Proceedings LREC workshop on adapting lexical and corpus resources to sublanguages and applications, Granada, 29-35, 1998

Toward Inductive Lexicons: a Case Study

Walter Daelemans[1,2], **Gert Durieux**[2], **Antal van den Bosch**[1]

[1] ILK Research Group
Computational Linguistics
Tilburg University
The Netherlands
{walter,antalb}@kub.nl

[2] CNTS
Linguistics
University of Antwerp
Belgium
durieux@uia.ua.ac.be

Abstract

Machine learning techniques can be used to make lexicons *adaptive*. The main problems in adaptation are the addition of lexical material to an existing lexical database, and the recomputation of sublanguage-dependent lexical information when porting the lexicon to a new domain or application. *Inductive lexicons* combine available lexical information and corpus data to alleviate these tasks. In this paper, we introduce the general methodology for the construction of inductive lexicons, and discuss empirical results on a case study using the approach: prediction of the gender of nouns in Dutch.

1. Introduction

In computational lexicography, lexicons of language engineering applications should come with acceptable lexical coverage, and with the information necessary for the intended applications. They should also come equipped with methods for the automatic extension and adaptation of the lexicon with new or modified lexical entries. Computational lexicology should therefore try to solve the following problem:

> All computational lexicons are inherently incomplete because of (i) missing lexical entries, and (ii) missing information about lexical entries.

On closer inspection, missing lexical entries are not really a problem: either we do not need them in a particular application or domain, and then we do not have to know that they exist, or we do need them, but then we will encounter at least some of their associated information (probably their spelling or pronunciation), and we will know some of the contexts they appear in. In that case, they are not missing, because this information is sufficient to construct a surprising amount of additional lexical information if we have some reasonably-sized corpora and lexical databases available, as we shall see, even if the latter only contain few lexical entries. The problem of missing lexical entries therefore reduces to the problem of extending existing lexical entries with additional information. Also the problem of the *adaptation* of lexical information to new domains reduces to problem (ii): lexical information has to be recomputed.

This paper addresses the *automatic extension and adaptation of lexicons using machine learning techniques*. We believe that machine learning techniques allow the accurate prediction of lexical information associated with new lexical items on the basis of extracted regularities from the lexical information already present in a computational lexicon in addition to corpus data. We propose a general methodology to lexical adaptation, and present a case study: the prediction of the gender of Dutch nouns on the basis of their spelling.

2. Memory-Based Learning

Although various machine learning paradigms are relevant to the lexical adaptation task, we have obtained particularly good results with Memory-based Learning (MBL). This approach to learning is founded on the hypothesis that performance in cognitive tasks (in our case NLP) is based on reasoning on the basis of similarity of new situations to *stored representations of earlier experiences*, rather than on the application of *mental rules* abstracted from earlier experiences (as in rule induction and rule-based processing). The approach has surfaced in different contexts using a variety of alternative names such as similarity-based, example-based, exemplar-based, analogical, case-based, instance-based, and lazy learning(Stanfill and Waltz, 1986; Cost and Salzberg, 1993; Kolodner, 1993; Aha, Kibler, and Albert, 1991; Aha, 1997). Historically, memory-based learning algorithms are descendants of the k-nearest neighbor (henceforth k-NN) algorithm (Cover and Hart, 1967; Devijver and Kittler, 1982; Aha, Kibler, and Albert, 1991).

A MBL system, visualized schematically in Figure 1, contains two components: a *learning component* which is memory-based (from which MBL borrows its name), and a *performance component* which is similarity-based. The learning component of MBL is memory-based as it involves adding training examples to memory; it is sometimes referred to as 'lazy' as memory storage is done without abstraction or restructuring. In the performance of an MBL system, the product of the learning component is used as a basis for mapping input to output; in the context of performing NLP tasks, this usually takes the form of performing classification. During classification, a previously unseen test example is presented to the system. Its similarity to all examples in memory is computed using a *similarity*

metric, and the category of the most similar example(s) is used as a basis for extrapolating the category of the test example.

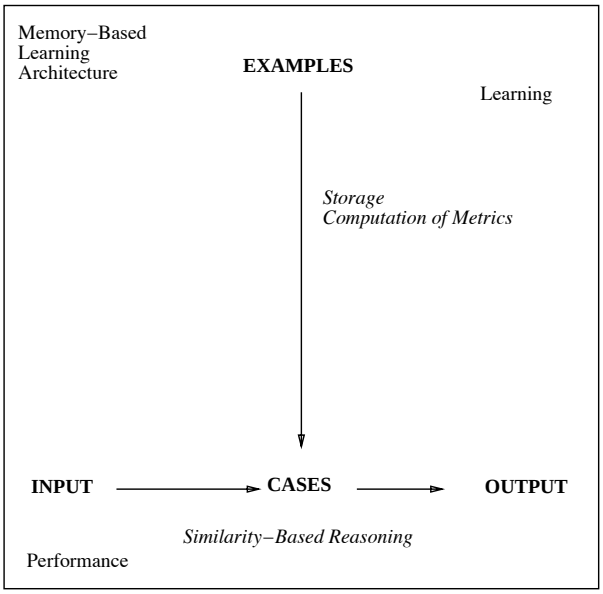

Figure 1: General architecture of an MBL system.

TiMBL[1] is the name of a software package developed by the ILK group at Tilburg University, containing variations of the memory-based learning algorithms IB1,IB1-IG, and MVDM (Aha, Kibler, and Albert, 1991; Daelemans and van den Bosch, 1992; Daelemans, Van den Bosch, and Weijters, 1997; Cost and Salzberg, 1993), and IGTREE (Daelemans, Van den Bosch, and Weijters, 1997), a decision-tree optimization of memory-based learning. Below, we outline the functioning of IB1-IG and IGTREE in Subsections and , respectively. We believe these algorithms to be especially suited for NLP problems in general and inductive lexicons in particular.

2.1. Weighted MBL in TIMBL: IB1-IG

IB1-IG (Daelemans and van den Bosch, 1992; Daelemans, Van den Bosch, and Weijters, 1997) is a memory-based learning algorithm that gathers a data base of instances (the *instance base* or case base) during learning. An instance consists of a fixed-length vector of n feature-value pairs, and an information field containing the classification of that particular feature-value vector. After the instance base is built, new (test) instances are classified by matching them to all instances in the instance base, and by calculating with each match the *distance* between the new instance X and the memory instance Y.

The most basic metric for patterns with symbolic features is the **Overlap metric** given in equations 1 and 2; where $\Delta(X, Y)$ is the distance between patterns X and Y, represented by n features, w_i is a weight for feature i, and δ is the distance per feature. The k-NN algorithm with this metric, and equal weighting for all features is called IB1 (Aha, Kibler, and Albert, 1991). Usually k is set to 1.

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

where:

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

We have made two additions to the original algorithm (Aha, Kibler, and Albert, 1991) in our version of IB1. First, in the case of nearest neighbor sets larger than one instance ($k > 1$ or ties), our version of IB1 selects the classification with the highest frequency in the class distribution of the nearest neighbor set. Second, if a tie cannot be resolved in this way because of equal frequency of classes among the nearest neighbors, the classification is selected with the highest overall occurrence in the training set.

The distance metric in equation 2 simply counts the number of (mis)matching feature values in both patterns. In the absence of information about feature relevance, this is a reasonable choice. Otherwise, we can add linguistic bias to weight or select different features (Cardie, 1996) or look at the behavior of features in the set of examples used for training. We can compute statistics about the relevance of features by looking at which features are good predictors of the class labels. Information Theory gives us a useful tool for measuring feature relevance in this way (Quinlan, 1986; Quinlan, 1993).

Information Gain (IG) weighting looks at each feature in isolation, and measures how much information it contributes to our knowledge of the correct class label. The Information Gain of feature f is measured by computing the difference in uncertainty (i.e. entropy) between the situations without and with knowledge of the value of that feature (Equation 3).

$$w_f = \frac{H(C) - \sum_{v \in V_f} P(v) \times H(C|v)}{si(f)} \tag{3}$$

$$si(f) = -\sum_{v \in V_f} P(v) \log_2 P(v) \tag{4}$$

Where C is the set of class labels, V_f is the set of values for feature f, and
$H(C) = -\sum_{c \in C} P(c) \log_2 P(c)$ is the entropy of the class labels. The probabilities are estimated from relative frequencies in the training set. The normalizing factor $si(f)$ (split info) is included to avoid a bias in favor of features with more values. It represents the amount of information needed to represent all values of the feature (Equation 4). The resulting IG values can then be used as weights in equation 1.

The possibility of automatically determining the relevance of features implies that many different and possibly irrelevant features can be added to the feature set. This is a very convenient methodology if theory does not constrain the choice enough beforehand, or if we wish to measure the importance of various information sources experimentally.

2.2. Optimized MBL in TiMBL: IGTREE

Using information gain rather than Overlap distance to

[1] The TiMBL software package is freely available for research purposes from the ILK web pages; consult URL http://ilk.kub.nl/.

define similarity in IB1 improves its performance on several NLP tasks (Daelemans and van den Bosch, 1992; Van den Bosch and Daelemans, 1993; Van den Bosch, 1997). The positive effect of information gain on performance prompted us to develop an alternative approach in which the instance memory is restructured in such a way that it contains the same information as before, but in a compressed decision tree structure. In this structure, instances are stored as paths of connected nodes and leaves contain classification information. Nodes are connected via arcs denoting feature values. Information gain is used to determine the order in which instance feature values are added as arcs to the tree. 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 optimized 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 tree 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. This implies that feature values that do not contribute to the disambiguation of the instance (i.e., the values of the features with lower information gain values than the lowest information gain value of the disambiguating features) are not stored in the tree.

Apart from compressing all training instances in the tree structure, the IGTREE algorithm also stores with each non-terminal node information concerning the *most probable* or *default* classification given the path thus far, according to the bookkeeping information maintained by the tree construction algorithm. This extra information is essential when processing unknown test instances. Processing an unknown input involves traversing the tree (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.

In sum, it can be said that in the trade-off between computation during learning and computation during classification, the IGTREE approach chooses to invest more time in organizing the instance base using information gain and compression, to obtain considerably simplified and faster processing during classification, as compared to IB1 and IB1-IG.

The generalization accuracy of IGTREE is comparable to that of IB1-IG; most of the time not significantly differing, and occasionally slightly (but statistically significantly) worse or even better. The two reasons for this surprisingly good accuracy are that (i) most 'unseen' instances contain parts that in fact fully match stored parts of training instances, and (ii) the probabilistic information stored at non-terminal nodes (i.e., the default classifications) still produce strong 'best guesses' when exact matching fails.

Because of the positive trade-off between storage and processing speed and accuracy, IGTREE is eminently suited as an implementation of the inductive lexicon concept, to which we turn now.

3. Inductive Lexicons

In its most general formulation, a computational lexicon is a set of lexical entries, and a lexical entry a set of predicates about some linguistic object. E.g. the lexical entry for a linguistic object labeled RED could be

pronunciation(RED)	/'rEd/
spelling(RED)	red
syncat(RED)	(ADJ N)

Lexical entries can correspond to various linguistic types of units: morphemes, base forms of words, word forms, idioms, phrases. The predicates can represent various types of linguistic knowledge: orthographic (spelling variants, hyphenation positions), phonetic/phonological (pronunciation representation, word stress pattern, syllable structure), morphological (morphological structure), syntactic (arguments, syntactic category, agreement features, even complete associated lexicalised syntactic trees as in Tree Adjoining Grammar), semantic/pragmatic (case frames, selection restrictions, semantic and pragmatic features). Lexical predicates may also refer to extra-linguistic knowledge (e.g. domain concepts). Rules for the derivation of lexical properties would normally be taken as part of the different linguistic domains they refer to, but in some lexicon architectures, these rules can belong conceptually to the lexicon as well.

The basic idea behind *inductive lexicons* is to use an available lexicon (however small), and, if available, a corpus, as a source to bootstrap lexical acquisition. Lexical predicates of newly encountered words are computed by reference to similar words previously encountered, for which the lexical information wanted is available. Depending on the lexical information to be predicted for the new word, different sources of information about the word are used as predictors.

Consider the following example. We have a small lexicon of word forms with their spelling, their pronunciation, and their possible syntactic categories. For each word in the lexicon we also have an index to positions in the corpus where that word occurs. Given a word for which no lexical information is available yet, we have its form (spelling) and its occurrences and context in the corpus as information. To compute lexical predicates for the new word, we can bootstrap from the available lexical information:

(i) to determine its possible syntactic categories: find known words which have a similar form (spelling, phonology) and a similar syntactic behavior (i.e. occur in similar syntactic contexts, given some operationalization of syntactic context), and extrapolate from their category,

(ii) to determine its pronunciation, extrapolate from known words in the lexicon with a spelling similar to the new word, to the pronunciation of that new word.

In this approach, therefore, an unknown target predicate of a lexical entry is predicted on the basis of known lexical predicates of the lexical entry itself, known target predicates and other predicates of other lexical entries, and (sometimes) also from corpus information (e.g. the contexts the lexical entry is found to occur in).

For each lexical predicate to be predicted (the target predicate), it is decided which sources of information (other lexical predicates or operationalisable corpus information) are relevant in its prediction. These sources of information are represented in terms of a feature-value vector. The next step is the construction of a *classifier* using e.g. MBL induction. The training material for this classifier is built from those lexical entries for which the target predicate is known. For each of these entries the input features and the associated output category (the target predicate) are collected, and this is used as training material for training the classifier. Inductive lexicons are neutral as far as lexical representation formalism is concerned. The only addition is the construction of a classifier for each lexical predicate (as far as it makes sense to try to predict that particular predicate). When using eager learning methods such as decision tree or rule induction, this classifier is an actual data structure; when using a lazy learning method (such as memory-based learning), the 'extracted' classifier is conceptual; the classification is done dynamically as needed from the lexical entries themselves, rather than from a data structure extracted from them. In sum, inductive lexicons fit a supervised learning paradigm, and can be either eager or lazy. In case of lazy learning, they are incremental, taking into account immediately any lexical entries added to the lexicon in predicting new lexical predicates, whereas eager learning methods call for explicit retraining when new lexical entries are added. However, incremental variants of decision tree learning also exist.

In the remainder of this section, we will illustrate the feasibility of this inductive lexicon architecture by means of a recent case study.

3.1. Gender Prediction

This case study will focus on a rather surprising use of phonological information in a syntactic problem domain: gender assignment in Dutch. It shows that information which at first sight should be added to lexicons by hand, can in some cases be usefully predicted from other lexical information (in this case phonology).

Genders form an important part of lexical structure and denote classes of nouns which can be distinguished syntactically by the agreements they take; agreeing elements (or *agreement targets*) are e.g. articles, demonstratives, adjectives or verbs. Under a sufficiently broad definition of agreement[2], control of anaphoric pronouns by their antecedent is covered as well, which is not without importance for Dutch. Historically, Dutch had a three-gender system, distinguishing the traditional categories of *masculine*, *feminine* and *neuter* (Dekeyser, 1980). Currently, the system is shifting towards a two-gender system, where the distinction between masculine and feminine is lost, and only the neuter/non-neuter opposition persists, as can be witnessed from Table 1.

Table 1: Agreement targets within singular NPs

	article	*demonstrative*		*adjective*
M	de	deze	die	-e
F	de	deze	die	-e
N	het	dit	dat	∅

Remnants of the three gender system, however, are still observed with pronominal anaphora, as Table 2 shows. Although in The Netherlands the masculine/feminine distinction is only preserved when the antecedents denote persons (male/female respectively)[3], in Flanders the opposition extends to non-human antecedents as well.

Table 2: Pronominal agreement targets (singular)

	personal	*possessive*	*relative*
M	hij	zijn	die
F	zij	haar	die
N	het	zijn	dat

Thus, gender identification, as exemplified by Dutch above, is ultimately a syntactic matter. Nevertheless, syntax may not always provide the necessary cues: consider e.g. a Natural Language Understanding system for Dutch, where the pronoun resolution component is faced with a feminine pronoun, while possible antecedents can only be diagnosed as non-neuter on the basis of agreement evidence. Clearly, proper assignment of the relevant items to their respective genders would be an important step towards disambiguation. Appropriate gender information in computational lexicons would therefore be an asset.

This problem of *gender assignment* is, of course, well-known and has traditionally been handled by the formulation of gender assignment rules (Corbett, 1991), which draw upon a number of different information sources: in *semantics-based* gender systems, meaning is sufficient to determine gender; here, oppositions such as animate/inanimate, human/non-human, etc. assign words to their respective genders. In *morphological* systems, word structure (both derivational and/or inflectional) is an important factor in gender assignment. In *phonological* systems, finally, the sound shape of a single wordform reliably indicates gender. The rule-based approach, however, is not without problems of its own. First, although all assignment systems are taken to have at least a semantic core, most languages employ

[2] E.g. "some systematic covariance between a semantic or formal property of one element and a formal property of another" (Steele, 1978).

[3] For non-human antecedents, the masculine forms are used.

different combinations of assignment criteria, which renders the identification of adequate rules difficult. Second, most assignment rules cover only specific portions of the lexicon, and complete coverage of the lexicon by the whole rule *set* is often not attained. Finally, varying numbers of exceptions exist, and having to list them separately begs the question of lexicon extension. For Dutch, a number of gender assignment rules have been formulated (Geerts et al., 1984), but none of them are entirely satisfactory. This has led some researchers to flatly deny the possibility of solving the gender assignment problem for Dutch: "The relationship between article and noun in Dutch is, except for a few exceptions, more or less arbitrary: the form the article takes is not systematically determined by any phonological, morphosyntactic, semantic, or conceptual features of the noun." (Deutsch and Wijnen, 1985).

To take up the challenge within the context of Inductive Lexicons, we conducted some exploratory experiments with MBL. The only assumptions made in constructing the classifier were that gender and phonological information are available (or can be obtained) for a sizeable part of the noun lexicon. Building on the observation that, crosslinguistically, there is often considerable overlap among various types of assignment criteria, the expectation was that —*pace* (Deutsch and Wijnen, 1985)—phonological information should make at least some headway in supplying gender information for unknown lemmata.

1. Data was extracted from the CELEX (Baayen, Piepenbrock, and Van Rijn, 1993) lexical database. Two series of 3 experiments were carried out, one for each relevant gender distinction. Experiments 1–3 involved 6090 noun lemmas; target categories were `M`(asculine), `F`(eminine), `N`(euter). Experiments 4–6 involved 7651 noun lemmas); here, target categories were `DE` and `HET`, for non-neuter and neuter resp. For each of the two series, the number of features was gradually increased over the three experiments: the simplest encoding (Experiments 1 and 4) only used onset, nucleus, and coda of the final syllable as features. For Experiments 2 and 5, onset, nucleus and coda of the initial syllable was added. Finally, for Experiments 3 and 6, the stress pattern and number of syllables were included as well, yielding a total of eight features per input example. An overview of the different encodings is given in Table 3.

Table 3: Encodings for Experiments 1–6

feature	*Exp1*	*Exp2*	*Exp3*	*Exp4*	*Exp5*	*Exp6*
Onset last	+	+	+	+	+	+
Nucleus last	+	+	+	+	+	+
Coda last	+	+	+	+	+	+
Onset first	−	+	+	−	+	+
Nucleus first	−	+	+	−	+	+
Coda first	−	+	+	−	+	+
Stress	−	−	+	−	−	+
Syllables	−	−	+	−	−	+

2. All tests were run with IB1-IG (Daelemans and Van den Bosch, 1992), basic memory-based learning augmented with *information gain* for feature weighting, using a single nearest neighbor. The test regime was *leave-one-out*. Results for the experiments are displayed in Tables 4 and 5[4]. From Table 4 it can be seen that the three-way gender distinction remains fairly well predictable, even though agreement marking for this distinction is disappearing from the language. The overall success rates are situated around 84%, which is significantly better than the claims of "arbitrariness of the Dutch gender system" would lead one to suspect. For the individual target categories, `F` is predicted best, with success scores around 90%, while the other two target categories reach scores of about 80%. Augmenting the number of features increases predictive accuracy.

Table 4: Success rates for Experiments 1–3

target	*Exp1*	*Exp2*	*Exp3*
`M`	79.99%	80.26%	81.54%
`F`	89.03%	88.91%	91.97%
`N`	81.58%	81.96%	80.75%
total	83.15%	83.35%	84.30%

Table 5: Success rates for Experiments 4–6

target	*Exp4*	*Exp5*	*Exp6*
`DE`	90.25%	90.49%	91.00%
`HET`	76.17%	77.26%	78.04%
total	86.37%	86.84%	87.65%

The results from Table 5 for the two-way distinction confirm the previous finding that augmenting the number of features yields higher success rates. Overall success rates are higher than for the previous experiment, with about 87% correct predictions; success rates for the individual target categories are comparable: around 90% for `DE` (non-neuter) and (slightly under) 80% for `HET` (neuter).

Even though these experiments were largely exploratory in nature, and little effort was made to maximize performance, the results indicate that an Inductive Lexicon approach to this problem is feasible. Whether these results are good enough to warrant practical application remains to be seen, although a glance at the confusion matrix for Experiment 3 (Table 6) might be instructive. Returning to our pronoun resolution problem from the introduction to this section, the main difficulty resided in the masculine/feminine distinction, for which agreement evidence within NPs is

[4] Exhaustive comparative experiments with IGTREE and C4.5 are subject of current work. Preliminary results indicate that IGTREE performs overall slightly worse than IB1-IG

lacking. It is precisely for this distinction that the classifier makes relatively few errors.

Table 6: Confusion matrix for Experiment 3

target	*predicted*		
	M	F	N
M	—	77	339
F	74	—	68
N	311	87	—

4. Conclusion

In this paper we introduced a machine learning solution to the problem that computational lexicons are never complete, and that to be useful, computational lexicons should have adaptive properties. Inductive Lexicons associate with each lexical predicate in the lexicon a classifier (a trained performance system) which makes it possible to compute this predicate for new lexical entries. Inductive Lexicons bootstrap on the knowledge implicit in the lexical entries already present in the lexicon (however small it may be), and if present, on information from corpora. We have shown on the basis of a case study that the approach is feasible.

We are well aware that a single successful case study does not prove the viability of an idea. However, in our research groups in Antwerp and Tilburg, several relevant lexical acquisition results have been obtained in different contexts with the same algorithms: assignment of syllable structure (Daelemans and van den Bosch, 1992), spelling-to-phonetics translation (Daelemans, Van den Bosch, and Weijters, 1997), word stress assignment (Daelemans, Gillis, and Durieux, 1994), various morphological problems (Daelemans, Berck, and Gillis, 1997; Van den Bosch, Daelemans, and Weijters, 1996), and as yet unpublished results for subcategorisation. Also in acquiring the interaction of contextual and lexical information, good results were obtained with part of speech tagging (Daelemans et al., 1996), PP-attachment (Zavrel, Daelemans, and Veenstra, 1997), word sense disambiguation, and chunking (partial parsing). Although many lexical features are much more structured and complex than the gender feature, the experiments mentioned in this paragraph show that by reformulating complex acquisition tasks as (cascades of) classification tasks, our machine learning methods are applicable to them.

5. Acknowledgements

This research was partially performed in the context of the "Induction of Linguistic Knowledge" research programme, partially supported by the Foundation for Language Speech and Logic (TSL), funded by the Netherlands Organization for Scientific Research (NWO). The research is also partially funded by a grant from the Flemish Government (Concerted Research Action University of Antwerp on *Computational Psycholinguistics*).

6. References

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

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

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

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

Corbett, G. 1991. *Gender*. Cambridge Textbooks in Linguistics. Cambridge University Press, Cambridge, UK.

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

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

Daelemans, W., S. Gillis, and G. Durieux. 1994. The acquisition of stress: a data-oriented approach. *Computational Linguistics*, 20(3):421–451.

Daelemans, W. and A. Van den Bosch. 1992. A neural network for hyphenation. In I. Aleksander and J. Taylor, editors, *Artificial Neural Networks 2*, volume 2, pages 1647–1650, Amsterdam. North-Holland.

Daelemans, W., A. Van den Bosch, and A. Weijters. 1997. IGTree: using trees for compression and classification in lazy learning algorithms. *Artificial Intelligence Review*, 11:407–423.

Daelemans, W., P. Berck, and S. Gillis. 1997. Data mining as a method for linguistic analysis: Dutch diminutives. *Folia Linguistica*, XXXI(1-2).

Daelemans, W. and A. van den Bosch. 1992. Generalisation performance of backpropagation learning on a syllabification task. In M. F. J. Drossaers and A. Nijholt, editors, *Proc. of TWLT3: Connectionism and Natural Language Processing*, pages 27–37, Enschede. Twente University.

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

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

Dekeyser, X. 1980. The diachrony of the gender systems in english and dutch. In J. Fisiak, editor, *Historical Morphology*, number 17 in Trends in Linguistics: Studies and Monographs. Mouton, The Hague, The Netherlands, pages 97–111.

Deutsch, W. and F. Wijnen. 1985. The article's noun and the noun's article: explorations into the representation and access of linguistic gender in dutch. *Linguistics*, 23:793–810.

Geerts, G., W. Haeserijn, J. de Rooij, and M. C. van den Toorn, editors. 1984. *Algemene Nederlandse Spraakkunst*. Wolters-Noordhof, Groningen, The Netherlands.

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

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

Quinlan, J. R. 1986. Induction of Decision Trees. *Machine Learning*, 1:81–206.

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

Steele, S. 1978. Word order variation: a typology study. In J. H. Greenberg, C. A. Ferguson, and E. A. Moravcsik, editors, *Universals of Human Language*, volume 4. Stanford University Press, Stanford, pages 585–623.

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

Van den Bosch, A. and W. Daelemans. 1993. Data-oriented methods for grapheme-to-phoneme conversion. In *Proceedings of the 6th Conference of the EACL*, pages 45–53.

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

Zavrel, J., W. Daelemans, and J. Veenstra. 1997. Resolving PP Attachment Ambiguities with Memory-Based Learning. In M. Ellison (Ed.), *Proceedings of the Workshop on Computational Natural Language Learning (CoNLL'97)*, Madrid, 1997, pages 136–144.