Mach Learn
DOI 10.1007/s10994-008-5082-6

# Matrix representations, linear transformations, and kernels for disambiguation in natural language

**Tapio Pahikkala · Sampo Pyysalo · Jorma Boberg · Jouni Järvinen · Tapio Salakoski**

Received: 22 December 2006 / Revised: 14 June 2007 / Accepted: 7 August 2008
Springer Science+Business Media, LLC 2008

**Abstract** In the application of machine learning methods with natural language inputs, the words and their positions in the input text are some of the most important features. In this article, we introduce a framework based on a word-position matrix representation of text, linear feature transformations of the word-position matrices, and kernel functions constructed from the transformations. We consider two categories of transformations, one based on word similarities and the second on their positions, which can be applied simultaneously in the framework in an elegant way. We show how word and positional similarities obtained by applying previously proposed techniques, such as latent semantic analysis, can be incorporated as transformations in the framework. We also introduce novel ways to determine word and positional similarities. We further present efficient algorithms for computing kernel functions incorporating the transformations on the word-position matrices, and, more importantly, introduce a highly efficient method for prediction. The framework is particularly suitable to natural language disambiguation tasks where the aim is to select for a single word a particular property from a set of candidates based on the context of the word. We demonstrate the applicability of the framework to this type of tasks using context-sensitive spelling error correction on the Reuters News corpus as a model problem.

**Keywords** Kernel methods · Feature transformations · Natural language processing · Natural language disambiguation

Editor: Dan Roth.

T. Pahikkala (✉) · S. Pyysalo · J. Boberg · J. Järvinen · T. Salakoski
University of Turku and Turku Centre for Computer Science (TUCS), 20014 Turku, Finland
e-mail: tapio.pahikkala@utu.fi

S. Pyysalo
e-mail: sampo.pyysalo@utu.fi

J. Boberg
e-mail: jorma.boberg@utu.fi

J. Järvinen
e-mail: jouni.jarvinen@utu.fi

T. Salakoski
e-mail: tapio.salakoski@utu.fi

## 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. When applying machine learning methods to natural language disambiguation, the text documents in which disambiguation is performed must be given a suitable representation. Such a representation needs to capture many features of the documents, and one of the most important choices is the selection of the representation that is given to the words in the documents. The most common choice for this purpose in many tasks is the bag-of-words (BoW) representation, in which the words are represented as word frequency vectors.

Although the BoW representation is simple and surprisingly effective in many cases, it discards much of the information contained in the original text, including information related to the positions of the words with respect to each other. However, positional information can be included, for example, by employing a representation in which words are replaced by word-position pairs (see e.g. Jurafsky and Martin 2000). This then means that identical words are considered as different features when they are located in different positions.

In this study, we propose a framework that allows systematic incorporation of word positions and facilitates the efficient use of similarity information of words and their positions. It brings together a number of our earlier proposals for incorporating positional information, discussed in detail in Sect. 6. The framework can be applied in tasks where word positions can be defined, and is particularly suitable when the aim is to disambiguate a single word based on its context by selecting a particular property of the word, such as the correct word sense or spelling, from a set of candidates. Similarly to how the BoW representation is commonly used together with more complex features, it might be beneficial to combine the use of the presented framework with other feature representations. However, as the word-position matrix representation extends the BoW and word-position feature representations, we only consider it in relation to these in this paper. The main contributions of the current work are summarized below.

- *Framework*. We describe a framework that is based on a word-position matrix representation of text, linear feature transformations of the word-position matrices, and kernel functions constructed from the transformations. This approach generalizes over previously introduced representations, including the BoW and word-position feature representations, and allows external sources of information on the similarity of words to be incorporated efficiently. The framework also naturally allows for combinations of word similarity and positional information to be applied.
- *Feature transformations*. We consider two categories of feature transformations: word and positional transformations. We describe how previously proposed techniques can be applied in the framework and introduce novel transformations, including ones derived directly from the training data. The performance gain obtained when using the framework with the introduced transformations is demonstrated in experiments using context-sensitive spelling error correction with the Reuters News corpus as a model problem.
- *Efficient computation*. We present efficient algorithms for training a kernel-based learning machine that uses the proposed framework. More importantly, we also introduce a method for the prediction of new examples with a trained linear regularized least-squares classifier that uses the framework, and show that prediction of the output variables can be performed as efficiently as in the BoW framework.

The paper is organized as follows. In Sect. 2, we briefly survey kernel methods and feature transformations, and present the representation of the input data points as word-position matrices. In Sect. 3, we introduce several types of linear feature transformations of the word-position matrices, namely, word and positional transformations, as well as an unsupervised method to construct the word and positional transformations from a data set. Section 4 concentrates on the computational issues related to the transformations and kernel functions constructed from the transformations. In Sect. 5, we describe experiments with the transformations and kernel functions. We discuss in detail several recently proposed kernels and their relationship to the framework proposed in this study in Sect. 6. Finally, we discuss the framework in a broader setting and describe possible extensions of the framework in Sect. 7. Section 8 concludes the paper.

## 2 Kernel methods and matrix representation of data

In this section, we first give a brief introduction to kernel methods and feature transformations. We then propose the word-position matrix representation of text.

### 2.1 Kernel methods

Kernel-based learning algorithms—see for example Schölkopf and Smola (2002) and Shawe-Taylor and Cristianini (2004)—can be viewed to consist of two modules: the learning algorithm and the kernel function. The learning algorithm can be, for example, a support vector machine (Vapnik 1998) or a regularized least-squares (RLS) classifier (Rifkin et al. 2003). The kernel function, which acts as an interface between the learning algorithm and the data, provides information to the learning algorithm in a form of the inner products of data points in some feature space into which the original data points are mapped. Formally, let $X$ denote the input space, which can be any set, and $H$ denote the feature space. For any mapping $\Phi : X \to H, k(x,z) = \langle \Phi(x), \Phi(z)\rangle$ is a kernel function. Note that if we have an efficient way to compute $k(x,z)$ directly, there is no need to explicitly compute the mapping $\Phi$, because the kernel-based learning algorithms need only the value of the inner product. In order to design a good kernel function for a particular learning problem, we may use prior knowledge of the problem (see e.g. Schölkopf et al. 1998 for a typical example of this approach).

In this paper, we consider only linear mappings of the data points. Let the mapping $\Phi$ be a linear transformation from $X$ to $H$, where $X$ is a vector space, and let the kernel function be

$$k(x,z) = \langle \Phi(x), \Phi(z)\rangle = \langle \widehat{\Phi}(x), z\rangle, \tag{1}$$

where $\widehat{\Phi} : X \to X$ is a symmetric positive semidefinite endomorphism. An endomorphism is a morphism from a vector space to itself. By symmetry, we indicate that, for any $x$ and $z$, $\langle \widehat{\Phi}(x), z\rangle = \langle x, \widehat{\Phi}(z)\rangle$. Positive semidefiniteness means that $\langle \widehat{\Phi}(x), x\rangle \geq 0$. Therefore, in this case an alternative way to compute the value of the kernel function for two data points is to transform only one of the two points with the endomorphism and compute its inner product with the non-transformed point. Below, we use the terms transformation and its corresponding endomorphism when referring to the correspondence between $\Phi$ and $\widehat{\Phi}$ defined in (1).

### 2.2 Representation of the data

The BoW model is a simple and common choice of text representation. In the context of kernel-based methods, the BoW representation of was first proposed by Joachims (1998) under the name word vector space kernel and since then it has been widely applied. A BoW vector is indexed with a set of words $\mathcal{W}$, that is, there is an element in the vector corresponding to each word in $\mathcal{W}$.

In this paper, we consider an alternate representation that is particularly appropriate to disambiguation tasks, which we use as a model problem. In disambiguation tasks, each input data point consists of a word to be disambiguated and the context words surrounding it. For this model problem, we use the following formalization for the data points. 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 is $r = 2s + 1$. If there are not enough words available in the text to the left or to the right from the ambiguous word, the positions are left empty. Let $n$ be the number of words and $\mathcal{W} = \{w_1, \dots, w_n\}$ be the set of words. We define our input data points to be word-position matrices $A \in \mathcal{M}_{n\times r}(\mathbb{R})$, 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 indexed from $-s$ to $s$, where the word to be disambiguated is at the position zero of its context. A word-position matrix $A$ generated from the context of an ambiguous word is a binary matrix in which the element $A_{i,j}$ corresponding to the word $w_i$ and the position $j$ has the value one if the word $w_i$ occurs at the position $j$ of the context and zero otherwise. 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 there is no word at that position, 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. Notice that BoW vectors can be created from the word-position matrices simply by summing up the matrix columns.

The word-position matrices are elements of the vector space $\mathcal{M}_{n\times r}(\mathbb{R})$. The Frobenius product is an inner product in this vector space

$$\langle A, B\rangle_F = \sum_{i,j} A_{i,j} B_{i,j}, \tag{2}$$

where $1 \le i \le n$ and $-s \le j \le s$.

## 3 Transformations of word-position matrices

In this section, we consider different ways to construct linear transformations for the word-position matrices. We start by separately introducing word and positional transformations in Sects. 3.1 and 3.2, respectively. Further, in Sect. 3.3 we present a method to derive transformations directly from a given data set. The composite transformations are presented in Sect. 4. The endomorphisms corresponding to the word and positional transformations (Propositions 1 and 2 in this section) are special cases of composite endomorphisms (Proposition 3). For more information on matrix analysis and linear algebra, see the book by Meyer (2000), for example.

### 3.1 Word transformations

We now consider linear transformations of word vectors. In the literature, such transformations are usually performed on BoW vectors to introduce similarities between words. Here we apply word-vector transformations on a word-position matrix $A$, so that each column of the matrix is transformed in the same way. An example of such word transformation is presented below.

Let the feature mapping in the context of kernel functions be

$$\Phi(A) = WA, \tag{3}$$

where $A \in \mathcal{M}_{n\times r}(\mathbb{R})$ is a word-position matrix, $W \in \mathcal{M}_{t\times n}(\mathbb{R})$ is here termed the word transformation matrix, and $t$ depends on the transformation. Then we obtain the following proposition.

**Proposition 1** *The endomorphism corresponding to the transformation* (3) *is*

$$\widehat{\Phi}(A) = \widehat{W}A,$$

*where* $\widehat{W} = W^T W \in \mathcal{M}_{n\times n}(\mathbb{R})$ *is the matrix of the endomorphism.*

The proposition is a special case of Proposition 3 proven in Sect. 4.

Typical linear transformations of BoW vectors are importance weights of words, whose use with support vector machines has been explored in detail by Leopold and Kindermann (2002) and Joachims (2002). The matrix $\widehat{W}$ corresponding to the importance weight transformation is a diagonal matrix whose elements are the importance weights of the words. By including also non-zero off-diagonal elements in the matrix $\widehat{W}$, word-word similarities can be taken into account in addition to the importance weights. There are several other potential sources of external information that can be used to construct the transformations. Siolas and d'Alché-Buc (2000) used WordNet as an external source of semantic similarity of words in order to perform semantic smoothing on BoW vectors. In addition, Gliozzo et al. (2005) applied WordNet Domains to construct the kernel functions for word sense disambiguation. Furthermore, word similarities could be obtained from a word-document matrix constructed from a large text corpus by latent semantic analysis techniques (Deerwester et al. 1990).

We now consider a simple example of a word transformation. For many natural language disambiguation tasks, the parts-of-speech (PoS) of context words are shown to provide useful additional information (see e.g. Jurafsky and Martin 2000). This external source of information can be used to construct a word transformation $W$ so that $W_{i,j}$ is one if the $j$th word can have the $i$th PoS and zero otherwise. For example, if $\mathcal{W} = \{$*composition*, *contribution*, *write*, *being*$\}$ and the possible PoS are *noun* and *verb*, then $n = 4, t = 2$, and we can construct the word transformation matrix and its corresponding endomorphism matrix as follows

$$W = \begin{pmatrix} 1 & 1 & 0 & 0.71 \\ 0 & 0 & 1 & 0.71 \end{pmatrix} \quad \text{and} \quad \widehat{W} = \begin{pmatrix} 1 & 1 & 0 & 0.71 \\ 1 & 1 & 0 & 0.71 \\ 0 & 0 & 1 & 0.71 \\ 0.71 & 0.71 & 0.71 & 1 \end{pmatrix}, \tag{4}$$

where the rows of $W$ are indexed by the parts-of-speech and the columns of $W$ as well as the rows and the columns of $\widehat{W}$ are indexed by the words. 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 either a noun or a verb. Note also that we have normalized the columns of $W$ so that the similarity of all words with themselves is 1 in the endomorphism matrix.

In addition to the PoS, any feature depending only on a word, not its context, can be included through word transformations in the way illustrated above. The fact that the size of the endomorphism matrix does not depend on the number of features included in this way allows the incorporation of a substantial amount of features extracted from the input words. To illustrate a common case of this type, let us consider the case of prefix and suffix features. By including word prefixes and suffixes of given lengths as features e.g. in a feature vector representation, it is possible to introduce similarity between words that would otherwise be considered completely distinct. As an example, consider a case where $\mathcal{W} = \{composition, contribution, cabinet, write\}$ and one- and two-character prefix features are included. Thus, the possible prefixes are $\{c, ca, co, w, wr\}$, $n = 4, t = 5$ and the word transformation and endomorphism matrices can be constructed as

$$W = \begin{pmatrix} 0.71 & 0.71 & 0.71 & 0 \\ 0 & 0 & 0.71 & 0 \\ 0.71 & 0.71 & 0 & 0 \\ 0 & 0 & 0 & 0.71 \\ 0 & 0 & 0 & 0.71 \end{pmatrix} \quad \text{and} \quad \widehat{W} = \begin{pmatrix} 1 & 1 & 0.71 & 0 \\ 1 & 1 & 0.71 & 0 \\ 0.71 & 0.71 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Note that the number of unique prefixes and suffixes grows with prefix/suffix length, and thus including prefix and suffix features of multiple different lengths increases the size of feature vectors for representing text to some extent. By contrast, when prefix and suffix features are equivalently included through a word transformation, the size of the endomorphism matrix is always $n \times n$, and, as discussed in Sect. 4.2, linear transformations can be incorporated in prediction without increasing computational complexity beyond that of the basic BoW representation. Thus, linearly calculable features, including by definition all features that can be directly calculated from the words without reference to their context, can in some cases be calculated more efficiently in the present framework than when naively incorporating the respective features in a feature vector representation.

The simple transformations presented above discard the original word identify information. In the first example, the words *composition* and *contribution* are identified with each other because their corresponding columns in $W$ are equal. This is a problem in practice because there are only a few PoS compared to the number of words, and therefore only applying this type of transformation loses information differentiating the words. Similar issues can occur also with other types of transformations, such as those introducing short word prefixes. These issues can be addressed by stacking the transformation with a identity transformation so that the feature space consists of the new features together with the old ones. Thus, in the above example, the transformed feature space will contain both word-position pairs as well as PoS-position pairs. To control the relative contribution of the two sets of features to the word similarities, we use a weighting parameter $\mu$ yielding the following block matrix

$$W_\mu = \begin{pmatrix} \sqrt{\mu} I \\ \sqrt{1-\mu} W \end{pmatrix}. \tag{5}$$

The endomorphism matrix corresponding to the stacked transformation matrix (5) is the following full rank word-word matrix

$$\widehat{W}_\mu = W_\mu^{\mathrm{T}} W_\mu = (1-\mu)\widehat{W} + \mu I, \tag{6}$$

that is, the endomorphism matrix corresponding to the original transformation with a shifted diagonal.

### 3.2 Positional transformations

We now consider positional transformations of the word-position matrices and relate our previous work on position-sensitive models to the proposed framework. One use of positional transformations is weighting, where some of the positions are considered to be more important than the others, analogously to importance weighting of words. For example, words that are close to the word to be disambiguated will often carry more useful information than distant words. In our previous work (Pahikkala et al. 2005a), we used positional weighting, originally introduced by Ginter et al. (2004), as a support vector kernel for disambiguation tasks. Positional transformations can be also used to "spread" an occurrence of a word in a certain position to other nearby positions, analogously to semantic smoothing techniques for words. For example, while the exact positions of words immediately before or after the word to be disambiguated are important, the exact positions of words far from the ambiguous word are less important, and therefore these distant positions can be considered similar to each other. We have previously proposed kernel functions that can be considered as implementations of this type of transformations (Pahikkala et al. 2005b, 2005c).

Linear positional transformations of the word-position matrices can be defined as

$$\Phi(A) = AP, \tag{7}$$

where $A \in \mathcal{M}_{n\times r}(\mathbb{R})$ is a word-position matrix, $P \in \mathcal{M}_{r\times u}(\mathbb{R})$ is here termed the positional transformation matrix, and $u$ depends on the transformation.

**Proposition 2** *The endomorphism corresponding to the transformation (7) is*

$$\widehat{\Phi}(A) = A\widehat{P},$$

*where $\widehat{P} = PP^{\mathrm{T}} \in \mathcal{M}_{r\times r}(\mathbb{R})$ is the position-position matrix of the endomorphism.*

The proposition is a special case of Proposition 3 proven in Sect. 4.

Note that if we set $P = (1\dots 1)^{\mathrm{T}}$, then the word-position matrices are transformed to BoW vectors. The weighted-BoW approaches used by Ginter et al. (2004) and Pahikkala et al. (2005a) can be implemented as a transformation $P \in \mathcal{M}_{r\times 1}(\mathbb{R})$ where the elements of the single column are the weights for the positions. Since there is only one column, the word-position matrices are transformed to weighted BoW vectors. Of course, positional weighting can be applied without "compressing" the positions to BoW by setting $P$ to be a diagonal matrix that has the weights on the diagonal. A simple example of a positional transformation that uses a combination of the BoW transform and positional weighting is given in the following along with the corresponding endomorphism matrix

$$P = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 2 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix} \quad \text{and} \quad \widehat{P} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 1 & 1 & 1 \\ 1 & 1 & 5 & 1 & 1 \\ 1 & 1 & 1 & 2 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix}, \tag{8}$$

where the context span $s$ is 2 and therefore the number of positions is $2s + 1 = 5$. The five rows of $P$ and the five rows and columns of $\widehat{P}$ correspond to the positions $-2, \dots, 2$. The first column of $P$ corresponds to the BoW transformation, that is, it maps each row of a word-position matrix to a feature that counts the occurrences of the word corresponding to the row. The three other columns correspond to features where a word occurs in a specific position near to the word to be disambiguated, which are often used in disambiguation tasks. Further, the weight given to the feature for the word at the position zero is higher than the weights for the positions $-1$ and 1, emphasizing the importance of that position.

In (8), the transformation is constructed by defining the values of the transformation matrix $P$. By doing this, we define the structure of the feature space into which the data points are mapped. However, in the context of kernel methods only the value of the inner product of the feature vectors is needed. Therefore, another way to construct the transformation is to define directly the values of the endomorphism matrix $\widehat{P}$. Below, we consider these two approaches of the transformation construction. First, we relate our previous work about the kernel functions on the word positions and use the kernels to define the endomorphism matrices. Next, we present a similar approach that we use to define the values of the transformation matrices.

In our previous work (Pahikkala et al. 2005b, 2005c), we have developed a kernel function that can be applied in the current framework to determine the elements of the endomorphism matrix $\widehat{P}$ without defining $P$. This approach is analogous to determining the endomorphism matrix corresponding to a word transformation based on an external source of word similarities. The support vector machine (SVM) classifier using the proposed kernel outperformed SVMs with the BoW and the weighted BoW kernels and SVM with no transformations performed on the word-position matrices. The applied endomorphism matrix was defined by

$$\widehat{P}_{p,q} = \exp(-\alpha(p^2 + q^2) - \theta(p - q)^2) + \beta, \tag{9}$$

where $\alpha$, $\theta$, and $\beta$ are parameters controlling three components of the function: weighting based on the distance of each position from the position zero, the effect of the distance of the positions from each other, and a BoW component, respectively. The parameter $\alpha$ controls the distance-based weighting so that for large values of $\alpha$ the value of the function decays faster than for small values as the distance of $p$ or $q$ from the position 0 increases. The parameter $\theta$ controls the effect of the mutual difference of the positions, so that for large values of $\theta$ the value of the function decays faster than for small values as the difference $p - q$ increases. Finally, $\beta$ corresponds to BoW, that is, a constant value given for any pair of positions. Note that the rows and columns of $\widehat{P}$ are indexed by positions, which range from $-s$ to $s$. Some parameterizations of (9) are illustrated in Fig. 1. The level curves in the general case (top left) are ellipses whose major axes are always parallel with the line $p = q$. The ellipses are circles if $\theta$ is zero (top right), giving weighting similar to the weighted BoW transformation. If we omit the positional weighting by setting $\alpha$ to zero (bottom left), we have a Gaussian kernel of positions which depends solely on the distances of the positions from each other. This can be considered as equivalent to an intermediate between the BoW transformation and the case where no positional transformations are made. If we let $\theta \to \infty$ (bottom right), the function is equivalent to a transformation that only weights the positions. For more detailed analysis of the function and its parameters, we refer to Pahikkala et al. (2005c).

We now present a procedure to construct $P$ inspired by kernel density estimation techniques (see e.g. Silverman 1986). In this approach, the feature space equals the input space and therefore we use the term word-position matrix also when referring to the transformed data points. The idea is to construct a transformation that spreads the word occurrences from their exact positions to the positions that are related to the original one. In other words, if a word is originally at a position $p$ when there is one nonzero element in the row corresponding to that word in the word-position matrix, the corresponding row of the transformed matrix contains also other nonzero values, of which positions with large values are more related to the position $p$ than positions with small values. A well-known way to perform this kind of transformation is the Gaussian function. Further, a common practice in density

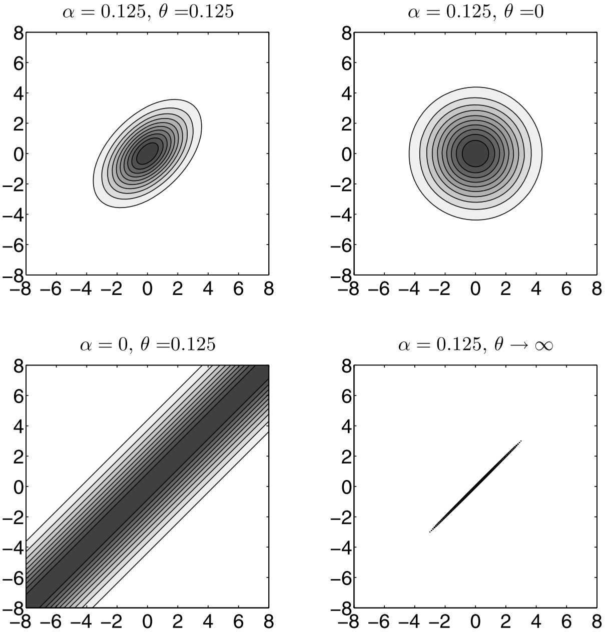

**Fig. 1** Gray-scale illustrations of the position-position matrices obtained using (9) with different parameterizations. Position pairs associated with larger values have *darker color* than those with smaller values

estimation is to use variable Gaussian widths that depend on the amount of available data in different regions.

Next, we construct a Gaussian function $\kappa_p(q)$ whose width depends on the word position $p$ so that the Gaussians used to spread the word occurrences in the vicinity of the position zero are sharp whereas the distant occurrences are spread more evenly to the other positions due to the wide Gaussians. We construct the positional transformation matrix as $P_{p,q} = \kappa_p(q)$. Note that again $P$ is indexed by positions and that the columns of $P$ determine how much the word occurrences are spread to the neighboring positions. The following realization of the function has a variable width controlled by a function $h(p)$:

$$\kappa_p(q) = \frac{\exp(-h(p)(p-q)^2)}{Z_p}, \tag{10}$$

where

$$Z_p = \sum_{-s \le q \le s} \exp(-h(p)(p-q)^2)$$

is a normalization constant for the columns of $P$. We use the Gaussian function $h(p) = \exp(-\gamma p^2)$, where $\gamma \ge 0$ is a parameter controlling the rate of spreading of positions with respect to the distance of the position 0. The matrices $P$ and $\widehat{P}$ obtained using (10) with different values of $\gamma$ are depicted in Fig. 2. In the illustrations of the transformation matrix $P$, the vertical axis corresponds to the original positions and the horizontal axis to the transformed ones. The variable width Gaussian functions determine how the word occurrences in the original positions are spread to other positions. The narrow ridge in the middle means that the occurrences in the nearby positions are spread only to the positions that are close to them while the distant occurrences are spread almost uniformly to the other positions.

From the depictions of the endomorphism matrices we observe the resemblance to the matrices obtained from the endomorphism (9). Close position pairs are again given larger

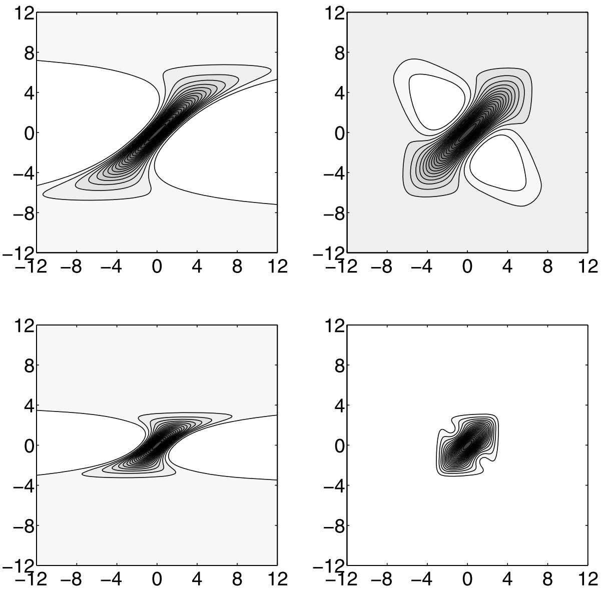

**Fig. 2** Gray-scale illustrations of the matrices $P$ (*left*) and $\widehat{P}$ (*right*) obtained using (10) with $\gamma = 0.125$ (*top*) and $\gamma = 0.5$ (*down*). Position pairs associated with larger values have *darker color* than those with smaller values

weight than the distant ones and the close position pairs in which the positions are on different sides of the word at position zero are penalized even more strictly than with (9). We speculate that the presented procedure is more advantageous than the one obtained by using (9) because in (10) there is only one parameter to be selected. In our experiments, we notice that the classification performance with the variable width Gaussian transformation is as good as with the previous one while the parameter selection requirements are considerably smaller.

### 3.3 Unsupervised transformations

The word transformations can be constructed using external sources of information, such as the parts-of-speech of the words, and the positional transformations are designed based on prior knowledge of the task. Information on word similarities can also be obtained directly from training data. For example, Wong et al. (1985) developed the generalized vector space model, in which word similarities were obtained from word co-occurrences in documents (see Shawe-Taylor and Cristianini 2004 for a more thorough discussion). This model uses a document-word matrix to represent the information. Inspired by this model, we will now consider a method that allows to construct not only the word but also the positional transformations automatically. By contrast to the generalized vector space model, we use word-position matrices. Let $A_i$ denote the $i$th word-position matrix in a data set and let

$$D = \sum_i A_i \tag{11}$$

be a word-position matrix obtained by summing all word-position matrices. Note that the construction of $D$ does not need the class information of the examples, and therefore we can use much more examples for that purpose (also the examples to be predicted) than we use to train the supervised machine learning methods. This is, because the unclassified data is often in greater abundance than the labeled training data. Some preprocessing steps

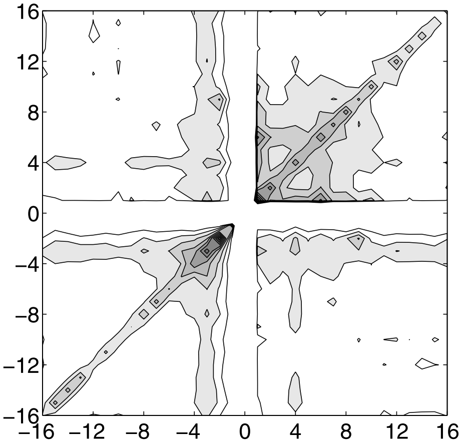

**Fig. 3** Gray-scale illustration of an example of an automatically generated $\widehat{P}$ from a confusion set *principal-principle*. Position pairs associated with larger values have *darker color* than those with smaller values

that are necessary when constructing transformations from $D$ are discussed in Sect. 5. The transpose of the obtained matrix can directly be used as a word-transformation $W = D^{\mathrm{T}}$ or as a positional transformation $P = D^{\mathrm{T}}$, whose corresponding endomorphisms are $\widehat{W} = DD^{\mathrm{T}}$ and $\widehat{P} = D^{\mathrm{T}}D$, respectively.

Let us consider the positional transformation $P = D^{\mathrm{T}}$ and the corresponding position-position matrices. We can think the $p$th column of $D$ as an empirical estimate of the word probability mass function of the position $p$. We observe that the value of an element of $\widehat{P}$ corresponding to a certain pair of positions is large if the word distributions of the two positions are similar and the distributions have low entropy, that is, the probability mass is substantially concentrated on a few words. The entropies of the word distributions of the nearby positions are very low compared to those of the distant positions and therefore the nearby positions get weighted much more than the distant ones which can be seen from the diagonal elements of $\widehat{P}$. This is analogous with the observations made by Yarowsky (1993, 1995). The word distributions in distant positions are close to the word distribution of the whole data set which makes the off-diagonal elements of $\widehat{P}$ corresponding to the pairs of distant positions medium-sized. On the other hand, some of the nearby position pairs have very small off-diagonal elements in $\widehat{P}$ because of the different low-entropy word distributions of the positions.

An example of a position-position matrix $\widehat{P}$ generated for a confusion set *principal-principle* is depicted in Fig. 3. Because of the nature of the task of context-sensitive spelling error correction, the ambiguous word has been removed from the contexts used to generate the matrix $D$. Therefore, there are only zero elements in the corresponding row and column of $\widehat{P}$. Of course, this removal may not be necessary in other types of tasks.

An unsupervised word transformation can be constructed analogously to the positional transformation. The $w$th row of $D$ can be considered as an empirical estimate of the position probability mass function of the word $w$. Note that the sum of the elements of a row depends of the frequency of the word, potentially causing high frequency words to get excessive weights—this effect can be removed by normalizing the rows. The value of an element of $\widehat{W}$ corresponding to a certain pair of words is large if the position distributions of the two words are similar and the distributions have low entropy, that is, the probability mass is substantially concentrated on a few positions. For example, if the ambiguous word is a singular noun, the words *is* and *was* may tend to occur frequently at the position one, while the occurrences of the word *are* at that position would be rare. Therefore, for that kind of disambiguation task, the words *is* and *was* would be considered similar, while *is* and *are* would be different. The practical aspects of applying the unsupervised transformations are discussed in more depth in Sect. 5.

## 4 Composite transformations and their computational complexity

In the previous sections, we considered word and positional transformations of the word-position matrices as well as their corresponding endomorphisms. Here we study composite transformations, that is, transformations that combine word and positional transformations. We additionally present efficient techniques for computing the kernel matrices and performing prediction for unseen examples, as well as discuss their computation complexity.

### 4.1 Composite transformations

To obtain the benefits of both the word and positional transformations, we can consider kernel functions over word-position matrices that correspond to a composite transformation of the two transformations. As above, we also show that the feature mappings of the data points can be considered from both the transformation and from the endomorphism point of view. In this case, the feature mapping for the input data points is

$$\Phi(A) = WAP, \tag{12}$$

where $A \in \mathcal{M}_{n\times r}(\mathbb{R})$ is a word-position matrix, $W \in \mathcal{M}_{t\times n}(\mathbb{R})$ is a word transformation matrix, and $P \in \mathcal{M}_{r\times u}(\mathbb{R})$ is a positional transformation matrix. Now $t$ and $u$ determine the dimension $tu$ of the feature space $H$ into which the word-position matrices are mapped.

We note that the explicit computation of the mapping and the inner products of the transformed word-position matrices is computationally very demanding. However, the computational complexity can often be decreased by applying the following proposition. A more general version of the proposition is proved by Vishwanathan et al. (2006). We present our proof for the proposition to gain more insight into computational complexity considerations.

**Proposition 3** *Let $\Phi(A) = WAP$ as in (12). Then the endomorphism of the input space corresponding to $\Phi$ as stated in (1) is*

$$\widehat{\Phi}(A) = \widehat{W}A\widehat{P}, \tag{13}$$

*where $\widehat{W} = W^{\mathrm{T}}W$ and $\widehat{P} = PP^{\mathrm{T}}$.*

*Proof* We show that $\langle\Phi(A), \Phi(B)\rangle_F = \langle\widehat{\Phi}(A), B\rangle_F$, and that $\widehat{\Phi}$ is symmetric and positive semidefinite. We begin by noting that the Frobenius product of matrices can be written as

$$\langle A, B\rangle_F = \mathrm{tr}(A^{\mathrm{T}}B),$$

where $\mathrm{tr}(M)$ denotes the trace of a square matrix $M$. Then

$$\begin{aligned}\langle\Phi(A), \Phi(B)\rangle_F &= \langle WAP, WBP\rangle_F\\ &= \mathrm{tr}(P^{\mathrm{T}}A^{\mathrm{T}}W^{\mathrm{T}}WBP)\\ &= \mathrm{tr}(PP^{\mathrm{T}}A^{\mathrm{T}}W^{\mathrm{T}}WB)\end{aligned}$$

$$\begin{aligned} &= \langle \widehat{W} A \widehat{P}, B \rangle_F \\ &= \langle \widehat{\Phi}(A), B \rangle_F, \end{aligned} \tag{14}$$

where (14) follows due to the cyclic property of the trace, that is, the trace of a product of matrices is equal to the trace of any cyclic permutation of the product provided that the matrices are permutated so that all products can be performed. The tenability of the equality $\langle \Phi(A), \Phi(B) \rangle_F = \langle A, \widehat{\Phi}(B) \rangle_F$ can be shown in a similar way.

Let vec be the vectorization operator which stacks the columns of a matrix in a column vector and let $\otimes$ denote the Kronecker product of matrices (see e.g. Magnus 1988). We observe that

$$\begin{aligned} \langle \widehat{\Phi}(A), B \rangle_F &= \mathrm{vec}(\widehat{W} A \widehat{P})^{\mathrm{T}} \mathrm{vec}(B) \\ &= \mathrm{vec}(A)^{\mathrm{T}} (\widehat{P} \otimes \widehat{W}) \mathrm{vec}(B), \end{aligned} \tag{15}$$

for the equality (15); see Magnus (1988), for example. The matrix $\widehat{P} \otimes \widehat{W}$ is symmetric and positive semidefinite, since it can be decomposed as $(P^{\mathrm{T}} \otimes W)^{\mathrm{T}} (P^{\mathrm{T}} \otimes W)$, and therefore $\langle \widehat{\Phi}(A), A \rangle_F \geq 0$ which proves the symmetry and positive semidefiniteness of $\widehat{\Phi}$. □

Next we consider how we can take advantage of the sparsity of the word-position matrices in order to speed up the computations. Because a context has at most one word in each position, there is at most one nonzero element in each column of the corresponding word-position matrix, and hence there are at most $r$ nonzero elements in the whole matrix. Thus, a fast way to compute the Frobenius product of two word-position matrices of which the other is not transformed, is to go through the list of nonzero elements of the non-transformed matrix and calculate the corresponding sum of products. A sparse word-position matrix is also fast to multiply by any transformation matrix. For example, if we transform a sparse word-position matrix with a rank $t$ word transformation matrix $W$, we need $O(rt)$ floating point operations because we sum one column of $W$ per each nonzero element of the word-position matrix. On the other hand, if the word-position matrix is already multiplied with a transformation matrix once, the sparsity property is lost and the subsequent matrix multiplications will be computationally more complex.

### 4.2 Kernel matrix computation

We now consider the computational complexity of the calculation of the kernel matrices when the transformations defined above are performed on the word-position matrices. We assume that the number of words $n$ is always larger than the context length $r$.

The kernel matrix used to train a kernel-based learning algorithm is the following square matrix of $m^2$ elements

$$K_{i,j} = k(x_i, x_j), \quad 0 \leq i, j \leq m,$$

where $x_i$ are training examples and $m$ is the size of the training set. The naive approach would be to calculate the Frobenius product of the transformed word-position matrices independently for each element of the kernel matrix. However, it is often possible to speed up the computation by transforming one training example at a time and calculating the elements of the corresponding row of the kernel matrix provided that the Frobenius product is faster to compute when one of the training examples is already transformed. Below we will discuss some situations when this is the case.

We consider two different ways to compute the kernel when both the word and the positional transformations are used (of course, even faster computational shortcuts can be found when only one of the two transformations, or no transformation, is applied). According to the Proposition 3, we can compute the kernel function in the following two ways

$$k(x,z) = \langle WAP, WBP\rangle_F \tag{16}$$

$$= \langle \widehat{W}A\widehat{P}, B\rangle_F, \tag{17}$$

where $A$ and $B$ are the word-position matrices corresponding to the data points $x$ and $z$, respectively. The computational complexity of kernel matrix calculation using (16) is

$$O(rtum + tum^2). \tag{18}$$

The first term of (18) corresponds to the matrix product of the matrix $WA \in \mathcal{M}_{t\times r}(\mathbb{R})$ with the matrix $P \in \mathcal{M}_{r\times u}(\mathbb{R})$ that is calculated for each training example (the complexity of the product $WA$ is $O(tr)$ due to the sparsity of $A$, and has no effect on the complexity). The second term corresponds to the Frobenius product of the matrices $WAP \in \mathcal{M}_{t\times u}(\mathbb{R})$ and $WBP \in \mathcal{M}_{t\times u}(\mathbb{R})$ computed for each element of the kernel matrix. The memory complexity of (16) is $O(tum)$ because we need to store $m$ dense $\mathcal{M}_{t\times u}(\mathbb{R})$-matrices into the memory.

The kernel (17) can be implemented using the following simple kernel function

$$k(x,z) = \sum_{p,q} \widehat{W}_{x(p),z(q)}\widehat{P}_{p,q},$$

where $x(p)$ and $z(p)$ denote the indices of the words in $\mathcal{W}$ that the contexts $x$ and $z$ have at the position $p$, respectively. The above kernel can be computed in $O(r^2)$ time for each of the $m^2$ elements of the kernel matrix provided that we have the matrices $\widehat{W}$ and $\widehat{P}$ stored in the memory. Therefore, the complexity of kernel matrix calculation using (17) is

$$O(r^2m^2). \tag{19}$$

Clearly, the magnitudes of the complexities (18) and (19) depends crucially on the matrix dimensions $t$ and $u$. Before discussing which of the alternatives (16) and (17) should be used in different cases, we consider two ways by which we can speed up the calculation of the alternative (16).

Firstly, we can assume that $t$ and $u$ are equal to the ranks of the matrices $W$ and $P$, respectively, and hence $t \le n$ and $u \le r$. This can be ensured by the following application of singular value decomposition to reshape the matrices. The singular value decomposition of, for example, a word transformation matrix $W \in \mathcal{M}_{t\times n}(\mathbb{R})$ is

$$W = U\Sigma V^{\mathrm{T}},$$

where $\Sigma \in \mathcal{M}_{n\times t}(\mathbb{R})$ is a diagonal matrix that contains the singular values of $W$. Here $U \in \mathcal{M}_{t\times t}(\mathbb{R})$ and $V \in \mathcal{M}_{n\times n}(\mathbb{R})$ are orthogonal matrices that contain the left and right singular vectors of $W$, respectively. The corresponding word-word matrix is

$$\widehat{W} = W^{\mathrm{T}}W = V\Sigma^{\mathrm{T}}U^{\mathrm{T}}U\Sigma V^{\mathrm{T}} = V\Sigma^{\mathrm{T}}\Sigma V^{\mathrm{T}} \tag{20}$$

because $U$ is orthogonal. Its eigenvalues are the diagonal elements of the matrix $\Sigma^{\mathrm{T}}\Sigma$. Note that the number of nonzero singular values equals the rank of the matrix. Therefore, we can also set $t$ to be equal to the rank, because the transformation matrix $W$ can be reshaped as $W := \Sigma^t V^{\mathrm{T}}$, where $\Sigma^t$ contains only the $t$ rows of $\Sigma$ with the nonzero singular values. This reshaping does not change the matrix of the endomorphism, because ${\Sigma^t}^{\mathrm{T}}\Sigma^t = \Sigma^{\mathrm{T}}\Sigma$. Positional transformation matrices can be reshaped analogously. If $L \in \mathcal{M}_{r\times r}(\mathbb{R})$ and $\Lambda \in \mathcal{M}_{r\times u}(\mathbb{R})$ contain the left singular vectors and singular values of $P$, respectively, then we set $P := L\Lambda^u$, where $\Lambda^u$ contains only the $u$ columns of $\Lambda$ with the nonzero singular values. The transformations can be performed more efficiently when the transformation matrices are compressed in the above described way.

Secondly, we consider speeding up the computation of (16) when the word transformation is stacked with an identity transformation as in (5). The stacking does not increase the computational complexity, because we first compute the value of the kernel with the low rank word transformation and then add the value of the kernel with the positional transformation only. This can be stated formally as:

$$\langle W_\mu AP, W_\mu BP\rangle_F = \langle WAP, WBP\rangle_F + \mu\langle A\widehat{P}, B\rangle_F, \tag{21}$$

where $W_\mu$ is the transformation defined in (5) and $\mu$ is the diagonal shift parameter. The matrices $A\widehat{P}$ for each training example can be computed and stored in memory in $O(r^2m)$ time and the second term of (21) can then be efficiently computed due to the sparsity of $B$.

We conclude that in most cases the alternative (16) is a better choice when the product of the ranks of the transformation matrices are low compared to the square of the context length. This is the case, for example, when the matrix $W \in \mathcal{M}_{t\times n}(\mathbb{R})$ is a mapping from the words to their part-of-speech information, because typically only about ten part-of-speech are used. The rank of the positional transformation is low, for example, when the all the word occurrences in the distant positions are mapped to bag-of-words kind of features as in (8). On the other hand, if the extracted contexts are short, when we only use the information of the nearby words and not the word distribution of the whole document for example, the alternative (17) may be preferred instead.

### 4.3 Prediction of unseen examples

Above we analyzed the time complexity of the kernel matrix calculation in several different cases because most of the kernel-based learning algorithms, such as support vector machine or regularized least-squares for example, are trained with the kernel matrix generated from the training examples. We now consider the prediction of new examples. When using the traditional dual formulation of the kernel-based learners, the prediction of the output of a new data point $z$ needs the calculation of the values of the kernel function between the new data point and the training examples

$$f(z) = \sum_x \alpha_x k(z, x),$$

where $x$ ranges over the training examples and $\alpha_x \in \mathbb{R}$ is the weight (see e.g. Vapnik 1998) of the training example $x$. Because of the linearity of the transformations, we also have a more efficient possibility to predict new data points. Recall that when using a linear kernel-based learner, we can calculate and store in memory the normal vector $M$ of the hyperplane corresponding to the learned solution

$$M = \sum_x \alpha_x \Phi(x) = \Phi\left(\sum_x \alpha_x x\right).$$

In our case, the normal vector $M$ is a word-position matrix. The prediction of a new data point $z$ would then be performed by calculating directly the Frobenius product $\langle \Phi(z), M\rangle_F$ of the mapped data point $\Phi(z)$ with the matrix $M$. Due to the sparsity of the word-position matrix $z$ and the Proposition 3, a faster prediction time is achieved by storing in the memory the following word-position matrix

$$\widehat{M} = \sum_x \alpha_x \widehat{\Phi}(x) = \widehat{\Phi}\left(\sum_x \alpha_x x\right) = \widehat{W}\left(\sum_x \alpha_x x\right)\widehat{P}. \tag{22}$$

Given $\widehat{M}$, the prediction of a new data point $z$ can be performed by calculating the Frobenius product $\langle z, \widehat{M}\rangle_F$ of the original data point $z$ with the matrix $\widehat{M}$. However, the matrix $z$ contains only at most $r$ nonzero elements, and therefore the prediction can be done via

$$f(z) = \sum_p \widehat{M}_{z(p),p}, \tag{23}$$

where $z(p)$ denotes the index of the word in $\mathcal{W}$ that the context $z$ has at the position $p$. When the context information of the data point $z$ is stored in memory as a list of $r$ word-position pairs, the computational complexity of predicting the label of a new data point with (23) is only $O(r)$. This form thus allows a considerably more efficient calculation of the kernel function in prediction than (16), where the transformation is be applied to the new data points. Using this form, in the framework prediction thus requires no more operations than prediction using a linear kernel and the very basic BoW vector representation.

## 5 Evaluation

To evaluate the performance of the proposed kernels, we apply them to the model disambiguation problem of context-sensitive spelling correction, where disambiguation is done at the level of words. A misspelling of an original word may belong to the language, such as, for example, *desert* misspelled as *dessert*. This kind of mistake cannot be detected by standard lexicon-based checkers, since *dessert* 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, {*piece*, *peace*} can be considered as a binary confusion set. In the context-sensitive spelling correction task, the correct spelling of a word must be disambiguated among the alternative spellings in the confusion set based on its context. 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.

To form the datasets, we use all 19 binary confusion sets among the 21 sets of commonly confused words used by Golding and Roth (1999) in their context-sensitive spelling correction experiments. We create the datasets from the Reuters News corpus (Rose et al. 2002), 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 at least one of the confusion set words. In each such document, every occurrence of a confusion set word 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 disregarded 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, we label each selected example, and 'remove' from the context of each example the confusion set word.

As a kernel based classification algorithm we use regularized least-squares (RLS) (Rifkin et al. 2003) that has been shown to have classification performance equivalent to the well-known support vector machines. The algorithm for learning a classification function $f : X \to \mathbb{R}$ that maps the input vectors $x \in X$ to real values can be considered as a special case of the following regularization problem known as Tikhonov regularization (for a more comprehensive introduction, see e.g. Rifkin 2002; Vapnik 1998):

$$\min_f \sum_{i=1}^{m} (f(x_i) - y_i)^2 + \lambda \|f\|_k^2, \tag{24}$$

where $x_i \in X$ and $y_i \in \{+1, -1\}$ are the training data points and their corresponding labels, $\lambda \in \mathbb{R}_+$ is a regularization parameter, and $\|\cdot\|_k$ is the norm in the reproducing kernel Hilbert space defined by a positive definite kernel function $k$. The second term is called a regularizer. The learned classification function can be used to predict positive or negative labels by selecting a threshold value.

By the representer theorem, the minimizer of (24) has the following form:

$$f(x) = \sum_{i=1}^{m} \alpha_i k(x, x_i),$$

where $\alpha_i \in \mathbb{R}$ and $k$ is the kernel function associated with the reproducing kernel Hilbert space mentioned above. Given a regularization parameter $\lambda$, a vector $\alpha \in \mathbb{R}^m$ that contains the coefficients $\alpha_i$ for the training examples is obtained as follows

$$\alpha = (K + \lambda I)^{-1} Y,$$

where $I$ is an identity matrix, $K$ is the kernel matrix, and $Y \in \mathbb{R}^m$ contains the labels $y_i$ of the training examples (for a proof, see e.g. Rifkin 2002). Our implementation of RLS is explained in more detail in Pahikkala et al. (2006b).

Because all of the extracted examples may not have as many context words, for example, when the ambiguous word is the first word of the document, we normalize the data in the input space as follows

$$\widetilde{A} = \frac{A}{\sqrt{\langle A, A \rangle_F}}$$

where $A$ is a binary word-position matrix corresponding to an extracted data point and $\widetilde{A}$ is the normalized one.

We measure the performance of the classifier with different kernels with the area under the ROC curve (AUC) (see e.g. Fawcett 2003), since it is shown to be a better performance measure than simple accuracy/error rate (see e.g Provost et al. 1998; Ling et al. 2003) for selecting between classifiers. AUC is preferred in particular in cases where the class distribution in skewed, and is equivalent to the Wilcoxon-Mann-Whitney statistic, that is, the probability that given a randomly chosen positive and negative example, the classifier can correctly determine which is which (Cortes and Mohri 2004). Formally,

$$\text{AUC} = \frac{1}{m_+ m_-} \sum_{y_i=+1, y_j=-1} \delta(f(x_i) > f(x_j)), \tag{25}$$

where $\delta(e)$ is one when the expression $e$ is true and zero otherwise, and $m_+$ and $m_-$ are the number of positive and negative examples, respectively. Note that as AUC is a symmetric measure, it makes no difference how the positive and negative class labels are assigned. While performance evaluation with AUC does not require a classification threshold value to be set, a suitable threshold can be selected when necessary by, for example, cross-validation. The statistical significance of the performance differences between the various transformations and the baseline method is tested using the Wilcoxon signed-ranks test (Wilcoxon 1945).

To select parameters, we first use the training set to select the context span, transformation parameters, and the RLS regularization parameter with a coarse grid search and ten-fold cross-validation, separately for each confusion set. The best performing parameter combinations are then selected and the classifiers are then trained with the training set. Finally, the performance of the classifiers is evaluated on the test set.

We start by comparing the RLS classification performance with the word-position matrix representation of the data to the performance with the bag-of-words (BoW) representation. The results of the comparison are presented in Table 1. The results with BoW representation are considerably worse than with the word-position representation and therefore, we consider the word-position representation without transformations as the baseline (B) method. Our earlier evaluation of BoW and the baseline method (Pahikkala et al. 2005b) using the Senseval-3 dataset gave different results, with the BoW representation outperforming the basic word-position matrix representation on 8 datasets out of 14 and performing better on average. The amount of training examples available in the Senseval-3 dataset was smaller than used here and we speculate that this difference and the use of the positional information of the words may, at least in part, account for the different results.

The tests of the different transformations are divided into three subsections. First, we test the classification performance with the positional transformations and then we test the word transformations. Finally, the classifiers using both the word and the positional transformations are tested. The RLS classification performances with the different transformations and their combinations are compared to the baseline in which no transformations are used. Naturally, the optimal context span and regularization parameter values are selected also for the baseline method. The performance level of the baseline method turned out to be very high for most of the confusion sets and hence there is not too much room for improvement. Nevertheless, we found statistically significant performance improvements for most of the experiments.

### 5.1 Experiments with positional transformations

We compare the baseline B to three different positional transformations or kernel functions: the BoW transformation, the unsupervised positional (UP) transformation described in Sect. 3.3, and the variable width (VW) Gaussian transformation obtained from the function (10). In preliminary experiments, we found out that the performance of VW is as good as the performance of our previous position-sensitive transformation (Pahikkala et al. 2005b, 2005c) corresponding (9). Because VW has only one parameter compared to the three parameters of (9), we decided to only use the VW transformation. The unsupervised positional transformation is constructed by summing up 10000 word-position matrices of unlabeled data. Each word-position matrix $D$ used in the UP-transformation experiments is smoothed with $\log(D + 1)$ and its columns are normalized with their 1-norms. The test results are presented in Table 1.

An interesting result is that while UP-transformation has no parameters at all it performs almost as well on average as the VW-transformation which has one parameter. Therefore,

**Table 1** The RLS classification performances with different positional transformations and kernels, namely, the bag-of-words (BoW), no-transformations baseline (B), unsupervised positional (UP) transformation, and variable width Gaussian (VW) transformation

| | BoW | B | UP | VW |
|---|---|---|---|---|
| I-me | 97.13 | 99.13 | **99.25** | 99.25 |
| accept-except | 99.38 | 99.80 | 99.82 | **99.86** |
| affect-effect | 97.88 | **98.64** | 98.55 | 98.57 |
| among-between | 91.94 | 94.28 | 95.37 | **95.58** |
| amount-number | 90.75 | 92.00 | 92.61 | **93.64** |
| begin-being | 97.43 | 98.97 | 98.75 | **99.03** |
| country-county | 92.77 | 93.14 | 98.01 | **98.38** |
| fewer-less | 72.44 | 78.13 | **83.05** | 80.18 |
| its-it’s | 93.01 | 97.08 | 98.38 | **98.67** |
| lead-led | 95.30 | 98.06 | **98.24** | 98.06 |
| maybe-may be | 88.30 | 94.93 | 96.19 | **96.26** |
| passed-past | 96.55 | 98.85 | 98.62 | **99.17** |
| peace-piece | 96.19 | 96.79 | 96.56 | **96.84** |
| principal-principle | 92.24 | 95.59 | 96.21 | **96.55** |
| quiet-quite | 95.68 | 98.53 | **98.82** | 98.58 |
| raise-rise | 93.82 | 98.41 | 98.68 | **98.72** |
| than-then | 97.69 | 99.05 | 99.02 | **99.33** |
| weather-whether | 97.58 | 99.22 | **99.65** | 99.59 |
| your-you’re | 94.69 | 97.02 | 96.71 | **97.43** |
| Average | 93.73 | 96.19 | 96.97 | 97.04 |

the UP-transformation, which also performs considerably better on average than the baseline and BoW, is an useful alternative when we have enough data available to construct it. Every positional transformation gives a statistically significant performance improvement over the baseline B except the BoW transformation which is statistically significantly worse than the baseline.

### 5.2 Experiments with word transformations

Three word transformations are compared against the baseline B, namely, the part-of-speech (PoS), WordNet Domains (WD) (see Magnini and Cavaglià 2000; Bentivogli et al. 2004), and the unsupervised word (UW) transformation described in Sect. 3.3. These transformations represent simple examples of word transformations and more sophisticated ones may be obtained using, for example, latent semantic indexing techniques. The unsupervised word transformation is constructed in the same manner as the unsupervised positional except that the rows of the matrix $D$ are normalized with their 2-norms. We observed in preliminary experiments that using the unsupervised transformation matrix without normalization gives excessive weight to common words (including common stop words such as *a*, *the* and *of*), leading to decreased classification performance. Further experiments suggested that the use of the 2-norm gives somewhat better results than the use of the 1-norm.

Note that there is an additional degree of freedom when selecting the context span for the word-position matrices used to construct the unsupervised transformation. This span determines the number of features in the vector each word is mapped to, and it should not be confused to the context span parameter used to construct the word-position matrices of the data points. We do not perform an extensive search for this context span, but preliminary results suggest that it is of minor importance. In the reported experiments, we set the span so that it encompasses 8 words to the left and right of the word to be disambiguated.

To determine the possible PoS of nouns, verbs, adjectives and adverbs for the PoS transformation, we used WordNet lookup, combined with the use of the WordNet *morphy* morphological analyzer for determining the PoS of inflected forms. All possibly applicable parts-of-speech were assigned to words, and no disambiguation was attempted. 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*. Closed-class words such as pronouns, prepositions and determiners are not found in WordNet and for these, PoS was assigned by table lookup. We further included separate PoS tags for numbers and punctuation. 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 and thus remain similar only to themselves.

The WordNet Domains transformation was constructed analogously to the PoS transformation. All senses in all applicable parts-of-speech were determined for each word and the relevant domains were extracted for each sense. The union of these sets of domains was then taken to form the set of all possible applicable domains for each word. Thus, for example, the word *march* was assigned, among others, the domains *time period*, *music* and *transport* based on the noun senses *the third month* and *marching music* and the verb sense *march in a procession*, respectively. Based on this approach, 8004 of the 10000 most common tokens could be assigned at least one applicable domain. The remaining tokens, including closed-class words (such as prepositions, determiners, conjunctions, and pronouns), most proper nouns and punctuation, were not assigned any domain.

All the transformations have the diagonal shift parameter and hence their computational complexities are quite close to each other. The only difference is the number of features in the vectors the words are mapped to. There are only ten different part-of-speech and the number of position features in UW is 17 (i.e. $2 \cdot 8 + 1$) while there are more than 200 different domains associated with WD transformation.

The test results are presented in Table 2. The classification performance using the PoS transformation is clearly the best among the word transformations, which is not surprising since the PoS information is known to be useful in natural language disambiguation tasks (see e.g. Jurafsky and Martin 2000). Every word-transformation gives a statistically significant performance improvement over the baseline B.

### 5.3 Experiments with composite transformations

We tested the RLS classification performance with the following transformation combinations: The unsupervised word and unsupervised positional (UWUP), the part-of-speech and unsupervised positional (PoSUP), as well as the part-of-speech and variable width Gaussian (PoSVW). We used the unsupervised positional transformation as a part of the combinations because it has no extra parameters and the parameters of the combinations are therefore easy to select. The part-of-speech and the variable width Gaussian transformations are, on the other hand, the best performing ones among the positional and word transformations, respectively. We also wanted to test the performance of the fully unsupervised transformation combination. To further speed up the parameter selection phase of PoSVW, we took the best performing width parameter of VW from the positional transformation experiments and selected the other parameters in with the grid search. The test results of the combinations are presented in Table 3. Every combination improved the average performance compared to the classifiers with a single transformation and the improvements are statistically significant.

**Table 2** The RLS classification performances with different word transformations, namely, the no-transformations baseline (B), part-of-speech (PoS) transformation, WordNet Domains (WD) transformation, and unsupervised word (UW) transformation

| | B | PoS | WD | UW |
|---|---|---|---|---|
| I-me | 99.13 | **99.48** | 99.23 | 99.45 |
| accept-except | 99.80 | 99.81 | **99.82** | 99.79 |
| affect-effect | 98.64 | **98.68** | 98.64 | 98.57 |
| among-between | 94.28 | 94.30 | **94.66** | 93.64 |
| amount-number | 92.00 | 91.91 | **92.60** | 92.21 |
| begin-being | 98.97 | 99.05 | 98.91 | **99.06** |
| country-county | 93.14 | **94.11** | 93.30 | 93.20 |
| fewer-less | 78.13 | **80.73** | 79.59 | 80.59 |
| its-it's | 97.08 | **98.99** | 98.01 | 98.01 |
| lead-led | 98.06 | **98.18** | 98.10 | 98.08 |
| maybe-may be | 94.93 | **96.30** | 95.46 | 95.84 |
| passed-past | 98.85 | **99.01** | 98.91 | 98.85 |
| peace-piece | 96.79 | **97.69** | 96.29 | 96.64 |
| principal-principle | 95.59 | 95.90 | 95.66 | **95.91** |
| quiet-quite | 98.53 | **98.94** | 98.72 | 98.83 |
| raise-rise | 98.41 | 98.26 | **98.49** | 98.24 |
| than-then | 99.05 | 99.08 | 99.05 | **99.18** |
| weather-whether | 99.22 | 99.32 | **99.35** | 99.34 |
| your-you're | 97.02 | **98.63** | 97.75 | 97.36 |
| Average | 96.19 | 96.76 | 96.45 | 96.46 |

**Table 3** The RLS classification performances with different combined transformations, namely, the no-transformations baseline (B), unsupervised word and unsupervised positional (UWUP) transformation, part-of-speech and unsupervised positional (PoSUP) transformation, and part-of-speech and variable width Gaussian (PoSVW) transformation

| | B | UWUP | PoSUP | PoSVW |
|---|---|---|---|---|
| I-me | 99.13 | 99.46 | **99.52** | 99.51 |
| accept-except | 99.80 | 99.82 | 99.83 | **99.89** |
| affect-effect | 98.64 | **98.79** | 98.71 | 98.67 |
| among-between | 94.28 | 95.32 | 95.37 | **95.58** |
| amount-number | 92.00 | 92.76 | 92.61 | **93.64** |
| begin-being | 98.97 | 98.98 | 99.04 | **99.19** |
| country-county | 93.14 | 98.18 | 98.30 | **98.52** |
| fewer-less | 78.13 | 86.30 | **86.86** | 84.30 |
| its-it's | 97.08 | 98.59 | 98.95 | **99.10** |
| lead-led | 98.06 | 98.24 | **98.32** | 98.19 |
| maybe-may be | 94.93 | 96.75 | **97.15** | 96.97 |
| passed-past | 98.85 | 98.62 | 98.82 | **99.28** |
| peace-piece | 96.79 | **98.50** | 97.36 | 97.64 |
| principal-principle | 95.59 | 96.40 | 96.18 | **96.68** |
| quiet-quite | 98.53 | 99.06 | **99.24** | 99.17 |
| raise-rise | 98.41 | 98.68 | **99.01** | 98.95 |
| than-then | 99.05 | 99.05 | 99.00 | **99.33** |
| weather-whether | 99.22 | **99.69** | 99.66 | 99.53 |
| your-you're | 97.02 | 97.31 | 98.42 | **98.77** |
| Average | 96.19 | 97.39 | 97.49 | 97.52 |

In summary, we find that the best-performing composite transformation achieves on average a 60% reduction in AUC error when compared against the BoW representation (93.73% for BoW and 97.52% for PoSVW) and a 35% reduction in AUC error compared to the untransformed word-position matrix representation (96.19% for B), demonstrating that the proposed methods can provide considerably increased performance.

## 6 Related work

In this section, we first describe related word and positional transformations considered in earlier studies on kernel methods, and then discuss related frameworks for constructing kernels suitable for natural language processing tasks.

Word similarities have been incorporated in kernel functions through a number of techniques. For example, kernels that perform semantic smoothing for word vectors were proposed by Siolas and d'Alché-Buc (2000), latent semantic kernels were introduced by Cristianini et al. (2002), and semantic diffusion kernels by Kandola et al. (2003). The semantic smoothing, latent semantic indexing (Deerwester et al. 1990) and semantic diffusion techniques can be applied in our framework as word transformations, as described in Sect. 3.1.

We have considered positional transformations in a number of previous studies. The distance of the context words from the word to be disambiguated was used as the basis of a weighted BoW representation in (Ginter et al. 2004). This approach was then used to construct a support vector kernel (Pahikkala et al. 2005a). Further, we introduced a position-sensitive kernel which incorporates distance-based weighting and positional smoothing (Pahikkala et al. 2005b, 2005c). We have also shown that the positional smoothing can be applied together with word similarity transformations when using Bayesian classifiers (Pahikkala et al. 2006a). These studies included evaluations against the BoW and word-position feature representations at the disambiguation-type tasks of context-sensitive spelling error correction, word sense disambiguation, and gene versus protein name disambiguation, finding a statistically significant advantage to the proposed methods in each evaluation. The application of these transformations in the current framework is discussed in detail in Sect. 3.2. The framework introduced in this study generalizes over all these positional kernels, allowing their use simultaneously with word transformations as described in Sect. 4.

We now discuss other proposed frameworks relevant to natural language processing tasks. A number of studies on kernels over discrete structures have recently been proposed that fall into the general convolution kernel framework proposed by Haussler (1999). For example, kernels over strings (Lodhi et al. 2002), word sequences (Cancedda et al. 2003), trees (Collins and Duffy 2001), and graphs (Gärtner et al. 2003; Suzuki et al. 2003) have been introduced. Of these, string kernels and word sequence kernels are in particular applicable to disambiguation tasks similar to the one considered in this work in the sense that the sequence of the words is the primary ordering information considered by these kernels. Word similarity information can be applied together also with word sequence kernels, as suggested by Shawe-Taylor and Cristianini (2004). However, the way positional information is incorporated in these kernels differs substantially from the approach taken in the current framework, and, in particular, positional distance weighting has not been considered in any of these methods. Positional similarity information can be applied also with sequence kernels, as we have suggested in Tsivtsivadze et al. (2006). In addition to allowing computation of the similarity of structured data, one of the characteristic properties of the convolution framework is that the feature spaces induced by the kernel functions are typically very high-dimensional, thus making explicit computation of the feature mapping and the inner product in the feature space infeasible. These kernel functions thus serve to facilitate the computation of similarity metrics where the features could not be computed explicitly.

Cumby and Roth (2002, 2003a) have introduced a general framework for extracting features from structured data expressed in terms of a feature description language. A family of kernel functions corresponding to the inner product of feature vectors generated according to the given feature descriptions was then introduced in Cumby and Roth (2003b). They show that the use of a restricted feature space defined in a "syntax-driven" fashion can be beneficial as irrelevant features can be excluded. It is also possible to explicitly represent the feature vector defined by the feature description language, allowing the description to be used with other, non-kernelized, learning algorithms. This explicit definition also facilitates the use of feature description languages together with our framework, as discussed further in Sect. 7. Kernel construction through syntax-driven feature definitions has also been discussed by Gärtner et al. (2004).

## 7 Discussion

In the description of the framework in the previous sections and in the experiments described in Sect. 5, we have compared linear transformations of word-position matrices to the simple BoW and word-position feature representations. In the following, we discuss the applicability of the framework in a broader setting.

In many natural language processing tasks, linear combinations of word-position features may not be sufficiently expressive. In particular, features derived in a nonlinear fashion from the text, such as parts-of-speech generated by tagging algorithms, and nonlinear combinations of features such as word bigrams (two words occurring consecutively in the text) are commonly used by state-of-the-art classification systems (see e.g. Tjong and De Meulder 2003; Carreras and Màrques 2004, 2005).

In applications where, for example, a part-of-speech (PoS) tagger is used to provide the PoS information of the context words, this information can be incorporated into the matrix representation by adding one row per each possible PoS tag. Note that this approach to integrating PoS information differs from that introduced in Sect. 3.1 in that PoS taggers cannot be expressed as a matrix of a linear transformation. Other similar classes of features whose position in the context can be defined, such as the inflectional categories of words (e.g. verb tense, noun plurality, etc.) as well as combinations of, for example, words with PoS tags (e.g. *water/NN*) could be similarly incorporated into the matrix representation.

When applied together with the BoW representation, bigram features are typically separately included as a "bag of bigrams" so that the linear kernel function corresponds to a sum of a BoW kernel and a bag-of-bigrams kernel. Thus, as the (transformed) word-position matrix generalizes over the BoW "component", it is natural to combine its use with bigram features as a kernel that is the sum of the kernel presented in this study and the bag-of-bigrams kernel. Other nonlinear features can also be applied together with the presented framework similarly to bigrams. As discussed in Sect. 4, the computational complexity of our kernel is very low due to its linearity. In cases where nonlinear features can be represented explicitly, they can be applied together with our kernel without increasing the complexity of the prediction. For example, the feature description language framework of Cumby and Roth (2002, 2003a) (discussed also in Sect. 6) offers a general way to define conjunctive features such as bigrams that can be explicitly represented, allowing also the decision hyperplane to be represented explicitly in the defined feature space. The framework of Cumby and Roth can thus be applied together with our framework. Such a combination could have the benefits of both the linear feature transformations presented in this study and a general set of nonlinear features. Closer integration of these two approaches could allow further benefits through, for example, the application of feature transformations on explicitly represented nonlinear features.

The combination of the framework with more complex nonlinear features would considerably extend its applicability from the simple disambiguation tasks considered in this study. Such combination and experiments establishing the performance of the methods in the large number of disambiguation tasks remains as future work.

## 8 Conclusion

In this paper, we propose a framework that is based on a word-position matrix representation of text, linear feature transformations of the word-position matrices, and kernel functions constructed from the transformations. We consider several ways to construct the word and positional transformations. The results of the experiments show that the classification performance of kernel based classifiers in the model problem could be improved with each of transformations separately, and the composite transformations further improved the performance. The best-performing composite transformation was found to reduce the AUC error by 60% compared to BoW and by 35% compared to the basic word-position feature representation.

We also present efficient algorithms for training a kernel-based learning machine that uses the proposed framework. More importantly, we also introduce a method for the prediction of new examples with a trained linear support vector machine kind of learners that use the framework, and show that prediction can be performed as efficiently as in the BoW framework.

The use of the linear feature transformations to construct kernel functions is a promising approach in general, because it provides an elegant and efficient way to incorporate external information into the classifier. The external information we incorporate into the classifiers via the linear feature transformations are just examples of a prior knowledge that is relevant to the classification tasks considered. It is, of course, possible to develop more sophisticated transformations and, moreover, other NLP tasks may prefer completely different kind of information. Also, the unsupervised word and positional transformations for the word-position matrices introduced in this paper are simple and surprisingly effective ways to incorporate extra information to the classifier.

**Acknowledgements** This work has been supported by Tekes, the Finnish Funding Agency for Technology and Innovation. We would like to thank CSC, the Finnish IT centre for science, for providing us extensive computing resources and Aleksandr Mylläri for his helpful comments on this manuscript.

## References

Bentivogli, L., Forner, P., Magnini, B., & Pianta, E. (2004). Revising wordnet domains hierarchy: Semantics, coverage, and balancing. In G. Sérasset, S. Armstrong, C. Boitet, A. Popescu-Belis, & D. Tufis (Eds.), *COLING 2004 workshop on multilingual linguistic resources* (pp. 101–108), Geneva, Switzerland.

Cancedda, N., Gaussier, E., Goutte, C., & Renders, J.-M. (2003). Word-sequence kernels. *Journal of Machine Learning Research*, *3*, 1059–1082.

Carreras, X., & Màrques, L. (2004). Introduction to the conll-2004 shared task: semantic role labeling. In *Proceedings of CoNLL-2004* (pp. 89–97). Boston: Association for Computational Linguistics.

Carreras, X., & Màrquez, L. (2005). Introduction to the CoNLL-2005 shared task: semantic role labeling. In *Proceedings of the ninth conference on computational natural language learning (CoNLL-2005)* (pp. 152–164). Ann Arbor: Association for Computational Linguistics.

Collins, M., & Duffy, N. (2001). Convolution kernels for natural language.

Cortes, C., & Mohri, M. (2004). Auc optimization vs. error rate minimization. In S. Thrun, L. Saul, & B. Schölkopf (Eds.), *Advances in neural information processing systems 16*. Cambridge: MIT Press.

Cristianini, N., Shawe-Taylor, J., & Lodhi, H. (2002). Latent semantic kernels. *Journal of Intelligent Information Systems*, *18*(2–3), 127–152.

Cumby, C. M., & Roth, D. (2002). Learning with feature description logics. In *Proceedings of the 12th international conference on inductive logic programming*.

Cumby, C. M., & Roth, D. (2003a). Feature extraction languages for propositionalized relational learning. In *Proceedings of the IJCAI'03 workshop on learning statistical models from relational data*.

Cumby, C. M., & Roth, D. (2003b). On kernel methods for relational learning. In T. Fawcett & N. Mishra (Eds.), *Proceedings of the twentieth international conference on machine learning* (pp. 107–114). Menlo Park: AAAI Press.

Deerwester, S. C., Dumais, S. T., Landauer, T. K., Furnas, G. W., & Harshman, R. A. (1990). Indexing by latent semantic analysis. *Journal of the American Society of Information Science*, *41*(6), 391–407.

Fawcett, T. (2003). *Roc graphs: notes and practical considerations for data mining researchers* (Technical Report HPL-2003-4). HP Labs, Palo Alto, CA.

Gärtner, T., Flach, P. A., & Wrobel, S. (2003). On graph kernels: hardness results and efficient alternatives. In *COLT* (pp. 129–143).

Gärtner, T., Lloyd, J. W., & Flach, P. A. (2004). Kernels and distances for structured data. *Machine Learning*, *57*(3), 205–232.

Ginter, F., Boberg, J., Järvinen, J., & Salakoski, T. (2004). New techniques for disambiguation in natural language and their application to biological text. *Journal of Machine Learning Research*, *5*, 605–621.

Gliozzo, A., Giuliano, C., & Strapparava, C. (2005). Domain kernels for word sense disambiguation. In *Proceedings of the 43rd annual meeting of the association for computational linguistics (ACL'05)* (pp. 403–410). Ann Arbor: Association for Computational Linguistics.

Golding, A. R., & Roth, D. (1999). A winnow-based approach to context-sensitive spelling correction. *Machine Learning*, *34*, 107–130.

Haussler, D. (1999). *Convolution kernels on discrete structures* (Technical Report UCS-CRL-99-10). University of California at Santa Cruz.

Joachims, T. (1998). Text categorization with support vector machines: learning with many relevant features. In C. Nédellec & C. Rouveirol (Eds.), *Lecture notes in computer science: Vol. 1398. Proceedings of the tenth European conference on machine learning* (pp. 137–142), Chemnitz, Germany, 1998. Heidelberg: Springer.

Joachims, T. (2002). *Kluwer international series in engineering and computer science: Vol. 668. Learning to classify text using support vector machines: methods, theory and algorithms*. Norwell: Kluwer Academic.

Jurafsky, D., & Martin, J. H. (2000). *Speech and language processing: an introduction to natural language processing, computational linguistics, and speech recognition*. Upper Saddle River: Prentice Hall PTR.

Kandola, J., Shawe-Taylor, J., & Cristianini, N. (2003). Learning semantic similarity. In S. T. Becker & K. Obermayer (Eds.), *Advances in neural information processing systems 15* (pp. 657–664). Cambridge: MIT Press.

Leopold, E., & Kindermann, J. (2002). Text categorization with support vector machines. How to represent texts in input space? *Machine Learning*, *46*(1–3), 423–444.

Ling, C. X., Huang, J., & Zhang, H. (2003). Auc: a statistically consistent and more discriminating measure than accuracy. In G. Gottlob & T. Walsh (Eds.), *Proceedings of the eighteenth international joint conference on artificial intelligence* (pp. 519–526). San Mateo: Morgan Kaufmann.

Lodhi, H., Saunders, C., Shawe-Taylor, J., Cristianini, N., & Watkins, C. (2002). Text classification using string kernels. *Journal of Machine Learning Research*, *2*, 419–444.

Magnini, B., & Cavaglià, G. (2000). Integrating subject field codes into WordNet. In *Second international conference on language resources and evaluation (LREC-2000)* (pp. 1413–1418). Athens: European Language Resources Association.

Magnus, J. R. (1988). *Linear structures*. London: Griffin.

Meyer, C. D. (2000). *Matrix analysis and applied linear algebra*. Philadelphia: Society for Industrial and Applied Mathematics.

Pahikkala, T., Ginter, F., Boberg, J., Järvinen, J., & Salakoski, T. (2005a). Contextual weighting for support vector machines in literature mining: an application to gene versus protein name disambiguation. *BMC Bioinformatics*, *6*(1), 157.

Pahikkala, T., Pyysalo, S., Boberg, J., Mylläri, A., & Salakoski, T. (2005b). Improving the performance of Bayesian and support vector classifiers in word sense disambiguation using positional information. In T. Honkela, V. Könönen, M. Pöllä, & O. Simula (Eds.), *Proceedings of the international and interdisciplinary conference on adaptive knowledge representation and reasoning* (pp. 90–97). Espoo: Helsinki University of Technology.

Pahikkala, T., Pyysalo, S., Ginter, F., Boberg, J., Järvinen, J., & Salakoski, T. (2005c). Kernels incorporating word positional information in natural language disambiguation tasks. In I. Russell & Z. Markov (Eds.), *Proceedings of the eighteenth international Florida artificial intelligence research society conference* (pp. 442–447), Clearwater Beach, FL. Menlo Park: AAAI Press.

Pahikkala, T., Boberg, J., Mylläri, A., & Salakoski, T. (2006a). Incorporating external information in Bayesian classifiers via linear feature transformations. In T. Salakoski, F. Ginter, S. Pyysalo, & T. Pahikkala (Eds.), *Lecture notes in computer science: Vol. 4139. Proceedings of the 5th international conference on NLP (FinTAL 2006)* (pp. 399–410). Heidelberg: Springer.

Pahikkala, T., Boberg, J., & Salakoski, T. (2006b). Fast n-fold cross-validation for regularized least-squares. In T. Honkela, T. Raiko, J. Kortela, & H. Valpola (Eds.), *Proceedings of the ninth Scandinavian conference on artificial intelligence (SCAI 2006)* (pp. 83–90). Espoo: Otamedia Oy.

Provost, F. J., Fawcett, T., & Kohavi, R. (1998). The case against accuracy estimation for comparing induction algorithms. In *ICML '98: proceedings of the fifteenth international conference on machine learning* (pp. 445–453). San Francisco: Morgan Kaufmann.

Rifkin, R. (2002). *Everything old is new again: a fresh look at historical approaches in machine learning*. PhD thesis, MIT.

Rifkin, R., Yeo, G., & Poggio, T. (2003). Regularized least-squares classification. In J. Suykens, G. Horvath, S. Basu, C. Micchelli, & J. Vandewalle (Eds.), *NATO science series III: computer and system sciences: Vol. 190. Advances in learning theory: methods, model and applications* (pp. 131–154). Amsterdam: IOS Press.

Rose, T. G., Stevenson, M., & Whitehead, M. (2002). The Reuters corpus volume 1: from yesterday's news to tomorrow's language resources. In M. G. Rodriguez & C. P. S. Araujo (Eds.), *Proceedings of the third international conference on language resources and evaluation*. Paris: ELRA.

Schölkopf, B., & Smola, A. J. (2002). *Learning with kernels*. Cambridge: MIT Press.

Schölkopf, B., Simard, P., Smola, A., & Vapnik, V. (1998). Prior knowledge in support vector kernels. In M. I. Jordan, M. J. Kearns, & S. A. Solla (Eds.), *Advances in neural information processing systems 10* (pp. 640–646). Cambridge: MIT Press.

Shawe-Taylor, J., & Cristianini, N. (2004). *Kernel methods for pattern analysis*. Cambridge: Cambridge University Press.

Silverman, B. W. (1986). *Density estimation for statistics and data analysis*. London: Chapman & Hall.

Siolas, G., & d'Alché-Buc, F. (2000). Support vector machines based on a semantic kernel for text categorization. In S.-I. Amari, C. L. Giles, M. Gori, & V. Piuri (Eds.), *Proceedings of the IEEE-Inns-Enns international joint conference on neural networks* (pp. 205–209), Como, Italy. Washington: IEEE Computer Society.

Suzuki, J., Hirao, T., Sasaki, Y., & Maeda, E. (2003). *Hierarchical directed acyclic graph kernel: methods for structured natural language data*.

Tjong, K. S. E. F., & De Meulder, F. (2003). Introduction to the conll-2003 shared task: language-independent named entity recognition. In W. Daelemans & M. Osborne (Eds.), *Proceedings of CoNLL-2003* (pp. 142–147). Edmonton: Association for Computational Linguistics.

Tsivtsivadze, E., Pahikkala, T., Boberg, J., & Salakoski, T. (2006). Locality-convolution kernel and its application to dependency parse ranking. In *The 19th international conference on industrial, engineering & other applications of applied intelligent systems*. Forthcoming.

Vapnik, V. (1998). *Statistical learning theory*. New York: Wiley.

Vishwanathan, S., Smola, A. J., & Vidal, R. (2006, to appear). Binet-Cauchy kernels on dynamical systems and its application to the analysis of dynamic scenes. *International Journal of Computer Vision*.

Wilcoxon, F. (1945). Individual comparisons by ranking methods. *Biometrics*, *1*, 80–83.

Wong, S. K. M., Ziarko, W., & Wong, P. C. N. (1985). Generalized vector space model in information retrieval. In *ACM SIGIR international conference on research and development in information retrieval* (pp. 18–25).

Yarowsky, D. (1993). One sense per collocation. In *Proceedings, ARPA human language technology workshop*, Princeton.

Yarowsky, D. (1995). Unsupervised word sense disambiguation rivaling supervised methods. In *Meeting of the association for computational linguistics* (pp. 189–196).