Incorporating External Information in Bayesian Classifiers Via Linear Feature Transformations

Tapio Pahikkala, Jorma Boberg, Aleksandr Mylläri, and Tapio Salakoski

Turku Centre for Computer Science (TUCS)
Department of Information Technology, University of Turku
Lemminkäisenkatu 14 A, FIN-20520 Turku, Finland
firstname.lastname@it.utu.fi

Abstract. Naive Bayes classifier is a frequently used method in various natural language processing tasks. Inspired by a modified version of the method called the flexible Bayes classifier, we explore the use of linear feature transformations together with the Bayesian classifiers, because it provides us an elegant way to endow the classifier with an external information that is relevant to the task. While the flexible Bayes classifier is based on the idea of using kernel density estimation to obtain the class conditional probabilities of continuously valued attributes, we use the linear transformations to smooth the feature frequency counts of discrete valued attributes. We evaluate the method on the context sensitive spelling error correction problem using the Reuters corpus. For this particular task, we define a positional feature transformation and a word feature transformation that take advantage of the positional information of the context words and the part-of-speech information of words, respectively. Our experimental results show that the performance of the Bayesian classifiers in the natural language disambiguation tasks can be improved with the proposed transformations and that the incorporation of external information via the linear feature transformations is a promising research direction.

1 Introduction

Many natural language processing applications require accurate resolution of the various kinds of ambiguity present in natural language, giving rise to a class of disambiguation problems. In this paper, we focus on lexical disambiguation problems, where disambiguation is done at the level of words. One such problem is the problem of context-sensitive spelling error correction, where the misspelling of the original word belongs to the language, such as, for example, *affect* misspelled as *effect* or vice versa. This mistake cannot be detected by standard lexicon-based checkers, since both words belongs to the English lexicon. A set of similar words that belong to the lexicon and that are often confused with the other words in the set is called a *confusion set*. For example, {*maybe*, *may be*} can be considered as a binary confusion set.

In our previous work [1,2,3], we have shown that the performance of the natural language disambiguation systems can be improved by taking advantage of

T. Salakoski et al. (Eds.): FinTAL 2006, LNAI 4139, pp. 399–410, 2006.

the positional information of the context words with position sensitive kernel functions. The methods considered in these studies were the support vector machine classifiers that have recently become the state-of-the-art machine learning algorithms. Naive Bayes classifier is another frequently used method in various natural language processing tasks that has been shown to have a high performance. John and Langley [4] introduced the flexible Bayes, a version of the naive Bayes classifier that uses kernel density estimation to estimate the class conditional probabilities of continuous attributes. We [3] used this method in context of word sense disambiguation to estimate the probabilities of word-position features. The experiment demonstrated that the performance of the Bayesian classifiers can be improved even if the attributes are ordinal (the word positions in the context of the ambiguous word in our case).

In this paper, we introduce a realization of the flexible Bayes classifier that is based on linear feature transformations, that is, we also relax the ordinality assumption of the attributes. To our knowledge, this has not been considered in the literature. Moreover, inspired by the good results obtained in the previous studies by transforming the positional information of the context words, we also define transformations for the words. We describe a method to construct the word transformations using an external source of information that is useful for the classification task in question. One such information source for the natural language disambiguation tasks is, for example, the part-of-speech information of the words (see e.g. Jurafsky and Martin [5]).

This paper is organized as follows. We start by describing the data representation and the Bayesian classifiers in Section 2. In Section 3, we consider the use of the linear feature transformations together with the Bayesian classifiers. The proposed transformations are evaluated in Section 4. We conclude the paper in Section 5 and discuss possible future directions.

2 Binary Classification with Bayesian Classifiers

We first describe the representation of the data we use in our study. Next, we give a definition of the naive Bayes classifier. Finally, we define the flexible Bayes classifier originally introduced by John and Langley [4].

2.1 Representation of the Data

In this paper, we consider the context sensitive spelling error correction as a model problem. In this kind of tasks, each data point consists of a word to be disambiguated and the context words surrounding it. We formalize the data points in the following way. Let s be a context span parameter that determines how many words to the left and to the right from the ambiguous word are included in the context, so that the size of the context window $r = 2s + 1$. If there are not enough words available in the text to the left or to the right from the ambiguous word, we use "empty" words to get a word sequence of length r. Let n be the number of words and $\mathcal{W} = \{w_1, \ldots, w_n\}$ be the set of

words. Let x be a data point. We define a representation of the data point x to be a word-position matrix $A_x \in \mathcal{M}_{n\times r}(\mathbb{R})$, where $\mathcal{M}_{n\times r}(\mathbb{R})$ is the set of $n \times r$-matrices whose elements belong to $\mathbb{R}$. The word positions of a context are defined as $-s, \ldots, 0, \ldots, s$, where the word to be disambiguated is in the position zero of its context. A word-position matrix A_x generated from the context of an ambiguous word is a binary matrix in which the element $A_x(i, j)$ corresponding to the word w_i and the position $p = j - s - 1$ has the value one if the word w_i occurs in the position p of the context and zero otherwise. We also observe that there can be at most one nonzero element in each column because each position can have only one word. If the word in a certain position of a context is not in $\mathcal{W}$ or if the position in the context is empty, the corresponding column has only zeros. On the other hand, the same word can occur in several positions in the context, and therefore the rows of the matrices can have several nonzero elements.

2.2 Naive Bayes Classifier

Let F_x be the set of all features that are contained in a data point x. According to our definition of the data points as word-position matrices, the word-position features in F_x correspond to the ones in the binary valued word-position matrix constructed from the data point x. For the Naive Bayes classifier, we use the following decision function:

$$d(x) = P(+1) \prod_{f\in F_x} P(f|+1) - P(-1) \prod_{f\in F_x} P(f|-1), \tag{1}$$

where $P(f|+1)$ and $P(f|-1)$ are the probabilities that the feature f appears in a positive and in a negative example, respectively, and $P(+1)$ and $P(-1)$ are the prior probabilities of the positive and negative classes. A new data point x is given a positive (resp. negative) class label, if the value of the decision function (1) for the data point is positive (resp. negative).

The probabilities can be directly estimated from the training data using maximum likelihood estimation (MLE) as follows. Let F denote the set of all possible features the data points can contain. For each class $y \in Y$ and feature $f \in F$,

$$P(y) = \frac{\mathrm{N}(y)}{\sum_{y'\in Y} \mathrm{N}(y')},$$
$$P(f|y) = \frac{\mathrm{N}(f, y)}{\sum_{f'\in F} \mathrm{N}(f', y)},$$

where $\mathrm{N}(y)$ is the number of examples in the class $y \in Y$, and $\mathrm{N}(f, y)$ is the feature frequency count of f conditional to the class y, that is, the number of times feature f appears in the examples of the class y. The MLE estimates are typically smoothed to avoid zero probabilities in prediction; in this paper we use add-δ smoothing, where all numbers of feature occurrences are incremented by $\delta > 0$ (in our experiments, we set $\delta = 0.001$) over the counted value (see e.g. Chen and Goodman [6]).

2.3 Flexible Bayes Classifier

John and Langley [4] introduced the flexible Bayes method, a version of the naive Bayes classifier that uses kernel density estimation (we refer to Silverman [7] for more information on kernel density estimation) to estimate the class conditional probabilities of continuous attributes (this method has also been described by Hastie et al. [8]). While the word-position random variable is discrete, and hence a histogram is a natural way to estimate its density, the estimate can still be bumpy because of the lack of training data.

We now define a version of a Bayesian classifier which is similar to the flexible Bayes classifier. The class conditional probabilities of the features are estimated as follows

$$P(f|y) = \frac{\sum_{f' \in F} \mathrm{N}(f', y) g(f, f')}{\sum_{f', f'' \in F} \mathrm{N}(f', y) g(f', f'')}, \tag{2}$$

where g is a kernel function. The estimate can be considered as a convolution of the sample empirical distribution of the features with the kernel function (see e.g. Hastie et al. [8]).

In our previous study [3], our features were the word-position pairs described above and we used, among the others, a kernel function $g(f, f') : F \times F \to \mathbb{R}^+$:

$$g(f, f') = g((w, p), (v, q)) = \begin{cases} exp(-\theta(p-q))^2 & \text{when} \quad w = v \\ 0 & \text{otherwise} \end{cases}, \tag{3}$$

where $\theta \geq 0$ is a parameter, and w (resp. v) is the word and p (resp. q) is its position. We refer to this approach as the smoothed position-sensitive model. Note that if we let $\theta \to \infty$, the flexible Bayes becomes the naive Bayes classifier with the word-position features. We refer to this case as the basic position-sensitive model. If we, on the other hand, set $\theta = 0$, the different positions are identified with each other and the obtained classifier is a naive Bayes classifier constructed from the bag-of-words features.

Below, we describe how the kernel (3) can be used together with the Bayesian classifiers via linear transformations of the estimated feature frequency counts. The formalization of the approach with the linear transformations provides us a better machinery to incorporate external information into the classifier than the Gaussian kernel we use above.

3 Linear Feature Transformations

We will now describe a realization of the Bayesian classifier (2) that uses linear transformations on the word-position matrices that contain the class conditional feature frequency counts estimated from the data. Recall that we have defined our data points to be word-position matrices whose columns are the word feature vectors for different positions. Let us define for the class y a word-position matrix $A_y \in \mathcal{M}_{n \times r}(\mathbb{R})$ whose elements contain the number of occurrences of each word-position feature in class y, that is, $A_y = \sum_{x \in X_y} A_x$, where A_x is a word-position

matrix corresponding to a training data point x and X_y is the set of training examples that belong to the class y. We will refer to the matrices A_y as class conditional feature frequency count matrices.

3.1 Constructing Linear Transformations for Position Vectors

We formulate the kernel function (3) as a linear transformation on the row vectors of the class conditional feature frequency count matrices. Let $P \in \mathcal{M}_{r\times r}(\mathbb{R})$ be a matrix of a transformation whose values are obtained as

$$P(i,j) = g((w,p),(w,q)), \tag{4}$$

where g is the function defined in (3), $p = i - s - 1$, and $q = j - s - 1$. The word w can be any word, because the value of the function (3) does not depend on it. We will call the matrix P a positional transformation.

We obtain the transformed feature frequency counts via the following matrix product of the original frequency counts and the positional transformation

$$\widehat{A_y} = A_y P.$$

When we normalize $\widehat{A_y}$ with the sum of the values of all its elements, we have the estimates of the class conditional probabilities of the features that can be used to construct a Bayesian classifier.

3.2 Constructing Linear Transformations for Word Vectors

Above, we defined a realization of the flexible Bayes classifier with a linear transformation of the position vectors. We can also construct linear transformations for the word vectors, that is, the column vectors of the class conditional feature frequency count matrices. Let $W \in \mathcal{M}_{n\times n}(\mathbb{R})$ be a matrix of the transformation that we call here a word transformation. The transformation is performed on the frequency count matrix A_y as

$$\widehat{A_y} = W A_y.$$

In the following, we consider a possible way to construct the word transformation.

In many natural language disambiguation tasks, the parts-of-speech (PoS) of context words are known to provide useful additional information (see e.g. Jurafsky and Martin [5]). We now use this external source of information to construct a linear feature transformation. Suppose that we know for each word all possible parts-of-speech it can have. Let us define a PoS-word matrix $V \in \mathcal{M}_{t\times n}(\mathbb{R})$ so that $V(i,j)$ is one if the jth word can have the ith PoS and zero otherwise. Let us consider an example in which the set of words consists of the words *composition*, *contribution*, *write*, and *being*, and the possible PoS are *noun* and *verb* (i.e. $n = 4$ and $t = 2$). Then

$$\begin{array}{l}\mathcal{W} = \{\textit{composition}, \textit{contribution}, \textit{write}, \textit{being}\} \\ \mathcal{P} = \{\textit{noun}, \textit{verb}\}\end{array} \text{ and } V = \begin{pmatrix} 1\,1\,0\,1 \\ 0\,0\,1\,1 \end{pmatrix},$$

whete $\mathcal{P}$ denotes the set of PoS. Notice that it is possible for words to have several nonzero values in their corresponding column in W. For example, the word *being* can be a noun or a verb. From this PoS-word matrix, we construct the following word-word similarity matrix

$$W = V^{\mathrm{T}}V = \begin{pmatrix} 1\,1\,0\,1 \\ 1\,1\,0\,1 \\ 0\,0\,1\,1 \\ 1\,1\,1\,2 \end{pmatrix}, \tag{5}$$

where the rows and the columns are indexed by the four example words. The words *composition* and *contribution* are identified with each other in practice because their corresponding columns in W are equal. The word *being* is similar to all other words to some extent because it shares common PoS with them. To ensure that each word position feature is transformed to an equal amount of probability mass, we normalize the column vectors of $\widehat{W}$ with their 1-norm. By 1-norm of a vector x, we mean $\sum_i |x_i|$.

In reality there are, however, only a few PoS compared to the number of words. Therefore too many words will get identified with each other which would lead to a too powerful smoothing effect. We can control the amount of smoothing by increasing the diagonal of the transformation matrix as follows

$$\widetilde{W} = W + \mu I,$$

where $I \in \mathcal{M}_{n\times n}(\mathbb{R})$ is an identity matrix and μ is a parameter. The diagonal shift has the effect that the transformed feature frequency counts are the sum of the original frequency counts multiplied by μ and the frequency counts smoothed with the PoS information of the words.

An appropriate value of the diagonal shift parameter μ can be selected, for example, with a cross-validation. For instance, if we set $\mu = 1$, the normalized and diagonal shifted transformation matrix (5) will become

$$\widetilde{W} = \begin{pmatrix} 1.33 & 0.33 & 0 & 0.2 \\ 0.33 & 1.33 & 0 & 0.2 \\ 0 & 0 & 1.5 & 0.2 \\ 0.33 & 0.33 & 0.5 & 1.4 \end{pmatrix}. \tag{6}$$

From (6) we observe that the words *composition* and *contribution* are no longer completely identical and the similarities between the other words are also decreased compared to the values of the diagonal elements. Analogously with the above described smoothed position-sensitive model which encompasses the basic position-sensitive and bag-of-words models as extreme cases, the model based on the diagonal shifted word transformation encompasses the model based only on the word features and the model that has only the PoS information of the words. We observe that if we let $\mu \to \infty$, the information of the PoS disappears and we have just the word-position features. On the other hand, if we set $\mu = 0$, the word-position features are completely replaced with the PoS-position features.

When we select the value of the parameter μ, we choose an intermediate between these two extremes.

Note that we need the PoS-word matrix V only for the construction of the similarity matrix W of the words. If we have a source of the similarity information of the words, we can also directly construct the matrix W in a similar way that the positional transformation matrix P is obtained in (4).

3.3 Combination of Transformations

Let P be the matrix of a positional transformation and W the matrix of a word transformation. If we use both of the transformations at the same time, we obtain the transformed feature frequency counts via the following matrix product

$$\widehat{A_y} = W A_y P, \tag{7}$$

where A_y is the matrix that contains the original feature frequency counts for the class y.

3.4 Implementation Issues

Let W denote the normalized word transformation described above and $\widetilde{W}$ its diagonally shifted version. In our experiments, we defined our set of words to consist of the 10000 most common words in our data set. Therefore the transformation matrix is of dimension 10000×10000 and hence the computation of the matrix product $\widetilde{W} A_y$ for each class y may be too tedious in practice. Fortunately, because of the small number of possible PoS (in our experiments we used 10 different PoS), we are able to speed up the computation. Let V denote the PoS-word matrix from which the word transformation is constructed in the way described above. Instead of using the diagonally shifted and normalized matrix (6) directly, we perform the transformation as

$$\widetilde{W} A_y = (V^{\mathrm{T}}(V \bullet Z) + \mu I) A_y \tag{8}$$

$$= V^{\mathrm{T}}((V \bullet Z) A_y) + \mu A_y, \tag{9}$$

where $V \bullet Z$ denotes an elementwise product of the matrices V and Z of equal dimensionality and Z is a normalization matrix that ensures that the 1-norms of the column vectors of W are equal to one. The parenthesis in (9) denote the order in which the calculations are performed in the implementation. The two matrix products performed consecutively in (9) are together much faster to compute than the matrix product in the left hand side of (8).

4 Experiments

We use the task of context-sensitive spelling correction as a model problem to evaluate the performance of the proposed approach. In this task, the correct

spelling of a word must be disambiguated among a set of alternative spellings based on its context, deciding, for example, between *country* and *county*. There is practical interest in improving methods at solving this task, and it is ideal as a model problem since a large dataset can be created without manual tagging. As high-quality texts such as newswire articles are widely available and unlikely to contain spelling errors, the required training and test examples can simply be extracted from such resources.

Golding and Roth [9] defined 21 sets of commonly confused words in their context-sensitive spelling correction experiments. 19 out of the 21 sets are binary (i.e. they consist only of two alternative spellings). For simplicity, we focus only on the 19 two-class problems in this paper. We create the datasets from the Reuters News corpus [10], extracting a training set of 1000 and a test set of 5000 examples for each confusion set as follows: we first search the corpus for documents containing either of the confusion set words. In each such document, every occurrence of either of the confusion set words forms a candidate example. We then form datasets of the required size by randomly selecting documents until they together contain sufficiently many examples; possible extra examples are randomly discarded from the last selected document. This sampling strategy assures that there is no overlap in documents between training and test examples. Finally, we assign one of the confusion set words the positive and the other the negative label, label each selected example, and remove from the context of each example the confusion set word.

We measure the performance of the classifier with different kernels with the area under the ROC curve (AUC) (see e.g. Fawcett [11]), and test the statistical significance of the performance difference between the various transformations and the baseline method using the Wilcoxon signed-ranks test [12].

In all the experiments, we select the value of the context span parameter s separately for each confusion set using a grid search (the grid points i.e. the possible values of s are $2^0, 2^1, \ldots, 2^6$). The grid searches are performed on the training data with a leave-one-out cross-validation (LOOCV). In some experiments, there are also other parameters to be selected, for example, the θ-parameter of the positional transformation. In those cases, we perform a two dimensional grid search over the possible values of the parameters s and θ.

First, we evaluate a naive Bayes (NB) classifier with the basic position sensitive (BP) model, that is, the NB classifier with the above described word-position features, and compare its performance to that of the NB classifier with the bag-of-words (BoW) model. The BP model clearly outperforms the BoW model on average as can be observed from Table 1. The performance gain is statistically significant ($p < 0.05$). The BoW model is better only with the data set *country-county*. In the performance comparison of our previous study [3] with the Senseval-3 data, the BoW model was better on average. Next, we test the smoothed position sensitive (SP) model, that is, the NB classifier whose class conditional probabilities are obtained from the positionally transformed feature frequency counts. The value of the θ-parameter is selected with a leave-one-out cross-validation with the training data together with the context span. Note that

Table 1. Comparison of naive Bayes classification performance with the bag-of-words (BoW) model, the basic position sensitive (BP) model, and the smoothed position sensitive (SP) model. The performances are measured with AUC. The performance difference of BP and BoW, SP and BoW, and SP and BP are denoted by Δ_1, Δ_2, and Δ_3, respectively.

	BoW	BP	Δ_1	SP	Δ_2	Δ_3
accept-except	99.17	99.76	0.59	99.80	0.63	0.04
affect-effect	97.51	98.80	1.29	98.85	1.34	0.05
among-between	88.57	90.56	1.99	90.99	2.42	0.43
amount-number	88.89	91.54	2.65	91.54	2.65	0.00
begin-being	97.77	98.61	0.84	98.60	0.82	-0.01
country-county	97.81	89.45	-8.36	97.81	0.00	8.36
fewer-less	78.67	78.79	0.11	80.79	2.11	2.00
I-me	97.17	99.27	2.11	99.27	2.10	-0.01
its-it's	94.72	96.91	2.19	96.89	2.16	-0.03
lead-led	94.60	96.90	2.30	96.92	2.32	0.02
maybe-may be	86.30	90.90	4.60	90.94	4.64	0.04
passed-past	96.12	98.71	2.59	98.75	2.62	0.03
peace-piece	97.20	97.68	0.48	97.2	0.00	-0.48
principal-principle	92.00	95.25	3.25	95.34	3.34	0.09
quiet-quite	96.07	98.68	2.60	98.68	2.60	0.00
raise-rise	90.72	97.39	6.67	97.23	6.51	-0.16
than-then	96.63	98.01	1.37	97.99	1.36	-0.01
weather-whether	97.25	98.90	1.65	98.95	1.70	0.05
your-you're	95.00	97.20	2.20	97.20	2.20	0.00
AVERAGE	93.80	95.44	1.64	95.99	2.19	0.55

this model encompasses both the BoW and the BP models as special cases because we obtain them when we set $\theta = 0$ and $\theta \to \infty$, respectively. Despite this, the classification performance with the SP model is slightly decreased compared to the performance with the BP model for some data sets (see Δ_3 in Table 1). This is possible, because the parameters selected using LOOCV with the training data are not necessarily optimal for the test data, that is, the parameter selection procedure overfits. Probably for this reason, we do not get the statistical significance for the performance difference of the BP and SP. However, the performance with SP model is always better or equal than that of the BoW model (see Δ_2 in Table 1) and the performance gain is statistically significant. The data set *country-county* favors the BoW model and this is detected by the parameter selection procedure, since it is the only one having an equal performance with the BoW and the SP models.

Next, we consider the Bayesian classifiers with the word transformations. The word transformation is constructed from the part-of-speech (PoS) information of words in the way described in Section 3. To obtain the PoS information, we used WordNet lookup, combined with the use of the WordNet *morphy* morphological analyzer for determining the PoS of inflected forms. All possibly applicable

Table 2. The naive Bayes classification performance with the basic position sensitive (BP) model compared to the performance of the Bayesian classifier with the word feature transformation (WT) (left), and to the performance of the Bayesian classifier with the combination (CB) of the word and the positional feature transformations (right). The performances are measured with AUC. The performance differences are denoted by Δ.

	BP	WT	Δ
accept-except	99.76	99.82	0.06
affect-effect	98.80	98.87	0.06
among-between	90.56	92.10	1.55
amount-number	91.54	91.37	-0.17
begin-being	98.61	98.81	0.19
country-county	89.45	91.97	2.52
fewer-less	78.79	82.66	3.87
I-me	99.27	99.34	0.07
its-it's	96.91	98.30	1.38
lead-led	96.90	97.49	0.59
maybe-may be	90.90	95.53	4.63
passed-past	98.71	98.96	0.25
peace-piece	97.68	98.63	0.95
principal-principle	95.25	96.15	0.90
quiet-quite	98.68	98.94	0.26
raise-rise	97.39	98.20	0.81
than-then	98.01	98.49	0.49
weather-whether	98.90	99.17	0.27
your-you're	97.20	98.32	1.12
AVERAGE	95.44	96.48	1.04

	BP	CB	Δ
accept-except	99.76	99.83	0.07
affect-effect	98.80	98.70	-0.10
among-between	90.56	93.91	3.36
amount-number	91.54	91.54	0.00
begin-being	98.61	98.81	0.20
country-county	89.45	97.44	7.99
fewer-less	78.79	80.26	1.47
I-me	99.27	99.27	-0.01
its-it's	96.91	98.16	1.25
lead-led	96.90	97.48	0.58
maybe-may be	90.90	95.52	4.61
passed-past	98.71	99.02	0.31
peace-piece	97.68	98.58	0.90
principal-principle	95.25	96.22	0.97
quiet-quite	98.68	98.95	0.27
raise-rise	97.39	97.99	0.61
than-then	98.01	98.48	0.47
weather-whether	98.90	99.14	0.24
your-you're	97.20	98.26	1.06
AVERAGE	95.44	96.71	1.28

parts-of-speech were assigned to words, for example, both the noun and verb PoS were assigned for the word *being*, which can be either a noun or an inflected form of the verb *be*. We used a table lookup to assign the PoS to the closed-class words, because they are not found in WordNet. Further, we included separate PoS tags for punctuation and numbers. Of the 10000 most common tokens, 8736 could be assigned at least one PoS using this procedure. The remaining 1264, consisting mostly of proper names but also containing, for example, abbreviations and multiword tokens such as *week-long*, were not assigned any PoS.

Similarly to the experiments with the positional transformations, we compare the performance of the word transformed Bayes classifier to the performance of the naive Bayes without the transformation, that is, the classifier with the BP model. Analogously to the positional transformation, we obtain the naive Bayes without transformations as a special case of the word transformed Bayes when we let $\mu \to \infty$. The results of the comparison are presented in Table 2. The performance gain between the word transformed Bayes and the Bayes without transformations is statistically significant.

Finally, we consider a Bayesian classifier together with the combination of the positional and the word transformation (see (7)). A performance comparison of the NB classifier without any transformations (i.e. with the BP model)

and the word and positionally transformed Bayesian classifier is presented in Table 2. The performance gain is statistically significant. Again, due to the overfitting of the parameter selection procedure, for some data sets the classification performance with the transformation combination is slightly lower than the performance without transformations. On average, the performance with the transformation combination is better than the performance when only the positional or the word transformations is used.

5 Discussion

In this paper, we experiment with linear transformations of the feature frequency count matrices together with the Bayesian classifiers. For the particular task of natural language disambiguation that we use as a model problem, we define two types of feature transformations. The positional transformation is obtained using a Gaussian kernel on the word positions, and the word transformation is constructed from the part-of-speech information of the words. The results of the experiments show that the performance of the Bayesian classifiers in the natural language disambiguation tasks can be improved with both types of the transformations. The proposed use of linear transformations is a promising research direction in general, because it provides an elegant way to incorporate external information into the classifier.

The Gaussian kernel for the positions and the part-of-speech information of words are just examples of the information that can be incorporated into the classifiers. In the future, we plan to investigate other possibilities to construct the transformations such as variable width Gaussian kernels for the word positions. Moreover, in addition to the part-of-speech information of the words, there are many other possibilities to define the word similarities from which the word transformations can be constructed. For example, techniques based on the latent semantic analysis [13] are popularly used in several natural language processing tasks.

The proposed use of linear feature transformations is, of course, not restricted to the Bayesian classifiers. The transformations can also be used to construct kernel functions that can be applied by the kernel based learning algorithms, such as support vector machines. While the naive Bayes classifier is usually faster to train and therefore useful in situations in which small computation times are of importance, the kernel based learning algorithms are at present considered to be the state-of-the-art. Further, in our earlier experiments with the senseval-3 data [3], we found that certain disambiguation problems prefer NB while others prefer support vector machines (NB was slightly better on average). We consider the possibilities of the kernel methods in another study [14].

Acknowledgments

This work has been supported by Tekes, the Finnish National Technology Agency. We would like to thank Jouni Järvinen for his helpful comments on this manuscript.

References

1. Pahikkala, T., Ginter, F., Boberg, J., Jarvinen, J., Salakoski, T.: Contextual weighting for support vector machines in literature mining: an application to gene versus protein name disambiguation. BMC Bioinformatics **6** (2005) 157
2. Pahikkala, T., Pyysalo, S., Ginter, F., Boberg, J., Järvinen, J., Salakoski, T.: Kernels incorporating word positional information in natural language disambiguation tasks. In Russell, I., Markov, Z., eds.: Proceedings of the Eighteenth International Florida Artificial Intelligence Research Society Conference, Clearwater Beach, Florida, AAAI Press, Menlo Park, California (2005) 442–447
3. Pahikkala, T., Pyysalo, S., Boberg, J., Mylläri, A., Salakoski, T.: Improving the performance of bayesian and support vector classifiers in word sense disambiguation using positional information. In Honkela, T., Könönen, V., Pöllä, M., Simula, O., eds.: Proceedings of the International and Interdisciplinary Conference on Adaptive Knowledge Representation and Reasoning, Espoo, Finland, Helsinki University of Technology (2005) 90–97
4. John, G.H., Langley, P.: Estimating continuous distributions in bayesian classifiers. In Besnard, P., Hanks, S., eds.: Proceedings of the Eleventh Annual Conference on Uncertainty in Artificial Intelligence, San Mateo, Morgan Kaufmann Publishers (1995) 338–345
5. Jurafsky, D., Martin, J.H.: Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition. Prentice Hall PTR, Upper Saddle River, New Jersey (2000)
6. Chen, S.F., Goodman, J.: An empirical study of smoothing techniques for language modeling. In Joshi, A., Palmer, M., eds.: Proceedings of the Thirty-Fourth Annual Meeting of the Association for Computational Linguistics, San Francisco, Morgan Kaufmann Publishers (1996) 310–318
7. Silverman, B.W.: Density Estimation for Statistics and Data Analysis. Chapman & Hall, London, UK (1986)
8. Hastie, T., Tibshirani, R., Friedman, J.H.: The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer Verlag, New York (2001)
9. Golding, A.R., Roth, D.: A winnow-based approach to context-sensitive spelling correction. Machine Learning **34** (1999) 107–130
10. Rose, T.G., Stevenson, M., Whitehead, M.: The Reuters Corpus Volume 1: From yesterday's news to tomorrow's language resources. In Rodriguez, M.G., Araujo, C.P.S., eds.: Proceedings of the Third International Conference on Language Resources and Evaluation, ELRA, Paris, France (2002)
11. Fawcett, T.: Roc graphs: Notes and practical considerations for data mining researchers. Technical Report HPL-2003-4, HP Labs, Palo Alto, California (2003)
12. Wilcoxon, F.: Individual comparisons by ranking methods. Biometrics **1** (1945) 80–83
13. Deerwester, S.C., Dumais, S.T., Landauer, T.K., Furnas, G.W., Harshman, R.A.: Indexing by latent semantic analysis. Journal of the American Society of Information Science **41** (1990) 391–407
14. Pahikkala, T., Pyysalo, S., Boberg, J., Järvinen, J., Salakoski, T.: Matrix representations, linear transformations, and kernels for natural language processing (2006) Submitted.