Modularity in Inductively-Learned Word Pronunciation Systems *

Antal van den Bosch[1], **Ton Weijters**[2], **Walter Daelemans**[1]

[1] ILK / Computational Linguistics
Tilburg University
P.O. Box 90153
NL-5000 LE Tilburg
The Netherlands
{antalb,walter}@kub.nl

[2] Department of Information Technology
Eindhoven University of Technology
P.O. Box 513
NL-5600 MB Eindhoven
The Netherlands
A.J.M.M.Weijters@tm.tue.nl

Abstract

In leading morpho-phonological theories and state-of-the-art text-to-speech systems it is assumed that word pronunciation cannot be learned or performed without in-between analyses at several abstraction levels (e.g., morphological, graphemic, phonemic, syllabic, and stress levels). We challenge this assumption for the case of English word pronunciation. Using IGTREE, an inductive-learning decision-tree algorithms, we train and test three word-pronunciation systems in which the number of abstraction levels (implemented as sequenced modules) is reduced from five, via three, to one. The latter system, classifying letter strings directly as mapping to phonemes with stress markers, yields significantly better generalisation accuracies than the two multi-module systems. Analyses of empirical results indicate that positive utility effects of sequencing modules are outweighed by cascading errors passed on between modules.

1 Introduction

Learning word pronunciation can be a hard task when the relation between the spelling of a language and its corresponding pronunciation is many-to-many. The English writing system and its pronunciation are a notoriously complex example, caused by an apparent conflict between *analogy* and *inconsistency*:

Analogy. When two words or word chunks have a similar spelling, they tend to have a similar pronunciation. This tendency (which generalises to other language tasks as well) is usually referred to as the *analogy principle*(De Saussure, 1916; Yvon, 1996; Daelemans, 1996).

Inconsistency. Much of the analogy in English word pronunciation is disrupted by productive and complex word morphology, word stress, and graphematics.

Influential pre-Chomskyan linguistic theories have been pointing at the analogy principle as the underlying principle for language learning (De Saussure, 1916), and at induction as the reasoning method for generalising from learned instances of language tasks to new instances through analogy (Bloomfield, 1933). However, methods and resources (e.g., computer technology) were not available then to demonstrate how induction through analogy could be employed to learn and model language tasks. Partly due to this lack of demonstrating power, Chomsky later stated

> "... I don't see any way of explaining the resulting final state [of language learning] in terms of any proposed general developmental mechanism that has been suggested by artificial intelligence, sensorimotor mechanisms, or anything else" (Chomsky, in (Piatelli–Palmarini, 1980), p. 100).

Chomsky's argument is based on the assumption that generic learning methods such as induction cannot discover autonomously essential levels of abstraction in language processing tasks. Applied to morpho-phonology, the argument states that generic learning methods are not able to discover morphology, graphematics, and stress patterns autonomously when learning word pronunciation, although this knowledge appears essential. Phonological and morphological theories, influenced by Chomskyan theory across the board since the publication of SPE (Chomsky and Halle, 1968), have generally adopted the idea of abstraction levels in various guises (e.g., levels, tapes, tiers, grids) (Goldsmith, 1976; Liberman and Prince, 1977; Koskenniemi, 1984; Mohanan, 1986). Although there is no general consensus on which levels of abstraction can be discerned in phonology and morphology, there is a rough, global agreement on the fact that words can be represented on different abstraction levels as

*This research was partially performed by the first and second author at the Department of Computer Science of the Universiteit Maastricht (The Netherlands), and partially in the context of the "Induction of Linguistic Knowledge" research programme, partially supported by the Foundation for Language Speech and Logic (TSL), funded by the Netherlands Organization for Scientific Research (NWO).

Antal van den Bosch, Ton Weijters and Walter Daelemans (1998) **Modularity in Inductively-Learned Word Pronunciation Systems.** In D.M.W. Powers (ed.) *NeMLaP3/CoNLL98: New Methods in Language Processing and Computational Natural Language Learning,* ACL, pp 185-194.

strings of letters, graphemes, morphemes, phonemes, syllables, and stress patterns.

According to these leading morpho-phonological theories, systems that (learn to) convert spelled words to phonemic words in one pass, i.e., without making use of abstraction levels, are assumed to be unable to generalise to new cases: going through the relevant abstraction levels is deemed essential to yield correct conversions of previously unseen words. This assumption implies that if one wants to build a system that converts text to speech, one should implement explicitly the relevant levels of abstraction. Such explicit implementations of abstraction levels can indeed be witnessed in many state-of-the-art speech synthesisers, implemented as (sequential) modules (Allen, Hunnicutt, and Klatt, 1987; Daelemans, 1988).

In this paper we challenge the assumption that levels of abstraction must be made explicit in learning and performing the word-pronunciation task. We do this by applying an inductive-learning algorithm from machine learning to word pronunciation. From a wealth of existing algorithms in machine learning (Mitchell, 1997), we choose IGTREE (Daelemans, Van den Bosch, and Weijters, 1997), an inductive-learning decision-tree learning algorithm. IGTREE is a fast algorithm which has been demonstrated to be applicable to language tasks (Van den Bosch and Daelemans, 1993; Van den Bosch, Daelemans, and Weijters, 1996; Daelemans, Van den Bosch, and Weijters, 1997). We construct IGTREE decision trees for word pronunciation, and perform empirical tests to estimate the trees' generalisation accuracy, i.e., their ability to process new, unseen word-pronunciation instances correctly.

Rather than constructing and testing a single system, our approach is to test different modularisations of the word-pronunciation task systematically, to allow for an empirical comparison of word-pronunciation systems with *and* without the explicit learning of abstraction levels. First, we train (by inductive learning) and test a word-pronunciation model reflecting linguistic assumptions on abstraction levels quite closely: the model is composed of five sequentially-coupled modules. Second, we train and test a model in which the number of modules is reduced to three, integrating two pairs of levels of abstraction. Third, we train and test a model performing word pronunciation in a single pass, i.e., without modular decomposition.

The paper is structured as follows: first, in Section 2 we provide a description of IGTREE, the data on which the IGTREE is trained and tested, and the applied experimental methodology. Second, in Section 3 we introduce the three word-pronunciation systems, and for each system we describe the experiments performed and discuss the results obtained. In Section 4 we compare the three systems and analyse the consequences of modularisation. Section 5 briefly mentions related work on inductive learning of word pronunciation. Section 6 summarises the results obtained and lists some points of discussion.

2 Algorithm, Data, Methodology

2.1 Algorithm: IGTREE

IGTREE (Daelemans, Van den Bosch, and Weijters, 1997) is a *top-down induction of decision trees* (TDIDT) algorithm (Breiman et al., 1984; Quinlan, 1993). TDIDT is a widely-used method in supervised machine learning (Mitchell, 1997). IGTREE is designed as an optimised approximation of the instance-based learning algorithm IB1-IG (Daelemans and Van den Bosch, 1992; Daelemans, Van den Bosch, and Weijters, 1997). In IGTREE, *information gain* is used as a guiding function to compress a data base of instances of a certain task into a decision tree[1]. Instances are stored in the tree as paths of connected nodes ending in leaves which contain classification information. Nodes are connected via arcs denoting feature values. Information gain is used in IGTREE to determine the order in which feature values are added as arcs to the tree. Information gain is a function from information theory, and is used similarly in ID3 (Quinlan, 1986) and C4.5 (Quinlan, 1993).

The idea behind computing the information gain of features is to interpret the training set (i.e., the set of task instances for which all classifications are given and which are used for training the learning algorithm) as an information source capable of generating a number of messages (i.e., classifications) with a certain probability. The information entropy H of such an information source can be compared in turn for each of the features characterising the instances (let n equal the number of features), to the average information entropy of the information source when the value of those features are known. Data-base information entropy $H(D)$ is equal to the number of bits of information needed to know the classification given an instance. It is computed by equation 1, where p_i (the probability of classification i) is estimated by its relative frequency in the training set.

$$H(D) = -\sum_i p_i log_2 p_i \qquad (1)$$

To determine the information gain of each of the n features $f_1 \ldots f_n$, we compute the average information entropy for each feature and subtract it from the information entropy of the data base. To compute the information entropy for a feature f_i, given in equation 2, we take the weighted average information entropy of the data base restricted to each possible value for the feature. The expression $D_{[f_i=v_j]}$

[1] IGTREE can function with any feature weighting method, such as gain ratio (Quinlan, 1993); for all experiments reported here, information gain was used.

refers to those patterns in the data base that have value v_j for feature f_i, j is the number of possible values of f_i, and V is the set of possible values for feature f_i. Finally, $|D|$ is the number of patterns in the (sub) data base.

$$H(D_{[f_i]}) = \sum_{v_j \in V} H(D_{[f_i=v_j]}) \frac{|D_{[f_i=v_j]}|}{|D|} \qquad (2)$$

Information gain of feature f_i is then obtained by equation 3.

$$G(f_i) = H(D) - H(D_{[f_i]}) \qquad (3)$$

In IGTREE, feature-value information is stored in the decision tree on arcs. The first feature values, stored as arcs connected to the tree's top node, are those representing the values of the feature with the highest information gain, followed at the second level of the tree by the values of the feature with the second-highest information gain, etc., until the classification information represented by a path is unambiguous. Knowing the value of the most important feature may already uniquely identify a classification, in which case the other feature values of that instance need not be stored in the tree. Alternatively, it may be necessary for disambiguation to store a long path in the tree.

Apart from storing uniquely identified class labels at leafs, IGTREE stores at each non-terminal node information on the most probable classification given the path so far. The most probable classification is the most frequently occurring classification in the subset of instances being compressed in the path being expanded. Storing the most probable class at non-terminal nodes is essential when processing new instances. Processing a new instance involves traversing the tree by matching the feature values of the test instance with arcs the tree, in the order of the feature information gain. Traversal ends when (i) a leaf is reached or when (ii) matching a feature value with an arc fails. In case (i), the classification stored at the leaf is taken as output. In case (ii), we use the most probable classification on the last non-terminal node most recently visited instead.

2.2 Data Acquisition and Preprocessing

The resource of word-pronunciation instances used in our experiments is the CELEX lexical data base of English (Burnage, 1990). All items in the CELEX data bases contain hyphenated spelling, syllabified and stressed phonemic transcriptions, and detailed morphological analyses. We extracted from the English data base of CELEX all the above information, resulting in a data base containing 77,565 unique items (word forms with syllabified, stressed pronunciations and morphological segmentations).

For use in experiments with learning algorithms, the data is preprocessed to derive fixed-size instances. In the experiments reported in this paper different morpho-phonological (sub)tasks are investigated; for each (sub)task, an instance base (training set) is constructed containing instances produced by *windowing* (Sejnowski and Rosenberg, 1987) and attaching to each instance the classification appropriate for the (sub)task under investigation. Table 1 displays example instances derived from the sample word booking. With this method, for each (sub) task an instance base of 675,745 instances is built.

In the table, six classification fields are shown, one of which is a composite field; each field refers to one of the (sub)tasks investigated here. M stands for morphological decomposition: determine whether a letter is the initial letter of a morpheme (class '1') or not (class '0'). A is graphemic parsing[2]: determine whether a letter is the first or only letter of a grapheme (class '1') or not (class '0'); a grapheme is a cluster of one or more letters mapping to a single phoneme. G is grapheme-phoneme conversion: determine the phonemic mapping of the middle letter. Y is syllabification: determine whether the middle phoneme is syllable-initial. S is stress assignment: determine the stress level of the middle phoneme. Finally, GS is integrated grapheme-phoneme conversion and stress assignment. The example instances in Table 1 show that each (sub)task is phrased as a classification task on the basis of windows of letters or phonemes (the stress assignment task S is investigated with both letters and phonemes as input). Each window represents a snapshot of a part of a word or phonemic transcription, and is labelled by the classification associated with the middle letter of the window. For example, the first letter-window instance ___book is linked with label '1' for the morphological segmentation task (M), since the middle letter b is the first letter of the morpheme *book*; the other instance labelled with morphological-segmentation class '1' is the instance with i in the middle, since i is the first letter of the (inflectional) morpheme ing. Classifications may either be binary ('1' or '0') for the segmentation tasks (M, A, and Y), or have more values, such as 62 possible phonemes (G) or three stress markers (primary, secondary, or no stress, S), or a combination of these classes (159 combined phonemes and stress markers, GS).

2.3 Methodology

Our empirical study focuses on measuring the ability of the IGTREE learning algorithm to use the knowledge accumulated during learning for the classification of new, unseen instances of the same (sub)task, i.e., we measure their generalisation accuracy. (Weiss and Kulikowski, 1991) describe *n-fold cross validation* (n-fold CV) as a procedure for mea-

[2] Graphemic parsing is not represented in the CELEX data. We used an automatic alignment algorithm (Daelemans and Van den Bosch, 1997) to determine which letters are the first or only letters of a grapheme.

instance	letter-window instances								phoneme-window instances				
	left		right	classifications					left		right	classif.	
number	context	focus	context	M	A	G	S	GS	context	focus	context	Y	S
1	_ _ _	b	o o k	1	1	/b/	1	/b/1	_ _ _	/b/	/u/ /-/ /k/	1	1
2	_ _ b	o	o k i	0	1	/u/	0	/u/0	_ _ /b/	/u/	/-/ /k/ /ı/	0	0
3	_ b o	o	k i n	0	0	/-/	0	/-/0	_ /b/ /u/	/-/	/k/ /ı/ /ŋ/	0	0
4	b o o	k	i n g	0	1	/k/	0	/k/0	/b/ /u/ /-/	/k/	/ı/ /ŋ/ /-/	1	0
5	o o k	i	n g _	1	1	/ı/	0	/ı/0	/u/ /-/ /k/	/ı/	/ŋ/ /-/ _	0	0
6	o k i	n	g _ _	0	1	/ŋ/	0	/ŋ/0	/-/ /k/ /ı/	/ŋ/	/-/ _ _	0	0
7	k i n	g	_ _ _	0	0	/-/	0	/-/0	/k/ /ı/ /ŋ/	/-/	_ _ _	0	0

Table 1: Example of instances generated from the word booking, with classifications for all of the subtasks investigated, viz. M, A, G, Y, S, and GS.

suring generalisation accuracy. For our experiments with IGTREE, we set up 10-fold CV experiments consisting of five steps. (i) On the basis of a data set, n partitionings are generated of the data set into one training set containing $((n-1)/n)$th of the data set, and one test set containing $(1/n)$th of the data set, per partitioning. For each partitioning, the three following steps are repeated: (ii) Information-gain values for all (seven) features are computed on the basis of the training set (cf. Subsection 2.1). (iii) IGTREE is applied to the training set, yielding an induced decision tree (cf. Subsection 2.1). (iv) The tree is tested by letting it classify all instances in the test set, which results in a percentage of incorrectly classified test instances. (v) When each of the n folds has produced an error percentage on test material, a mean generalisation error of the learned model is computed. (Weiss and Kulikowski, 1991) argue that by using n-fold CV, preferably with $n \geq 10$, one can retrieve a good estimate of the true generalisation error of a learning algorithm given an instance base. Mean results can be employed further in significance tests. In our experiments, $n = 10$, and one-tailed t-tests are performed.

3 Three word-pronunciation architectures

Our experiments are grouped in three series, each involving the application of IGTREE to a particular word-pronunciation system. The architectures of these systems are displayed in Figure 1. In the following subsections, each system is introduced, an outline is given of the experiments performed on the system, and the results are briefly discussed.

3.1 M-A-G-Y-S

The architecture of the M-A-G-Y-S system is inspired by SOUND1 (Hunnicutt, 1976; Hunnicutt, 1980), the word-pronunciation subsystem of the MITALK text-to-speech system (Allen, Hunnicutt, and Klatt, 1987). When the MITALK system is faced with an unknown word, SOUND1 produces on the basis of that word a phonemic transcription with stress markers (Allen, Hunnicutt, and Klatt, 1987). This word-pronunciation process is divided into the following five processing components:

1. *morphological segmentation*, which we implement as the module referred to as M;
2. *graphemic parsing*, module A;
3. *grapheme-phoneme conversion*, module G;
4. *syllabification*, module Y;
5. *stress assignment*, module S.

The architecture of the M-A-G-Y-S system is visualised in the left of Figure 1. It can be seen that the representations include direct output from previous modules, as well as representations from earlier modules. For example, the S module takes as input the syllable boundaries generated by the Y module, but also the phoneme string generated by the G module, and the morpheme boundaries generated by the M module.

M-A-G-Y-S is put to the test by applying IGTREE in 10-fold CV experiments to the five subtasks, connecting the modules after training, and measuring the combined score on correctly classified phonemes and stress markers, which is the desired output of the word-pronunciation system. An individual module can be trained on data from CELEX directly as input, but this method ignores the fact that modules in a working modular system can be expected to generate some amount of error. When one module generates an error, the subsequent module receives this error as input, assumes it is correct, and may generate another error. In a five-module system, this type of cascading errors may seriously hamper generalisation accuracy. To counteract this potential disadvantage, modules can also be trained on the output of previous modules. Modules cannot be expected to learn to repair completely random, irregular errors, but whenever a previous module makes *consistent* errors on a specific input, this may be recognised by the subsequent module. Having detected a consistent error, the subsequent module is

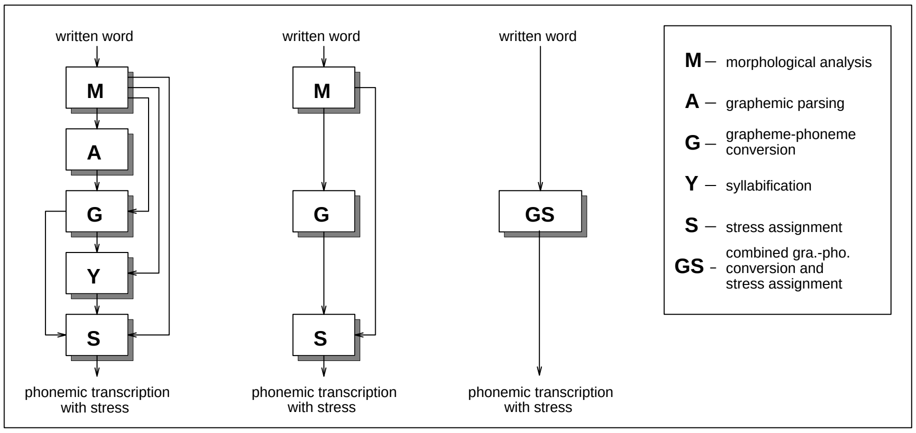

Figure 1: Architectures of the three investigated word-pronunciation systems. Left: M-A-G-Y-S; middle: M-G-S; right: GS. Rectangular boxes represent modules; the letter in the box corresponds to the subtask as listed in the legenda (far right). Arrows depict data flows from the raw input or a module, to a module or the output.

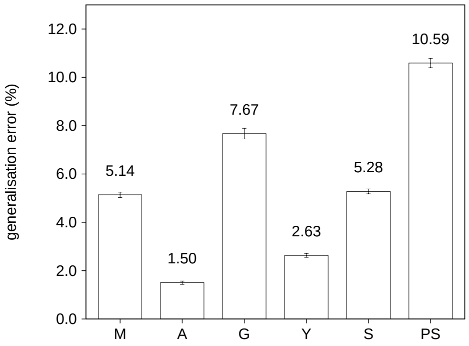

Figure 2: Generalisation errors on the M-A-G-Y-S system in terms of the percentage of incorrectly classified test instances by IGTREE on the five subtasks M, A, G, Y, and S, and on phonemes and stress markers jointly (PS).

then able to repair the error and continue with successful processing. Earlier experiments performed on the tasks investigated in this paper have shown that classification errors on test instances are indeed consistently and significantly decreased when modules are trained on the output of previous modules rather than on data extracted directly from CELEX (Van den Bosch, 1997). Therefore, we train the M-A-G-Y-S system, with IGTREE, by training the modules of the system on the output of predecessing modules. We henceforth refer to this type of training as *adaptive* training, referring to the adaptation of a module to the errors of a predecessing module.

Figure 2 displays the results obtained with IGTREE under the adaptive variant of M-A-G-Y-S. The figure shows all percentages (displayed above the bars; error bars on top of the main bars indicate standard deviations) of incorrectly classified instances for each of the five subtasks, and a joint error on incorrectly classified phonemes with stress markers, which is the desired output of the system. The latter classification error, labelled PS in Figure 2, regards classification of an instance as incorrect if either or both of the phoneme and stress marker is incorrect. The figure shows that the joint error on phonemes and stress markers is 10.59% of test instances, on average. Computed in terms of transcribed words, only 35.89% of all test words are converted to stressed phonemic transcriptions flawlessly. The joint error is lower than the sum of the errors on the G subtask and the S subtask, 12.95%, suggesting that about 20% of the incorrectly classified test instances involve an incorrect classification of both the phoneme and the stress marker.

3.2 M-G-S

The subtasks of graphemic parsing (A) and grapheme-phoneme conversion (G) are clearly related. While A attempts to parse a letter string into graphemes, G converts graphemes to phonemes. Although they are performed independently in M-A-G-Y-S, they can be integrated easily when the class-'1'-instances of the A task are mapped to their associated phoneme rather than '1', and the class-'0'-instances are mapped to a *phonemic null*, /-/, rather than '0' (cf. Table 1). This task integration is also used in the NETTALK model (Sejnowski and Rosenberg, 1987). A similar argument can be made for integrating the syllabification and stress assignment modules into a single stress-assignment module. Stress markers, in our definition of the stress-assignment subtask, are placed solely on the positions which are also marked as syllable boundaries (i.e., on syllable-initial phonemes). Removing the

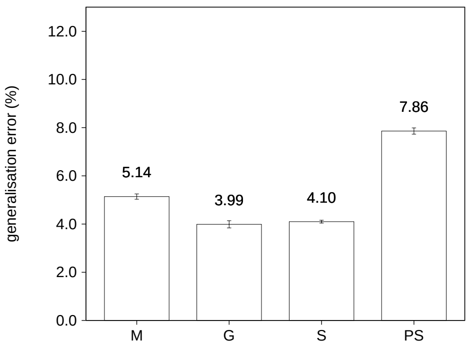

Figure 3: Generalisation errors on the M-G-S system in terms of the percentage of incorrectly classified test instances by IGTREE on the three subtasks M, G, and S, and on phonemes and stress markers jointly (PS).

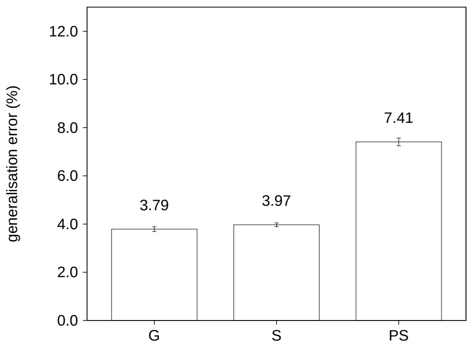

Figure 4: Percentage of generalisation errors made by IGTREE on the GS task, in terms of the percentage incorrectly classified test instances as well as on phonemes and stress assignments computed separately.

syllabification subtask makes finding those syllable boundaries which are relevant for stress assignment an integrated part of stress assignment. Syllabification (Y) and stress assignment (S) can thus be integrated in a single stress-assignment module S.

When both pairs of modules are reduced to single modules, the three-module system M-G-S is obtained. Figure 1 displays the architecture of the M-G-S system in the middle. Experiments on this system are performed analogous to the experiments with the M-A-G-Y-S system; Figure 3 displays the average percentages of generalisation errors generated by IGTREE on the three subtasks and phonemes and stress markers jointly (the error bar labelled PS).

Removing graphemic parsing (A) and syllabification (Y) as explicit in-between modules yields better accuracies on the grapheme-phoneme conversion (G) and stress assignment (S) subtasks than in the M-A-G-Y-S system. Both differences are significant; for G, ($t(19) = 43.70, p < 0.001$), and for S ($t(19) = 32.00, p < 0.001$). The joint accuracy on phonemes and stress markers is also significantly better in the M-G-S system than in the M-A-G-Y-S system ($t(37.50, p < 0.001$). Different from M-A-G-Y-S, the sum of the errors on phonemes and stress markers, 8.09%, is hardly more than the joint error on PSs, 7.86%: there is hardly an overlap in instances with incorrectly classified phonemes and stress markers. The percentage of flawlessly processed test words is 44.89%, which is markedly better than the 35.89% of M-A-G-Y-S.

3.3 GS

GS is a single-module system in which only one classification task is performed in one pass. The GS task integrates grapheme-phoneme conversion and stress assignment: to classify letter windows as corresponding to a *phoneme with a stress marker* (PS). In the GS system, a PS can be either (i) a phoneme or a phonemic null with stress marker '0', or (ii) a phoneme with stress marker '1' (i.e., the first phoneme of a syllable receiving primary stress), or (iii) a phoneme with stress marker '2' (i.e., the first phoneme of a syllable receiving secondary stress). The simple architecture of GS, which does not reflect any linguistic expert knowledge about decompositions of the word-pronunciation task, is visualised as the rightmost architecture in Figure 1. It only assumes the presence of letters at the input, and phonemes and stress markers at the output. Table 1 displays example instance PS classifications generated on the basis of the word booking. The phonemes with stress markers (PSs) are denoted by composite labels. For example, the first instance in Table 1, ___book, maps to class label /b/1, denoting a /b/ which is the first phoneme of a syllable receiving primary stress.

The experiments with GS were performed with the same data set of word pronunciation as used with M-A-G-Y-S and M-G-S. The number of PS classes (i.e., all possible combinations of phonemes and stress markers) occurring in this data base of tasks is 159. Figure 4 displays the generalisation errors in terms of incorrectly classified test instances. The figure also displays the percentage of classification errors made on phonemes and stress markers computed separately.

IGTREE yields significantly better generalisation accuracy on phonemes and stress markers, both jointly and independently. In terms of PSs, the accuracy on GS is significantly better than that of M-G-S with ($t(19) = 40.48, p < 0.001$), and that of M-A-G-Y-S with ($t(19) = 6.90, p < 0.001$). Its accuracy on flawlessly transcribed test words, 59.38%, is also considerably better than that of the modular systems. Compared to accuracies reported in related research on learning English word pronunciation (Sejnowski and Rosenberg, 1987; Wolpert, 1990; Diet-

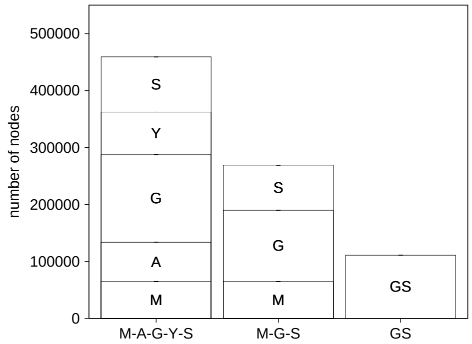

Figure 5: Average numbers of nodes in the decision trees generated by IGTREE for the M-A-G-Y-S, M-G-S, and GS systems. Compartments indicate the numbers of nodes needed for the trees of the subtasks specified by their labels.

terich, Hild, and Bakiri, 1995; Yvon, 1996) and on general quality demands of text-to-speech applications, an error of 3.79% on phonemes and 30.62% on words can be considered adequate, though still not excellent (Yvon, 1996; Van den Bosch, 1997).

4 Comparisons of M-A-G-Y-S, M-G-S, and GS

We have given significance results showing that, under our experimental conditions and using IGTREE as the learning algorithm, optimal generalisation accuracy on word pronunciation is obtained with GS, the system that does not incorporate any explicit decomposition of the word-pronunciation task. In this section we perform two additional comparisons of the three systems. First, we compare the sizes of the trees constructed by IGTREE on the three systems; second, we analyse the positive and negative effects of learning the subtasks in their specific systems' context.

Tree sizes

An advantage of using less or no decompositions in terms of computational efficiency is the total amount of memory needed for storing the trees. Although the application of IGTREE generally results in small trees that fit well inside small computer memories (for our modular (sub)tasks, tree sizes vary from 64,821 nodes for the M-modules to 153,678 nodes for the G-module in M-A-G-Y-S, occupying 453,747 to 1,075,746 bytes of memory), keeping five trees in memory would not be a desirable feature for a system optimised on memory use. Figure 5 displays the summed number of nodes for each of the four IGTREE-trained systems under the adaptive variant. Each bar is divided into compartments indicating the amount of nodes in the trees generated for each of the modular subtasks.

Figure 5 shows that the model with the best generalisation accuracy, GS, is also the model taking up the smallest number of nodes. The amount of nodes in the single GS tree, 111,062, is not only smaller than the sum of the amount of nodes needed for the G and S modules in the M-G-S system (204,345 nodes); it is even smaller than the single tree constructed for the G subtask in the M-G-S system (125,182 nodes).

A minor difference in tree size can be seen between the trees built for the G-module in the M-G-S system, 125,182 nodes, and the G-module in the M-A-G-Y-S system, 153,678 nodes. A similar difference can be seen for the S-modules, taking up 79,163 nodes in the M-G-S system, and 96,998 nodes in the M-A-G-Y-S system. The size of the trees built for modules appears to increase when the module is preceded by more modules, which suggests that IGTREE is faced with a more complex task, including potentially erroneous output from more modules, when building a tree for a module further down a sequence of modules.

Utility effects

The particular sequence of the five modules as in the M-A-G-Y-S system reflects a number of assumptions on the *utility* of using output from one subtask as input to another subtask. Morphological knowledge is useful as input to grapheme-phoneme conversion (e.g., to avoid pronouncing ph in loophole as /f/, or red in barred as /rɛd/); graphemic parsing is useful as input to grapheme-phoneme conversion (e.g., to avoid the pronunciation of gh in through); etc. Thus, feeding the output of a module A into a subsequent module B implies that one expects to perform better on module B with A's input than without. The accuracy results obtained with the modules of the M-A-G-Y-S, M-G-S, and GS systems can serve as tests for their respective underlying utility assumptions, when they are compared to the accuracies obtained with their subtasks learned in isolation.

To measure the *utility effects* of including the outputs of modules as inputs to other modules, we performed the following experiments:

1. We applied IGTREE in 10-fold CV experiments to each of the five subtasks M, A, G, Y, and S, only using letters (with the M, A, G, and S subtasks) or phonemes (with the Y and the S subtasks) as input, and their respective classification as output (cf. Table 1). The input is directly extracted from CELEX. These experiments provide the baseline score for each subtask, and are referred to as the *isolated* experiments.
2. We applied IGTREE in 10-fold CV experiments to all subtasks of the M-A-G-Y-S, M-G-S, and GS systems, training *and* testing on input extracted directly from CELEX. The results from these experiments reflect what would be the accuracy of

sub-task	% generalisation error				
	isolated	ideal	(utility)	actual	(utility)
M-A-G-Y-S					
M	5.14	5.14	(0.00)	5.14	(0.00)
A	1.39	1.66	(−0.27)	1.50	(−0.11)
G	3.72	3.68	(+0.04)	7.67	(−3.95)
Y	0.45	0.75	(−0.30)	2.63	(−2.16)
S	7.96	2.67	(+5.29)	5.28	(+2.68)
M-G-S					
M	5.14	5.14	(0.00)	5.14	(0.00)
G	3.72	3.66	(+0.06)	3.99	(−0.27)
S	7.96	3.97	(+3.99)	4.10	(+3.86)
GS					
G	3.72	–	–	3.79	(−0.07)
S	4.71	–	–	3.97	(+0.74)

Table 2: Overview of utility effects of learning subtasks (M, A, G, Y, and S) as modules or partial tasks in the M-A-G-Y-S, M-G-S, and GS systems. For each module, in each system, the utility of training the module with ideal data (middle) and actual, modular data under the adaptive variant (right), is compared against the accuracy obtained with learning the subtasks in isolation (left). Accuracies are given in percentage of incorrectly classified test instances.

the modular systems when each module would perform perfectly flawless. We refer to these experiments as *ideal*.

With the results of these experiments we measure, for each subtask in each of the three systems, the utility effect of including the input of preceding modules, for the ideal case (with input straight from CELEX) as well as for the actual case (with input from preceding modules). A utility effect is the difference between IGTREE's generalisation error on the subtask in modular context (either ideal or actual) and its accuracy on the same subtask in isolation. Table 2 lists all computed utility effects.

For the case of the M-A-G-Y-S system, it can be seen that the only large utility effect, even in the ideal case, could be obtained with the stress-assignment subtask. In the isolated case, the input consists of phonemes; in the M-A-G-Y-S system, the input contains morpheme boundaries, phonemes, and syllable boundaries. The ideal positive effect on the S module of 5.29% less errors turns out to be a positive effect of 2.68% in the actual system. The latter positive effect is outweighed by a rather large negative utility effect on the grapheme-phoneme conversion task of −3.95%. Both the A and Y subtasks do not profit from morphological boundaries as input, even in the ideal case; in the actual M-A-G-Y-S system, the utility effect of including morphological boundaries from M and phonemes from G in the syllabification module Y is markedly negative: −2.16%.

In the M-G-S system, the utility effects are generally less negative than in the M-A-G-Y-S system. There is a small utility effect in the ideal case with including morphological boundaries as input to grapheme-phoneme conversion; in the actual M-G-S system, the utility effect is negative (−0.27%). The stress-assignment module benefits from including morphological boundaries and phonemes in its input, both in the ideal case and in the actual M-G-S system.

The GS system does not contain separate modules, but it is possible to compare the errors made on phonemes and stress assignments separately to the results obtained on the subtasks learned in isolation. Grapheme-phoneme conversion is learned with almost the same accuracy when learned in isolation as when learned as partial task of the GS task. Learning the grapheme-phoneme task, IGTREE is neither helped nor hampered significantly by learning stress assignment simultaneously. There is a positive utility effect in learning stress assignment, however. When stress assignment is learned in isolation with letters as input, IGTREE classifies 4.71% of test instances incorrectly, on average. (This is a lower error than obtained with learning stress assignment on the basis of phonemes, indicating that stress assignment should take letters as input rather than phonemes.) When the stress-assignment task is learned along with grapheme-phoneme conversion in the GS system, a marked improvement is obtained: 0.74% less classification errors are made.

Summarising, comparing the accuracies on modular subtasks to the accuracies on their isolated counterpart tasks shows only a few positive utility effects in the actual system, all obtained with stress assignment. The largest utility effect is found on the stress-assignment subtask of M-G-S. However, this positive utility effect does not lead to optimal accuracy on the S subtask; in the GS system, stress assignment is performed with letters as input, yielding the best accuracy on stress assignment in our investigations, viz. 3.97% incorrectly classified test instances.

5 Related work

The classical NETTALK paper by (Sejnowski and Rosenberg, 1987) can be seen as a primary source of inspiration for the present study; it has been so for a considerable amount of related work. Although it has been criticised for being vague and presumptuous and for presenting generalisation accuracies that can be improved easily with other learning methods (Stanfill and Waltz, 1986; Wolpert, 1990; Weijters, 1991; Yvon, 1996), it was the first paper to investigate grapheme-phoneme conversion as an interesting application for general-purpose learning algorithms. However, few reports have been made on the joint accuracies on stress markers and phonemes in work on the NETTALK data. To our knowledge, only (Shavlik, Mooney, and Towell, 1991) and (Dietterich, Hild, and Bakiri, 1995) provides such reports. In terms of incorrectly processed test instances, (Shavlik, Mooney, and Towell, 1991) obtain better performance with the back-propagation algorithm trained on distributed output (27.7% errors) than with the ID3 (Quinlan, 1986) decision-tree algorithm (34.7% errors), both trained and tested on small non-overlapping sets of about 1,000 instances. (Dietterich, Hild, and Bakiri, 1995) reports similar errors on similarly-sized training and test sets (29.1% for BP and 34.4% for ID3); with a larger training set of 19,003 words from the NETTALK data and an input encoding fifteen letters, previous phoneme and stress classifications, some domain-specific features, and error-correcting output codes ID3 generates 8.6% errors on test instances (Dietterich, Hild, and Bakiri, 1995), which does not compare favourably to the results obtained with the NETTALK-like GS task (a valid comparison cannot be made; the data employed in the current study contains considerably more instances).

An interesting counterargument against the representation of the word-pronunciation task using fixed-size windows, put forward by Yvon (Yvon, 1996), is that an inductive-learning approach to grapheme-phoneme conversion should be based on associating variable-length chunks of letters to variable-length chunks of phonemes. The chunk-based approach is shown to be applicable, with adequate accuracy, to several corpora, including corpora of French word pronunciations and, as mentioned above, the NETTALK data (Yvon, 1996). Experiments on other (larger) corpora, comparing both approaches, would be needed to analyse their differences empirically.

6 Discussion

We have demonstrated that a decision-tree learning algorithm, IGTREE, is able to learn English word pronunciation with modest to adequate generalisation accuracy: the less the learning task is decomposed in subtasks, the more adequate the generalisation accuracy obtained by IGTREE is. The best generalisation accuracy is obtained with the GS system, which does not decompose the task at all. The general disadvantage of the investigated modular systems is that modules do not perform their tasks flawlessly, while their expert-based decompositions do assume flawless performance. In practice, modules produce a considerable amount of irregular errors which cause subsequent modules to generate subsequent 'cascading' errors. Only the subtask of stress assignment is shown to be learned more successfully on the basis of modular input.

The best-performing system, GS, is trained to map windows of letters to combined class labels representing phonemes and stress markers. Compared to the M-A-G-Y-S and M-G-S systems, the GS system (i) lacks an explicit morphological segmentation and (ii) learns stress assignment jointly with grapheme-phoneme conversion on the basis of letter windows rather than phoneme windows. These two advantageous properties of the GS system lead to three suggestions. First, it appears better to leave morphological segmentation an implicit subtask; it can be left to the learning algorithm to extract the necessary morphological information needed to disambiguate between alternative pronunciations directly from the letter-window input. Second, letter-window instances provide the most reliable source of input for both grapheme-phoneme conversion and stress assignment. Third, stress assignment and grapheme-phoneme conversion can be integrated in one task, i.e., to map letter instances to 'stressed phonemes'.

A warning on the scope of these suggestions needs to be issued. The results described here are not only dependent of the resource (CELEX) and the (sub)task definitions (classification of windowed instances), but also on the use of IGTREE as the learning algorithm. The CELEX data appears robust and provides an abundance of English word pronunciations, not an inappropriately skewed subset of the English vocabulary. The windowing method appears a salient method to rephrase language tasks as classification tasks based on fixed-length inputs. It is not clear, however, to what extent IGTREE can be held responsible for the low accuracy on M-A-G-Y-S and M-G-S; IGTREE may be negatively sensitive in terms of generalisation accuracy to irregular errors in the input of a modular subtask. Although irregular errors are an inherent problem for modular systems, other learning algorithms may be able to handle such errors differently. Experiments with back-propagation learning applied to the same modular systems show siginficantly worse performance than that of IGTREE (Van den Bosch, 1997). It might be possible that instance-based learning algorithms (e.g., IB1-IG (Daelemans and Van den Bosch, 1992; Daelemans, Van den Bosch, and Weijters, 1997)), which have been demonstrated to outperform IGTREE on several language tasks (Daelemans, Gillis, and Durieux, 1994; Van den Bosch, Daelemans, and Weijters, 1996; Van den Bosch, 1997), perform better on the modular systems. Although such systems trained with IB1-IG would be computationally rather inefficient (Van den Bosch, 1997), employing IB1-IG in learning modular subtasks may lead to other differences in accuracy between modular systems.

A conclusion to be drawn from our study is that it is possible to learn the complex language task of English word pronunciation with a general-purpose inductive-learning algorithm, with an adequate level of generalisation accuracy. The results suggest that the necessity of decomposing word-pronunciation in several subtasks should be reconsidered carefully when designing an accuracy-oriented word-pronunciation system. Undesired errors generated by sequenced modules may outweigh the desired positive utility effects easily.

Acknowledgements

We thank Eric Postma, Maria Wolters, David Aha, Bertjan Busser, Jakub Zavrel, and the other members of the Tilburg ILK group for fruitful discussions.

References

Allen, J., S. Hunnicutt, and D. Klatt. 1987. *From text to speech: The MITalk system*. Cambridge, UK: Cambidge University Press.

Bloomfield, L. 1933. *Language*. New York: Holt, Rinehard and Winston.

Breiman, L., J. Friedman, R. Ohlsen, and C. Stone. 1984. *Classification and regression trees*. Belmont, CA: Wadsworth International Group.

Burnage, G., 1990. CELEX*: A guide for users*. Centre for Lexical Information, Nijmegen.

Chomsky, N. and M. Halle. 1968. *The sound pattern of English*. New York, NY: Harper and Row.

Daelemans, W. 1988. Grafon: A grapheme-to-phoneme system for Dutch. In *Proceedings Twelfth International Conference on Computational Linguistics (COLING-88), Budapest*, pages 133–138.

Daelemans, W. 1996. Experience-driven language acquisition and processing. In M. Van der Avoird and C. Corsius, editors, *Proceedings of the CLS Opening Academic Year 1996-1997*. Tilburg: CLS, pages 83–95.

Daelemans, W., S. Gillis, and G. Durieux. 1994. The acquisition of stress: a data-oriented approach. *Computational Linguistics*, 20(3):421–451.

Daelemans, W. and A. Van den Bosch. 1992. Generalisation performance of backpropagation learning on a syllabification task. In M. F. J. Drossaers and A. Nijholt, editors, *TWLT3: Connectionism and Natural Language Processing*, pages 27–37, Enschede. Twente University.

Daelemans, W. and A. Van den Bosch. 1997. Language-independent data-oriented grapheme-to-phoneme conversion. In J. P. H. Van Santen, R. W. Sproat, J. P. Olive, and J. Hirschberg, editors, *Progress in Speech Processing*. Berlin: Springer-Verlag, pages 77–89.

Daelemans, W., A. Van den Bosch, and A. Weijters. 1997. IGTree: using trees for classification in lazy learning algorithms. *Artificial Intelligence Review*, 11:407–423.

De Saussure, F. 1916. *Course de linguistique générale*. Paris: Payot. edited posthumously by C. Bally and A. Riedlinger.

Dietterich, T. G., H. Hild, and G. Bakiri. 1995. A comparison of ID3 and backpropagation for English text-to-speech mapping. *Machine Learning*, 19(1):5–28.

Goldsmith, J. 1976. An overview of autosegmental phonology. *Linguistic Analysis*, 2:23–68.

Hunnicutt, S. 1976. Phonological rules for a text-to-speech system. *American Journal of Computational Linguistics*, Microfiche 57:1–72.

Hunnicutt, S. 1980. Grapheme-phoneme rules: a review. Technical Report STL QPSR 2-3, Speech Transmission Laboratory, KTH, Sweden.

Koskenniemi, K. 1984. A general computational model for wordform recognition and production. In *Proceedings of the Tenth International Conference on Computational Linguistics / 22nd Annual Conference of the ACL*, pages 178–181.

Liberman, M. and A. Prince. 1977. On stress and linguistic rhythm. *Linguistic Inquiry*, (8):249–336.

Mitchell, T. 1997. *Machine learning*. New York, NY: McGraw Hill.

Mohanan, K. P. 1986. *The theory of lexical phonology*. Dordrecht: D. Reidel.

Piatelli–Palmarini, M., editor. 1980. *Language learning: The debate between Jean Piaget and Noam Chomsky*. Cambridge, MA: Harvard University Press.

Quinlan, J. R. 1986. Induction of decision trees. *Machine Learning*, 1:81–206.

Quinlan, J. R. 1993. C4.5*: Programs for machine learning*. San Mateo, CA: Morgan Kaufmann.

Sejnowski, T. J. and C. S. Rosenberg. 1987. Parallel networks that learn to pronounce English text. *Complex Systems*, 1:145–168.

Shavlik, J. W., R. J. Mooney, and G. G. Towell. 1991. Symbolic and neural learning algorithms: An experimental comparison. *Machine Learning*, 6:111–143.

Stanfill, C. and D. Waltz. 1986. Toward memory-based reasoning. *Communications of the* ACM, 29(12):1213–1228.

Van den Bosch, A. 1997. *Learning to pronounce written words, a study in inductive language learning*. Ph.D. thesis, Universiteit Maastricht.

Van den Bosch, A. and W. Daelemans. 1993. Data-oriented methods for grapheme-to-phoneme conversion. In *Proceedings of the 6th Conference of the EACL*, pages 45–53.

Van den Bosch, A., W. Daelemans, and A. Weijters. 1996. Morphological analysis as classification: an inductive-learning approach. In K. Oflazer and H. Somers, editors, *Proceedings of NeMLaP-2, Ankara, Turkey*, pages 79–89.

Weijters, A. 1991. A simple look-up procedure superior to NETtalk? In *Proceedings of* ICANN*-91, Espoo, Finland*.

Weiss, S. and C. Kulikowski. 1991. *Computer systems that learn*. San Mateo, CA: Morgan Kaufmann.

Wolpert, D. H. 1990. Constructing a generalizer superior to NETtalk via a mathematical theory of generalization. *Neural Networks*, 3:445–452.

Yvon, F. 1996. *Prononcer par analogie: motivation, formalisation et évaluation*. Ph.D. thesis, Ecole Nationale Supérieure des Télécommunication, Paris.