Careful Abstraction from Instance Families in Memory-Based Language Learning*

Antal van den Bosch
ILK Research Group (http://ilk.kub.nl)
Computational Linguistics, Tilburg University
P.O. Box 90153, NL-5000 LE Tilburg, The Netherlands
email Antal.vdnBosch@kub.nl

Abstract

Empirical studies in inductive language learning point at pure memory-based learning as a successful approach to many language learning tasks, often performing better than learning methods that abstract from the learning material. The possibility is left open, however, that limited, careful abstraction in memory-based learning may be harmless to generalization, as long as the disjunctivity of language data is preserved. We test this hypothesis, by comparing empirically a range of careful abstraction methods, focusing particularly on methods that (i) generalize instances and (ii) perform oblivious (partial) decision-tree abstraction. These methods are applied to a selection of language learning tasks, and their generalization performance as well as memory item compression rates are collected. On the basis of the results we conclude that when combined with feature weighting or value distance metrics, careful abstraction equals or outperforms pure memory-based learning, yet mainly on small data sets. In the concluding case study involving large data sets, we find that the FAMBL algorithm, a new careful abstractor which merges families of instances, performs close to pure memory-based learning, though it equals it only on three of the six tasks. On the basis of the gathered empirical results, we discuss the incorporation of the notion of *instance families*, i.e. carefully generalized instances, in memory-based language learning.

1 Introduction

Memory-based learning has been studied for some time now as an approach to learning language processing tasks, and is found by various studies to be successful, attaining adequate to excellent generalization accuracies on realistic, complex tasks as different as hyphenation, semantic parsing, part-of-speech tagging, morphological segmentation, and word pronunciation (Daelemans and Van den Bosch, 1992a; Cardie, 1994; Cardie, 1996; Daelemans et al., 1996; Van den Bosch, 1997). Recent studies in inductive language learning (Van den Bosch, 1997; Daelemans, Van den Bosch, and Zavrel, 1999) provide indications that forgetting task instances during learning tends to hinder generalization accuracy of the trained classifiers, especially when these instances are estimated to be exceptional. Learning algorithms that do not forget anything about the learning material, i.e. pure memory-based learning algorithms, are found to obtain the best accuracies for the tasks studied when compared to decision-tree or edited memory-based learning algorithms, which tend to 'forget'. The detrimental effect of forgetting exceptions is especially witnessed in word pronunciation, although it is not fully clear yet what distinguishes this task in this respect from other language learning tasks such as part-of-speech tagging, base-NP chunking, or prepositional-phrase attachment.

Learning algorithms equipped with the ability to forget (e.g. editing in IB3, Aha, Kibler, and Albert (1991), or pruning in C4.5, Quinlan (1993)) do so quite strongly by default.

*To appear in W. Daelemans (guest editor), *Journal of Experimental and Theoretical Artificial Intelligence*. Special Issue on Memory-Based Language Processing.

For example, if its default parameter values are not overruled by the user, C4.5 tends to perform a considerable amount of abstraction (i.e. decision-tree pruning) when trained on typical language learning tasks. Although this default bias tend to produce small trees which allow very fast classification, generalization performance on some language learning tasks is markedly worse as compared to that of memory-based classifiers. However, tuning these parameters towards less pruning in decision trees or limited forgetting of instances in edited memory-based learning, i.e. *careful* (or weak) abstraction, can yield performances that are close or equal to those of pure memory-based learning, at least for some language learning tasks (Daelemans, Van den Bosch, and Zavrel, 1999).

Thus, findings from recent studies support the hypothesis that careful abstraction in inductive language learning may be an equal alternative, at least for some language learning tasks, to pure memory-based learning. Advantages of the careful abstraction approach over pure memory-based learning may be that it may produce a more compact model of the learning material, and that it may better reflect some linguistic aspects of the data. Although computational efficiency is not at the focus of this article, it can also be hypothesized that more compact representations of the learning material allow faster classification. The topic of this article is to investigate existing approaches to careful abstraction in memory-based learning, and to perform empirical tests on language learning tasks to collect indications for the efficacy of careful abstraction in memory-based language learning.

We use the term abstraction here as denoting the *forgetting of learning material during learning*. This *material* notion of abstraction is not to be confused with the *informational* abstraction from learning material exhibited by *weighting metrics* (Salzberg, 1991; Cost and Salzberg, 1993; Wettschereck, Aha, and Mohri, 1997), and, more generally, by the abstraction bias in all memory-based learning approaches that high similarity between instances is to be preferred over lower similarity, and that only the most similar items are to be used as information source for extrapolating classifications. Informational abstraction does not lead to the explicit removal of data from memory; material abstraction does[1].

In this article, comparisons are made among memory-based learning algorithms that feature both material abstraction and weighting metrics. Where possible, both types of abstraction are separated to allow for the comparison of minimal pairs of algorithmic settings, and a proper discussion of the results. The empirical part of this article describes two case studies in which only parts of the total experimental matrix between algorithms, abstraction methods, metrics, and language learning tasks are covered; it is hoped that these case studies provide indications that support the broader, secondary goal of the paper: to show that apart from instances, memory-based language learning may consider *instance families*, i.e. carefully generalized instances, as working units in learning and processing.

The article is structured as follows. Section 2 summarizes methods for careful abstraction in memory-based learning; it reviews existing approaches, and presents FAMBL, a new memory-based learning algorithm that abstracts carefully by merging (families of) instances in memory. In Section 3 a range of memory-based learning algorithms performing careful abstraction is applied to the task of grapheme-phoneme conversion. While this task is represented by a relatively small data set, the second series of experiments, described in Section 4, investigates three careful abstraction algorithms applied to data sets with systematically increased magnitude, of grapheme-phoneme conversion, word pronunciation, and morphological segmentation. Concluding the experimental body of the article, Section 5 deals with the application of FAMBL, in comparison with its parent (pure memory-based) learning algorithm IB1-IG, and IGTREE, a decision-tree optimization of IB1-IG, to six language learning tasks represented by large data sets of examples. In Section 6, the efficacy of careful generalization over families of instances is discussed, and the idea of viewing these families as linguistic units is outlined. Finally, Section 6 identifies future research.

2 Careful abstraction in memory-based learning

Memory-based learning, also known as instance-based, example-based, lazy, case-based, exemplar-based, locally weighted, and analogical learning (Stanfill and Waltz, 1986; Aha, Kibler,

[1] Of course, informational abstraction becomes material abstraction when it expresses the fact that some part of the data has no (zero-level) information.

and Albert, 1991; Salzberg, 1991; Kolodner, 1993; Aha, 1997; Atkeson, Moore, and Schaal, 1997), is a class of supervised inductive learning algorithms for learning classification tasks (Shavlik and Dietterich, 1990). Memory-based learning treats a set of labeled (pre-classified) training instances as points in a multi-dimensional feature space, and stores them as such in an *instance base* in memory (rather than performing some abstraction over them).

An instance consists of a fixed-length vector of n feature-value pairs, and an information field containing the classification of that particular feature-value vector. After the instance base is built, new (test) instances are classified by matching them to all instances in the instance base, and by calculating with each match the *distance*, given by a distance function $\Delta(X, Y)$ between the new instance X and the memory instance Y. The memory instances with the smallest distances are collected, and the classifications associated with these nearest neighbors are merged and extrapolated to assign a classification to the test instance.

The most basic distance function for patterns with symbolic features is the *overlap metric* given in Equations 1 and 2; where $\Delta(X, Y)$ is the distance between patterns X and Y, represented by n features, and δ is the distance between feature values.

$$\Delta(X, Y) = \sum_{i=1}^{n} \delta(x_i, y_i) \tag{1}$$

where:

$$\delta(x_i, y_i) = 0 \; if \; x_i = y_i, \; else \; 1 \tag{2}$$

Classification in memory-based learning systems is basically performed by the k-nearest neighbor (k-NN) classifier (Cover and Hart, 1967; Devijver and Kittler, 1982), with k usually set to 1.

Early work on the k-NN classifier pointed at advantageous properties of the classifier in terms of generalization accuracies, under certain assumptions, because of its reliance on full memory (Fix and Hodges, 1951; Cover and Hart, 1967). However, the trade-off downside of full memory is computational inefficiency of the classification process, as compared to parametric classifiers that do abstract from the learning material. Therefore, several early investigations were performed into *editing* methods: finding criteria for the removal of instances from memory (Hart, 1968; Gates, 1972) without harming classification accuracy. Other studies on editing also explored the possibilities of detecting and removing noise from the learned data, so that classification accuracy might even improve (Wilson, 1972; Devijver and Kittler, 1980).

The renewed interest in the k-NN classifier from the late 1980s onwards in the AI-subfield of machine learning (Stanfill and Waltz, 1986; Stanfill, 1987; Aha, Kibler, and Albert, 1991; Salzberg, 1991) caused several new implementations of ideas on criteria for editing, but also other approaches to abstraction in memory-based learning emerged. In this section we start with a brief overview of approaches to *editing of instances during learning*. We then discuss one approach in which memory-based learning is optimized using *oblivious (partial) decision-tree search*. We conclude our overview with a discussion of two approaches that *carefully merge instances* into more general expressions. Consequently we present FAMBL, a carefully-abstracting memory-based learning algorithm. FAMBL merges groups of very similar instances (families) into family expressions.

As a side-note, we mention that in this section we use *grapheme-phoneme conversion* as a benchmark language learning task from which examples are drawn illustrating the functioning of the described approaches. The task is also in focus in Section 3. Grapheme-phoneme conversion is a well-known benchmark task in machine learning (Sejnowski and Rosenberg, 1987; Stanfill and Waltz, 1986; Stanfill, 1987; Lehnert, 1987; Wolpert, 1989; Shavlik, Mooney, and Towell, 1991; Dietterich, Hild, and Bakiri, 1995). We define the task as the conversion of fixed-sized instances representing parts of words to a class representing the phoneme of the instance's middle letter. To generate the instances, windowing is used (Sejnowski and Rosenberg, 1987). Table 1 displays four example instances and their classifications. For example, the first instance in table 1, _hearts_ (the '_' denotes empty outside-word positions), maps to class label /A:/, denoting an elongated short 'a'-sound to which the middle letter 'a' maps. In this study, we chose a fixed window width of nine letters, which offers sufficient context information for adequate performance (in terms of the upper bound on error demanded by applications in speech technology) (Van den Bosch, 1997).

Features									
1	2	3	4	5	6	7	8	9	Class
_	_	h	e	a	r	t	s	_	/A:/
_	b	o	o	k	i	n	g	_	/k/
i	t	i	e	s	_	_	_	_	/z/
_	_	_	_	b	a	s	i	c	/b/

Table 1: Example instances of the grapheme-phoneme conversion learning task. All instances are characterized by nine features (letters) and one class label, which is the phonemic mapping of the middle letter of the instance.

The task is deliberately picked for illustration purposes because it has been claimed and demonstrated that pure memory-based learning is successful in learning it (Stanfill and Waltz, 1986; Wolpert, 1989), also when compared to learning methods that abstract (Van den Bosch, 1997; Daelemans, Van den Bosch, and Zavrel, 1999). The task appears more sensitive to harmful effects of abstraction than other tasks investigated in the literature, and consequently, abstraction should be especially careful in learning grapheme-phoneme conversion. This task bias, upheld for now for the purpose of illustration, is abandoned in Sections 4 and 5 when series of experiments on other language learning tasks are described.

2.1 Existing approaches

We distinguish between three types of careful abstraction during learning:

1. **Editing** (Hart, 1968; Wilson, 1972; Aha, Kibler, and Albert, 1991): removing instances according to a classification-related utility threshold they do not reach. Editing is not careful in principle, but the approaches that are discussed here and that are included in the empirical comparison (i.e. IB2 and IB3, Aha, Kibler, and Albert (1991)) collect statistical support for an editing operation to be harmless.
2. **Oblivious (partial) decision-tree abstraction** (Daelemans, Van den Bosch, and Weijters, 1997): compressing (parts of) instances in the instance base into (parts of) decision-trees. Part of the motivation to perform top-down induction of decision trees (TDIDT) is the presence of clear differences in the relative importance of instance features, allowing features to be strictly ordered in matching (Quinlan, 1986). The approach is dependent on the use of a feature-weighting metric.
3. **Carefully merging instances** (Salzberg, 1991; Wettschereck and Dietterich, 1995; Domingos, 1996): merging multiple instances in single generalised instances. Generalised instances can be represented by conjunctions of *disjunctions of* feature values, or rules with wild-cards.

In the following subsections we outline approaches to each of these three types of careful abstraction during learning.

2.1.1 Editing

In memory-based learning, it seems sensible to keep any instance in memory that plays a positive role in the correct classification of other instances within memory or of new, unseen instances. Alternatively, when it plays no role at all, or when it is disruptive for classification, it may be discarded from memory. These two options form the bases of two approaches to editing found in the literature:

1. *Delete instances that can be deleted without harming the classification performance of the memory-based classifier.* Performance changes can only be measured on the instances stored in memory. The assumption is made that lack of performance loss measured on the instances in memory, transfers to lack of performance loss on unseen instances. Early examples of this approach are the Condensed Nearest Neighbor classifier proposed by Hart (1968), the Reduced Nearest Neighbor classifier (Gates, 1972) and the Iterative

Condensation Algorithm (Swonger, 1972). The IB2 algorithm (Aha, Kibler, and Albert, 1991) is a more recent example.

2. *Delete instances of which the classification is different from the majority class of their nearest neighbors.* Such instances may play a disruptive role in classification, since apparently they are positioned in a part of instance space dominated by another class than their own. Early approaches to this type of editing include Wilson (1972), Tomek (1976), and Devijver and Kittler (1980). In IB3 (Aha, Kibler, and Albert, 1991) the approach is to delete instances estimated to cause misclassifications of (unseen) neighbor instances according to a *class prediction strength* estimate. Again, such estimates can only be based on the material available in memory, but are assumed to apply to unseen material as well.

Aha, Kibler, and Albert (1991) describe a class of instance-based (memory-based) learning algorithms that learn incrementally, i.e. instance by instance. Two of these algorithms, IB2 and IB3, are compared in the case study of Section 3. We describe them briefly here. First, IB2 edits instances from memory that are classified correctly by their neighborhood[2]. Instances eventually stored in memory are instances of which the nearest neighbors have *different* classifications. The assumption is that such instances mark the boundary of an area in which all instances are labeled with the same class; the instances that would be positioned in the centre of such areas are not stored, since their position is safeguarded by the boundary instances surrounding it. This safety assumption makes IB2 a careful abstractor that may economize on memory usage considerably, but it also makes it sensitive to noise (Aha, Kibler, and Albert, 1991).

IB3 extends IB2; it attempts to compensate for the fitting of noise. During incremental learning, records are updated for each instance on its successfulness as nearest neighbor in the classification of all subsequently-stored instances (using some additional bootstrapping heuristics during the beginning stages of learning). On the basis of these records, a statistical test determines whether the instance should be regarded as noise and should be edited. An instance is edited when its classification accuracy is significantly lower than its class' observed frequency.

Although it may be profitable for generalization accuracy to edit noise, it is crucial that the estimation of the noisiness of instances does not mark productive instances (i.e. good classifiers for their own class) as noise, simply because they are in a small disjunct on their own. For some language learning tasks, among which grapheme-phoneme conversion and word pronunciation, the instance space is highly disjunct, and editing the smallest disjuncts causes lower generalization performance almost immediately (Daelemans, Van den Bosch, and Zavrel, 1999). For an instance in a small disjunct to be stored in memory in IB3 and not be deleted later on during learning, it is essential that it is at least once the nearest neighbor to an instance of the same class, which for very small disjuncts (e.g. those containing single instances) is unlikely to happen. IB3 may not be careful enough to these types of data.

2.1.2 Oblivious (partial) decision-tree abstraction

Top-down induction of decision trees (TDIDT) (Breiman et al., 1984; Quinlan, 1986) is, seen from a memory-based learning perspective, a method to compress an instance base into a more compact structure which in principle is able to preserve all information needed to classify all instances in the original training set correctly. In decision trees, instances are represented as paths of arcs that connect nodes. Nodes represent subsets of instances; arcs represent feature value information. All paths stem from a single root node, which represents the whole instance base. From this root node, arcs grow that represent values of a specific feature. Usually, the first feature to be tested in the first level of arcs stemming from the root node is the feature that is estimated to be the most relevant for the classification task at hand. Testing on its values is estimated to lead to unambiguous class information the quickest (on average). An arc (representing a match on a specific feature value) leads either to a non-ending node or a leaf node. Leaf nodes represent subsets of instances with homogeneous class labeling (where "homogeneousness" may be subject to parametrized thresholds). Non-ending nodes represent

[2] Strictly speaking, editing in IB2 is not editing from memory, since left-out instances are only held temporarily in memory at the moment when their local statistics on whether they should be included are computed.

a subset of instances of which the class labeling is not homogeneous, and which need more feature tests (on the feature that is next in the relevance-ranking to begin with) to break down the subset of instances into smaller subsets with homogeneous class labeling.

When a decision tree is fully induced, it has become a compressed version of the original instance base. It is equivalent to pure memory-based learning with respect to classification of the material in the training set when the decision tree is not pruned (i.e. when the expansion of arcs is not halted until nodes represent fully homogeneously labeled instance subsets). At the same time, however, decision trees can leave out (abstract) quite a lot of the original data. All feature values that do not contribute to the disambiguation of a single instance are forgotten. With respect to the training material, this can be considered a very sensible type of careful abstraction. However, this is not necessarily the case with respect to new data, because most decision-tree learning algorithms apply a (usually highly efficient) classification strategy that deviates from finding the (set of) nearest neigbours. Taking the example of the well-known decision-tree learning algorithms C4.5 (Quinlan, 1993), a deterministic traversal is made from the root node down, matching feature values in the order of their importance, until either a mismatch occurs (no matching arc is found) or a leaf node is met. In the latter case, the unambiguous class label of the instance subset represented by that leaf node is returned. Otherwise, the most frequently occurring class of the instance subset represented by the last visited non-ending node is returned.

Thus, classification in a typical TDIDT algorithm, such as C4.5, is based on class labeling information of instance subsets represented at nodes, and is strictly constrained by the ordering of features encoded in the tree, since no backtracking is allowed. In contrast with pure memory-based learning, a mismatch at an important feature blocks the potential matching of values at less important features.

In this article we focus on IGTREE as an instance of a carefully-abstracting TDIDT algorithm. Feature relevance in IGTREE, as in C4.5, is estimated by computing the *information gain* of features. The function for computing information gain (henceforth IG) weighting looks at each feature of instances in an instance base in isolation, and provides an estimate of how much information it contributes to predicting the correct classification. The information gain of feature f is measured by computing the difference in instance-base entropy (i.e. uncertainty, or scrambledness of information) between the situations without and with knowledge of the value of that feature, as displayed in Equation 3:

$$IG_f = \frac{H(C) - \sum_{v \in V_f} P(v) \times H(C|v)}{si(f)} \tag{3}$$

$$si(f) = -\sum_{v \in V_f} P(v) \log_2 P(v) \tag{4}$$

Where C is the set of class labels, V_f is the set of values for feature f, 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. The normalizing factor $si(f)$ (split info) is included to avoid a bias in favour of features with more values. It represents the amount of information needed to represent all values of the feature (Equation 4).

IGTREE computes the ranking of features once, on the whole training set (which is in contrast with C4.5, in which feature ranking is recomputed at every non-ending node). Moreover, IGTREE does not prune; it continues to expand arcs from non-ending nodes that represent subsets of instances that are not fully homogeneous in their class labeling.

2.1.3 Carefully merged instances

Paths in decision trees can be seen as generalized instances. In IGTREE and C4.5 this generalization is performed up to the point where no actual instance is left in memory; all is converted to nodes and arcs. Counter to this decision-tree compression, approaches exist that start with storing individual instances in memory, and carefully merge some of these instances to become a single, more general instance, only when there is some evidence that this operation is not harmful to generalization performance. Although overall memory is compressed, the memory still contains individual items on which the same k-NN-based classification can be performed. The abstraction occurring in this approach is that after a merge, the merged

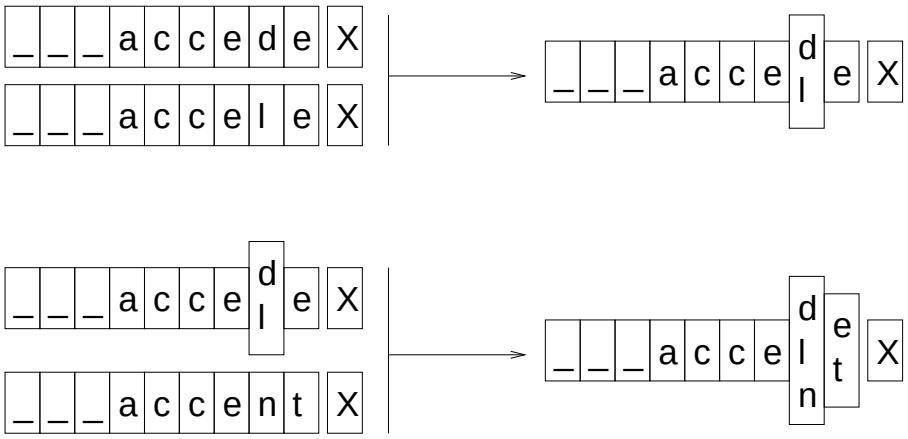

Figure 1: Two examples of the generation of a new hyperrectangle in NGE: from a new instance and an individual exemplar (top) and from a new instance and the hyperrectangle from the top example (bottom).

instances incorporated in the new generalized instance are deleted individually, and cannot be reconstructed. Example approaches to merging instances are NGE (Salzberg, 1991) and its batch variant BNGE (Wettschereck and Dietterich, 1995), and RISE (Domingos, 1996). We provide brief discussions of two of these algorithms: NGE and RISE.

NGE (Salzberg, 1991), an acronym for *Nested Generalized Exemplars*, is an incremental learning theory for merging instances (or exemplars, as Salzberg prefers to refer to instances stored in memory) into *hyperrectangles*, a geometrically motivated term for merged exemplars. Partly analogous to IB2 and IB3, NGE[3] adds instances to memory in an incremental fashion (at the onset of learning, the memory is seeded with a small number of randomly-picked instances). Every time a new instance is presented, it is matched with all exemplars in memory, which can be individual or merged exemplars (hyperrectangles). When it is classified correctly by its nearest neighbor (an individual exemplar or the smallest matching hyperrectangle), the new example is merged with it, yielding a new, more general hyperrectangle.

Figure 1 illustrates two mergings of instances of the grapheme-phoneme conversion task with exemplars. On the top of figure 1, the new instance __accele, labeled with class X (a code for the two-phoneme pronunciation /ks/), is merged with the single-instance exemplar __accede (also of class X), to form the generalized exemplar __acce{d,l}e mapping to their common class X (here, brackets denote the disjunction of values of a single feature). At the bottom of the figure, __acce{d,l}e–X is merged with __accent–X to form the more general __acce{d,l,n}{e,t}–X. *Abstraction* occurs because it is not possible to retrieve the individual instances that are nested in the generalized exemplar; new *generalization* occurs because the generalized exemplar not only matches fully with its nested instances, but also matches fully with possible instances in which feature-value combinations occur that were not present in the nested instances: it would also match __accene, __accelt, and __accedt perfectly.

Notice that with symbolic feature values, the geometrically intuitive concept of nesting becomes void. Since no real-valued distance is computed between symbolic feature values, but rather the simple all-or-none overlap similarity metric applies (Salzberg, 1991; Aha, Kibler, and Albert, 1991), merging yields flat *disjunctions of conjunctions* of feature values, as illustrated in figure 1.

When a new instance is misclassified by its nearest neighbor, it is merged with the second-nearest neighbor if that neighbor would classify the new instance correctly (a "second-chance" heuristic, Salzberg (1991)). If not, the new instance is added to memory as an individual exemplar. It may be inside an existing hyperrectangle, thus representing an exception marked with a different class (a "hole") within the instance space bounded by that hyperrectangle.

Matching between new instances and (merged) exemplars in the implementation of NGE

[3]Salzberg (1991) makes an explicit distinction between NGE as a theory, and the learning algorithm EACH as the implementation; we will use NGE here to denote both.

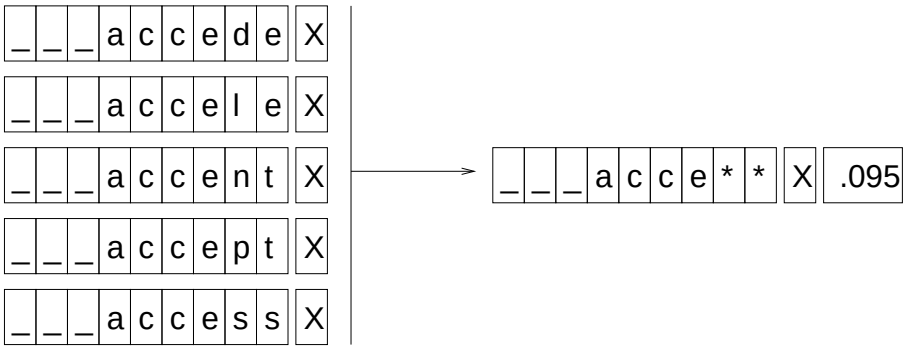

Figure 2: An example of an induced rule in RISE, displayed on the right, with the set of instances that it covers (and from which it was generated) on the left.

is augmented with two additional heuristics: (i) using the *class prediction strength* of an exemplar as a multiplication factor in the similarity function, and (ii) using incrementally-learned global feature weights set according to their contribution to classification error. For details on these weighting metrics the reader is referred to Salzberg (1991), and to Cost and Salzberg (1993) for an elaboration of the class-prediction strength metric.

RISE (*Rule Induction from a Set of Exemplars*) (Domingos, 1995; Domingos, 1996) is a multi-strategy learning method that combines memory-based learning (viz. PEBLS, Cost and Salzberg (1993)) with rule-induction (Michalski, 1983; Clark and Niblett, 1989; Clark and Boswell, 1991). As in NGE, the basic method is that of a memory-based learner and classifier, only operating on a more general type of instance. RISE learns a memory filled with *rules* which are all derived from individual instances. Some rules are instance-specific, and other rules are generalized over sets of instances.

RISE inherits parts of the rule induction method of CN2 (Clark and Niblett, 1989; Clark and Boswell, 1991). CN2 is an incremental rule-induction algorithm that attempts to find the "best" rule governing a certain amount of instances in the instance base that are not yet covered by a rule. "Goodness" of a rule is estimated by computing its apparent accuracy (which is class prediction strength, Cost and Salzberg (1993)) with Laplace correction (Niblett, 1987; Clark and Boswell, 1991). Rule induction ends when all instances are abstracted (covered by rules).

RISE induces rules in a careful manner, operating in cycles. At the onset of learning, all instances are converted to instance-specific rules. During a cycle, for each rule a search is made for the nearest instance not already covered by it that has the same class. If such an instance is found, rule and instance are merged into a more general rule. When identical rules are formed, they are joined. The assumption is made that performance retainment on the training set (i.e. the generalization accuracy of the rule set on the original instances they were based on) also helps performance on test material to be retained at the level of the most accurate of its parent algorithms. At each cycle, the goodness of the rule set on the original training material (the individual instances) is monitored. RISE halts when this accuracy measure does not improve (which may already be the case in the first cycle, yielding a plain memory-based learning algorithm). In a series of experiments it is shown that RISE can improve on its memory-based parent PEBLS, as well as on its rule-induction parent CN2, on a considerable number of benchmark learning tasks (Domingos, 1996).

Applied to symbolically-valued data, RISE creates rules that are left-hand side conjunctions of conditions coupled with a right-hand side consequent being the rule's class. A condition couples a feature to one value in an equality. A rule may contain only one condition per feature, and may contain no conditions at all. Figure 2 illustrates the merging of individual instances into a rule. The rule contains seven non-empty conditions, and two empty ones (filled with wild-cards, '*', in the figure). The rule now matches on every instance beginning with __acce, and receives a goodness score (i.e. its apparent accuracy on the training set with Laplace correction) of 0.095.

Instance classification is done by searching for the best-matching rule, always selecting the

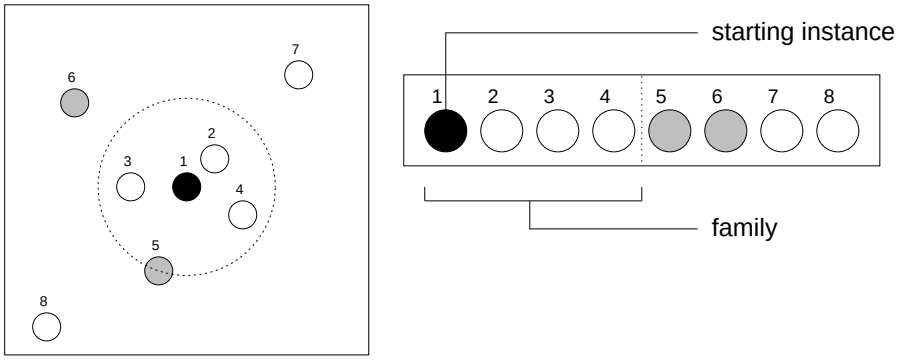

Figure 3: An example of a family in a two-dimensional instance space (left). The family, at the inside of the dotted circle, spans the focus instance (black) and the three nearest neighbors labeled with the same class (white). When ranked in the order of distance (right), the family boundary is put immediately before the first instance of a different class (grey).

rule with the highest Laplace accuracy (Clark and Boswell, 1991). As a heuristic add-on for dealing with symbolic values, RISE incorporates a value-difference metric (Stanfill and Waltz, 1986; Cost and Salzberg, 1993) by default, called the *simplified value-difference metric* (SVDM) due to its simplified treatment of feature-value occurrences in the VDM function (Domingos, 1996).

2.2 FAMBL: merging instance families

FAMBL, for *FAMily-Based Learning*, is a new algorithm that constitutes an alternative approach to careful abstraction over instances. The core idea of FAMBL is to transform an instance base into a set of *instance family expressions*. An instance family expression is a hyperrectangle, but the procedure for merging instances differs from that in NGE or in RISE. First, we outline the ideas and assumptions underlying FAMBL. We then give a procedural description of the learning algorithm.

2.2.1 Instance families

Classification of an instance in memory-based learning involves a search for the nearest neighbors of that instance. The value of k in k-NN determines how many of these neighbors are used for extrapolating their (majority) classification to the new instance. A fixed k ignores (smoothes) the fact that an instance is often surrounded in instance space by a number of instances of the same class that is actually larger or smaller than k. We refer to such variable-sized set of same-class nearest neighbors as an instance's *family*. The extreme cases are on the one hand instances that have a nearest neighbor of a different class, i.e. they have no family members and are a family on their own, and on the other hand instances that have as nearest neighbors all other instances of the same class.

Thus, families are class clusters, and the number and sizes of families in a data set reflect the *disjunctivity* of the data set: the degree of scatteredness of classes into clusters. In real-world data sets, the situation is generally somewhere between the extremes of total disjunctivity (one instance per cluster) and no disjunctivity (one cluster per class). Many types of language data appear to be quite disjunct (Daelemans, Van den Bosch, and Zavrel, 1999). In highly disjunct data, classes are scattered among many small clusters, which means that instances have few nearest neighbors of the same class on average.

Figure 3 illustrates how FAMBL determines the family of an instance in a simple two-dimensional example instance space. All nearest neighbors of a randomly-picked starting instance (marked by the black dot) are searched and ranked in the order of their distance to the starting instance. Although there are five instances of the same class in the example space, the family of the starting instance contains only three instances, since its fourth-nearest instance is of a different class.

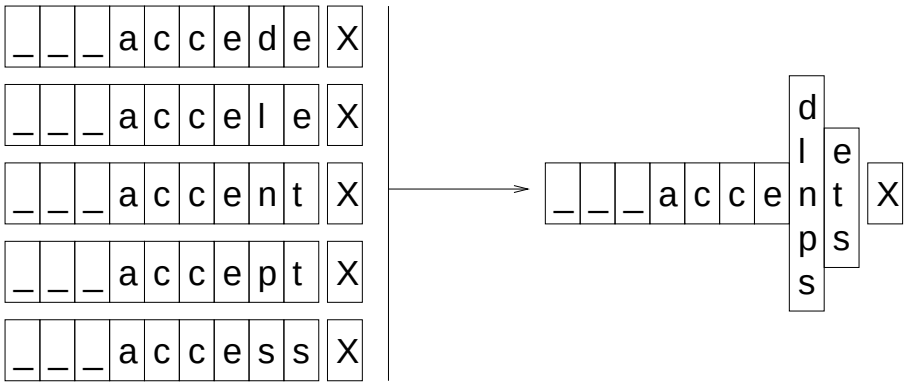

Figure 4: An example of family creation in FAMBL. Four grapheme-phoneme instances, along with their token occurrence counts (left) are merged into a family expression (right).

Families are converted in FAMBL to *family expressions*, which are hyperrectangles, by merging all instances belonging to that family simultaneously. Figure 4 illustrates the creation of a family expression from an instance family. In contrast with NGE,

- family expressions are created in one operation, rather than by step-wise nesting of each individual family member.
- a family is abstracted only once and is not merged later on with other instances or family expressions.
- families cannot contain "holes", i.e. instances with different classes, since the definition of family is such that family abstraction halts as soon as the nearest neighbor with a different class is met in the local neighborhood.
- FAMBL is non-incremental: it operates on a complete instance base stored in memory.

The general modus of operation of FAMBL is that it randomly picks instances from an instance base one by one from the set of instances that are not already part of a family. For each newly-picked instance, FAMBL determines its family, generates a family expression from this set of instances, and then marks all involved instances as belonging to a family (so that they will not be picked as starting point or member of another family). FAMBL continues determining families until all instances are marked as belonging to a family.

Notice that families reflect the locally optimal k surrounding the instance around which the family is created. The locally optimal k is a notion that is also used in locally-weighted learning methods (Vapnik and Bottou, 1993; Wettschereck and Dietterich, 1994; Wettschereck, 1994; Atkeson, Moore, and Schaal, 1997); however, these methods do not abstract from the learning material. In this sense, FAMBL can be seen as a locally-weighted abstractor.

2.2.2 The FAMBL algorithm

The FAMBL algorithm has a learning component and a classification component. The learning component of FAMBL is composed of two stages: a *probing* stage and a *family extraction* stage.

The probing stage (schematized in figure 6) is a preprocessing stage to the actual family extraction as outlined in the previous Subsection. The reason for preprocessing is visualized in figure 5. The random selection of instances to be a starting point for family creation can be quite unfortunate. When, for example, the middle instance in the left part of figure 5 is selected first, a seven-instance family is formed with relatively large within-family distances. Moreover, three other instances that are actually quite close to members of this big family become isolated and are necessarily extracted later on as single-instance families. The situation in the right part of figure 5 displays a much more desirable situation, in which the space is more evenly divided between only two families instead of four.

In the probing stage, all families are extracted randomly and straightforwardly, while records are maintained of (i) the size of each family, and (ii) the average distance between the starting instance of each family and the other instances in the family. When all instances are

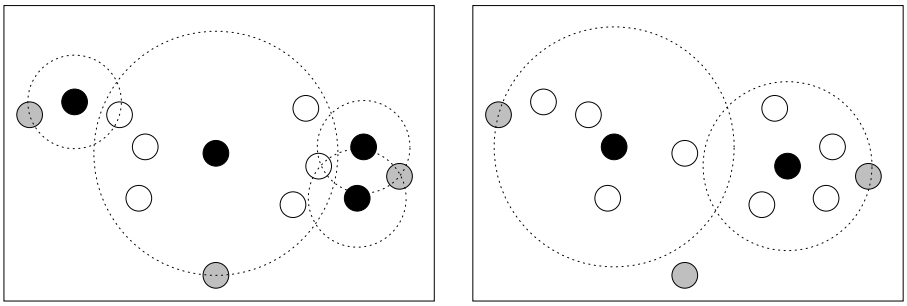

Figure 5: Illustration of the need for the preprocessing stage in FAMBL. The left figure shows a big family with seven members, forcing the remaining three instances to be their own family. The right figure shows the same space, but with two other starting points for family creation, displaying a more evenly divided space over two families. Black instances denote starting points for family creation; white instances are of the same class as the starting points, and grey instances are of a different class.

captured in families, the median of both records are computed, and *both medians are used as threshold values for the second, actual family extraction phase.* This means that in the family extraction phase,

1. no family is extracted that has more members than the probed median number of members.
2. no family is extracted that has an average distance from the starting instance to the other family members larger than the probed median value.

Thus, the actual family extraction phase applies extra careful abstraction, under the assumption that it is better to have several medium-sized, adjacent families of the same class than one big family overlapping the medium ones except for some adjacent boundary instances that get isolated.

It should be noted that with symbolic feature values (and without value-difference metrics), the k nearest neighbors in any matching operation may not be the same as the number of different distances found in this nearest-neighbor set. For example, five nearest neighbors may be equally-best matching with an new instance when they all differ in the same single feature value. The k in FAMBL, including the median value found in the probing phase, is chosen to refer to the k number of different *distances* found among the nearest neigbours.

A schematized overview of the two FAMBL learning phases is displayed in figures 6 and 7.

After learning, the original instance base is discarded, and further classification is based only on the set of family expressions yielded by the family-extraction phase. Classification in FAMBL works analogously to classification in pure memory-based learning, and classification in NGE: a match is made between a new test instance and all stored family expressions. When a family expression records a disjunction of values for a certain feature, matching is perfect when one of the disjunctive values matches the value at that feature in the new instance. When two or more family expressions of different classes match equally well with the new instance, the class is selected with the highest occurrence summed over the matching expressions. When the tie remains, the class is selected that occurs the most frequently in the complete family expression set.

We conclude our description of the FAMBL algorithm by noting that FAMBL allows for the inclusion of informational abstraction in the form of feature-weighting, instance-weighting and value-difference metrics. For comparisons with other algorithms, as described in the next section, we have included some of these metrics as options in FAMBL. Weighting metrics are likely to have a profound effect on family extraction. For example, a study by Van den Bosch (1997) suggests that using information-gain feature weighting (Quinlan, 1986) in pure memory-based learning (viz. IB1-IG, Daelemans and Van den Bosch (1992a)), can yield considerably bigger families.

Procedure FAMBL PROBING PHASE:

Input: A training set TS of instances $I_{1...n}$, each instance being labeled with a family-membership flag set to $FALSE$

Output: Median values of family size, M_1, and within-family distance, M_2

$i = 0$

1. Randomize the ordering of instances in TS
2. While not all family-membership flags are $TRUE$, Do
   - While the family-membership flag of I_i is $TRUE$ Do increase i
   - Compute NS, a ranked set of nearest neighbors to I_i with the same class as I_i, among all instances with family-membership flag $FALSE$. Nearest-neighbor instances of a different class with family-membership flag $TRUE$ are still used for marking the boundaries of the family.
   - Record the number of members in the new virtual family: $|NS| + 1$
   - Record the average distance of instances in NS to I_i
   - Set the membership flags of I_i and all instances in NS to $TRUE$
3. Compute M_1
4. Compute M_2

Figure 6: Schematized overview of the probing phase in FAMBL.

3 Careful abstraction applied to a small data set

We performed a series of experiments concerning the application of a selection of careful-abstracting methods, described in the previous section, to grapheme-phoneme conversion. It is intended to provide some initial indications for the efficacy of different careful abstraction approaches as compared to pure memory-based learning, including variants of both approaches which use weighting metrics. As said, the chosen benchmark task is known for its sensitivity to abstraction, so that it is likely that any differences in abstraction methods show up clearly in results obtained with this task.

From an original instance base of 77 565 word-pronunciation pairs extracted from the CELEX lexical data base of English (Baayen, Piepenbrock, and van Rijn, 1993) we created ten equal-sized data sets each containing 7757 word-pronunciation pairs. Using windowing (cf. Section 2) and partitioning of this data in 90% training and 10% test instances, ten training and test sets are derived containing on average 60 813 and 6761 instances, respectively. These are token counts; in the training sets, 54 295 instance types occur (on average). Notice that these training and test sets are not constructed as is usual in 10-fold cross validation (Weiss and Kulikowski, 1991), where training sets largely overlap. Here, each training and test set is derived from words that are not used in any other training or test set[4]. This approach, using relatively small data sets as compared to earlier studies (Van den Bosch, 1997), was taken to cope with the extreme memory demands of some of the algorithms tested. Furthermore, we believe that the approach alleviates some of the objections raised against using t-tests for significance testing with dependent data sets (Salzberg, 1997; Dietterich, 1998).

In this series, one experiment consists of applying one algorithm to each of the ten training sets, and a test on each of the respective test sets[5]. Apart from the careful abstractors IB2, IB3, IGTREE, RISE, NGE, and FAMBL, we include the pure memory-based learning algorithm IB1(-IG) (Aha, Kibler, and Albert, 1991; Daelemans and Van den Bosch, 1992b) in the comparison. Each experiment yields (i) the mean generalization accuracy, in percentages correctly classified test instances, (ii) a standard deviation on this mean, and (iii) a count on the number of items in memory (i.e. instances or merged instances). Table 2 lists these experimental outcomes

[4]Nevertheless, since words are partly similar and since windowing only looks at parts of words, there is some overlap in the instances occurring in the different sets.

[5]Experiments with IB1, IB1-IG, and IGTREE were performed using the TIMBL software package, available at `http://ilk.kub.nl/`.

Procedure FAMBL FAMILY-EXTRACTION PHASE:

Input: A training set TS of instances $I_{1...n}$, each instance being labeled with a family-membership flag set to $FALSE$

Output: A family set FS of family expressions $F_{1...m}$, $m \leq n$

$i = f = 0$

1. Randomize the ordering of instances in TS
2. While not all family-membership flags are $TRUE$, Do
   - While the family-membership flag of I_i is $TRUE$ Do increase i
   - Compute NS, a ranked set of nearest neighbors to I_i with the same class as I_i, among all instances with family-membership flag $FALSE$. Nearest-neighbor instances of a different class with family-membership flag $TRUE$ are still used for marking the boundaries of the family.
   - Compute the number of members in the new virtual family: $|NS| + 1$
   - Compute the average distance of all instances in NS to I_i: $A_{NS,I}$
   - While $((|NS + 1| > M_1) OR (A_{NS,I} > M_2))$ Do remove the most distant family member to I in NS
   - Set the membership flags of I_i and all remaining instances in NS to $TRUE$
   - Merge I_i and all instances in NS into the family expression F_f and store sthis expression along with a count of the number of instance merged in it
   - $f = f + 1$

Figure 7: Schematized overview of the family-extraction phase in FAMBL.

for all algorithms tested. As some of these algorithms use metrics for weighting: IG, class-prediction strength (CPS), or value-difference metrics (VDM), we have marked the use of such metrics explicitly in separate columns, using 'x' for denoting the presence of a metric in the algorithm in that row. Table 3 displays the significance test results (one-tailed t-tests) performed with each pair of algorithms. Significant results (with $p < 0.05$, marked with asterisks) indicate that the algorithm in the row has a significantly higher generalization accuracy than the algorithm in the column.

The results indicate a group of six best-performing algorithms that, in pair-wise comparisons, do not perform significantly differently: (i) FAMBL with IG weighting and the MVDM metric, (ii) IB1 with IG and MVDM, (iii) RISE with SVDM and the CPS metric with Laplace correction, (iv) IB1 with IG, (v) FAMBL with IG, and (vi) IGTREE. All algorithms using IG weighting are in this best-performing group.

Of course, our focus is on the effects of careful abstraction. Within the group of four best-performing algorithms, three are careful abstractors, performing careful merging of instances: RISE, FAMBL, and IGTREE. RISE is able to shrink its rule base down to 20 252 rules. As compared to the 54 295 instance types stored in pure memory-based learning (IB1), RISE obtains an item compression of 62.7%. FAMBL compresses less: 41.2% (with IG and MVDM) or 35.3% (with IG). IGTREE is able to compress the instance bases into trees that contain, on average, only 5923 leaf nodes, which are counted as items here, but which do not directly relate to instances. In sum, we see that the careful abstraction approaches RISE, FAMBL, and IGTREE reduce the number of memory items, while at the same time they equal the performance of the pure memory-based approach.

It is relevant to our central topic to analyse the effects of careful abstraction *without* additional weighting, thus drawing on the results obtained with IB1, IB2, IB3, and FAMBL without weighting. Here, pure memory-based learning is at a significant advantage: IB1 is significantly more accurate than each of the three carefully-abstracting methods. In turn, IB2 performs significantly better than IB3 and FAMBL, and IB3 performs significantly better than FAMBL. For this task, (i) editing in IB2 and IB3 is harmful; (ii) the noise reduction in IB3 is extra harmful (which is in accordance with the findings of Daelemans, Van den Bosch, and

algorithm	metrics IG	CPS	VDM	generalization accuracy (%)	±	number of memory items
FAMBL	x		x	88.98	0.64	31 948
IB1	x		x	88.91	0.63	54 294
RISE		x	x	88.88	0.61	20 252
IB1	x			88.83	0.61	54 294
FAMBL	x			88.82	0.61	35 141
IGTREE	x			88.46	0.53	5923
FAMBL			x	87.96	0.68	34 052
IB1			x	87.88	0.66	54 294
IB1				78.07	0.79	54 294
IB2				74.27	0.66	19 917
IB3				73.33	0.45	19 196
FAMBL				72.26	0.76	31 889
NGE		x		61.79	0.86	25 627
C4.5	x			87.10	0.43	20 889
Naive Bayes				74.16	0.58	–

Table 2: Overview of generalization accuracies and memory usage obtained with pure memory-based learning and careful abstraction methods, as well as with C4.5 and Naive Bayes.

Zavrel (1999)); and (iii) family extraction is even more harmful than incremental editing. For FAMBL, this provides an indication that IG weighting may be essential for the success of the approach. In other words, for this task, IG weighting yields a rescaling of the instance space better suited for generalization.

Finally, we make two side observations. First, NGE, with its default class-prediction strength weighting and incremental learning, performs significantly worse than IB2, IB3, and FAMBL. It is unclear what contributes most to this low accuracy: its strategy to nest hyperrectangles incrementally, or the class-prediction strength weighting. RISE is able to profit from CPS weighting, albeit with Laplace correction. NGE's performance may therefore be hampered mainly by its incrementality, which may also hamper IB2 and IB3 as compared to IB1. Second, when we compare the performances of the memory-based learning algorithms with decision-tree learning in C4.5 and the Naive Bayes classifier (Langley, Iba, and Thompson, 1992) (displayed at the bottom of table 2), we can determine roughly that C4.5 with default parameter values performs worse than most weighted memory-based methods on this task, a result that is in agreement with Van den Bosch (1997) and Daelemans, Van den Bosch, and Zavrel (1999), and that the Naive Bayes classifier, which may serve as a good baseline classifier, performs at about the level of IB2.

4 Effects of data set size magnitude on careful abstraction

The dataset of grapheme-phoneme conversion task instances used in Section 3 is small. It is constructed from 7757 words, which is much less than is usually contained in a pronunciation lexicon; in fact, the pronunciation lexicon available to us contains ten times as much words, which may all be used for data set generation. It is conceivable that data set size matters in the differences between careful abstractors and pure memory-based learning. Adding more instances to the data set may alter the average number, size and within-family distance of the disjuncts in that data set. The outcome of careful abstraction may be affected by this change at a different scale than it may affect pure memory-based learning.

In this section we report on experiments in which the careful abstractors FAMBL with IG weighting, RISE, and IGTREE, and the pure memory-based learning algorithm IB1 with IG weighting, are applied to data sets of increasing size, of the grapheme-phoneme conversion investigated in the previous section (henceforth referred to as GP), English grapheme-phoneme

algorithm	IB1 IG MVDM	RISE CPS SVDM	IB1 IG	FAMBL IG	IGTREE	FAMBL MVDM	IB1 MVDM	IB1	IB2	IB3	FAMBL	NGE CPS
FAMBL-IG-MVDM						**	**	***	***	***	***	***
IB1-IG-MVDM	—					**	**	***	***	***	***	***
RISE-SVDM-CPS	—	—				**	**	***	***	***	***	***
IB1-IG	—	—	—			**	**	***	***	***	***	***
FAMBL-IG	—	—	—	—		**	**	***	***	***	***	***
IGTREE	—	—	—	—	—	*	*	***	***	***	***	***
FAMBL-MVDM	—	—	—	—	—	—		***	***	***	***	***
IB1-MVDM	—	—	—	—	—	—	—	***	***	***	***	***
IB1	—	—	—	—	—	—	—	—	***	***	***	***
IB2	—	—	—	—	—	—	—	—	—	***	***	***
IB3	—	—	—	—	—	—	—	—	—	—	***	***
FAMBL	—	—	—	—	—	—	—	—	—	—	—	***

Table 3: Significance results: one-tailed t-test outcomes for algorithm pairs, expressing whether the algorithm in the row has a significant lower generalization error than the algorithm in the column. '*' denotes $p < 0.05$; '**' denotes $p < 0.01$; '***' denotes $p < 0.001$.

conversion combined with stress assignment (i.e. word pronunciation, henceforth GS), and English morphological segmentation (henceforth MS). We briefly characterize the two latter tasks. Details on the numbers of instances, features, values per feature, and classes in the full data sets of the two tasks are listed in table 4:

Grapheme-phoneme conversion combined with stress assignment (i.e. word pronunciation) is similar to the grapheme-phoneme conversion task illustrated earlier, but with two differences: (i) the windows only span seven letters, and (ii) the class represents a combined phoneme and a stress marker. The stress marker part denotes whether the phoneme is the first of a syllable receiving primary or secondary stress. For example, class '/b/1' indicates a phoneme /b/ that is the first phoneme of a syllable receiving primary stress, which would be the class label of the instance __book from the word booking. See (Van den Bosch, 1997) for more details.

Morphological segmentation is the segmentation of words into labeled morphemes. Each instance represents a window snapshot of a word of nine letters. Its class represents the presence or absence of a morpheme boundary immediately before the middle letter. If present, it also encodes the type of morpheme starting at that position, i.e. whether it is a stem, an inflection, a stress-neutral affix, or a stress-affecting affix. For example, the word booking is composed of the stem book and the inflection ing; consequently, the first instance generated from the word is __booki with class 'present–stem', the second __bookin with class 'absent', the fifth booking__ with class 'present-inflection', the sixth ooking__ with class 'absent', etc. See (Van den Bosch, Daelemans, and Weijters, 1996) for more details.

Task	# Features	# Values of feature 1	2	3	4	5	6	7	8	9	10	11	# Classes	# Data set instances
GP	9	42	42	42	42	41	42	42	42	42			61	675 745
GS	7	42	42	42	41	42	42	42					159	675 745
MS	9	42	42	42	42	41	42	42	42	42			2	573 544

Table 4: Specifications of the three investigated data sets of the GP, GS, and MS learning tasks: numbers of features, values per feature, classes, and instances.

All three tasks are in the morpho-phonological domain, at the word level. They are selected

to allow RISE to be in the comparison, since the available implementation of RISE is too memory-intensive with features with more than a few hundred values (due to its unabridged storage of SVDM matrices). All of the three tasks are represented by features that carry letters as values, of which there are 42 (the empty character and letters with diacritics are considered individual letters).

From each dataset, we extracted three smaller datasets, containing 1/1000th, 1/100th, and 1/10th of the words in the original data set, respectively. Each data set and the complete data sets themselves were used in a 10-fold CV experiment with IB1-IG, FAMBL, RISE, and IGTREE. RISE was not applied to the full data sets due to the computational inefficiency of RISE's current available implementation, and time and computing constraints. It should be noted that many current implementations of careful-abstraction learning algorithms do not allow for experiments on large data sets; however, (i) algorithmic optimizations may resolve part of the problem, and (ii) computer technology developments continue to soften the need for optimizations.

The generalization accuracies yielded by the four learning algorithms are visualised in the left half of figure 8 as learning curves. On a global level, they show that generalization accuracies differ less between algorithms with increasing data set sizes. There is reason to believe that RISE, trained and tested on the full data sets, would perform similarly to the other three algorithms. The differences between algorithms differ in sign and significance over the tasks and data set sizes. RISE displays significant advantage over the other algorithms on some reduced data sets (it is significantly better with $p < 0.01$ than the second-best algorithm in the 1/100th GS task, with $p < 0.05$ in the 1/100th MS task, and with $p < 0.01$ in the 1/10th MS task).

The overall best performances are obtained with IB1-IG on the complete data sets. With the GP task, the difference between IB1-IG and FAMBL is not significant. With the two other tasks, it is significant (with $p < 0.01$ on the GS task, and with $p < 0.001$ on the MS task). IGTREE's performance loss compared to IB1-IG and FAMBL is significant on the largest data sets of GP and MS; IGTREE performs worse than both IB1-IG and FAMBL on the full GP and MS data sets at the $p < 0.001$ level.

The right half of figure 8 displays the memory item compression levels obtained by the three careful abstractors as opposed to IB1-IG. The unit of counting is the abstract notion of memory item (cf. table 2), which is an instance type (i.e. not instance token) in IB1-IG, a family expression in FAMBL, a rule in RISE, and an end node in IGTREE. First, larger data sets allow for more compression. All careful abstractors are able to reduce the number of generalised instances (families, rules, end nodes) further, relative to the total number of instance types, when the number of training instances increases. Apparently, more of the learning material can be merged together. Second, IGTREE yields the overall highest compression rates. When coupled with the generalization accuracy results displayed in the left half of the figure, it becomes clear that IGTREE's careful abstraction is quite powerful, yielding competitive performance on the 1/1000th and 1/100th data sets, and losing not too much on the larger sets. Yet, as noted above, these small differences are statistically significant. Third, RISE compresses more than FAMBL on the tested data set sizes. This is at the cost of considerably longer learning times (in the order of 10 to 100 times longer than those of FAMBL), due to the possibly large number (in observed practice, between 5 and 15) of computationally costly rule induction cycles needed to arrive at the final rule set.

Summarizing this section, it can be stated that data set magnitude does have an effect of the difference between careful-abstracting methods and pure memory-based learning. Advantages in terms of better generalization performance of careful abstraction methods are found with reduced data sets; when running experiments with the full data sets, these advantages disappear. On the other hand, the compression rates yielded by the careful abstractors (notably IGTREE) are computationally interesting.

5 Careful abstraction with realistically-sized language learning data sets

In Section 4 we introduced data sets representing tasks in the morpho-phonological domain that in their unabridged form contained over 500,000 instances. Similarly-sized data sets are

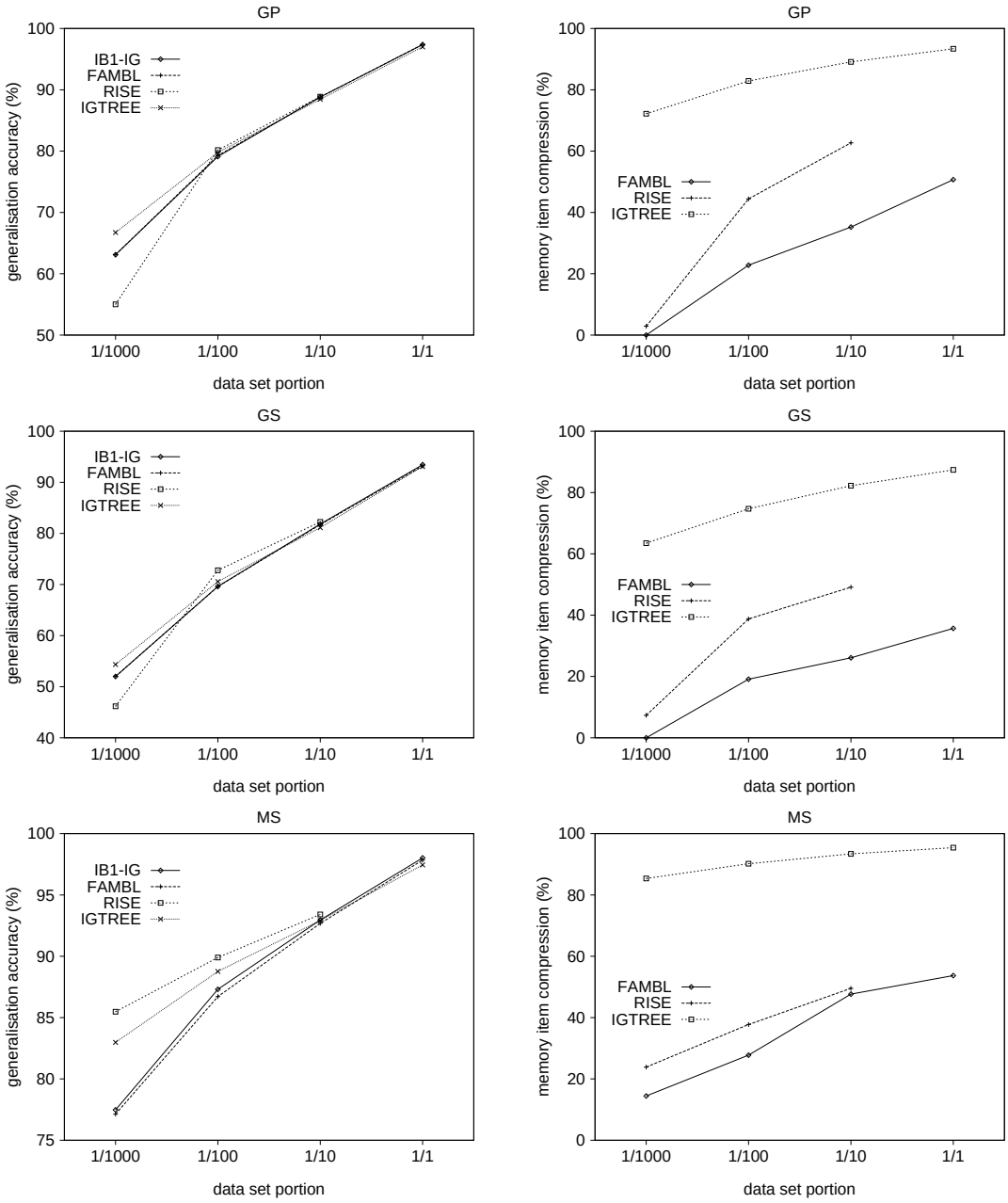

Figure 8: Learning curves in generalization accuracy (left) and memory item compression (right) for 1/1000, 1/100, 1/10 and full data sets of the GP, GS, and MS tasks, as obtained with IB1-IG, FAMBL, RISE, and IGTREE. Generalization accuracy denotes the percentage of correctly classified test instances. Memory item compression is expressed in the percentage of items that is removed by FAMBL, RISE, or IGTREE, as compared to IB1-IG.

available for many other language learning tasks. It is not uncommon in inductive language learning studies to use such data sets, because learning curve studies show generalization improvements at very large data set sizes (cf. Section 4 and Daelemans, Berck, and Gillis (1997), Van den Bosch (1997)); second, because many language data sets represent sparse instance spaces: many samples are needed to represent these tasks at realistic scales.

As in Section 4, we were not able to apply all available careful abstraction algorithms to these large data sets. We were, however, able to apply FAMBL to them, and present the results of these experiments here. We performed a series of 10-fold cross validation experiments (Weiss and Kulikowski, 1991) with FAMBL, augmented with IG feature weighting, on grounds of the positive effect of this metric on FAMBL's performance on the grapheme-phoneme conversion task reported in Section 3. We already reported on the performance of FAMBL on the complete data sets of three morpho-phonological tasks in Section 4. In addition, FAMBL is applied to language data sets used in earlier studies (Veenstra, 1998; Daelemans, Van den Bosch, and Zavrel, 1999). The joint selection of data sets represents a range of language tasks: grapheme-phoneme conversion, word pronunciation, morphological segmentation, base-NP chunking, PP attachment, and POS tagging. The first three tasks have been introduced in the previous Sections 3 and 4. Table 5 lists the numbers of instances, feature values, and classes of the data sets of the three new tasks. We briefly outline the underlying tasks that these three data sets represent.

Base-NP chunking (henceforth NP) is the segmentation of sentences into non-recursive NPs (Abney, 1991). Veenstra (1998) used the Base-NP tag set as presented in (Ramshaw and Marcus, 1995): *I* for inside a Base-NP, *O* for outside a Base-NP, and *B* for the first word in a Base-NP following another Base-NP. As an example, the IOB tagged sentence: "The/I postman/I gave/O the/I man/I a/B letter/I ./O" results in the following Base-NP bracketed sentence: "[The postman] gave [the man] [a letter]." The data are based on the same material as used by Ramshaw and Marcus (1995) which is extracted from the Wall Street Journal text in the parsed Penn Treebank (Marcus, Santorini, and Marcinkiewicz, 1993). An instance (constructed for each focus word) consists of features referring to words (two left-neighbor and one right-neighbor word), their part-of-speech tags, and IOB tags (predicted by a first-stage classifier) of the focus and the two left and right neighbor words. See Veenstra (1998) for more details, and (Daelemans, Van den Bosch, and Zavrel, 1999) for a series of experiments on the data set also used here.

PP attachment (henceforth PP) is the attachment of a PP in the sequence VP NP PP (VP = verb phrase, NP = noun phrase, PP = prepositional phrase). The data consists of four-tuples of words, extracted from the Wall Street Journal Treebank (Ratnaparkhi, Reynar, and Roukos, 1994). They took all sentences that contained the pattern VP NP PP and extracted the head words from the constituents, yielding a V N1 P N2 pattern (V = verb, N = noun, P = preposition). For each pattern they recorded whether the PP was attached to the verb or to the noun in the treebank parse. For example, the sentence *"he eats pizza with a fork"* would yield the pattern eats, pizza, with, fork, verb.. A contrasting sentence would be *"he eats pizza with anchovies"*: eats, pizza, with, anchovies, noun. From the original data set, used in statistical disambiguation methods by Ratnaparkhi, Reynar, and Roukos (1994) and Collins and Brooks (1995), and in a memory-based learning experiment by Zavrel, Daelemans, and Veenstra (1997), (Daelemans, Van den Bosch, and Zavrel, 1999) took the train and test set together to form the data also used here.

POS tagging (henceforth POS) is short for part-of-speech tagging of word forms in context. Many words in a text are ambiguous with respect to their morphosyntactic category (part-of-speech). Each word has a set of lexical possibilities, and the local context of the word can be used to select the most likely category from this set (Church, 1988). For example in the sentence *"they can can a can"*, the word *can* is tagged as modal verb, main verb and noun respectively. We assume a POS-tagger that processes a sentence from the left to the right by classifying instances representing words in their contexts (as described in Daelemans et al. (1996)). The word's already tagged left context is represented by the disambiguated categories of the two words to the left, the word itself and its ambiguous right context are represented by categories which denote ambiguity classes (e.g. verb-or-noun).

Task	# Features	# Values of feature											# Classes	# Data s et insta nces
		1	2	3	4	5	6	7	8	9	10	11		
PP	4	3474	4612	68	5780								2	23 898
NP	11	20 231	20 282	20 245	20 263	86	87	86	89	3	3	3	3	251 124
POS	5	170	170	498	492	480							169	1 046 152

Table 5: Specifications of the three additionally investigated data sets of the NP, PP, and POS learning tasks: numbers of features, values per feature, classes, and instances.

For each task FAMBL is compared with IB1-IG and IGTREE. Table 6 lists the generalization accuracies obtained in these comparisons, on the six tasks. The results of IB1-IG and IGTREE on the GS, NP, PP, and POS tasks are reproduced from (Daelemans, Van den Bosch, and Zavrel, 1999). One-tailed t-tests yield significance results that show at a general level that (i) IB1-IG is significantly more accurate than IGTREE on all tasks; (ii) FAMBL is significantly more accurate than IGTREE on all but the GS task, and (iii) IB1-IG is significantly more accurate than FAMBL on the MS, GS, and PP tasks. The latter results show that FAMBL's careful abstraction is not careful enough on these tasks.

Task	IB1-IG				FAMBL-IG			IGTREE	
	%	±	>FAMBL?	>IGTREE?	%	±	>IG TREE?	%	±
GP	97.37	0.09		***	97.32	0.09	***	96.99	0.09
MS	98.02	0.05	***	***	97.84	0.06	***	97.45	0.06
GS	93.45	0.15	**	***	93.22	0.24		93.09	0.15
NP	98.07	0.05		***	98.04	0.05	***	97.28	0.08
PP	83.48	1.16	**	***	81.80	1.14	***	78.28	1.79
POS	97.86	0.05		***	97.83	0.04	***	97.75	0.03

Table 6: Generalization accuracies (percentages of correctly classified test instances, with standard deviations) of IB1-IG, FAMBL, and IGTREE on the MS, GS, NP, and PP tasks. Asterisks denote the outcomes of one-tailed t-tests, denoting a significantly better accuracy of the algorithm in the top row compared to the algorithm in the second row. '**' denotes $p < 0.01$; '***' denotes $p < 0.001$.

On closer inspection of the experimental results, the behaviour of FAMBL on the six tasks turns out to be varying. We monitored for all experiments the number of families that FAMBL probed and extracted, including the median sizes and within-family distances it found during probing. Table 7 displays these results averaged over the ten experiments performed on each task. The table also displays two additional quantities: (i) the measured *clusteredness*, which reflects the number of disjunct clusters (families) per class, averaged over classes, weighted by their frequency, and (ii) the percentage of compression over the number of memory items (instance types vs. family expressions). Both quantities are reported for the probing stage as well as the family stage.

Table 7 illustrates how much abstraction is attained by FAMBL. In the probing phases, the clusteredness of classes is already in the order of a thousand, except for the PP task. In the family extraction phase, clusteredness in the GP, MS, and NP tasks reaches levels in the order of ten thousand. The numbers of extracted families are also very high (e.g. 162 954 for the GP task in the family extraction phase). The increased numbers of families and the clusteredness in the family extraction phase as compared to the probing phase are the direct effect of using the thresholds computed in the probing phase. The thresholds on k illustrate that family extraction is strictly limited to $k = 1$ in the GS and PP tasks, i.e. family members are allowed to mismatch in only one feature value. With GP, MS, and POS, two mismatching features are allowed; with NP three.

In the extraction phase, compression (the percentage reduction on the number of items in memory, from instances in pure memory-based learning to family expressions) ranges from 31.0% with the GS task to 75.1% with POS, which is considerable. The lowest compression is obtained with GS, on which FAMBL did not outperform IGTREE, and the highest compression is obtained with NP and POS, both on which FAMBL equaled with IB1-IG. This would suggest

Task	generalization accuracy %	±	probing stage #	cluster-edness	% item compr.	k	median distance	extraction stage #	cluster-edness	% item comp.
GP	97.32	0.09	31 224	1908	90.5	2	0.096	162 954	10 230	50.7
MS	97.84	0.06	18 783	8435	93.4	2	0.078	131 776	74 616	53.7
GS	93.22	0.24	37 457	1693	83.2	1	0.084	153 441	8445	31.0
NP	98.04	0.05	5238	2413	97.7	3	0.155	59 376	29 072	72.4
PP	81.80	1.14	157	78	99.3	1	0.078	6414	3193	70.1
POS	97.83	0.04	11 397	479	98.0	2	0.283	140 488	4766	75.1

Table 7: Measurements on FAMBL output during the probing stage and the family extraction stage on each of the six learning tasks. Probing stage measurements are the number of probed families, the clusteredness of the classes, the item compression compared to the original instance base, and the median family size and within-family distance. Extraction stage measurements are the number of probed families, the clusteredness of the classes, and the item compression compared to the original instance base. The generalization accuracies are repeated from table 6.

that the underlying assumptions of FAMBL apply successfully to the NP and POS tasks, and that the GS task data has properties that FAMBL is less prepared to handle adequately. We have two working hypotheses on what these properties might be:

1. The GS data is very disjunct: FAMBL detects a relatively high number of families during probing and family extraction. The random selection of starting points for family extraction, although heuristically patched with the preprocessing of the probing phase, may still lead to unwanted effects as illustrated in figure 5 when data is very disjunct.
2. FAMBL tends to blur *feature interaction*: it allows the combination of feature values that never occurred in that constellation in the learning material, while for some tasks, including GS, this generalization may be unwanted. For example, considering the example of figure 3, it may be actually counterproductive for the family expression in this figure to fully match with __accepe or __accedt, which are nonsensical, but for which it is in any case unclear whether the cc would be pronounced a stressed /ks/.

Feature interaction on a local basis is ignored in all memory-based learning methods mentioned in this paper; we return to this matter in Section 6 in which we discuss openings to add feature-interaction methods in memory-based learning.

While our focus is on language learning, FAMBL is a machine learning algorithm that may also be tested according to more standard machine learning methodology. In machine learning it is common to compare algorithms on series of benchmark data sets of very different nature, to avoid the comparison of algorithms on data on which one is tuned to, and the other is not typically suited for (Salzberg, 1997). FAMBL may have a bias to language-like data. Only a small part of typical benchmark data sets is language-related. We have applied FAMBL to a selection of benchmark tasks from the UCI repository (Blake, Keogh, and Merz, 1998) with only symbolic feature values, using tenfold cross-validation experiments. Table 8 shows the results from the probing phase, repeating some of the results obtained with the language data sets earlier. In terms of clusteredness and numbers of probed families, only the PP data is near some of the benchmark data sets. The other three language learning task data sets are so much bigger that any further comparisons with benchmark data sets become blurred by this factor. Although we plan to pursue investigating any task bias of FAMBL, benchmark data will need to be found of which some at least should approach, and others should exceed the data set sizes of our language data. The generation of artificial data may be needed (Aha, 1992).

6 Discussion

We have reported on three case studies of applying abstraction methods in memory-based learning in a careful manner, to language learning tasks. In a first study, on learning

Task	data set size	# probed families	cluster-edness	median k	% memory item compr.
GP	675 745	31 224	1908	2	90.5
MS	573 544	18 783	8435	2	93.4
GS	675 745	37 457	1693	1	83.2
NP	251 124	5238	2413	3	97.7
PP	23 898	157	78	1	99.3
POS	1 046 152	11 397	479	2	98.0
audiology	226	72	7	1	60.5
kr vs kp	3198	137	69	3	95.2
mushroom	8124	23	11	40	99.7
nursery	12 961	682	198	2	94.2
soybean-l	683	90	9	2	84.3
splice	3190	334	128	3	87.7
tic-tac-toe	958	161	84	3	81.4
votes	435	38	19	1	87.7

Table 8: Comparison of data set size, average numbers of probed families, clusteredness, median family size (k), and memory item compressions, between the language data and a selection of symbolic UCI benchmark data, as measured in FAMBL's probing phase (averaged over 10-fold cross validation experiments).

grapheme-phoneme conversion from a moderately-sized data set, we found that the careful abstractors RISE, FAMBL, and IGTREE were able to equal the generalization accuracy of their pure memory-based counterpart IB1-IG. All best-performing algorithms implement (combinations of) weighting metrics; without them, any careful abstraction is harmful to generalization performance as compared to pure memory-based learning.

The second case study on morpho-phonological data sets of increasing size showed that careful abstraction (FAMBL with IG, RISE, and IGTREE) yielded competitive generalization accuracy as compared to pure memory-based learning (IB1-IG) mainly with small data sets. Any performance advantage of careful abstraction disappeared with the largest data sets. On the other hand, the compression capabilities of all tested careful abstractors increase with larger data sets.

In a third case study we applied the pure memory-based learning algorithm IB1-IG and the careful abstractors FAMBL and IGTREE to a range of language learning tasks (reproducing some of the work presented in Daelemans, Van den Bosch, and Zavrel (1999)). While IGTREE, which creates oblivious decision trees, performed worst overall, thus displaying harmful abstraction, FAMBL performed closer to IB1-IG, though equaling it on only three of the six tasks (grapheme-phoneme conversion, base-NP chunking, and part-of-speech tagging). Closer analyses of the learned models indicated that tasks such as word pronunciation may have properties (such as very high disjunctivity or local feature interaction) that FAMBL does not handle adequately.

Although the obtained results do not point at the superfluousness of pure memory-based learning, we have seen that carefully transforming an instance base into a set of generalized instances may yield compact models that perform close or equal to their pure memory-based counterparts. Careful abstraction in current approaches turns out to be not careful enough sometimes, and these current approaches may be failing to detect fine-grained and local feature interactions. Nevertheless, we believe these problems may be tackled within the framework of careful abstraction, yielding a general approach that may demonstrate that generalized instances can be working units in memory-based learning and processing.

In addition, we note, qualitatively and briefly, some interesting features of family-based learning that allow for further research. Consider the examples displayed in table 9 of actual family expressions as found by FAMBL on the five of the six language learning tasks investigated here (examples of the POS task are left out for reasons of clarity). For each task, three examples are given. Curly brackets bound the disjunctions of merged values at single features. We note

two general characteristics of families we see being extracted by FAMBL:

1. In every family, there is at least one feature (usually with a high, or the highest IG) that has one fixed value. In some examples, most or all features are fixed (e.g, in the 'dioxide' example of the M3 task, apparently disambiguating the segmentation between i and o in this particular word with other ...io... sequences in which this segmentation does not occur).
2. Values that are grouped on one feature are sometimes linguistically related (just as they will have, on average, small value difference, as value difference metrics compute precisely such class-related co-occurrences as happening in families, only on a global level). In cases where values are letters, graphematically and phonetically close groups, such as {o,e,i}, tend to reoccur. In cases where values are words, grouped values often appear to display some sort of syntactic–semantic relatedness.

In sum, there appears to be information hidden in extracted families that may be useful for other purposes, or for further abstraction. We now turn to our final remarks on future topics of research that elaborate on these indications.

Task	Example family expression	class
GP	_ {b,p,c,t,w} r i c k {k,l,i,s} {i,n,_} {l,e,g,_}	-
	{o,e,i} {u,s} {s,l,t} n e s s _ _	I
	_ _ _ r e {w,o} {r,i} {i,r} {t,e}	i
GS	{_,n,a} {n,c} o n - {f,v} {i,l,o}	0n
	{e,f,g,i,k,l,m,p,r,s,u,v} e t e d _ _	0I
	{_,o,y} s {y,a} n t {h,a} {e,c,x}	0n
MS	_ _ _ _ u n {d,s} i {g,d,s,v}	2
	_ _ d i o x i d e	c
	{j,u,r,h} {i,a} {g,n} g l i e r _	0
NP	the {news,notion,time,understanding} that {British, Mr.} DT NN IN NP I I I	O
	{of,solar} {vans,systems,reporters} and {light,TV} {IN,JJ } NNS CC NN I I {O,I}	O
	{sluggish,possible,second} {growth,sale,quarter} or {even,o ther,second} JJ NN CC JJ I I I	I
PP	{taken,'s,casts,has,is,play} {case,nothing,light,sketches,num ber,outfielder} on side	V
	boost stake in {conglomerate,business,maker}	N
	adding {confusion,argument,insult,land,measures,money,penny,voi ces} to ...	
	... {situation,arsenal,injury,residence,it,balances,tax,ch orus}	V

Table 9: Examples of probed families for five of the language learning tasks investigated in Sections 3–5.

6.1 Future research

Summarizing, we identify four strands of future research that we view as relevant follow-ups to the case studies described in this article and other work on abstraction in memory-based learning. First, from the language learning perspective:

- Families extracted in FAMBL show interesting grouping of values that may be generalized further, e.g. by summarizing frequently reoccurring groups using single identifiers or wild-cards. Moreover, merged value groups represent a sort of non-hierarchical clustering, that may be used as (or transformed into) an information source for the learning task itself, or to other related learning tasks.
- All memory-based learning in the approaches mentioned in this article ignore feature interaction: the phenomenon of feature combinations that also contribute significant information to classification when taken together. This may also refer to exploring the interaction of specific values of different features. The phenomenon is addressed in recent work on maximum entropy models applied to language learning (Ratnaparkhi, 1998) and Winnow algorithms (Golding and Roth, 1998 forthcoming). We view the incorporation of feature interaction in memory-based learning as feasible, certainly when an integrative approach is taken combining memory-based learning with these related areas.

As regards the learning and classification algorithms presently constituting FAMBL, we identify the following two topics of further research as most relevant:

- Asymptotic analyses of storage, learning, and classification in FAMBL need to be made, and at least compared with those for the related approaches RISE and NGE. FAMBL's current implementation is fast, but empirical comparisons of learning speed of different algorithms with different levels of optimizations in their implementations does not constitute sound data for concluding that FAMBL is also theoretically more efficient.
- Investigations should be performed into equipping FAMBL with less random and more sensible-heuristic-driven (yet non-parametric) probing and selection of families, provided that the current speed of FAMBL is not harmed seriously. It is important to investigate what unwanted effects may be caused by FAMBL's random-selection approach to probing and extraction of families, and how to possibly counteract them.

While a first target of these future investigations will be to make FAMBL handle certain types of data more adequately, we hope to arrive at an integrative view on careful abstraction and the nature of the basic storage units in memory-based learning in general, also connecting to maximum-entropy, Winnow, and possibly also connectionist approaches to abstraction from full memory storage.

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) unde rproject number 300–75–301. The author wishes to thank Walter Daelemans, Jakub Zavrel, Jorn Veenstra, Ton Weijters, Ko van der Sloot, and the other members of the Tilburg ILK group for stimulating discussions and support, and for their earlier work on the NP, PP, and POS data sets. Software code for the RISE algorithm was kindly provided by Pedro Domingos; Dietrich Wettschereck kindly granted permission to use his implementation of NGE.

References

Abney, S. 1991. Parsing by chunks. In *Principle-Based Parsing*. Kluwer Academic Publishers, Dordrecht.

Aha, D. W. 1992. Generalizing from case studies: a case study. In *Proceedings of the Ninth International Conference on Machine Learning*, pages 1–10, San Mateo, CA. Morgan Kaufmann.

Aha, D. W. 1997. Lazy learning: Special issue editorial. *Artificial Intelligence Review*, 11:7–10.

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

Atkeson, C., A. Moore, and S. Schaal. 1997. Locally weighted learning. *Artificial Intelligence Review*, 11(1–5):11–73.

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

Blake, C., E. Keogh, and C.J. Merz. 1998. UCI repository of machine learning databases.

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

Cardie, Claire. 1994. *Domain Specific Knowledge Acquisition for Conceptual Sentence Analysis*. Ph.D. thesis, University of Massachusets, Amherst, MA.

Cardie, Claire. 1996. Automatic feature set selection for case-based learning of linguistic knowledge. In *Proc. of Conference on Empirical Methods in NLP*. University of Pennsylvania.

Church, K. W. 1988. A stochastic parts program and noun phrase parser for unrestricted text. In *Proceedings of the Second Applied NLP Conference (ACL)*.

Clark, P. and R. Boswell. 1991. Rule induction with CN2: Some recent improvements. In *Proceedings of the Sixth European Working Session on Learning*, pages 151–163. Berlin: Springer Verlag.

Clark, P. and T. Niblett. 1989. The CN2 rule induction algorithm. *Machine Learning*, 3:261–284.

Collins, M.J and J. Brooks. 1995. Prepositional phrase attachment through a backed-off model. In *Proc. of Third Workshop on Very Large Corpora*, Cambridge.

Cost, S. and S. Salzberg. 1993. A weighted nearest neighbour 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., P. Berck, and S. Gillis. 1997. Data mining as a method for linguistic analysis: Dutch diminutives. *Folia Linguistica*, XXXI(1-2).

Daelemans, W. and A. Van den Bosch. 1992a. 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. Twente University.

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

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

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

Daelemans, W., J. Zavrel, P. Berck, and S. Gillis. 1996. MBT: A memory-based part of speech tagger generator. In E. Ejerhed and I.Dagan, editors, *Proc. of Fourth Workshop on Very Large Corpora*, pages 14–27. ACL SIGDAT.

Devijver, P. A. and J. Kittler. 1980. On the edited nearest neighbor rule. In *Proceedings of the Fifth International Conference on Pattern Recognition*. The Institute of Electrical and Electronics Engineers.

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

Dietterich, T. G. 1998. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation*, 10(7).

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.

Domingos, P. 1995. The RISE 2.0 system: A case study in multistrategy learning. Technical Report 95-2, University of California at Irvine, Department of Information and Computer Science, Irvine, CA.

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

Fix, E. and J. L. Hodges. 1951. Disciminatory analysis—nonparametric discrimination; consistency properties. Technical Report Project 21-49-004, Report No. 4, USAF School of Aviation Medicine.

Gates, G. W. 1972. The reduced nearest neighbor rule. *IEEE Transactions on Information Theory*, 18:431–433.

Golding, A. R. and D. Roth. 1998, forthcoming. A Winnow-based approach to spelling correction. *Machine Learning*.

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

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

Langley, P., W. Iba, and K. Thompson. 1992. An analysis of Bayesian classifiers. In *Proceedings of the Tenth Annual Conference on Artificial Intelligence*, pages 223–228. AAAI Press and MIT Press.

Lehnert, W. 1987. Case-based problem solving with a large knowledge base of learned cases. In *Proceedings of the Sixth National Conference on Artificial Intelligence (AAAI-87)*, pages 301–306, Los Altos, CA. Morgan Kaufmann.

Marcus, M., B. Santorini, and M.A. Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Michalski, R. S. 1983. A theory and methodology of inductive learning. *Artificial Intelligence*, 11:111–161.

Niblett, T. 1987. Constructing decision trees in noisy domains. In *Proceedings of the Second European Working Session on Learning*, pages 67–78, Bled, Yugoslavia. Sigma.

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

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

Ramshaw, L.A. and M.P. Marcus. 1995. Text chunking using transformation-based learning. In *Proc. of third workshop on very large corpora*, pages 82–94, June.

Ratnaparkhi, A. 1998. *Maximum Entropy Models for Natural Language Ambiguity Resolution*. Ph.D. thesis, University of Pennsylvania.

Ratnaparkhi, A., J. Reynar, and S. Roukos. 1994. A maximum entropy model for prepositional phrase attachment. In *Workshop on Human Language Technology*, Plainsboro, NJ, March. ARPA.

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

Salzberg, S. L. 1997. On comparing classifiers: Pitfalls to avoid and a recommended approach. *Data Mining and Knowledge Discovery*, 1(3).

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

Shavlik, J. W. and T. G. Dietterich, editors. 1990. *Readings in Machine Learning*. Morgan Kaufmann, San Mateo, CA.

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

Stanfill, C. 1987. Memory-based reasoning applied to English pronunciation. In *Proceedings of the Sixth National Conference on Artificial Intelligence (AAAI-87)*, pages 577–581, Los Altos, CA. Morgan Kaufmann.

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

Swonger, C. W. 1972. Sample set condensation for a condensed nearest neighbor decision rule for pattern recognition. In S. Watanabe, editor, *Frontiers of Pattern Recognition*. Academic Press, Orlando, Fla, pages 511–519.

Tomek, I. 1976. An experiment with the edited nearest-neighbor rule. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-6(6):448–452.

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.

Vapnik, V. and L. Bottou. 1993. Local algorithms for pattern recognition and dependencies estimation. *Neural Computation*, 5(6):893–909.

Veenstra, J. B. 1998. Fast NP chunking using memory-based learning techniques. In F. Verdenius and W. van den Broek, editors, *Proceedings of BENELEARN'98*, pages 71–79, Wageningen. ATO-DLO.

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

Wettschereck, D. 1994. 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 comparative evaluation of feature-weighting methods for a class of lazy learning algorithms. *Artificial Intelligence Review, special issue on Lazy Learning*, 11:273–314.

Wettschereck, D. and T. G. Dietterich. 1994. Locally adaptive nearest neighbor algorithms. In J. D. Cowan *et al.*, editor, *Advances in Neural Information Processing Systems*, volume 6, pages 184–191, Palo Alto,CA. Morgan Kaufmann.

Wettschereck, D. and T. G. Dietterich. 1995. An experimental comparison of the nearest-neighbor and nearest-hyperrectangle algorithms. *Machine Learning*, 19:1–25.

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. 1989. Constructing a generalizer superior to NETtalk via mathematical theory of generalization. *Neural Networks*, 3:445–452.

Zavrel, J., W. Daelemans, and J. Veenstra. 1997. Resolving PP attachment ambiguities with memory-based learning. In M. Ellison, editor, *Proc. of the Workshop on Computational Language Learning (CoNLL'97)*, ACL, Madrid.