In: *Proceedings of CoNLL-2000 and LLL-2000*, pages 119–122, Lisbon, Portugal, 2000.

A Default First Order Family Weight Determination Procedure for WPDV Models

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

Weighted Probability Distribution Voting (WPDV) is a newly designed machine learning algorithm, for which research is currently aimed at the determination of good weighting schemes. This paper describes a simple yet effective weight determination procedure, which leads to models that can produce competitive results for a number of NLP classification tasks.

1 The WPDV algorithm

Weighted Probability Distribution Voting (WPDV) is a supervised learning approach to classification. A case which is to be classified is represented as a feature-value pair set:

$$F_{case} = \{\{f_1 = v_1\}, ..., \{f_n = v_n\}\}$$

An estimation of the probabilities of the various classes for the case in question is then based on the classes observed with similar feature-value pair sets in the training data. To be exact, the probability of class C for F_{case} is estimated as a weighted sum over all possible subsets F_{sub} of F_{case}:

$$\hat{P}(C) = N(C) \sum_{F_{sub} \subset F_{case}} W_{F_{sub}} \frac{freq(C \mid F_{sub})}{freq(F_{sub})}$$

with the frequencies ($freq$) measured on the training data, and $N(C)$ a normalizing factor such that $\sum \hat{P}(C) = 1$.

In principle, the weight factors $W_{F_{sub}}$ can be assigned per individual subset. For the time being, however, they are assigned for groups of subsets. First of all, it is possible to restrict the subsets that are taken into account in the model, using the size of the subset (e.g. F_{sub} contains at most 4 elements) and/or its frequency (e.g. F_{sub} occurs at least twice in the training material). Subsets which do not fulfil the chosen criteria are not used. For the subsets that are used, weight factors are not assigned per individual subset either, but rather per "family", where a family consists of those subsets which contain the same combination of feature types (i.e. the same f_i).

The two components of a WPDV model, distributions and weights, are determined separately. In this paper, I will use the term *training set* for the data on which the distributions are based and *tuning set* for the data on the basis of which the weights are selected. Whether these two sets should be disjunct or can coincide is one of the subjects under investigation.

2 Family weights

The various family weighting schemes can be classified according to the type of use they make of the tuning data. Here, I use a very rough classification, into weighting scheme *orders*.

With 0^{th} **order weights**, no information whatsoever is used about the data in tuning set. Examples of such rudimentary weighting schemes are the use of a weight of $k!$ for all subsets containing k elements, as has been used e.g. for wordclass tagger combination (van Halteren et al., To appear), or even a uniform weight for all subsets.

With 1^{st} **order weights**, information is used about the individual feature types, i.e.

$$W_{F_{sub}} = \prod_{\{i \mid \{f_i = v_i\} \in F_{sub}\}} W_{f_i}$$

First order weights ignore any possible interaction between two or more feature types, but have the clear advantage of corresponding to a reasonably low number of weights, viz. as many as there are feature types.

With n^{th} **order weights**, interaction patterns are determined of up to n feature types and the family weights are adjusted to compensate for the interaction. When n is equal to the total number of feature types, this corresponds to weight determination per individual family. n^{th} order weighting generally requires much larger numbers of weights, which can be expected to lead to much slower tuning procedures. In this paper, therefore, I focus on first order weighting.

3 First order weight determination

As argumented in an earlier paper (van Halteren, 2000a), a theory-based feature weight determination would have to take into account each feature's *decisiveness* and *reliability*. However, clear definitions of these qualities, and hence also means to measure them, are as yet sorely lacking. As a result, a more pragmatic approach will have to be taken. Reliability is ignored altogether at the moment,[1] and decisiveness replaced by an entropy-related measure.

3.1 Initial weights

The weight given to each feature type f_i should preferably increase with the amount of information it contributes to the classification process. A measure related to this is *Information Gain*, which represents the difference between the entropy of the choice with and without knowledge of the presence of a feature (cf. Quinlan (1986)). As do Daelemans et al. (2000), I opt for a factor proportional to the feature type's *Gain Ratio*, a normalising derivative of the Information Gain value. The weight factors W_{f_i} are set to an optimal multiplication constant C times the measured Gain Ratio for f_i. C is determined by calculating the accuracies for various values of C on the tuning set[2] and selecting the C which yields the highest accuracy.

[1] It may still be present, though, in the form of the abovementioned frequency threshold for features.

[2] If the tuning set coincides with the training set, all parts of the tuning procedure are done in leave-one-out mode: in the WPDV implementation, it is possible to (virtually) remove the information about each individual instance from the model when that specific instance has to be classified.

3.2 Hill-climbing

Since the initial weight determination is based on pragmatic rather than theoretical considerations, it is unlikely that the resulting weights are already the optimal ones. For this reason, an attempt is made to locate even better weight vectors in the n-dimensional weight space. The navigation mechanism used in this search is *hill-climbing*. This means that systematic variations of the currently best vector are investigated. If the best variation is better than the currently best vector, that variation is taken as the best vector and the process is repeated. This repetition continues until no better vector is found.

In the experiments described here, the variation consists of multiplication or division of each individual W_{f_i} by a variable V (i.e. $2n$ new vectors are tested each time), which is increased if a better vector is found, and otherwise decreased. The process is halted as soon as V falls below some pre-determined threshold.

Hill-climbing, as most other optimalization techniques, is vulnerable to overtraining. To lessen this vulnerability, the WPDV hill-climbing implementation splits its tuning material into several (normally five) parts. A switch to a new weight vector is only taken if the accuracy increases on the tuning set as a whole and does not decrease on more than one part, i.e. some losses are accepted but only if they are localized.

4 Quality of the first order weights

In order to determine the quality of the WPDV system, using first order weights as described above, I run a series of experiments, using tasks introduced by Daelemans et al. (1999):[3]

The **Part-of-speech tagging** task (POS) is to determine a wordclass tag on the basis of disambiguated tags of two preceding tokens and undisambiguated tags for the focus and two following tokens.[4] 5 features with 170–480 values; 169 classes; 837Kcase training; 2x105Kcase test.

The **Grapheme-to-phoneme conversion with stress** task (GS) is to determine the pronunciation of an English grapheme, including

[3] I only give a rough description of the tasks here. For the exact details, I refer the reader to Daelemans et al. (1999).

[4] For a overall WPDV approach to wordclass tagging, see van Halteren (2000b).

Table 1: Accuracies for the POS task (with the training set *ah* tested in leave-one-out mode)

Weighting scheme	Test set		
	ah	*i*	*j*
Comparison			
Naive Bayes		96.41	96.24
TiMBL (k=1)		97.83	97.79
Maccent (freq=2;iter=150)		98.07	98.03
Maccent (freq=1;iter=300)		98.13	98.10
WPDV 0^{th} order weights			
1	97.66	97.71	97.63
$k!$	96.86	96.92	96.86
WPDV initial 1^{st} order			
tune = *ah* (10GR)	*98.14*	98.16	98.12
tune = *i* (12GR)	98.14	*98.17*	98.12
tune = *j* (11GR)	98.14	98.16	*98.13*
WPDV with hill-climbing			
tune = *ah* (30 steps)	*98.17*	98.21	98.15
tune = *i* (20 steps)	98.15	*98.20*	98.12
tune = *j* (20 steps)	98.15	98.18	*98.16*

Table 2: Accuracies for the GS task (with the training set *ah* tested in leave-one-out mode)

Weighting scheme	Test set		
	ah	*i*	*j*
Comparison			
Naive Bayes		50.05	49.98
TiMBL (k=1)		92.25	92.02
Maccent (freq=2;iter=150)		79.41	79.36
Maccent (freq=1;iter=300)		80.43	80.35
WPDV 0^{th} order weights			
1	90.99	90.49	90.25
$k!$	92.77	92.05	91.89
WPDV initial 1^{st} order			
tune = *ah* (30GR)	*93.27*	92.74	92.52
tune = *i* (25GR)	93.24	*92.76*	92.54
tune = *j* (25GR)	93.24	92.76	*92.54*
WPDV with hill-climbing			
tune = *ah* (34 steps)	*93.29*	92.77	92.53
tune = *i* (28 steps)	93.25	*92.79*	92.53
tune = *j* (12 steps)	93.24	92.76	*92.54*

presence of stress, on the basis of the focus grapheme, three preceding and three following graphemes. 7 features with 42 values each; 159 classes; 540Kcase training; 2x68Kcase test.

The **PP attachment** task (PP) is prepositional phrase attachment to either a preceding verb or a preceding noun, on the basis of the verb, the noun, the preposition in question and the head noun of the prepositional complement. 4 features with 3474, 4612, 68 and 5780 values; 2 classes; 19Kcase training; 2x2Kcase test.

The **NP chunking** task (NP) is the determination of the position of the focus token in a base NP chunk (at beginning of chunk, in chunk, or not in chunk), on the basis of the words and tags for two preceding tokens, the focus and one following token, and also the predictions by three newfirst stage classifiers for the task.[5] 11 features with 3 (first stage classifiers), 90 (tags) and 20K (words) values; 3 classes; 201Kcase training; 2x25Kcase test.[6]

For each of the tasks, sections *a* to *h* of the data set are used as the training set and sections *i* and *j* as (two separate) test sets. All three are also used as tuning sets. This allows a comparison between tuning on the training set itself and on a held-out tuning set. For comparison with some other well-known machine learning algorithms, I complement the WPDV experiments with accuracy measurements for three other systems: 1) A system using a **Naive Bayes** probability estimation; 2) **TiMBL**, using memory based learning and probability estimation based on the nearest neighbours (Daelemans et al., 2000),[7] for which I use the parameters which yielded the best results according to Daelemans et al. (1999); and 3) **Maccent**, a maximum entropy based system,[8] for which I use both the default parameters, viz. a frequency threshold of 2 for features to be used and 150 iterations of improved iterative scaling, and a more ambitious parameter setting, viz. a threshold of 1 and 300 iterations.

The results for various WPDV weights, and the other machine learning techniques are listed in Tables 1 to 4.[9] Except for one case (PP with tune on *j* and test on *i*), the first order weight WPDV results are all higher than those for the

[5] For a WPDV approach to a more general chunking task, see my contribution to the CoNLL shared task, elsewhere in these proceedings.

[6] The number of feature combinations for the NP task is so large that the WPDV model has to be limited. For the current experiments, I have opted for a maximum size for F_{sub} of four features and a threshold frequency of two observations in the training set.

[7] `http://ilk.kub.nl/`.

[8] `http://www.cs.kuleuven.ac.be/~ldh`.

[9] The accuracy is shown in italics whenever the tuning set is equal to the test set, i.e. when there is an unfair advantage.

Table 3: Accuracies for the PP task (with the training set *ah* tested in leave-one-out mode)

Weighting scheme	Test set		
	ah	*i*	*j*
Comparison			
Naive Bayes		82.68	82.64
TiMBL (k=1)		83.43	81.97
Maccent (freq=2;iter=150)		81.00	80.25
Maccent (freq=1;iter=300)		79.41	79.79
WPDV 0^{th} order weights			
1	80.83	82.26	81.46
$k!$	80.76	82.30	81.30
WPDV initial 1^{st} order			
tune = *ah* (21GR)	*82.89*	83.64	82.38
tune = *i* (15GR)	82.82	*83.81*	82.55
tune = *j* (11GR)	82.60	83.26	*82.76*
WPDV with hill-climbing			
tune = *ah* (19 steps)	*83.10*	83.72	82.68
tune = *i* (18 steps)	82.95	*84.06*	82.80
tune = *j* (16 steps)	82.65	83.10	*82.93*

Table 4: Accuracies for the NP task (with the training set *ah* tested in leave-one-out mode)

Weighting scheme	Test set		
	ah	*i*	*j*
Comparison			
Naive Bayes		96.52	96.49
TiMBL (k=3)		98.34	98.22
Maccent (freq=2;iter=150)		97.89	97.75
Maccent (freq=1;iter=300)		97.66	97.45
WPDV 0^{th} order weights			
1	97.56	97.77	97.69
$k!$	97.74	97.97	97.87
WPDV initial 1^{st} order			
tune = *ah* (380GR)	*98.19*	98.38	98.26
tune = *i* (60GR)	98.14	*98.39*	98.17
tune = *j* (360GR)	98.19	98.38	*98.27*
WPDV with hill-climbing			
tune = *ah* (50 steps)	*98.36*	98.54	98.44
tune = *i* (34 steps)	98.25	*98.57*	98.33
tune = *j* (12 steps)	98.19	98.38	*98.27*

comparison systems.[10] 0^{th} order weights generally do not reach this level of accuracy.

Hill-climbing with the tuning set equal to the training set produces the best results overall. It always leads to an improvement over initial weights of the accuracies on both test sets, although sometimes very small (GS). Equally important, the improvement on the test sets is comparable to that on the tuning/training set. This is certainly not the case for hill-climbing with the tuning set equal to the other test set, which generally does not reach the same level of accuracy and may even be detrimental (climbing on PP_j).

Strangely enough, hill-climbing with the tuning set equal to the test set itself sometimes does not even yield the best quality for that test set (POS with test set *i* and especially NP with *j*). This shows that the weight→accuracy function does have local maxima, and the increased risk for smaller data sets to run into a sub-optimal one is high enough that it happens in at least two of the eight test set climbs.

[10] The accuracies for TiMBL are lower than those found by Daelemans et al. (1999): POS_i 97.95, POS_j 97.90, GS_i 93.75, GS_j 93.58, PP_i 83.64, PP_j 82.51, NP_i 98.38 and NP_j 98.25. This is due to the use of eight part training sets instead of nine. The extreme differences for the GS task show how much this task depends on individual observations rather than on generalizations, which probably also explains why Naive Bayes and Maximum Extropy (Maccent) handle this task so badly.

In summary, hill-climbing should preferably be done with the tuning set equal to the training set. This is not surprising, as the leave-one-out mechanism allows the training set to behave as held-out data, while containing eight times more cases than a test set turned tuning set. The disadvantage is a much more time-intensive hill-climbing procedure, but when developing an actual production model, the weights only have to be determined once and the results appear to be worth it most of the time.

References

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. 2000. TiMBL: Tilburg Memory Based Learner, version 3.0, reference manual. Tech. Report ILK-00-01, ILK, Tilburg University.

H. van Halteren. 2000a. Weighted Probability Distribution Voting, an introduction. In *Computational linguistics in the Netherlands, 1999.*

H. van Halteren. 2000b. The detection of inconsistency in manually tagged text. In *Proc. LINC2000.*

H. van Halteren, J. Zavrel, and W. Daelemans. To appear. Improving accuracy in NLP through combination of machine learning systems. *Computational Linguistics.*

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