Proceedings of NEMLAP 1996, Ankara

Morphological Analysis as Classification: an Inductive-Learning Approach

Antal van den Bosch[(i)], Walter Daelemans[(ii)], Ton Weijters[(i)]

[(i)] Dept. of Computer Science / MATRIKS
Maastricht University
PO Box 616, 6200 MD Maastricht, THE NETHERLANDS

[(ii)] Computational Linguistics
Tilburg University
PO Box 90153, 5000 LE Tilburg, THE NETHERLANDS

Abstract. Morphological analysis is an important subtask in text-to-speech conversion, hyphenation, and other language engineering tasks. The traditional approach to performing morphological analysis is to combine a morpheme lexicon, sets of (linguistic) rules, and heuristics to find a most probable analysis. In contrast we present an inductive learning approach in which morphological analysis is reformulated as a segmentation task. We report on a number of experiments in which five inductive learning algorithms are applied to three variations of the task of morphological analysis. Results show (i) that the generalisation performance of the algorithms is good, and (ii) that the *lazy learning* algorithm IB1-IG performs best on all three tasks. We conclude that lazy learning of morphological analysis as a classification task is indeed a viable approach; moreover, it has the strong advantages over the traditional approach of avoiding the knowledge-acquisition bottleneck, being fast and deterministic in learning and processing, and being language-independent.

1 Introduction

Morphological analysis is often deemed to be an important, if not essential subtask in linguistic modular systems for text-to-speech processing [2] and hyphenation [4]. In text-to-speech processing, it serves to prevent the wrong application of grapheme-phoneme conversion rules across morpheme boundaries (e.g., preventing carelessly from being pronounced as /kə'rɛləslai/). In hyphenation (in British English and to a lesser degree in Dutch), it guides the placement of hyphens at certain morpheme boundaries (e.g., preventing looking from being hyphenated as loo-king). Morphological analysis also plays a crucial role in applications such as part-of-speech tagging (assigning the correct morpho-syntactic category to words in context), for obtaining a reasonable analysis of words not present in the lexicon.

The traditional approach to performing morphological analyses presupposes the availability of a morpheme lexicon, spelling rules, morphological rules, and heuristics to prioritise possible analyses of a word according to their plausibility (e.g., see the DECOMP module in the MITtalk system [2]). In contrast, the approach described in this paper presupposes a morphologically analysed corpus of words (rather than a corpus of morphemes), and an inductive learning algorithm trained to segment spelling words into morphemes in the form of a simple classification task.

In this paper, we will first outline what we mean by rephrasing a linguistic problem as a classification task, and we will introduce five inductive-learning algorithms which are applied to

this task. Then, in section 2, we give an overview of the traditional approach to morphological analysis and introduce our alternative reformulation. In section 3 we present and analyse the results of the application of the learning algorithms to this classification task. We conclude this paper with a summary of the obtained results and a discussion of the differences between the traditional approach to morphological analysis and a inductive-learning approach, in section 4.

1.1 Reformulating linguistic problems as classification tasks

Most linguistic problems can be seen as context-sensitive mappings from one representation to another (e.g., from text to speech; from a sequence of spelling words to a parse tree; from a parse tree to logical form, from source language to target language, etc.). The typical traditional approach to language engineering problems is to build a description of the general rules governing these mappings, describe additional subregularities, and list the remaining exceptions to the rules and subregularities. The acquisition of this knowledge is labour-intensive and costly. In contrast to this hand-crafting approach, an inductive machine-learning method approaches a linguistic problem in a data-oriented way, i.e., it automatically gathers the knowledge needed for solving the problem by considering instances of the problem. By 'instance' we mean a data structure containing an input and its associated 'solution'; its classification. The knowledge implicitly present in the collection of instances is used to classify new instances of the same problem.

Most linguistic tasks can be described as classification tasks, i.e., given a description of an input in terms of a number of feature-values, a classification of the input is performed. Two types of classification tasks can be discerned [5]:

- **Identification**: given a set of possible classifications and an input of feature values, determine the correct classification for this input. For example, given a letter surrounded by a number of neighbours (e.g., a in have), determine the phonemic transcription of that letter.
- **Segmentation**: given a set of possible boundary classes and an input consisting of a focus position in its immediate context, determine whether a boundary is associated with the focus position, and if so, which one. For example, determine if the b in table marks the boundary of a syllable.

Differences exist in the ways inductive algorithms extract knowledge from the available instances. In *lazy learning* (such as memory-based learning [14, 5]), there is no abstraction of higher-level data structures such as rules or decision trees at learning time; learning consists of simply storing the instances in memory. A new instance of the same problem is solved by retrieving those instances from memory that match the new instance best (according to a similarity metric), and by extrapolating from the solutions of these 'nearest neighbours'. The *memory-based learning approach* therefore does not distinguish between regularities and individual exceptions; rule-like behaviour is the result of the interaction between the memory contents and the similarity metric used. In *eager learning* approaches (such as C4.5 or connectionist learning), abstract data structures (matrices of connection weights in connectionist networks, decision trees in C4.5) are extracted from the learning material during learning.

In previous research we have demonstrated the application of the memory-based (lazy) learning approach to several linguistic problems, e.g., segmentation as in hyphenation and syllabification [6, 17], and identification as in grapheme-phoneme conversion [18, 16, 7], and stress assignment [8]. In most cases, the memory-based (lazy) approach outdid the more eager inductive algorithms. We believe that in a 'noisy' domain such as natural language, abstracting

from the training instances is a bad idea because any one instance (however 'exceptional' from the point of view of the learning algorithm) can potentially be a model for new instances.

In this paper, we will demonstrate that the segmentation approach of memory-based learning is also applicable to morphological parsing. We will compare the approach to alternative inductive machine-learning algorithms. First, we provide a brief summary of the inductive-learning algorithms used in the experiments reported in this paper.

1.2 Algorithms and methods for inductive learning

Inductive learning in its most straightforward form is exhibited by memory-based *lazy learning* algorithms such as IB1 [1] and variations (e.g., IB1-IG [6, 9]), in which all instances are fully stored in memory, and in which classification involves a pass along all stored instances. To optimise memory lookup and minimise memory usage, more eager learning algorithms are available that compress the memory in such a way that most relevant knowledge is retained and stored in a quickly accessible form, and redundant knowledge is removed. Examples of such algorithms are the decision-tree algorithms IGTREE [9] and C4.5 [12]. Another popular inductive algorithm is the connectionist Back-propagation (BP) [13] learning algorithm. We provide a summary of the basic functions of these learning algorithms.

1. IB1 [1] constructs a data base of instances (the *instance base*) during learning. An instance consists of a fixed-length vector of n feature-value pairs, and an information field containing the classification(s) of that particular feature-value vector. When the feature-value vector is associated to more than one classification (i.e., when its classification is ambiguous), the occurrences of the different classifications in the learning material are counted and stored with the instance. After the instance base is built, new instances are classified by IB1 by matching them to all instances in the instance base, and calculating with each match the *distance* between the new instance X and the memory instance Y, $\Delta(X, Y)$, using the function in equation 1.

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

   where $W(fi)$ is the weight of the ith feature (in IB1, this weight is equal for all features), and $\delta(x_i, y_i)$ is the distance between the values of the ith feature in instances X and Y. When the values of the instance features are symbolic, as with our linguistic tasks, a simple distance function for $\delta(x_i, y_i)$ is used (equation 2).

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

   The (most frequently occurring) classification of the memory instance Y with the smallest $\Delta(X, Y)$ is then taken as the classification of X.
2. IB1-IG [6, 9] differs from IB1 in the weighting function $W(f_i)$ (cf. equation 1). This function computes for each feature, over the full instance base, its *information gain*, a function from information theory that is also used in ID3 [11] and C4.5 [12] (for more details, cf. Daelemans and Van den Bosch [6]). In short, the information gain of a feature expresses its relative importance compared to the other features in performing the mapping from input to classification. This weighting function gives right to the fact that for some tasks, some features are far more important than other features. When information gain is used as the weighting function in the similarity function (equation 1), instances that match on an important feature are regarded as more alike than instances that match on an unimportant feature.

3. IGTREE [9] compresses an instance base into a decision tree. Instances are stored in the tree as paths of connected nodes, and leaves containing classification information. Nodes are connected via arcs denoting feature values. Information gain is used in IGTREE to determine the order in which instance feature values are added as arcs to the trie. The reasoning behind this compression is that when the computation of information gain points to one feature clearly being the most important in classification, search can be restricted to matching a test instance to those memory instances that have the same feature value as the test instance at that feature. Instead of indexing all memory instances only once on this feature, the instance memory can then be optimised further by examining the second most important feature, followed by the third most important feature, etc. A considerable compression is obtained as similar instances share partial paths. The trie structure is compressed even more by restricting the paths to those input feature values that disambiguate the classification from all other instances in the training material. The idea is that it is not necessary to fully store an instance as a path when only a few feature values of the instance make the instance classification unique. In applications to linguistic tasks, IGTREE is shown to obtain compression factors of 90% or more as compared to IB1/IB1-IG [16, 7].

   IGTREE also stores with each non-terminal node information concerning the *most probable* or *default* classification given the path thus far, according to the classification bookkeeping information maintained by the trie construction algorithm. This extra information is essential when processing new instances. Processing a new instance involves traversing the trie (i.e., matching all feature-values of the test instance with arcs in the order of the overall feature information gain), and either retrieving a classification when a leaf is reached (i.e., an exact match was found), or using the default classification on the last matching non-terminal node if an exact match fails. For more details on IGTREE, see Daelemans et al. [9].

4. C4.5 [12] is a well-known decision-tree algorithm which basically uses the same type of strategy as IGTREE to compress an instance base into a compact tree. To this purpose, standard C4.5 also uses information gain, or *gain ratio* [12] to select the most important feature in tree building; however, in contrast to IGTREE, C4.5 recomputes this function for each node in the tree. Another difference with IGTREE is that C4.5 implements a pruning stage, in which parts of the tree are removed as they are estimated to contribute to instance classification below a certain utility threshold.

5. BP [13] is an artificial-neural-network learning rule, which operates on multi-layer feed-forward networks (MFNs). In these networks, feature-values of instances are encoded as activation patterns in the input layer, and the network is trained to produce an activation pattern at the output layer representing the desired classification. In contrast to the previously described algorithms, BP does not accumulate its knowledge by literally storing (parts of) instances in memory or by constructing a decision tree on the basis of them. Rather, BP tunes the connections between units in the input layer and the hidden layer, and between units of the hidden layer and the output layer, during a training phase in which all training instances are presented several times to the network. The BP learning algorithm, which is a gradient descent algorithm, attempts to set the connections between the layers with increasing subtlety, aiming at minimisation of the error on the training material. After training, the units at the hidden layer encode an intermediary representation that captures (in an often opaque way) some essential information from both the input (the feature-values) and the output (the desired classification). These representations are non-symbolic, and do not lend themselves easily for inspection, in contrast to the previously described symbolic algorithms.

When one plans to apply learning algorithms to classification tasks, it is important to establish a method for interpreting the results from such experiments beforehand. In our experiments, we are primarily interested in the *generalisation accuracy* of trained models, i.e., the ability of these models to use their accumulated knowledge to classify new instances that were not in the training material. A method that gives a good estimate of the generalisation performance of an algorithm on a given instance base, is *10-fold cross-validation* [19]. This method generates on the basis of an instance base 10 partitionings into a training set (90%) and a test set (10%), resulting in 10 experiments and 10 results per learning algorithm and instance base. Significance tests such as one-tailed t-tests can be applied to the outcomes of 10-fold cross-validation experiments with several learning algorithms trained on the same data.

2 Morphological analysis

2.1 Traditional approaches

The traditional approach to morphological analysis basically presupposes three components: (i) a morpheme lexicon, (ii) a set of spelling rules and morphological rules to discover possible analyses of morphologically complex words, and (iii) prioritising heuristics to choose the most probable analysis from sets of possible analyses. We briefly illustrate the functioning of this type of analysis by taking DECOMP's processing as an example, and the word scarcity as the example word [2]:

1. In a morpheme lexicon covering the English language, a first analysis divides scarcity into scar and city.
2. The analysis scar|city is validated by a finite-state automaton covering the possible sequences of morphemes in English words; furthermore, an analysis-cost heuristic assigns an integer-valued cost to the combination of the two noun stems.
3. Using spelling rules for letter deletion in inflection and compounding in English, the system suspects that the analysis scarce|ity is also possible, as ity may have deleted the e of scarce. This analysis, which is validated by the morpheme-sequence finite-state automaton, yields a lower cost than scar|city, as the analysis-cost heuristic assigns a lower value to a derivational affix than to a second stem.
4. As no further spelling-change rules can be applied to the analysis with the lowest cost, scarce|ity, the process ends by producing this analysis.

It is argued in Allen et al. (1987) that a morpheme lexicon containing 10,000 morphemes is effective in a text-to-speech system. Neologisms, a problem for purely lexicon-based approaches, seldomly contain new morphemes. The morpheme-sequence finite-state automaton, the spelling rules, and the analysis-cost heuristic are in principle not very complex in terms of processing. They demand, however, a considerable amount of knowledge acquisition and fine-tuning. Another serious problem with these analysis components is that the number of analyses of morphologically complex words may become very much larger (near exponential in the number of morphemes) for longer words.

Morphological analysis on a probabilistic basis, using only a morpheme lexicon, an analyses generator, and a probabilistic function to determine the analysis with the highest probability [10] does not suffer from the disadvantageous knowledge acquisition and fine-tuning phase, but is nevertheless also confronted with an explosion of the number of generated analyses.

2.2 Inductive-learning approach

In contrast to this decomposition into three components, we reformulate the task of morphological analysis as a one-pass segmentation task, in which an input (a sequence of letters with a focus position) is to be classified as marking a morpheme boundary at that focus position. This classification approach demands that the number of input features be fixed, hence we cannot use whole words as input. Instead, we convert a word into fixed-sized instances of which the middle letter is mapped to a class denoting a morpheme boundary decision. To generate fixed-sized instances, we adopt the windowing scheme proposed by Sejnowski and Rosenberg (1987) which generates fixed-sized snapshots of words.

In its most basic form, the classification of each instance denotes whether the focus letter of the instance maps to a morpheme boundary ('YES', or '1') or not ('NO', or '0'). However, distinguishing between only '1' and '0' does not take into account that morphological theory generally distinguishes between several types of morphemes. For the case of English, a family tree of morphemes would for example be the one displayed in Figure 1.

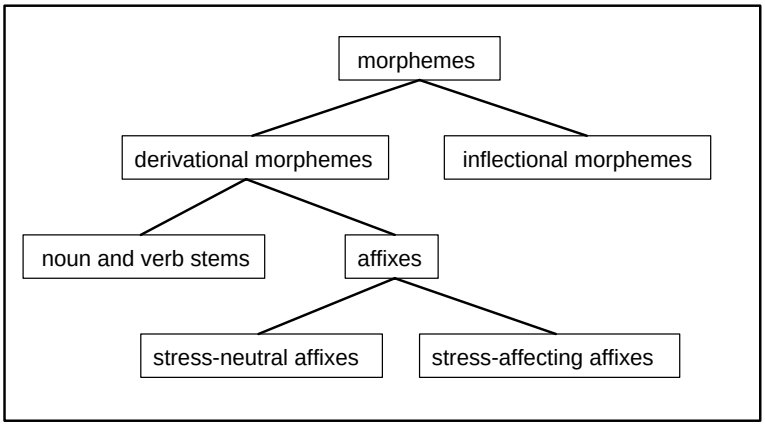

Figure 1. Family tree of English morphemes.

Distinguishing between, for example, stress-neutral and stress-affecting affixes would be directly helpful as input knowledge for performing the stress-assignment task in a text-to-speech system. However, distinguishing between types of morphemes according to this theory also introduces a certain amount of pre-wired linguistic knowledge. With this in mind we extended the task of morphological analysis into three different tasks, with increasing implicit linguistic knowledge encoded in the classes:

task M1 - decide whether the focus letter marks the beginning of
- a morpheme: map to class '1',
- no morpheme: class '0'.

task M2 - decide whether the focus letter marks the beginning of
- a derivational morpheme: class 'd',
- an inflectional morpheme: class 'i',
- no morpheme: class '0'.

task M3 - decide whether the focus letter marks the beginning of
- a noun or verb stem: class 's',
- a stress-neutral affix: class '1',

– a stress-affecting affix: class '2',
– an inflectional morpheme: class 'i',
– no morpheme: '0'.

Applying the windowing method to the example word abnormalities leads to the instances displayed in Table 1, listing for each of the three tasks their appropriate classifications. The morphological analysis of the full word is simply the concatenation of the instance classifications, in which all classifications other than '0' mark morpheme boundaries.

INSTANCE NUMBER	LEFT CONTEXT	FOCUS LETTER	RIGHT CONTEXT	CLASSIFICATION M1	M2	M3
1	_ _ _	a	b n o	1	d	1
2	_ _ a	b	n o r	0	0	0
3	_ a b	n	o r m	1	d	s
4	a b n	o	r m a	0	0	0
5	b n o	r	m a |	0	0	0
6	n o r	m	a | i	0	0	0
7	o r m	a	| i t	1	d	1
8	r m a	|	i t i	0	0	0
9	m a |	i	t i e	1	d	2
10	a | i	t	i e s	0	0	0
11	| i t	i	e s _	0	0	0
12	i t i	e	s _ _	1	i	i
13	t i e	s	_ _ _	0	0	0

Table 1. Instances with morphological analysis classifications derived from the word ab|norm|al||iti|es. The three classification fields belong to tasks M1, M2, and M3, respectively. Denotations of the classification labels is as follows: 0 = no morpheme boundary; 1 = morpheme boundary with M1, and stress-neutral affix with M3; 2 = stress-affecting affix; d = derivational boundary; i = inflectional boundary; s = stem boundary.

As can be seen from Table 1, a morpheme boundary is assigned to the position at which a new morpheme begins, regardless of the spelling changes that may have occurred in the vicinity of that position. For example, the analysis displayed in Table 1 states that the 'surface' form iti is a stress-affecting affix, although its 'deep' form is ity. A second characteristic of our representation of morpheme boundaries, is that it is non-hierarchic. Although morpheme hierarchy may be important in determining the part-of-speech of a word [2], it is not necessary to have a full hierarchical analysis when the morphological analysis is used as input to a text-to-speech system.

3 Experiments

3.1 Data collection and algorithmic parameters

The source for the morphological data used in our experiments is CELEX [3], a large lexical data base of English, Dutch, and German. We extracted from the English data base all relevant information on wordforms relating to spelling and morphology, and created a lexicon of 65,558 morphologically analysed words. This lexicon was used to create instance bases for the M1, M2, and M3 tasks, each containing 573,544 instances.

For completeness, the learning parameters of the five algorithms described in section 1, viz. IB1, IB1-IG, IGTREE, C4.5, and BP, as used in our experiments, are the following: (i) IB1 and IB1-IG implement 1-nearest neighbour matching; (ii) C4.5 uses the gain ratio criterion, default pruning, and no subsetting of feature-values; (iii) BP uses a network with 294 input units (letters are locally coded), 50 hidden units, and 2, 3, or 5 output units (classes are locally coded), a learning rate of 0.1, a momentum of 0.4, and an update tolerance of 0.2. IGTREE's functioning is not governed by parameters.

3.2 Results

We applied the five algorithms to the three tasks, performing with each algorithm and each task a 10-fold cross-validation experiment [19]. We computed for each 10-fold cross-validation experiment the average percentage of incorrectly processed test words. A word is incorrectly processed when *one or more* instance classifications associated with the instances derived from the word are incorrect (i.e., when one or more of the segmentations is incorrect). Figure 2 displays these generalisation errors. The algorithms are ordered on their performance on task M1.

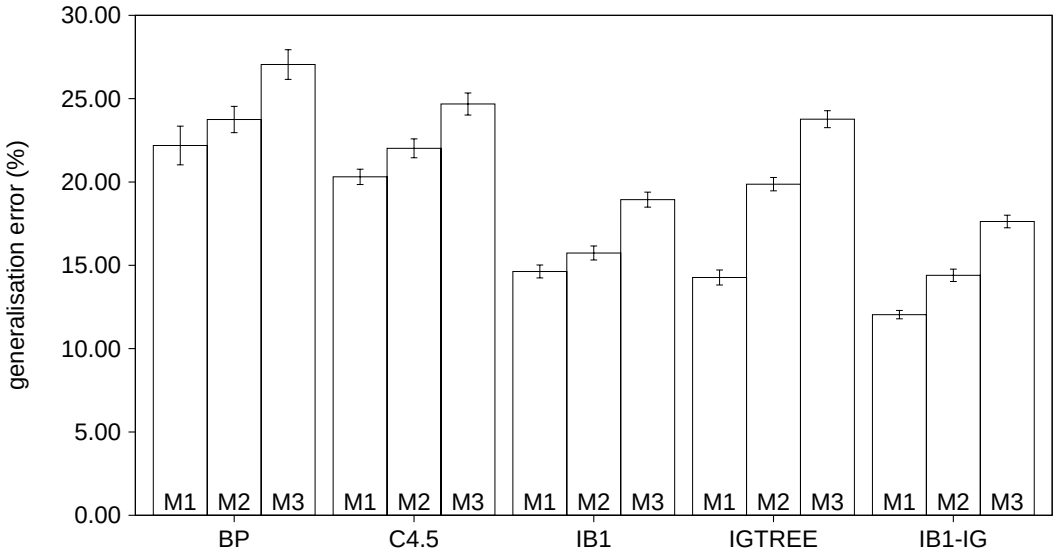

Figure 2. Generalisation errors in terms of the percentage of incorrectly classified test words, with standard deviations (error bars) of five algorithms applied to the three variations of the task of English morphological analysis M1, M2, and M3.

The best performing algorithm on tasks M1, M2, and M3 is IB1-IG. Its performance is significantly better compared to all other algorithms in all three tasks with $p < 0.001$. On task M1, the algorithm performing second best to IB1-IG (12.04% incorrectly processed test words) is IGTREE (14.27%) (level of significance $t(19) = 13.56, p < 0.001$). On task M2, the second best algorithm is IB1 (15.74%) ; IB1-IG processes 14.40% test words incorrectly; ($t(19) = 7.64, p < 0.001$). On task M3, IB1-IG incorrectly processes 17.63% of the test words, again followed by IB1 with 18.94% ($t(19) = 6.95, p < 0.001$).

Interesting is the fact that IGTREE performs well on M1, but performs relatively bad on M2 and M3. IGTREE is known to perform worse when the information gain of the input features displays a low variance [9], i.e., when there is little difference between the 'relative importance' of the input features. This suggests that the information-gain values of the features with tasks M2 and M3 have less outspoken differences than with M1, which is indeed the case, as is displayed in Figure 3. For all three tasks, Figure 3 displays the fact the letter immediately preceding the focus letter is the most important one in the segmentation task.

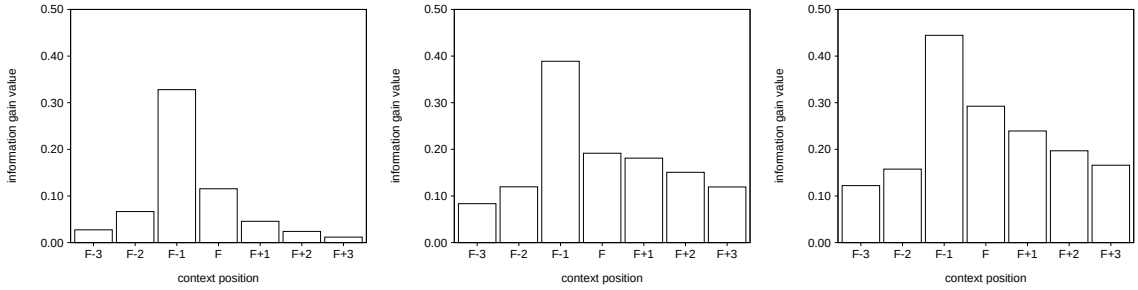

Figure 3. Information-gain values of the features of tasks M1 (left), M2 (middle), and M3 (right), computed over the full instance bases.

A more general observations on the basis of the results displayed in Figure 2 is that tasks M1, M2, and M3 are increasingly difficult to learn for all algorithms. Distinguishing between more output classes with a finer linguistic granularity obviously increases the difficulty of learning the task. The results in Figure 2 also provide an indication that the performance of the best algorithms is quite good, considering (i) the test words are not seen by the algorithms during training, and (ii) the test words are dictionary words, rather than words from a written text corpus: they are on the average morphologically more complex than words from a corpus. When the generalisation performance is expressed in terms of incorrectly classified instances, low error rates are obtained. For example, trained on M1, IB1-IG classifies only 1.65% of all test instances incorrectly (1.97% on M2, and 2.46% on M3).

As an illustration, we provide some examples of segmentations generated by IB1-IG on the first partition of task M1. Most errors are related to (apparent) morphological ambiguities: incorrect boundary insertions in ear|ly, nav|y, and co|a|ed, and missed boundaries in printable, upland, and manslaught|er. Some examples of correctly segmented words that are morphologically complex are horse|whip, nut|ti|est, steep|en, veto|es, and dis|agree|able|ness.

4 Conclusions

We have demonstrated the applicability of an inductive machine-learning approach to morphological analysis, by reformulating the problem as a segmentation task in which letter sequences are classified as marking different types of morpheme boundaries. The generalisation performance of inductive-learning algorithms to the task is good.

An interesting result is that within the class of inductive learning algorithms, generalization accuracy correlates with the degree of eagerness of the inductive algorithm used; best results are obtained with memory-based learning (IB1-IG), a lazy learning algorithm retaining full memory of all training instances with a classification-task-related feature-weighting similarity function.

The methods abstracting most from the instances perform worst. This corroborates our hypothesis that because of the intricate interaction of regularities, subregularities and exceptions present in this task as well as in most other linguistic problems we studied, lazy learning methods are superior to eager learning methods.

In comparison with the traditional approach, in which morphological analysis is performed by a system containing several components, the inductive learning approach applied to a reformulation of the problem as a classification task of the segmentation type, has a number of advantages:

- it presupposes no more linguistic knowledge than explicitly present in the corpus used for training, i.e., it avoids a knowledge-acquisition bottleneck;
- it is language-independent, as it functions on any morphologically analysed corpus in any language;
- learning is automatic and fast;
- processing is deterministic, non-recurrent (i.e., it does not retry analysis generation) and fast, and is only linearly related to the length or morphological complexity of words.

Nevertheless, it also displays two disadvantages:

- it produces an analysis that lacks hierarchy of morphemes;
- it does not recover the 'deep' form of morphemes.

Future work on inductive learning of morphological analysis should include a thorough performance comparison with existing traditional systems for morphological analysis, based on linguistic theory and heuristics such as DECOMP [2] as well as with probabilistic systems [10]. Secondly, we aim at integrating trained models of morphological analysis into larger systems, to investigate whether the enrichment of spelling input with morpheme boundary information improves the generalisation performance of other learning systems trained on, e.g., stress assignment, grapheme-phoneme conversion, and part-of-speech prediction of unknown words.

References

1. Aha, D. W., Kibler, D., & Albert, M. (1991). Instance-based learning algorithms. *Machine Learning*, 7, 37–66.
2. Allen, J., Hunnicutt, S., and Klatt, D. (1987). *From Text to Speech: The MITalk System.* Cambridge, UK: Cambidge University Press.
3. Burnage, G. (1990). *CELEX: A Guide for Users.* Centre for Lexical Information, Nijmegen.
4. Daelemans, W. (1989). Automatic hyphenation: Linguistics versus engineering. In: F.J. Heyvaert and F. Steurs (Eds.), *Worlds behind Words*, 347–364. Leuven University Press.
5. Daelemans, W. (1995). Memory-based lexical acquisition and processing. In: P. Steffens (Ed.), *Machine Translation and the Lexicon*, Springer Lecture Notes in Artificial Intelligence 898, 85–98.
6. Daelemans, W., Van den Bosch, A. (1992). Generalisation performance of backpropagation learning on a syllabification task. In M. Drossaers & A. Nijholt (Eds.), *TWLT3: Connectionism and Natural Language Processing.* Enschede: Twente University.
7. Daelemans, W., Van den Bosch, A. (1994). A language-independent, data-oriented architecture for grapheme-to-phoneme conversion. In *Proceedings of ESCA-IEEE Speech Synthesis Conference '94*, New York.
8. Daelemans, W., Gillis, S., & Durieux, G. (1994). The acquisition of stress, a data-oriented approach. *Computational Linguistics*, 20, 421–451.
9. Daelemans, W., Van den Bosch, A., & Weijters, A. (1996). IGTree: Using trees for compression and classification in lazy learning algorithms. To appear in *Artificial Intelligence Review*, special issue on lazy learning.

10. Heemskerk, J. (1993). *A probabilistic context-free grammar for disambiguation in morphological parsing.* Technical Report 44, ITK, Tilburg University.
11. Quinlan, J. (1986). Induction of decision trees. *Machine Learning*, 1, 81–206.
12. Quinlan, J. (1993). *C4.5: Programs for Machine Learning.* San Mateo, CA: Morgan Kaufmann.
13. Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning internal representations by error propagation. In Rumelhart, D. E. and McClelland, J. L., editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, volume 1: Foundations, pages 318–362. Cambridge, MA: The MIT Press.
14. Stanfill, C., and Waltz, D. (1986). Toward memory-based reasoning. *Communications of the ACM*, 29, 1213–1228.
15. Sejnowski, T. J., Rosenberg, C. S. (1987). Parallel networks that learn to pronounce English text. *Complex Systems*, 1, 145–168.
16. Van den Bosch, A., Daelemans, W. (1993). Data-oriented methods for grapheme-to-phoneme conversion. In *Proceedings of the 6th Conference of the EACL*, 45–53. Utrecht: OTS.
17. Van den Bosch, A., Weijters, A., van den Herik, H. J., and Daelemans, W. (1995b). The profit of learning exceptions. In *Proceedings of the 5th Belgian-Dutch Conference on Machine Learning, BENELEARN'95*, 118–126.
18. Weijters, A. (1991). A simple look-up procedure superior to NETtalk? In *Proceedings of the International Conference on Artificial Neural Networks*, Espoo, Finland.
19. Weiss, S. & Kulikowski, C. (1991). *Computer Systems That Learn.* San Mateo: Morgan Kaufmann.