# Instance Families in Memory-Based Language Learning

*Antal van den Bosch*

Tilburg University

## Abstract

Empirical studies in inductive language learning point at pure memory-based learning as a successful approach to many language processing tasks, often performing better than 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. We test this hypothesis by investigating a careful abstraction method that generalizes instances into instance families. The method is applied to a range of language learning tasks. Results show that the method reduces memory requirements to a reasonable to considerable degree, while being able to maintain the performance accuracy of pure memory-based learning on three of the six tasks studied. We discuss the inclusion of the concept of instance families as a working unit 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. It 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 1992, Cardie 1994, Cardie 1996, Daelemans, Zavrel, Berck and Gillis 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.

Nevertheless, the relative computational inefficiency of classification in pure memory-based learning remains a reason to investigate learning methods that economize on memory and the time needed for classification. Although earlier studies show that methods that forget tend to yield lower generalization accuracies, their results do provide indications that when forgetting is performed with care, the performance loss may be tolerable (Daelemans et al. 1999). In the current paper, we investigate *generalized instances*, a careful-abstraction approach within memory-based learning that aims to do roughly the same as decision-tree learning, while maintaining the basic learning and classification functions of pure memory-based learning. The idea underlying generalized instances is that certain sets of instances (i.e. examples of a task to be learned) are so much alike in terms of their behaviour in memory-based classification, that they may be replaced safely by a single more general instance that carries the joint functionality in memory-based classification of its merged members.

In this paper we investigate whether careful abstraction with generalized instances is a way of reducing memory requirements, while attaining a generalization performance at or close to the level of pure memory-based learning. To this end, we have performed a case study in which we apply FAMBL, a careful-abstracting memory-based learning algorithm that generalizes instances, to six language processing tasks: grapheme-phoneme conversion, word pronunciation, morphological segmentation, base-NP chunking, PP attachment and part-of-speech tagging. In this paper we describe the case study and discuss its results. Analogous to the results reported in Daelemans et al. (1999) on the effects of decision-tree abstraction, the results show that generalizing instances into instance family expressions leads to (i) reasonable to considerable memory item compression, and to (ii) equal or slightly lower generalization accuracy on the six investigated tasks.

The paper is structured as follows. Section 2 provides a brief introduction into memory-based learning, and presents the FAMBL algorithm. Section 3 describes the empirical case study in which FAMBL, in comparison with its parent (pure memory-based) learning algorithm IB1-IG, is applied to the six language learning tasks. In Section 4, the efficacy of careful generalization over families of instances is discussed, and the idea of viewing these families as linguistic units is outlined.

## 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 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. 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 instances with symbolic features is the *overlap metric* $\Delta(X,Y) = \sum_{i=1}^{n} \delta(x_i, y_i)$, i.e. the distance between patterns $X$ and $Y$, represented by $n$ features, where $\delta = 0\ if\ x_i = y_i,\ else\ 1$ (the distance between matching values is zero, and is one between mismatching values). 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 $k$, which determines the number of nearest neighbours on which classification is based, usually set to 1.

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 et al. 1991, Salzberg 1991) yielded several implementations of ideas on memory-based algorithms that invest some effort in the learning phase in generalizing instances. The proposed algorithms start with storing individual instances in memory, and then carefully merge groups of nearest-neighbour instances labeled with the same class 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 instances as the basic working units on which the same $k$-NN-based classification can be performed. The abstraction occurring in this approach is that after a merge, the merged instances incorporated in the new generalized instance are deleted individually, and cannot be reconstructed. In this paper we use the FAMBL algorithm to test this kind of generalization over instances.

## 2.1 FAMBL: generalizing instances into instance families

FAMBL, for *FAMily-Based Learning*, is an algorithm that performs careful abstraction over instances. The core idea of FAMBL is to transform an instance base into a set of *instance family expressions*. We outline the ideas and assumptions underlying FAMBL. We then give a procedural description of the learning algorithm.

### 2.1.1 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 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 scattering of classes into clusters. Many types of language data appear to be quite disjunct (Daelemans et al. 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.

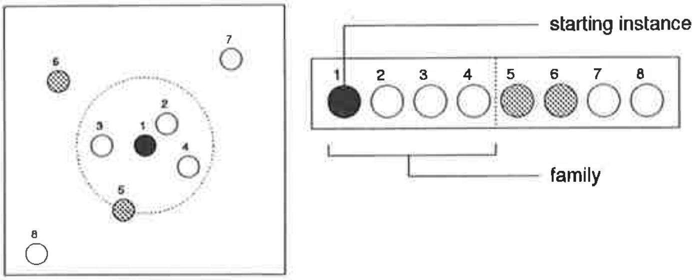

Figure 1: An example of a family in a two-dimensional instance space (left). The family, at the inside of the dotted circle, spans the focus instance (black) and the three nearest neighbors labeled with the same class (white). When ranked in the order of distance (right), the family boundary is put immediately before the first instance of a different class (grey).

Figure 1 illustrates how FAMBL determines the family of an instance in a simple two-dimensional euclidean 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.

When an instance's family is determined, the whole family (including the starting instance) is converted in FAMBL to *family expressions*, i.e. generalized instances, by merging all instances belonging to that family simultaneously. Figure 2 illustrates the creation of a family expression from an instance family. It shows five instances of grapheme-phoneme mappings. Each instance denotes a focus letter ('c' in all five instances) surrounded by four left and four right neighbouring letters, and each is associated with the phoneme denoting the pronunciation of the middle letter ('X' in all five instances, i.e. a /ks/ pronunciation). All values of all

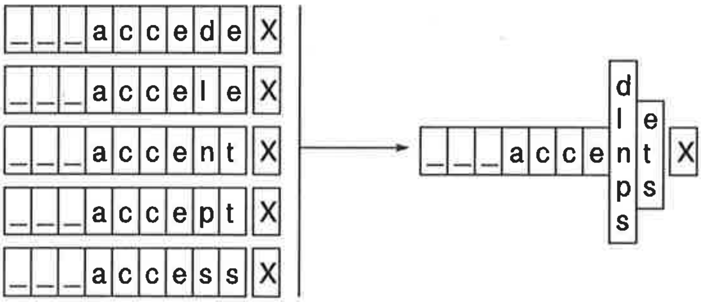

Figure 2: An example of family creation in FAMBL. Five grapheme-phoneme instances (left) are merged into a family expression (right).

features are merged into the family expression displayed in the right of Figure 2. When feature values mismatch between the family members, they are summed up as a disjunction of values at their particular feature slot.

The general modus of operation of FAMBL is that it selects instances in a random sequence 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.

### 2.1.2 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 4) is a preprocessing stage to the actual family extraction as outlined in the previous Subsection. The reason for preprocessing is visualized in Figure 3. The random selection of starting points for family creation can be quite unfortunate. When, for example, the middle instance in the left part of Figure 3 is selected first, a seven-instance family is formed with relatively large distances between the family members. 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 3 displays a much more desirable situation, in which the space is more evenly divided between only two families instead of four.

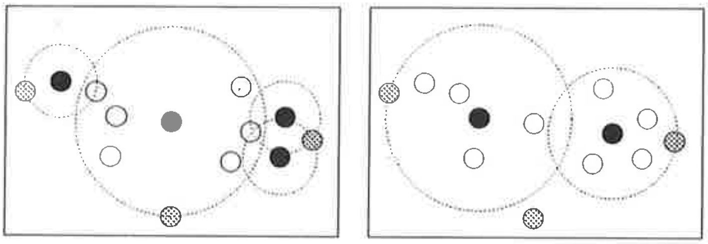

Figure 3: Illustration of the need for the preprocessing stage in FAMBL. The left figure shows a big family with seven members, forcing the remaining three instances to be their own family. The right figure shows the same space with two other starting points for family creation, with two more evenly divided families. Black instances denote starting points for family creation; white instances have the same class as the starting points, and grey instances have a different class.

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

```
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 4: Schematized overview of the probing phase in FAMBL.

distance between the starting instance of each family and the other instances in the family. When all instances are captured in families, the medians 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, schematized in Figure 5, (i) no family is extracted that has more members than the probed median number of members, and (ii) no family is extracted that has an average distance from the starting instance to the other family members larger than the probed median value. Summarizing, 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. This is a heuristic solution to the problem depicted in Figure 3, and is not guaranteed to solve all of the problematic situations as depicted in the left part of the figure. Further research should focus on devising a more principled solution, that will probably need to take the local neighbourhood of families into account, rather than following heuristically-set thresholds that reflect only global aspects of the data.

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 is analogous to classification in pure memory-based learning: 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 cer-

```
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 this expression
          along with a count of the number of instance merged in it
        • $f = f + 1$
```

Figure 5: Schematized overview of the family-extraction phase in FAMBL.

tain 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.

FAMBL allows for the inclusion of informational abstraction in the form of feature-weighting, instance-weighting and value-difference metrics. To compare FAMBL to pure memory-based learning, as described in the next section, we have included information-gain feature weighting (Quinlan 1986, Daelemans and Van den Bosch 1992) in FAMBL. Information-gain feature weighting is used to differentiate between features that are important to the classification task at hand, and features that are not. In addition to classification in pure memory-based learning, it introduces a factor $w_i$ in the distance function $\Delta(X,Y)$ given earlier: $\Delta(X,Y) = \sum_{i=1}^{n} w_i\, \delta(x_i, y_i)$. $w_i$, the information gain of feature $i$, is 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: $w_f = \frac{H(C) - \sum_{v \in V_f} P(v) \times H(C|v)}{si(f)}$, where $si(f) = -\sum_{v \in V_f} P(v) \log_2 P(v)$, $C$ is the set of class labels, $V_f$ is the set of values for feature $f$, and $H(C) =$

| | # | # Values of feature | | | | | | | | | | | # | # Data set |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Task | Features | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | Classes | 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 |
| 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 1: Specifications of the six data sets of the GP, GS, MS, NP, PP, and POS learning tasks: numbers of features, values per feature, classes, and instances.

$-\sum_{c \in C} P(c) \log_2 P(c)$ 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 favor of features with more values (Quinlan 1993). Pure memory-based learning with information-gain feature weighting was introduced in Daelemans and Van den Bosch (1992) as an extension to IB1 (Aha et al. 1991), and was called IB1-IG. In numerous language learning studies, IB1-IG has been shown to outperform the unweighted IB1 (Van den Bosch 1997, Daelemans et al. 1999).

We conclude our description of the FAMBL algorithm by noting that it is related to two earlier approaches to generalized instances: NGE (Salzberg 1991) and RISE (Domingos 1996). Part of the motivation for the design of FAMBL is to have a faster version of the process of generalizing instances as compared to that in NGE and in RISE. Current implementations of the latter two algorithms scale badly with data sets of the magnitude as discussed in this paper. The main difference between FAMBL and both NGE and RISE is that FAMBL is not incremental (NGE) or cyclic (RISE): families in FAMBL are extracted once and not grown later on. An additional differences between FAMBL and RISE is that RISE uses wildcards to represent a disjunction of feature values in a generalized expression, while FAMBL always lists the full disjunction of feature values. For more discussion on NGE and RISE and an empirical comparison with FAMBL, cf. Van den Bosch (1999, to appear).

## 3 Careful abstraction applied to six language learning tasks

We applied FAMBL to six language data sets that have been used in earlier studies (Van den Bosch, Daelemans and Weijters 1996, Veenstra 1998, Daelemans et al. 1999, Van den Bosch 1999, to appear). The 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. Table 1 lists the numbers of instances, feature values, and classes of the data sets of the six tasks. We briefly outline the underlying tasks that these three data sets represent.

**Grapheme-phoneme conversion** (henceforth referred to as GP) is the mapping between the spelling of a word and its phonemic transcription. We define the task as the mapping 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). Example instances and their classifications were given earlier in Figure 2. 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). The data used in the experiments described here are derived from the CELEX lexical data base of English (Baayen, Piepenbrock and van Rijn 1993).

**Grapheme-phoneme conversion combined with stress assignment** (henceforth referred to as GS) is similar to the GP task, but differs in two respects: (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/ being the first of a syllable receiving primary stress (e.g. as in the instance ___book from the word booking). See Van den Bosch (1997) for more details.

**Morphological segmentation** (henceforth MS) 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', etc. See Van den Bosch (1996) for more details.

**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 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 et al. (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 et al. (1994) and Collins and Brooks (1995), and in a memory-based learning experiment by Zavrel, Daelemans and Veenstra (1997), Daelemans et al. (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. 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).

For each task FAMBL is compared with IB1-IG by applying both algorithms to the six data sets in a 10-fold cross validation setup (Weiss and Kulikowski 1991). Table 2 lists the average generalization accuracies obtained in these comparisons. The results of IB1-IG on the GS, NP, PP, and POS tasks are reproduced from Daelemans et al. (1999). One-tailed $t$-tests yield significance results that show at a general level that IB1-IG is significantly more accurate than FAMBL on the MS, GS, and PP tasks. On the GP, NP, and POS tasks, the difference is not significant.

| | IB1-IG | | | FAMBL-IG | |
|---|---|---|---|---|---|
| Task | % | ± | >FAMBL? | % | ± |
| GP | 97.37 | 0.09 | | 97.32 | 0.09 |
| MS | 98.02 | 0.05 | *** | 97.84 | 0.06 |
| GS | 93.45 | 0.15 | ** | 93.22 | 0.24 |
| NP | 98.07 | 0.05 | | 98.04 | 0.05 |
| PP | 83.48 | 1.16 | ** | 81.80 | 1.14 |
| POS | 97.86 | 0.05 | | 97.83 | 0.04 |

Table 2: Generalization accuracies (percentages of correctly classified test instances, with standard deviations) of IB1-IG and FAMBL on the GP, GS, MS, NP, PP, and POS tasks. Asterisks denote the outcomes of one-tailed $t$-tests, denoting a significantly better accuracy of IB1-IG as compared to that of FAMBL. '**' denotes $p < 0.01$; '***' denotes $p < 0.001$.

We also 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 3 displays these results averaged over the ten experiments performed on each task. The table also displays two additional quantities:

| | generalization accuracy | | probing stage | | | median | | extraction stage | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Task | % | ± | # | cluster-edness | % item compr. | size | distance | # | 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 3: 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 2.

(i) the measured *clusteredness*, i.e. 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—note that this item compression cannot be translated directly to raw memory compression, since family expressions take up more space than instance types). Both quantities are reported for the probing stage as well as the family stage.

Table 3 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 median thresholds computed in the probing phase.) The thresholds on family size found in the probing stages show that family extraction is strictly limited to size one 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. Together with the high clusteredness, these numbers indicate that all six datasets are highly disjunct. Families seldom contain more than a handful of instances (the PP data set being the exception). Put in other words, when two instances mismatch in only a few features, they can be expected to have a different class (two out of seven in the case of the GS task; two out of four with PP; three out of nine with GP and MS; three out of five with POS; or four out of eleven with NP).

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, and the highest compression is obtained with NP and POS. On both latter tasks FAMBL equaled IB1-IG. This would suggest that the underlying assumptions of FAMBL apply successfully to the NP and POS tasks, and that the GS task data has properties that FAMBL handles less adequately. We have two working hypotheses on what these properties might be. First, 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 3 when data is extremely disjunct. Second, 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 1, 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/.

The incorporation of feature interaction, both in FAMBL and in IB1-IG, is a relevant focus point for future research (cf. recent work on maximum entropy models applied to language learning (Ratnaparkhi 1998) and Winnow algorithms (Golding and Roth 1998, forthcoming)).

## 4 Discussion

In a case study we applied the pure memory-based learning algorithm IB1-IG and its careful-abstracting variant FAMBL to a range of language learning tasks. FAMBL performed close 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. On the other hand, FAMBL was shown to be able to reduce the number of items in memory considerably (which, depending on the implementation, can translate to substantial reduction of allocated computer memory), particularly for base-NP chunking (72%) and part-of-speech tagging (75%).

The results on the six selected tasks again provide indications for the claim that has been stated on several occasions now (Van den Bosch 1997, Daelemans et al. 1999, Van den Bosch 1999, to appear), that language data are highly disjunct. Learning algorithms that ignore, forget, or cut across disjuncts (families) tend to lose generalization accuracy (Daelemans et al. 1999). Learning algorithms that attempt to preserve the boundaries that separate small disjuncts from each other, be it decision trees that are forced to be more careful than default (Daelemans et al. 1999), or carefully-abstracting memory-based learners such as FAMBL that produce generalized instances, attain levels of generalization performance that are equal to, or close to, those of pure memory-based learners.

In addition, we note, qualitatively and briefly, some interesting features of

| 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,other,second} . . . | |
| | . . . JJ NN CC JJ I I I | I |
| PP | {taken,'s,casts,has,is,play} {case,nothing,light,sketches,number,outfielder} on side | V |
| | boost stake in {conglomerate,business,maker} | N |
| | adding {confusion,argument,insult,land,measures,money,penny,voices} to . . . | |
| | . . . {situation,arsenal,injury,residence,it,balances,tax,chorus} | V |

Table 4: Examples of probed families for five of the language learning tasks investigated in Section 3.

family-based learning that allow for further research. Consider the examples displayed in Table 4 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. First, in every family, there is at least one feature (usually with a high 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. . . instances in which there is no morphological segmentation. This high degree of 'fixed' values is reflected also in the low median family sizes (Table 3). Again, if instances differ in more than a few feature values, they can be expected to have another class. Second, values that are grouped on one feature are sometimes linguistically related. 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, 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.

**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). The author wishes to thank Walter Daelemans, Jakub Zavrel, Jorn Veenstra, 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.

## References

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

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

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

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

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

Cardie, C.(1994), *Domain Specific Knowledge Acquisition for Conceptual Sentence Analysis*, PhD thesis, University of Massachusets, Amherst, MA.

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

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

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

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

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

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

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

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

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

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

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

Quinlan, J.(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. and Marcus, M.(1995), Text chunking using transformation-based learning, *Proc. of Third Workshop on Very Large Corpora*, pp. 82–94.

Ratnaparkhi, A.(1998), *Maximum Entropy Models for Natural Language Ambiguity Resolution*, PhD thesis, University of Pennsylvania.

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

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

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

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

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

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

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

Van den Bosch, A. (1999, to appear), Careful abstraction from instance families in memory-based language learning, In W. Daelemans (Ed.), *Journal of Experimental and Theoretical Artificial Intelligence*. special issue on Memory-Based Language Processing.

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

Veenstra, J. B.(1998), Fast NP chunking using memory-based learning techniques, *in* F. Verdenius and W. Van den Broek (eds), *Proceedings of BENELEARN'98*, ATO-DLO, Wageningen, pp. 71–79.

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

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