Simultaneous Feature Selection and Parameter Optimization for Memory-based Natural Language Processing

Anne Kool Jakub Zavrel
Walter Daelemans*
CNTS Language Technology Group
University of Antwerp, UIA, Universiteitsplein 1, 2610 Antwerpen, Belgium

Abstract

We investigate the effects of (i) feature subset selection, (ii) parameter optimization, and (iii) simultaneous feature selection and parameter optimization in Memory-based Natural Language Processing (MBLP). We use a simple genetic algorithm for this problem and compare it to two iterative search methods on some typical tasks in natural language processing: part-of-speech tagging of known and unknown words, and grapheme to phoneme conversion with stress assignment. We find that (i) feature selection always outperforms the MBLP variant without selection, (ii) optimization of parameters for each specific task is beneficial, and (iii) the combination of parameter optimization and feature selection performed simultaneously can lead to more accurate classification results. However, we have found no indications on our data that GAs reach a significantly better accuracy than the iterative methods, and, in general, the approach promises larger gains for more effective search methods.

In: Feelders (ed.), Proceedings 10th BENELEARN meeting, p. 93-100, 2000

1 Memory-Based Language Processing

Memory-Based Language Processing (Daelemans, van den Bosch, and Zavrel, 1999) is based on the idea that language acquisition should be seen as the incremental storage of exemplars of specific tasks, and language processing as analogical reasoning on the basis of these stored exemplars. These exemplars take the form of a vector of, typically, nominal features, describing a linguistic problem and its context, and an associated class symbol representing the solution to the problem.

Most Natural Language Processing (NLP) tasks are characterised by the interaction of few regularities with many subregularities and "pockets of exceptions", and by the interaction of various sources of information (e.g. lexical and contextual). We briefly describe how MBLP handles these problems.

1. Exceptions. A memory-based strategy, in which no rules are handcrafted or induced and in which low-frequency events and exceptions are kept available for analogical reasoning, can empirically be shown to solve this problem better than eager learning systems (Daelemans, van den Bosch, and Zavrel, 1999). Eager learning methods "forget" relevant exceptional information, because of their pruning and frequency-based abstraction methods. The basic algorithm we use is a variant of IB1 (Aha, Kibler, and Albert, 1991) in which the distance between a test item and each memory item is defined as the number of features for which they have a different value (overlap metric). Classification of a new test item is done by assigning the most frequent category among the k most similar memory item(s).

2. Information source interaction. IB1 does not solve the problem of modeling the interaction between the various sources of information considered relevant in solving the task. E.g., in assigning a syntactic class to an unknown word both aspects of the form (morphology) of the word, and of the context in which the word occurs are relevant. In an

*Research funded by CELE, S.AI.L Trust V.Z.W., Ieper, Belgium.

MBLP approach, this interaction can be modeled by means of *feature weighting*, the assignment of a relevance weight to the different features of the input vector reflecting their importance in computing similarity between vectors. This feature relevance assignment may be based on information-theoretic or statistical notions. Further refinement of the distance measure can be achieved by calculating a separate distance for each pair of values of the same feature.

Previous experiments with memory-based and other machine learning methods have shown that finding suitable parameter settings for a system is an important factor in the success or failure of a learning method. In memory-based learning, apart from finding a good value for the number of k nearest neighbours, it is important to calculate reasonable estimates about the relevance of the features. Memory-based Language Processing seems to benefit from the use of feature weights in its similarity function and *feature selection* can also improve both accuracy and efficiency by discarding some features altogether because of their irrelevance or even counter-productivity in learning to solve the task. Moreover, parameter optimization and feature selection or weighting are likely to interact. In this paper we apply an evolutionary computing approach to these problems.

We believe the applicability of this approach goes well beyond MBLP. Other machine learning algorithms are confronted with similar feature weighting, feature selection, and parameter optimization problems, and our results may be relevant for these other algorithms as well.

The rest of this paper is structured as follows. In section 2, the search space for our experiments is defined and we state our hypotheses. In Section 3 we present our use of Genetic Algorithms, and more traditional search methods for searching the defined space. Section 4 describes the data used, and presents our experimental results. Section 5 mentions related approaches and draws conclusions.

2 Problems and Hypotheses

In a scenario where all information contributes in the same manner to the classification, the non-weighted IB1 metric will perform reasonably well on a given task. But on most of our data, classification accuracy is higher when weight assignment methods are used to determine to what degree features are good predictors of the class labels. As we cannot identify the best weighting scheme beforehand, we have found it useful, for each new task, to exhaustively search for the best setting. Implemented in our memory-based learner are five different options for relevance assignment: either no weighting is used which implies that all features are treated as equally relevant, or information-theoretic measures (gain ratio, information gain) are used or alternatively statistical methods (chi-square, shared variance) can be used. In these experiments we will try to identify the optimal weight measure for each new dataset.
Feature weighting methods assign continuous weights to the features, but some features may be completely irrelevant and disturb the classification. Better results can be obtained by leaving these features out. Faced with insufficient domain knowledge to make restrictions beforehand, we do not know which features can safely be discarded from the feature set. Binary feature selection (where a feature is either present or not present during classification) is not a straightforward task: not all features are always equally informative, some features may even be completely irrelevant, or a number of features may correlate in a specific way such that their combined presence influences the accuracy of the classifier.

We have also found it useful on some tasks to use the information that two values of a feature are similar, this technique is referred to as modified value difference metric. The MVDM method determines the similarity of the values of a feature by looking at co-occurence of values with target classes. Data sparsity can be a problem in practical applications, many values occur only once in the whole data set. This means that if two such values occur with the same class, MVDM will regard them as identical, and if they occur with two different classes their distance will be maximal. Ideally, we would like to apply this metric only to certain features for which sufficient examples are present in the data.

Parameter optimization for the number of k nearest neighbours and the different weight assignment methods can be performed exhaustively. But combining the search for a good feature subset and for optimal parameter settings implies a large and complex space of possible settings and relations between features and parameters. While examining different feature settings, the criteria for parameter settings change as well. It is in optimization problems like this that methods such as evolutionary algorithms promise to be of use. In particular, there is a complex relationship between the type of feature settings, and the optimal setting of the other parameters. For example, a correlation can often be observed between the use of MVDM and the benefit from using a larger number of nearest neighbours.

In the experiments we tested the following hypotheses:

1. Feature selection methods will filter out irrelevant features that could disturb the classification and the GA may catch correlations between features better than the more classical methods.
2. Parameter optimization for each specific dataset will enhance the performance of the classifier.
3. Applying feature subset selection and parameter optimization simultaneously with a GA may have the advantage of finding the dependencies between a feature subset and the different parameters, drawing profit from the algorithm's inherent parallelism, whereas iterative methods may fail to capture these relations.

3 Genetic Algorithms

Genetic algorithms are a domain-independent search technique and are well-suited for exploration of large and complex search spaces. As they are blind towards the problem domain (the only information they have is an evaluation function), they are an interesting algorithm when little domain-knowledge can be supplied. Genetic algorithms start with a collection of randomly initialized hypothesis solutions in the form of symbolic strings, called chromosomes or individuals. The chromosome string has several genes, each of which represents a particular feature with a certain value. This population of chromosomes is cyclically evolved through a number of generations and at each run genetically inspired operators steer the hypothesis space towards new and promising regions, evaluating, selecting, combining and mutating the hypothesis strings until a stop criterium is reached.

In the experiments, we linked our memory-based learner TiMBL[1] to PGAPack[2]. During the feature subset selection experiments the string is composed of binary values, indicating presence or absence of a feature. During the simultaneous optimization experiments, the first gene in the string encodes the values for k (only odd values are used, to avoid ties), the second gene indicates which weight settings are used and the remaining genes are reserved for the features. In these experiments we look at feature selection as an optimization process, where each feature has three possible values: a feature can either be present, it can be absent or its MVD can be calculated. Each feature-gene can take on any of these three values and subset selection is then optimization of these values for the specific features. The fitness of the strings is determined by running the memory-based learner with each string on a validation set, and returning the resulting accuracy as a fitness value for that string. Hence, selection with the GA is an instance of a *wrapper* approach as opposed to a *filter* approach such as information gain (Kohavi and John, 1995).

For comparison with evolutionary feature selection, we include two popular classical wrapper methods: backward elimination (henceforth BA) and forward selection (henceforth FO). Both are greedy searchers. Forward selection starts from an empty set of features and backward selection begins with a full set of features. At each further addition (or deletion, for backward elimination) the feature with the highest accuracy increase (resp. lowest accuracy decrease) is selected, until improvement stalls (resp. performance drops). The backward elimination /forward selection principle is kept for comparison with simultaneous feature selection and parameter optimization. Forward selection starts with every value set to zero and backward elimination starts with every value set to one. Each value is flipped iteratively while the other values remain unchanged, the value with the highest accuracy increase (resp. lowest accuracy decrease) is selected, until improvement stalls (resp. performance drops). One would

[1] TiMBL is available from http://ilk.kub.nl/ and the algorithms are described in more detail in (Daelemans et al., 1999).

[2] A software environment for evolutionary computation developed by D. Levine, Argonne National Laboratory, available from ftp://ftp.mcs.anl.gov/pub/pgapack/

expect these methods to perform worse than the GA, because their greedy search can have difficulties with relationships accross several parameters and features.

4 Experiments

4.1 Data

We have tested our hypotheses on three natural language datasets involving two tasks: part-of-speech tagging and grapheme to phoneme conversion with stress assignment.

The part-of-speech (POS) data set is based on the TOSCA tagged LOB corpus of English. There are two versions of the data, one involved with predicting the part-of-speech for "known" words, and one for unknown words. The features represent information about the word to be tagged (focus) and its context. For unknown words the focus provides no information about the possible categories of the word and we have to rely on context and wordform only, this data set contains 65275 instances. The features used are the coded POS-tags of two words before and two words after the focus word to be tagged, the last three letters of the focus word, and information on hyphenation and capitalisation. The known words data set is larger and contains 1045541 cases, each instance has two extra features: the focus word itself and its ambiguity class. There are 111 possible classes (part of speech tags) to predict.

In the example, the word 'electing', ending in $-ing$, preceded by a single left-quote (') and a verb (stop), and followed by a single noun (life) and a plural noun (peers), is classified as a verb.

Features									Class
Ap	Bn	AN	Cg	i	n	g	0	0	DZ

Table 1: Example of unknown words tagging task.

The grapheme-phoneme data is based on the CELEX dictionary for English. The mapping to be learned is from a letter in context to a phonetic representation with stress markers. We used three letters of left and right context, so that each instance has seven features. The dataset consists of 77565 words, the total number of instances amounts to 675745.

In the example, the focus letter 's', preceded by a three-letter left context and followed by a three-letter right context, receives the pronunciation 'z' and is marked as unstressed '0'.

Features							Class
l	-	u	s	i	n	g	0z

Table 2: Example of grapheme to phoneme conversion task.

4.2 Method

Because wrapper methods get their evaluation feedback directly from accuracy measurements on a validation set, we have split our data into 80% training material and 10% validation material. The settings obtained on the validation set are then evaluated on a final 10% held-out test set. This method was used for all experiments.

Parameter settings for the genetic algorithm were kept constant: a population size of 20, a two-point crossover probability of 0.85, a mutation rate of 0.006, an elitist replacement strategy (where the best string of a previous generation is copied to the new population), and tournament selection. The populations were evolved for a maximum of 100 generations or stopped when no change had occurred for over 20 generations.

The default settings for the memory-based classifier are weighted overlap (IB1-IG), using gain ratio as a relevance assignment measure, with the value for k (nearest neighbours) set to one.

4.3 Results

4.3.1 Exhaustive search

As a baseline we have done an exhaustive search of three parameters: ($k \in \{1, 3, 5, ..., 21\}$,$w \in \{0, 1, 2, 3, 4\}$[3], $metric \in \{O(verlap), M(vdm)\}$). The difference with the other experiments is that here, the feature set was not optimized. The other difference is that here the search is guaranteed to find the optimum, whereas in the other experiments we can end up in local maxima.

To gain understanding of the behavior of MBL on these datasets, we have plotted the score on the validation set in Figure 1. The results of the best parameter combination for each task on the test set are included in Tables 3, 4, and 5.

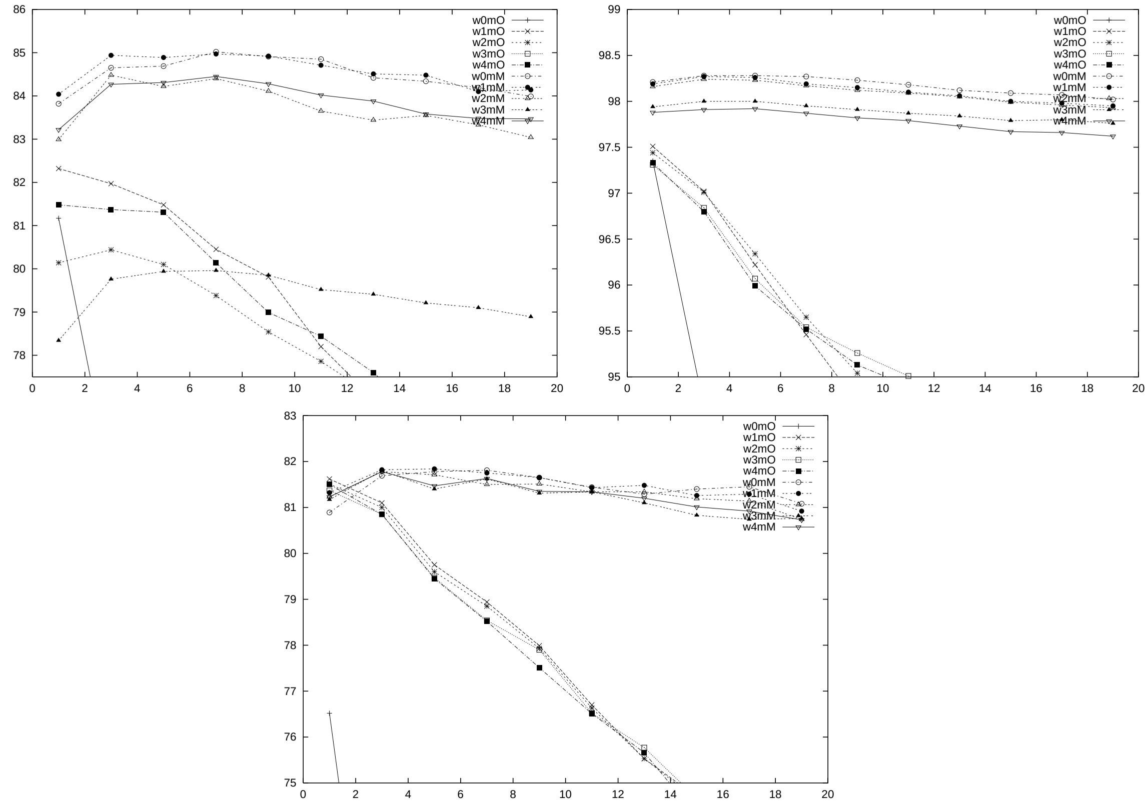

Figure 1: Score on POS unknown, POS known, and GS for exhaustive parameter optimization. The line labels are of the form wXmY, where X is the value of the weight parameter, and Y is either O(verlap) or M(vdm). The accuracy on the validation set (vertical axis) is plotted against the value of the k parameter (horizontal axis)

What we see in these graphs is that there is a clear grouping into three types of behavior. The first is the unweighted overlap metric, whose accuracy degrades very quickly with increasing k. The second group is the weighted overlap metric, which also degrades with increasing k, but much less quickly. A different behaviour can be observed for the third group, using the MVDM metric. Here the performance actually improves with larger values of k, and remains high throughout a large range of k. Finally, it seems that for these three tasks the setting of the weight parameter is of little importance.

The reason for this lies in the fact TiMBL is actually a k nearest *distances* algorithm. When no weighting is used, the first distance already contains many tied neighbours, and the second or third or further distance will contain very large numbers of instances that are one or two or more features dissimilar from the test instance than the most similar set. For the weighted overlap metric many of these ties are broken, because the weights will prefer

[3]The meaning of these settings is listed in the Appendix.

mismatches on unimportant features, and hence rank more relevant instances at lower distance "buckets". The MVDM metric assigns a unique distance to each pair of feature values, so the ranking of the nearest neighbours in the "distance buckets" is freed from all arbitrary ties, and the behaviour is closer to a true k-nearest *neighbours* algorithm. Note that selecting k nearest neighbours (instead of distances) with the Overlap metric would not likely improve upon the behaviour of TiMBL with $k = 1$, because the arbitrary ties would remain a problem.

4.3.2 Simultaneous optimization

In the following three tables we show the accuracy measurements of our experiments on the three datasets. The actual settings found by each method are presented in Tables 6, 7, and 8 in the Appendix. We can see that *a)* exhaustive search for optimal parameter settings improves the classification accuracy and that *b)* selection of a subset of features leads to similar or better results with a reduction in the number of features used. For *c)* simultaneous parameter optimization and feature selection, shows improvement for the POS known words task (significant; McNemar's chi-square; p<0.001), and the GS task (not significant; p=0.318). The exhaustive search for optimal parameters is better than the simultaneously optimized case for POS unknown (but not significantly; p=0.684).

	Default Parameters	Optimized Parameters
All Features	DE 82.63	EX 85.43
Optimized Features	GA 84.39	GA 84.88
	BA 84.39	BA 85.15
	FO 84.54	FO 84.97

Table 3: POS unknown words tagging task.

	Default Parameters	Optimized Parameters
All Features	DE 97.51	EX 98.31
Optimized Features	GA 98.30	GA 98.19
	BA 98.30	BA 98.43
	FO 98.28	FO 98.39

Table 4: POS known words tagging task.

	Default Parameters	Optimized Parameters
All Features	DE 81.62	EX 81.69
Optimized Features	GA 81.56	GA 81.99
	BA 81.56	BA 81.45
	FO 81.62	FO 81.62

Table 5: Grapheme to phoneme conversion with stress (GS).

The results, however, suggest that the application of either evolutionary, backward or forward selection does not lead to major differences in accuracy. Statistical tests (McNemar's chi-square) confirm that the differences in results cannot be interpreted as significantly better for either evolutionary, backward or forward selection, except on the POS known words data set, where both iterative methods perform significantly better (p<0.05) for simultaneous parameter optimization and feature selection. An interesting point is that the search space of the exhaustive search is in fact subsumed in the space of simultaneous optimization of features and parameters. So, an ideal search algorithm would always be able to at least equal the score of the exhaustive case. However, since this does not always happen in our experiments we can conclude that the realization of the promise of the approach is still in need of better search algorithms.

5 Conclusions and Related Research

The issue of feature-relevance assignment is well-documented in the machine learning literature. Excellent comparative surveys are (Wettschereck, Aha, and Mohri, 1997) and (Wettschereck and Aha, 1995) or (Blum and Langley, 1997). Feature subset selection by means of evolutionary algorithms was investigated by Skalak (1994), Vafaie and de Jong (1992), and Yang and Honavar (1997). Other work deals with evolutionary approaches for continuous feature weight assignment such as Wilson and Martinez (1996), or Punch and Goodman (1993).

The conclusions from these papers are in agreement with our findings on the natural language data: feature selection generally improves accuracy with a reduction in the number of features used. However, we have found no results (on these particular data) that indicate an advantage of an evolutionary feature selection approach over the more classical iterative methods. The optimization of small numbers of parameters is always to be recommended, and can be done by an exhaustive search on the validation set. Finally, the simultaneous application of feature selection and parameter optimization has shown some performance gains, but further work on better search algorithms is needed to realize the full potential of the approach.

6 References

Aha, D., D. Kibler, and M. Albert. 1991. Instance-based learning algorithms. In *Machine Learning Vol. 6, pp 37-66.*

Blum, A. and P. Langley. 1997. Selection of relevant features and examples in machine learning. In *Machine Learning: Artificial Intelligence,97, pp 245-271.*

Daelemans, W., A. van den Bosch, and J. Zavrel. 1999. Forgetting exceptions is harmful in language learning. In *Machine Learning, special issue on natural language learning, 34 , pp 11-43.*

Daelemans, W., J. Zavrel, K. van der Sloot, and A. van den Bosch. 1999. Timbl: Tilburg memory based learner, version 2.0, reference guide. Ilk technical report 99-01, ILK.

Kohavi, R. and G.H. John. 1995. Wrappers for feature subset selection. In *Artificial Intelligence Journal, Special Issue on Relevance Vol.97, pp 273-324.*

Punch, W. F., E.D. Goodman, Lai Chia-Shun Min Pei, P. Hovland, and R. Enbody. 1993. Further research on feature selection and classification using genetic algorithms. In *Proceedings of the Fifth International Conference on Genetic Algorithms, pp 557.*

Skalak, D. B. 1994. Prototype and feature selection by sampling and random mutation hill climbing algorithms. In *Proceedings of the eleventh International Conference on Machine Learning, pp 293-301.*

Vafaie, H. and K. de Jong. 1992. Genetic algorithms as a tool for feature selection in machine learning. In *Machine Learning, Proceeding of the 4th International Conference on Tools with Artificial Intelligence, pp 200-204.*

Wettschereck, D. and D. Aha. 1995. Weighting features. In *Proceedings of the First International Conference on Case-Based Reasoning, ICCBR-95, pp 347-358.*

Wettschereck, D., D. Aha, and T. Mohri. 1997. A review and empirical evaluation of feature weighting methods for a class of lazy learning algorithms. In *Artificial Intelligence Review Vol.11, pp 273-314.*

Wilson, D. and T. Martinez. 1996. Instance-based learning with genetically derived attribute weights. In *Proceedings of the International Conference on Artificial Intelligence, Expert Systems, and Neural Networks, pp 11-14.*

Yang, J. and V. Honavar. 1997. Feature subset selection using a genetic algorithm. In *Genetic Programming 1997: Proceedings of the Second Annual Conference, pp 380.*

Appendix

The following tables show which feature subset and parameter values were found by each method. The first column represents the technique used for the features, the second the technique applied to set the parameters. In the third and fourth colums we respectively read the values for k and the weight settings. The remaining columns show the different values for the features.

LEGEND:

f = 0: the feature is absent from classification; f = 1: the feature is present during classification; f = 2: the MVDM is calculated for that feature

k = n: k can take any odd value between 1 up to 21

w = 0: no weighting is used; w = 1: gain ratio weighting is used; w = 2: information gain weighting is used; w = 3: chi-squared weighting is used; w = 4: shared variance weighting is used

Features	Parameters	k	w	f	f	f	f	f	f	f	f	f
Default	Default	1	1	1	1	1	1	1	1	1	1	1
Default	Exhaustive	7	0	2	2	2	2	2	2	2	2	2
Evolutionary	Default	1	1	1	2	2	0	1	2	2	2	2
Backward	Default	1	1	1	2	2	0	1	2	2	2	2
Forward	Default	1	1	1	2	2	0	1	1	1	2	1
Evolutionary	Evolutionary	9	1	1	2	2	2	1	2	1	1	1
Backward	Backward	3	1	1	2	2	1	1	2	1	2	2
Forward	Forward	3	1	1	2	2	1	1	1	1	2	1

Table 6: POS unknown words tagging task.

Features	Parameters	k	w	f	f	f	f	f	f	f	f	f	f	f
Default	Default	1	1	1	1	1	1	1	1	1	1	1	1	1
Default	Exhaustive	3	0	2	2	2	2	2	2	2	2	2	2	2
Evolutionary	Default	1	1	0	2	2	2	2	2	0	0	1	2	1
Backward	Default	1	1	0	1	2	2	2	1	0	1	0	0	2
Forward	Default	1	1	0	1	2	1	2	1	0	1	0	0	1
Evolutionary	Evolutionary	3	3	0	0	2	2	2	2	0	1	2	2	0
Backward	Backward	3	2	0	2	2	2	2	2	0	1	0	0	2
Forward	Forward	3	2	0	1	2	1	2	1	0	1	0	0	0

Table 7: POS known words tagging task.

Features	Parameters	k	w	f	f	f	f	f	f	f
Default	Default	1	1	1	1	1	1	1	1	1
Default	Exhaustive	5	1	2	2	2	2	2	2	2
Evolutionary	Default	1	1	1	1	2	2	2	1	1
Backward	Default	1	1	1	1	2	2	2	1	1
Forward	Default	1	1	1	1	1	1	1	1	1
Evolutionary	Evolutionary	3	1	1	2	2	1	2	2	1
Backward	Backward	1	4	1	1	2	2	2	1	1
Forward	Forward	1	1	1	1	1	1	1	1	1

Table 8: Grapheme to phoneme conversion with stress (GS).