1993). DEC derives a set of transformation rules from two-level data that, for pronunciation prediction, consists of the grapheme string and the phoneme string. DEC requires an explicitly defined specificity hierarchy that is utilised during learning. For pronunciation prediction this hierarchy is typically a symmetrically expanding context size, starting with right-first expansion. During rule extraction the most general rules (according to the specificity hierarchy) are extracted by generating one rule per training instance. If all rules agree the rule extraction process is completed; if not, the triggering pattern of all conflicting rules are expanded by one character. This process is repeated until all conflicts are resolved. Once all the rules have been extracted, these are then ordered in a tree structure, and the probabilities of different outcomes at each intermediate node are calculated. During pronunciation prediction the tree is traversed until a leaf is reached and the corresponding rule executed. If a leaf is not reached, the most probable outcome of the last intermediate node is generated.

The Default&Refine algorithm can easily be used to mimic the behaviour of DEC. First, the template generation rules should specify an increasing feature context, with the different features ordered according to the specificity hierarchy as used by a comparative DEC implementation. In addition, Default&Refine must be restricted only to consider a rule as a candidate if all the less specific versions of that rule are already included in the rule set. While the DEC rule set will contain more rules, the two rule sets will exhibit the same behaviour on an independent test set. Default&Refine does not calculate explicit probabilities for unseen cases, but

Table 6
Example of patterns available during training

Pattern	Class	Instances
-a-bc	p_1	3
-a-bd	p_2	2
-a-be	p_2	2

The number of training instances per pattern and class combination is also shown.

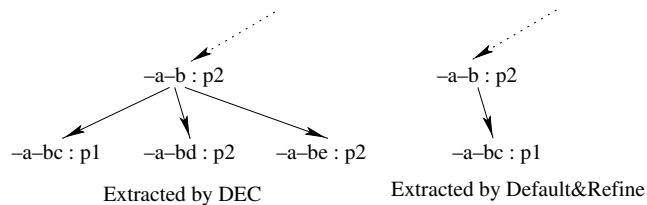


Fig. 9. Rules extracted by DEC based on patterns from Table 6.

implicitly selects back-off rules using the same probabilities. For example, if a number of training instances associated with the three patterns shown in Table 6 occur during training, DEC and Default&Refine will extract the two different rule sets shown in Fig. 9. Even though these rule sets differ, their classification results will be equivalent.

During prediction, whether rules are applied by finding the first applicable rule according to rule extraction order (as in the case of Default&Refine) or by traversing the tree from the top down until a leaf (or the last intermediate node) is reached, the same rule will be selected. Default&Refine can therefore be viewed as a generalisation of DEC; the additional flexibility afforded by this generalisation is seen to impact learning beneficially.

4.2. Transformation-based learning

Transformation-based learning (TBL) as defined by Brill (1995) is another technique that generates transformation rules from two-level data. The TBL algorithm uses an ‘initial state annotator’ to assign initial outcomes to all training instances. The TBL ‘learner’ then uses a greedy algorithm to select the transformation (from a predefined set) that will result in the largest reduction in errors, and assigns new outcomes to all training instances accordingly. This process is repeated until no further error reduction is possible.

While the rationale for each of the two approaches is quite different, Default&Refine and TBL are very similar algorithms. It is possible to view Default&Refine learning as a simplification of TBL, with a triggering environment that is limited to the first level of data (i.e. during pronunciation prediction only graphemes are considered in the triggering environment, not phonemes also). Default&Refine does not require an initial state annotator but as an a priori assignment can typically be used as a plausible initial state annotator for TBL, this difference does not constitute any substantial advantage of Default&Refine.

The real trade-off between the two algorithms lies in balancing the generality of TBL with the simplicity of Default&Refine. The Default&Refine algorithm does not use intermediate data the way TBL does, and in its non-recursive instantiation (as described in this paper) cannot solve problems that require a rule triggering environment consisting of two levels. These disadvantages are offset by a simplicity that results in a straightforward implementation and improved understandability of the rules extracted. While for TBL a whole sequence of rules may affect a single outcome, Default&Refine always triggers a single rule to achieve any outcome, and rules may therefore be analysed and understood in isolation. While Default&Refine can be viewed as a restrictive version of TBL, limiting the algorithm in this way results in both simplified implementation and improved performance.

4.3. Van Coile’s inductive learning

In van Coile (1990) an algorithm is described for the inductive learning of grapheme-to-phoneme rules that is very closely related to the Default&Refine algorithm. (We thank an anonymous reviewer for pointing out this work.) Conceptually the two algorithms are very similar; differences relate mainly to two aspects: (1) Default&Refine’s use of feature templates and (2) different implementational choices made.

The use of templates in Default&Refine is a generalisation (analogous to that used in TBL) that provides flexibility when applying the algorithm to different tasks, but does not influence the essence of the conceptual approach. When restricting Default&Refine to the single template defined by Eq. (3), the two approaches are

very similar apart from the implementation choices made in order to be able to deal with the large rule space being searched.

As mentioned in Section 2.5, the number of contexts that needs to be considered when doing a greedy search of all possible applicable contexts can be extremely large. Conceptually, Default&Refine does not limit the number of contexts considered in any way: by organising the contexts in an efficient tree structure, and growing only the parts of the tree required, as and when necessary, an implementation can be achieved that is efficient even for large databases. This then does not require heuristic techniques to limit the search space, as Van Coile's algorithm does.

The latter algorithm keeps track of three parameters: C , the length of all contexts currently being considered, M , the maximal context that occurred in the previously determined rules and a performance increase threshold T . Only the best rule of context length C is considered at a time. A new rule is accepted only if the increase in performance exceeds the value T , where T is lowered with a factor α whenever the context length exceeds M with a window parameter β . Whenever the threshold is reset, the C is also reset to 1, and the process repeated.

4.4. Pronunciation by analogy

Pronunciation by analogy (PbA) techniques use chunks of words contained in a training dictionary to assemble the pronunciation of an unknown test instance according to various strategies. This technique has been studied by various researchers (Dedina and Nusbaum, 1991; Yvon, 1996; Damper et al., 1999) and typically achieves high asymptotic accuracy, especially for languages with an irregular spelling system, such as English.

The two underlying philosophies are quite different: while PbA utilises chunks of data, carefully controlling the amount of overlap, Default&Refine considers one grapheme at a time, ignoring the potentially useful information contained in the identity of neighbouring rules. Still, it is interesting to note that as the context size of rules increases, this distinction between context-dependent rules and analogy-based methods starts to blur. For example, if a highly irregular word such as 'autostrade' is predicted with Default&Refine, the actual rules triggered typically have very large contexts, as shown in Table 7. While variables such as degree of overlap or number of chunks contained in the pronunciation are not controlled as such, the rules being triggered actually resemble analogy-based chunks to an extent.

In the comparative experiments reported in this paper we found PbA to outperform Default&Refine with regard to asymptotic accuracy when trained on English dictionaries, but Default&Refine was seen to exhibit a much steeper learning curve, learning more quickly on fewer training instances. Also, it seems as if Default&Refine tends to make smaller mistakes on more words when trained to a similar level of accuracy as PbA (which tends to make less mistakes on word-level, the mistakes tending to be larger – a larger number of wrong phonemes per incorrect word). Which error behaviour is more suitable depends on the specific application environment envisaged.

4.5. Decision trees

Decision tree learning utilises a set of predefined questions and training data to generate a decision tree: a tree structure consisting of questions that can be asked about a data item being classified. Each question splits the data into disjunct subsets. During classification of a test instance the decision tree is traversed according to

Table 7

Actual Default&Refine rules triggered during prediction of the irregular word 'autostrade'

-a-utostrade →/au/	-t- →/t/
a-u- →/ϕ/	-r- →/r/
-t- →/t/	ostr-a-d →/aa/
ut-o-***r →/ou/	-d- →/d/
-s- →/s/	ostrad-e- →/ei/

A * indicates the 'any grapheme' set.

the answer to each question until a final classification is obtained. During each stage of decision tree learning a single question is selected such that the resulting split of the training data is the ‘cleanest’ according to some measure (e.g. information gain). Based on the selected question, the data are split into subsets and the process repeated per subset.

The most significant difference between Default&Refine learning and decision tree learning is that Default&Refine considers all training data for every transformation learned, and not only a subset as decision tree learning does. Default&Refine learning can be viewed as a form of decision tree learning where the split criterion is the transformation resulting in maximum *net_move* (as defined in Eq. (6)) where *net_move* is calculated across the entire training set. Also, the new transformation is then added as a leaf to *all* current leaves, and not only to a single place in the tree as during decision tree learning. Note that where the transformation triggering environments of two leaves in a single branch conflict, the second leaf will never be invoked during classification, and may be left out during learning.

While decision trees subdivide the training data into smaller and smaller subsets during training, Default&Refine seeks for confirmation across the entire set of training data prior to selecting the next rule. This is analogous to drawing a decision boundary in a continuous space and then using information from both sides of the decision boundary when drawing the next boundary. While Default&Refine does not fit as easily into the learning framework of decision tree learning, understanding the differences provides an interesting perspective on the algorithm and its superior performance on the pronunciation-prediction task.

4.6. Other classifiers

While Default&Refine is further removed from classifiers such as a k -nearest neighbour classifier, kernel function or a multilayer perceptron it is still interesting to note a few significant differences between Default&Refine and these other classifiers. Firstly, the Default&Refine algorithm orders features explicitly. In contrast to the classifiers mentioned, features are not weighted, but rather, only a subset of features is considered at a time: the algorithm remains ‘blind’ to additional features unless these are required by the disambiguation process.

Default&Refine rule extraction can be viewed as a process whereby all training instances are ordered in the search space according to generality. Now the nearest neighbours at any distance are identified with a distance metric that changes depending on where in the search space the compared instance occurs. (The features to consider when evaluating ‘closeness’ become fewer as the training instances become more general.) Once all instances at the closest distance have been identified (according to the single applicable distance metric at that point in the search space), a majority vote determines the class.

Another difference relates to how Default&Refine deals with errors in the training data. Like most instance-based learning algorithms, Default&Refine also memorises all instances in the training data. Though it tends not to generalise from isolated errors, it will reproduce previously seen errors during testing. Default&Refine is therefore less robust against training errors than a k -nearest neighbour classifier with k greater than one. It is possible to create a more robust version of Default&Refine by discarding later rules: rules created by a number of words fewer than a threshold. However, for pronunciation prediction it is detrimental to discard this ‘tail’ of rules created by rare pronunciations, as even exceptions tend to occur in small families supporting generalisation (Daelemans et al., 1999), as can also be seen from Fig. 7a.

5. Conclusion

Default&Refine is a conceptually simple and highly practical algorithm that has a number of attractive properties, including rapid generalisation from small data sets, good asymptotic accuracy, and the production of compact rule sets. Apart from reliance on some level of ‘fallback behaviour’ as discussed in this paper, the algorithm does not rely on any language-specific assumptions and performs well over a range of pronunciation prediction tasks.

In this paper we have shown that the algorithm learns more efficiently (achieves higher accuracy on smaller data sets) than any of the other pronunciation prediction algorithms studied. This is an important characteristic when developing pronunciation models for the many under-resourced languages of the world. In addition, we

have demonstrated both the compactness of the standard rule set extracted, as well as the ability of the algorithm to generate an arbitrarily small rule set in such a way that the trade-off between rule set size and accuracy is well controlled.

Further work related to this algorithm has shown it to be fairly robust to noise in the training data (Davel, 2005), able to incorporate pronunciation variants (Davel and Barnard, 2006), and useful for the detection of errors in the training data (Davel and Barnard, 2005). Two versions have been defined, standard Default&Refine as described here and incremental Default&Refine (Davel and Barnard, 2005), a version that performs local optimisation, and consequently has slightly reduced accuracy in exchange for computational efficiency. Incremental Default&Refine is typically used in interactive pronunciation bootstrapping systems between synchronisation events, when standard Default&Refine performs a global optimisation.

Default&Refine has been integrated into our pronunciation lexicon bootstrapping system² and used to generate pronunciation dictionaries in a number of South African languages, including isiZulu, Sepedi, Afrikaans, Setswana and isiXhosa. Current work includes the generation of dictionaries in several other under-resourced languages, the formalisation of a recursive version of Default&Refine (able to create rules containing two-level triggering environments) and the application of this algorithm to other natural language processing tasks.

Acknowledgements

We would like to thank Piet Mertens for providing us with access to the FONILEX data, and Veronique Hoste and Alan Black for providing us with access to their FONILEX and OALD experimental data, respectively. We would also like to thank Tasanawan Soonklang and Bob Damper for kindly running the set of pronunciation by analogy comparative experiments summarized in Sections 3.2.1 and 3.3, specifically for the purposes of this paper.

References

- Andersen, O., Kuhn, R., Lazarides, A., Dalsgaard, P., Haas, J., Noth, E., 1996. Comparison of two tree-structured approaches for grapheme-to-phoneme conversion. In: *Proceedings of the International Conference on Spoken Language Processing (ICSLP)*, vol. 3, Philadelphia, PA, pp. 1700–1703.
- Black, A., Lenzo, K., Pagel, V., 1998. Issues in building general letter to sound rules. In: *Proceedings of the 3rd ESCA Workshop on Speech Synthesis*, Jenolan Caves, Australia, pp. 77–80.
- Brill, E., 1995. Transformation-based error-driven learning. *Computational Linguistics* 21, 543–565.
- Daelemans, W., van den Bosch, A., Zavrel, J., 1999. Forgetting exceptions is harmful in language learning. *Machine Learning* 34 (1–3), 11–41.
- Daelemans, W., Zavrel, J., van der Sloot, K., van den Bosch, A., 2004. Timbl: Tilburg Memory based Learner, Version 5.1, Reference Guide. <<http://ilk.uvt.nl/download/pub/papers/ilk0402.pdf>>.
- Damper, R., Marchand, Y., Adamson, M., Gustafson, K., 1999. Evaluating the pronunciation component of text-to-speech systems for English: a performance comparison of different approaches. *Computer Speech and Language* 13 (April), 155–176.
- Damper, R., Marchand, Y., Marsters, J., Bazin, A., 2005. Aligning text and phonemes for speech technology applications using an EM-like algorithm. *International Journal of Speech Technology* 8, 147–160.
- Davel, M., Barnard, E., 2003. Bootstrapping for language resource generation. In: *Proceedings of the Symposium of the Pattern Recognition Association of South Africa*, South Africa, pp. 97–100.
- Davel, M., Barnard, E., 2004a. A default-and-refinement approach to pronunciation prediction. In: *Proceedings of the Symposium of the Pattern Recognition Association of South Africa*, South Africa, pp. 119–123.
- Davel, M., Barnard, E., 2004b. The efficient creation of pronunciation dictionaries: machine learning factors in bootstrapping. In: *Proceedings of Interspeech*, Jeju, Korea, pp. 2781–2784.
- Davel, M., Barnard, E., 2004c. The efficient creation of pronunciation dictionaries: human factors in bootstrapping. In: *Proceedings of Interspeech*, Jeju, Korea, pp. 2797–2800.
- Davel, M., Barnard, E., 2005. Bootstrapping pronunciation dictionaries: practical issues. In: *Proceedings of Interspeech*, Lisboa, Portugal, pp. 1561–1564.
- Davel, M., Barnard, E., 2006. Developing consistent pronunciation models for phonemic variants. In: *Proceedings of Interspeech*, Pittsburgh, PA, pp. 1260–1263.
- Davel, M., 2005. *Pronunciation modelling and bootstrapping*. Ph.D. Thesis, University of Pretoria.
- Dedina, M., Nusbaum, H., 1991. PRONOUNCE: a program for pronunciation by analogy. *Computer Speech and Language* 5, 55–64.

² *DictionaryMaker* (Davel and Barnard, 2003) available for download from <http://dictionarymaker.sourceforge.net>.

- Henderson, L., 1985. On the use of the term ‘grapheme’. *Language and Cognitive Processes* 1, 135–148.
- Hoste, V., Daelemans, W., Sang, E., Gillis, S., 2000. Meta-learning for phonemic annotation of corpora. In: *Proceedings of the International Conference on Machine Learning (ICML-2000)*, pp. 375–382.
- Kohonen, T., 1986. Dynamically expanding context, with application to the correction of symbol strings in the recognition of speech. In: *Proceedings of the 8th International Conference on Pattern Recognition (8th ICPR)*, Paris, France, pp. 1148–1151.
- Marchand, Y., Damper, R., 2000. A multistrategy approach to improving pronunciation by analogy. *Computational Linguistics* 26, 195–219.
- Maskey, S., Tomokiyo, L., Black, A., 2004. Bootstrapping phonetic lexicons for new languages. In: *Proceedings of Interspeech*, Jeju, Korea, October, pp. 69–72.
- Mertens, P., Vercammen, F., 1998. Fonilex Manual. Technical Report, K.U.Leuven CCL. <<http://bach.arts.kuleuven.ac.be/fonilex/>>.
- Mitten, R., 1992. Computer-usable version of Oxford Advanced Learner’s Dictionary of Current English. Technical Report, Oxford Text Archive. <<http://ota.ahds.ac.uk/texts/0154.html>>.
- Sejnowski, T., Rosenberg, C., 1987. Parallel networks that learn to pronounce English text. *Complex Systems*, 145–168.
- Torkkola, K., 1993. An efficient way to learn English grapheme-to-phoneme rules automatically. In: *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, vol. 2, Minneapolis, MN, pp. 199–202.
- van Coile, B., 1990. Inductive learning of grapheme-to-phoneme rules. In: *International Conference on Spoken Language Processing (ICSLP-1990)*, pp. 765–768.
- van den Bosch, A., 1990. Wrapped progressive sampling search for optimizing learning algorithm parameters. In: *Proceedings of the Sixteenth Belgian–Dutch Artificial Intelligence Conference*, Groningen, The Netherlands, pp. 219–226.
- Viterbi, A., 1967. Error bounds for convolutional codes and a asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory* 13, 260–269.
- Yvon, F., 1996. Grapheme-to-phoneme conversion using multiple unbounded overlapping chunks. In: *Proceedings of Conference on New Methods in Natural Language Processing (NeMLaP)*, Ankara, Turkey, pp. 218–228.