Feature Weighting for Co-occurrence-based Classification of Words

Viktor PEKAR
CLG, U. of Wolverhampton
Wolverhampton
UK, WV1 1SB
v.pekar@wlv.ac.uk

Michael KRKOSKA
Mentasys GmbH
Schonfeldstrasse 8
Karlsruhe, Germany, 76131
michael@mentasys.de

Steffen STAAB
Ontoprise GmbH & Institute
AIFB, U. of Karlsruhe
Karlsruhe, Germany, 76128
staab@aifb.uni-karlsruhe.de

Abstract*

The paper comparatively studies methods of feature weighting in application to the task of cooccurrence-based classification of words according to their meaning. We explore parameter optimization of several weighting methods frequently used for similar problems such as text classification. We find that successful application of all the methods crucially depends on a number of parameters; only a carefully chosen weighting procedure allows to obtain consistent improvement on a classifier learned from non-weighted data.

1 Introduction

Lexical repositories like thesauri and lexicons are today a key component of many NLP technologies, where they serve as background knowledge for processing the semantics of text. But, as is well known, manual compilation of such resources is a very costly procedure, and their automated construction is an important research issue.

One promising possibility to speed up the lexical acquisition process is to glean the semantics of words from a corpus by adopting the *co-occurrence model* of word meaning. Previous research has investigated a wide range of its applications, including automatic construction of thesauri, their enrichment, acquisition of bilingual lexicons, learning of information extraction patterns, named entity classification and others.

The basic idea behind the approach is that the distribution of a word across lexical contexts (other words and phrases it co-occurs with) is highly indicative of its meaning. The method represents the meaning of a word as a vector where each feature corresponds to a context and its value to the frequency of the word's occurring in that context. Once such representation is built, machine learning techniques can be used to perform various lexical acquisition tasks, e.g. automatically classify or cluster words according to their meaning.

However, using natural language words as features inevitably results in very noisy representations. Because of their inherent polysemy and synonymy, many context words become ambiguous or redundant features. It is therefore desirable to determine a measure of usefulness of each feature and weight it accordingly. Still, despite a wide variety of feature weighting methods existing in machine learning, these methods are poorly explored in application to lexical acquisition. There have been a few studies (e.g., Lin, 1998; Ciaramita, 2002; Alfonseca and Manandhar, 2002) where word representations are modified through this or that kind of feature weighting. But in these studies it is performed only as a standard pre-processing step on the analogy with similar tasks like text categorization, and the choice of a particular weighting procedure is seldom motivated. To our knowledge, there is no work yet on evaluation and comparison of different weighting methods for lexical acquisition.

The goal of this paper is to comparatively study a number of popular feature weighting methods in application to the task of word classification.

The structure of the paper is the following. Section 2 more formally describes the task of feature weighting. Section 3 describes the weighting methods under study. Section 4 details the experimental data, classification algorithms used, and evaluation methods. Section 5 is concerned with the results of the experiments and their discussion. Section 6 presents conclusions from the study.

2 Two feature weighting strategies

In machine learning, feature weighting before classification is performed with the purpose to reflect how much particular features reveal about class membership of instances. The weights of features are determined from their distribution across training classes, which is why the weighting procedure can be called supervised. In the context of word classification this procedure can be formalized as follows.

Let us assume that each word $n \in N$ of the training set is represented as a feature vector, consisting of features $f \in F$, and that each n is assigned a class label $c \in C$, i.e. $\forall n \exists c \in C: n \hat{I} c$. For

* The study was partially supported by the Russian Foundation Basic Research grant #03-06-80008.

each f, from its distribution across C, a certain function computes its relevance score, specific to each class. This score can be used directly as its local weight $w(f,c)$. Alternatively, from class-specific weights of a feature, one can compute its single global weight, using some globalization policy. For example, as a global weight one can use the maximum local weight of f across all classes $w_{glob}(f)=\max_{c\in C} w(f,c)$. After the weights have been applied to the training data, a classifier is learned and evaluated on the test data.

A key decision in the weighting procedure is to choose a function computing $w(f,c)$. Such functions typically try to capture the intuition that the best features for a class are the ones that best *discriminate* the sets of its positive and negative examples. They determine $w(f,c)$ from the distribution of f between c and $\bar{c}$, attributing greater weights to those f that correlate with c or $\bar{c}$ most. In the present study we include three such functions widely used in text categorization: mutual information, information gain ratio and odds ratio.

There is another view on feature scoring that it is sometimes adopted in classification tasks. According to this view, useful are those features that are shared by the largest number of positive examples of c. The purpose of emphasizing these features is to *characterize* the class without necessarily discriminating it from other classes. Functions embodying this view assess $w(f,c)$ from the distribution of f across $n \hat{I} c$, giving greater weight to those f that are distributed most uniformly. Although they do not explicitly aim at underpinning differences between classes, these functions were shown to enhance text retrieval (Wilbur and Sirotkin, 1992) and text categorization (Yang and Pedersen, 1997). In this paper we experimented with term strength, a feature scoring function previously shown to be quite competitive in information retrieval. Since term strength is an unsupervised function, we develop two supervised variants of it tailoring them for the classification task.

3 Feature Weighting Functions

3.1 Mutual Information

Mutual information (MI) is an information-theoretic measure of association between two words, widely used in statistical NLP. Pointwise MI between class c and feature f measures how much information presence of f contains about c:

$$MI(f,c)=\log\frac{P(f,c)}{P(f)P(c)} \quad (1)$$

3.2 Gain Ratio

Gain Ratio (GR) is a normalized variant of Information Gain (IG), introduced into machine learning from information theory (Quinlan, 1993). IG measures the number of bits of information obtained about presence and absence of a class by knowing the presence or absence of the feature[1]:

$$IG(f,c)=\sum_{d\in\{c,\bar{c}\}}\sum_{g\in\{f,\hat{f}\}}P(g,d)\log\frac{P(g,d)}{P(g)P(d)} \quad (2)$$

Gain Ratio aims to overcome one disadvantage of IG which is the fact that IG grows not only with the increase of dependence between f and c, but also with the increase of the entropy of f. That is why features with low entropy receive smaller IG weights although they may be strongly correlated with a class. GR removes this factor by normalizing IG by the entropy of the class:

$$GR\ (f,c)=\frac{IG(g,c)}{-\sum_{g\in\{f,\hat{f}\}}P(g)\log P(g)} \quad (3)$$

3.3 Odds Ratio

Odds Ratio (OR) is used in information retrieval to rank documents according to their relevance on the basis of association of their features with a set of positive documents. Mladenic (1998) reports OR to be a particularly successful method of selecting features for text categorization. The OR of a feature f, given the set of positive examples and negative examples for class c, is defined as[2]:

$$OR(f,c)=\frac{p(f\mid c)\cdot(1-p(f\mid\bar{c}))}{(1-p(f\mid c))\cdot p(f\mid\bar{c})} \quad (4)$$

3.4 Term Strength

Term Strength (TS) was introduced by Wilbur and Sirotkin (1992) for improving efficiency of document retrieval by feature selection. It was later studied in a number of works by Yang and her colleagues (e.g., Yang and Pedersen, 1997), who found that it performs on par with best discriminative functions on the document categorization task. This method is based on the idea that most valuable features are shared by related documents. It defines the weight of a

[1] Strictly speaking, the definition does not define IG, but conditional entropy $H(c|f)$; the other ingredient of the IG function, the entropy of c, being constant and thus omitted from actual weight calculation.

[2] In cases when $p(f|c)$ equals 1 or $p(f|\bar{c})$ equals 0, we mapped the weight to the maximum OR weight in the class.

feature as the probability of finding it in some document *d* given that it has also appeared in the document *d'*, similar to *d*. To calculate TS for feature *f*, for each *n* we first retrieved several related words *n'* using a distributional similarity measure, thus preparing a set of pairs (*n, n'*). The TS weight for *f* was then calculated as the conditional probability of *f* appearing in *n* given that *f* appears also in *n'* (the ordering of words inside a pair is ignored):

$$TS(f) = P(f \in n \mid f \in n') \quad (5)$$

An important parameter in TS is the threshold on the similarity measure used to judge two words to be sufficiently related. Yang and Pedersen determined this threshold by first deciding how many documents can be related to a given one and then finding the average minimum similarity measure for this number of neighbors over all documents in the collection. It should be noted that TS does not make use of the information about feature-class associations and therefore is unsupervised and can be used only for global feature weighting.

We introduce two supervised variants of TS, which can be applied locally: TSL_1 and TSL_2. The first one is different from TS in that, firstly, related words for *n* are looked for not in the entire training set, but within the class of *n*; secondly, the weight for a feature is estimated from the distribution of the feature across pairs of members of only that class:

$$TSL_1(f,c) = P(f \in n \mid f \in n'), \text{with } n, n' \in c \quad (6)$$

Thus, by weighting features using TSL_1 we aim to increase similarity between members of a class and disregard possible similarities across classes.

Both TS and TSL_1 require computation of similarities between a large set of words and thus incur significant computational costs. We therefore tried another, much more efficient method to identify features characteristic of a class, called TSL_2. As TSL_1, it looks at how many members of a class share a feature. But instead of computing a set of nearest neighbors for each member, it simply uses all the words in the class as the set of related words. TSL_2 is the proportion of instances which possess feature *f* to the total number of instances in *c*:

$$TSL_2(f,c) = \frac{|\{n \in c \mid f \in n\}|}{|\{n \in c\}|} \quad (7)$$

Table 1 illustrates the 10 highest scored features according to five supervised functions for the class {*ambulance, car, bike, coupe, jeep, motorbike, taxi, truck*} (estimated from the BNC co-occurrence data described in Section 4).

MI	GR	OR	TSL_1	TSL_2
see_into	knock_by	die_after	see	see
die_after	climb_of	drive_into	drive	drive
drive_into	die_after	remand_to	take	get
remand_to	drive_into	privatise	get	take
run_from	remand_to	make_about	get_into	get_into
privatise	privatise	force_of	hear	park
release_into	make_about	plan_with	need	hear
switch_to	force_of	recover_in	call	wait_for
make_about	plan_with	start_up	send	need
entrust_to	recover_in	explode_in	go_by	call

Table 1. 10 highest scored features for class {*ambulance, car, bike, coupe, jeep, motorbike, taxi, truck*} according to MI, GR, OR, TSL_1, TSL_2

The examples vividly demonstrate the basic differences between the functions emphasizing discriminative features vs. those emphasizing characteristic features. The former attribute greatest weights to very rare context words, some of which seem rather informative (*knock_by, climb_of, see_into*), some also appear to be occasional collocates (*remand_to, recover_in*) or parsing mistakes (*entrust_to, force_of*). In contrast, the latter encourage frequent context words. Among them are those that are intuitively useful (*drive, park, get_into*), but also those that are too abstract (*see, get, take*). The inspection of the weights suggests that both feature scoring strategies are able to identify different potentially useful features, but at the same time often attribute great relevance to quite non-informative features. We next describe an empirical evaluation of these functions.

4 Experimental Settings

4.1 Data

The evaluation was carried out on the task of classifying English nouns into predefined semantic classes. The meaning of each noun $n \hat{I} N$ was represented by a vector where features are verbs $v \hat{I} V$ with which the nouns are used as either direct or prepositional objects. The values of the features were conditional probabilities $p(v|n)$. Two different datasets were used in the experiments: verb-noun co-occurrence pairs extracted from the British National Corpus (BNC)[3] and from the Associated Press 1988 corpus (AP)[4]. Rare nouns were filtered: the BNC data contained nouns that appeared with at least 5 different verbs and the AP data contained 1000 most frequent nouns, each of which appeared

[3] http://www.wlv.ac.uk/~in8113/data/bnc.tar.gz

[4] http://www.cs.cornell.edu/home/llee/data/sim.html

with at least 19 different verbs. Co-occurrences that appeared only once were removed.

To provide the extracted nouns with class labels needed for training and evaluation, the nouns were arranged into classes using WordNet in the following manner. Each class was made up of those nouns whose most frequent senses are hyponyms to a node seven edges below the root level of WordNet. Only those classes were used in the study that had 5 or more members. Thus, from the BNC data we formed 60 classes with 514 nouns and from the AP data 42 classes with 375 nouns.

4.2 Classification algorithms

Two classification algorithms were used in the study: *k* nearest neighbors (*k*NN) and Naïve Bayes, which were previously shown to be quite robust on highly dimensional representations on tasks including word classification (e.g., Ciaramita 2002).

The *k*NN algorithm classifies a test instance by first identifying its *k* nearest neighbors among the training instances according to some similarity measure and then assigning it to the class that has the majority in the set of nearest neighbors. We used the weighted *k*NN algorithm: the vote of each neighbor was weighted by the score of its similarity to the test instance.

As is well known, *k*NN's performance is highly sensitive to the choice of the similarity metric. Therefore, we experimented with several similarity metrics and found that on both datasets Jensen-Shannon Divergence yields the best classification results (see Table 1). Incidentally, this is in accordance with a study by (Dagan et al., 1997) who found that it consistently performed better than a number of other popular functions.

Similarity function	BNC	AP
Jensen-Shannon	**41.67**	**41.33**
L1 distance	38.15	39.72
Jaccard	36.82	37.01
Cosine	36.80	34.95
Skew Divergence	35.82	37.34
L2 distance	24.15	26.62

Table 2. Comparison of similarity functions for the *k*NN algorithm.

Jensen-Shannon Divergence measures the (dis)similarity between a train instance *n* and test instance *m* as:

$$J(n,m) = \frac{1}{2}[D(n \| avg_{n,m}) + D(m \| avg_{n,m})] \quad (8)$$

where *D* is the Kullback Leibler divergence between two probability distributions *x* and *y*:

$$D(x \| y) = \sum_{v \in V} p(v \mid x) \log \frac{p(v \mid x)}{p(v \mid y)} \quad (9)$$

and $avg_{n,m}$ is the average of the distributions of *n* and *m*.

In testing each weighting method, we experimented with k = 1, 3, 5, 7, 10, 15, 20, 30, 50, 70, and 100 in order to take into account the fact that feature weighting typically changes the optimal value of k. The results for *k*NN reported below indicate the highest effectiveness measures obtained among all k in a particular test.

The Naïve Bayes algorithm classifies a test instance *m* by finding a class *c* that maximizes $p(c|V_m \in m)$. Assuming independence between features, the goal of the algorithm can be stated as:

$$\text{argmax}_i \; p(c_i \mid V_m) \approx \text{argmax}_i \; p(c_i) \prod_{v \in V_m} p(v \mid c_i) \quad (10)$$

where $p(c_i)$ and $p(v|c_i)$ are estimated during the training process from the corpus data.

The Naïve Bayes classifier adopted in the study was the binary independence model, which estimates $p(v|c_i)$ assuming the binomial distribution of features across classes. In order to introduce the information inherent in the frequencies of features into the model all input probabilities were calculated from the real values of features, as suggested in (Lewis, 1998).

4.3 Evaluation method

To evaluate the quality of classifications, we adopted the ten-fold cross-validation technique. The same 10 test-train splits were used in all experiments. Since we found that the difficulty of particular test sets can vary quite a lot, using the same test-train splits allowed for estimation of the statistical significance of differences between the results of particular methods (one-tailed paired t-test was used for this purpose). Effectiveness was first measured in terms of precision and recall, which were then used to compute the F_β score[5]. The reported evaluation measure is microaveraged F scores.

As a baseline, we used the *k*-nn and the Naïve Bayes classifiers trained and tested on non-weighted instances.

5 Results

5.1 Term Strength

We first describe experiments on finding the most optimal parameter settings for Term Strength.

As was mentioned in Section 3.4, an important parameter of term strength that needs to be tuned for

[5] *b* was set to 1.

a task is the similarity threshold which is used to judge a pair of words to be semantically related. Since in both datasets the minimum number of words in a class was 5, we chose 4 to be the number of words that can be related to any given word. Finding the four nearest neighbors for each word in the collection, we calculated the average minimum similarity score that a pair of words must have in order to be considered related. However, since words vary a lot in terms of the amount of corpus data available on them, the average similarity threshold might be inappropriate for many words. Therefore we tried also another way to select pairs of related words by simply taking the four most similar words for each particular word. Table 4 compares the two methods of locating related words (significant improvements at a=0.05 are shown in bold).

	kNN		Naïve Bayes	
	TS	TSL_1	TS	TSL_1
BNC				
Threshold	39.54	36.84	41.67	37.97
Top 4 words	40.90	**39.74**	41.86	**42.64**
AP				
Threshold	42.12	40.22	37.80	33.82
Top 4 words	42.12	**44.45**	38.07	**36.47**

Table 4. Two methods of identifying semantically related words for TS and TSL_1.

We see that using a similarity threshold indeed produces worse results, significantly so for TSL_1. In the rest of the experiments we used a fixed number of related words in calculating TS and TSL_1.

5.2 Globalization methods

Before comparing global and local variants of the functions, we studied three ways to derive a global weight for a feature: (1) using the maximum local relevance score of a feature across all classes, (2) its weighted average score (the contribution of each class-specific score is weighted by the size of the class), and (3) the sum of all local scores. The results are shown on Tables 5 and 6 (in bold are the figures that are significantly different from the second-best achievement at a=0.05).

	MI	GR	OR	TSL_1	TSL_2
BNC					
max	42.83	**48.88**	**46.35**	**37.9**	**41.52**
wavg	41.29	45.95	42.65	26.47	27.24
sum	41.29	45.95	42.65	26.46	27.24
AP					
max	43.17	43.44	44.77	32.48	35.95
wavg	42.93	43.98	41.	37.31	38.12
sum	43.20	43.99	41.61	37.04	37.85

Table 5. Globalization methods on *k*NN.

	MI	GR	OR	TSL_1	TSL_2
BNC					
max	**46.52**	42.82	43.96	**37.17**	**38.53**
wavg	43.4	41.45	43.98	21.	24.5
sum	40.48	43.59	45.15	18.66	22.96
AP					
max	39.68	38.6	42.07	**38.1**	**38.64**
wavg	39.15	42.1	40.23	33.82	35.15
sum	39.68	42.38	40.76	34.1	35.96

Table 6. Globalization methods on Naïve Bayes.

As one can see, using a maximum local weight is usually the best method of globalization. Its performance is often significantly higher than that of the other methods. The explanation for this can be the fact that a feature often has very high scores relative to specific classes, while in the rest of the classes its weight is low. Using its weighted average score or a sum of local scores results in obscuring its high relevance to some classes. In contrast, the maximum local score does reflect high relevance of the feature to these classes. If, in addition to that, the feature appears in very few classes, it is unlikely that its being weighted too highly interferes with the representations of irrelevant classes. This is confirmed by the fact that the maximum weight is noticeably better on the BNC dataset, which contains much more rare features than the AP one.

5.3 Global vs. Local Weighting

In carrying out either local or global weighting, there is a choice either to weight only training instances or also test instances before their classification. The test instance can be weighted either by the global weights or by the local weights of the class it is compared with. Tables 7 and 8 present the results of the evaluation of the functions along two dimensions: (1) local versus global weighting and (2) weighted versus un-weighted test instances. As before, the results for those methods whose superiority over other ones is statistically significant appear in bold.

		MI	GR	OR	TS	TSL_1	TSL_2
BNC							
gl	y	42.83	**48.88**	**46.35**	34.72	32.48	35.95
loc	y	28.43	35.84	34.29	-	15.38	20.75
gl	n	40.12	39.74	38.36	**40.90**	38.35	36.99
loc	n	40.32	40.33	39.72	-	39.74	**41.29**
AP							
gl	y	43.17	**43.44**	**44.77**	38.12	37.9	41.52
loc	y	37.31	31.74	37.04	-	33.06	36.68
gl	n	41.59	40.78	40.51	**42.12**	39.25	40.24
loc	n	40.74	37.86	41.34	-	**44.45**	**43.70**

Table 7. Local vs. global weighting schemas on *k*NN.

		MI	GR	OR	TS	TSL_1	TSL_2
BNC							
gl	y	46.52	42.82	43.96	33.87	37.17	38.53
loc	y	41.84	43.79	38.32	-	36.01	39.53
gl	n	45.54	42.63	40.87	**41.86**	41.65	45.54
loc	n	43.99	38.93	44.38	-	42.64	46.53
AP							
gl	y	39.68	**38.60**	**42.07**	36.22	38.10	38.64
loc	y	36.50	31.72	37.04	-	33.56	35.66
gl	n	39.16	35.44	38.65	38.07	39.43	39.95
loc	n	38.89	27.96	38.15	-	36.47	39.42

Table 8. Local vs. global weighting schemas on Naïve Bayes.

The results are largely consistent both across the datasets and across the classification methods. Discriminative functions are almost always best in their global variants; when applying them globally, it is also advisable to weight test instances. In contrast, the characteristic functions TSL_1 and TSL_2 are usually better when applied locally. It is also noteworthy that all the variants of TS fare better when test instances are not weighted.

We believe that the good performance of the global versions of MI, GR, and OR should be explained by the fact that features they weight highest are rare and likely to appear only in one class so that using the same weight for all classes does not cause confusion between them. It is also beneficial to weight test instances globally, because this guarantees that most features of a test instance always have a non-zero weight. With characteristic functions, however, highest weighted are rather frequent features which are often present in other classes as well. Using the same weight of these features for all classes therefore fails to differentiate classes from each other. Local TSL_1 and TSL_2 are more advantageous. Although individual features they weight highest may be mediocre separators, usually several such features are given prominence within a class. Taken collectively they appear to be able to successfully discriminate a class from other classes.

An interesting observation is that the combination of a local schema with weighted test instances is very undesirable with all the functions. The reason for this is that very often a test instance has many features different from those in the training class to which it is being compared. Because of this, these features receive zero local weights, which renders the representation of the test instance extremely sparse.

Table 9 shows how the performance of the most optimal settings for the six studied function compares with the baseline (improvements on the baseline are in bold).

	kNN		Naïve Bayes	
	BNC	AP	BNC	AP
MI	**42.83**	**43.17**	**46.52**	**39.68**
GR	**48.88**	**43.44**	43.79	38.60
OR	**46.35**	**44.77**	43.96	**42.07**
TS	40.90	**42.12**	41.86	38.07
TSL_1	39.74	**44.45**	42.64	**39.43**
TSL_2	41.29	**43.70**	**46.53**	**39.95**
baseline	41.67	41.33	45.55	39.16

Table 9. The most optimal settings for MI, GR, OR, TS, TSL_1 and TSL_2 compared to the baselines.

All the functions often show superiority over the baseline, except for TS which only once slightly outperformed it. However, statistical significance of the improvement was registered only for MI and OR on the BNC data, using the kNN classifier, which was 17% and 11% better than the baseline correspondingly.

Comparing discriminative and characteristic weighting functions we see that the supervised variants of TS frequently perform on a par with MI, GR, and OR. Particularly, TSL_2 was the best performer on Naive Bayes, BNC and the second best on kNN, AP. We also see that the supervised variants of TS very often surpass its original unsupervised variant, but the improvement is significant only for TSL_2, on the BNC dataset using Naive Bayes (at a=0.001).

5.4 Correlations between the functions

As was mentioned before, an informal inspection of features emphasized by different functions suggests that the discriminative functions tend to give greater weights to rare features, while characteristic ones to frequent features. In order to see if this results in disagreement between them as to the classifications of test instances, we measured the extent to which classifications resulting from MI, GR, OR, TSL_1, and TSL_2 overlap. For this purpose, we calculated the Kappa coefficient for all the 10 possible pairs of these functions. The results are reported in Table 10.

	GR	OR	TSL_1	TSL_2
MI	0.676 0.762	0.711 0.729	0.584 0.668	0.801 0.788
GR		0.873 0.855	0.483 0.617	0.571 0.703
OR			0.473 0.614	0.588 0.695
TSL_1				0.658 0.721

Table 10. The agreement in classifications using Naïve Bayes between MI, GR, OR, TSL_1, and TSL_2 on the BNC and AP datasets.

On results from both datasets, we see that the highest agreement is indeed between MI, OR, and GR and between TSL_1 and TSL_2. Interestingly, there is also a relatively strong correlation between classification resulting from using MI and TSL_2. The lowest agreement is between the discriminative functions and TSL_1.

	kNN		Naïve Bayes	
	BNC	AP	BNC	AP
MI	42.83	43.17	46.52	39.68
GR	48.88	43.44	43.79	38.60
OR	46.35	44.77	43.96	42.07
TSL_1	39.74	44.45	42.64	39.43
TSL_2	41.29	43.70	46.53	39.95
$MI*TSL_1$	40.51	**45.27**	44.2	38.1
$GR*TSL_1$	42.47	43.2	43.6	34.37
$OR*TSL_1$	41.49	44.72	**46.32**	37.3
$MI*TSL_2$	**44.81**	**45.04**	**46.73**	**42.36**
$GR*TSL_2$	47.53	**44.77**	44.17	37.55
$OR*TSL_2$	46.35	**45.29**	46.5	40.47

Table 11. Combinations of TSL_1 and TSL_2 with MI, GR, and OR.

5.5 Combination of the functions

An obvious question is whether the effectiveness of classifications can be increased by combining the discriminative and the characteristic weights of a feature given that both provide useful, but different kinds of evidence about the correct class label of test instances. To investigate this, we tried combining each of the discriminative weights of a feature with each of its supervised characteristic weights in the following manner. First, both kinds of weights were estimated from non-weighted training data. Then they were applied one after the other to the training data. During the test procedure, test instances were weighted only with the global weights. The results of these experiments are shown in Table 11. Results for those combined weighting methods which outperformed both of the component functions are shown in bold.

Certain combined weighting procedures did improve on both of the component methods. However, none of them showed an improvement over 2.5% on the best of the component weighting methods (no significance for any of the improvements could be established).

6 Conclusion

In the paper we studied several feature weighting methods in application to automatic word classification. Our particular focus was on the differences between those weighting methods which encourage features discriminating classes from each other (odds ratio, gain ratio, mutual information) and those which favor features that best characterize classes (term strength).

We find that classification of words into flatly organized classes is a very challenging task with quite low upper and lower bounds, which suggests that a considerable improvement on the baseline is hard to achieve. We explicitly explored parameterization of the weighting functions, finding that the choice of certain parameters, notably the application of local vs. global weights and weighted vs. un-weighted test instances, is critical for the performance of the classifier. We find that the most optimal weighting procedure often brings the performance of a classifier significantly closer to the upper bound, achieving up to 17% improvement on the baseline.

We find that discriminative and characteristic weighting procedures are able to identify different kinds of features useful for learning a classifier, both consistently enhancing the classification accuracy. These findings indicate that although individual characteristic features may be less powerful class separators, several such features, taken collectively, are helpful in differentiating between classes.

References

E. Alfonseca and S. Manandhar. 2002. *Extending a lexical ontology by a combination of distributional semantics signatures*. In Proceedings of EKAW'02, pp.1-7.

M. Ciaramita. 2002. *Boosting automatic lexical acquisition with morphological information*. In Proceedings of the ACL-02 Workshop on Unsupervised Lexical Acquisition. pp.17-25.

I. Dagan, L. Lee, and F. C. N. Pereira. 1997. *Similarity-based methods for word sense disambiguation.* In Proceedings of ACL'97, pp. 56-63.

D. Lewis. 1998. *Naive (Bayes) at forty: The independence assumption in information retrieval.* In Proceedings of ECML'98, pp.4-15.

D. Lin (1998) *Automatic retrieval and clustering of similar words*. In Proceedings of COLING-ACL'98, pp. 768-773.

D. Mladenic. 1998. *Feature subset selection in text learning.* In Proceedings of ECML'98, pp.95-100.

J.R. Quinlan. 1993. *C4.5: Programs for Machine Learnin*g. San Mateo, CA: Morgan Kaufmann.

J.W. Wilbur and K. Sirotkin. 1992. The automatic identification of stopwords. Journal of Information Science, (18):45-55.

Y. Yang and J.O. Pedersen. 1997. A comparative study on feature selection in text categorization. Proceedings of ICML'97, pp. 412-420.