# Memory-Based Part of Speech Tagging *

Walter Daelemans
walter.daelemans@kub.nl

Jakub Zavrel
zavrel@kub.nl

Peter Berck
peter.berck@uia.ua.ac.be

Steven Gillis
steven.gillis@uia.ua.ac.be

**Abstract**

We introduce a memory-based approach to part of speech tagging. Memory-based learning is a form of supervised learning based on similarity-based reasoning. The part of speech tag of a word in a particular context is extrapolated from the most similar cases held in memory. Supervised learning approaches are useful when a tagged corpus is available as an example of the desired output of the tagger. Based on such a corpus, the tagger-generator automatically builds a tagger which is able to tag new text the same way, diminishing development time for the construction of a tagger considerably. Memory-based tagging shares this advantage with other statistical or machine learning approaches. Additional advantages specific to a memory-based approach include (i) the relatively small tagged corpus size sufficient for training, (ii) incremental learning, (iii) explanation capabilities, (iv) flexible integration of information in case representations, (v) its non-parametric nature, (vi) reasonably good results on unknown words without morphological analysis, and (vii) fast learning and tagging. In this paper we show that a large-scale application of the memory-based approach is feasible: we obtain a tagging accuracy that is on a par with that of known statistical approaches, and with attractive space and time complexity properties when using IGTREE, a tree-based formalism for indexing and searching huge case bases. The use of IGTREE has as additional advantage that optimal context size for disambiguation is dynamically computed.

## 1 Introduction

Part of Speech (POS) tagging is a process in which syntactic categories are assigned to wordsIt can be seen as a mapping from sentences to strings of

*This is a revised, expanded version of Daelemans et al. (1996). Research of the first author was done while he was a visiting scholar at NIAS (Netherlands Institute for Advanced Studies) in Wassenaar. Thanks to Antal van den Bosch, Ton Weijters, and Gert Durieux for discussions about tagging, IGTREE, and machine learning of natural language. Thanks to the CLIN referees for useful comments.

tags.

| Input | Output |
|---|---|
| John will join the board | np md vb dt nn |

Automatic tagging is useful for a number of applications: as a preprocessing stage to parsing, in information retrieval, in text to speech systems, in corpus linguistics, etc. The two factors determining the syntactic category of a word are its lexical probability (e.g. without context, *man* is more probably a noun than a verb), and its contextual probability (e.g. after a pronoun, *man* is more probably a verb than a noun, as in *they man the boats*). Several approaches have been proposed to construct automatic taggers. Most work on statistical methods has used $n$-gram models or Hidden Markov Model-based taggers, e.g. (Church, 1988; DeRose, 1988; Cutting et al., 1992; Merialdo, 1994). In these approaches, a tag sequence is chosen for a sentence that maximizes the product of lexical and contextual probabilities as estimated from a tagged corpus.

In rule-based approaches, words are assigned a tag based on a set of rules and a lexicon. These rules can either be hand-crafted (Garside, Leech, and Sampson, 1987; Klein and Simmons, 1963; Greene and Rubin, 1971), or learned, as in (Hindle, 1989) or the transformation-based error-driven approach of Brill (1992).

In the memory-based approach, a set of cases is kept in memory. Each case consists of a word (or a lexical representation for the word) with preceding and following context, and the corresponding category for that word in that context. A new sentence is tagged by selecting for each word in the sentence and its context the most similar case(s) in memory, and extrapolating the category of the word from these 'nearest neighbors'. A memory-based approach has features of both learning rule-based taggers (each case can be regarded as a very specific rule, the similarity based reasoning as a form of conflict resolution and rule selection mechanism) and of stochastic taggers: it is fundamentally a form of $k$-nearest neighbors ($k$-nn) modeling, a well-known non-parametric statistical pattern recognition technique. The approach in its basic form is computationally expensive, however; each new word in context that has to be tagged, has to be compared to each pattern kept in memory. In this paper we show that a heuristic case base compression formalism (Daelemans et al., 1996), makes the memory-based approach computationally attractive.

## 2 Memory-Based Learning

Memory-based Learning is a form of *supervised, inductive* learning from *examples*. Examples are represented as a vector of feature values with an associated category label. During training, a set of examples (the *training set*) is presented in an incremental fashion to the classifier, and added to

memory. During testing, a set of previously unseen feature-value patterns (the *test set*) is presented to the system. For each test pattern, its distance to all examples in memory is computed, and the category of the least distant instance(s) is used as the predicted category for the test pattern. The approach is based on the assumption that reasoning is based on direct reuse of stored experiences rather than on the application of knowledge (such as rules or decision trees) abstracted from experience.

In AI, the concept has appeared in several disciplines (from computer vision to robotics), using terminology such as similarity-based, example-based, memory-based, exemplar-based, case-based, analogical, lazy, nearest-neighbour, and instance-based (Stanfill and Waltz, 1986; Kolodner, 1993; Aha, Kibler, and Albert, 1991; Salzberg, 1990). Ideas about this type of analogical reasoning can be found also in non-mainstream linguistics and pyscholinguistics (Skousen, 1989; Derwing and Skousen, 1989; Chandler, 1992; Scha, 1992). In computational linguistics (apart from incidental computational work of the linguists referred to earlier), the general approach has only recently gained some popularity: e.g. Cardie (1994): syntactic and semantic disambiguation; Daelemans (1995): an overview of work in the early nineties on memory-based computational phonology and morphology; Jones (1996): an overview of example-based machine translation research; Federici and Pirrelli (forthcoming).

## 2.1 Similarity Metric

Performance of a memory-based system (accuracy on the test set) crucially depends on the distance metric (or similarity metric) used. The most straightforward distance metric would be the one in Equation 1, where $X$ and $Y$ are the patterns to be compared, and $\delta(x_i, y_i)$ is the distance between the values of the $i$-th feature in a pattern with $n$ features.

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

Distance between two values is measured using Equation 2, an overlap metric, for symbolic features (we will have no numeric features in the tagging application).

$$\delta(x_i, y_i) = \begin{cases} 0 & \text{if } x_i = y_i \\ 1 & \text{otherwise} \end{cases} \tag{2}$$

We will refer to this approach as IB1 (Aha, Kibler, and Albert, 1991). We extended the algorithm described there in the following way: in case a pattern is associated with more than one category in the training set (i.e. the pattern is ambiguous), the distribution of patterns over the different categories is kept, and the most frequently occurring category is selected when the ambiguous pattern is used to extrapolate from.

## 2.2 Feature Relevance Weighting

In this distance metric, all features describing an example are interpreted as being equally important in solving the classification problem, but this is not necessarily the case. In tagging, the focus word to be assigned a category is obviously more relevant than any of the words in its context. We therefore weigh each feature with its *information gain*; a number expressing the average amount of reduction of training set information entropy when knowing the value of the feature (Daelemans and van den Bosch, 1992; Quinlan, 1993; Hunt, Marin, and Stone, 1966) (Equation 6). We will call this algorithm IB-IG. Many other methods to weigh the relative importance of features have been designed, both in statistical pattern recognition and in machine learning (see (Wettschereck, Aha, and Mohri, 1996) for an overview).

The main idea of *information gain weighting* is to interpret the training set as an information source capable of generating a number of messages (the different category labels) with a certain probability. The information entropy of such an information source can be compared in turn for each feature to the average information entropy of the information source when the value of that feature is known. Database information entropy is equal to the number of bits of information needed to know the category given a pattern. It is computed by Equation 3, where $p_i$ (the probability of category $i$) is estimated by its relative frequency in the training set.

$$H(D) = -\sum_i p_i log_2 p_i \tag{3}$$

For each feature, it is now computed what the information gain is of knowing its value. To do this, we compute the average information entropy for this feature and subtract it from the information entropy of the database. To compute the average information entropy for a feature (Equation 4), we take the average information entropy of the database restricted to each possible value for the feature. The expression $D_{[f=v]}$ refers to those patterns in the database that have value $v$ for feature $f$, $V$ is the set of possible values for feature $f$. Finally, $|D|$ is the number of patterns in a (sub)database.

$$H(D_{[f]}) = \sum_{v_i \in V} H(D_{[f=v_i]}) \frac{|D_{[f=v_i]}|}{|D|} \tag{4}$$

A well-known disadvantageous property of information gain is that it tends to favour features with many values. This bias can be rectified, as suggested in (Quinlan, 1993), by normalizing the information gain of a feature by dividing it by the number of bits required to determine the feature (which depends on its number of values).

$$\textit{split-info}(f) = -\sum_{v_i \in V} \frac{|D_{[f=v_i]}|}{|D|} log_2 \frac{|D_{[f=v_i]}|}{|D|} \tag{5}$$

Information gain is then obtained by Equation 6, and scaled to be used as a weight for the feature during distance computation.

$$G(f) = H(D) - \frac{H(D_{[f]})}{\textit{split-info}(f)} \tag{6}$$

Finally, the distance metric in Equation 1 is modified to take into account the information gain weight associated with each feature.

$$\Delta(X, Y) = \sum_{i=1}^{n} G(f_i)\delta(x_i, y_i) \tag{7}$$

# 3 Memory-Based Language Processing

Memory-Based Learning is a *classification* paradigm. Given a description of a 'case' in terms of feature-value pairs, a category label is produced. In tagging, a case description is a focus word to be disambiguated and its context, and the category label is a syntactic tag, drawn from a finite tag set known beforehand.

A method often necessary to arrive at the context information needed in a classification approach is the *windowing* approach (as used in (Sejnowski and Rosenberg, 1987) for grapheme to phoneme conversion with neural networks), in which an imaginary window is moved one item at a time over an input string where one item in the window (usually the middle item or the last item) acts as a focus item, and the rest as the context. We will apply this approach for tagging as well.

The mapping from sentences to a series of tags will then be approximated by a function from a focus word and its fixed-width context to the disambiguated tag belonging to the focus word. By doing this, the mapping becomes a *classification* task amenable to Memory-Based Learning (see Table 1).

| Input | | | | | Output |
|---|---|---|---|---|---|
| Left Context | | Focus | Right Context | | Target |
| = | = | John | will | join | np |
| = | John | will | join | the | md |
| John | will | join | the | board | vb |
| will | join | the | board | = | dt |
| join | the | board | = | = | nn |

Table 1: Tagging as a classification task.

The tree-based indexing of the case-base, which will be described in the next section, will restrict an initially large context to only the relevant features.

# 4 IGTrees

Memory-based learning is an expensive algorithm: of each test item, all feature values must be compared to the corresponding feature values of all training items. Without optimisation, it has an asymptotic retrieval complexity of $O(NF)$, where $N$ is the number of items in memory, and $F$ the number of features. The same asymptotic complexity is of course found for memory storage in this approach. We use IGTrees (Daelemans et al., 1996) to compress the memory. IGTree is a heuristic approximation of the IB-IG algorithm.

Hardware solutions to the complexity problem have also been proposed: massively parallel computing (Stanfill and Waltz, 1986) or even wafer-scale integration (Kitano, 1993). For numeric features kd-trees have been proposed (Friedman, Bentley, and Finkel, 1977) as a solution on single-processor machines. The advantages of this approach do not generalize easily to symbolic features, however.

## 4.1 The IGTree Algorithms

IGTree combines two algorithms: one for compressing a case base into a trees, and one for retrieving classification information from these trees. During the construction of IGTree decision trees, cases are stored as paths of connected nodes. All nodes contain a test (based on one of the features) and a class label (representing the default class at that node). Nodes are connected via arcs denoting the outcomes for the test (feature values). A feature relevance ordering technique (in this case information gain, see Section 2.1) is used to determine the order in which features are used as tests in the tree. This order is fixed in advance, so the maximal depth of the tree is always equal to the number of features, and at the same level of the tree, all nodes have the same test (they are an instance of *oblivious decision trees*, cf. (Langley and Sage, 1994). The reasoning behind this reorganisation (which is in fact a compression) is that when the computation of feature relevance points to one feature clearly being the most important in classification, search can be restricted to matching a test case to those stored cases that have the same feature value at that feature. Besides restricting search to those memory cases that match only on this feature, the case memory can be optimised by further restricting search to the second most important feature, followed by the third most important feature, etc. A considerable compression is obtained as similar cases share partial paths.

Instead of converting the case base to a tree in which all cases are fully represented as paths, storing all feature values, we compress the tree even more by restricting the paths to those input feature values that disambiguate the classification from all other cases in the training material. The idea is that it is not necessary to fully store a case as a path when only a few feature

values of the case make its classification unique. This implies that feature values that do not contribute to the disambiguation of the case classification (i.e. the values of the features with lower feature relevance values than the the lowest value of the disambiguating features) are *not* stored in the tree. In our tagging application, this means that only context feature values that actually contribute to disambiguation are used in the construction of the tree.

Leaf nodes contain the unique class label corresponding to a path in the tree. Non-terminal nodes contain information about the *most probable* or *default* classification given the path thus far, according to the bookkeeping information on class occurrences maintained by the tree construction algorithm. This extra information is essential when using the tree for classification. Finding the classification of a new case involves traversing the tree (i.e. matching all feature values of the test case with arcs in the order of the overall feature information gain), and either retrieving a classification when a leaf is reached, or using the default classification on the last matching non-terminal node if a feature-value match fails.

A final compression is obtained by pruning the derived tree. All leaf-node daughters of a mother node that have the same class as that node are removed from the tree, as their class information does not contradict the default class information already present at the mother node. Again, this compression does not affect IGTREE's generalisation performance.

The recursive algorithms for tree construction (except the final pruning) and retrieval are given in Figures 1 and 2. For a detailed discussion, see (Daelemans et al., 1996).

## 4.2 IGTree Complexity

The asymptotic complexity of IGTREE (i.e. in the worst case) is extremely favorable. Complexity of searching a query pattern in the tree is proportional to $F*log(V)$, where $F$ is the number of features (equal to the maximal depth of the tree), and $V$ is the average number of values per feature (i.e. the average branching factor in the tree). In IB1, search complexity is $O(N*F)$ (with $N$ the number of stored cases). Retrieval by search in the tree is independent from the number of training cases, and therefore especially useful for large case bases. Storage requirements are proportional to $N$ (compare $O(N*F)$ for IB1). Finally, the cost of building the tree on the basis of a set of cases is proportional to $N*log(V)*F$ in the worst case (compare $O(N)$ for training in IB1).

In practice, for our part-of-speech tagging experiments, IGTREE retrieval is 100 to 200 times faster than normal memory-based retrieval, and uses over 95% less memory.

Procedure **BUILD-IG-TREE**:

Input:

- A training set $T$ of cases with their classes (start value: a full case base),
- an information-gain-ordered list of features (tests) $F_i...F_n$ (start value: $F_1...F_n$).

Output: A (sub)tree.

1. If $T$ is unambiguous (all cases in $T$ have the same class $c$), create a leaf node with class label $c$.
2. Else if $i = (n + 1)$, create a leaf node with as label the class occurring most frequently in $T$.
3. Otherwise, until $i = n$ (the number of features)
   - Select the first feature (test) $F_i$ in $F_i...F_n$, and construct a new node $N$ for feature $F_i$, and as default class $c$ (the class occurring most frequently in $T$).
   - Partition $T$ into subsets $T_1...T_m$ according to the values $v_1...v_m$ which occur for $F_i$ in $T$ (cases with the same value for this feature in the same subset).
   - For each $j\epsilon\{1, ..., m\}$: BUILD-IG-TREE $(T_j, F_{i+1}...F_n)$, connect the root of this subtree to $N$ and label the arc with $v_j$.

Figure 1: Algorithm for building IGTrees ('BUILD-IG-TREE').

## 5 Architecture of the Tagger

The architecture takes the form of a *tagger generator*: given a corpus tagged with the desired tag set, a POS tagger is generated which maps the words of new text to tags in this tag set according to the same systematicity. The construction of a POS tagger for a specific corpus is achieved in the following way. Given an annotated corpus, three datastructures are automatically extracted: a *lexicon*, a case base for *known words* (words occurring in the lexicon), and a case base for *unknown words*. Case Bases are indexed using IGTree. During tagging, each word in the text to be tagged is looked up in the lexicon. If it is found, its lexical representation is retrieved and its context is determined, and the resulting pattern is looked up in the known words case base. When a word is not found in the lexicon, its lexical representation is computed on the basis of its form, its context is determined, and the resulting pattern is looked up in the unknown words case base. In

Procedure **SEARCH-IG-TREE**:

Input:

- The root node $N$ of a subtree (start value: top node of a complete IGTree),
- an unlabeled case $I$ with information-gain-ordered feature values $f_i ... f_n$ (start value: $f_1 ... f_n$).

Output: A class label.

1. If $N$ is a leaf node, output default class $c$ associated with this node.
2. Otherwise, if test $F_i$ of the current node does not originate an arc labeled with $f_i$, output default class $c$ associated with $N$.
3. Otherwise,
   - new node $M$ is the end node of the arc originating from $N$ with as label $f_i$.
   - SEARCH-IG-TREE ($M, f_{i+1} ... f_n$)

Figure 2: Algorithm for searching IGTREEs ('SEARCH-IG-TREE').

each case, output is a best guess of the category for the word in its current context. In the remainder of this section, we will describe each step in more detail. We start from a *training set* of tagged sentences $T$.

## 5.1 Lexicon Construction

A lexicon is extracted from $T$ by computing for each word in $T$ the number of times it occurs with each category. When using e.g. the first 2 million words of the Wall Street Journal corpus[1] as $T$, the word *once* would get the lexical definition *RB: 330; IN: 77*, i.e. *once* was tagged 330 times as an adverb, and 77 times as a preposition/subordinating conjunction.[2]

Using these lexical definitions, a new, possibly ambiguous, tag is produced for each word type. E.g. *once* would get a new tag, representing the category of words which can be both adverbs and prepositions/conjunctions (RB-IN). Frequency order is taken into account in this process: if there would be words which, like *once*, can be RB or IN, but more frequently IN

[1] ACL Data Collection Initiative CD-ROM 1, September 1991.

[2] We disregarded a category associated with a word when less than 10% of the word tokens were tagged with that category. This way, noise in the training material is filtered out. The value for this parameter will have to be adapted for other training sets, and was chosen here to maximise generalization accuracy (accuracy on tagging unseen text).

than RB (e.g. the word *below*), then a different tag (IN-RB) is assigned to these words. The original tag set, consisting of 44 morphosyntactic tags, was expanded this way to 419 (possibly ambiguous) tags. In the WSJ example, the resulting lexicon contains 57962 word types, 7464 (13%) of which are ambiguous. On the same training set, 76% of word *tokens* are ambiguous.

When tagging a new sentence, words are looked up in the lexicon. Depending on whether or not they can be found there, a case representation is constructed for them, and they are retrieved from either the known words case base or the unknown words case base.

## 5.2 Known Words

A case consists of information about a focus word to be tagged, its left and right context, and an associated category (tag) valid for the focus word in that context.

There are several types of information which can be stored in the case base for each word, ranging from the words themselves to intricate lexical representations. In the preliminary experiments described in this paper, we limited this information to the possibly ambiguous tags of words (retrieved from the lexicon) for the focus word and its context to the right, and the disambiguated tags of words for the left context (as the result of earlier tagging decisions). Table 2 is a sample of the case base for the first sentence of the corpus (*Pierre Vinken, 61 years old, will join the board as a nonexecutive director nov. 29*) when using this case representation. The final column shows the target category; the disambiguated tag for the focus word. We will refer to this case representation as `ddfat` (`d` for disambiguated, `f` for focus, `a` for ambiguous, and `t` for target). The information gain values are given as well.

A search among a selection of different context sizes suggested `ddfat` as a suitable case representation for tagging known words. An interesting property of memory-based learning is that case representations can be easily extended with different sources of information if available (e.g. feedback from a parser in which the tagger operates, semantic types, the words themselves, lexical representations of words obtained from a different source than the corpus, etc.). The information gain feature relevance ordering technique achieves a delicate relevance weighting of different information sources when they are fused in a single case representation. The window size used by the algorithm will also dynamically change depending on the information present in the context for the disambiguation of a particular focus symbol (see (Schütze and Singer, 1994; Pereira, Singer, and Tishby, 1995) for similar approaches).

| word | case representation | | | | target |
|---|---|---|---|---|---|
| | d | d | f | a | |
| IG | .06 | .22 | .82 | .23 | |
| Pierre | = | = | np | np | np |
| Vinken | = | np | np | , | np |
| , | np | np | , | cd | , |
| 61 | np | , | cd | nns | cd |
| years | , | cd | nns | jj-np | nns |
| old | cd | nns | jj-np | , | jj |
| , | nns | jj | , | md | , |
| will | jj | , | md | vb | md |
| join | , | md | vb | dt | vb |
| the | md | vb | dt | nn-np | dt |
| board | vb | dt | nn-np | in-rb | nn |
| as | dt | nn | in-rb | dt | in |
| a | nn | in | dt | jj | dt |
| nonexecutive | in | dt | jj | nn-np | jj |
| director | dt | jj | nn-np | np | nn |
| nov. | jj | nn | np | cd | np |
| 29 | nn | np | cd | . | cd |
| . | np | cd | . | = | . |

Table 2: Case representation and information gain pattern for known words.

### 5.3 Unknown Words

If a word is not present in the lexicon, its ambiguous category cannot be retrieved. In that case, a category can be guessed only on the basis of the *form* or the *context* of the word. Again, we take advantage of the data fusion capabilities of a memory-based approach by combining these two sources of information in the case representation, and having the information gain feature relevance weighting technique figure out their relative relevance (see (Schmid, 1994; Samuelsson, 1994) for similar solutions).

In most taggers, some form of morphological analysis is performed on unknown words, in an attempt to relate the unknown word to a known combination of known morphemes, thereby allowing its association with one or more possible categories. After determining this ambiguous category, the word is disambiguated using context knowledge, the same way as known words. Morphological analysis presupposes the availability of highly language-specific resources such as a morpheme lexicon, spelling rules, morphological rules, and heuristics to prioritise possible analyses of a word according to their plausibility. This is a serious knowledge engineering bottleneck when the goal is to develop a language and annotation-independent tagger generator.

In our memory-based approach, we provide morphological information (especially about suffixes) indirectly to the tagger by encoding the three last letters of the word as separate features in the case representation. The first letter is encoded as well because it contains information about prefix and capitalization of the word. Context information is added to the case representation in a similar way as with known words. It turned out that in combination with the 'morphological' features, a context of one disambiguated tag of the word to the left of the unknown word and one ambiguous category of the word to the right, gives good results. We will call this case representation `pdassst`:[3] three suffix letters (`s`), one prefix letter (`p`), one left disambiguated context words (`d`), and one ambiguous right context word (`a`). As the chance of an unknown word being a function word is small, and cases representing function words may interfere with correct classification of open-class words, only open-class words are used during construction of the unknown words case base.

Table 3 shows part of the case base for unknown words.

| word | case representation | | | | | | target |
|---|---|---|---|---|---|---|---|
| | p | d | a | s | s | s | |
| IG | .21 | .21 | .14 | .15 | .20 | .32 | |
| Pierre | P | = | np | r | r | e | np |
| Vinken | V | np | , | k | e | n | np |
| 61 | 6 | , | nns | = | 6 | 1 | cd |
| years | y | cd | jj-np | a | r | s | nns |
| old | o | nns | , | o | l | d | jj |
| join | j | md | dt | o | i | n | vb |
| board | b | dt | in-rb | a | r | d | nn |
| nonexecutive | n | dt | nn-np | i | v | e | jj |
| director | d | jj | np | t | o | r | nn |
| nov. | n | nn | cd | o | v | . | np |
| 29 | 2 | np | . | = | 2 | 9 | cd |

Table 3: Case representation and information gain pattern for unknown words.

## 5.4 Control

Figure 3 shows the architecture of the tagger-generator: a tagger is produced by extracting a lexicon and two case-bases from the tagged example corpus. During tagging, the control is the following: words are looked up in the lexicon and separated into known and unknown words. They are retrieved

[3]These parameters (optimal context size and number of suffix features) were again optimised for generalization accuracy.

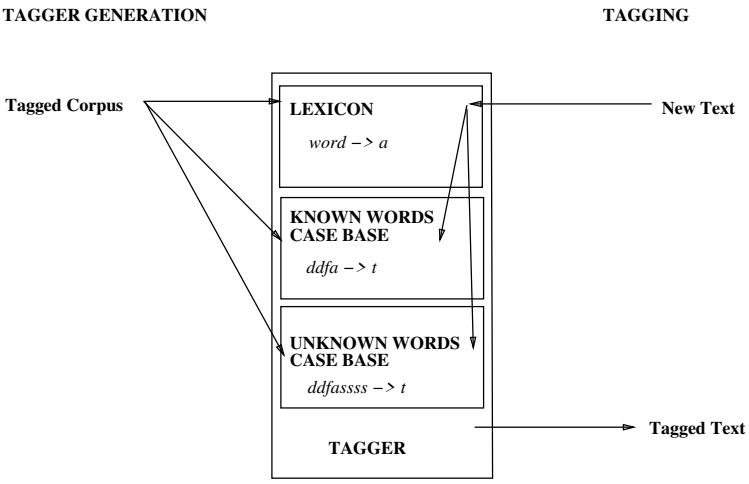


Figure 3: Architecture of the tagger-generator: flow of control.

from the known words case base and the unknown words case base, respectively. In both cases, context is used, in the case of unknown words, the first and three last letters of the word are used instead of the ambiguous tag for the focus word. As far as *disambiguated* tags for left context words are used, these are of course not obtained by retrieval from the lexicon (which provides ambiguous categories), but by using the previous decisions of the tagger.

## 5.5 IGTrees for Tagging

As explained earlier, both case bases are implemented as IGTrees. For the known words case base, paths in the tree represent *variable size* context widths. The first feature (the expansion of the root node of the tree) is the focus word, then context features are added as further expansions of the tree until the context disambiguates the focus word completely. Further expansion is halted at that point. In some cases, short context sizes (e.g. corresponding to bigrams) are sufficient to disambiguate a focus word, in other cases, more context is needed. IGTrees provide an elegant way of automatic determination of optimal context size. In the unknown words case base, the tree representation provides an automatic integration of information about the form and the context of a focus word not encountered before. In general, the top levels of the tree represent the morphological information (the three suffix letter features and the prefix letter), while the deeper levels contribute contextual disambiguation.

# 6 Experiments

In this section, we report first results on our memory-based tagging approach. In a first set of experiments, we compared our IGTREE implementation of memory-based learning to more traditional implementations of the approach. In further experiments we studied the performance of our system on predicting the category of both known and unknown words.

### Experimental Set-up

The experimental methodology was taken from Machine Learning practice (e.g. (Weiss and Kulikowski, 1991): independent training and test sets were selected from the original corpus, the system was trained on the training set, and the generalization accuracy (percentage of correct category assignments) was computed on the independent test set. Storage and time requirements were computed as well. Where possible, we used a 10-fold cross-validation approach. In this experimental method, a data set is partitioned ten times into 90% training material, and 10% testing material. Average accuracy provides a reliable estimate of the generalization accuracy.

## 6.1 Experiment 1: Comparison of Algorithms

Our goal is to adhere to the concept of memory-based learning with full memory while at the same time keeping memory and processing speed within attractive bounds. To this end, we applied the IGTREE formalism to the task. In order to prove that IGTREE is a suitable candidate for practical memory-based tagging, we compared three memory-based learning algorithms: (i) IB1, a slight extension (to cope with symbolic values and ambiguous training items) of the well-known *k-nn* algorithm in statistical pattern recognition (see (Aha, Kibler, and Albert, 1991)), (ii) IB1-IG, an extension of IB1 which uses feature relevance weighting (described in Section 2), and (iii) IGTREE, a memory- and processing time saving heuristic implementation of IB1-IG (see Section 3). Table 4 lists the results in generalization accuracy, storage requirements and speed for the three algorithms using a `ddfat` pattern, a 100,000 word training set, and a 10,000 word test set. In this experiment, accuracy was tested on known words only.

| Algorithm | Accuracy | Time | Memory (Kb) |
|---|---|---|---|
| IB1 | 92.5 | 0:43:34 | 977 |
| IB1-IG | 96.0 | 0:49:45 | 977 |
| IGTree | 96.0 | 0:00:29 | 35 |

Table 4: Comparison of three memory-based learning techniques.

The IGTREE version turns out to be better or equally good in terms of

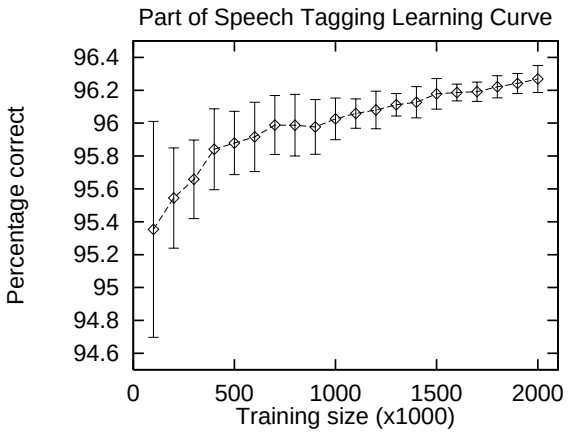


Figure 4: Learning curve for tagging

generalization accuracy, but also is more than 100 times faster for tagging of new words[4], and compresses the original case base to 4% of the size of the original case base. This experiment shows that for this problem, we can use IGTREE as a time and memory saving approximation of memory-based learning (IB-IG version), without loss in generalization accuracy. The time and speed advantage of IGTREE grows with larger training sets.

## 6.2 Experiment 2: Learning Curve

A ten-fold cross-validation experiment on the first two million words of the WSJ corpus shows an average generalization performance of IGTREE (on known words only) of 96.3%. We did 10-fold cross-validation experiments for several sizes of datasets (in steps of 100,000 memory items), revealing the learning curve in Figure 4. Training set size is on the $X$-axis, generalization performance as measured in a 10-fold cross-validation experiment is on the $Y$-axis. the 'error' range indicate averages plus and minus one standard deviation on each 10-fold cross-validation experiment.[5]

Already at small data set sizes, performance is relatively high. With increasingly larger data sets, the performance becomes more stable (witness the error ranges). It should be noted that in this experiment, we assumed correctly disambiguated tags in the left context. In practice, when using our tagger, this is of course not the case because the disambiguated tags in the left context of the current word to be tagged are the result of a previous

[4] In training, i.e. building the case base, IB1 and IB1-IG (4 seconds) are faster than IGTREE (26 seconds) because the latter has to build a tree instead of just storing the patterns.

[5] We are not convinced that variation in the results of the experiments in a 10-fold-cv set-up is statistically meaningful (the 10 experiments are not independent), but follow common practice here.

decision of the tagger, which may be a mistake. To test the influence of this effect we performed a third experiment.

### 6.3 Experiment 3: Overall Accuracy

We performed the complete tagger generation process on a 2 million words training set (lexicon construction and known and unknown words case-base construction), and tested on 200,000 test words. Performance on known words, unknown words, and total are given in Table 5. In this experiment, numbers were not stored in the known words case base; they are looked up in the unknown words case base.

| | Accuracy | Percentage |
|---|---|---|
| Known | 96.7 | 94.5 |
| Unknown | 90.6 | 5.5 |
| Total | 96.4 | 100.0 |

Table 5: Accuracy of IGTREE tagging on known and unknown words

## 7 Related Research

A case-based approach, similar to our memory-based approach, was also proposed by Cardie (1993a; Cardie (1994) for sentence analysis in limited domains (not only POS tagging but also semantic tagging and structural disambiguation). We will discuss only the reported POS tagging results here. Using a fairly complex case representation based on output from the CIRCUS conceptual sentence analyzer (22 local context features describing syntactic and semantic information about a five-word window centered on the word to be tagged, including the words themselves, and 11 global context features providing information about the major constituents parsed already), and with a tag set of 18 tags (7 open-class, 11 closed class), she reports a 95% tagging accuracy. A decision-tree learning approach to feature selection is used in this experiment (Cardie, 1993b; Cardie, 1994) to discard irrelevant features. Results are based on experiments with 120 randomly chosen sentences from the TIPSTER JV corpus (representing 2056 cases). Cardie (p.c.) reports 89.1% correct tagging for unknown words. Percentage unknown words was 20.6% of the test words, and overall tagging accuracy (known and unknown) 95%. Notice that her algorithm gives no initial preference to training cases that match the test word during its initial case retrieval. On the other hand, after retrieving the top k cases, the algorithm does prefer those cases that match the test word when making its final predictions. So, it is understandable that the algorithm is doing better on words that it has seen during training as opposed to unknown words.

In our memory-based approach, feature weighting (rather than feature selection) for determining the relevance of features is integrated much more smoothly with the similarity metric, and our results are based on experiments with a larger corpus (3 million cases). Our case representation is (at this point) simpler: only the (ambiguous) tags, not the words themselves or any other information are used. The most important improvement is the use of IGTREE to index and search the case base, solving the computational complexity problems a case-based approach would run into when using large case bases.

An approach based on $k$-nn methods (such as memory-based and case-based methods) is a statistical approach, but it uses a different kind of statistics than Markov model-based approaches. $k$-nn is a non-parametric technique; it assumes no fixed type of distribution of the data. The most important advantages compared to current stochastic approaches are that (i) few training items (a small tagged corpus) are needed for relatively good performance, (ii) the approach is incremental: adding new cases does not require any recomputation of probabilities, and (iii) it provides explanation capabilities, and (iv) it requires no additional smoothing techniques to avoid zero-probabilities; the IGTREE takes care of that.

Compared to hand-crafted rule-based approaches, our approach provides a solution to the knowledge-acquisition and reusability bottlenecks, and to robustness and coverage problems (similar advantages motivated Markov model-based statistical approaches). Compared to *learning* rule-based approaches such as the one by Brill (1992), a $k$-nn approach provides a uniform approach for all disambiguation tasks, more flexibility in the engineering of case representations, and a more elegant approach to handling of unknown words (see e.g. (Cardie, 1994)).

## 8 Conclusion

We have shown that a memory-based approach to large-scale tagging is feasible both in terms of accuracy (comparable to other statistical approaches), and also in terms of computational efficiency (time and space requirements) when using IGTREE to compress and index the case base. The approach combines some of the best features of learned rule-based and statistical systems (small training corpora needed, incremental learning, understandable and explainable behavior of the system). More specifically, memory-based tagging with IGTREEs has the following advantages.

- Accurate generalization from small tagged corpora. Already at small corpus size (300–400 K tagged words), performance is good. These corpus sizes can be easily handled by our system.

- Incremental learning. New 'cases' (e.g. interactively corrected output of the tagger) can be incrementally added to the case bases, continually improving the performance of the overall system.
- Explanation capabilities. To explain the classification behavior of the system, a path in the IGTREE (with associated defaults) can be provided as an explanation, as well as nearest neighbors from which the decision was extrapolated.
- Flexible integration of information sources. The feature weighting method takes care of the optimal fusing of different sources of information (e.g. word form and context), automatically.
- Automatic selection of optimal context. The IGTREE mechanism (when applied to the known words case base) automatically decides on the optimal context size for disambiguation of focus words.
- Non-parametric estimation. The IGTREE formalism provides automatic, nonparametric estimation of classifications for low-frequency contexts (it is similar in this respect to backed-off training), but avoids non-optimal estimation due to false intuitions or non-convergence of the gradient-descent procedure used in some versions of backed-off training.
- Reasonably good results on unknown words without morphological analysis. On the WSJ corpus, unknown words can be predicted (using context and word form information) for more than 90%.
- Fast learning and tagging. Due to the favorable complexity properties of IGTREEs (lookup time in IGTREEs does not depend on the number of cases), both tagger generation and tagging are extremely fast. Tagging speed in our current implementation is about 1000 words per second.

We have barely begun to optimise the approach: a more intelligent similarity metric would also take into account the differences in similarity between different values of the same feature. E.g. the similarity between the tags `rb-in-nn` and `rb-in` should be bigger than the similarity between `rb-in` and `vb-nn`. Apart from linguistic engineering refinements of the similarity metric, we are currently experimenting with statistical measures to compute such more fine-grained similarities (e.g. (Stanfill and Waltz, 1986; Cost and Salzberg, 1993)).

## References

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

Brill, E. 1992. A simple rule-based part-of-speech tagger. In *Proceedings Third ACL Applied*, pages 152–155, Trento, Italy.

Cardie, C. 1993a. A case-based approach to knowledge acquisition for domain-specific sentence analysis. In *AAAI-93*, pages 798–803.

Cardie, C. 1993b. Using decision trees to improve case-based learning. In *Proceedings of the Tenth International Conference on Machine Learning*, pages 25–32.

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

Chandler, S. 1992. Are rules and modules really necessary for explaining language? *Journal of Psycholinguistic research*, 22(6):593–606.

Church, K. 1988. A stochastic parts program and noun phrase parser for unrestricted text. In *Proceedings Second ACL Applied NLP*, pages 136–143, Austin, Texas.

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

Cutting, D., J. Kupiec, J. Pederson, and P. Sibun. 1992. A practical part of speech tagger. In *Proceedings Third ACL Applied NLP*, pages 133–140, Trento, Italy.

Daelemans, Walter. 1995. Memory-based lexical acquisition and processing. In P. Steffens, editor, *Machine Translation and the Lexicon*, volume 898 of *Lecture Notes in Artificial Intelligence*. Springer, Berlin, pages 85–98.

Daelemans, Walter and Antal van den Bosch. 1992. Generalisation performance of backpropagation learning on a syllabification task. In Mark Drossaers and Anton Nijholt, editors, *TWLT3: Connectionism and Natural Language Processing*, pages 27–38, Enschede. Twente University.

Daelemans, Walter, Jacub Zavrel, Peter Berck, and Steven Gillis. 1996. Mbt: A memory-based part of speech tagger generator. In *Proceedings of the Fourth Workshop on Very Large Corpora*, pages 14–27, Copenhagen. ACL SIGDAT.

DeRose, S. 1988. Grammatical category disambiguation by statistical optimization. *Computational Linguistics*, 14:31–39.

Derwing, Bruce L. and Royal Skousen. 1989. Real time morphology: Symbolic rules or analogical networks. *Berkeley Linguistic Society*, 15:48–62.

Federici, Stefano and Vito Pirelli. forthcoming. Analogy, computation and linguistic theory. In Daniel Jones, editor, *New Methods in Language Processing*. UCL Press, London.

Friedman, Jerome, Jon Bentley, and R. Ari Finkel. 1977. An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software*, 3(3):209–227.

Garside, R., G. Leech, and G. Sampson. 1987. *The computational analysis of English: A corpus-based approach*. Longman, London.

Greene, B.B. and G.M Rubin. 1971. Automatic grammatical tagging of english. Technical report, Department of Linguistics, Brown University, Providence RI.

Hindle, Donald. 1989. Acquiring disambiguation rules from text. In *Proceedings, 27th Annual Meeting of the Association for Computational Linguistics*, Vancouver, BC.

Hunt, E., J. Marin, and P. Stone. 1966. *Experiments in Induction*. Academic Press, New York.

Jones, Daniel. 1996. *Analogical Natural Language Processin*. UCL Press, London.

Kitano, H. 1993. Challenges of massive parallelism. In *proceedings of IJCAI*, pages 813–834.

Klein, S. and R. Simmons. 1963. A grammatical approach to grammatical coding of english words. *JACM*, 10:334–347.

Kolodner, J. 1993. *Case-Based Reasoning*. Morgan Kaufmann, San Mateo.

Langley, Pat and S. Sage. 1994. Oblivious decision trees and abstract cases. In David W. Aha, editor, *Case-Based Reasoning: Papers from the 1994 Workshop*, Technical Report WS-94-01, Menlo Park, CA. AAAI Press.

Merialdo, B. 1994. Tagging english text with a probabilistic model. *Computational Linguistics*, 20 (2):155–172.

Pereira, F., Y. Singer, and N. Tishby. 1995. 'beyond word n-grams. In *Proceedings Third Workshop on Very Large Corpora*, pages 95–106, Cambridge Mass. MIT.

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

Salzberg, Steven. 1990. A nearest hyperrectangle learning metho. *Machine Learning*, 6:251–276.

Samuelsson, C. 1994. Morphological tagging based entirely on bayesian inference. In *Proceedings of the 9th Nordic Conference on Computational Linguistics*, Sweden. Stockholm University.

Scha, Remco. 1992. Virtuele grammatica's en creatieve algoritmen. *Gramma/TTT*, 1 (1):57–77.

Schmid, H. 1994. Part-of-speech tagging with neural networks. In *Proceedings of COLING*, Kyoto, Japan.

Schütze, H. and Y. Singer. 1994. Part-of-speech tagging using a variable context markov model. In *Proceedings of ACL 1994*, Las Cruces, New Mexico.

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

Skousen, Royal. 1989. *Analogical Modeling of Language*. Kluwer, Dordrecht.

Stanfill, C. and D. Waltz. 1986. Toward memory-based reasoning. *Communications of the ACM*, 29:1212–1228.

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

Wettschereck, D., D. W. Aha, and T. Mohri. 1996. A review and comparative evaluation of feature weighting methods for lazy learning algorithms. Technical report aic-95-012, Navy Center for Applied Research in Artificial Intelligence, Washington, DC.