Learning to predict pitch accents and prosodic boundaries in Dutch

Erwin Marsi[1], **Martin Reynaert**[1], **Antal van den Bosch**[1],
Walter Daelemans[2], **Véronique Hoste**[2]

[1] Tilburg University
ILK / Computational Linguistics and AI
Tilburg, The Netherlands
{e.c.marsi,reynaert,
antal.vdnbosch}@uvt.nl

[2] University of Antwerp,
CNTS
Antwerp, Belgium
{daelem,hoste}@uia.ua.ac.be

Abstract

We train a decision tree inducer (CART) and a memory-based classifier (MBL) on predicting prosodic pitch accents and breaks in Dutch text, on the basis of shallow, easy-to-compute features. We train the algorithms on both tasks individually and on the two tasks simultaneously. The parameters of both algorithms and the selection of features are optimized per task with iterative deepening, an efficient wrapper procedure that uses progressive sampling of training data. Results show a consistent significant advantage of MBL over CART, and also indicate that task combination can be done at the cost of little generalization score loss. Tests on cross-validated data and on held-out data yield F-scores of MBL on accent placement of 84 and 87, respectively, and on breaks of 88 and 91, respectively. Accent placement is shown to outperform an informed baseline rule; reliably predicting breaks other than those already indicated by intra-sentential punctuation, however, appears to be more challenging.

1 Introduction

Any text-to-speech (TTS) system that aims at producing understandable and natural-sounding output needs to have on-board methods for predicting prosody. Most systems start with generating a prosodic representation at the linguistic or symbolic level, followed by the actual phonetic realization in terms of (primarily) pitch, pauses, and segmental durations. The first step involves placing pitch accents and inserting prosodic boundaries at the right locations (and may involve tune choice as well). Pitch accents correspond roughly to pitch movements that lend emphasis to certain words in an utterance. Prosodic breaks are audible interruptions in the flow of speech, typically realized by a combination of a pause, a boundary-marking pitch movement, and lengthening of the phrase-final segments. Errors at this level may impede the listener in the correct understanding of the spoken utterance (Cutler et al., 1997). Predicting prosody is known to be a hard problem that is thought to require information on syntactic boundaries, syntactic and semantic relations between constituents, discourse-level knowledge, and phonological well-formedness constraints (Hirschberg, 1993). However, producing all this information – using full parsing, including establishing semanto-syntactic relations, and full discourse analysis – is currently infeasible for a real-time system. Resolving this dilemma has been the topic of several studies in pitch accent placement (Hirschberg, 1993; Black, 1995; Pan and McKeown, 1999; Pan and Hirschberg, 2000; Marsi et al., 2002) and in prosodic boundary placement (Wang and Hirschberg, 1997; Taylor and Black, 1998). The commonly adopted solution is to use *shallow* information sources that approximate full syntactic, semantic and discourse information, such as the words of the text themselves, their part-of-speech tags, or their information content (in general, or in the text

Proceedings of the 41st Annual Meeting of the Association
for Computational Linguistics, July 2003, pp. 489-496.

at hand), since words with a high (semantic) information content or load tend to receive pitch accents (Ladd, 1996).

Within this research paradigm, we investigate pitch accent and prosodic boundary placement for Dutch, using an annotated corpus of newspaper text, and machine learning algorithms to produce classifiers for both tasks. We address two questions that have been left open thus far in previous work:

1. Is there an advantage in inducing decision trees for both tasks, or is it better to not abstract from individual instances and use a memory-based k-nearest neighbour classifier?
2. Is there an advantage in inducing classifiers for both tasks individually, or can both tasks be learned together.

The first question deals with a key difference between standard decision tree induction and memory-based classification: how to deal with exceptional instances. Decision trees, CART (Classification and Regression Tree) in particular (Breiman et al., 1984), have been among the first successful machine learning algorithms applied to predicting pitch accents and prosodic boundaries for TTS (Hirschberg, 1993; Wang and Hirschberg, 1997). Decision tree induction finds, through heuristics, a minimally-sized decision tree that is estimated to generalize well to unseen data. Its minimality strategy makes the algorithm reluctant to remember individual outlier instances that would take long paths in the tree: typically, these are discarded. This may work well when outliers do not reoccur, but as demonstrated by (Daelemans et al., 1999), exceptions do typically reoccur in language data. Hence, machine learning algorithms that retain a memory trace of individual instances, like memory-based learning algorithms based on the k-nearest neighbour classifier, outperform decision tree or rule inducers precisely for this reason.

Comparing the performance of machine learning algorithms is not straightforward, and deserves careful methodological consideration. For a fair comparison, both algorithms should be objectively and automatically optimized for the task to be learned. This point is made by (Daelemans and Hoste, 2002), who show that, for tasks such as word-sense disambiguation and part-of-speech tagging, tuning algorithms in terms of feature selection and classifier parameters gives rise to significant improvements in performance. In this paper, therefore, we optimize both CART and MBL individually and per task, using a heuristic optimization method called *iterative deepening*.

The second issue, that of task combination, stems from the intuition that the two tasks have a lot in common. For instance, (Hirschberg, 1993) reports that knowledge of the location of breaks facilitates accent placement. Although pitch accents and breaks do not consistently occur at the same positions, they are to some extent analogous to phrase chunks and head words in parsing: breaks mark boundaries of intonational phrases, in which typically at least one accent is placed. A learner may thus be able to learn both tasks at the same time.

Apart from the two issues raised, our work is also practically motivated. Our goal is a good algorithm for real-time TTS. This is reflected in the type of features that we use as input. These can be computed in real-time, and are language independent. We intend to show that this approach goes a long way towards generating high-quality prosody, casting doubt on the need for more expensive sentence and discourse analysis.

The remainder of this paper has the following structure. In Section 2 we define the task, describe the data, and the feature generation process which involves POS tagging, syntactic chunking, and computing several information-theoretic metrics. Furthermore, a brief overview is given of the algorithms we used (CART and MBL). Section 3 describes the experimental procedure (ten-fold iterative deepening) and the evaluation metrics (F-scores). Section 4 reports the results for predicting accents and major prosodic boundaries with both classifiers. It also reports their performance on held-out data and on two fully independent test sets. The final section offers some discussion and concluding remarks.

2 Task definition, data, and machine learners

To explore the generalization abilities of machine learning algorithms trained on placing pitch accents and breaks in Dutch text, we define three classification tasks:

Pitch accent placement – given a word form in its sentential context, decide whether it should be accented. This is a binary classification task.

Break insertion – given a word form in its sentential context, decide whether it should be followed by a boundary. This is a binary classification task.

Combined accent placement and break insertion – given a word form in its sentential context, decide whether it should be accented *and* whether it should be followed by a break. This is a four-class task: *no accent and no break*; *an accent and no break*; *no accent and a break*; *an accent and a break.*

Finer-grained classifications could be envisioned, e.g. predicting the *type* of pitch accent, but we assert that finer classification, apart from being arguably harder to annotate, could be deferred to later processing given an adequate level of precision and recall on the present task.

In the next subsections we describe which data we selected for annotation and how we annotated it with respect to pitch accents and prosodic breaks. We then describe the implementation of memory-based learning applied to the task.

2.1 Prosodic annotation of the data

The data used in our experiments consists of 201 articles from the ILK corpus (a large collection of Dutch newspaper text), totalling 4,493 sentences and 58,097 tokens (excluding punctuation). We set apart 10 articles, containing 2,905 tokens (excluding punctuation) as held-out data for testing purposes. As a preprocessing step, the data was tokenised by a rule-based Dutch tokeniser, splitting punctuation from words, and marking sentence endings.

The articles were then prosodically annotated, without overlap, by four different annotators, and were corrected in a second stage, again without overlap, by two corrector-annotators. The annotators' task was to indicate the locations of accents and/or breaks that they preferred. They used a custom annotation tool which provided feedback in the form of synthesized speech. In total, 23,488 accents were placed, which amounts to roughly one accent in two and a half words. 8627 breaks were marked; 4601 of these were sentence-internal breaks; the remainder consisted of breaks at the end of sentences.

2.2 Generating shallow features

The 201 prosodically-annotated articles were subsequently processed through the following 15 feature construction steps, each contributing one feature per word form token. An excerpt of the annotated data with all generated symbolic and numeric[1] features is presented in Table 1.

Word forms (Wrd) – The word form tokens form the central unit to which other features are added.

Pre- and post-punctuation – All punctuation marks in the data are transferred to two separate features: a pre-punctuation feature (PreP) for punctuation marks such as quotation marks appearing before the token, and a post-punctuation feature (PostP) for punctuation marks such as periods, commas, and question marks following the token.

Part-of-speech (POS) tagging – We used MBT version 1.0 (Daelemans et al., 1996) to develop a memory-based POS tagger trained on the Eindhoven corpus of written Dutch, which does not overlap with our base data. We split up the full POS tags into two features, the first (PosC) containing the main POS category, the second (PosF) the POS subfeatures.

Diacritical accent – Some tokens bear an orthographical diacritical accent put there by the author to particularly emphasize the token in question. These accents were stripped off the accented letter, and transferred to a binary feature (DiA).

NP and VP chunking (NpC & VpC) – An approximation of the syntactic structure is provided by simple noun phrase and verb phrase chunkers, which take word and POS information as input and are based on a small number of manually written regular expressions. Phrase boundaries are encoded per word using three tags: 'B' for chunk-initial words, 'I' for chunk-internal words, and 'O' for words outside chunks. The NPs are identified according to the base principle of one semantic head per chunk (non-recursive, base NPs). VPs include only verbs, not the verbal complements.

IC – Information content (IC) of a word w is given by $IC(w) = -log(P(w))$, where P(w) is esti-

[1] Numeric features were rounded off to two decimal points, where appropriate.

mated by the observed frequency of w in a large disjoint corpus of about 1.7 GB of unannotated Dutch text garnered from various sources. Word forms not in this corpus were given the highest IC score, i.e. the value for hapax legomenae (words that occur once).

Bigram IC – IC on bigrams (BIC) was calculated for the bigrams (pairs of words) in the data, according to the same formula and corpus material as for unigram IC.

TF*IDF – The TF*IDF metric (Salton, 1989) estimates the relevance of a word in a document. Document frequency counts for all token types were obtained from a subset of the same corpus as used for IC calculations. TF*IDF and IC (previous two features) have been succesfully tested as features for accent prediction by (Pan and McKeown, 1999), who assert that IC is a more powerful predictor than TF*IDF.

Phrasometer – The phrasometer feature (PM) is the summed log-likelihood of all n-grams the word form occurs in, with n ranging from 1 to 25, and computed in an iterative growth procedure: log-likelihoods of $n + 1$-grams were computed by expanding all stored n-grams one word to the left and to the right; only the $n + 1$-grams with higher log-likelihood than that of the original n-gram are stored. Computations are based on the complete ILK Corpus.

Distance to previous occurrence – The distance, counted in the number of tokens, to previous occurrence of a token within the same article (D2P). Unseen words were assigned the arbitrary high default distance of 9999.

Distance to sentence boundaries – Distance of the current token to the start of the sentence (D2S) and to the end of the sentence (D2E), both measured as a proportion of the total sentence length measured in tokens.

2.3 CART: Classification and regression trees

CART (Breiman et al., 1984) is a statistical method to induce a classification or regression tree from a given set of instances. 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. Each node in the CART tree contains a binary test on some categorical or numerical feature in the input vector. In the case of classification, the leaves contain the most likely class. The tree building algorithm starts by selecting the feature test that splits the data in such a way that the mean impurity (entropy times the number of instances) of the two partitions is minimal. The algorithm continues to split each partition recursively until some stop criterion is met (e.g. a minimal number of instances in the partition). Alternatively, a small stop value can be used to build a tree that is probably overfitted, but is then pruned back to where it best matches some amount of held-out data. In our experiments, we used the CART implementation that is part of the Edinburgh Speech Tools (Taylor et al., 1999).

2.4 Memory-based learning

Memory-based learning (MBL), also known as instance-based, example-based, or lazy learning (Stanfill and Waltz, 1986; Aha et al., 1991), is a supervised inductive learning algorithm for learning classification tasks. Memory-based learning treats a set of training instances as points in a multi-dimensional feature space, and stores them as such in an *instance base* in memory (rather than performing some abstraction over them). After the instance base is stored, new (test) instances are classified by matching them to all instances in memory, and by calculating with each match the *distance*, given by a distance function between the new instance X and the memory instance Y. Cf. (Daelemans et al., 2002) for details. Classification in memory-based learning is performed by the k-NN algorithm (Fix and Hodges, 1951; Cover and Hart, 1967) that searches for the k 'nearest neighbours' according to the distance function. The majority class of the k nearest neighbours then determines the class of the new case. In our k-NN implementation[2], equidistant neighbours are taken as belonging to the same k, so this implementation is effectively a k-nearest distance classifier.

3 Optimization by iterative deepening

Iterative deepening (ID) is a heuristic search algorithm for the optimization of algorithmic parameter

[2] All experiments with memory-based learning were performed with TiMBL, version 4.3 (Daelemans et al., 2002).

Wrd	PreP	PostP	PosC	PosF	DiA	NpC	VpC	IC	BIC	Tf*Idf	PM	D2P	D2S	D2E	A	B	AB
De	=	=	Art	bep,zijdofmv,neut	0	B	O	2.11	5.78	0.00	4	9999	0.00	0.94	-	-	- -
bomen	=	=	N	soort,mv,neut	0	I	O	4.37	7.38	0.16	4	17	0.06	0.89	A	-	A-
rondom	=	=	Prep	voor	0	O	O	4.58	5.09	0.04	4	17	0.11	0.83	-	-	- -
de	=	=	Art	bep,zijdofmv,neut	0	B	O	1.31	5.22	0.00	5	20	0.17	0.78	-	-	- -
molen	=	=	N	soort,ev,neut	0	I	O	5.00	7.50	0.18	5	9	0.22	0.72	A	-	A-
bij	=	=	Prep	voor	0	O	O	2.50	3.04	0.00	6	9999	0.28	0.67	-	-	- -
de	=	=	Art	bep,zijdofmv,neut	0	B	O	1.31	6.04	0.00	6	3	0.33	0.61	-	-	- -
scheepswerf	=	=	N	soort,ev,neut	0	I	O	5.63	8.02	0.03	4	9999	0.39	0.56	-	-	- -
Verolme	=	=	N	eigen,ev,neut	0	I	O	6.38	7.59	0.05	0	9999	0.44	0.50	A	-	A-
moeten	=	=	V	trans,ott,3,ev	0	B	O	2.99	6.77	0.01	4	9999	0.61	0.33	-	-	- -
verkassen	=	,	V	trans,inf	0	I	O	5.75	5.99	0.02	4	9999	0.67	0.28	A	B	AB
vindt	=	=	V	trans,ott,3,ev	0	O	B	3.51	8.50	0.00	6	9999	0.72	0.22	-	-	- -
molenaar	=	=	N	soort,ev,neut	0	B	O	5.95	8.50	0.05	0	9999	0.78	0.17	-	-	- -
Wijbrand	=	=	N	eigen,ev,neut	0	I	O	7.89	8.50	0.11	0	38	0.83	0.11	A	-	A-

Table 1: Symbolic and numerical features and class for the sentence *De bomen rondom de scheepswerf Verolme moeten verkassen, vindt molenaar Wijbrandt.* 'Miller Wijbrand thinks that the trees surrounding the mill near shipyard Verolme have to relocate.'

and feature selection, that combines classifier wrapping (using the training material internally to test experimental variants) (Kohavi and John, 1997) with progressive sampling of training material (Provost et al., 1999). We start with a large pool of experiments, each with a unique combination of input features and algorithmic parameter settings. In the first step, each attempted setting is applied to a small amount of training material and tested on a fixed amount of held-out data (which is a part of the full training set). Only the best settings are kept; all others are removed from the pool of competing settings. In subsequent iterations, this step is repeated, exponentially decreasing the number of settings in the pool, while at the same time exponentially increasing the amount of training material. The idea is that the increasing amount of time required for training is compensated by running fewer experiments, in effect keeping processing time approximately constant across iterations. This process terminates when only the single best experiment is left (or, the n best experiments).

This ID procedure can in fact be embedded in a standard 10-fold cross-validation procedure. In such a 10-fold CV ID experiment, the ID procedure is carried out on the 90% training partition, and the resulting optimal setting is tested on the remaining 10% test partition. The average score of the 10 optimized folds can then be considered, as that of a normal 10-fold CV experiment, to be a good estimation of the performance of a classifier optimized on the full data set.

For current purposes, our specific realization of this general procedure was as follows. We used folds of approximately equal size. Within each ID experiment, the amount of held-out data was approximately 5%; the initial amount of training data was 5% as well. Eight iterations were performed, during which the number of experiments was decreased, and the amount of training data was increased, so that in the end only the 3 best experiments used all available training data (i.e. the remaining 95%). Increasing the training data set was accomplished by random sampling from the total of training data available. Selection of the best experiments was based on their F-score (van Rijsbergen, 1979) on the target class (accent or break). F-score, the harmonic mean of precision and recall, is chosen since it directly evaluates the tasks (placement of accents or breaks), in contrast with classification accuracy (the percentage of correctly classified test instances) which is biased to the majority class (to place *no* accent or break). Moreover, accuracy masks relevant differences between certain inappropriate classifiers that do not place accents or breaks, and better classifiers that do place them, but partly erroneously.

The initial pool of experiments was created by systematically varying feature selection (the input features to the classifier) and the classifier settings (the parameters of the classifiers). We restricted these selections and settings within reasonable bounds to keep our experiments computationally feasible. In particular, feature selection was limited to varying the size of the window that was used to model the local context of an instance. A uniform window (i.e. the same size for all features) was applied to all features except DiA, D2P, D2S, and D2E. Its size (win) could be 1, 3, 5, 7, or 9, where

$win = 1$ implies no modeling of context, whereas $win = 9$ means that during classification not only the features of the current instance are taken into account, but also those of the preceding and following four instances.

For CART, we varied the following parameter values, resulting in a first ID step with 480 experiments:

- the minimum number of examples for leaf nodes (stop): 1, 10, 25, 50, and 100
- the number of partitions to split a float feature range into (frs): 2, 5, 10, and 25
- the percentage of training material held out for pruning (held-out): 0, 5, 10, 15, 20, and 25 (0 implies no pruning)

For MBL, we varied the following parameter values, which led to 1184 experiments in the first ID step:

- the number of nearest neighbours (k): 1, 4, 7, 10, 13, 16, 19, 22, 25, and 28
- the type of feature weighting: Gain Ratio (GR), and Shared Variance (SV)
- the feature value similarity metric: Overlap, or Modified Value Difference Metric (MVDM) with back-off to Overlap at value frequency tresholds 1 (L=1, no back-off), 2, and 10
- the type of distance weighting: None, Inverse Distance, Inverse Linear Distance, and Exponential Decay with $\alpha = 1.0$ (ED1) and $\alpha = 4.0$ (ED4)

4 Results

4.1 Tenfold iterative deepening results

We first determined two sharp, informed baselines; see Table 2. The informed baseline for accent placement is based on the content versus function word distinction, commonly employed in TTS systems (Taylor and Black, 1998). We refer to this baseline as CF-rule. It is constructed by accenting all content words, while leaving all function words (determiners, prepositions, conjunctions/complementisers and auxiliaries) unaccented. The required word class information is obtained from the POS tags. The baseline for break placement, henceforth PUNC-rule, relies solely on punctuation. A break is inserted after any sequence of punctuation symbols containing one or more characters from the set {,!?:;()}. It should be noted that both baselines are simple rule-based algorithms that have been manually optimized for the current training set. They perform well above chance level, and pose a serious challenge to any ML approach.

$Target:$	$Method:$	$Prec:$	$Rec:$	$F:$
Accent	CF-rule	66.7	94.9	78.3
	CART	78.6 $\pm$2.8	85.7 $\pm$1.1	82.0 $\pm$1.7
	MBL	80.0 $\pm$2.7	86.6 $\pm$1.4	**83.6** $\pm$1.6*
	CART^C	78.7 $\pm$3.0	85.6 $\pm$0.8	82.0 $\pm$1.6
	MBL^C	81.0 $\pm$2.7	86.1 $\pm$1.1	83.4 $\pm$1.5*
Break	PUNC-rule	99.2	75.7	85.9
	CART	93.1 $\pm$1.5	82.2 $\pm$3.0	87.3 $\pm$1.5
	MBL	95.1 $\pm$1.4	81.9 $\pm$2.8	**88.0** $\pm$1.5*
	CART^C	94.5 $\pm$0.8	80.2 $\pm$3.1	86.7 $\pm$1.6
	MBL^C	95.7 $\pm$1.1	80.7 $\pm$3.1	87.6 $\pm$1.7*

Table 2: Precision, recall, and F-scores on accent, break and combined prediction by means of CART and MBL, for baselines and for average results over 10 folds of the Iterative Deepening experiment; a * indicates a significant difference ($p < 0.01$) between CART and MBL according to a paired t-test. Superscript ^{C} refers to the combined task.

From the results displayed in Table 2, the following can be concluded. First, MBL attains the highest F-scores on accent placement, 83.6, and break placement, 88.0. It does so when trained on the ACCENT and BREAK tasks individually. On these tasks, MBL performs significantly better than CART (paired t-tests yield $p < 0.01$ for both differences).

Second, the performances of MBL and CART on the *combined* task, when split in F-scores on accent and break placement, are rather close to those on the *accent* and *break* tasks. For both MBL and CART, the scores on accent placement as part of the combined task versus accent placement in isolation are not significantly different. For break insertion, however, a small but significant drop in performance can be seen with MBL ($p < 0.05$) and CART ($p < 0.01$) when it is performed as part of the COMBINED task.

As is to be expected, the optimal feature selections and classifier settings obtained by iterative deepening turned out to vary over the ten folds for both MBL and CART. Table 3 lists the settings producing the best F-score on accents or breaks. A window of 7 (i.e. the features of the three preceding and following word form tokens) is used by CART and MBL for accent placement, and also for break insertion by CART, whereas MBL uses a window of

Target:	*Method:*	*Setting:*
Accent	CART	win=7, stop=50, frs=5, held-out=5
	MBL	win=7, MVDM with L=5, k=25, GR, ED4
Break	CART	win=7, stop=25, frs=2, held-out=5
	MBL	win=3, MVDM with L=2, k=28, GR, ED4

Table 3: Optimal parameter settings for CART and MBL with respect to accent and break prediction

just 3. Both algorithms ($stop$ in CART, and k in MBL) base classifications on minimally around 25 instances. Furthermore, MBL uses the Gain Ratio feature weighting and Exponential Decay distance weighting. Although no pruning was part of the Iterative Deepening experiment, CART prefers to hold out 5% of its training material to prune the decision tree resulting from the remaining 95%.

4.2 External validation

We tested our optimized approach on our held-out data of 10 articles (2,905 tokens), and on an independent test corpus (van Herwijnen and Terken, 2001). The latter contains two types of text: 2 newspaper texts (55 sentences, 786 words excluding punctuation), and 17 email messages (70 sentences, 1133 words excluding punctuation). This material was annotated by 10 experts, who were asked to indicate the preferred accents and breaks. For the purpose of evaluation, words were assumed to be accented if they received an accent by at least 7 of the annotators. Furthermore, of the original four break levels annotated (i.e. no break, light, medium, or heavy), only medium and heavy level breaks were considered to be a break in our evaluation. Table 4 lists the precision, recall, and F-scores obtained on the two tasks using the single-best scoring setting from the 10-fold CV ID experiment per task. It can be seen that both CART and MBL outperformed the CF-rule baseline on our own held-out data and on the news and email texts, with similar margins as observed in our 10-fold CV ID experiment. MBL attains an F-score of 86.6 on accents, and 91.0 on breaks; both are improvements over the cross-validation estimations. On breaks, however, both CART and MBL failed to improve on the PUNC-rule baseline; on the news and email texts they perform even worse. Inspecting MBLs output on these text, it turned out that MBL does emulate the PUNC-rule baseline, but that it places additional breaks at positions not marked by punctuation. A considerable portion of these non-punctuation breaks is placed incorrectly – or at least different from what the annotators preferred – resulting in a lower precision that does not outweigh the higher recall.

$Target:$	$Test\ set$	$Method:$	$Prec:$	$Rec:$	$F:$
Accent	Held-out	CF-rule	73.5	94.8	82.8
		CART	84.3	86.1	85.2
		MBL	87.0	86.3	86.6
	News	CF-rule	52.2	92.9	66.9
		CART	62.7	92.5	74.6
		MBL	66.3	89.2	76.0
	Email	CF-rule	54.3	91.0	68.0
		CART	66.8	88.5	76.1
		MBL	71.0	88.5	78.8
Break	Held-out	PUNC-rule	99.5	83.7	90.9
		CART	92.6	88.9	90.7
		MBL	95.5	87.0	91.0
	News	PUNC-rule	98.8	93.1	95.9
		CART	80.6	95.4	87.4
		MBL	89.3	95.4	92.2
	Email	PUNC-rule	93.9	87.0	90.3
		CART	81.6	90.2	85.7
		MBL	83.0	91.1	86.8

Table 4: Precision, recall, and F-scores on accent and break prediction for our held-out corpus and two external corpora of news and email texts, using the best settings for CART and MBL as determined by the ID experiments.

5 Conclusion

With shallow features as input, we trained machine learning algorithms on predicting the placement of pitch accents and prosodic breaks in Dutch text, a desirable function for a TTS system to produce synthetic speech with good prosody. Both algorithms, the memory-based classifier MBL and decision tree inducer CART, were automatically optimized by an Iterative Deepening procedure, a classifier wrapper technique with progressive sampling of training data. It was shown that MBL significantly outperforms CART on both tasks, as well as on the combined task (predicting accents and breaks simultaneously). This again provides an indication that it is advantageous to retain individual instances in memory (MBL) rather than to discard outlier cases as noise (CART).

Training on both tasks simultaneously, in one model rather than divided over two, results in generalization accuracies similar to that of the individually-learned models (identical on accent placement, and slightly lower for break placement).

This shows that learning one task does not seriously hinder learning the other. From a practical point of view, it means that a TTS developer can resort to one system for both tasks instead of two.

Pitch accent placement can be learned from shallow input features with fair accuracy. Break insertion seems a harder task, certainly in view of the informed punctuation baseline PUNC-rule. Especially the precision of the insertion of breaks at other points than those already indicated by commas and other 'pseudo-prosodic' orthographic mark up is hard. This may be due to the lack of crucial information in the shallow features, to inherent limitations of the ML algorithms, but may as well point to a certain amount of optionality or personal preference, which puts an upper bound on what can be achieved in break prediction (Koehn et al., 2000).

We plan to integrate the placement of pitch accents and breaks in a TTS system for Dutch, which will enable the closed-loop annotation of more data using the TTS itself and on-line (active) learning. Moreover, we plan to investigate the perceptual cost of false insertions and deletions of accents and breaks in experiments with human listeners.

Acknowledgements

Our thanks go out to Olga van Herwijnen and Jacques Terken for the use of their TTS evaluation corpus. All research in this paper was funded by the Flemish-Dutch Committee (VNC) of the National Foundations for Research in the Netherlands (NWO) and Belgium (FWO).

References

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

A.W. Black. 1995. Comparison of algorithms for predicting pitch accent placement in English speech synthesis. In *Proceedings of the Spring Meeting of the Acoustical Society of Japan*.

L. Breiman, J. Friedman, R. Ohlsen, and C. Stone. 1984. *Classification and regression trees*. Wadsworth International Group, Belmont, CA.

C.J. van Rijsbergen. 1979. *Information Retrieval*. Buttersworth, London.

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

A. Cutler, D. Dahan, and W.A. Van Donselaar. 1997. Prosody in the comprehension of spoken language: A literature review. *Language and Speech*, 40(2):141–202.

W. Daelemans and V. Hoste. 2002. Evaluation of machine learning methods for natural language processing tasks. In *Proceedings of LREC-2002, the third International Conference on Language Resources and Evaluation*, pages 755–760.

W. Daelemans, 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.

W. Daelemans, A. van den Bosch, and J. Zavrel. 1999. Forgetting exceptions is harmful in language learning. *Machine Learning, Special issue on Natural Language Learning*, 34:11–41.

W. Daelemans, J. Zavrel, K. van der Sloot, and A. van den Bosch. 2002. TiMBL: Tilburg Memory Based Learner, version 4.3, reference guide. Technical Report ILK-0210, ILK, Tilburg University.

E. Fix and J. L. Hodges. 1951. Discriminatory analysis—nonparametric discrimination; consistency properties. Technical Report Project 21-49-004, Report No. 4, USAF School of Aviation Medicine.

J. Hirschberg. 1993. Pitch accent in context: Predicting intonational prominence from text. *Artificial Intelligence*, 63:305–340.

P. Koehn, S. Abney, J. Hirschberg, and M. Collins. 2000. Improving intonational phrasing with syntactic information. In *ICASSP*, pages 1289–1290.

R. Kohavi and G. John. 1997. Wrappers for feature subset selection. *Artificial Intelligence Journal*, 97(1–2):273–324.

D. R. Ladd. 1996. *Intonational phonology*. Cambridge University Press.

E. Marsi, G.J. Busser, W. Daelemans, V. Hoste, M. Reynaert, and A. van den Bosch. 2002. Combining information sources for memory-based pitch accent placement. In *Proceedings of the International Conference on Spoken Language Processing, ICSLP-2002*, pages 1273–1276.

S. Pan and J. Hirschberg. 2000. Modeling local context for pitch accent prediction. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics*, Hong Kong.

S. Pan and K. McKeown. 1999. Word informativeness and automatic pitch accent modeling. In *Proceedings of EMNLP/VLC'99*, New Brunswick, NJ, USA. ACL.

F. Provost, D. Jensen, and T. Oates. 1999. Efficient progressive sampling. In *Proceedings of the Fifth International Conference on Knowledge Discovery and Data Mining*, pages 23–32.

G. Salton. 1989. *Automatic text processing: The transformation, analysis, and retrieval of information by computer*. Addison–Wesley, Reading, MA, USA.

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

P. Taylor and A. Black. 1998. Assigning phrase breaks from part-of-speech sequences. *Computer Speech and Language*, 12:99–117.

P. Taylor, R. Caley, A. W. Black, and S. King, 1999. *Edinburgh Speech Tools Library, System Documentation Edition 1.2*. CSTR, University of Edinburgh.

O. van Herwijnen and J. Terken. 2001. Evaluation of pros-3 for the assignment of prosodic structure, compared to assignment by human experts. In *Proceedings Eurospeech 2001 Scandinavia, Vol.1*, pages 529–532.

M. Q. Wang and J. Hirschberg. 1997. Automatic classification of intonational phrasing boundaries. *Computer Speech and Language*, 6(2):175–196.