Eurospeech99, Budapest Hungary Sept. 5-10, 1999, pp. 2123-2126.

MACHINE LEARNING OF WORD PRONUNCIATION: THE CASE AGAINST ABSTRACTION

Bertjan Busser *Walter Daelemans* *Antal van den Bosch*

ILK / Computational Linguistics, Tilburg University, The Netherlands
{G.J.Busser,Walter.Daelemans,Antal.vdnBosch}@kub.nl

ABSTRACT

Word pronunciation can be learned by inductive machine learning algorithms when it is represented as a classification task: classify a letter within its local word context as mapping to its pronunciation. On the basis of generalization accuracy results from empirical studies, we argue that word pronunciation, particularly in complex spelling systems such as that of English, should not be modelled in a way that abstracts from exceptions. Learning methods such as decision tree and backpropagation learning, while trying to abstract from noise, also throw away a large number of useful exceptional cases. Our empirical results suggest that a memory-based approach which stores all available word-pronunciation knowledge as cases in memory, and generalises from this lexicon via analogical reasoning, is at all times the optimal modelling method.

1. INTRODUCTION

Word pronunciation can be learned by inductive machine learning algorithms when it is phrased as a classification task: classify a letter within its local word context as mapping to its pronunciation (e.g., phonemes and stress markers). Learning cases are drawn from a (preferrably large) pronunciation lexicon. Standard learning paradigms that have been applied to this type of classification-rephrasing of the word pronunciation task include error back-propagation in multi-layered perceptrons [12, 15] and decision-tree learning [8]. Both methods abstract from the learning material by compressing it into a data structure that can be used to classify new cases. The abstraction capacities of both artificial neural networks and decision trees are often highlighted as the cause for the relative success of both approaches, also on this task [15, 8]. However, several studies have been published that challenge this claim [19, 18, 7], by demonstrating that *memory-based learning* approaches yield superior accuracy to both backpropagation and decision-tree learning. In word pronunciation, each atypical spelling-pronunciation mapping is potentially a productive one: the word it occurs in (or a morphologically derived or inflected form) may always reoccur later on. Abstracting methods that delete noise tend to delete productive atypicalities as well.

In the memory-based learning approach, examples (cases) of word pronunciations (which, as in [15], take the form of a letter surrounded by some left and right neighbour letters, coupled with its associated pronunciation) are simply stored in memory. After training, pronunciations of new words can be constructed by decomposing the new word in similarly-formatted cases, and matching these cases to those in memory. Exact matches occur with cases from words that were also in the learning material, but also with words that are in part similar to learned words, since cases represent only parts of words. In these cases, the memorised pronunciation is simply copied. When no exact match between cases is available, because one or more letters mismatch, the typical memory-based learning algorithm performs an analogical reasoning step by searching for the most similar case or group of cases in memory, and extrapolating its class to the new case. Memory-based models of word pronunciation can thus be seen as "generalizing" or "inductive" lexicons: they can fully reproduce the information that was originally in the lexicon used for training, but at the same time they can extrapolate best-guess pronunciations for new words.

In this paper we review the results obtained in experiments in which four abstracting machine learning algorithms are applied to English word pronunciation, in comparison to a standard pure memory-based learning algorithm.

2. EMPIRICAL STUDIES

We synthesize the results obtained in previously reported and new experiments in which abstracting machine learning algorithms are applied to English word pronunciation, in comparison to a pure memory-based learning algorithm IB1-IG [3]. Before reporting on results in Subsection 2.3., we briefly introduce the En-

Features							
Left context			Focus	Right context			Class
1	2	3	4	5	6	7	label
_	h	e	a	r	t	s	0A:
b	o	o	k	i	n	g	0k
t	i	e	s	_	_	_	0z
_	_	a	f	a	r	_	1f

Table 1. Cases of the word pronunciation learning task.

glish word-pronunciation data used in all experiments in Subsection 2.1., and we describe IB1-IG in Subsection 2.2.

2.1. Data set characteristics

As sketched in the introductory section, we define the word-pronunciation task as the conversion of fixed-sized cases representing a letter in its local word context to a class representing the phoneme and the stress marker of that letter. To generate the cases, windowing is used [15]. Table 1 displays four cases and their classifications. Classifications, i.e., phonemes with stress markers, are denoted by composite labels. For example, the first case in Table 1, _hearts, maps to class label 0A:, denoting an elonged short 'a'-sound which is not the first phoneme of a syllable receiving primary stress. In this study, we chose a fixed window width of seven letters, which offers sufficient context information for adequate generalization performance [16]. From CELEX [2] we extracted, on the basis of its lexical data base of 77,565 words with their corresponding phonemic transcription with stress markers, a data base containing 675,745 cases. The number of classes (i.e., all possible combinations of phonemes and stress markers) occurring in this data base is 159.

2.2. Memory-based learning in IB1-IG

IB1-IG [4, 6] is a lazy learning algorithm that builds a data base of cases (the *case base*) during learning. A case consists of a fixed-length vector of n feature-value pairs (letters), and information field containing the classification of that particular feature-value vector (stress + phoneme). After the case base is built, new cases are classified by matching them to all cases in the case base, and by calculating with each match the *distance* between the new case X and the memory case Y, $\Delta(X,Y) = \sum_{i=1}^{n} w_i \, \delta(x_i, y_i)$, where n is the number of features, w_i is a weight for feature i, and $\delta = 0 \; if \; x_i = y_i, \; else \; 1$ is the distance per feature. Classification in memory-based learning is performed by the k-NN algorithm, that searches for the k 'nearest neighbors' (best-matching cases) of a new case using the $\Delta(X,Y)$ function. The majority class of the k nearest neighbours then determines the class of the new case. Usually, and in our experiments, k is set to 1.

The weight (importance) of a feature i, w_i, is estimated by computing its *information gain* (IG), which is the difference in uncertainty (entropy) within the set of cases between the situations without and with knowledge of the value of that feature: $w_i = H(C) - \sum_{v \in V_i} P(v) \times H(C|v)$, where C is the set of class labels, V_i is the set of values for feature i, and $H(C) = -\sum_{c \in C} P(c) \log_2 P(c)$ is the entropy of the class labels. The probabilities are estimated from relative frequencies in the training set.

2.3. Effects of abstraction

We present four approaches to abstraction in inductive learning: error back-propagation in multi-layer feedforward networks, top-down induction of decision trees, editing exceptional cases in memory-based learning, and generalizing cases in memory-based learning. The order in which they are presented reflects a decreasing amount of abstraction performed during learning; error backpropagation learning in artifical neural networks represents very strong abstraction, while generalizing cases tends to result in limited abstraction.

2.3.1. Error back-propagation in multi-layer feed-forward networks

Error back-propagation (BP) [12] is a learning algorithm designed for *multilayer feed-forward networks* (MFNs), a well-known type of artificial neural networks. MFNs are composed of interconnected simple computing elements organised in three or more layers: an input layer representing feature values, an output layer representing the class, and one or more hidden layers. Usually, the numbers of units in hidden layers are below those of the number of cases of the task to be learned; the network learns the task by encoding the input-output mapping in a relatively compact way.

We used a bitwise (local) encoding of feature values and classes, resulting in an input layer of 294 units, and an output layer of 159 units. In two experiments, we set the hidden layer to contain 50 and 200 units, respectively. For all technical details regarding other parameter settings in this particular experiment, the reader is referred to [16].

2.3.2. Top-down induction of decision trees

C5.0, a commercial version of C4.5 [11], is a *top-down induction of decision trees* (TDIDT) algorithm. On the basis of an case base of, C5.0 constructs a decision tree which compresses the classification information in the case base by exploiting differences in relative importance (information gain) of different features. Cases are stored in the tree as paths of connected nodes end-

ing in leaves which contain classification information. Nodes are connected via arcs denoting feature values.

In our experiments, we chose to use the default parameter settings of C5.0, which are set to moderate pruning of parts of the tree estimated to represent noisy or exceptional cases. Two parameters determine the amount of pruning in C5.0. The c parameter denotes the pruning confidence level, which ranges between 0% and 100%. This parameter computes the binomial probability of misclassifications within the set of cases represented at that node. By default, $c = 25\%$. The m parameter governs the minimum number of cases represented by a node. When $m > 1$, single exceptional or noisy cases are not stored in the tree. By default, $m = 2$.

2.3.3. *Editing exceptional cases in memory-based learning*

When cases in a memory-based learning system are never used as nearest neighbors in classifying other cases, or when they are even disruptive for classification, they may be discarded from memory. These two options form the bases of two approaches to editing found in the literature: (1) delete cases that can be deleted without harming the classification performance of the memory-based classifier [10], and (2) delete cases of which the classification is different from the majority class of their nearest neighbors [1].

We have implemented the latter type of editing on top of IB1-IG, for which we have implemented the class-prediction strength (CPS) function as proposed by [13]. This the ratio of the number of times a case is a nearest neighbour of another case with the same class and the number of times that the case is the nearest neighbour of another case regardless of the class. A CPS close to 0.0 indicates that the case is a bad predictor of classes of other cases, presumably indicating that the case is exceptional. After ranking all cases in the training sets of each experiment, we removed, in subsequent trials, the top 1%, 5%, 10%, and 50% of the cases with the lowest CPS.

2.3.4. *Generalizing cases in memory-based learning*

Algorithms that carefully merge cases start with storing all individual cases in memory, and then carefully merge pairs or groups of same-class nearest-neighbor cases to become single, more general cases, only when there is some evidence that these merging operations are not harmful to generalization performance. Generalised cases can be represented by conjunctions of *disjunctions of* feature values, or rules with wild-cards.

Although overall memory is compressed, the memory still contains individual items on which the same k-NN-based classification can be performed as in the pure memory-based case. The abstraction occurring in this approach is that after a merge, the merged cases incorporated in the new generalized case cannot be reconstructed individually. Example approaches to merging cases are NGE [14] and RISE [9]. The experiments reported here are performed with FAMBL, which features roughly the same functionality as NGE and RISE, but is optimized for learning speed (see [17] for more details).

2.3.5. *A comparison of memory requirements and generalization accuracy*

All mentioned learning algorithms and the pure memory-based learner IB1-IG were presented to the 675,745-case data set described at the onset of this section. They were applied to this data set using a 10-fold cross-validation setup, which means that the data set was divided ten times into a 90% training set and a 10% test set, and all algorithms were trained and tested on each of these ten partitionings. All reported results are averaged over the ten folds. The results are summarized in Table 2. The results show that the algorithm that uses most memory, the pure memory-based learner, IB1-IG, performs best. All other algorithms perform significantly worse, when analysed with one-tailed t-tests, with $p < 0.05$. Error back-propagation compresses most, and performs worst on test material; with more hidden units (200 rather than 50), generalization accuracy improves. C5.0 with default parameter yields a fair compression, but at the cost of about 1% in accuracy. Also, editing in IB1-IG leads to significant accuracy losses, even at the level of 1% editing. Only FAMBL, which merges cases in a careful manner, yields a generalization accuracy that is fairly close to that of IB1-IG; still, it suffers a significant loss.

3. CONCLUSIONS

On the basis of the unequivocal results from the empirical studies, we conclude that word pronunciation should not be modelled by an abstracting paradigm. In word pronunciation, there are no exceptions that can be reliably abstracted from: each atypical spelling-pronunciation mapping that a spelling system may hold (and even the most regular ones do, e.g. with loan words), is a productive one: the word it is a part of (or a morphologically derived or inflected form) may always reoccur later on. Abstracting methods discard noise but have no basis to distinguish real noise from these productive exceptions and therefore do not incorporate the latter into the induced model.

As an better alternative, we suggest a memory-based approach in which all available lexical pronunciation

Algorithm	Generalization accuracy %	Memory requirement Kilobytes
error back-propagation, 50 hidden units	87.86	105
error back-propagation, 200 hidden units	90.32	415
C5.0 decision trees	92.48	1187
50 % low CPS-editing in IB1-IG	59.50	890
10 % low CPS-editing in IB1-IG	88.25	1602
5 % low CPS-editing in IB1-IG	91.04	1691
1 % low CPS-editing in IB1-IG	93.01	1762
generalizing cases in IB1-IG (FAMBL)	93.22	1725
IB1-IG	93.45	1780

Table 2. Average generalization accuracies (percentages of correctly classified cases) and data storage memory requirements (Kilobytes) of five abstracting algorithms and pure memory-based learning (IB1-IG).

knowledge is stored as cases in memory. Such an approach can be made to generalise by combining memory lookup with analogical reasoning for previously unseen cases, thus constituting an "inductive lexicon" [5].

Acknowledgements

This research was done in the context of the "Induction of Linguistic Knowledge" (ILK) research programme, supported partially by the Netherlands Organization for Scientific Research (NWO) and by the Cooperation Association of Tilburg University and Eindhoven Technical University (SOBU).

REFERENCES

[1] D. W. Aha, D. Kibler, and M. Albert. Instance-based learning algorithms. *Machine Learning*, 6:37–66, 1991.

[2] R. H. Baayen, R. Piepenbrock, and H. van Rijn. *The CELEX lexical data base on CD-ROM*. Linguistic Data Consortium, Philadelphia, PA, 1993.

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

[4] W. Daelemans and A. Van den Bosch. A neural network for hyphenation. In I. Aleksander and J. Taylor, editors, *Artificial Neural Networks 2*, volume 2, pages 1647–1650, Amsterdam, 1992. North-Holland.

[5] W. Daelemans, A. Van den Bosch, and G. Durieux. Toward inductive lexicons: a case study. In *Proceedings of the LREC Workshop on Adapting Lexical and Corpus Resources to Sublanguages and Applications*, pages 29–35, Granada, Spain, 1998.

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

[7] W. Daelemans, A. Van den Bosch, and J. Zavrel. Forgetting exceptions is harmful in language learning. *Machine Learning*, 11:11–43, 1999.

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

[9] P. Domingos. Unifying instance-based and rule-based induction. *Machine Learning*, 24:141–168, 1996.

[10] P. E. Hart. The condensed nearest neighbor rule. *IEEE Transactions on Information Theory*, 14:515–516, 1968.

[11] J.R. Quinlan. C4.5*: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA, 1993.

[12] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. In D. E. Rumelhart and J. L. McClelland, editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, volume 1: Foundations, pages 318–362. The MIT Press, Cambridge, MA, 1986.

[13] S. Salzberg. *Learning with nested generalised exemplars*. Kluwer Academic Publishers, Norwell, MA, 1990.

[14] S. Salzberg. A nearest hyperrectangle learning method. *Machine Learning*, 6:277–309, 1991.

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

[16] A. Van den Bosch. *Learning to pronounce written words: A study in inductive language learning*. PhD thesis, Universiteit Maastricht, 1997.

[17] A. Van den Bosch. Careful abstraction from instance families in memory-based language learning. *Journal of Experimental and Theoretical Artificial Intelligence*, 1999. To appear.

[18] A. Van den Bosch and W. Daelemans. A distributed, yet symbolic model of text-to-speech processing. to appear in P. Broeder and J. Murre (Eds.), *Models of language learning: inductive and deductive approaches*. Cambridge, MA: MIT Press, 1997., forthcoming.

[19] A. Weijters. A simple look-up procedure superior to NETtalk? In *Proceedings of the International Conference on Artificial Neural Networks* - ICANN*-91, Espoo, Finland*, 1991.