Learning

Forgetting Exceptions is Harmful in Language Learning

WALTER DAELEMANS, ANTAL VAN DEN BOSCH, JAKUB ZAVREL

ILK // Computational Linguistics
Tilburg University
P.O. Box 9015
NL-5000 LE Tilburg
The Netherlands
Phone +31/13/4666700

WALTER DAELEMANS, ANTAL VAN DEN BOSCH, JAKUB ZAVREL

Abstract: We show that in language learning, contrary to received wisdom, keeping except

Keywords: memory-based learning, natural language learning, edited nearest neighbor classifier, decision-tree learning

1. Introduction

Memory-based reasoning (Stanfill and Waltz, 1986) is founded on the hypothesis that performance in real-world tasks (in our case language processing) is based on reasoning on the basis of similarity of new situations to stored representations of earlier experiences, rather than on the application of mental rules abstracted from earlier experiences as in rule-based processors. The type of reasoning associated with such an approach is called lazy learning (Aha, 1996). The approach has emerged in different contexts using a variety of alternative names such as: exemplar-based, case-based, instance-based, locally weighted, nearest-neighbor, and Montessori-based (Stark and Yilmaz, 1996; Card and Duda, 1997; Kibler, 1992). 

Exceptional training examples are a part of the minimal training instances and the learning algorithm must be able to reason with them. A possible explanation for this behavior is that the degree of abstraction of earlier experiences can be used in the design of abstraction functions. ...
Memory-based learning is lazy as it involves adding training examples (feature-value vectors with associated categories) to memory without abstraction or restructuring. During classification, a previously unseen test example is presented to the system. Its similarity to all examples in memory is computed using a similarity metric, and the category of the most similar example(s) is used as a basis for extrapolating the category of the test example. A key feature of memory-based learning is that, normally, all examples are stored in memory and no attempt is made to simplify the model by eliminating noise, low-frequency events, or exceptions. Although it is clear that noise in the training data can harm accurate generalization, this work focuses on the problem that, for language learning tasks, it is very difficult to discriminate between noise on the one hand, and valid exceptions and sub-regularities that are important for reaching good accuracy on the other hand.

The goal of this paper is to provide empirical evidence that for a range of language learning tasks, memory-based learning methods tend to achieve better generalization accuracies than (i) memory-based methods combined with training set editing techniques in which exceptions are explicitly forgotten, i.e., removed from memory, and (ii) decision-tree learning in which some of the information from the training data is either forgotten (by pruning) or made inaccessible (by the eager construction of a model). We explain these results in terms of the properties of memory-based learning and the data characteristics of the tasks.

First, we show in Section 4 that two criteria for editing instances in memory-based learning, viz. low typicality and low class prediction strength, are generally responsible for a decrease in generalization accuracy.

Second, memory-based learning is demonstrated in Section 5 to be mostly at an advantage, and sometimes at a par with decision-tree learning as far as generalization accuracy is concerned. The advantage is puzzling at first sight, as both decision-tree and memory-based learning methods are based on similar principles: (i) classification of test instances on the basis of their similarity to training instances (in the form of instances themselves in ib1-ig or in the form of hyper-rectangles containing subsets of partly-similar training instances in c50 and igtree), and (ii) use of information entropy as a heuristic to constrain the space of possible generalizations (as a feature weighting method in ib1-ig, and as a split criterion in c50 and igtree).

Our hypothesis is that both effects are due to the fact that ib1-ig keeps all training instances as possible sources for classification, whereas both the edited versions of ib1-ig and the decision-tree learning algorithms c50 and igtree make abstraction of test instances with associated categories (to improve without distortion of the feature-value vectors with associated categories).
tions from irregular and low-frequency events. In language learning tasks, where sub-regularities and small families of exceptions typically abound, the latter is detrimental to generalization performance. Our results suggest that forgetting exceptional training instances is harmful to generalization accuracy for a wide range of language-learning tasks. This finding contrasts with a consensus in supervised machine learning that forgetting exceptions by pruning boosts generalization accuracy (Quinlan, 1993), and with studies emphasizing the role of forgetting in learning (Markovich and Scott, 1988; Salganico, 1993).

Section 6 places our results in a broader machine learning and language learning context, and attempts to describe the properties of language data and memory-based learning that are responsible for the 'forgetting exceptions is harmful' effect. For our data sets, the abstraction and pruning techniques studied do not succeed in reliably distinguishing noise from productive exceptions, an effect we attribute to a special property of language learning tasks: the presence of many exceptions that tend to occur in groups or pockets in instance space, together with noise introduced by corpus coding methods. In such a situation, the best strategy is to keep all training data to generalize from.

2. Learning methods

In this section, we describe the three algorithms we used in our experiments. IB1-IG is used for studying the effect of editing exceptional training instances, and in a comparison to the decision tree methods C5.0 and IGTree.

IB1-IG (Daelemans and Van den Bosch, 1992; Daelemans, Van den Bosch, and Weijters, 1997) is a memory-based (lazy) learning algorithm that builds a database of instances (the instance base) during learning. An instance consists of a fixed-length vector of feature-value pairs, and a 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 between the new instance \( X \) and the stored instance \( Y \). The most basic metric for instances with symbolic features is the overlap metric given in Equations 1 and 2; where \( h(x) = 1 \) if \( x \) and \( y \) match, and \( h(x) = 0 \) otherwise. The distance between instances \( X \) and \( Y \) is therefore the distance per feature, and is given by the expression:

\[
(d(x,y)) = \frac{\sum_{i=1}^{n} w_i d(x_i,y_i)}{\sum_{i=1}^{n} w_i}
\]

where:

\[
h(x) = \frac{\sum_{i=1}^{n} w_i d(x_i,y_i)}{\sum_{i=1}^{n} w_i}
\]

and:

\[
h(x) = \prod_{i=1}^{n} h(x_i)
\]

These two metrics are used in our experiments. In a comparison to the decision tree methods C5.0 and IGTree, we describe the three algorithms we used in our experiments.
We have made two additions to the original algorithm in our version of lib1. First, in the case of nearest neighbor sets larger than one instance (k > 1 or ties), our version of lib1 selects the classification with the highest frequency in the class distribution of the nearest neighbor set. Second, if a tie cannot be resolved in this way because of equal frequency of classes among the nearest neighbors, the classification is selected with the highest overall occurrence in the training set.

The distance metric in Equation (2) simply counts the number of matching feature values in both instances. In the absence of information about feature relevance, this is a reasonable choice. Otherwise, we can add linguistic bias to weight or select different features (Cardie, 1999b). We can compute statistics about the relevance of features by looking at which features are good predictors of the class labels. Information gain (IG) weighting looks at each feature in isolation and measures how much information it contributes to our knowledge of the correct class label. The information gain of feature \( f \) is measured by computing the difference in uncertainty (i.e., entropy) between the situations without and with knowledge of the value of that feature (Equation (3)).

\[
\text{IG}(f) = H(C) - \frac{1}{|A|} \sum_{a \in A} \left( \sum_{v \in V_f} \frac{P(v)}{P(f)} \log \frac{P(v)}{P(v)} \right)
\]

where \( C \) is the set of class labels, \( A \) is the set of values for feature \( f \), and \( H(C) \) is the entropy of the class label probability distribution. The probabilities are estimated from relative frequencies in the training set. The normalizing factor \( \frac{1}{|A|} \) is included to avoid a bias in favor of features with more values. It represents the amount of information needed to represent all feature values. The possibilities are estimated from relative frequencies in the training set. The resulting IG values can then be used as weights in Equation (1).
The prototype algorithm was originally developed as a method to compress and in-
crease query and memory load time performance (Quinlan, 1993). The
ance of a commercial version of C4.5 (Quinlan, 1993), performs top-down induction

2.3. IGGREE

seting picks higher evaluation accuracy for the c demonstration.

2.3. IGGREE

seting picks higher evaluation accuracy for the c demonstration.

2.3. IGGREE

seting picks higher evaluation accuracy for the c demonstration.

2.3. IGGREE

seting picks higher evaluation accuracy for the c demonstration.

2.3. IGGREE

seting picks higher evaluation accuracy for the c demonstration.

2.3. IGGREE

seting picks higher evaluation accuracy for the c demonstration.

2.3. IGGREE

seting picks higher evaluation accuracy for the c demonstration.

2.3. IGGREE

seting picks higher evaluation accuracy for the c demonstration.

2.3. IGGREE

seting picks higher evaluation accuracy for the c demonstration.
3. Benchmark Language Learning Tasks

We investigate four language learning tasks that jointly represent a wide range of different types of tasks in the NLP domain: (1) Grapheme-Phoneme Conversion, (2) Part-of-Speech Tagging (POS), (3) Prepositional Phrase Attachment (PPA), and (4) Base Noun Phrase Chunking (NP). In this section,
We introduce each of the four tasks, and describe for each task the data collected.
We define the task as the conversion of fixed-sized instances representing parts of words to a class representing the phoneme and the stress marker of the instance's middle letter. We henceforth refer to the task as gs, an acronym of grapheme-phoneme conversion and stress assignment. To generate the instances, windowing is used (Sejnowski and Rosenberg, 1987). Table 2 (top) displays four example instances and their classifications. Classifications, i.e., phonemes with stress markers, are denoted by composite labels. For example, the first instance in Table 2, hearts, maps to class label 0A:0, denoting an elongated short /ə/-sound which is not the first phoneme of a syllable receiving primary stress. In this study, we choose a fixed window of seven letters, which offers sufficient context information for adequate performance of the task. We extract from this corpus, which contains 255,000 word instances, the phonetic transcription, POS: 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 (Charniak, 1993). For example, in the sentence "they can canacan," the word can is tagged as modal verb, main verb, and noun, respectively. We assume a tagger architecture that processes a sentence from the left to the right by classifying instances representing the words in their contexts (as described in Daelemans et al., 1996). The words' 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. The data set for the part-of-speech tagging task, henceforth referred to as the pos task, was extracted from the LOB corpus. The full data set contains 1,046,152 instances. The "lexicon" of ambiguity classes was constructed from the first 90% of the corpus only, and hence the data contains unknown words. To avoid a complicated architecture, we treat unknown words the same as known words, i.e., their ambiguous category is simply "unknown," and they can only be classified on the basis of their context. A disambiguating approach is simply not known, and they can only be classified on the basis of their context.
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 example, the sentence "He eats pizza with a fork." would yield the pattern: eats, pizza, with, fork, verb.

The pp data set is also known as the detection of boundaries between phrases (e.g.

Phrase chunking is defined as the detection of boundaries between phrases (e.g., noun phrases or verb phrases) in sentences. Chunking can be seen as a 'light' form of parsing. In NP chunking, sentences are segmented into non-recursive NP's, so-called baseNP's (Abney, 1991). NP chunking can, for example, be used to reduce the complexity of sub-sequential parsing or to identify named entities for information retrieval.

To perform this task, we used the baseNP tag set as presented in (Ramshaw and Marcus, 1995):

I for inside a baseNP, O for outside a baseNP, and B for the first word in a baseNP following another baseNP. As an example, the phrase "The postman gave the man a letter." would be represented as [The postman, I], [the, O], [man, B] and [a letter, I].

Due to the large number of possible word combinations and the contrived nature of the test set, our NP chunker often produced incorrect results. However, when given the four-tuple, the humans, when given the whole sentence, performed with an accuracy of 88.2% and 90.6%, respectively.

Due to the large number of possible word combinations and the contrived nature of the test set, our NP chunker often produced incorrect results. However, when given the four-tuple, the humans, when given the whole sentence, performed with an accuracy of 88.2% and 90.6%, respectively.
consists of two stages, and in this paper we have used instances from the second stage. An instance (constructed for each focus word) consists of features referring to words, POS tags, and IOB tags (predicted by the first stage) of the focus and the two immediately adjacent words. The dataset contains a total of 251,124 instances.

Experimental method

We used 10-fold CV (Weiss and Kulikowski, 1991) in all experiments comparing classifiers (Section 5). In this approach, the initial data set at the level of instances is partitioned into ten subsets. Each subset is taken in turn as a test set, and the remaining nine combined to form the training set. Means are reported, as well as standard deviation from the mean. In the editing experiments (Section 4), the first train-test partition of the 10-fold CV was used for comparing the effect on the test set classification accuracy of applying different editing schemes on the training set.

Having introduced the machine learning methods and data sets, we focus on the effect of editing exceptional instances in memory-based learning, which is the topic of this paper.

Editing exceptions in memory-based learning is harmful

The editing of instances from memory in memory-based learning or the k-nearest-neighbor classifier (Hart, 1968; Wilson, 1972; Devijver and Kittler, 1980) serves two objectives: to minimize the number of instances in memory for reasons of speed or storage, and to minimize generalization error by removing noisy instances, prone to being responsible for generalization errors. Two basic types of editing, corresponding to these goals, can be found in the literature:

Editing superfluous regular instances: delete instances for which the deletion does not harm the classification accuracy of their own class in the training set (Hart, 1968).

Editing unproductive exceptions: delete instances that are incorrectly classified by their neighbors in the training set (Wilson, 1972), or vice versa, delete instances that are bad class predictors for their neighbors (Aha, Kibler, and Albert, 1991).

We present experiments in which both types of editing are performed on the basis of two criteria that estimate the exceptionality of instances: typicality (Zhang, 1992) and class prediction strength (Salzberg, 1990) (henceforth referred to as CPS). Unproductive exceptions are edited by taking the instances with the lowest typicality or CPS, and superfluous regular instances are edited by taking the instances with the highest typicality or CPS. Experimental results are obtained by running the classifier with the basis of the two editing criteria for each of the two editing types, and comparing the performance on the test set. The two types of editing are performed on the basis of the two criteria that estimate the exceptionality of instances.
We present the results of the editing experiments in Subsection 4.2.

To evaluate exceptionalism, we define a new metric called 'exceptional centroid' (EC).
An instance type is typical when its intra-concept similarity is larger than its inter-concept similarity, which results in a typicality larger than 1. An instance type is atypical when its intra-concept similarity is smaller than its inter-concept similarity, which results in a typicality between 0 and 1. Around typicality value 1, instances cannot be sensibly called typical or atypical; Zhang (1999) refers to such instances as boundary instances.

We adopt typicality as an editing criterion here, and use it for editing instances with low typicality as well as instances with high typicality. Low-typical instances can be seen as exceptions, or bad representatives of their own class and could therefore be pruned from memory, as one can argue that they cannot support productive generalizations. This approach has been advocated by Ting (1998) as a method to achieve significant improvements in certain domains.

Typical instances are parts of words of which the focus letter is always pronounced the same way. Low-typical instances tend to involve inconsistent or noisy associations between an unambiguous word class of the focus word and a different word class as classification. Such inconsistencies can be largely attributed to corpus annotation errors. Focus tags of high-typical pos instances are already unambiguous. The examples of low-typical pp instances represent minority exceptions or noisy instances in which it is questionable whether the chosen classification is right (recall that human annotators agree only on 88% of the instances in the data set, cf., Subsection 3). While the low-typical np instances seem to be partly noisy, other otherwise difficult to interpret. High-typical instances are clear-cut cases in which a noun occurring between a determiner and a finite verb is correctly classified as being inside an NP.
### Table 1

**Examples of low- and high-typical instances of the gs, pos, pp, and np learning tasks.** For each instance, its typicality value is given. A class-prediction strength of 0 indicates that the instance type is a bad predictor of its own class, and a perfect predictor of another instance type regardless of the class. An instance is the nearest neighbor of another instance type regardless of the class.

<table>
<thead>
<tr>
<th>Feature values</th>
<th>Class</th>
<th>Typicality</th>
</tr>
</thead>
<tbody>
<tr>
<td>Call the lawyer's phone number</td>
<td>nom</td>
<td>0.99</td>
</tr>
<tr>
<td>Understand the latest stress level</td>
<td>nom</td>
<td>0.99</td>
</tr>
<tr>
<td>Exchange coordinates of the room</td>
<td>nom</td>
<td>0.99</td>
</tr>
<tr>
<td>Directs the officer through the intersection</td>
<td>noun</td>
<td>0.99</td>
</tr>
<tr>
<td>Cleanse German of muck</td>
<td>verb</td>
<td>0.99</td>
</tr>
<tr>
<td>Excluding categories of food</td>
<td>noun</td>
<td>0.99</td>
</tr>
<tr>
<td>Underscoring lack of stress</td>
<td>noun</td>
<td>0.99</td>
</tr>
<tr>
<td>Calls frenzy of legislating</td>
<td>noun</td>
<td>0.99</td>
</tr>
<tr>
<td>Generally a bit safer</td>
<td>rb</td>
<td>0.99</td>
</tr>
</tbody>
</table>

**Table 2**

<table>
<thead>
<tr>
<th>Feature values</th>
<th>Class</th>
<th>Typicality</th>
</tr>
</thead>
<tbody>
<tr>
<td>that the company hopes in the next quarter</td>
<td>OB</td>
<td>1</td>
</tr>
<tr>
<td>that the bank supports the loan with the next quarter</td>
<td>OB</td>
<td>1</td>
</tr>
<tr>
<td>that the professor wins in the next quarter</td>
<td>OB</td>
<td>1</td>
</tr>
<tr>
<td>I know that we are going to win the next quarter</td>
<td>OB</td>
<td>1</td>
</tr>
<tr>
<td>I know that we are not going to win the next quarter</td>
<td>OB</td>
<td>1</td>
</tr>
</tbody>
</table>

- Class-prediction strength of 1 indicates a perfect predictor of its own class, and a bad predictor of another instance type regardless of the class. An instance is the nearest neighbor of another instance type regardless of the class.
with a maximal cps could be edited to some degree without harming generalization: strong class predictors may be abundant and some may be safely forgotten since other instance types may be strong enough to support the class predictions. In Table 4, examples from the four tasks of instances with low (top three) and high (bottom three) cps are displayed. Many instances of overlap between the removed types were measured.

To measure to what degree the two criteria are indeed different means of measuring the overlap of instance types, the percentage of overlap was measured for each data set. As can be seen in Figure 1, the two measures mostly have little overlap, certainly for editing below 10%. The reason for this is that typicality is based on global properties of the data set, whereas class prediction strength is based only on the local neighborhood of each instance. For example, there are more words beginning with algo than those beginning with 2ae. The overlap of instance types is also displayed. Many instances of overlap between the removed types were measured.

In Table 4, examples from the four tasks were used to justify the selection of specific training instances. We performed a series of experiments in which each of the four data sets, systematical editing, and the four typicality and class prediction strength criteria were used to justify forgetting specific training instances. For each of the four data sets, we created eight edited instance bases by removing 1%, 2%, 5%, 10%, 20%, 50%, and 100% of the instance tokens according to the criterion from a single training set (the training set of the first 10-fold CV partition). IB/IG was then trained on each of the edited training sets, and tested on the original unedited test set (of the first 10-fold CV partition).
### Examples of instances with low class prediction strength (top three) and high class prediction strength (bottom three) of the gs, pos, pp, and np tasks. For each instance, the class prediction strength is shown along with the feature values.

#### gs

<table>
<thead>
<tr>
<th>Feature Values</th>
<th>Class Prediction Strength</th>
</tr>
</thead>
<tbody>
<tr>
<td>work force are well</td>
<td>low</td>
</tr>
<tr>
<td>share price could zoom</td>
<td>low</td>
</tr>
<tr>
<td>I think you will try to</td>
<td>high</td>
</tr>
</tbody>
</table>

#### pos

<table>
<thead>
<tr>
<th>Feature Values</th>
<th>Class Prediction Strength</th>
</tr>
</thead>
<tbody>
<tr>
<td>be one of program</td>
<td>low</td>
</tr>
<tr>
<td>be one of restructuring</td>
<td>low</td>
</tr>
<tr>
<td>be one of stimulus</td>
<td>low</td>
</tr>
</tbody>
</table>

#### pp

<table>
<thead>
<tr>
<th>Feature Values</th>
<th>Class Prediction Strength</th>
</tr>
</thead>
<tbody>
<tr>
<td>allowed access notwithstanding designations</td>
<td>high</td>
</tr>
<tr>
<td>had yield during week</td>
<td>high</td>
</tr>
<tr>
<td>is one of strategy</td>
<td>high</td>
</tr>
<tr>
<td>is one of restructuring</td>
<td>high</td>
</tr>
</tbody>
</table>

#### np

<table>
<thead>
<tr>
<th>Feature Values</th>
<th>Class Prediction Strength</th>
</tr>
</thead>
<tbody>
<tr>
<td>of KLM Royal Dutch</td>
<td>low</td>
</tr>
<tr>
<td>in ethics charges against</td>
<td>low</td>
</tr>
<tr>
<td>I drink to your share price could zoom</td>
<td>low</td>
</tr>
</tbody>
</table>

---

The general trend we observe in the results obtained with the editing experiments is that editing on the basis of typicality and class-prediction strength, whether low or high, is not beneficial, and is ultimately harmful to generalization accuracy. More specifically, we observe a trend that editing instances with high typicality or high class prediction strength, whether low or high, is not beneficial, and is ultimately harmful to generalization accuracy.
high cps is less harmful than editing instance types with low typicality or low class prediction strength. The results are summarized in Figure 2. The results show that in any case for our data sets, editing serves neither of its original goals. If the goal is a decrease of speed and memory requirements, editing criteria should allow editing of 50% or more without a serious decrease in generalization accuracy. Instead, we see disastrous effects on generalization accuracy at much lower editing rates, sometimes even at 1%. When the goal is improving generalization accuracy by removing noise, the focus of the editing experiments in this paper of course differs, as discussed in Section 4. The results show that in any case for our data sets, editing serves neither of its original goals. If the goal is a decrease of speed and memory requirements, editing criteria should allow editing of 50% or more without a serious decrease in generalization accuracy. Instead, we see disastrous effects on generalization accuracy at much lower editing rates, sometimes even at 1%.

Figure 2. Generalization accuracies (in terms of % of correctly classified test instances) of the four tasks with increasing percentages of edited instance tokens, according to the four tested editing criteria. To compute the statistical significance of the effect of editing, the output for each criterion was compared to the output of the unedited classifier. The resulting cross-tabulation of hits and misses was subjected to McNemar's test (Dietterich, 1998 in press). Differences with \( p < 0.05 \) are reported as significant.

A detailed look at the results per data set shows the following results. Editing
Another way to study the influence of exceptional instances on generalization accuracy is the adaptation of decision tree learning.

The editing results on the pos task (top right of Figure 2) indicate that editing on the basis of low typicality or low class prediction strength leads to significant decreases in generalization accuracy even with the smallest amount (1%) of edited instance types. Editing on the basis of high typicality and high cps can be performed up to 10% and 5% respectively without significant performance loss. For this data set, the drop in performance is radical only for low typicality.

Editing on the pp task (bottom left of Figure 2) results in significant decreases of generalization accuracy with respectively 5% and 10% of edited instance tokens of low typicality and low cps. Editing with high typicality and high cps can be performed up to 20% and 10% respectively without significant performance loss, but accuracies drop dramatically when 30% or more of high-typical or high-cps instance types are edited.

Finally, editing on the np data (bottom right of Figure 2) can be done without significant generalization accuracy loss with either the low or the high cps criterion, up to respectively 30% and 10%. Editing with low or high typicality, however, is harmful to generalization immediately from editing 1% of the instance tokens.

In sum, the experiments with editing on the basis of criteria estimating the exceptionality of instances show that forgetting of exceptional instances in memory-based learning while safeguarding generalization accuracy can only be performed to a very limited degree by (i) replacing instance tokens by instance types with frequency information (which is trivial and is done by default in ib1-ig) and (ii) removing small amounts of minority ambiguities with low (0.0) cps. Even the highest amounts of minority ambiguities with low (0.0) cps.

In the experiments with editing on the basis of criteria estimating the exceptionality of instances, the performance loss due to editing is significant in most cases.

Another way to study the influence of exceptional instances on generalization accuracy is to compare ib1-ig, introduced in Section 2, to inductive algorithms that abstract from exceptional instances by means of pruning or other devices.
5.1. Results

Ordered on a continuum representing how exceptional instances are handled, we perform additional experiments with $c=0$, with

<table>
<thead>
<tr>
<th>Task</th>
<th>$c=0$</th>
<th>$c=1/5$</th>
<th>$c=1/4$</th>
<th>$c=1/2$</th>
<th>$c=1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{gs}$</td>
<td>$93.4%$</td>
<td>$87.6%$</td>
<td>$80.0%$</td>
<td>$72.6%$</td>
<td>$67.6%$</td>
</tr>
<tr>
<td>$\text{pos}$</td>
<td>$93.7%$</td>
<td>$90.4%$</td>
<td>$90.4%$</td>
<td>$89.7%$</td>
<td>$90.7%$</td>
</tr>
<tr>
<td>$\text{pp}$</td>
<td>$86.9%$</td>
<td>$90.6%$</td>
<td>$90.6%$</td>
<td>$89.7%$</td>
<td>$90.7%$</td>
</tr>
<tr>
<td>$\text{np}$</td>
<td>$98.0%$</td>
<td>$98.0%$</td>
<td>$98.0%$</td>
<td>$98.0%$</td>
<td>$98.0%$</td>
</tr>
</tbody>
</table>

Table 5 displays the generalization accuracies, measured in percentages of correctly classified test instances, for $\text{ib-ig}$, $\text{igtree}$, and $\text{c=0}$ on the four tasks. We were unfortunately unable to finish the $\text{c=0}$ experiment on the $\text{np}$ data set for memory reasons (running on a SUN Sparc 5 with 160 Mb internal memory and 386 Mb swap space). The statistical significance of the differences between the algorithms is summarized in Table 6. We performed a one-tailed paired t-test between the results of the 10 CV runs.

As the results in these Tables show, $\text{ib-ig}$ has significantly better generalization accuracy than $\text{igtree}$ for all data sets. In two of the three data sets where the comparison is feasible, $\text{ib-ig}$ significantly outperforms $\text{c=0}$. For the $\text{pos}$ data set, $\text{c=0}$ outperforms $\text{ib-ig}$ with a small but statistically significant difference.

Table 5. Generalization accuracies (in terms of percentages of correctly classified test instances) on the $\text{gs}$, $\text{pos}$, $\text{pp}$, and $\text{np}$ tasks, by $\text{ib-ig}$, $\text{igtree}$, and $\text{c=0}$, with default settings ($c=2/5$, $m=2$).
Table 1. Significance of the differences between the generalization performances of ib1-ig, c5.0opt, c5.0def, and igtree, for the four tasks. A one-tailed paired t-test (df = 9) was performed, to see whether the generalization accuracy of the algorithm to the left is better than that of the algorithm to the right (indicated by a greater than \( > \) or a less than \(<\) sign), or the other way around (indicated by a less than \(<\) or a greater than \(>\) sign).

<table>
<thead>
<tr>
<th>Algorithm 1</th>
<th>Algorithm 2</th>
<th>GS</th>
<th>POS</th>
<th>PP</th>
<th>np</th>
</tr>
</thead>
<tbody>
<tr>
<td>( c ) = 0.0 and ( m ) = 1</td>
<td></td>
<td>75.0</td>
<td>77.5</td>
<td>80.0</td>
<td>82.5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>85.0</td>
<td>87.5</td>
<td>90.0</td>
<td>92.5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>95.0</td>
<td>97.5</td>
<td>100.0</td>
<td></td>
</tr>
</tbody>
</table>

Figure 3 displays the effect on generalization accuracy of varying the \( c \) parameter from 0 to 100 (left) and the \( m \) parameter from 1 to 10 (right). Performance of c5.0 with increasing \( c \) parameter (left) and increasing \( m \) parameter (right) for the gs, pos, and pp tasks. Since pruning is used, there is no difference in the number of nodes for the pos task and the other tasks. Although there is a slight increase in the number of nodes as \( c \) increases, the performance of c5.0 remains relatively constant. The direct effect of changing both parameters is shown in Figure 4; small values of \( c \) lead to smaller trees, as do large values of \( m \).
Figure 4. The effect of the number of nodes generated by c0.0 with increasing c parameter [left] and m parameter [right].

Table 7 compares c0.0 with default settings (c0.0def) to c0.0 with "lazy" parameter setting. The differences are significant in terms of generalization accuracy in terms of percentage of correctly classified test instances on the gs, pos, and pp tasks, but not for the pp data set.

<table>
<thead>
<tr>
<th>Task</th>
<th>Generalization Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>gs</td>
<td>93.4/13.1</td>
</tr>
<tr>
<td>pos</td>
<td>79.2/9.3</td>
</tr>
<tr>
<td>pp</td>
<td>80.9/14.9</td>
</tr>
</tbody>
</table>

These parameter tuning results indicate that decision-tree pruning is not beneficial to generalization accuracy, but neither is it generally harmful. Only on the gs task are strong decreases in generalization accuracy found with decreasing c, while small decreases in performance are witnessed with increasing m for the pos and pp tasks. Likewise, small decreases in generalization accuracy found with decreasing c are seen in generalization accuracy, but neither is a generally harmful. Only on the gs task.

Efficiency

In addition to generalization accuracy, which is the focus of our attention in this research, efficiency, measured in terms of training and testing speed and in terms of memory requirements, is also an important criterion to evaluate learning algorithms. For training, ib is fastest as it reduces to storing instances and in terms of memory requirements, is also an important criterion to evaluate learning algorithms. In this research, efficiency, measured in terms of training and testing speed and in terms of memory requirements, is also an important criterion to evaluate learning algorithms.
6. Why forgetting exceptions is harmful

In this Section, we have shown that when comparing the generalization accuracy of ib1-ig to that of decision tree methods, we see the same results as in our experiments on editing: different types of abstraction (some of them explicitly aimed at removing exceptional instances) do not succeed in generalizing to explain a better generalization accuracy than ib1-ig. However, for some data sets, a lower generalization accuracy than ib1-ig is observed. Therefore, for some data sets, we see the same results as in our experiment on the first partition of each of the data sets.

<table>
<thead>
<tr>
<th>Time (seconds)</th>
<th>Task</th>
<th>c5.0</th>
<th>igtree</th>
<th>ib1-ig</th>
</tr>
</thead>
<tbody>
<tr>
<td>train</td>
<td>test</td>
<td>total</td>
<td>train</td>
<td>test</td>
</tr>
<tr>
<td>gs</td>
<td>-</td>
<td>-</td>
<td>2406.79</td>
<td>-</td>
</tr>
<tr>
<td>pos</td>
<td>-</td>
<td>-</td>
<td>2724.46</td>
<td>-</td>
</tr>
<tr>
<td>pp</td>
<td>-</td>
<td>-</td>
<td>2219.51</td>
<td>-</td>
</tr>
<tr>
<td>np</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>2152.20</td>
</tr>
</tbody>
</table>

These two tables in seconds (elapsed wall clock time) for the first partition of each of the four data sets provide the details of the comparisons of each of the data sets.

In this section, we explain why forgetting exceptional instances, either by editing them from memory or by pruning them from decision trees is harmful to generalization accuracy. We explain this effect on the basis of the properties of this type of task and the properties of the learning algorithms.
algorithms used. Our approach of studying data set properties, to find an explanation for why one type of inductive algorithm rather than another is better suited for learning a type of task, is in the spirit of Aha (1992) and Michie, Spiegelhalter, and Taylor (1994).

Properties of language processing tasks

Language processing tasks are usually described as complex mappings between representations: from spelling to sound, from strings of words to parse trees, from parse trees to semantic formulas, etc. These mappings can be approximated by cascades of classification tasks (Ratnaparkhi, 1997; Daelemans, 1996; Cardie, 1996; Magerman, 1994) which makes them amenable to machine learning approaches.

One of the most salient characteristics of natural language processing mappings is that they are noisy and complex. Apart from some regularities, they contain many sub-regularities and pockets of exceptions. In other words, apart from a core of generalizable regularities, there is a relatively large periphery of irregularities (Daelemans, 1996). In rule-based NLP, this problem has to be solved using mechanisms such as rule ordering, subsumption, inheritance, or default reasoning (in linguistics this type of "priority to the most specific" mechanism is called the elsewhere condition). In the feature-vector-based approximations of these complex language processing mappings, this property is reflected in the high degree of disjunctivity of the instance space: classes exhibit a high degree of polymorphism. Another issue we study in this Section is the usefulness of exceptional as opposed to more regular instances in classification.

Degree of polymorphism

Several quantitative measures can be used to show the degree of polymorphism: the number of clusters (i.e., groups of nearest-neighbour instances belonging to the same class), the number of disjunct clusters per class (i.e., the numbers of separate clusters per class), or the numbers of prototypes per class (Aha, 1992). We approach these issues by looking at the average number of nearest neighbors found surrounding instances; the x-axis of Figure 5 denotes the numbers of friendly neighbors (a.k.a. closest instances; i.e., instances of the same class) found, whereas the y-axis denotes the cumulative percentage of occurrences of friendly-neighbour clusters of particular sizes.
cumulative % instances

Figures 5/6. The cumulative percentage of friendly/-neighbor clusters of sizes 0 to 4, as found in the data sets of the grammatical, paragraph, and noun tasks.

In sum, we find indications for a high disjunctivity or polymorphism of the language data sets investigated in this study. Other studies in which machine learning algorithms are applied to language data, and in which special attention is paid to learning exceptions, mention similar indications (e.g., Money and Call 1995; Van den Bosch et al. 1995). However, the question whether these phenomena are related to the high disjunctivity of the language data, and in which special attention is paid to learning exceptions, remains an open one, and will be a focal point in future research.
strength cannot be removed from the training data. For this purpose, we have looked at the instances that are actually used in the memory-based classification process to classify the test instances. We call the nearest neighbors that were used to classify test instances the support set. The distribution of typicality and CPS over the support set can be seen in Figure 6. The support set can be divided into support for correct decisions (Right) and errors (Wrong). The average number of neighbors for correct decisions is approximately the same as for errors. The figures clearly show that even instances with respect to low typicality (below 1.0) or low CPS (below 0.5) are more often used to support correct decisions than errors. However, this does not mean that the support set can be seen in Figure 6. The distribution of correct and incorrect instances on the support set is approximately equal. To classify test instances, we call the nearest neighbors that were used to classify these instances cannot be removed from these neighbors. For this purpose, we have

\[ (u, s, \nabla) > (s', s, \nabla) \iff u > s \]
Figure 6. Histograms per typicality (left) and class prediction strength (right) of the neighbors present in support sets for each of the four tasks. For each range (indicated at the x-axes), the number of instances leading to a correct classification (Right) and to a misclassification (Wrong) is displayed as a bar.
from all training instances available for extrapolation in those cases where more specific information is not available. Decision trees can also be described as back-off estimators of the class probability conditioned on the combination of the features' values. However, here some schemata are not available for extrapolation. Even in a decision tree without any pruning, such abstraction takes place. Once a test instance matches any schemata of the visible node, the set of schemata from which it can receive a classification is restricted to those that match all features with a higher weight. The depth of the visible node is the distance from all schemata in the tree. As the condition of features is constant throughout the tree, a more direct illustration of this matter can be given for the limited accessibility of specific support sets and thus of lower accuracy.

Figures 7 and 8 show why this limitation of schemata can be harmful. In this figure, percentage correct for our data sets plotted as a function of distance between the test instance and its nearest neighbor. The distances are normalized between zero and one, and discretized into a maximum of ten evenly spaced intervals to make a comparison across data sets possible. Figure 7 shows why this limitation of schemata can be harmful. In this figure, percentage correct for our data sets plotted as a function of specificity. The decrease of the accuracy seen in the graph clearly confirms the intuition that an extrapolation from a more specific support set is more likely to be correct. Pruning from a more specific support set is more likely to be correct.

A more direct illustration of this matter can be given for the limited accessibility of schemata in the tree. As the ordering of features is constant throughout the tree, the schemata that are accessible at a given node in the tree are limited to those schemata that match all features with a higher weight. The depth of the visible node is the distance from all schemata in the tree. As the condition of features is constant throughout the tree, a more direct illustration of this matter can be given for the limited accessibility of specific support sets and thus of lower accuracy.
The average distance at which classification takes place for the four tasks with $\gamma = 2$, $\gamma = 3$, and $\gamma = 5$.

We performed experiments with $\gamma = 2$. The results with $\gamma = 3$ and $\gamma = 5$ have been split out in four conditions: FF, FT, TF, and TT. The first letter refers to $\text{ib}^{-1}$ giving a False or True answer, the second refers in the same manner to the output of $\text{igtree}$. The third column gives the number of instances for that condition. The $\text{igtree}$ distances have been computed from an unpruned tree.

Average IG Overlap Distance (number of instances) between the test pattern and the branch of the tree, using the ig weights. To make the comparison fair, we have used an unpruned $\text{igtree}$. Table 9 shows the average distances at which classifications were made for the four tasks at hand.

Table 9: Average IG Overlap Distance (number of instances)

<table>
<thead>
<tr>
<th>Task</th>
<th>FF</th>
<th>FT</th>
<th>TF</th>
<th>TT</th>
</tr>
</thead>
<tbody>
<tr>
<td>ib^{-1}</td>
<td>0.90 0.90 1.00 1.00</td>
<td>0.90 0.90 1.00 1.00</td>
<td>0.90 0.90 1.00 1.00</td>
<td>0.90 0.90 1.00 1.00</td>
</tr>
<tr>
<td>igtree</td>
<td>0.89 0.89 0.93 0.93</td>
<td>0.89 0.89 0.93 0.93</td>
<td>0.89 0.89 0.93 0.93</td>
<td>0.89 0.89 0.93 0.93</td>
</tr>
</tbody>
</table>

Because the schema was not accessible, because the exchange was not accessible, because the exchange was not accessible, because the exchange was not accessible.

Increasing $k$ as an aside, we note that we have reported solely on experiments with $k = 1$. Although it is not directly related to "forgetting," taking a larger value of $k$ can also be considered as a type of abstraction, because it is estimated from a somewhat smoothed region of the instance space. Only on the basis of the results described so far, we cannot claim that $k = 1$ is the optimal setting for our experiments. The results discussed above suggest that the average $k'$ actually surrounding an instance is larger than 1, although many instances have only one or no friendly neighbors, especially in the case of the $\text{gs}$ task.

The latter suggests that a considerable amount of vagueness is found in instances that are highly similar; matching with $k > 1$ may be considered as a type of abstraction, becoming a property while it can also be considered as a type of abstraction, because it is estimated from a somewhat smoothed region of the instance space. Only on the basis of the results described so far, we cannot claim that $k = 1$ is the optimal setting for our experiments. The results discussed above suggest that the average $k'$ actually surrounding an instance is larger than 1, although many instances have only one or no friendly neighbors, especially in the case of the $\text{gs}$ task.

The latter suggests that a considerable amount of vagueness is found in instances that are highly similar; matching with $k > 1$ may be considered as a type of abstraction, becoming a property while it can also be considered as a type of abstraction, because it is estimated from a somewhat smoothed region of the instance space. Only on the basis of the results described so far, we cannot claim that $k = 1$ is the optimal setting for our experiments. The results discussed above suggest that the average $k'$ actually surrounding an instance is larger than 1, although many instances have only one or no friendly neighbors, especially in the case of the $\text{gs}$ task.
Table 1. Generalization accuracies (in terms of percentages of correctly classified test instances) on the gs, pos, pp, and np tasks, by ib(-ig) with k = 1, 2, 3, and 5.

<table>
<thead>
<tr>
<th>Task</th>
<th>k = 1</th>
<th>k = 2</th>
<th>k = 3</th>
<th>k = 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>gs</td>
<td>93.1%</td>
<td>94.2%</td>
<td>94.8%</td>
<td>95.1%</td>
</tr>
<tr>
<td>pos</td>
<td>97.0%</td>
<td>97.5%</td>
<td>98.0%</td>
<td>98.2%</td>
</tr>
<tr>
<td>pp</td>
<td>85.0%</td>
<td>86.0%</td>
<td>87.5%</td>
<td>88.5%</td>
</tr>
<tr>
<td>np</td>
<td>80.5%</td>
<td>81.5%</td>
<td>82.5%</td>
<td>83.5%</td>
</tr>
</tbody>
</table>

In this Section, we have tried to interpret our empirical results in terms of properties of the data and of the learning algorithms used. A salient characteristic of our language learning tasks, shown most clearly in the gs data set but also present in the other data sets, is the presence of a high degree of class polymorphism (high disjunctivity). In many cases, these small disjuncts constitute productive (pocket of) exceptions which are useful in producing accurate extrapolations to new data.

ib(-ig), through its implicit parallelism and its feature relevance weighting, is better suited than memory-based learning to make the most specific relevant patterns in memory available for extrapolation from.

Related research Daelemans (1995) provides an overview of memory-based learning work on phonological and morphological tasks (grapheme-to-phoneme conversion, syllabification, hyphenation, morphological synthesis, word stress assignment) at Tilburg University and the University of Antwerp in the early nineties. The present paper directly builds on the results obtained in that research. More recently, the approach has also been applied to part-of-speech tagging (morphosyntactic disambiguation), field-name tagging, and the resolution of disambiguation problems (Daelemans and Van den Bosch, 1996; Van den Bosch, Daelemans, and Weijters, 1996). Combinations of separate modules have been shown to produce improved results. For example, the use of a model-based approach for the recognition of named entities (Daelemans, 1999), provides an overview of memory-based learning work on phone-
In the recent literature on statistical language learning, which currently still largely adheres to the hypothesis that what is exceptional (improbable) is unimportant, similar results as those discussed here for machine learning have been reported. In Bod (1995), a data-oriented approach to parsing is described in which a treebank is used as a `memory' and in which the parse of a new sentence is computed by reconstruction from subtrees present in the treebank. It is shown that removing all hapaxes (unique subtrees) from memory degrades generalization performance from 96% to 72%. Bod notes that this seems to contradict the fact that probabilities based on sparse data are not reliable. (Bod, 1995, p. 68). In the same vein, Collins and Brookes (1995) show that when applying the back-off estimation technique (Katz, 1987) to learning prepositional-phrase attachment, removing all events with a frequency of less than 5 degrades generalization performance from 84.4% to 76.2%. In Dagan, Lee, and Pereira (1997), finally, a similarity-based estimation method is compared to back-off and maximum-likelihood estimation on a pseudo-word sense disambiguation task. Again, a positive effect of events with frequency 1 in the training set on generalization accuracy is noted.

In the context of statistical language learning, it is also relevant to note that as far as comparable results are available, statistical techniques, which also abstract from exceptional events, never obtain a higher generalization accuracy than ib1-ig(Daelemans, 1995; Zavrel and Daelemans, 1997; Zavrel, Daelemans, and Veenstra, 1997). Reliable comparisons (in the sense of methods being compared on the same train and test data) with the empirical results reported here cannot be made, however.

In the machine learning literature, the problem of small disjuncts in concept learning has been studied before by Quinlan (1991), who proposed more accurate error estimation methods for small disjuncts, and by Holte, Ackerman, and Porter (1989). The latter define a small disjunct as one that has small coverage (i.e., a small number of training items are correctly classified by it). This definition differs from ours, in which small disjuncts are those that have few neighbors with the same category. Nevertheless, similar phenomena are noted: sometimes small disjuncts constitute a significant portion of an induced definition, and it is hard to distinguish productive small disjuncts from noise (see also Daniuk and Provost, 1993). A maximum-simplicity bias for small disjuncts is proposed to make small disjuncts less error-prone. Memory-based learning is of course a good way of implementing this remedy (as noted, e.g., in Aha, 1992). This prompted Ting (1994b) to propose a composite learner with an instance-based component for small disjuncts and a concept learning component with an instance-knowledge component for small disjuncts. A further development along these lines is proposed in Bod (1996b), where a schematic of the approach is described in which the treebank is used as a `memory' and in which the parse of a new sentence is computed by reconstruction from subtrees present in the treebank. The figure shows how the parsing process is decomposed into two stages, where the first stage performs a treebank-based reconstruction from the treebank, and the second stage performs an instance-based reconstruction. The figure illustrates how the two stages interact to produce the final parse. This hybrid learner improves upon the baseline for several definitions of `small disjunct' for most of the data sets studied. Similar results have recently been reported by Domingos (1996), where a schematic of the approach is described in which the treebank is used as a `memory' and in which the parse of a new sentence is computed by reconstruction from subtrees present in the treebank. The figure shows how the parsing process is decomposed into two stages, where the first stage performs a treebank-based reconstruction from the treebank, and the second stage performs an instance-based reconstruction. The figure illustrates how the two stages interact to produce the final parse.
specific (as in the decision tree methods used in this paper), make it a useful approach for our purposes class as well.

8. Conclusion and Future Research
opportunism can be downplayed from hybridization.

The Table, which incorporates more information and additional weighting in the first column, may not be meaningful when the table is not expanded to its full length. In the second column, we see that the data set is more detailed, on hybrid and people's face. The column remains much the same, and people's face is fully expanded.

The last column is made up of the large amount of information which lends into hybridization.

The top section is made up of the large amount of information which lends into hybridization without the

References

We would then like to thank the hybrid for pointing out this potential issue.

A hybrid between such domains is not only possible, but necessary.

In our full POS tagger we have a separate classifier for chunks of words which generate

Acknowledgments

This research was done in the context of the “Induction of Linguistic Knowledge”.

Notes

/1/. The LOB corpus is available from icame, the International Computer Archive of Modern and Medieval English; consult http://www.hdl.uib.no/icame.html for more information.

/2/. In our full POS tagger we have a separate classifier for chunks of words which generate


/4/. TiMBL, which incorporates ib1/ig and igtree and additional weighting metrics and search optimizations, can be downloaded from http://ilk.kub.nl/.

We plan to expand on the encouraging results on other data sets using a hybrid of ib1/ig and ib2/ig that leaves schemes accessible when there is no clear feature-relevance distinction. We plan to expand on the encouraging results on other data sets using a hybrid of ib1/ig and ib2/ig that leaves schemes accessible when there is no clear feature-relevance distinction.