# ABSTRACTION CONSIDERED HARMFUL: LAZY LEARNING OF LANGUAGE PROCESSING

Walter Daelemans

Computational Linguistics Tilburg University, The Netherlands,
and Center for Dutch Language and Speech, University of Antwerp, Belgium
walter.daelemans@kub.nl

In: van den Herik, J. and T. Weijters (eds.) *Benelearn-96. Proceedings of the 6th Belgian-Dutch Conference on Machine Learning.* MATRIKS: Maastricht, The Netherlands, 3--12, 1996

## 1 Empirical Learning of Natural Language

Browsing through the Machine Learning literature, we find that language learning is not a hot topic in machine learning, and that most work in the area addresses either the computational modeling of child language acquisition or the extraction of domain knowledge from text. The algorithms used are also predominantly analytic (symbol-level learning) rather than empirical (knowledge-level learning). While these are obviously interesting research areas and approaches, the absence of more research on the *empirical learning of language knowledge and behaviour from text and speech data* strikes me as strange. After all, the main problem of the AI discipline of Natural Language Processing (NLP) is a *knowledge acquisition bottleneck*: for each new language, domain, theoretical framework, and application, linguistic knowledge bases (lexicons, rule sets, grammars) have to be built basically from scratch. This problem is not surprising, given the complexity of NLP, and the difficulty of finding 'hard and fast' rules governing language processing.

There are at least three reasons why Machine Learning researchers should become more interested in NLP as an application area.

- **Availability of Large Datasets.** NLP problems provide realistically sized training sets for inductive algorithms. Datasets of tens or hundreds of thousands of instances are readily available. Traditional "benchmark datasets" usually contain far less instances. Experimenting with linguistic datasets will provide a more realistic evaluation of the practical usefulness of algorithms, and will force algorithm designers to work on performance issues.

- **Real-World Application.** "Hand-crafting" NLP knowledge bases has proven to be infeasible or unaffordable for most practical applications. The market pull for applications in NLP (especially Text Analysis and Machine Translation) is enormous, but has not been matched by current language technology due to the knowledge acquisition bottleneck. ML techniques may help in realising the enormous market potential for NLP applications.

- **Complexity of Tasks.** Data sets describing language problems at all levels of description (beneath, at, and above the word level) exhibit a complex interaction of regularities, sub-regularities, pockets of exceptions, idiosyncratic exceptions, and noise. As such, they are a perfect model for a large class of other poorly-understood real world problems (e.g. medical diagnosis) for which it is less easy to find large amounts of data. A better understanding of which algorithms work best for this class of problems will transfer to many other problem classes.

In the remainder of the paper, I will describe an approach to empirical learning of language knowledge which we have been investigating the last five years in Tilburg and Antwerp in the context of the ATILA project. At this point I would like to acknowledge the contributions of my current and former colleagues to this effort, especially those of Antal van den Bosch (Tilburg, now Maastricht), Steven Gillis, Gert Durieux, and Peter Berck (Antwerp), and Jakub Zavrel (Tilburg).

## 2 NLP as a Cascade of Classification Tasks

To solve NLP knowledge acquisition bottlenecks we need ML methods, but in order to achieve any results, we must show that the important NLP tasks can indeed be formulated in a ML framework.

Tasks in NLP are context-sensitive mappings between representations (e.g., from text to speech, from spelling to parse tree, from parse tree to logical form, from source language to target language, etc.). These mappings tend to be many-to-many and complex because they can typically only be described by means of the conflicting regularities, sub-regularities, and exceptions alluded to earlier. E.g., in a *hyphenation programme* for a word processor (the simplest and most prosaic NLP problem imaginable), possible positions for hyphens have to be found in a spelling representation; the task is to find a mapping from a spelling representation to a syllable representation. In Dutch, even this simple task is not trivial because the phonological regularities governing syllable structure are sometimes overruled by more specific constraints from morphology (the morphological structure of the word and the nature of its affixes). On top of that, there are constraints which are conventional or which derive from the foreign origin of words. Linguistic engineering (handcrafting) of a rule set and exception lists for this type of problem is time-consuming, costly, and does not necessarily lead to accurate and robust systems.

Empirical Learning (inductive learning from examples) is fundamentally a *classification* paradigm. Given a description in terms of feature-value pairs of an input, a category label is produced. This category should normally be taken from a finite inventory of possibilities, known beforehand. It is our claim that *all* useful linguistic tasks can be redefined this way and can thus be taken on in a ML context. All linguistic problems can be described as mappings of two kinds: *disambiguation* (or identification) and *segmentation* (identification of boundaries) (see Daelemans, 1995).

- **Disambiguation**. Given a set of possible categories and a relevant context in terms of attribute values, determine the correct category for this context. Instances of disambiguation include *part of speech tagging* (disambiguating the syntactic category of a word), *grapheme-to-phoneme conversion*, *lexical selection in generation*, *morphological synthesis*, *word sense disambiguation*, *term translation*, and *stress assignment*.

- **Segmentation**. Given a target and a context, determine whether a boundary is associated with this target, and if so which one. Examples include *syllabification*, *hyphenation*, *morphological analysis*, and *constituent boundary detection*.

In such a perspective, complex NLP tasks like *parsing* can be defined as a *cascade* of segmentation tasks (finding constituent boundaries) and disambiguation tasks (deciding the morphosyntactic category of words, and the label of constituents, and resolving attachment ambiguities).

An approach often used to arrive at the classification representation needed is the *windowing* method (as used in Sejnowski & Rosenberg, 1986 for text to speech), 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 target item, and the rest as the context.

## 3 Generalization without Abstraction

Especially in Language Engineering applications (building commercially useful NLP applications), the main evaluation criteria for systems are (i) efficiency in terms of space and time requirements, and (ii) accuracy in terms of behaviour of the system on previously unseen input. In a learning framework, this criterion is defined in terms of generalization accuracy –the percentage of correct outputs associated with inputs the system was not trained on– and is usually estimated using cross-validation.

Many empirical learning methods (e.g. top down induction of decision trees, rule induction methods, and supervised connectionist learning algorithms) are *eager* learning methods. They achieve generalization by means of *abstraction*. The original training data is abstracted into a representational structure (condition-action rules, trees, weight matrices) that models the regularities governing the input-output associations in the training data. However, abstraction is not a prerequisite for generalization. There is a class of *lazy learning* algorithms that is based on the fundamental idea that *a system can generalize what it has learned by using its experiences directly, instead of using abstractions extracted from them.* From the research we did on inductive learning for NLP tasks, it becomes clear that this lazy learning approach achieves better generalization accuracy than eager learning algorithms. A possible explanation for this is the structure of NLP tasks discussed earlier: apart from a number of clear generalizations, a lot of subregularities and exceptions exist in the data. Exceptions tend to come in 'families'. It is therefore advantageous to *keep exceptions* (some family members may turn up during testing) rather than abstracting away from them: *being there is better than being probable.*

In the remainder of this section I will shortly present the basics of the lazy learning approach. Later sections will describe applications of this idea to NLP tasks.

Lazy 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.

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, nearest-neighbour, and instance-based (Stanfill and Waltz, 1986; Kolodner, 1993; Aha et al. 1991; Salzberg, 1990). Ideas about this type of analogical reasoning can be found also in non-mainstream linguistics and pyscholinguistics (Skousen, 1989; Derwing & 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 (1996).

### 3.1 Similarity Metric

Performance of a lazy learning 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).

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

We will refer to this approach as IB1 (Aha et al., 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.

### 3.2 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. 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 & van den Bosch, 1992, Quinlan, 1993; Hunt et al. 1966) (Equation 3). 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 et al. 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 (Quinlan, 1993). This bias can be rectified (as suggested by Quinlan) 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).

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

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

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

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

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

We have experimented with additional metrics for both *feature weighting* (e.g. value difference metrics which assign different differences between different pairs of values) and *exemplar weighting* (e.g. metrics assigning different weights to exemplars in memory depending on their success in classification, or based on some intrinsic property such as typicality of the exemplar for its task).

It is important to emphasize that the metrics used in Lazy learning should be *domain-independent*: they are not linguistically informed, and are applicable to domains as different as medical diagnosis and robotics. This is crucial, because otherwise, the acquisition bottleneck would move from the linguistic engineering of rules and knowledge bases to the tailoring of linguistically motivated metrics.

### 3.3 Asymptotic Complexity

Lazy 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. Hardware solutions to the complexity problem have been proposed: massively parallel computing (Stanfill & Waltz, 1986) or even wafer-scale integration (Kitano, 1993). For numeric features kd-trees have been proposed (Friedman et al., 1977) as a solution on single-processor machines. The advantages of this approach do not generalize easily to symbolic features, however. We developed IGTrees (Daelemans et al., 1996) to compress memory for symbolic features. IGTree is a heuristic approximation of the IB-IG algorithm. 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).

## 4 Lazy Learning of Language Processing Problems

In our research, we have applied this approach to a number of linguistic engineering problems: *hyphenation* (segment a word into syllables taking into account morphological structure), *grapheme-to-phoneme conversion* (identify the pronunciation of words), *stress assignment* (identify the stress pattern of words), *morphology* (both synthesis and analysis), and *tagging* (identify for each word in a text its morpho-syntactic category). See Daelemans (1995) for a discussion of the general approach, and van den Bosch & Daelemans, 1992, 1993; Daelemans & van den Bosch, 1992ab, 1993, 1994; van den Bosch et al. 1996; Daelemans et al. 1994, 1995, 1996abc for the details.

There are some general trends which become clear when analysing the results of all these experiments. First, the most striking result is that the accuracy of the induced systems is always comparable and often better than hand-crafted systems, at a fraction of the development effort and time. This proves the point that ML techniques may help considerably in solving knowledge acquisition bottlenecks in NLP. Second, when comparing different Lazy Learning variants (notably IB1, IB-IG, and IGTree) to other more eager learning approaches (TDIDT, Backprop learning), we find that IB-IG (the simplest Lazy Learning algorithm, extended with information-entropy-based feature weighting and a probabilistic decision rule) *always* obtains the best generalization accuracy. The picture is less clear for second place. Thirdly, in the same comparsion, there is a tendency that the more eager the technique is, the less accurate generalization becomes. More theoretical and empirical work is needed to explain and refine these results.

We will briefly describe the tagging application as an example of the approach, and because of its obvious practical applications (Daelemans et al. 1996c for details).

### 4.1 An Illustration: Part-of-Speech Tagging

The problem of POS tagging (morphosyntactic desambiguation) is the following: given a text, provide for each word in the text its contextually disambiguated part of speech (morphosyntactic category). I.e. transform a string of words into a string of tags. E.g., the sentence `John hit Pete .` should be mapped to `Noun Verb Noun Punc`. The target category inventory (tag set) may range from extremely simple (order 10) to extremely complex (order 1000). Tagging is a hard task because of the massive ambiguity in natural language text (ambiguity also depends of course on the tag-set used). E.g. in the example above, `hit` can be both a noun and a verb, context determines that in this case it is a verb. The correct category of a word depends on both its lexical probability $Pr(cat|word)$, and its contextual probability $Pr(cat|context)$.

A good tagger is instrumental in a large number of language engineering applications (ranging from text-to-speech over parsing to information retrieval). However, target tagset and training corpus differ from one application to the next, and making a tagger by hand is expensive and difficult. Therefore, robust, accurate taggers which can be automatically learned from example corpora are a commercially interesting product.

There are rule-based systems (hand made or using rule-induction), and statistical systems (using (hidden) markov modeling and dynamic programming). Although a thorough and reliable comparison of these approaches has not yet been achieved, it seems to be the case that all approaches converge to a 96-97% accuracy on new text from the same type as the training material. This may seem pretty good, but when looking at accuracy on sentences, this means only about 33% of sentences are correctly tagged completely.

The architecture of our lazy learning tagger 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* (associating words to possible tags as evidenced in the training corpus), a case base for *known words* (words occurring in the lexicon), and a case base for *unknown words*. Case Bases are compressed using IGTree for efficiency. 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 disambiguated using extrapolation from nearest neighbours 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 disambiguated using extrapolation from nearest neighbours in the unknown words case base. In each case, output is a best guess of the category for the word in its current context.

For known words, cases consist 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. For unknown words, a tag can be guessed only on the basis of the *form* or the *context* of the word. In our lazy learning approach, we provide word form 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.

For evaluation, 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. Generalization performance on known words (96.7%), unknown words (90.6%), and total (96.4%) is competitive with alternative hand-crafted and statistical approaches, and both training and testing speed are excellent (text tagging is possible with a speed of 200 words per second). In this case, the use of IGTrees as a heuristic approximation to IB-IG did not result in a loss of generalization accuracy while at the same time accounting for a spectacular decrease in memory and time consumption: IGTree retrieval is 100 to 200 times faster than IB-IG retrieval, and uses over 95% less memory. Due to its heuristic nature, however, accuracy is not guaranteed to be the same as IB-IG for any application.

# 5 Conclusions

We started with the observation that few of the inductive Machine Learning research projects address the both conceptually and commercially attractive problems of Natural Language Processing. This is a pity, because empirical learning methods have proven to be excellently applicable to the classification problems to which most of NLP can be reduced. Taking on NLP problems in ML may lead to different perspectives in ML theory and practice, however. Learning algorithms should be suited to learn from large amounts of data, and should be able to cope with the complex interactions of regularities, subregularities and exceptions prevalent in NLP tasks. Lazy Learning methods, when optimised for speed and storage requirements, seem to be a good choice.

More specifically, the following claims were made in this paper (sometimes implicitly). I list them here to facilitate discussion.

- The NLP knowledge acquisition bottlenecks can be solved with empirical ML methods.
- All interesting and useful problems in NLP can be re-formulated as classification problems (or cascades of classification problems).
- A propositional language seems to be expressive enough to describe the necessary input and output representations of NLP tasks.
- Lazy Learning works better than Eager Learning for NLP problems. Abstraction is harmful. Being there is better than being probable.
- Domain bias in defining similarity metrics for Lazy Learning is both unneccessary and unwanted.

# 6 References

Recent papers can be retrieved as postscript from
http://itkwww.kub.nl:2080/tki/Docs/Projects/Walter/pubs-long.html
or search Alta Vista with `walter daelemans recent`.

Aha, D. W., Kibler, D., & Albert, M. (1991). 'Instance-based learning algorithms'. *Machine Learning*, 7, 37–66.

van den Bosch, A. and W. Daelemans. (1992). Linguistic Pattern Matching Capabilities of Connectionist Networks. In: Jan van Eijck and Wilfried Meyer Viol (Eds.) *Computational Linguistics in the Netherlands. Papers from the Second CLIN-meeting 1991.* Utrecht: OTS, 40-53.

Van den Bosch, A. and Daelemans, W. (1993). 'Data-oriented methods for grapheme-to-phoneme conversion.' Proceedings of the Sixth conference of the European chapter of the ACL, ACL, 45–53.

Van den Bosch, A., W, Daelemans, T. Weijters. (1996). 'Morphological Analysis as Classification: an Inductive-Learning Approach.' *Proceedings of NEMLAP 1996*, Ankara, Turkey.

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.

Daelemans, W. (1995). 'Memory-based lexical acquisition and processing.' In Steffens, P., editor, *Machine Translation and the Lexicon*, Lecture Notes in Artificial Intelligence 898. Berlin: Springer, 85–98.

Daelemans, W., Van den Bosch, A. (1992a). 'Generalisation performance of backpropagation learning on a syllabification task.' In M. Drossaers & A. Nijholt (Eds.), *TWLT3: Connectionism and Natural Language Processing*. Enschede: Twente University, 27–38.

Daelemans, W. and A. van den Bosch. (1992b)'A Neural Network for Hyphenation.' In: I. Aleksander and J. Taylor (eds.) *Artificial Neural Networks II: Proceedings of the International Conference on Artificial Neural Networks.*, Elsevier Science Publishers, 1647-1650.

Daelemans, W. and A. van den Bosch. (1993). 'TABTALK: Reusability in Data-oriented grapheme-to-phoneme conversion.' *Proceedings of Eurospeech*, Berlin, 1459-1466.

Daelemans, W. and A. van den Bosch. (1994). 'A language-independent, data-oriented architecture for grapheme-to-phoneme conversion.' In: *Proceedings of the ESCA-IEEE conference on Speech Synthesis*, New York, 199-203.

Daelemans, W., S. Gillis and G. Durieux. (1994). 'The Acquisition of Stress, a data-oriented approach.' *Computational Linguistics* 20 (3), 421-451.

Daelemans, W., P. Berck, and S. Gillis. (1995). 'Linguistics as Data Mining: Dutch Diminutives', in Andernach, T., M. Moll, and A. Nijholt (eds). *CLIN V, Papers from the Fifth CLIN Meeting*, 59-72.

Daelemans, W., J. Zavrel, P. Berck, S. Gillis. (1996). 'MBT: A Memory-Based Part of Speech Tagger-Generator'. In: E. Ejerhed and I. Dagan (eds.) *Proceedings of the Fourth Workshop on Very Large Corpora*, Copenhagen, Denmark, 14-27.

Daelemans, W., P. Berck, S. Gillis. (1996).'Unsupervised Discovery of Phonological Categories through Supervised Learning of Morphological Rules.' *Proceedings of the 16th International Conference on Computational Linguistics* (COLING-96), Copenhagen, Denmark, 95-100.

Daelemans, W., Van den Bosch, A., Weijters, T. (1996). 'IGTree: Using Trees for Compression and Classification in Lazy Learning Algorithms.' In Aha, D. (ed.). *AI Review Special Issue on Lazy Learning*, forthcoming.

Derwing, B. L. and Skousen, R. (1989). 'Real Time Morphology: Symbolic Rules or Analogical Networks'. *Berkeley Linguistic Society* 15: 48–62.

Federici S. and V. Pirelli. (1996). 'Analogy, Computation and Linguistic Theory.' In Jones, D. (ed.) *New Methods in Language Processing.* London: UCL Press, forthcoming.

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

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

Jones, D. *Analogical Natural Language Processing.* (1996). London: UCL Press.

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

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

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

Salzberg, S. (1990) 'A nearest hyperrectangle learning method'. *Machine Learning* 6, 251–276.

Scha, R. (1992) 'Virtuele Grammatica's en Creatieve Algoritmen.' *Gramma/TTT* 1 (1), 57–77.

Skousen, R. (1989). *Analogical Modeling of Language.* Dordrecht: Kluwer.

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

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

Wettschereck, D., Aha, D.W. & Mohri, T. (1996). 'A review and comparative evaluation of feature weighting methods for lazy learning algorithms' Technical Report AIC-95-012. Washington, DC: Naval Research Laboratory, Navy Center for Applied Research in Artificial Intelligence.