Studying feature selection methods applied to classification tasks in natural language processing

by

Katalin Pajkossy

A Thesis Submitted to the Graduate Faculty
of Eötvös Loránd University of Sciences in Partial Fulfillment
of the
Requirements for the Degree

Master of Science

Budapest, Hungary

2013

Studying feature selection methods applied to classification tasks in natural language processing

by

Katalin Pajkossy

Approved:

Major Professor: Dr. András Kornai

Committee:

Electronic Version Approved:

Dean of the Graduate School
Eötvös Loránd University of Sciences
January 2013

Table of Contents

Page

Introduction . . . 1

1 Preliminaries . . . 3

1.1 Classification based on a probabilistic model . . . 3

1.2 Logistic regression . . . 4

1.3 Feature selection methods . . . 5

1.4 Evaluating classifiers . . . 6

1.5 Evaluating feature selection methods . . . 7

1.6 The software used . . . 7

2 Classification to part-of-speech categories . . . 8

2.1 The task . . . 8

2.2 The feature set . . . 8

2.3 Collecting appropriate training data . . . 10

2.4 Results of binary classification . . . 11

2.5 Results of multinomial classification . . . 16

2.6 Testing different regularization methods . . . 18

2.7 Summary of results . . . 21

3 Elision in Hungarian nouns . . . 23

3.1 The task . . . 23

3.2 Results of χ^2 selection . . . 24

3.3 Results of brain damage selection . . . 25

3.4 Results of LASSO selection . . . 26

3.5 Summary of the results . . . 27

4 Noun phrase chunking . . . 28

4.1 The task . . . 28

4.2 The HunTag sequential tagger . . . 28

4.3 Training data . . . 29

4.4 Feature set . . . 29

4.5 Results of χ^2 selection 30
4.6 Results of L_1 selection 30
4.7 Comparison of different methods with identical number of features . . . 32
4.8 Summary of results 33

Bibliography 34

Introduction

In this thesis we discuss the problem of *feature selection* in machine learning (ML) on algorithms which solve certain tasks of basic importance in natural language processing (NLP). Machine learning algorithms operate by making generalizations based on a set of examples, called training data. We study supervised learning, which is the most important ML method of NLP. Here the training data set contains labeled data points characterized by a set of features, and the task is to construct a function which predicts the appropriate label. In the statistical approach both the features and the labels are treated as random variables. For the vast majority of NLP tasks the currently best working algorithms are based on statistical models.

The feature selection problem, also known as the problem of *variable (subset) selection*, arises when the number of attributes that can be measured for a single data point is very large. Important examples in the field of NLP include document classification, where the appearance of any word in a document can be considered a feature, and speech recognition, where speech records consisting of typically 3-5 sounds (phonemes) per second are digitized typically with 16 bits resolution and with a sampling rate of 44.1 kHz, yielding about ten thousand 16-bit features per phoneme, while linguistic theory requires only about twenty 1-bit features (called *phonological features*, see Chapter 3). In general, the feature set is to be narrowed for three main reasons. First, to reduce computational resources (time and memory). Second, to improve the accuracy of the predictive model by dimensionality reduction and the removal of irrelevant features. Third, to find the relevant features and thereby constructing a more interpretable model.

In this work we study the use of feature selection methods for several classification problems. In these problems the data points are words, and the classes are categories of linguistics. The feature sets reflect certain properties of the words and their context within the actual texts. The problem of the very large number of irrelevant features emerges naturally in these tasks.

The thesis is structured as follows. Chapter 1 contains a brief overview of the notions regarding the problem studied in this work. In Section 1.1 we start with the basic notions of Statistical Learning Theory following [20]. Afterwards we summarize the principles of practical machine learning algorithms. In Sect.1.2 the method of logistic regression is described.

In Sect. 1.3 we review the main feature selection methods, based on [6], [8] and [4] and also discuss the techniques used in this work in more detail. In Sections 1.4-1.6 we review the methodology and software used in this work. We do not define the linguistic notions that we rely on precisely, but we add references as we go along.

In Chapter 2 we discuss a part-of-speech (POS) classification task, where originally there have been about 2,100 features. As we show, it is sufficient to use only about 200 features to obtain comparable or even better results.

In Chapter 3 a binary classification problem is discussed, that of distinguishing certain Hungarian stems like *piszok*, which undergo elision under certain conditions, e.g. in accusative form, (consider *piszkot, *piszokot* – ungrammatical forms are denoted by a prepended asterisk) from those which do not (e.g. *billog,* cf. *billogot, *billgot*). A logistic regression model, trained to solve this task ([16],[17]) relies on over a million features. We conclude that a small portion, only 1,500 features, are sufficient to train high accuracy classifiers.

In Chapter 4 we discuss the so-called shallow parsing or noun phrase-chunking task, where we start with 2.4 million features, and succeed in reducing this by 95%.

Acknowledgments

The help and advice of my advisor, Dr. András Kornai, and of all members of the Human Language Technology group (MTA SZTAKI) are highly appreciated.

Chapter 1
Preliminaries

1.1 Classification based on a probabilistic model

The *classification* task is defined as follows. Let S be a set of data points, with each data point being characterized by a vector $\mathbf{x} \in X$ and by a value $y \in -1, 1$. The task is to construct a function $g : X \to Y$ predicting the label of future data points. In the current work we assume X to be $\mathbb{R}^m$, and we restrict our attention to functions of the form

$$g = sign(\mathbf{w}^T\mathbf{x}). \tag{1.1}$$

Here $f_w = \mathbf{w}^T\mathbf{x}$ is called a *predictor function*. The algorithms studied here are based on a probabilistic model; $P_{\mathbf{x},y}$ is an unknown probability distribution over $X \times Y$ and $S = \left\{\mathbf{x}, y\right\}_{i=1}^N$ is a sample drawn from $P_{\mathbf{x},y}$. For given risk function $V : \mathbb{R}^2 \to \mathbb{R}$ the task is to find a predictor f_w minimizing the *expected risk*

$$I_{f_w} = \int_{X \times Y} V(f_w(\mathbf{x}), y) P_{\mathbf{x},y} \, \mathrm{d}\mathbf{x} \, \mathrm{d}y. \tag{1.2}$$

The minimizer f_0 is called the *target function*. Since $P_{\mathbf{x},y}$ is unknown, the target function can not be found in practice. An estimation of the expected risk is provided by the *empirical risk*

$$I_{emp}[f_w, N] = \frac{1}{N} \sum_{\mathbf{x},y \in S} V(f_w(\mathbf{x}), y). \tag{1.3}$$

For the *estimation error* $Err_{f_w,N} - I_{emp}[f_w, N]$, several probabilistic upper bounds apply. The classical result is the following:

$$Err_{f_w,N} < \sqrt{\frac{(m+1)\ln\frac{2eN}{m+1} - \ln\frac{\nu}{4}}{N}} \tag{1.4}$$

with probability at least $1 - \nu$. This is an instance of a class of more general bounds, called *Vapnik - Chervonenkis bounds* (see [20]).

We now briefly review the terminology used in this work. S is called the *training data*, containing *training instances*. Components of $\mathbf{x}$ are called *features*, y is called the *label*. A

learner or learning algorithm is a triple $L = \langle F, V, M_{V,F} : \mathcal{S} \to F \rangle$; where F is the set of possible predictors called the *hypothesis space*, V is a risk function, and the domain of L contains the possible sets of training data. The process of finding

$$M(S) = \operatorname*{argmin}_{f \in F} \sum_{\mathbf{x}, y \in S} V(f(x), y) \tag{1.5}$$

is called *training*. The output of the whole process (which corresponds to Eq.1.1) is called a *classifier*. Classifiers using a linear predictor function are called *linear classifiers*. In practical applications large estimation error (called *overfitting*) emerges as one of the central problems. As Eq. (1.4) suggests, it is most likely to occur when the dimensionality of $\mathbf{x}$ is large compared to the size of the available data. The learner *using a subset I of features* is $\langle F', V, M' \rangle$, where the change is obtained by the hypothesis space containing predictors of the form

$$f = \sum_{x_i \in I} w_i x_i. \tag{1.6}$$

The *regularized* version of learner L is $\langle F, V', M' \rangle$. Here the change is obtained by adding a *regularization term* to the risk function. The resulting risk function is of the form

$$V'(f_{\mathbf{w}}(x), y) = \lambda p(\mathbf{w}) + \sum_{\mathbf{x}, y \in S} (V(f_{\mathbf{w}}(\mathbf{x}), y)), \tag{1.7}$$

where p is called the *penalty function*, with parameter λ. The use of regularization to avoid overfitting is common when training linear classifiers [21]. In this work we use $\|.\|_2^2$ (L_2) and $\|.\|_1$ (L_1) as penalty functions.

1.2 Logistic regression

In this work we use linear classifiers obtained by regularized logistic regression. Logistic regression is characterized by the use of risk function

$$V(\mathbf{x}^T \mathbf{w}, y) = \frac{1}{1 + e^{-y \mathbf{x}^T \mathbf{w}}} \tag{1.8}$$

For practical reasons we minimize the logarithm of this function, as it will have the same minimum point, so we are looking at

$$\operatorname*{argmin}_{\mathbf{w}} \sum_{\mathbf{x}, y \in S} -\log(1 + e^{-y \mathbf{x}^T \mathbf{w}}). \tag{1.9}$$

The case of $n > 2$ classes can be dealt with using a combination of binary classifiers. A method for this, called *one-versus-the-rest* (OVR), is the use of n classifiers, each separating

one of the classes from the union of the other classes. Let $\mathbf{w}_i$ denote the normal vector of the hyperplane corresponding the ith class. A new point $\mathbf{x}$ is assigned to the class for which the value $\mathbf{x}^T\mathbf{w}$ is maximal. Another method, called *one-versus-one* (OVO), is the use of $\binom{n}{2}$ classifiers, one for each pair of classes. A new data point $\mathbf{x}$ will be assigned to the class having the most votes.

For an arbitrary data point $\mathbf{x}$, based on the value of $f(\mathbf{x})$ a probability value of $\mathbf{x}$ belonging to either of the classes can be assigned thus:

$$p(y = 1|\mathbf{x}) = \frac{e^{-\mathbf{x}^T\mathbf{w}}}{1 + e^{-\mathbf{x}^T\mathbf{w}}} \tag{1.10}$$

$$p(y = -1|\mathbf{x}) = \frac{1}{1 + e^{-\mathbf{x}^T\mathbf{w}}} \tag{1.11}$$

With n classes, the use of multiple logistic regression according to the OVR method provides the following probability values:

$$p(y = i|\mathbf{x}) = \frac{\frac{e^{-\mathbf{x}^T\mathbf{w}_i}}{1+e^{-\mathbf{x}^T\mathbf{w}_i}}}{\sum_{j=1}^{n} \frac{e^{-\mathbf{x}^T\mathbf{w}_j}}{1+e^{-\mathbf{x}^T\mathbf{w}_j}}} \tag{1.12}$$

1.3 Feature selection methods

For a particular learning algorithm the *feature subset selection* task is to find a feature subset I minimizing the expected risk of the classifier trained with the use of I. Feature selection algorithms solving this task can be classified according to how they interact with the chosen learning algorithm.

1. Wrappers use the learning algorithm as a black box to evaluate feature sets while performing a heuristic search in the space of all possible subsets. At each step of the search the classifier trained using the current set is evaluated, which yields the evaluation of the current subset. Several widely used methods for evaluating classifiers will be discussed in Sect.1.4. Wrapper methods vary in the search strategy used [9].

2. Embedded methods perform variable selection as part of the training process. A class of embedded methods perform a search procedure in the space of feature subsets. The search is guided by the estimated change in the value of the risk function. In this work we use a simple version of such a technique, called *brain damage* or *brain surgery* [4]. We start with the whole feature set, train a classifier using it, and retain the features which have large parameter weights. Another way of embedded feature selection is the addition of a *sparsity*

term to the risk function, which enforces a subset of parameters to be exactly zero at the optimum. An important example for sparsity term, also used in this work is $\lambda\|.\|_1$. The feature selection technique obtained with it is called LASSO, for "Least Absolute Shrinkage And Selection Operator" [19]. The optimal value of λ, which controls the number of selected features, can be determined using cross-validation.

3. Filter methods work prior to the learning phase itself, independent of the chosen learning algorithm. A class of filter methods searches for a small subset of features which determine the labels on the training instances [9]. Other, more simple filter methods score features according to their individual relevance, estimated from the training sample with the use of some statistical test. The selected subset is the union of the features with the highest scores. In this work we use filters based on χ^2 values as a baseline in order to evaluate embedded methods.

1.4 Evaluating classifiers

Classifiers are usually evaluated by their performance on a *held out* test data set, which has not been used for training. We use two percentage quantities for measuring performance. For evaluating multinomial classifiers and for evaluating binary classifiers on a test data set containing similar number of instances from the two classes (Sect.2), we use *accuracy*, defined as

$$\frac{C}{N} \cdot 100\%, \tag{1.13}$$

where C denotes the number of correctly classified instances, and N denotes the number of all instances in the test set. For evaluating binary classifiers on a test data set containing sharply different number of training instances from the two classes (Sect.3), we use *F-score*, defined as the harmonic mean of two quantities. The first quantity is *recall*, defined as

$$\frac{TP}{TP + FN} \cdot 100, \tag{1.14}$$

where TP, for "true positive" denotes the number of instances correctly classified into class 1, and FN, for "false negative" denotes the number of instances incorrectly classified into class 0. The second quantity is *precision*, defined as

$$\frac{TP}{TP + FP} \cdot 100, \tag{1.15}$$

where TP denotes the same quantity as above, and FP, for "false positive" denotes the number of training instances incorrectly classified into class 1.

If there are only a few labeled training data points available, the usual process of evaluation, called cross-validation ([1]) is to perform several splits on the training data to distinct training and test sets, train and evaluate a classifier on these, and average the results in the end. An important variant for this technique is called *n-fold cross-validation*, which is the case when the training data is divided into n parts, and the results are obtained by training n models on $\frac{n-1}{n}$ of the data and evaluated on the remaining $\frac{1}{n}$ part. In this work we use this method with $n = 10$.

1.5 Evaluating feature selection methods

A feature selection algorithm is evaluated by evaluating the classifier which was trained using its output. When evaluating the classifier, at each validation step the training and test sets are separated and only the training part is used as input for the feature selection method. All feature selection methods used in this work have a parameter which determines the size of the feature set retained. For the version of brain damage used here it is simply the number of features retained after the first training, for L_1 regularization it is the parameter λ, and for χ^2 test it is the critical value for rejecting the null hypothesis of independence.

In Chapter 2 our main concern is to evaluate feature sets of identical sizes provided by different feature selection methods, and we disregard the question of finding the optimal parameter of the method in focus. In Chapter 3 we use distinct development and evaluation sets for setting the parameter of the feature selection method and for evaluating the resulting model.

1.6 The software used

The binary logistic regression models were trained using the Python interface of the LIBLINEAR Library. LIBLINEAR is an open source library for large scale classification [5]. Multinomial feature selection with the OVO method and χ^2 feature selection were implemented using the Scikit Python module.[11].

Chapter 2
Classification to part-of-speech categories

2.1 The task

In this chapter we review the process of classifying word forms to different word categories (parts-of-speech, POS). We considered two types of POS categorization. The first is the adjective/adverb/noun/verb POS system of traditional English grammar.

The second is a more detailed POS system containing 36 categories. A subset of these is the refinement of the traditional categories, (e.g. noun and noun in plural are distinct categories), the remaining are linguistic classes of function words (e.g. preposition, pronoun). This feature set was used when annotating the the Penn Treebank [10]. Treebanks are large sets of texts (called *corpora* by linguists), with linguistic annotation for each sentence, containing also the POS category of each word. The Penn Treebank is the most widely used English treebank, containing approximately 7 million POS-tagged words.

2.2 The feature set

The features used in this work to characterize words reflect their typical environments in English sentences. The idea behind using this feature set for POS-classification rests on the structuralists' concept of POS-categories. In structuralist theory, POS classes are equivalence classes of the *distributional equivalence* in the set of all sentences of a given language [7]. In other words, two words belong to the same POS category if they appear in the same environments. As information source on typical environments of words we used the Google NGram data set [2].

This data set contains frequency counts of word token sequences generated from text collected from public web pages. Word tokens are the result of the segmentation of a continuous text stream by means of *tokenization*: words, punctuation signs, and in some application sentence boundaries, or even expressions ('New York') count as distinct tokens. From the Google NGram data set we used the frequency counts of word tokens (*unigrams*) and word token sequences of length two (*bigrams*).

As environments, we considered the (immediate) left and right neighborhood of the 350 most frequent words in the Google statistics. These will be referred to as *diagnostic words*. The diagnostic words contain most of the English function words, punctuation signs, sentence boundaries, some numbers, dates, and several words related to the use of the web such as 'page' or 'search'.

For each diagnostic word we used two distinct feature sets for its proportion in the left and in the right environments of the word in focus. Let $F_b(w_1, w_2)$ denote the relative frequency of the sequence '$w_1 w_2$' in the bigram data, and $F_u(w_3)$ the unigram relative frequency of w_3. Let w_d denote the given diagnostic word, w_e the word examined. Based upon the value of

$$ratio_{left} = \frac{F_b(w_d, w_e)}{F_u(w_d) \cdot F_u(w_e)} \tag{2.1}$$

or

$$ratio_{right} = \frac{F_b(w_e, w_d)}{F_u(w_d) \cdot F_u(w_e)} \tag{2.2}$$

we assigned w_e one of three binary features, which indicate whether the word sequence composed of w_d and w_e (in one or the other order) is typical/not typical/unlikely in English. The upper and lower limits of 'not typical' were set to 5 and 0.2 manually (i.e. based on the author's subjective judgment). This method creates $6 \cdot 350 = 2100$ binary features.

Table 2.1: Some features assigned to *they*

left			right		
typical	not typical	unlikely	typical	not typical	unlikely
before	could	Web	did	list	'
did	not	!	know	who	same
what	–	16	do	then	24
as	us	8	are	support	3
do	have	5	could	first	We

Table 2.2: Some features assigned to *went*

left			right		
typical	not typical	unlikely	typical	not typical	unlikely
which	one	be	to	just	days
We	has	e	back	one	or
they	There	4	into	here	information
she	line	to	before	where	International
he	back	's	off	'	“

Since Google NGram does not contain bigram frequencies under 40, Eqs. 2.1 and 2.2 are not informative in cases when the frequency of the word in focus is low. That is why training instances of low frequencies were not always assigned one of the ‘unlikely/likely/typical’ triple for each diagnostic word. This also causes the feature set not to be completely redundant. We experimented with two different feature sets: the *wide* set specified above, and a *narrow* set that was obtained by omitting the ‘typical’ values, so that only 1400 features are left. As we shall see in Sect. 2.4, neither of the two proved to be clearly superior to the other, and when not stated otherwise, the test result reviewed refer to using the broad set.

2.3 Collecting appropriate training data

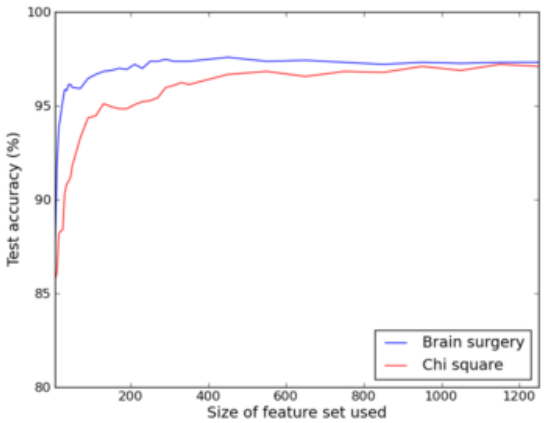
To avoid having training instances without any features, we considered word forms of large frequencies only; in the top 100 thousand unigram frequency in the Google NGram data set. Since our goal was not to develop a practical POS-labeling algorithm, we did not use word forms with more than one possible POS label (outside of a few experiments not reported here). These two factors heavily restricted the size of available training data for each used word class, although not uniformly. For the adverb/adjective/noun/verbs task the training data were taken from the central part of the vocabulary entries of the Longman Dictionary of Contemporary English, called the Longman Defining Vocabulary (LDV). We used word forms labeled here as having unambiguous POS - category. For the task of the Penn categories the training data were collected from the Penn Treebank, so that for a certain class we only used word forms having all instances annotated with the same POS - label. For readers not familiar with NLP we note that these issues do not arise when developing practical POS-labeling algorithms, which use all instances of word forms in a text as training instances.

2.4 Results of binary classification

We started our experiments by training binary models for separating each pair of the adverb/adjective/noun/verbs classes, and also for separating each word class from its complement. We used L_2 regularized logistic regression for training. At first we used the whole feature set, and evaluated the classifiers obtained with the use of the LDV training data. We used 3600 adjectives, 1000 nouns, 1000 verbs and 800 adverbs. The cross-validation results were between the limits 95.33% and 99.2%, and those classifiers performed slightly better which separated two distinct classes (between 97.8% and 99.2%), than those separating a class from its complement (between 95.33% and 97.81%). Next we performed the brain damage procedure (or BD, as defined in Sect.1.3,1.), while varying the size of the feature set retained. As Figs. 2.1 and 2.2 show, the cross-validation accuracy of the classifiers trained with 150 - 200 features was near the optimum (see panel (a) of Fig.2.1), reached the optimum (see panel (a) of Fig.2.1, and panel (b) of Fig.2.2), or even surpassed the crossvalidation accuracy of the classifier using the whole feature set (see panel (b) of Fig.2.2). The two figures also exemplify the slightly higher test accuracies of classifiers separating two distinct classes (see rightmost results in Fig.2.1), than those separating a class from its complement (see rightmost results in Fig.2.2). As a baseline we also performed χ^2 based feature selection on the same sets of training data. Classifiers trained using feature sets selected by BD performed on average 2-3% better than those trained using feature sets selected according to their high χ^2 values.

Figure 2.1: Results of BD and χ^2 selection

(a) Adverb classifier



(b) Noun classifier

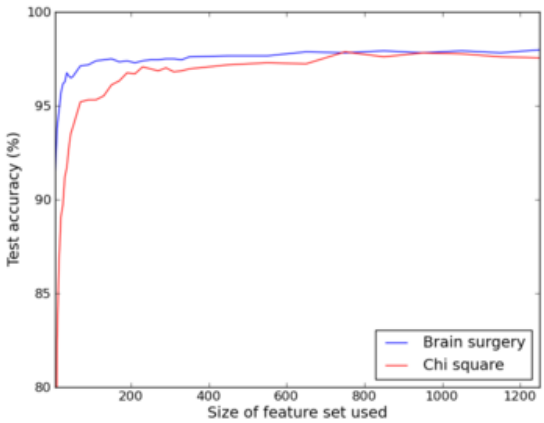
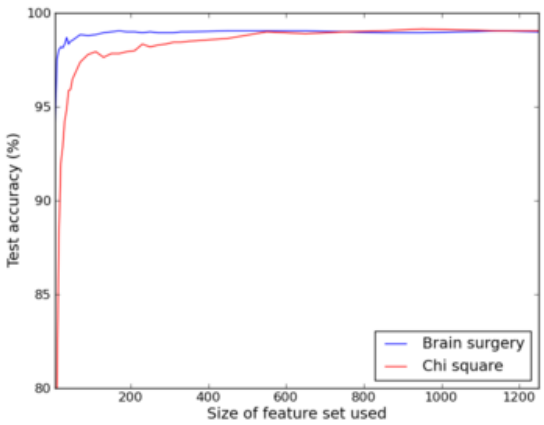
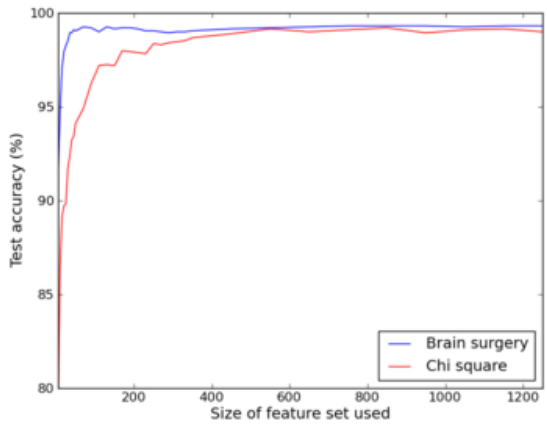


Figure 2.2: Results of BD and χ^2 selection

(a) Adverb - Noun classifier



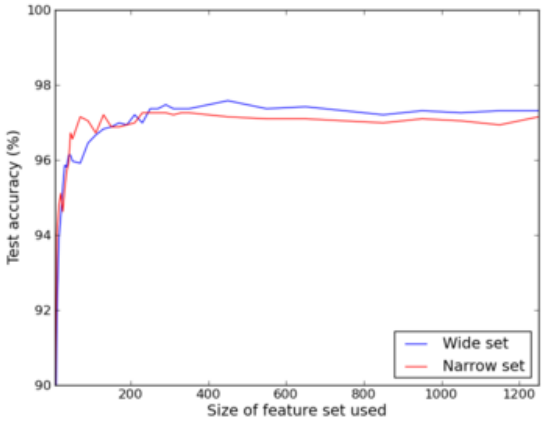
(b) Adverb - Verb classifier



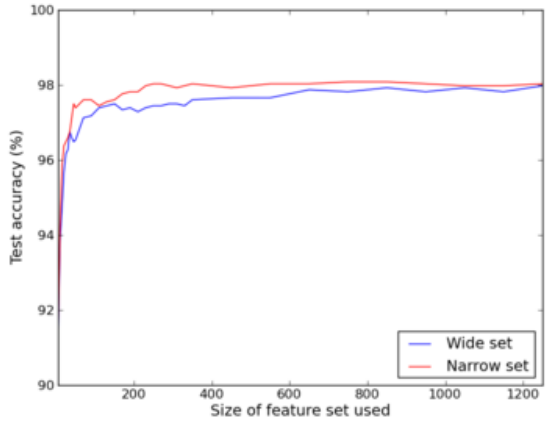
We repeated the same experiments with the use of the narrow feature set. Contrary to our expectations using the narrow set did not lead to a better result in all cases. Figure 2.3 shows examples of cases when the broad (see panel (a)), and when the narrow (see panel (b)) set proved to be more useful.

Figure 2.3: Results of using the broad and narrow sets prior to BD

(a) Adverb classifier



(b) Noun classifier

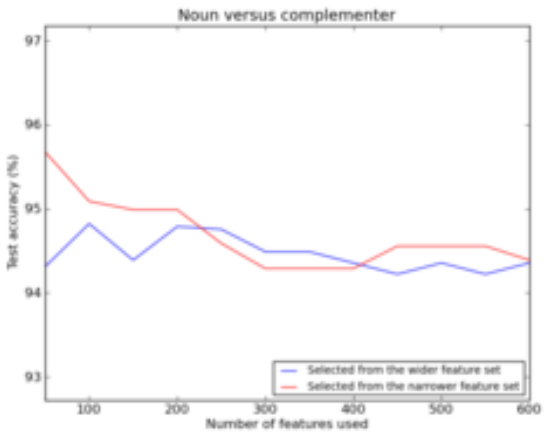


We compared the two sets when using few training data points. For these experiments we used 70 adjectives, 17 adverbials, 22 nouns and 20 verbs only. In order to achieve a more reliable evaluation of the methods, the experiments were performed with the use of distinct training samples (obtained by splitting the original training data into 16 parts), and in the

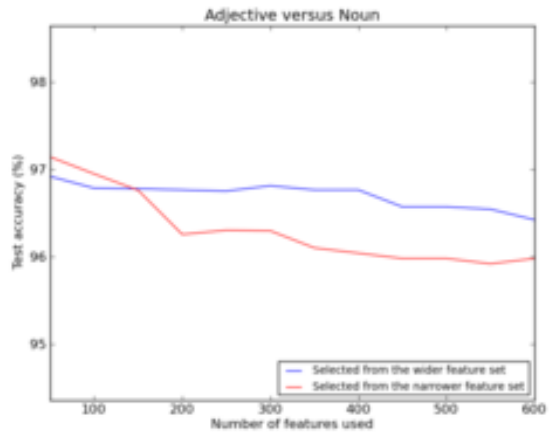
end the results were averaged. Again we found that neither of the two sets proved better for all cases. Figure 2.5 shows examples of this (as above, the red curve denotes the wide set and the blue the narrow set). As Figure 2.5 also shows, when using small training samples, classifiers trained on less features tested better, indicating overfitting.

Figure 2.4: Results of using broad and narrow sets prior to BD, using few training data

(a) Noun classifier



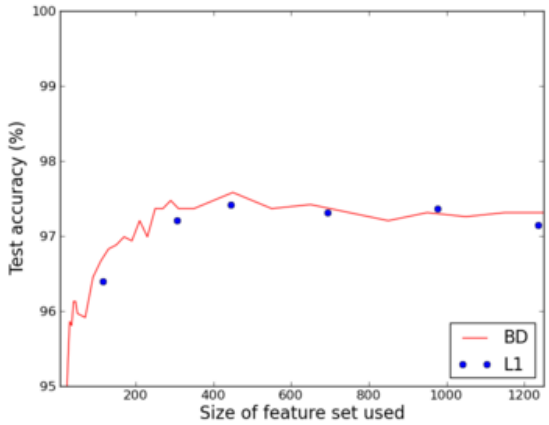
(b) Adjective-Noun classifier



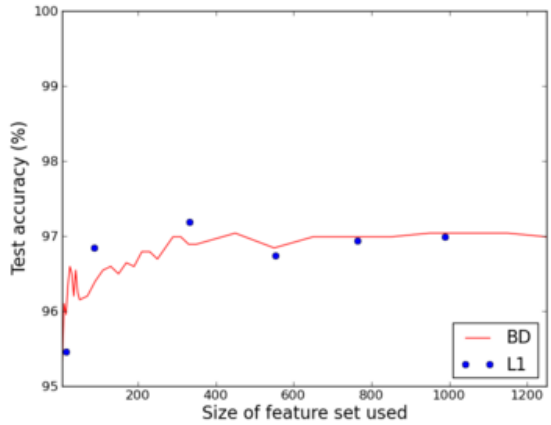
We evaluated another embedded feature selection method, L_1 regularization (as defined in 1.3,2.) on our whole LDV training data. Among the training data sets the L_1 selection proved superior on the adjective - non-adjective data sets to BD procedure. On the other pairs of data sets the two methods proved similar. We then evaluated the methods on smaller sets of adjective training data in order to test whether the difference is due to the different size of these data sets (originally we used 2·3600 training instances for adjective classification and cca. 2·1000 training instances for the other classes). On two subsets of the original adjective set (of sizes 1200) the L_1 method proved better. We found that the L_1 method was superior when having the value of the λ parameter between the limits when evaluated on larger training sets, see Fig 2.5 where the order (a-d) follows the increasing size of training sets (800, 1000, 1100, 3600).

Figure 2.5: Results of L_1 based feature selection and BD

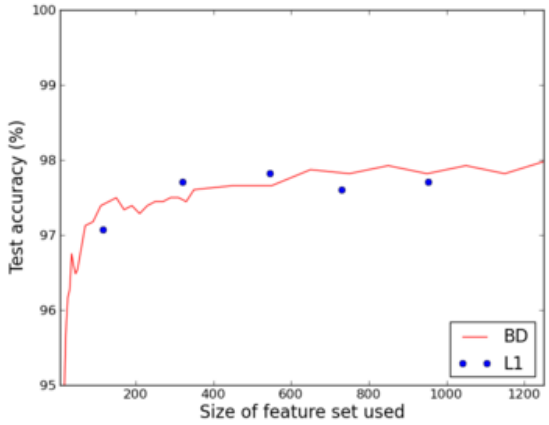
(a) Adverb classifier



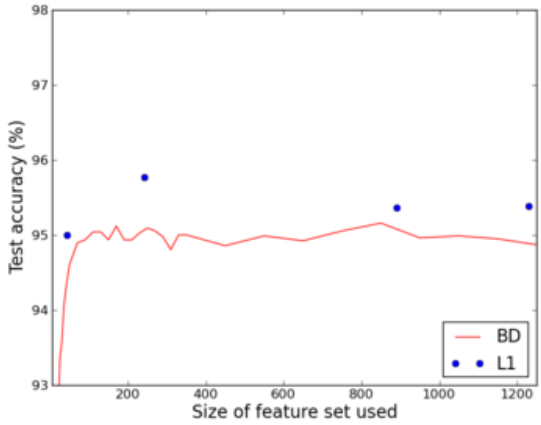
(b) Verb classifier



(c) Noun classifier

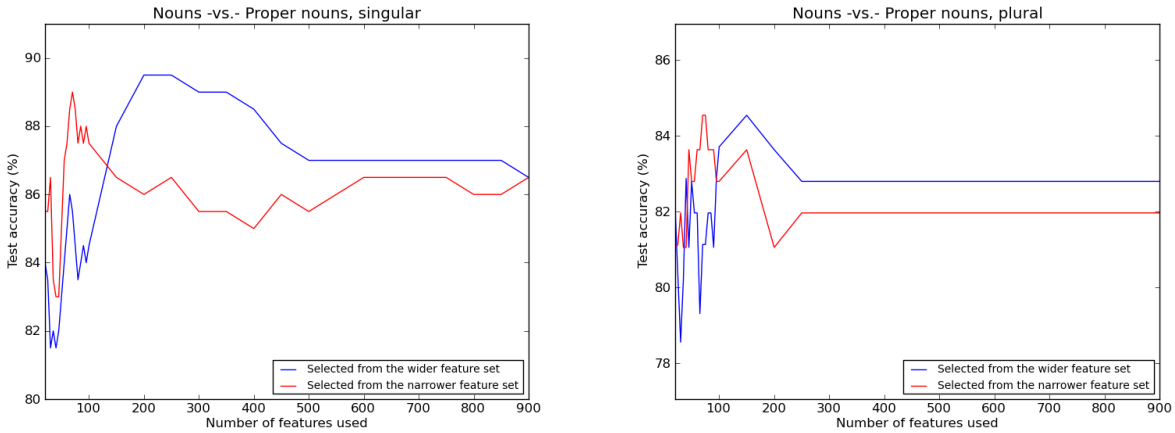


(d) Adjective classifier



In the Google n-gram data we had a sample of at least 10 items for 21 from the 36 categories. For training classifiers separating two POS categories we used these samples. For training classifiers separating POS-categories and their complements, the words in the complement set were taken from the Treebank's vocabulary, so that no instance of it is tagged with the corresponding POS-tag. For certain categories there were fewer training instances than 100 (Modal 30; Foreign 73; Determiner 10; Adjective,comparative 30; Preposition 40; Verb,non-3rd.person,singular present -30; Personal pronoun 24; Interjection 19; Proper noun, plural 15). For these we used all instances we had, for the other categories we used a training sample of size 100. For all complement classes we used a training sample of size 100. The figures below show some results of this procedure.

Figure 2.6: Separating Common nouns and Proper nouns



Some of these tasks are much harder than the basic POS classification tasks. This is clear from the results, which were in the 95-98% range for the basic tasks (see Fig. 2.5), but are 10% lower in Fig. 2.6. Often there is very little in the context that can help us distinguish common nouns from proper names, and humans performing the classification rely on information, real world knowledge, that is not encoded in the features. (A practical system would also use capitalization and other word-internal features.)

In other cases, such as singular vs. plural, word-internal features (e.g. whether the final character is *s*) would be very useful, but given the broad grammatical impact of plurality (carried to the verb by subject-predicate agreement) context is again largely sufficient.

Figure 2.7: Separating Nouns in singular and plural form

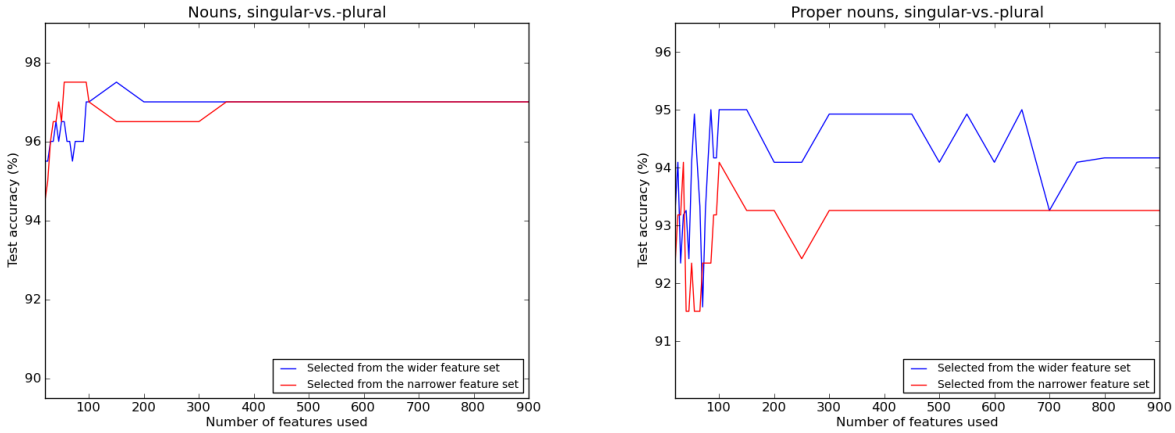
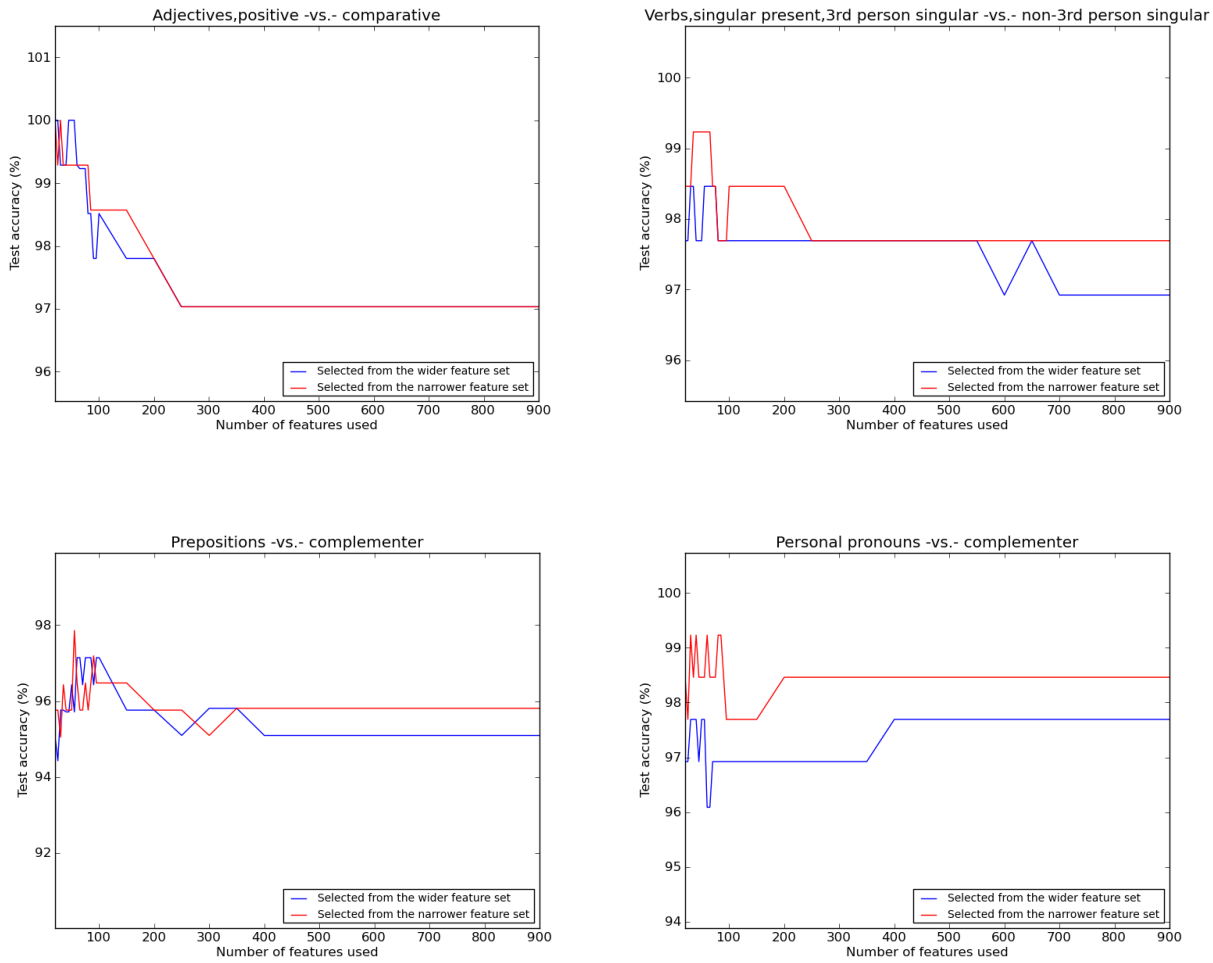


Figure 2.8: Overfitting



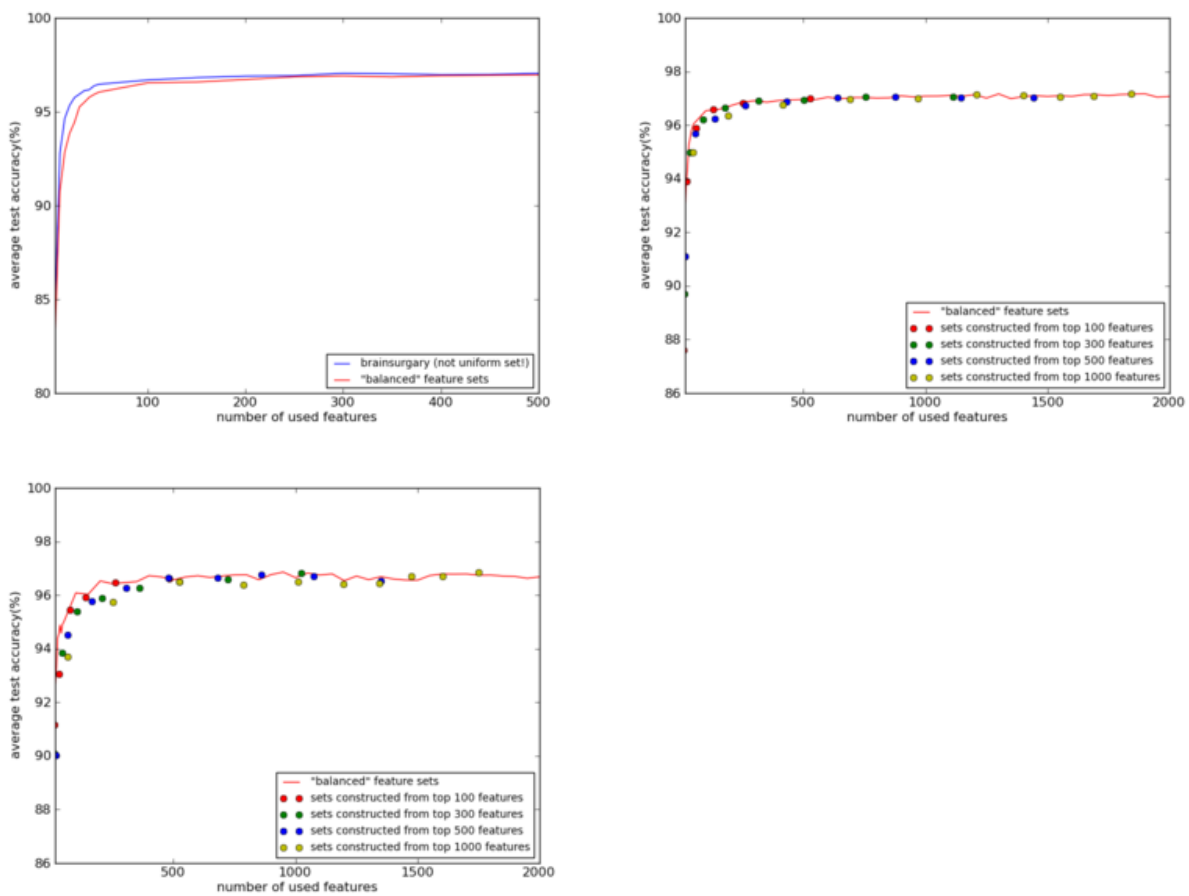
Finally, Fig. 2.8 shows the overfitting effect we discussed earlier: the best results are obtained when only 5% of the broad set (less than a 100 of 2,100 features) are used.

2.5 Results of multinomial classification

In this section we review our results by training multinomial models on the same data sets. We used multiple logistic regression. For multinomial feature selections we tested several simple heuristics of combining the scores which came from the binary models. In one experiment (Fig. 2.9, top left panel) we compared the features obtained from brain surgery to the 'balanced' feature set obtained by taking an equal number of the best features from each classifier. Since the difference is noticeable only when the number of features is low (less than 50), in the other two experiments (top right and bottom left panels) we used balancing as the baseline.

These two experiments, which differ only in the basic data sets, used the following feature selection algorithm: take the top k features in each pairwise classifiers ($k = 100, 300, 500, 1000$), and use only those that appear in at least n of these sets. Again, the differences are most noticeable when the overall number of features is low, compare e.g. the first yellow dot on the bottom panel to the second red dot: for the same absolute number of features selecting from the 'cream of the cream', the top 100 features of pairwise classifiers, is 2% better than selecting from the broad sets (top 1000 features).

Figure 2.9: Testing feature sets containing features selected by many binary models

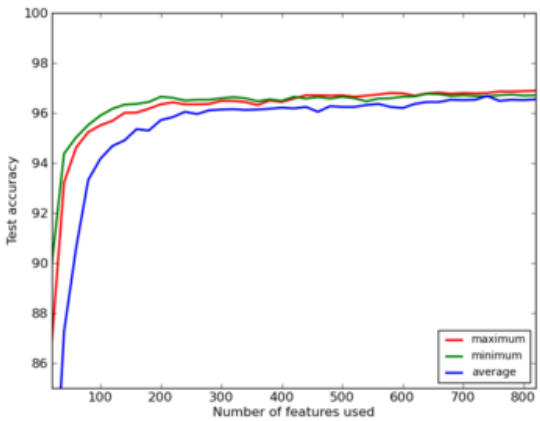


Another set of experiments used the absolute values, rather than the ranking, of the feature values in the pairwise classifiers. Let feature i obtain the weight w_{ij} in pairwise classifier j. Three strategies were tried: in the 'average' strategy (blue curves) we used $\sum_j w_{ij}$ as the figure of merit for feature i, in the 'maximum' strategy we used $\max_j w_{ij}$ (red curve), and in the 'minimum' strategy we used $\min_j w_{ij}$. In all cases, all 2100 features were ranked by

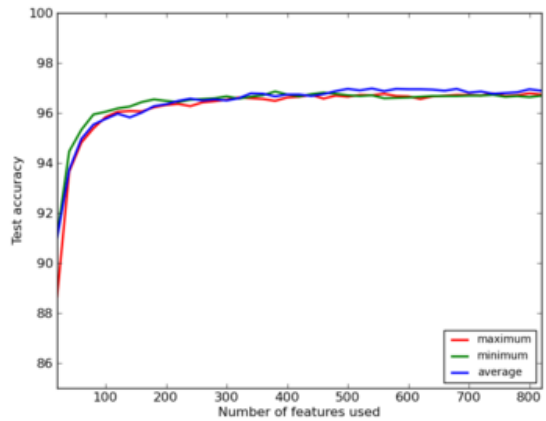
their figure of merit, and the final multinomial classifier kept the r highest ranked according to this figure.

Figure 2.10: Testing feature selection based on scores provided by logistic regression models

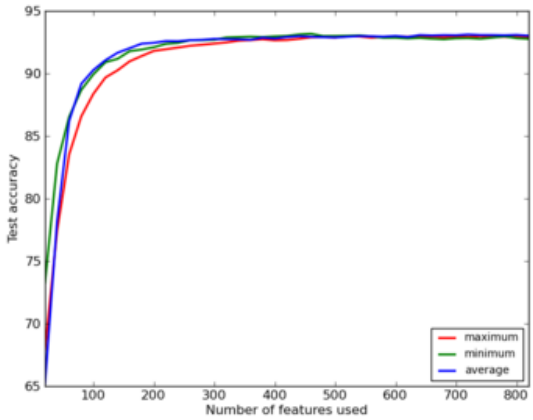
(a) Longman words, OVR method



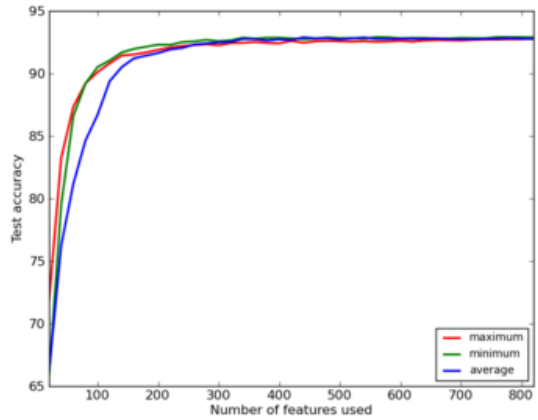
(b) Longman words, OVO method



(c) Penn words, OVR method



(d) Penn words, OVO method



In the region of interest (200 features or fewer) it is the 'minimum' method that appears to work best, and the 'average' method, somewhat surprisingly, is the worst.

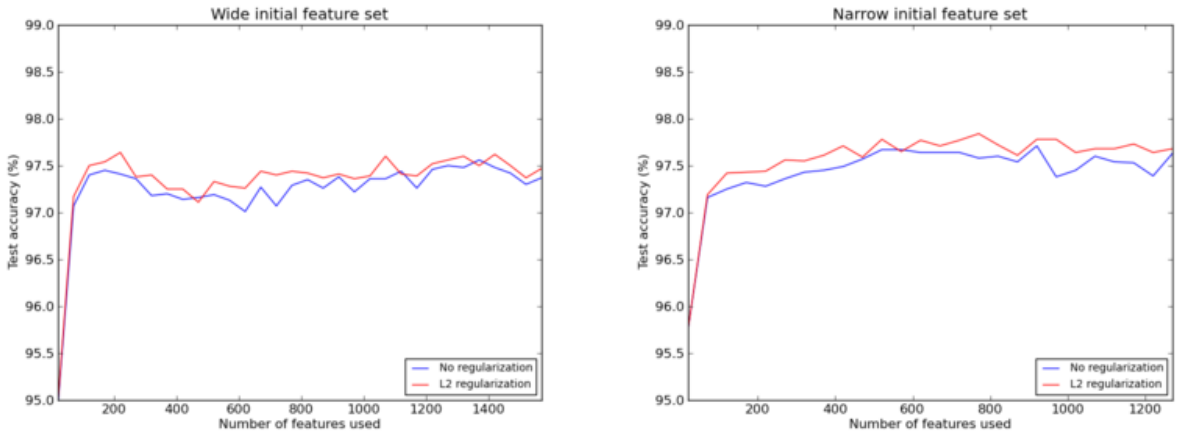
2.6 Testing different regularization methods

In this section we review results about testing different regularization methods. First we trained classifiers for separating nouns, using training samples of 5000 nouns and 5000 words

from other categories. The classifiers were tested with 10-fold cross-validation. In the 'A' condition (Fig 2.11, blue curves) the classifier was trained with all features and no regularization – this yielded 97.4% accuracy.

In the 'B' condition (Fig 2.11, red curves) the classifier was trained using L_2 regularization. Performance improved slightly, to 97.5%, see the right asymptotes in Fig. 2.11. The left panel gives the results for the wide, the right panel for the narrow feature set.

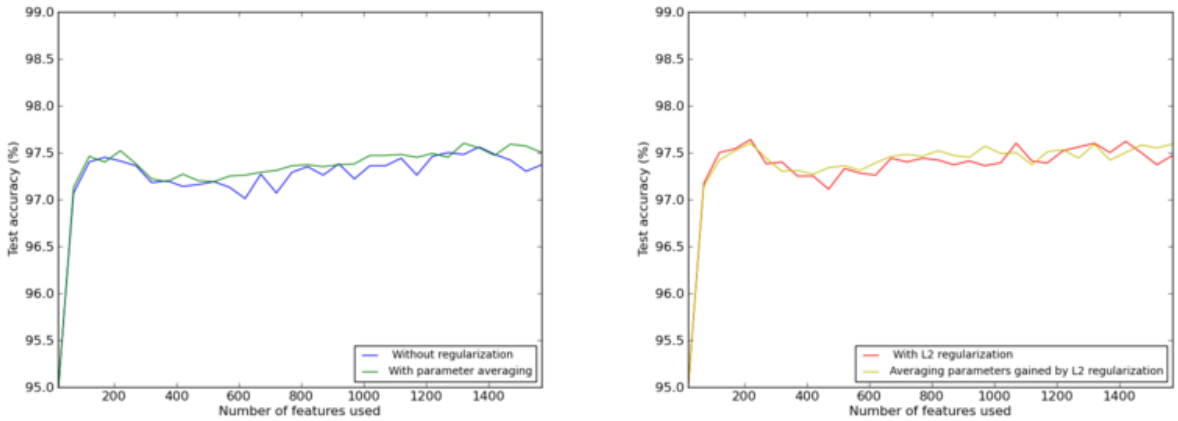
Figure 2.11: Testing the brain damage procedure



We performed the brain damage procedure based on the weights of model 'A', with different sizes of feature sets retained, and tested the classifiers obtained by using these feature sets. As Fig. 2.11 shows, retaining 200 features proved enough for reaching near optimum test accuracies, with or without regularization.

We tested other regularization methods and their combinations as well, while training models on feature sets reduced with the use of the brain damage procedure. The first regularization method we study, *parameter averaging*, is the following. The training data was divided into ten slices. Ten models, using all features, were trained on the union of all possible nine-tenths, and tested on the tenth slice. Then the parameter values of the five highest test accuracies were averaged to yield the parameter vector of the final model. The second method, *gaussian smoothing*, is the addition of the square of the L_2 norm as a regularization term to the objective function used at training.

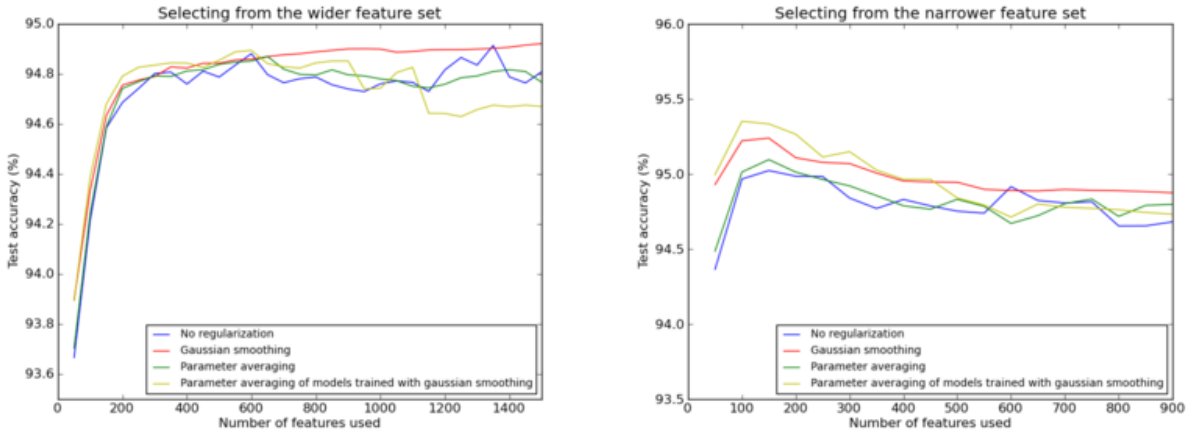
Figure 2.12: Accuracy of the models constructed by averaging parameter values



The effect of the above methods on the performance of the model was measured on the same binary classification task of noun vs. other. The left panel of Fig. 2.12 shows results with no regularization (blue curve) vs. parameter averaging (green curve), but with no L_2 regularization. The right panel shows L_2 regularization by itself (red curve) and in combination with parameter averaging (yellow curve). As can be seen, parameter averaging adds very little to the results, but makes them more robust.

Next we tested models which were trained in an extreme data-starved condition, only 20 samples, using feature sets of different sizes selected by the brain damage procedure. All models were tested on the same data set containing 5000 nouns and 3500 non-nouns. For each training data size the plots show the average test accuracy of models trained on distinct samples; 16 sets of size $2 \cdot 20$ were used.

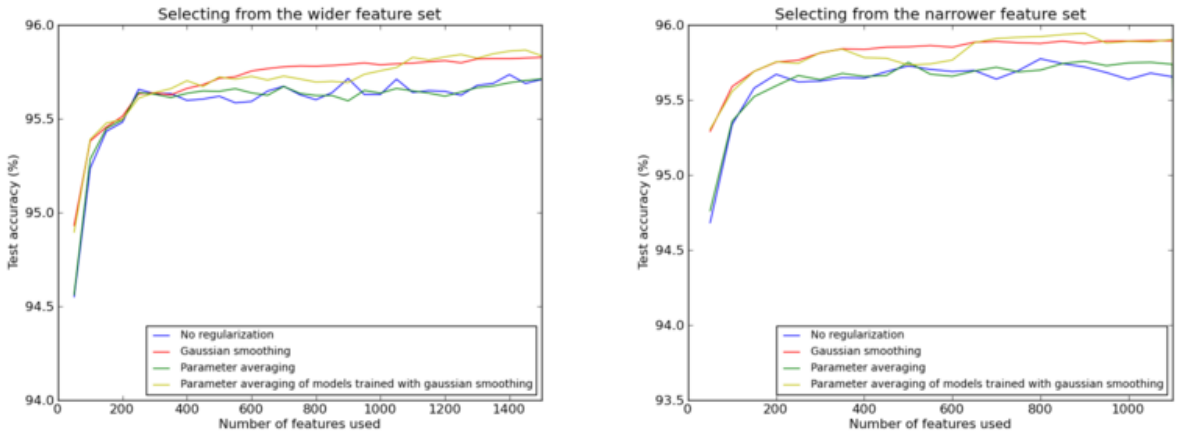
Figure 2.13: Regularization in extreme data-starved condition



In the left panel of Figure 2.13 we see compare the same regularization methods as in Figure 2.12, except we restrict the training set to 20 samples. When the number of features is not unrealistically large compared to the number of data points (200 features, the same number as in the full data condition), parameter averaging is helpful, but as the number of features begins to significantly exceed what the data would warrant, gaussian smoothing becomes more advantageous. In the right panel of Figure 2.13 we see the same effect, but performance declines even more rapidly once more than 10% of the features are used.

In a slightly less data starved condition (80 vectors, tested on 10 different sets and results averaged), we find similar results, see Figure 2.14: parameter averaging improved the test accuracy for small number of features. For a large number of features averaging the parameters (in particular parameters gained by gaussian smoothing) made the performance decline, while gaussian smoothing (without averaging) still improved test accuracy. The use of the different initial samples resulted in different size of optimum feature numbers; when using the wider initial set the results improved with the extension all along (with having reached the near-optimum at 250-300 features), while when using the narrower set the optimum is reached at 150 features, and with the addition of more features the performance declines. The use of the narrower initial feature set resulted in higher test accuracies.

Figure 2.14: Regularization in data-starved condition



2.7 Summary of results

The picture emerging from the above experiments is not uniform. In data-starved conditions, we find parameter averaging a reasonable approach, but if there is sufficient training data, gaussian smoothing makes more sense.

Unsurprisingly, the use of larger training samples resulted in higher overall test accuracies, in particular in case of the narrow feature set where the addition of more features improved performance all along (overfitting avoided). The effect of the regularization methods on the test accuracy had similar characteristics.

Chapter 3
Elision in Hungarian nouns

3.1 The task

In this chapter we discuss a binary classification task of separating noun stems according to whether they exhibit elision under certain conditions, e.g. in accusative form (the phenomenon is discussed in [17]). We study the restriction of a feature set containing binary features which characterize phoneme sequences of the suffix of the word stem in focus. The set of features corresponding to a sequence $a_{n-i}a_{n-i+1}\dots a_{n-j}$ equals

$$\bigoplus_{k=n-i}^{n-j} S_k \tag{3.1}$$

where S_k is a set of *phonological features* characterizing a_k. This feature set, and certain variations of it were used when training classifiers with the use of logistic regression, using a training set of 1075 syncopic and 48392 non-syncopic noun stems. Details on choosing the set of phonological features and the collection of the training data are discussed in [17]. The resulting classifiers were evaluated with tenfold cross-validation, using the F-score measure. The best cross-validation result was obtained when using a feature set containing only features corresponding to indices $i = 1, 2 \dots 7$ and $j = 0 \dots n-i+1$ [16]. feature set will be referred to here as the *short set*. There where over a million (1,057,667) features having value 1 on some instance of the training data, omitting those that were 0 everywhere filters this feature set down to 1,043,945 features.

In this work we use the original feature set as basis to feature selection. We compare L_1 feature selection, the brain damage procedure, and feature selection based on high χ^2 values. As a first step we held out $\frac{1}{4}$ of the training data, on which we later validated our classifiers, trained on the other $\frac{3}{4}$ part of the training data. The parts obtained by this split will be called *validation* and *development* data. For each method the parameter determining the size of the feature set was set using only the development set, with the use of ten-fold crossvalidation. After setting the parameter, we evaluated the classifier trained on the whole

development set, using a feature set obtained by the feature selection method parametrized with the optimal value.

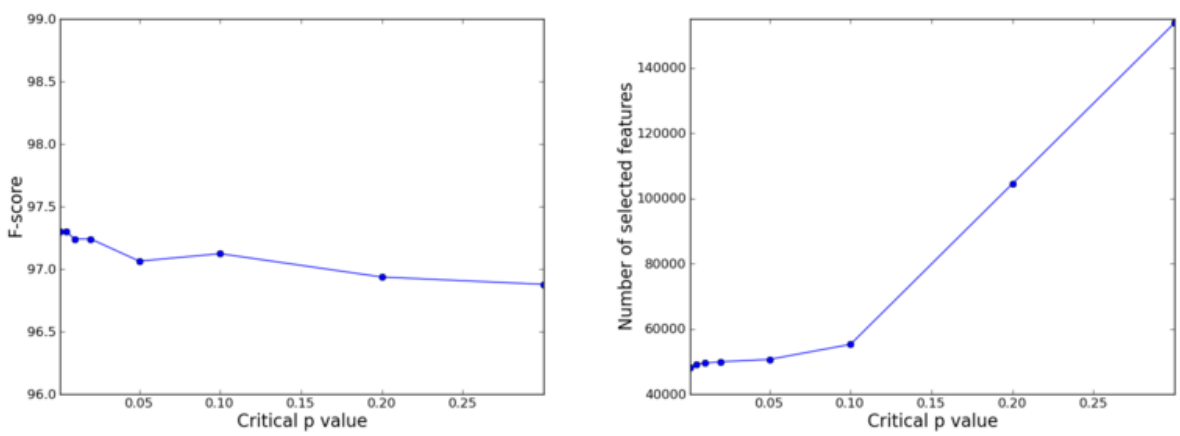
We compared the results to those obtained by the use of the whole, and the short feature set. On this split of the data there was no difference between the performance of the two classifiers trained on the whole and the short set, (97.7), and the cross validation score on the development data did not vary either (97.43).

3.2 Results of χ^2 selection

We started with critical values between the limits 0.001 and 0.3. As the right panel of Fig.3.1 shows, commonly used critical values for rejecting the independence hypothesis resulted in feature sets of sizes around 45-50 thousand. As left panel of Fig.3.1 shows, the best F-score results were obtained when using p=0.001 as critical value.

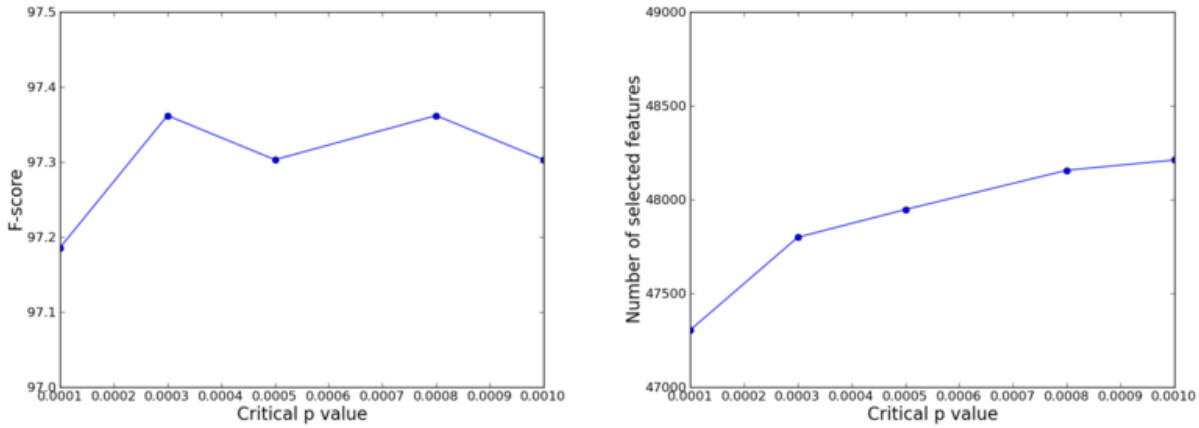
Figure 3.1: Cross-validation results on the development data

(a) F-score results (b) Sizes of selected sets



We proceeded with the use of critical values between the limits 0.00001 and 0.001. As the right panel of Fig.3.2 shows, the size of the selected feature set did not shrink considerably, nor did the cross-validation result change (as the left panel of Fig.3.2 shows).

Figure 3.2: Following cross-validation results on the development data



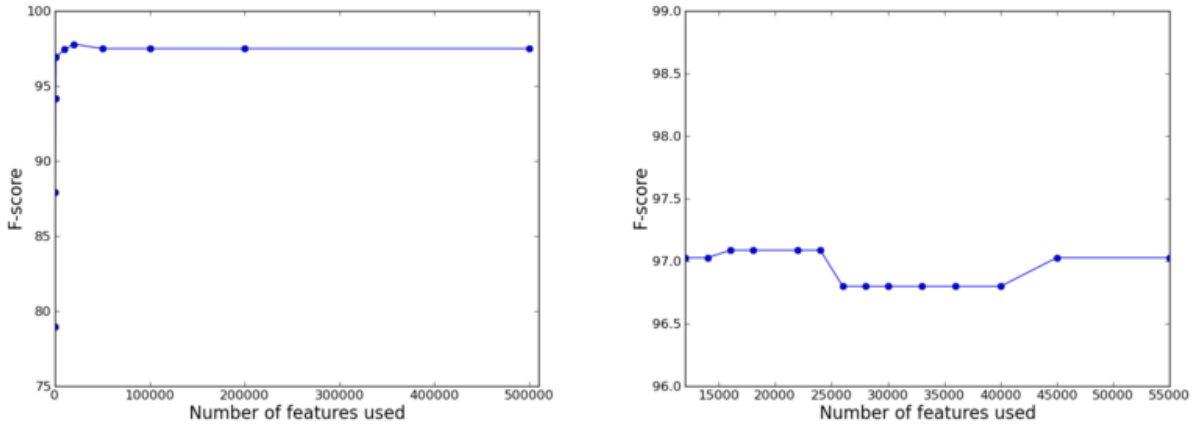
Based upon these results we chose the critical value to be 0.008. The classifier trained on the whole development data, using the feature set obtained by a χ^2 selection with this parameter value evaluated with F-score 97.41 on the validation data.

3.3 Results of brain damage selection

We started with feature sets of sizes between and 500 and 500,000. The left panel of Figure shows the resulting cross-validation results. Based on these, we continued with testing the retaining of feature sets of sizes between the limits 12,000 and 55,000. These results are shown in the right panel of Fig. 3.3.

Figure 3.3: Cross-validation results on the development data

(a) Retaining 500 to 500k features (b) Retaining 12k to 55k features



Based upon these results we trained a classifier on the whole development data, with the use of a feature set of size 20,000, selected by BD. The classifier evaluated with F-score 97.41 on the validation data.

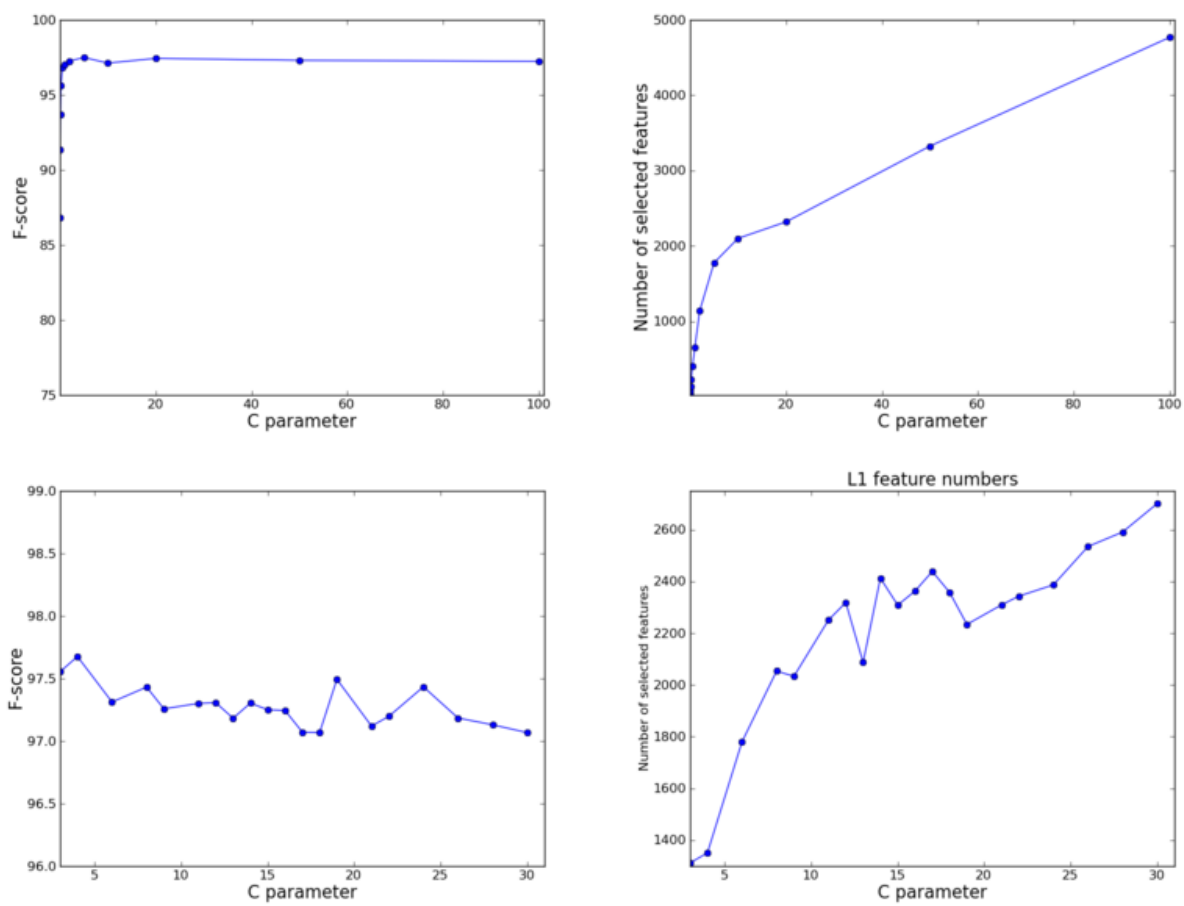
3.4 Results of LASSO selection

In the case of LASSO selection the parameter is the weight of the penalty term. The software package used in this work (see [5]) implements the minimalization of the objective function

$$g_{\mathbf{w}} = \|\mathbf{w}\|_1 + C \cdot \sum_{\vec{x},y \in S} (V(f_{\mathbf{w}}(\mathbf{x}), y)). \tag{3.2}$$

We started with C parameters between the limit and 100. Based upon the first results, (top left panel of Figure 3.4), we tested parameter values between the limits 3 and 30 (bottom left panel). As the right side panels show, the optimal results were obtained with the use of 2-3000 features only. For the final validation we chose the value of parameter C to be 4.

Figure 3.4: Cross-validation results on the development data



The classifier trained on the development data with the feature set selected by L_1 selection using parameter 4 (which corresponds to $\lambda = \frac{1}{4}$ according to what defined in Sect.1.3) evaluated with F-score 97.7.

3.5 Summary of the results

Table 4.1 summarizes our results. On this task, L_1 selection proved clearly superior to the other methods, providing as good classification results with 1/600th of the original features as the full set. Brain damage was second best, but it required an order of magnitude more features, and lost something in accuracy. χ^2, requiring twice as many features to reach the same accuracy as BD, was the worst.

While BD and χ^2 provided about as much reduction here (2-5% of the original set size) as in the POS task discussed in Chapter 2, L_1 normalization did much better on this task, possibly because the positive (syncopic) set is so much smaller than the negative set. We note that the methods of Recski and Rung obtain 97.5% on this problem.

Table 3.1: All results

Method	F on dev.data	#feat	Ft on val.data
no selection	97.43	960,000	97.70
χ^2	97.30	46,000	97.41
BD	97.08	20,000	97.41
L_1 sel.	97.67	1500	97.70

Chapter 4
Noun phrase chunking

4.1 The task

In this chapter we discuss the task of identifying maximal noun phrases in Hungarian text. A *noun phrase* is a group of words, containing a *head noun* which determines the behavior of the group as a unit in the sentence. Noun phrases can be embedded in each other, for instance both 'Pali öltönye' and 'a Pali öltönye mellett lévő kabát' are noun phrases. A noun phrase is called *maximal* if there is no other noun phrase containing it.

This is an instance of a more general task, called *text chunking*. Chunking is the task of dividing text into non-overlapping phrases so that syntactically related words are assigned to the same phrase [18]. Since the chunks are non-overlapping, the task is equivalent to the proper distribution of three labels, corresponding to 'beginning of chunk', 'inside of chunk' and 'not in chunk'. Some chunker methods use a refinement of this label system, either by differentiating/refining chunk types, or using a separate label for 'end of chunk'. In our work, we used the 8 labels that are present in the HunTag system [15]. The performance of the chunkers is evaluated using the F-score.

4.2 The HunTag sequential tagger

We use a software package developed for general sentence labeling tasks [15]. These applications are also called *sequential taggers*. HunTag is a *stochastic tagger*, that is, it works by assigning a probability value to the possible labelings of the input sentence, and choosing the label sequence with the highest probability value assigned. For assigning probability values a combination of multiple logistic regression and Hidden Markov Modeling or HMM [12] is used for modeling the language. Given an input sequence $w_1 w_2 ... w_n$, the label sequence $l_1 l_2 ... l_n$ is assigned the probability value

$$\prod_{i \in 1...n} p(l_i | w_{i-k} \dots w_{i+k}) \cdot \frac{P'(l_{i-1}, l_i)^\lambda}{P(l_i)^\lambda}. \tag{4.1}$$

The optimal labeling of a sequence is determined using the Viterbi algorithm. For more detail, see [15]. The parameters of the tagger - distributions P, P' and p, the integer k, and the value λ are to be determined with optimizing them using labeled training data. The P and P' distributions are obtained by frequency counts of labels and label pairs in the training data. The p distribution is obtained as a result of a classifier trained with the use of multiple logistic regression (as defined in Sect1.2). This classifier, assigning labels to words, uses features corresponding to the properties of words surrounding the word in focus, within the radius k. As we will see, this feature set can be large in practice and so the feature selection can improve the speed of the tagger. We measure the quality of the feature selection by the F-score of the classifier.

4.3 Training data

The training data for chunking was created using the Szeged Treebank [3], a corpus of Hungarian texts with manually verified syntactic and morphological annotation. For each token we extracted the available morphological analysis and converted it to KR-codes [13], a formalism that allows us to represent grammatical features of a word in a hierarchical structure. The corpus contains texts of various genres (journal articles, literature, essays written by high school students, etc.), all of which were included in both train and test datasets.

4.4 Feature set

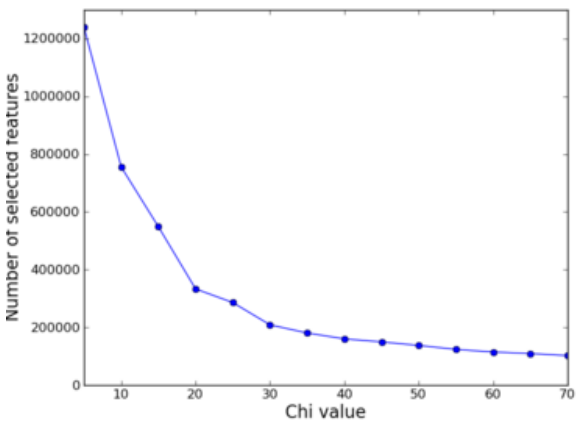
When converting the chunking task to a supervised learning problem, each word will be represented by features based on word form and morphological analysis. The structure of the KR-formalism makes it possible to represent each grammatical feature in a straightforward manner. A word with the KR-code `NOUN<CAS<ACC>><POSS>`, for example, will receive the features `kr=NOUN, kr=CAS_ACC,` and `kr=POSS`. Character trigrams of a word are also added to the representation. Finally, [14] also defines a feature that encodes information about the sequence of part-of-speech tags near a given word: "if a word in position i of a sentence is denoted by w_i and its POS-tag by p_i then the values for the POS pattern feature for w_i will be all subintervals of the series $p_{i-r} \dots p_{i+r}$."

4.5 Results of χ^2 selection

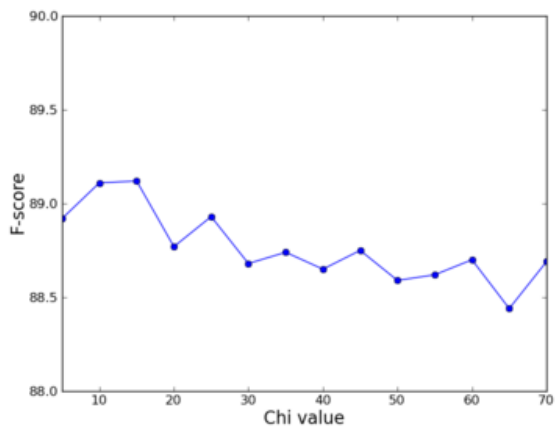
We started with selecting feature sets according to high χ^2 value. We started testing the selection defined by critical χ^2 values within the limits 5 and 75. After calculating the χ^2 values for each pair of features and labels, we retained all features which were assigned high χ^2 value indicating strong correlation with any of the labels.

Figure 4.1: Results of using χ^2 selection

(a) Number of features selected



(b) F-score results

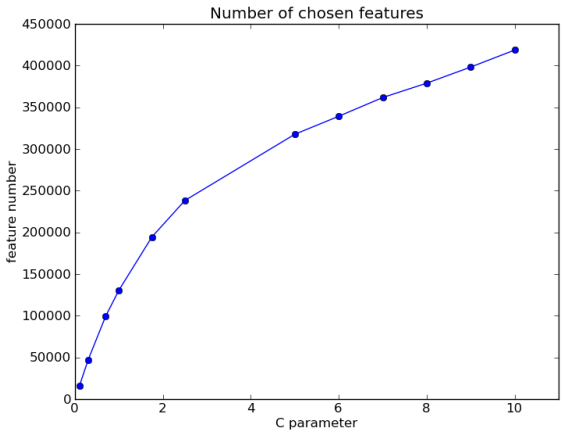


4.6 Results of L_1 selection

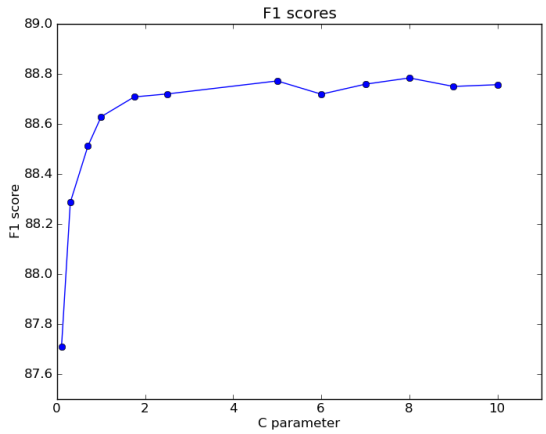
Next we selected feature sets with the use of L_1 regularization, for several C values. As in Sect.3.4 the parameter value C corresponds to $\lambda = \frac{1}{C}$ according to the definition in Sect.1.3. After the first multiple logistic regression training, we selected all features which were assigned nonzero parameter value by any of the resulting binary classifiers. We evaluated the taggers trained with the use of these feature sets. Fig. 4.2 shows the result of this experiment, obtained by tenfold cross-validation on our whole training data.

Figure 4.2: Results of using L_1 selection

(a) Average number of features selected



(b) Cross-validation F-score results

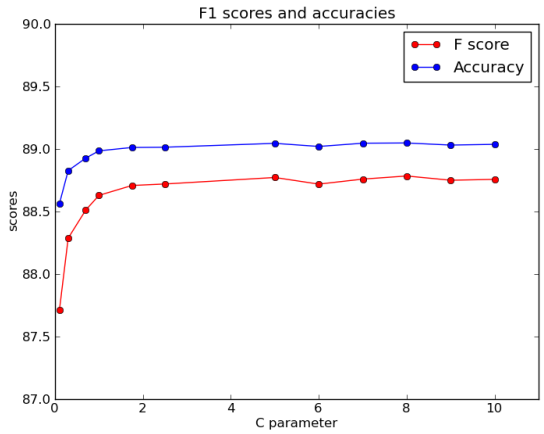


We evaluated a tagger on the test data, which was trained on the whole training data. The feature set was selected by the same method, based on a multiple logistic regression model trained with L_1 regularization, using the parameter value $C = 8$. We chose this parameter value based on the previous results. The tagger, using 40100 features only evaluated with an F-score of 88.86.

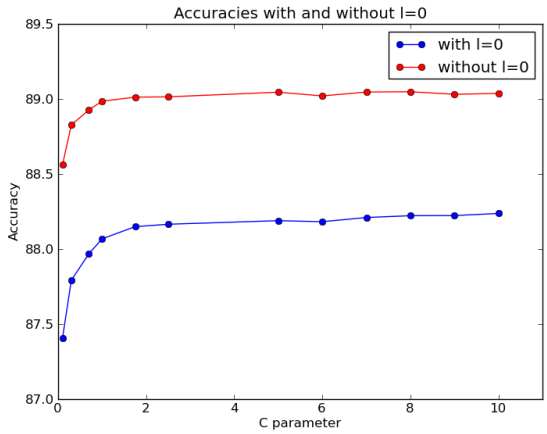
We investigated whether using accuracy or F-score as our main figure of merit makes any difference in the dynamics of feature selection, and concluded that only the absolute numbers change, the relative performances show the same dynamics (see the left panel of Fig. 4.3).

Figure 4.3: The impact of language modeling

(a) Accuracy and F-score as a function of C



(b) Effect of language model



The chunker used in these tests also has an option for combining the scores with an external HMM language model. Whether we turn this model on makes a noticeable difference in the absolute numbers, but again the relative performances are unaffected (see the right panel of Fig. 4.3).

4.7 Comparison of different methods with identical number of features

In the final set of experiments, we used the brain damage procedure (again using the average, maximum, and minimum figures of merit (see 2.5), this time for the 8 binary classifiers corresponding to the 8 tags used in HunTag. Note that neither choice leads a classifier as good as provided by L_1 selection.

Table 4.1: BD results with identical number of features (140k)

Method	F-score on test data
brain damage, average	88.46
brain damage, minimum	88.58
bran damage, maximum	88.37
L_1 sel at $C = 1$	89.01

4.8 Summary of results

Of the three feature selection methods tested, again L_1 normalization was the best, but the results from the brain damage procedure are comparable. χ^2 selection, while much easier to compute, provides inferior results. In term of CPU time, the best taggers improved by about 20%, from 114.3 to 91.5 seconds.

Bibliography

[1] S. Arlot and A. Celisse. A survey of cross-validation procedures for model selection. *Statistics Surveys*, 4:40–79, 2010.

[2] T. Brants and A. Franz. Web 1t 5-gram version 1. 2006.

[3] D. Csendes, J. Csirik, T. Gyimóthy, and A. Kocsor. The Szeged Treebank. In *Lecture Notes in Computer Science: Text, Speech and Dialogue*, pages 123–131, 2005.

[4] Yann Le Cun, John S. Denker, and Sara A. Solla. Optimal brain damage. In *Advances in Neural Information Processing Systems*, pages 598–605. Morgan Kaufmann, 1990.

[5] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. Liblinear: A library for large linear classification. *J. Mach. Learn. Res.*, 9:1871–1874, June 2008.

[6] Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *J. Mach. Learn. Res.*, 3:1157–1182, March 2003.

[7] Zellig S. Harris. *Structural linguistics / by Zellig S. Harris*. University of Chicago Press, Chicago :, 1960.

[8] Trevor Hastie, Robert Tibshirani, and J. H. Friedman. *The elements of statistical learning: data mining, inference, and prediction: with 200 full-color illustrations*. New York: Springer-Verlag, 2001.

[9] Ron Kohavi and George H. John. Wrappers for feature subset selection. *Artif. Intell.*, 97(1-2):273–324, December 1997.

[10] Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. Building a large annotated corpus of english: the penn treebank. *Comput. Linguist.*, 19(2):313–330, June 1993.

[11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau,

M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine Learning in Python . *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[12] R. Lawrence Rabiner. A tutorial on Hidden Markov Models and selected applications in speech recognition. In *Proceedings of the IEEE*, volume 77, pages 257–286, 1989.

[13] P. Rebrus, A. Kornai, and D. Varga. Egy általános célú morfológiai annotáció [a general-purpose annotation of morphology]. *Általános Nyelvészeti Tanulmányok*, to appear.

[14] G. Recski. NP-chunking in Hungarian, 2010. M.A. thesis, Eötvös Loránd University.

[15] G. Recski, D. Varga, A. Zséder, and A. Kornai. Fonevi csoportok azonosítása magyar-angol párhuzamos korpuszban [Identifying noun phrases in a parallel corpus of English and Hungarian]. *VI. Magyar Számitógépes Nyelvészeti Konferencia [6th Hungarian Conference on Computational Linguistics]*, 2009.

[16] Gábor Recski and András Rung. Identifying epenthetic nouns using maximum entropy classification. *To be published*, 2013.

[17] András Rung. *Magyar főnévi alaktani jelenségek analógiás megközelítésben.* Phd dissertation, Eötvös Loránd University, 2012.

[18] Erik F. Tjong Kim Sang and Jorn Veenstra. Representing text chunks. In Henry S. Thompson and Alex Lascarides, editors, *Proceedings of the Ninth conference of the European chapter of the Association for Computational Linguistics (EACL '99)*, pages 173–179. Association for Computational Linguistics, 1999.

[19] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.

[20] V.N. Vapnik. An overview of statistical learning theory. *IEEE Transactions on Neural Networks*, 10(5):988 –999, sep 1999.

[21] Jian Zhang and Yiming Yang. Robustness of regularized linear classification methods in text categorization. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, SIGIR '03, pages 190–197, New York, NY, USA, 2003. ACM.