COMBINING INFORMATION SOURCES FOR MEMORY-BASED PITCH ACCENT PLACEMENT

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

[1] Tilburg University
ILK / Computational Linguistics and AI
Tilburg, The Netherlands

[2] University of Antwerp,
CNTS
Antwerp, Belgium

ABSTRACT

We describe results on pitch accent placement in Dutch text obtained with a memory-based learning approach. The training material consists of newspaper texts that have been prosodically annotated by humans, and subsequently enriched with linguistic features and informational metrics using generally available, low-cost, shallow, knowledge-poor tools. We report on the effects of context-modelling and the nearest neighbours parameter (k), and show the advantage of combining features of a different nature, where the best performance yields a cross-validated F-score of 82. Evaluation on an independent test corpus shows that our approach outperforms existing TTS systems for Dutch.

1. INTRODUCTION

Better prosody, both at the symbolic and the phonetic level, is generally thought to have a significant impact on the quality of synthetic speech. In this paper we focus on one crucial aspect of prosody: pitch accent placement. The placement of pitch accent on certain stressable words in a sentence has been shown to be related to various types of information, including phonological well-formedness constraints, lexical properties, syntactic constituent boundaries, syntactic and semantic relations, and discourse-level knowledge [1]. Some of these are hard, if not impossible, to obtain automatically from unrestricted text by current NLP standards. In the research described here, we investigate whether linguistic features and informational metrics which can be computed using generally available, low-cost, shallow, knowledge-poor tools are suitable predictive features for pitch accent prediction when used with machine learning techniques. More specifically, a memory-based learner is trained to predict whether words should be accented or not. Input consists of the current word, some of the preceding and following words, as well as lexical, syntactic, and information-theoretic features associated with these words. We intend to show that this approach goes a long way towards generating naturally-sounding pitch accent patterns, casting doubt on the need for more expensive sentence and discourse analysis.

In Section 2 we define the task, describe the data we used, and the annotation process which involves lemmatising, tagging, chunking, and computing informational metrics. Furthermore, a brief overview is given of the memory-based language processing algorithms we used in all experiments. Section 3 reports on systematic experiments with single features and combinations of features as input, for various settings of window size and k, an important parameter of the learning algorithm. Section 4 then reports on the performance of our approach on an independent test corpus which was designed to evaluate the pitch accent component of Dutch TTS systems, showing that our approach compares favourably to state-of-the-art systems. The final section offers concluding remarks.

2. TASK DEFINITION, DATA, AND MACHINE LEARNING METHOD

We define pitch accent placement as a binary task: given a word form in its sentential context, decide whether it should receive a pitch accent or not. This implies three types of abstractions. First, our scope is limited to predicting pitch accents at the symbolic level. This is in agreement with the division of labour between linguistic processing and phonetic implementation found in virtually all speech synthesis systems. That is, generating actual F_0 values presupposes symbolic input, but is a problem of a different nature that should be solved with other methods.

Second, the task is defined at its most coarse-grained level. Finer-grained classifications, such as predicting the *type* of pitch accent (e.g. H*L or L*H), could be envisioned. However, we assert that the latter classification, apart from being arguably harder to annotate, could be deferred to later processing given an adequate level of performance on the binary task.

Third, it is well known that accent placement allows for a considerable amount of variation, reflecting personal style and taste. Nevertheless, on many occasions presence or lack of an accent turns out to be crucial for the interpretation and naturalness of the speech. Until evaluation metrics have been developed that take the difference between essential and merely optional accents into account, we measure performance by matching predicted accents with those specified by a human annotator in a straightforward way.

2.1. Prosodic annotation of the data

The data used in our experiments contains 128 Dutch newspaper articles, totalling 2,896 sentences and 42,912 tokens (including punctuation). As a preprocessing step, the data was tokenised by a rule-based Dutch tokeniser, splitting punctuation from words, and marking sentence endings. Tokenisation errors were manually corrected.

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 mark individual tokens as bearing accent. They used a custom annotation tool which provided feedback in the form of synthesized speech. In total, 14,457 accents were placed (approximately one in three words).

2.2. Adding shallow information sources

The 128 prosodically-annotated articles were subsequently processed through the following seven shallow analysis steps, each contributing one annotation symbol per token:

POS tagging – We used a fast, accurate multi-tagger ensemble [2], which reached an accuracy of 93.4% (using manually corrected tags on our data as the reference). Its learning material did not overlap with our base data.

Lemmatising – An automatic, memory-based lemmatiser was used to lemmatise all non-punctuation tokens (based on [3]). It was trained on a large Dutch morpho-lexical database. In ambiguous situations (e.g. in cases where a word form can be a noun or a verb), POS tags are used for disambiguation.

NP and VP chunking – Syntactic structure is provided by simple noun phrase and verb phrase chunkers which take word and POS information as input. They are implemented as finite-state machines compiled from a small number of regular expressions. Boundaries of the phrases 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 principle of one semantic head per chunk (non-recursive, base NPs). The VPs include only the words that are verbs, not the verbal complements.

IC – Information Content of a word w is given by $IC(w) = -log(P(w))$, where P(w) is estimated by the observed frequency of w. Using log base 10, we calculated IC on the basis of a large and disjoint corpus of about 620 Mb of newspaper text. Hence, no IC value is available for words that have not yet been not encountered. These were given the highest IC score observed, i.e. the value for hapaxes in the larger corpus.

TF*IDF – The TF*IDF metric [4] estimates the relevance of a word in a document. Document frequency counts for all token types were obtained from the same corpus as used for IC calculations.

Distance – For each token, the distance, in number of tokens, to its most recent occurrence within the same article was counted. Words occurring for the first time in an article were assigned the arbitrary high default distance of 9999.

Together with the original tokens, this shallow processing yields eight sources of information to be used as predictive features in the learning task. An excerpt of the annotated data is presented in Table 1.

2.3. Memory-based learning

Memory-based learning, also known as instance-based, example-based, or lazy learning [5, 6], is a supervised inductive learning algorithm for learning classification tasks. Memory-based learning treats a set of labeled (pre-classified) 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).

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 stored, new (test) instances are classified by matching them to all instances in the instance base, and by calculating with each match the *distance*, given by a distance function $\Delta(X, Y)$ between the new instance X and the memory instance Y. In our experiments we used the "overlap" distance function $\Delta(X, Y) = \sum_{i=1}^{n} w_i \, \delta(x_i, y_i)$, where n is the number of features, w_i is a weight for feature i, and $\delta = 0 \; if \; x_i = y_i, \; else \; 1$ is the distance per feature. For numeric features, the distance equals their absolute difference normalised to 1. Classification in memory-based learning is performed by the k-NN algorithm that searches for the k 'nearest neighbours' according to the $\Delta(X, Y)$ function. The majority class of the k nearest neighbours then determines the class of the new case. With symbolic feature values, distance ties can occur when two nearest neighbours mismatch with the test instance on the same feature value, while all three instances have different values. In the k-NN implementation[1] we used, equidistant neighbours are taken as belonging to the same k, so this implementation is effectively a k-nearest distance classifier.

The weight (importance) of a feature i, w_i, is estimated by computing its *gain ratio* GR_i. To compute a feature's GR, we first compute its *information gain* IG_i, which is the difference in uncertainty (entropy) within the set of cases between the situations without and with knowledge of the value of that feature: $IG_i = H(C) - \sum_{v \in V_i} P(v) \times H(C|v)$, where C is the set of class labels, V_i is the set of values for feature i, 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 frequency counts in the training set. To derive the GR, the feature's IG is divided by the entropy of the feature values, the *split info* $si_i = -\sum_{v \in V_i} P(v) \log_2 P(v)$: $GR_i = \frac{IG_i}{si_i}$.

The strength of memory-based language processing in the task at hand is that it provides, through its feature weighting method, a natural way of combining the shallow information sources in an optimal way. In addition, it performs no abstraction (in contrast to most other machine learning algorithms), which allows it to deal with productive but low-frequency exceptions [6].

3. EXPERIMENTS

3.1. Baselines

In order to evaluate the performance of our approach, we determined two baselines for accent placement:

1. RND: a chance-level baseline formed by randomly assigning n accents, where n equals the number of accents assigned in our corpus.
2. C/F: a content vs. function word baseline formed by accenting all content words, while leaving all function words (determiners, prepositions, conjunctions, complementisers and auxiliaries) unaccented.

Table 2 shows the performance of both baselines in terms of accuracy, precision, recall, and $F_{\beta=1}$ with respect to the class 'accent'. $F_{\beta=1}$ represents a harmonic average between precision and recall. Since these are considered to be of equal importance to the present task, we assume that $\beta = 1$, and henceforth refer to $F_{\beta=1}$ as F.

[1] All experiments with memory-based learning were performed with TiMBL, version 4 [7].

Table 1. Annotation for the sentence *Zo ontstond deze bijlage.* ('That gave rise to this supplement.'), where the first eight columns show the different information sources which are used for accent placement (last column)

WORD	POS	LEMMA	NP	VP	IC	TF*IDF	DIST	ACCENT
Zo	BW()	zo	O-NP	O-VP	3.47	0.002967	9999	–
ontstond	WW(pv,verl,ev)	ontstaan	O-NP	B-VP	4.44	0.006178	9999	–
deze	VNW(aanw,det,stan, prenom,met-e,rest)	deze	B-NP	O-VP	2.86	0.001357	9999	A
bijlage	N(soort,ev,basis,zijd, stan)	bijlage	I-NP	O-VP	5.78	0.011120	9999	A
.	LET(.)	.	O-NP	O-VP	1.29	0.000090	10	–

Table 2. Performance of the baselines RND (randomly) and C/F (content vs. function word based) as a percentage of accuracy, precision, recall, and F

	Accuracy	*Precision*	*Recall*	F
RND	58.6	38.6	38.6	38.6
C/F	79.7	63.4	94.1	75.8

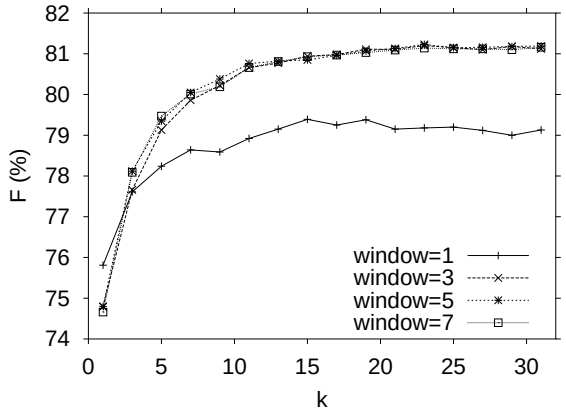

Fig. 1. Average F as a function of k for 4 different window sizes in CV experiments with both POS and IC.

3.2. Relevance of context and k

In a first set of experiments, we investigated the effect of context size and k. These and all other reported experiments were carried out using 10-fold cross-validation. Folds were created containing an approximately equal number of instances, but without splitting any of the newspaper articles. For the memory-based learner, we used its default setting and numeric distance calculations for the numeric features IC, TF*IDF and DISTANCE. Here, we took the features POS and IC, as these received the highest Gain Ratio values (cf. Section 2.3). Context was modelled by applying a window to these features, that is, by adding features of the preceding and following tokens to the set of input features for the current token. We tried window sizes of 7 (3 left, 3 right), 5, 3, and 1 (no context). In addition, the parameter k, which stands for the number of nearest neighbours that the memory-based learner takes into account during classification, was varied in the range of uneven numbers from 1 to 31. Figure 1 shows the performance as a function of k for different window sizes. The maximal accuracy attained is 86.7 ± 0.8 (81.3 ± 1.2 F), which is significantly higher than both baselines.

From these results we conclude that taking context into account improves performance, although window sizes larger than three appear to have no extra effect. This indicates that more context than only very local (one word to the left, one to the right of the focus word) does not provide more disambiguating information. Moreover, context modelling appears to become effective only when at least five nearest neighbours are taken into account. Apparently, the majority class among small numbers of nearest neighbours is unreliable, suggesting that many nearest neighbour pairs have different classifications. The reliability of the majority class stabilises only after about eleven or more neighbours.

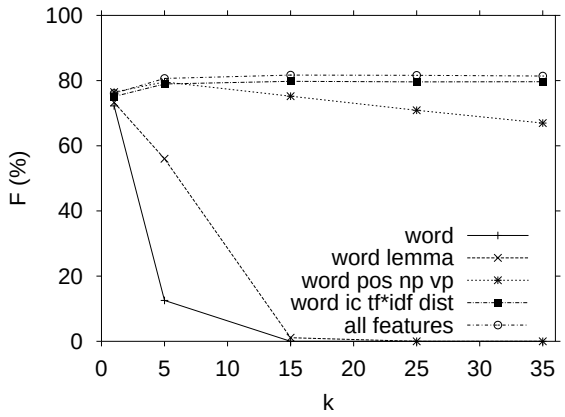

Fig. 2. Average F in 10-fold CV experiments with $window = 3$ on all features, as a function of k, for 5 different feature combinations

3.3. Feature combinations

In a subsequent set of experiments, we explored additional combinations of features. The WORD feature, which is the most basic feature freely available to any accent placement algorithm, was combined with a lexical feature (LEMMA), with syntactic features (POS, NP, VP), with informational features (IC, TF*IDF, DISTANCE), and finally with all features available. On the basis of the results in the previous section, we chose a window size of three, but varied settings of parameter k.

Figure 2 plots the performance in F as a function of k for each of the feature combinations. WORD with or without LEMMA drops to an F of 0.0 at higher k values. This is because nearest neighbour sets at these levels of k (yielding k nearest distances of many more than k instances) always contain a majority of unaccented instances, which means that no accents at all are assigned. This effect is slightly counteracted when WORD is coupled with the syntactic features POS, NP, and VP, but after a peak at $k = 5$, the same majority effect decreases performance. In contrast, when

Table 3. Comparison of three TTS systems for Dutch (FD, KIK, and RS) with our approach (MBL) w.r.t. accent placement in news text and email as a percentage of accuracy, recall, precision, and F

	News				*Email*			
System	*Acc*	*Prec*	*Rec*	F	*Acc*	*Prec*	*Rec*	F
FD	76	58	84	68	81	66	80	72
KIK	75	57	80	67	77	62	73	67
RS	74	55	84	66	75	57	88	69
MBL	86	69	88	**77**	86	72	82	**77**

WORD is coupled with the numeric informational features IC, TF*IDF, and DISTANCE, results stabilise at high levels for $k \geq 5$. Best performance is obtained by combining all features. The best result (using all features, a window size of three, and $k = 15$) is an F of 81.7 ± 1.4 (87.2 ± 1.0 accuracy) with a precision of 78.9 ± 1.9 and a recall of 84.7 ± 1.4. Although the differences between the experiment that combines all features versus the experiment with WORD plus the numeric informational features seem small, they are significant at all values of k with at least $p < 0.01$ according to one-tailed t tests.

Interestingly, the precision of the experiment using all features, 78.9, is higher than the best precision obtained in the three other experiments, 77.6, with WORD and the numeric informational features; moreover, its recall of 84.7 is higher than the best recall of the other experiments, 84.3, with WORD and the syntactic features. Thus, integrating all features not only combines the best of both the syntactic and informational features, but improves on them.

4. COMPARISON WITH OTHER SYSTEMS

Although the integration of these particular information sources has not been tested before, there have been other reports of success in accent placement using shallow information sources or simple heuristics, e.g. [1, 8]. A direct quantitative comparison with these other accent placement systems and approaches is not allowed because of differences in language, text genre, annotation or transcription conventions, speaking style, and evaluation method. The only way to have a sound comparison is by testing different approaches on the same test corpus (see also Black [9]).

For an objective comparison with other systems, therefore, we tested on an independent test corpus [10] containing 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, indicating the accents they preferred. For the purpose of evaluation, words were assumed to be accented if they received an accent by at least 7 of the annotators. This corpus was originally designed to evaluate three competitive TTS systems for Dutch.[2] Table 3 shows a comparison of these systems to our approach (*MBL*), which is the classifier that performed best in the previous section. The F scores show that our system outperforms the others by at least 9-11% for news text and 5-10% for email, which is mainly due to a substantially improved precision.

[2]FD (Fluent Dutch) is a commercial product of Fluent Speech; KIK is a former joint research system by Nijmegen University, KPN (a Dutch telephone company) and Eindhoven University of Technology; RS (RealSpeak) is a commercial product by L&H.

5. CONCLUSION

Starting from a text corpus with human-annotated accents, we have developed a system for accent placement that first enriches the input words with lexical, syntactic and informational features by means of generally available, low-cost, knowledge-poor tools, and subsequently employs a memory-based learner to predict accents. We have shown that a limited amount of context-modelling is effective when combined with high values of k, and that the memory-based learner can successfully combine features of a different nature. In independent evaluation showed that our approach outperforms existing TTS systems. Future research will involve including word collocation and prosodic boundary information.

6. ACKNOWLEDGEMENT

Our thanks go out to Olga van Herwijnen and Jacques Terken for permitting the use of their TTS evaluation corpus and to Marc Swerts, Hans Paijmans, and Steven Gillis for discussions and support. 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).

7. REFERENCES

[1] J. Hirschberg, "Pitch accent in context: Predicting intonational prominence from text," *Artificial Intelligence*, vol. 63, pp. 305–340, 1993.

[2] W. Daelemans, J. Zavrel, P. Berck, and S. Gillis, "MBT: A memory-based part of speech tagger-generator," in *Proceedings of the 4th Workshop on Very Large Corpora*, E. Ejerhed and I. Dagan, Eds., Copenhagen, Denmark, 1996, pp. 14–27.

[3] A. Van den Bosch and W. Daelemans, "Memory-based morphological analysis," in *Proceedings of the 37th Annual Meeting of the ACL*, New Brunswick, NJ, 1999, pp. 285–292, ACL.

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

[5] D. W. Aha, D. Kibler, and M. Albert, "Instance-based learning algorithms," *Machine Learning*, vol. 6, pp. 37–66, 1991.

[6] W. Daelemans, A. Van den Bosch, and J. Zavrel, "Forgetting exceptions is harmful in language learning," *Machine Learning, Special issue on Natural Language Learning*, vol. 34, pp. 11–41, 1999.

[7] W. Daelemans, J. Zavrel, K. Van der Sloot, and A. Van den Bosch, "TiMBL: Tilburg Memory Based Learner, version 4.0, reference manual," Tech. Rep. ILK-0104, ILK, Tilburg University, 2001.

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

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

[10] O.M. van Herwijnen and J.M.B. Terken, "Evaluation of automatic assignment of prosodic structure by Dutch TTS-systems," Unpublished report, 2000.