

# Weighted Probability Distribution Voting, an introduction

**Hans van Halteren**

Dept. of Language and Speech  
University of Nijmegen  
P.O. Box 9103  
6500 HD Nijmegen  
The Netherlands  
hvh@let.kun.nl

## Abstract

This paper introduces a new machine learning technique, Weighted Probability Distribution Voting (WPDV). During learning, WPDV determines the output class probability distribution for each input feature, both atomic and complex. During classification, WPDV takes all input features that occur in the new input and adds the corresponding probability distributions, each multiplied by a weight factor which depends on the feature or feature type. The output class with the highest sum is then selected. Apart from the basic mechanism of WPDV, the paper describes some principles for weight selection and feature restriction. Finally, WPDV is shown to produce results which are better than those of other state-of-the-art machine learning systems for several NLP tasks.

## 1 Introduction

Many tasks that play a role in natural language processing can be formulated as classification tasks, i.e. tasks in which an output, taken from a finite set of possible values, is calculated on the basis of a specific set of input information units. An example is wordclass tagging, where the input consists of features of the token to be tagged and its context, and the output consists of a wordclass tag. Classification tasks can generally be handled relatively well with machine learning techniques. A variety of machine learning techniques has already been applied to NLP classification tasks, e.g. decision trees (cf. e.g. Quinlan (1993)), neural networks (cf. e.g. Lawrence et al. (1995)), case bases (cf. e.g. Daelemans et al. (1997) or (1999b)) and maximum entropy models (cf. e.g. Berger et al. (1996) or Mikheev (1998)).

Even though there are many machine learning techniques already, I believe that there is

still room for more. This belief is based primarily on the results of van Halteren et al. (1998) and (To appear), which show that a combination of the outputs of a number of wordclass taggers yields more accurate results than the individual taggers. The method that executes this combination most effectively is a weighted voting method, dubbed Weighted Probability Distribution Voting (WPDV). As WPDV manages to outperform a number of existing machine learning systems for the combination task, even while using a rudimentary weighting scheme, it seems worthwhile to investigate the performance of the system at tasks other than combination. I intend to carry out this investigation in a stepwise fashion. In the first step, I apply the system to several NLP tasks which have been the subject of earlier machine learning experiments (c.f. Daelemans et al. (1999a)), while still using only very simple weighting schemes. If this application shows competitive results as well, the next step is the determination of a procedurally clear and practically usable determination of good (but probably not optimal) weights. At the end of this step, I will be in a position to initiate a more thorough task-oriented comparison with the other machine learning methods used for NLP. The third and final step can then be the search for a theoretically motivated weight determination procedure, which will hopefully lead to even better weights and could give insights into WPDV's potential for combination or hybridization with other available algorithms.

In this paper, I introduce the basics of WPDV and describe the results of the first step of my investigation. I start with some background and terminology (Section 2), and an introduction of the two NLP tasks which I use as examples (Section 3). After this I can explain in detail

how WPDV works and what the special advantages of the technique are (Section 4). In Section 5, I explore the efficiency of a few elementary feature weighting systems for the two example tasks. As WPDV models tend to be very large, I then examine how model size can be limited without loss of too much quality (Section 6). Finally, I summarise the results and indicate the immediate follow-up research on WPDV (Section 7).

## 2 Classification in NLP

In most applications of machine learning systems in the area of NLP, the NLP task is recast as a classification task, i.e. the input is presented in the form of a list of properties and the output requested in the form of a class identifier, which must be taken from a given finite set. A useful example is syntactic wordclass tagging. Suppose we want to know which kind of “that” is used in the sentence “They gave the impression that trade was improving.” We can state a number of properties of the situation, e.g. the word is “that”, the wordclass of the previous word is noun and the wordclass of the next word is also noun. These properties can then function as input to a classification system which has been trained on a large number of such triples. In this case, there are several possible answers, but the correct one, that “that” is a subordinating conjunction here, has the highest probability given this information.<sup>1</sup>

What I have called “properties” above are usually called “features” in the literature. However, the term “feature” is used for several other entities as well, e.g. combinations of properties. For clarity’s sake, I prefer to define my own terminology in this paper. I call each individual property at the input side an *Indicator* and the property at the output side a *Class*. The list of Indicators and the Class together form a *Case*, which is the working unit in each classification task.<sup>2</sup> Case representations tend to use abbrevi-

<sup>1</sup>Obviously, use of more information can lead to better probability estimates. For each NLP task, there are two choices to be made: which properties of the situation to use (and which representation of those properties), and which machine learning mechanism to use to get from the property list to a probability estimate. This paper deals only with the second question.

<sup>2</sup>During classification, the Class part of the Case is unknown (at least to the system), so that the Case is a

ated form, e.g. in the example above, the Case consists of the Indicators {prev=N, word=that, next=N} and the Class {pos=CONJ(sub)}.<sup>3</sup> Any set (i.e. combination) of Indicators forms a *Feature*. Features are subdivided into *Atomic Features*, which contain a single Indicator, e.g. {prev=N} or {word=that}, and *Complex Features*, which contain any other number of Indicators, e.g. {prev=N, next=N} or the complete set {prev=N, word=that, next=N}. Note that the empty set is also a possible Feature. The final two terms are used to define groupings of Indicators and Features and are needed for the explanation of weighting systems below. An *Indicator Family* is a set of all the Indicators of the same type, i.e. sharing an “x=” prefix in my Indicator notation, e.g. prev=N and prev=VB are of the same Family. A *Feature Family* is a set of all the Features consisting of the same combination of Indicator Families, e.g. {prev=N, next=N} and {prev=PRON, next=VB} belong to the same Family.

In these terms, the classification task entails finding the most probable Class (output) for each Case on the basis of the set of Indicators (input). There are many ways to organise the search. An intuitively clear approach is the use of a spatial model: every Indicator sequence is viewed as a vector in a multi-dimensional space. When presented with a new case, the system can determine the probabilities for all Classes from the location in the space. This can e.g. be done by using divisions of the space (e.g. Support Vectors; cf. Burges (1998)), generally repeated to yield ever more refined positioning (Decision Trees; cf. e.g. Quinlan (1993)). Another spatial approach is to look for those training cases which are closest to the current case (Nearest Neighbours; cf. Daelemans et al. (1999b)).

It is also possible to take a more decompositional approach, in which the presence of

list of Indicators combined with “the unknown Class”. As the unknown Class is usually not explicitly mentioned, the term Case is therefore also often used to refer to the list of Indicators alone.

<sup>3</sup>In some systems, e.g. TiMBL (cf. Daelemans et al. (1999b)), it is assumed that each Case has the same number of Indicators and that each Indicator position always holds the same kind of information, e.g. the first position lists the previous wordclass. In such systems the Indicator type need not be given, resulting in lists like {N, that, N}.

Table 1: Example Cases of the GS task.

Prev3	Prev2	Prev1	Focus	Next1	Next2	Next3	Class
=	h	e	a	r	t	s	0A:
b	o	o	k	i	n	g	0k
t	i	e	s	=	=	=	0z
=	=	a	f	a	r	=	1f

Table 2: Example Cases of the TAG task.

Prev2	Prev1	Focus (ambiguous)	Next1 (ambiguous)	Next2 (ambiguous)	Class
=	SQSO	VB	VBG	NN	VB
NNS	BEZ	TO/IN	BE	VCN/VBD	TO
NP	HVZ	VB/VBN/VBD	RP/IN	AT	VBN
=	=	PP3	MD	RN	PP3

all possible Features in a new Case is determined and used in a parametrized calculation to yield the desired probabilities. The calculation is usually based on an underlying statistical model, e.g. assuming that the atomic features are independent allows a simple multiplication of feature-dependent parameters (Naive Bayes; cf. e.g. Gale et al. (1993)). As the atomic features are hardly ever independent, more complicated models tend to yield better results (e.g. Maximum Entropy; cf. Mikheev (1998)).

There are yet other approaches, which do not fall as conveniently in my general classification. Neural Network approaches are best seen as a group, yet the class of a specific system may depend on the network topology. The SNOW system is an example of a hybrid approach, combining spatial division with neural network methods (cf. Roth and Zelenko (1998)).

WPDV follows the decompositional approach but deviates in that it has no obvious underlying statistical model, as will become clear below.

### 3 Example NLP Tasks

The experiments described in this paper are based on two NLP tasks, viz. grapheme to phoneme conversion with stress, and wordclass tag selection. These two tasks have earlier been studied by Daelemans et al. (1999a)). I use the same Case collections and Features as are used in that paper, but, instead of using cross-validation, I only use a single 90%–10% split

into training material and test material.<sup>4</sup> These fixed training and test sets are used in several experiments with WPDV using different parameter settings and also, for comparison, with TiMBL (Daelemans et al., 1999b).

In the grapheme-to-phoneme-with-stress task (GS), the system has to suggest the pronunciation of an English grapheme in a specific word and indicate whether it should be stressed. The Case collection is derived from the CELEX database (cf. Baayen et al. (1993)). The Indicators are the grapheme in question and up to three previous and three next graphemes (see Table 1). These Indicators have up to 42 different values (see Table 4 below). The output consists of one of 159 Classes in which phoneme and stress information are combined. The training set consists of 608K and the test set of 68K Cases.

In the wordclass tag selection task (TAG), the system has to select a tag from the LOB tagset for a word in an English sentence. The Case collection is derived from the tagged LOB corpus (cf. Johansson (1986)). The Indicators are the potential tags of the focus word, the correct tags for the previous two words and the potential tags of the next two words (Table 2).<sup>5</sup> The disambiguated Indicator tags have up to

<sup>4</sup>The training sets I use consist of parts *b* to *j*, the test sets of part *a* of the respective collections.

<sup>5</sup>This is an easier task than the normal tagging task, as the tags for the previous two positions are the correct ones rather than the ones predicted by the tagger.

Table 3: An example of WPDV classification.

Feature	Class probabilities				Weight	Weights x probabilities			
	CS	DT	WPR	IN		CS	DT	WPR	IN
{prev=NN, word=that, next=NN}	0.56	0.34	0.10	0.00	6	3.36	2.04	0.60	0.00
{prev=NN, word=that}	0.62	0.02	0.36	0.00	2	1.24	0.04	0.72	0.00
{prev=NN, next=NN}	0.00	0.01	0.00	0.50	2	0.00	0.02	0.00	1.00
{word=that, next=NN}	0.22	0.74	0.01	0.00	2	0.44	1.48	0.02	0.00
{prev=NN}	0.02	0.00	0.01	0.28	1	0.02	0.00	0.01	0.28
{word=that}	0.65	0.20	0.12	0.00	1	0.65	0.20	0.12	0.00
{next=NN}	0.00	0.03	0.00	0.11	1	0.00	0.03	0.00	0.11
{}	0.02	0.01	0.00	0.11	0.01	0.00	0.00	0.00	0.00
Total						5.71	3.81	1.47	1.39

170 values and the undisambiguated ones up to 497 (see Table 5 below). The output is one of the 169 LOB tags. The training set consists of 941K and the test set of 105K Cases.

#### 4 WPDV models

When presented with a Case, WPDV has to estimate the likelihood of the possible Classes on the basis of the Indicators present in that Case. How it does this exactly is best explained by way of an example. Let us return to the example mentioned above, viz. the determination of the most likely wordclass for “that” when it is both preceded and followed by a noun. In terms of the LOB tagset, the Case at hand becomes {prev=NN, word=that, next=NN}.<sup>6</sup> WPDV first lists which combinations of Indicators, i.e. which Features, are present in the current Case. There are eight of these, which are shown in the first column of Table 3. For each of these Features, WPDV then takes the probability distribution over the various Classes as determined from those Cases in the training material which contain the Feature in question. Obviously, all Classes are taken into consideration in this process, but in this example we will concentrate on four of them, viz. the three tags most often observed for “that”, CS (subordinating conjunction), DT (determiner) and WPR (wh-pronoun), and one which is found often between two nouns, IN (preposition). Columns 2 to 5 of the table show the probabilities for these Classes, given each Feature. After the eight distributions are known, WPDV adds the probabilities for each Class, using weights to give the more informative Features more influence in the

decision. In this example, we follow van Halteren et al. (To appear) where the simple assumption is used that weight can be based on the number of Indicators in the Feature and that a weight of  $N!$  for a Feature with  $N$  Indicators is sufficient to make the larger Features dominate the smaller ones. This leads to the weights in column 6 of the table and the weighted distributions in columns 7 to 10. The result of the weighted addition is shown at the bottom of the table. The highest sum is found for the Class CS, which is therefore selected as being the most likely Class.<sup>7</sup>

As becomes clear from the example, a WPDV model uses two interrelated, but separate, components to estimate Class likelihoods, viz. the probability distributions and the weights. The first component can be derived straightforwardly from the training data, as it merely consists of the number of times each Class is observed with each Feature. WPDV does not need a generating probabilistic model which approximates the observed distributions (like e.g. Maximum Entropy), but uses the observed probability distributions themselves, which can be stored in the form of absolute counts. This type of storage has two special advantages. First of all, it is a trivial task to implement incremental learning.<sup>8</sup> Secondly, the presence of abso-

<sup>7</sup>In this particular example, the WPDV mechanism appears to go through a great deal of work, just to arrive at the same result as the distribution for the full Case ({prev=NN, word=that next=NN}) would have provided. However, the full Case may not be present in the training material, or at least not often enough to yield reliable statistics.

<sup>8</sup>Although optimal weights may have to be recalculated.

<sup>6</sup>NN is the LOB tag used for common nouns.

Table 4: Properties of the Indicators in the GS task.

Indicator Family	Number of values	Information Gain	Gain Ratio
Prev3	42	0.28	0.07
Prev2	42	0.40	0.10
Prev1	42	0.91	0.21
Focus	41	3.09	0.72
Next1	42	0.95	0.23
Next2	40	0.47	0.12
Next3	38	0.31	0.08

lute counts leaves open the possibility of total cross-validation on the training data: when testing (or determining weights), the current case can be conceptually removed from the training set by subtracting one from the corresponding Class count and the total count for each Feature.<sup>9</sup> There is, however, also a disadvantage to storing counts for all observed Features. As the number of Features grows exponentially in the number of Indicator types, the size of the probability distribution component of the model can quickly grow too large for practical use. Below, I will examine some strategies to restrict the Features actually used, and hence the model size.

While the probability distribution component (when not too large) can be constructed easily, the weight component is more of a challenge. For tasks with more varied Indicator sets than those found in the tagger combination task (van Halteren et al., To appear), the use of a weight of  $N!$  for all Features of size  $N$  proves too simplistic, as will become clear in the following section.

## 5 Weights for WPDV

The second component of a WPDV model, the set of Feature weights, determines how the various probability distributions have to be combined into a single Class likelihood estimate. The question, then, is how to find weights which lead to a good performance for each NLP (or other) task. Intuitively, the weight for a Feature should increase with the amount of knowl-

<sup>9</sup>This same trick could of course be used in all systems where absolute counts are stored. However, it is only valid if no other part of the model, e.g. the parametrization, has made use of the removed case, e.g. TiMBL stores absolute counts, but also uses the training set to calculate Indicator weights.

Table 5: Properties of the Indicators in the TAG task.

Indicator Family	Number of values	Information Gain	Gain Ratio
Prev2	169	0.43	0.08
Prev1	170	1.40	0.27
Focus	497	4.99	0.84
Next1	489	1.51	0.26
Next2	478	0.59	0.10

edge it contributes to the determination of the correct Class, its *Informativity*. We can distinguish at least two major factors in a Feature’s Informativity. The first is its *Decisiveness*, the degree to which the presence of a Feature reduces the difficulty of choice within the Class set. This is closely related to the notion of *Information Gain*, which measures the difference between the entropy of the choice with and without knowledge of the presence of the Feature (cf. Quinlan (1986) and (1993)), and we could therefore attempt to use Information Gain as our Decisiveness factor. The second factor is the *Reliability* of the information contributed by the Feature, i.e. how closely the probability distribution which is observed in the training data models the actual probability distribution. Apart from the fact that Reliability increases with the number of observed instances of the Feature, it is as yet unclear how it should be measured. This is especially problematic for very low numbers of instances, a common problem for machine learning systems dealing with NLP data (the *Sparse Data Problem*). Furthermore, it is also far from clear how Decisiveness and Reliability can be combined into an Informativity measure from which we can calculate good weights.

Given the absence of a theoretically likely weight determination system, it is necessary to start with intuitively likely approximations. One of these has already been described above: use a weight of  $N!$  for each Feature of size  $N$ , so that more information weighs more heavily than less information. For the tagger combination task, this works surprisingly well (van Halteren et al., To appear). However, in that task, the Indicator Families are of roughly the same importance. This is not the case for the GS and TAG tasks, where some Indicator Families, especially the focus, contribute much more

Table 6: Accuracy measurements for various WPDV weighting schemes.

	GS	TAG
Baseline: TiMBL	93.58	97.86
WPDV with weight =		
1	91.50	97.68
$N!$	93.44	96.92
$1 * \Pi_I(10 * GainRatio(I))$	92.71	<b>98.15</b>
$N! * \Pi_I(10 * GainRatio(I))$	<b>93.82</b>	97.94

information than others, as becomes clear from Tables 4 and 5. Here, it seems useful to increase the weight when the ingredients of the Feature consist of more informative Indicator Families. For these initial experiments, I opt for a factor proportional to the Indicator Family’s *Gain Ratio*, a normalising derivative of the Information Gain value on the basis of the entropy of the Indicator (cf. Quinlan (1993) or Daelemans et al. (1999b)).

Some results of experiments based on size- and ingredient-dependent weights are shown in Table 6. The first line gives the accuracy of TiMBL for the two tasks, so that it can serve as a baseline to compete with. The other lines list accuracy measurements for WPDV, for experiments with or without a size-dependent factor (i.e. with or without multiplication by  $N!$ ) and with or without an ingredient-dependent factor (i.e. with or without multiplication by the product of the Gain Ratio’s of the included Indicator Families, each multiplied by 10). For both tasks, addition of an ingredient-dependent factor leads to a significant improvement, as expected. Rather unexpected, on the other hand, is that the size-dependent factor is only useful for the GS task. For TAG, its inclusion is even detrimental.

The results in Table 6 show that WPDV, even using first approximation weights again manages to outperform TiMBL.<sup>10</sup> Given the status of TiMBL, this places WPDV at the level of state-of-the-art or better. However, it is

<sup>10</sup>We have to point out that we have only run our experiments with a single training-test split, and that WPDV’s high score might therefore be a lucky coincidence. However, as an anonymous reviewer remarked, the results of TiMBL reported on GS and TAG (Daelemans et al., 1997) on various partitionings show standard deviations that seem to suggest that our conclusion is valid.

also clear that weight determination is task-dependent. It will be necessary to investigate the effects of different weights thoroughly (cf. van Halteren (Submitted)). Although the WPDV algorithm allows weights per individual Feature, technical as well as theoretical considerations suggest that, for the time being, it is better to use Feature and/or Indicator Family-based weights. These can be determined either on the basis of training set properties, as above, or by hill-climbing methods.<sup>11</sup> However, hill-climbing (like other optimization strategies) assumes a correlation between accuracy on the training set and the test set, so that the presence of such a correlation will also have to be investigated. Once it becomes clear what the (near-)optimal weights are, it is hopefully possible to relate these to measurable properties of the training data or, alternatively, held-out tuning data. If this is indeed the case, it may even be possible to take the step from Feature Family weights to weights for individual Features.

## 6 Reducing WPDV model size

A disadvantage of WPDV is the size of its models. When unrestricted, the number of Features quickly runs into the millions (e.g. 4,133,507 for TAG and 5,855,259 for GS). Since only a small number of Features is active for a specific Case, the execution time remains acceptable. The memory requirements, however, do not. During training, memory needs are especially high (e.g. up to 480Mb for TAG and 750Mb for GS); during classification, the data can be stored more efficiently (e.g. 125Mb for TAG and 200Mb for GS), but the requirements still tax the capabilities of the average machine. The question, then, is how we can reduce model size without losing too much classification accuracy. The most likely answer lies in the observation that there is bound to be redundancy in the Feature set, i.e. that there are Features which are not really needed because the information they contribute is the same as that provided by other Features. As the number of Features is too large to identify combinations of Features which are mutually redundant, it will be neces-

<sup>11</sup>In hill-climbing, systematic variations of a weight vector are tested. The best-performing vector is selected and the process is repeated until a maximum accuracy is reached.

Table 7: The effects of Feature size limitation.

-d	Number of Features(GS)	Score(GS)	-d	Number of Features(TAG)	Score(TAG)
-7	5,855,259	93.82			
-6	5,642,052	93.82			
-5	4,468,737	93.32	-5	4,133,507	98.15
-4	2,132,021	90.42	-4	3,571,207	98.16
-3	377,286	82.36	-3	1,712,743	98.11
-2	17,895	68.88	-2	204,183	97.31
-1	259	49.02	-1	1,778	93.66
1,3,5,7	2,909,573	93.79	1,3,5	2,072,638	97.98

Table 8: The effects of Feature frequency limitation.

-f	Number of Features(GS)	Score(GS)	Number of Features(TAG)	Score(TAG)
1	5,855,259	93.82	4,133,507	98.15
2	3,975,716	92.19	1,375,699	98.00
3	2,649,930	91.56	850,011	97.91
5	1,710,624	90.46	496,933	97.90
10	898,068	88.45	253,753	97.38

sary to use properties of individual Features as a basis for their elimination.

The current WPDV implementation has several parameters to specify reasons for Feature elimination (or rather selection). The first parameter (-d) limits the size (dimensionality) of the Features, i.e. only Features consisting of the specified numbers of Indicators are to be used in the model. It allows specification of a maximum (e.g. -d:3 states that Features can contain at most 3 Indicators), a minimum (e.g. -d:4- states Features must have at least 4 Indicators) or a range (e.g. -d:2-4 allows features with 2 to 4 Indicators), but it is also possible to specify a list of sizes (e.g. -d:1,2,4,6). The effects of size limitation for GS and TAG are shown in Table 7. For GS, a maximum size does not appear to be the best choice. High maxima do not yield any useful model size reduction and lower maxima lead to high quality loss. A spread-out size selection appears to be better, as -d:1,3,5,7 produces a model about half the size of the full model with virtually the same quality. For TAG, the situation is reversed, as -d:1,3,5 loses too much quality, but a maximum of 3 produces a model which is still almost as good as the full model.

Another option is a frequency threshold on Features (-f). In this case, a Feature has to be observed a certain number of times in the training data in order to be used in the model, e.g. -f:3 means it has to be present at least 3 times. Measurements for GS and TAG (Table 8) show that frequency thresholds are an efficient way to reduce model size, but may be too coarse-grained to keep quality loss under control (cf. Daelemans et al. (1999a) for similar observations with memory based learning).

A final option is an Informativity threshold (-i). Since it is as yet unclear what Informativity is exactly (see above), this is currently implemented as an entropy-based threshold. A parameter setting of -i:0.2 means that the entropy of the Class probability distribution with that Feature is allowed to be at most 0.2 times the entropy over all cases. Measurements for GS and TAG (Table 9) show that this interpretation of Informativity is practically useless. At higher thresholds, the model size decreases practically nothing and at the first substantial drop, the quality plummets as well.

For both GS and TAG, the most effective way to reduce model size and keep quality loss in bounds is by limiting the Feature size. How-

Table 9: The effects of Feature entropy limitation.

-i	Number of Features(GS)	Score(GS)	Number of Features(TAG)	Score(TAG)
1.0	5,855,259	93.82	4,133,507	98.15
0.5	5,776,574	93.82	4,089,727	98.14
0.2	5,287,570	92.83	3,933,620	98.10
0.1	4,529,645	89.28	3,714,652	97.82

ever, the nature of the optimal size limitation differs between the two. Whether this can be explained from the nature of the data will have to be looked into at a later time. Furthermore, it should theoretically be possible to use the Informativity of Features for their selection. The substitution of entropy for Informativity does not work out, but such a use will certainly have to be investigated as soon as a more workable definition of Informativity is available. Whatever the means of limitation, it appears that, for the tasks studied here, it is possible to produce a qualitatively acceptable model of less than half the size of the full model.

## 7 Conclusion

After having proved its state-of-the-art quality on the tagger combination task in earlier experiments (van Halteren et al., To appear), Weighted Probability Distribution Voting shows competitive results for two further NLP tasks. For both Grapheme to Phoneme with Stress and Part-of-Speech Tagging, it yields a more accurate classification of unseen data than the memory-based TiMBL system, which itself was shown to perform better than or comparable to the Decision Tree system C5.0 (cf. Daelemans et al. (1999a)). It has yet to be determined how WPDV compares to other high-quality methods (e.g. Maximum Entropy) for these two tasks, as well as how it performs on other tasks, but we can already conclude that it is of sufficient quality to continue the investigation of WPDV models. For now, this investigation should focus on two areas:

- The primary problem for WPDV modelling is the determination of good weights for any specific task. Future research will have to show whether these can be calculated on the basis of an Informativity measure or whether more heuristic approaches such as

hill-climbing are needed. In the latter case it will also have to be investigated how well the weight-accuracy functions on training and test data correlate, i.e. whether we can determine where training stops and over-training starts.

- Another matter which has to be looked into more closely is the reduction of model size. For the current tasks, the memory requirements are still acceptable. However, when the number of values per Indicator increases or, more problematically, the number of Indicators grows larger, we will need some (preferably structured) Feature selection mechanism. A Feature-size-based selection is easiest and may be sufficient, but the best selection method proves to be task-dependent and future study will have to show whether it can be based on measurable properties of the data.

Once these fundamental questions have been answered satisfactorily, the WPDV implementation can be extended with automatic training procedures and made available more widely. At that time, it can also be compared more thoroughly to the other state-of-the-art machine learning systems, using various criteria such as accuracy, speed, memory use and user-friendliness. I fully expect this comparison to show that WPDV deserves to become part of the standard machine learning toolbox and to be used on many more tasks, both in NLP and elsewhere.

## References

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

- cent Della Pietra. 1996. Maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1).
- C.J.C. Burges. 1998. A tutorial on support vector machines for pattern recognition. *Data mining and knowledge discovery*, 2(2).
- W. Daelemans, 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.
- W. Daelemans, A. Van den Bosch, and J. Zavrel. 1999a. 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. 1999b. TiMBL: Tilburg Memory Based Learner, version 2.0, reference manual. Technical Report ILK-9901, ILK, Tilburg University.
- W. Gale, K. Church, and D. Yarowsky. 1993. A method for disambiguating word senses in a large corpus. *Computing in the Humanities*, 1:415–439.
- H. van Halteren, J. Zavrel, and W. Daelemans. 1998. Improving data-driven word-class tagging by system combination. In *Proc. COLING/ACL98*, pages 491–497, Montreal, Canada, August 10-14.
- H. van Halteren, J. Zavrel, and W. Daelemans. To appear. Improving accuracy in NLP through combination of machine learning systems. *Computational Linguistics*.
- H. van Halteren. Submitted. Systematic investigation of a parameter space: weights for WPDV models.
- S. Johansson. 1986. *The tagged LOB Corpus: User's Manual*. Norwegian Computing Centre for the Humanities, Bergen, Norway.
- S. Lawrence, C.L. Giles, and Fong S. 1995. On the applicability of neural network and machine learning methodologies to natural language processing. Technical Report UMIACS-TR-95-64 and CS-TR-3479, UMIACS, University of Maryland.
- A. Mikheev. 1998. Feature lattices for maximum entropy models. In *Proc. COLING/ACL98*, pages 848–854, Montreal, Canada, August 10-14.
- J.R. Quinlan. 1986. Induction of Decision Trees. *Machine Learning*, 1:81–206.
- J.R. Quinlan. 1993. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA.
- D. Roth and D. Zelenko. 1998. Part of speech tagging using a network of linear separators. In *Proc. COLING/ACL98*, pages 1136–1142, Montreal, Canada, August 10-14.