Do Not Forget: Full Memory in Memory-Based Learning of Word Pronunciation *

Antal van den Bosch and Walter Daelemans
Tilburg University, ILK
P.O. Box 90153, NL-5000 LE Tilburg
The Netherlands
{antalb,walter}@kub.nl

Abstract

Memory-based learning, keeping full memory of learning material, appears a viable approach to learning NLP tasks, and is often superior in generalisation accuracy to eager learning approaches that abstract from learning material. Here we investigate three *partial* memory-based learning approaches which remove from memory specific task instance types estimated to be exceptional. The three approaches each implement one heuristic function for estimating exceptionality of instance types: (i) typicality, (ii) class prediction strength, and (iii) friendly-neighbourhood size. Experiments are performed with the memory-based learning algorithm IB1-IG trained on English word pronunciation. We find that removing instance types with low prediction strength (ii) is the only tested method which does not seriously harm generalisation accuracy. We conclude that keeping full memory of types rather than tokens, and excluding minority ambiguities appear to be the only performance-preserving optimisations of memory-based learning.

1 Introduction

Memory-based learning of classification tasks is a branch of supervised machine learning in which the learning phase consists simply of storing all encountered instances from a training set in memory (Aha, 1997). Memory-based learning algorithms do not invest effort during learning in abstracting from the training data, such as eager-learning (e.g., decision-tree algorithms, rule-induction, or connectionist-learning algorithms, (Quinlan, 1993; Mitchell, 1997)) do. Rather, they defer investing effort until new instances are presented. On being presented with an instance, a memory-based learning algorithm searches for a best-matching instance, or, more generically, a set of the k best-matching instances in memory. Having found such a set of k best-matching instances, the algorithm takes the (majority) class with which the instances in the set are labeled to be the class of the new instance. Pure memory-based learning algorithms implement the classic k-nearest neighbour algorithm (Cover and Hart, 1967; Devijver and Kittler, 1982; Aha, Kibler, and Albert, 1991); in different contexts, memory-based learning algorithms have also been named lazy, instance-based, exemplar-based, memory-based, case-based learning or reasoning (Stanfill and Waltz, 1986; Kolodner, 1993; Aha, Kibler, and Albert, 1991; Aha, 1997))

Memory-based learning has been demonstrated to yield accurate models of various natural language tasks such as grapheme-phoneme conversion, word stress assignment, part-of-speech tagging, and PP-attachment (Daelemans, Van den Bosch, and Weijters, 1997a). For example, the memory-based learning algorithm IB1-IG (Daelemans and Van den Bosch, 1992; Daelemans, Van den Bosch, and Weijters, 1997b), which extends the well-known IB1 algorithm (Aha, Kibler, and Albert, 1991) with an information-gain weighted similarity metric, has been demonstrated to perform adequately and, moreover, consistently and significantly better than *eager-learning* algorithms which do invest effort in abstraction during learning (e.g., decision-tree learning (Daelemans, Van den Bosch, and Weijters, 1997b; Quinlan, 1993), and connectionist learning (Rumelhart, Hinton, and Williams, 1986)) when trained and tested on a range of morpho-phonological tasks (e.g., morphological segmentation, grapheme-phoneme conversion, syllabification, and word stress assignment) (Daelemans, Gillis, and Durieux, 1994; Van den Bosch, Daelemans, and Weijters, 1996; Van den Bosch, 1997). Thus, when learning NLP tasks, the abstraction occurring in decision trees (i.e., the explicit *forgetting* of information considered to be redundant) and in connectionist networks (i.e., a non-symbolic encoding and decoding in relatively small numbers of connection

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

Antal van den Bosch and Walter Daelemans (1998) **Do Not Forget: Full Memory in Memory-Based Learning of Word Pronunciation.** In D.M.W. Powers (ed.) *NeMLaP3/CoNLL98: New Methods in Language Processing and Computational Natural Language Learning,* ACL, pp 195-204.

weights) both hamper accurate generalisation of the learned knowledge to new material.

These findings appear to contrast with the general assumption behind eager learning, that data representing real-world classification tasks tends to contains (i) redundancy and (ii) exceptions: redundant data can be compressed, yielding smaller descriptions of the original data; some exceptions (e.g., low-frequency exceptions) can (or should) be discarded since they are expected to be bad predictors for classifying new (test) material. However, both redundancy and exceptionality cannot be computed trivially; heuristic functions are generally used to estimate them (e.g., functions from information theory (Quinlan, 1993)). The lower generalisation accuracies of both decision-tree and connectionist learning, compared to memory-based learning, on the above-mentioned NLP tasks, suggest that these heuristic estimates may not be the best choice for learning NLP tasks. It appears that in order to learn such tasks successfully, a learning algorithm should not forget (i.e., explicitly remove from memory) any information contained in the learning material: it should not abstract from the individual instances.

An obvious type of abstraction that is not harmful for generalisation accuracy (but that is not always acknowledged in implementations of memory-based learning) is the straightforward abstraction from tokens to types with frequency information. In general, data sets representing natural language tasks, when large enough, tend to contain considerable numbers of duplicate sequences mapping to the same output or class. For example, in data representing word pronunciations, some sequences of letters, such as ing at the end of English words, occur hundreds of times, while each of the sequences is pronounced identically, viz. /ɪŋ/. Instead of storing all individual sequence tokens in memory, each set of identical tokens can be safely stored in memory as a single sequence type with frequency information, without loss of generalisation accuracy (Daelemans and Van den Bosch, 1992; Daelemans, Van den Bosch, and Weijters, 1997b). Thus, forgetting instance tokens and replacing them by instance types may lead to considerable computational optimisations of memory-based learning, since the memory that needs to be searched may become considerably smaller.

Given the safe, performance-preserving optimisation of replacing sets of instance tokens by instance types with frequency information, a next step of investigation into optimising memory-based learning is to measure the effects of *forgetting instance types* on grounds of their exceptionality, the underlying idea being that the more exceptional a task instance type is, the more likely it is that it is a bad predictor for new instances. Thus, exceptionality should in some way express the unsuitability of a task instance type to be a best match (nearest neighbour) to new instances: it would be unwise to copy its associated classification to best-matching new instances. In this paper, we investigate three criteria for estimating an instance type's exceptionality, and removing instance types estimated to be the most exceptional by each of these criteria. The criteria investigated are

1. typicality of instance types;
2. class prediction strength of instance types;
3. friendly-neighbourhood size of instance types;
4. random (to provide a baseline experiment).

We base our experiments on a large data set of English word pronunciation. We briefly describe this data set, and the way it is converted into an instance base fit for memory-based learning, in Section 2. In Section 3 we describe the settings of our experiments and the memory-based learning algorithm IB1-IG with which the experiments are performed. We then turn to describing the notions of typicality, class-prediction strength, and friendly-neighbourhood size, and the functions to estimate them, in Section 4. Section 5 provides the experimental results. In Section 6, we discuss the obtained results and formulate our conclusions.

2 The word-pronunciation data

Converting written words to stressed phonemic transcription, i.e., word pronunciation, is a well-known benchmark task in machine learning (Stanfill and Waltz, 1986; Sejnowski and Rosenberg, 1987; Shavlik, Mooney, and Towell, 1991; Dietterich, Hild, and Bakiri, 1990; Wolpert, 1990). We define the task as the conversion of fixed-sized instances representing parts of words to a class representing the phoneme and the stress marker of the instance's middle letter. To generate the instances, windowing is used (Sejnowski and Rosenberg, 1987). Table 1 displays example instances and their classifications generated on the basis of the sample word booking. Classifications, i.e., phonemes with stress markers (henceforth 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. In this study, we chose a fixed window width of seven letters, which offers sufficient context information for adequate performance, though extension of the window decreases ambiguity within the data set (Van den Bosch, 1997).

The task, henceforth referred to as GS (Grapheme-phoneme conversion and Stress assignment) is similar to the NETTALK task presented by Sejnowski and Rosenberg (1986), but is performed on a larger corpus of 77,565 English word-pronunciation pairs, extracted from the CELEX lexical data base (Burnage, 1990). Converted into fixed-sized instance, the

instance number	left context			focus letter	right context			classification
1	_	_	_	b	o	o	k	/b/1
2	_	_	b	o	o	k	i	/u/0
3	_	b	o	o	k	i	n	/-/0
4	b	o	o	k	i	n	g	/k/0
5	o	o	k	i	n	g	_	/ɪ/0
6	o	k	i	n	g	_	_	/ŋ/0
7	k	i	n	g	_	_	_	/-/0

Table 1: Example of instances generated for the word-pronunciation task from the word booking.

full instance base representing the GS task contains 675,745 instances. The task features 159 classes (combined phonemes and stress markers). The coding of the output as 159 atomic ('local') classes combining grapheme-phoneme conversion and stress assignment is one out of many types of output coding (Shavlik, Mooney, and Towell, 1991), e.g., distributed bit coding using articulatory features (Sejnowski and Rosenberg, 1987), error-correcting output coding (Dietterich, Hild, and Bakiri, 1990), or split discrete coding of grapheme-phoneme conversion and stress assignment (Van den Bosch, 1997). While these studies point at back-propagation learning (Rumelhart, Hinton, and Williams, 1986), using distributed output code, as the better performer as compared to ID3 (Quinlan, 1986), a symbolic inductive-learning decision tree algorithm (Dietterich, Hild, and Bakiri, 1990; Shavlik, Mooney, and Towell, 1991), unless ID3 was equipped with error-correcting output codes and additional manual tweaks (Dietterich, Hild, and Bakiri, 1990). Systematic experiments with the data also used in this paper have indicated that both back-propagation and decision-tree learning (using either distributed or atomic output coding) are consistently and significantly outperformed by memory-based learning of grapheme-phoneme conversion, stress assignment, and the combination of the two (Van den Bosch, 1997), using atomic output coding. Our choice for atomic output classes in the present study is motivated by the latter results.

3 Algorithm and experimental setup

3.1 Memory-based learning in IB1-IG

In the experiments reported here, we employ IB1-IG (Daelemans and Van den Bosch, 1992; Daelemans, Van den Bosch, and Weijters, 1997b), which has been demonstrated to perform adequately, and significantly better than eager-learning algorithms on the GS task (Van den Bosch, 1997). IB1-IG constructs an instance base during learning. An instance in the instance base consists of a fixed-length vector of n feature-value pairs (here, $n = 7$), an information field containing the classification of that particular feature-value vector, and an information field containing the occurrences of the instance with its classification in the full training set. The latter information field thus enables the storage of instance types rather than the more extensive storage of identical instance tokens. After the instance base is built, new (test) instances are classified by matching them to all instance types in the instance base, and by calculating with each match the *distance* between the new instance X and the memory instance type Y, $\Delta(X, Y)$, using the function given in Eq. 1:

$$\Delta(X,Y) = \sum_{i=1}^{n} W(f_i)\delta(X_i, Y_i), \qquad (1)$$

where $W(fi)$ is the weight of the ith feature, and $\delta(x_i, y_i)$ is the distance between the values of the ith feature in the instances X and Y. When the values of the instance features are symbolic, as with the GS task (i.e., feature values are letters), a simple distance function is used (Eq. 2):

$$\delta(X_i, Y_i) = 0 \; if \; X_i = Y_i \; else \; 1. \qquad (2)$$

The classification of the memory instance type Y with the smallest $\Delta(X, Y)$ is then taken as the classification of X. This procedure is also known as 1-NN, i.e., a search for the single nearest neighbour, the simplest variant of k-NN (Devijver and Kittler, 1982).

The weighting function of IB1-IG, $W(f_i)$, represents the *information gain* of feature f_i. Weighting features in k-NN classifiers such as IB1-IG is an active field of research (cf. (Wettschereck, 1995; Wettschereck, Aha, and Mohri, 1997), for comprehensive overviews and discussion). Information gain is a function from information theory also used in ID3 (Quinlan, 1986) and C4.5 (Quinlan, 1993). The information gain of a feature expresses its relative relevance compared to the other features when performing the mapping from input to classification.

The idea behind computing the information gain of features is to interpret the training set 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 3, where p_i (the probability of classification i) is estimated by its relative frequency in the training set.

$$H(D) = -\sum_i p_i log_2 p_i \quad (3)$$

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 average information entropy for a feature f_i, given in equation 4, we take the average information entropy of the data base restricted to each possible value for the feature. The expression $D_{[f_i=v_j]}$ 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.

$$H(D_{[f_i]}) = \sum_{v_j \in V} H(D_{[f_i=v_j]}) \frac{|D_{[f_i=v_j]}|}{|D|} \quad (4)$$

Information gain of feature f_i is then obtained by equation 5.

$$G(f_i) = H(D) - H(D_{[f_i]}) \quad (5)$$

Using the weighting function $W(f_i)$ acknowledges the fact that for some tasks, such as the current GS task, some features are far more relevant (important) than other features. Using it, instances that match on a feature with a relatively high information gain are regarded as less distant (more alike) than instances that match on a feature with a lower information gain.

Finding a nearest neighbour to a test instance may result in two or more candidate nearest-neighbour instance types at an identical distance to the test instance, yet associated with different classes. The implementation of IB1-IG used here handles such cases in the following way. First, IB1-IG selects the class with the highest occurrence within the merged set of classes of the best-matching instance types. In case of occurrence ties, the classification is selected that has the highest overall occurrence in the training set. (Daelemans, Van den Bosch, and Weijters, 1997b).

3.2 Setup

We performed a series of experiments in which IB1-IG is applied to the GS data set, systematically edited according to each of the three tested criteria (plus the baseline random criterion) described in the next section. We performed the following global procedure:

1. We partioned the full GS data set into a training set of 608,228 instances (90% of the full data set) and a test set of 67,517 instances (10%). For use with IB1-IG, which stores instance types rather than instance tokens, the data set was reduced to contain 222,601 instance types (i.e., unique combinations of feature-value vectors and their classifications), with frequency information.
2. For each exceptionality criterion (i.e., typicality, class prediction strength, friendly-neighbourhood size, and random selection),
   (a) we created four edited instance bases by removing 1%, 2%, 5%, and 10% of the most exceptional instance types (according to the criterion) from the training set, respectively.
   (b) For each of these increasingly edited training sets, we performed one experiment in which IB1-IG was trained on the edited training set, and tested on the original unedited test set.

4 Three estimations of exceptionality

We investigate three methods for estimating the (degree of) exceptionality of instance types: typicality, class prediction strength, and friendly-neighbourhood size.

4.1 Typicality

In its common meaning, "typicality" denotes roughly the opposite of exceptionality; atypicality can be said to be a synonym of exceptionality. We adopt a definition from (Zhang, 1992), who proposes a typicality function. Zhang computes typicalities of instance types by taking both their feature values and their classifications into account (Zhang, 1992). He adopts the notions of *intra-concept similarity* and *inter-concept similarity* (Rosch and Mervis, 1975) to do this. First, Zhang introduces a distance function similar to Equation 1, in which $W(f_i) = 1.0$ for all features (i.e., flat Euclidean distance rather than information-gain weighted distance), in which the distance between two instances X and Y is normalised by dividing the summed squared distance by n, the number of features, and in which $\delta(x_i, y_i)$ is given as Equation 2. The normalised distance function used by Zhang is given in Equation 6.

$$\Delta(X,Y) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(W(f_i)\delta(x_i,y_i))^2} \quad (6)$$

The intra-concept similarity of instance X with classification C is its similarity (i.e., $1-$distance) with all instances in the data set with the same classification C: this subset is referred to as X's *family*, $Fam(X)$. Equation 7 gives the intra-concept similarity function $Intra(X)$ ($|Fam(X)|$ being the number of instances in X's family, and $Fam(X)^i$ the ith instance in that family).

$$Intra(X) = \frac{1}{|Fam(X)|} \sum_{i=1}^{|Fam(X)|} 1.0 - \Delta(X, Fam(X)^i) \quad (7)$$

All remaining instances belong to the subset of unrelated instances, $Unr(X)$. The inter-concept similarity of an instance X, $Inter(X)$, is given in Equation 8 (with $|Unr(X)|$ being the number of instances unrelated to X, and $Unr(X)^i$ the ith instance in that subset).

$$Inter(X) = \frac{1}{|Unr(X)|} \sum_{i=1}^{|Unr(X)|} 1.0 - \Delta(X, Unr(X)^i) \quad (8)$$

The typicality of an instance X, $Typ(X)$, is the quotient of X's intra-concept similarity and X's inter-concept similarity, as given in Equation 9.

$$Typ(X) = \frac{Intra(X)}{Inter(X)} \quad (9)$$

An instance type is typical when its intra-concept similarity is larger than its inter-concept similarity, which results in a typicality larger than 1. An instance type is atypical when its intra-concept similarity is smaller than its inter-concept similarity, which results in a typicality between 0 and 1. Around typicality value 1, instances cannot be sensibly called typical or atypical; (Zhang, 1992) refers to such instances as *boundary* instances.

In our experiments, we compute the typicality of all instance types in the training set, order them on their typicality, and remove 1%, 2%, 5%, and 10% of the instance types with the lowest typicality, i.e., the most atypical instance types. In addition to these four experiments, we performed an additional eight experiments using the same percentages, and editing on the basis of (i) instance types' typicality (by ordering them in reverse order) and (ii) their indifference towards typicality or atypicality (i.e., the closeness of their typicality to 1.0, by ordering them in order of the absolute value of their typicality subtracted by 1.0). The experiments with removing typical and boundary instance types provide interesting comparisons with the more intuitive editing of atypical instance types.

Table 2 provides examples of four atypical, boundary, and typical instance types found in the training set. Globally speaking, (i) the set of atypical instances tend to contain foreign spellings of loan words; (ii) there is no clear characteristic of boundary instances; and (iii) 'certain' pronunciations, i.e., instance types with high typicality values often involve instance types of which the middle letters are at the beginning of words or immediately following a hyphen, or high-frequency instance types, or instance types mapping to a low-frequency class that always occurs with a certain spelling (class frequency is not accounted for in Zhang's metric).

4.2 Class-prediction strength

A second estimate of exceptionality is to measure how well an instance type predicts the class of all instance types within the training set (including itself). Several functions for computing class-prediction strength have been proposed, e.g., as a criterion for removing instances in memory-based (k-nn) learning algorithms, such as IB3 (Aha, Kibler, and Albert, 1991) (cf. earlier work on edited k-nn (Wilson, 1972; Voisin and Devijver, 1987)); or for weighting instances in the EACH algorithm (Salzberg, 1990; Cost and Salzberg, 1993). We chose to implement the straightforward class-prediction strength function as proposed in (Salzberg, 1990) in two steps. First, we count (a) the number of times that the instance type is the nearest neighbour of another instance type, and (b) the number of occurrences that when the instance type is a nearest neighbour of another instance type, the classes of the two instances match. Second, the instance's class-prediction strength is computed by taking the ratio of (b) over (a). An instance type with class-prediction strength 1.0 is a perfect predictor of its own class; a class-prediction strength of 0.0 indicates that the instance type is a bad predictor of classes of other instances, presumably indicating that the instance type is exceptional.

We computed the class-prediction strength of all instance types in the training set, ordered the instance types according to their strengths, and created edited training sets with 1%, 2%, 5%, and 10% of the instance types with the lowest class prediction strength removed, respectively. In Table 3, four sample instance types are displayed which have class-prediction strength 0.0, i.e., the lowest possible strength. They are never a correct nearest-neighbour match, since they all have higher-frequency counterpart types with the same feature values. For example, the letter sequence ___algo occurs in two types, one associated with the pronunciation /'æ/ (viz., primary-stressed /æ/, or 1æ in our labelling), as in algorithm and algorithms; the other associated with the pronunciation /"æ/ (viz. secondary-stressed /æ/ or 2æ), as in algorithmic. The latter instance type occurs less frequently than the former, which is the reason that the class of the former is preferred over the latter. Thus, an ambiguous type with a minority class (a *minority ambiguity*) can never be a correct predictor, not even

instance types								
atypical			boundary			typical		
feature values	class	typicality	feature values	class	typicality	feature values	class	typicality
ureaucr	0əʊ	0.428	cheques	0ks	1.000	___oilf	1ɔɪ	7.338
freudia	0ɔɪ	0.442	elgium_	0-	1.000	etectio	0kʃ	8.452
_tissue	0ʃ	0.458	laby___	0aɪ	1.000	ow-by-b	0b	9.130
__czech	0-	0.542	manna__	0-	1.000	ng-iron	2aɪə	12.882

Table 2: Examples of atypical (left), boundary (middle), and typical (left) instance types in the training set. For each instance (seven letters and a class mapping to the middle letter), its typicality value is given.

feature values	class	cps
___algo	2æ	0.0
ck-benc	1b	0.0
erby___	0aɪ	0.0
reface_	0eɪ	0.0

Table 3: Examples of instance types with the lowest possible class prediction strength (cps) 0.0.

feature values	class	fns
___edib	2ɛ	0
___edib	1ɛ	0
echnocr	1n	0
soirée_	0r	0

Table 5: Examples of instance types with the lowest possible friendly-neighbourhood size (fns) 0, i.e., no friendly neighbours.

for itself, when using IB1-IG as a classifier, which always prefers high frequency over low frequency in case of ties.

4.3 Friendly-neighbourhood size

A third estimate for the exceptionality of instance types is counting by how many nearest neighbours of the same class an instance type is surrounded in instance space. Given a training set of instance types, for each instance type a ranking can be made of all of its nearest neighbours, ordered by their distance to the instance type. The number of nearest-neighbour instance types in this ranking with the same class, henceforth referred to as the friendly-neighbourhood size, may range between 0 and the total number of instance types of the same class. When the friendly neighbourhood is empty, the instance type only has nearest neighbours of different classes. The argumentation to regard a small friendly neighbourhood as an indication of an instance type's exceptionality, follows from the same argumentation as used with class-prediction strength: when an instance type has nearest neighbours of different classes, it is vice versa a bad predictor for those classes. Thus, the smaller an instance type's friendly neighbourhood, the more it could be regarded exceptional.

To illustrate the computation of friendly-neighbourhood size, Table 4 lists four examples of possible nearest-neighbour rankings (truncated at ten nearest neighbours) with their respective number of friendly neighbours. The Table shows that the number of friendly neighbours is the number of similarly-labeled instances counted from left to right in the ranking, until a dissimilarly-labeled instance occurs.

Friendly-neighbourhood size and class-prediction strength are related functions, but differ in their treatment of class ambiguity. As stated above, instance types may receive a class-prediction strength of 0.0 when they are minority ambiguities. Counting a friendly neighbourhood does not take class ambiguity into account; each of a set of ambiguous types necessarily has no friendly neighbours, since they are eachother's nearest neighbours with different classes. Thus, friendly-neighbourhood size does not discriminate between minority and majority ambiguities. In Table 5, four sample instance types are listed with friendly-neighbourhood size 0. While some of these instance types without friendly neighbours in the training set (perhaps with friendly neighbours in the test set) are minority ambiguities (e.g., ___edib 2ɛ), others are majority ambiguities (e.g., ___edib 1ɛ), while others are not ambiguous at all but simply have a nearest neighbour at some distance with a different class (e.g., soirée_ 0r).

5 Results

Figure 1 displays the generalisation accuracies in terms of incorrectly classified test instances obtained with all performed experiments. The leftmost point in the Figure, from which all lines originate, indicates the performance of IB1-IG when trained on the full data set of 222,601 types, viz. 6.42% incorrectly classified test instances (when computed in terms of incorrectly pronounced test words, IB1-IG pronounces 64.61 of all test words flawlessly).

The line graph representing the four experiments in which instance types are removed randomly can be seen as the baseline graph. It can be expected

nearest neighbour rank #										#
1	2	3	4	5	6	7	8	9	10	fn
$\circ$ 1	$\times$ 2	$\circ$ 3	$\circ$ 3	$\circ$ 3	$\circ$ 4	$\times$ 4	$\times$ 5	$\times$ 5	$\times$ 6	1
$\circ$ 1	$\circ$ 1	$\circ$ 1	$\circ$ 1	$\circ$ 1	$\circ$ 1	$\circ$ 2	$\circ$ 2	$\circ$ 3	$\times$ 4	9
$\times$ 2	$\circ$ 2	$\circ$ 2	$\circ$ 2	$\circ$ 2	$\circ$ 2	$\times$ 3	$\times$ 3	$\times$ 3	$\times$ 4	0
$\circ$ 1	$\circ$ 1	$\circ$ 1	$\times$ 3	$\times$ 4	$\times$ 4	$\times$ 4	$\times$ 4	$\times$ 5	$\circ$ 6	3

Table 4: Four examples of nearest-neighbour rankings and their respective numbers of friendly neighbours (fn). Each ranked nearest neighbour is identified by its match ($\circ$) or mismatch ($\times$) with the target instance the ranking is computed for, and a number denoting its distance to the target instance.

that removing instances randomly leads to a degradation of generalisation performance. The upward curve of the line graph denoting the experiments with random selection indeed shows degrading performance with increasing numbers of left-out instance types. The relative decrease in generalisation accuracy is 2.0% when 1% of the training material is removed randomly, 3.8% with 2% random removal, 10.7% with 5% random removal, and 20.7% with 10% random removal.

Surprisingly, the only experiments showing lower performance degradation than removal by random selection are those with class-prediction strength; the other criteria for removing exceptional instances lead to *worse* degradations. It does not matter whether instance types are removed on grounds of their typicality: apparently, a markedly low, neutral, or high typicality value indicates that the instance type is (on average) important, rather than removable. The same applies to friendly-neighbourhood size: instances with small neighbourhood sizes appear to contribute significantly to performance on test material. It is remarkable that the largest errors with 1% and 2% removal are obtained with the friendly-neighbourhood size criterion: it appears that on average, the instances with few or no nearest neighbours are important in the classification of test material.

When using class-prediction strength as removal criterion, performance does not degrade until about 5% of the instance types with the lowest strength are removed from memory. The reason is that class-prediction strength is the only criterion that detects minority ambiguities, i.e., instance types with prediction strength 0.0, that cannot contribute to classification since they are always overshadowed by their counterpart instance types with majority classes, even for their own classification. In the training set, 9,443 instance types are minority ambiguities, i.e., 4.2% of the instance types (accounting for 3.8% of the instance tokens in the original token set).

Thus, among the tested methods for reducing the memory needed for storing an instance base in memory-based learning, only two relatively trivial methods are performance-preserving while accounting for a substantial reduction in the amount of memory needed by IB1-IG:

1. Replacing instance tokens by instance types accounts for a reduction of about 63% of memory needed to store instances, excluding the memory needed to store frequency information. When frequency information is stored in two bytes per instance type, the memory reduction is about 54%.

2. Removing instance types that are minority ambiguities on top of the type/token-reduction accounts only for an additional memory reduction of 2%, i.e., for a total memory reduction of 65%; 56% with two-byte frequency information stored per instance.

6 Discussion and future research

As previous research has suggested (Daelemans, 1996; Daelemans, Van den Bosch, and Weijters, 1997a; Van den Bosch, 1997), keeping full memory in memory-based learning of word pronunciation strongly appears to yield optimal generalisation accuracy. The experiments in this paper show that optimisation of memory use in memory-based learning while preserving generalisation accuracy can only be performed by (i) replacing instance tokens by instance types with frequency information, and (ii) removing minority ambiguities. Both optimisations can be performed straightforwardly; minority ambiguities can be traced with less effort than by using class-prediction strength. Our implementation of IB1-IG described in (Daelemans and Van den Bosch, 1992; Daelemans, Van den Bosch, and Weijters, 1997b) already makes use of this knowledge, albeit partially (it stores class distributions with letter-window types).

Our results also show that atypicality, non-typicality, and typicality (Zhang, 1992), and friendly-neighbourhood size are all estimates of exceptionality that indicate the importance of instance types for classification, rather than their removability. As far as these estimates of exceptionality are viable, our results suggest that exceptions should be kept in memory and not be thrown away.

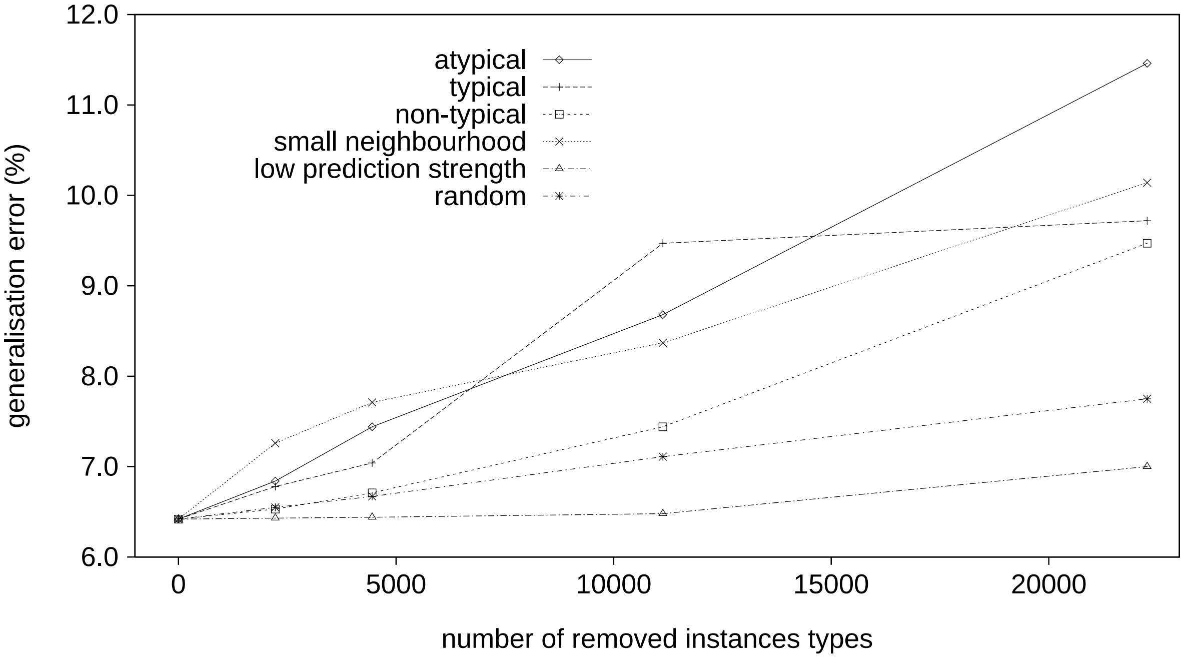

Figure 1: Generalisation errors (percentages of incorrectly classified test instances of TRIBL-IG, with increased numbers of edited instances, according to the tested exceptionality criteria atypical, typical, boundary, small neighbourhood, low prediction strength, and random selection. Performances, denoted by points, are measured when 1%, 2%, 5%, and 10% of the most exceptional instance types are edited.

Lazy vs. eager; not stable vs. unstable

From the results in this paper and those reported earlier (Daelemans, Van den Bosch, and Weijters, 1997a; Van den Bosch, 1997), it appears that no compromise can be made on memory-base learning in terms of abstraction by forgetting without losing generalisation accuracy. Consistently lower performances are obtained with algorithms that forget by constructing decision trees or connectionist networks, or by editing instance types. Generalisation accuracy appears to be related to the dimension *lazy-eager* learning; for the GS task (and for many other language tasks, (Daelemans, Van den Bosch, and Weijters, 1997a)), it is demonstrated that memory-based lazy learning leads to the best generalisation accuracies.

Another explanation for the difference in performance between decision-tree, connectionist, and editing methods versus pure memory-based learning is that the former generally display high *variance*, which is the portion of the generalisation error caused by the *unstability* of the learning algorithm (Breiman, 1996a). An algorithm is unstable when small perturbations in the learning material lead to large differences in induced models, and stable otherwise; pure memory-based learning algorithms are said to be very stable, and decision-tree algorithms and connectionist learning to be unstable (Breiman, 1996a). High variance is usually coupled with low bias, i.e., unstable learning algorithms with high variance tend to have few limitations in the freedom to approximate the task or function to be learned) (Breiman, 1996b). Breiman points out that often the opposite also holds: a stable classifier with a low variance can display a high bias when it cannot represent data adequately in its available set of models, but it is not clear whether or how this applies to pure memory-based learning as in IB1-IG; its success in representing the GS data and other language tasks quite adequately would rather suggest that IB1-IG has both low variance and low bias. Apart from the possibility that the lazy and eager learning algorithms investigated here and in earlier work do not have a strongly contrasting bias, we conjecture that the editing methods discussed here, and some specific decision-tree learning algorithms investigated earlier (i.e., IGTREE (Daelemans, Van den Bosch, and Weijters, 1997b), a decision tree learning algorithm that is an approximate optimisation of IB1-IG) have a similar variance to that of IB1-IG; they are virtually as stable as IB1-IG. We base this conjecture on the fact that the standard deviations of both decision-tree learning and memory-based learning trained and tested on the GS data are not only very small (in the order of 1/10 percents), but also hardly different (cf. (Van den Bosch, 1997) for details and examples). Only connectionist networks trained with back-propagation and decision-tree learning with pruning display larger standard deviations when accuracies are averaged over experiments (Van den Bosch, 1997); the stable-unstable dimension might play a role there, but not in the difference between pure memory-based learning and edited memory-based learning.

Future research

The results of the present study suggest that the following questions be investigated in future research:

- The tested criteria for editing can be employed as instance weights as in EACH (Salzberg, 1990) and PEBLS (Cost and Salzberg, 1993), rather than as criteria for instance removal. Instance weighting, preserving pure memory-based learning, may add relevant information to similarity matching, and may improve IB1-IG's performance.
- Different data sets of different sizes may contain different portions of atypical instances or minority ambiguities. Moreover, data sets may contain pure noise. While atypical or exceptional instances may (and do) return in test material, the chances of noise to return is relatively minute. Our results generalise to data sets with approximately the characteristics of the GS dataset. Although there are indications that data sets representing other language tasks indeed share some essential characteristics (e.g., memory-based learning is consistently the best-performing algorithm), more investigation is needed to make these characteristics explicit.

Acknowledgements

We thank the members of the ILK group, Ton Weijters, and Eric Postma for fruitful discussions, and the anonymous reviewers for relevant comments and suggestions.

References

Aha, D. W., editor. 1997. *Lazy learning.* Dordrecht: Kluwer Academic Publishers. reprinted from: *Artificial Intelligence Review*, **11**:1–5.

Aha, D. W., D. Kibler, and M. Albert. 1991. Instance-based learning algorithms. *Machine Learning*, 7:37–66.

Breiman, L. 1996a. Bagging predictors. *Machine Learning*, 24(2).

Breiman, L. 1996b. Bias, variance and arcing classifiers. Technical Report 460, University of California, Statistics Department, Berkeley, CA.

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

Cost, S. and S. Salzberg. 1993. A weighted nearest neighbor algorithm for learning with symbolic features. *Machine Learning*, 10:57–78.

Cover, T. M. and P. E. Hart. 1967. Nearest neighbor pattern classification. *Institute of Electrical and Electronics Engineers Transactions on Information Theory*, 13:21–27.

Daelemans, W. 1996. Abstraction considered harmful: lazy learning of language processing. In H. J. Van den Herik and A. Weijters, editors, *Proceedings of the Sixth Belgian–Dutch Conference on Machine Learning*, pages 3–12, Maastricht, The Netherlands. MATRIKS.

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., A. Van den Bosch, and A. Weijters. 1997a. Empirical learning of natural language processing tasks. Lecture Notes in Artificial Intelligence, , number 1224, pages 337–344. Berlin: Springer-Verlag.

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

Devijver, P. .A. and J. Kittler. 1982. *Pattern recognition. A statistical approach.* London, UK: Prentice-Hall.

Dietterich, T. G., H. Hild, and G. Bakiri. 1990. A comparison of ID3 and backpropagation for English text-to-speech mapping. Technical Report 90–20–4, Oregon State University.

Kolodner, J. 1993. *Case-based reasoning.* San Mateo, CA: Morgan Kaufmann.

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

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.

Rosch, E. and C. B. Mervis. 1975. Family resemblances: studies in the internal structure of categories. *Cognitive Psychology*, 7:??–??

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

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

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., W. Daelemans, and A. Weijters. 1996. Morphological analysis as classification: an inductive-learning approach. In K. Oflazer and H. Somers, editors, *Proceedings of the Second International Conference on New Methods in Natural Language Processing, NeMLaP-2, Ankara, Turkey*, pages 79–89.

Voisin, J. and P. A. Devijver. 1987. An application of the Multiedit-Condensing technique to the reference selection problem in a print recognition system. *Pattern Recognition*, 5:465–474.

Wettschereck, D. 1995. *A study of distance-based machine-learning algorithms.* Ph.D. thesis, Oregon State University.

Wettschereck, D., D. W. Aha, and T. Mohri. 1997. A review and empirical evaluation of feature weighting methods for a class of lazy learning algorithms. *Artificial Intelligence Review*, 11:273–314.

Wilson, D. 1972. Asymptotic properties of nearest neighbor rules using edited data. *Institute of Electrical and Electronic Engineers Transactions on Systems, Man and Cybernetics*, 2:408–421.

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

Zhang, J. 1992. Selecting typical instances in instance-based learning. In *Proceedings of the International Machine Learning Conference 1992*, pages 470–479.