A Weighted Polynomial Information Gain Kernel for Resolving Prepositional Phrase Attachment Ambiguities with Support Vector Machines

Bram Vanschoenwinkel* **Bernard Manderick**
Vrije Universiteit Brussel, Department of Computer Science, Computational Modeling Lab
Pleinlaan 2, 1050 Brussel, Belgium
{bvschoen@ – bernard@arti}.vub.ac.be

Abstract

We introduce a new kernel for Support Vector Machine learning in a natural language setting. As a case study to incorporate domain knowledge into a kernel, we consider the problem of resolving Prepositional Phrase attachment ambiguities. The new kernel is derived from a distance function that proved to be succesful in memory-based learning. We start with the Simple Overlap Metric from which we derive a Simple Overlap Kernel and extend it with Information Gain Weighting. Finally, we combine it with a polynomial kernel to increase the dimensionality of the feature space. The closure properties of kernels guarantee that the result is again a kernel. This kernel achieves high classification accuracy and is efficient in both time and space usage. We compare our results with those obtained by memory-based and other learning methods. They make clear that the proposed kernel achieves a higher classification accuracy.

1 Introduction

An important issue in natural language analysis is the resolution of structural ambiguity. A sentence is said to be structurally ambiguous when it can be assigned to more than one syntactic structure [Zavrel *et al.*, 1997]. In *Prepositional Phrase (PP)* attachment one wants to disambiguate between cases where it is uncertain whether the PP attaches to the verb or to the noun.

Example 1 *Consider the following two sentences:*

1. *I bought the shirt with pockets.*
2. *I washed the shirt with soap.*

In sentence 1, **with** *modifies the noun* **shirt** *because* **with pockets** *(PP) describes the* **shirt**. *In sentence 2 however,* **with** *modifies the verb* **washed** *because* **with soap** *(PP) describes how the shirt is* **washed** *[Ratnaparkhi, 1998].*

This type of attachment ambiguity is easy for people to resolve because they can use their world knowledge [Stetina and Nagao, 1997]. A computer program usually cannot rely on that kind of knowledge.

This problem has already been tackled using memory-based learning like for example k-nearest neighbours. Here, the training examples are first stored in memory and classification of a new example is done based on the closest example stored in memory. Therefore, one needs a function that expresses the distance or similarity between examples. There already exist several dedicated distance functions to solve all kind of natural language problems using memory-based learning [Veenstra *et al.*, 2000; Zavrel *et al.*, 1997; Daelemans *et al.*, 2002].

We will use a Support Vector Machine (SVM) to tackle the problem of PP attachment disambiguation. Central to SVM learning is the kernel function $K : X \times X \rightarrow \Re$ where X contains the examples and the kernel K calculates an inner product in a second space, the feature space F. This product expresses how similar examples are.

Our goal is to combine the power of SVMs with the distance functions that are well-suited for the probem for which they were designed. Deriving a distance from a kernel is straightforward, see Section 2.1. However, deriving a kernel from a distance is not trivial since kernels must satisfy some extra conditions, i.e. being a kernel is a much stronger condition than being a distance. In this paper we will describe a method that shows how such dedicated distance functions can be used as a basis for designing kernels that sequentially can be used in SVM learning.

We use the PP attachment problem as a case study to illustrate our approach. As a starting point we take the *Overlap Metric* that has been succesfully used in memory-based learning for the same problem [Zavrel *et al.*, 1997].

Section 2 will give a short overview of the theory of SVMs together with some theorems and definitions that are needed in Section 4. Based on [Zavrel *et al.*, 1997], section 3 gives an overview of metrics developed for memory-based learning applied to the PP attachment problem. In Section 4 the new kernels will be introduced. Finally Sections 5 and 6 give some experimental results and a conclusion of this work.

2 Support Vector Machines

For simplicity, in our explanation we will consider the case of binary classification only, i.e. we consider an input space X with input vectors $\mathbf{x}$ and a target space $D = \{1, -1\}$. The

*Author funded by a doctoral grant of the institute for advancement of scientific technological research in Flanders (IWT).

goal of the SVM is to assign every $\mathbf{x}$ to one of two classes $D = \{1, -1\}$. The decision boundary that separates the input vectors belonging to different classes is usually an arbitrary $n-1$-dimensional manifold if the input space X is n-dimensional.

One of the basic ideas behind SVMs is to have a mapping Φ from the original input space X into a high-dimensional *feature space* F that is a Hilbert space, i.e. a complete vector space provided with an inner product. Separation of the transformed feature vectors $\Phi(\mathbf{x})$ in F is done linearly, i.e. by a hyperplane. Cover's theorem states that any consistent training set can be made linear separable provided the dimension of F is high enough.

However, transforming the training set $TS = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_i, ..., \mathbf{x}_N\} \subset X$ into such a higher-dimensional space incurs both computational and learning-theoretic problems. The high dimensionality of F makes it very expensive both in terms of memory and time to represent the feature vectors $\Phi(\mathbf{x}_i)$ corresponding to the training vectors $\mathbf{x}_i$. Moreover, separating the data in this way exposes the learning system to the risk of *overfitting* the data if the separating hyperplane is not chosen properly.

SVMs sidestep both difficulties [Vapnik, 1998]. First, overfitting is avoided by choosing the unique *maximum margin hyperplane* among all possible hyperplanes that can separate the data in F. This hyperplane maximizes the distance to the closest data points.

Second, the maximum margin hyperplane in F can be represented entirely in terms of training vectors $\mathbf{x}_i \in TS$ and a kernel K.

Definition: A *kernel* is a function $K : X \times X \to \Re$ so that for all $\mathbf{x}$ and $\mathbf{y}$ in X, $K(\mathbf{x}, \mathbf{y}) = < \Phi(\mathbf{x}), \Phi(\mathbf{y}) >$ where Φ is a (non-linear) mapping from the input space X into the Hilbert space F provided with the inner product $< ., . >$ [Cristianini and Shawe-Taylor, 2000].

To be more precise, once we have chosen a kernel K we can represent the maximal margin hyperplane (or decision boundary) by a *linear equation* in $\mathbf{x}$

$$\sum_{\mathbf{x}_i \in TS} \alpha_i d_i K(\mathbf{x}_i, \mathbf{x}) + b = 0 \qquad (1)$$

where the training vectors $\mathbf{x}_i$ and their class labels d_i are given. The decision boundary is determined completely once we know the α_i and b. They are obtained by optimizing a convex quadratic objective function with linear constraints. Moreover, most of the α_i prove to be zero. By definition the vectors $\mathbf{x}_i$ corresponding with non-zero α_i are called the *support vectors* SV and this set consists of those data points that lie closest to the hyperplane and thus are the most difficult to classify.

In order to classify a new point $\mathbf{x}_{new}$, one determines the sign of

$$\sum_{\mathbf{x}_i \in SV} \alpha_i d_i K(\mathbf{x}_i, \mathbf{x}_{new}) + b$$

If this sign is positive $\mathbf{x}_{new}$ belongs to class 1, if negative to class -1, if zero $\mathbf{x}_{new}$ lies on the decision boundary. Note that we have now restricted the summation to the set SV of support vectors because the other α_i are zero anyway.

To conclude, SVM can sidestep the above two difficulties because neither the feature space F nor the map Φ from the input space X into F are explicitly defined, they are replaced by the kernel K that operates on vectors of the input space X.

2.1 Some Properties of Kernels

The selection of an appropriate kernel K is the most important design decision in SVMs since it implicitly defines the feature space F and the map Φ. A SVM will work correctly even if we don't know the exact form of the features that are used in F. Moreover, the kernel expresses prior knowledge about the patterns being modelled, encoded as a similarity measure between two vectors [Brown *et al.*, 1999].

But not all maps over $X \times X$ are kernels. Since a kernel K is related to an inner product, cfr. the definition above, it has to satisfy some conditions that arise naturally from the definition of an inner product and are given by Mercer's theorem: the map must be continuous and positive definite [Vapnik, 1998].

In this paper we will use following methods to construct kernels [Cristianini and Shawe-Taylor, 2000]:

M1 Making kernels from kernels: Based on the fact that kernels satisfy a number of closure properties. In this case, the Mercer conditions follow naturally from the closure properties of kernels.

M2 Making kernels from features: Start from the features of the input vectors and obtain a kernel by working out their inner product. A feature is a component of the input vector. In this case, the Mercer conditions follow naturally from the definition of an inner product.

The set of kernels is closed under certain operations, this allows us to make new kernels from existing ones (see M1). We mention only the closure property used in the paper:

Closure properties of kernels: Let $\Phi : X \to \Re^n$ be a map and K_1 be a kernel over $\Re^n \times \Re^n$ then the following function is a kernel $K(\mathbf{x}, \mathbf{y}) = K_1(\Phi(\mathbf{x}), \Phi(\mathbf{y}))$.

Another interesting property is that we can derive a distance d from a kernel K as follows:

Distance derived from a kernel: Let K be a kernel over $X \times X$, then d defined as $d(\mathbf{x}, \mathbf{y}) = \sqrt{K(\mathbf{x}, \mathbf{x}) - 2K(\mathbf{x}, \mathbf{y}) + K(\mathbf{y}, \mathbf{y})}$ is a distance on X and consequentially (X, d) is a metric space.

3 Metrics for Memory-based Learning

In this section, we will focus on the distance functions [Zavrel *et al.*, 1997; Cost and Salzberg, 1993] used for memory-based learning with symbolic values. First, we will have a look at the *Simple Overlap Metric* (SOM) and next we will discuss *Information Gain Weighting* (IGW). Memory-based learning is a class of machine learning techniques where training instances are stored in memory first and classification of new instances later on is based on the distance (or similarity) between the new instance and the closest training instances that have already been stored in memory. A well-known example of memory-based learning is *k-nearest neighbours classification*. We will not go into further detail, the literature offers

many books and articles that provide a comprehensive introduction to memory-based learning, e.g. [Mitchell, 1997]. For our purpose, the important thing to remember is that we will be working with symbolic values like strings over an alphabet of characters. Instances are n-dimensional vectors $\mathbf{a}$ and $\mathbf{b}$ in some universe Ω^n. The components of the vectors in Ω^n are strings, i.e. $a_i, b_i \in \Sigma^*$, with Σ^* the set of all strings over some alphabet Σ. Notice that for $n = 1$, Ω^n becomes Ω and the elements of Ω are strings in Σ^*. In order for the inner products defined below to be meaningful, the set of strings Ω has to be finite. As a result the set of vectors Ω^n is also finite as required.

3.1 Simple Overlap Metric

The most basic metric for vectors with symbolic values is the *Simple Overlap Metric*(SOM) [Zavrel *et al.*, 1997]:

$$d_{SOM} : \Omega^n \times \Omega^n \rightarrow \Re^+ : d_{SOM}(\mathbf{a}, \mathbf{b}) = \sum_{i=1}^{n} \delta(a_i, b_i) \quad (2)$$

with

$$\delta(a_i, b_i) = 1 - \delta_{a_i}(b_i) \quad (3)$$

and

$$\delta : \Omega \times \Omega \rightarrow \{1, 0\} : \delta_{a_i}(b_i) = 1 \; if \; a_i = b_i, \; else \; 0 \quad (4)$$

Here $d_{SOM}(\mathbf{a}, \mathbf{b})$ is the distance between the vectors $\mathbf{a}$ and $\mathbf{b}$, represented by n features and δ is the distance per feature. The k-nearest neighbour algorithm equipped with this metric is called IB1 [Aha *et al.*, 1991]. The IB1 algorithm simply counts the number of (mis)matching feature values in both vectors. This is a reasonable choice if we have no information about the importance of the different features. But if we do have information about feature importance then we can add linguistic bias to weight the different features.

3.2 Information Gain Weighting

Information Gain Weighting (IGW) measures for every feature i separately how much information it contributes to our knowledge of the correct class label. The Information Gain (IG) of a feature i is measured by calculating the entropy between the cases with and without knowledge of the value of that feature:

$$w_i = H(C) - \frac{\sum_{v \in V_i} P(v) \times H(C \,|v\,)}{si(i)} \quad (5)$$

$$si(i) = -\sum_{v \in V_i} P(v) log_2 P(v)$$

Where C is the set of class labels, V_i is the set of values for feature i, $H(C) = -\sum_{c \in C} P(c) log_2 P(c)$ is the entropy of the class labels and $si(i)$ is the split info. The resulting weights w_i can be used to extend Equation 2 with weights (see Equation 6). The k-nearest neigbour algorithm equipped with this metric is called IB1-IG [Zavrel *et al.*, 1997] and the corresponding distance called d_{IG}:

$$d_{IG} : \Omega^n \times \Omega^n \rightarrow \Re^+ : d_{IG}(\mathbf{a}, \mathbf{b}) = \sum_{i=1}^{n} w_i \delta(a_i, b_i) \quad (6)$$

with δ like in Eqs. 3 and 4.

4 A Kernel for Natural Language settings

In this section, we will describe the kernel that we have constructed from the IGW metric d_{IG} from the previous section. A kernel will be a function $K : \Omega^n \times \Omega^n \rightarrow \Re$ and the mapping to the feature space F will be of the form $\Phi : \Omega^n \rightarrow \Re^{\Omega^n}$. We will begin with the simple, one-dimensional, unweighted case (Section 4.1). Next, we will extend it to n dimensions (Section 4.2) and add IGW (Section 4.3) and finally in Section 4.4 the Polynomial Information Gain (PIG) kernel will be introduced.

4.1 A Simple Overlap Kernel (SOK) in one dimension

If we only want to consider vectors in a one-dimensional space then we have to take a closer look at Equations 3 and 4. To derive a kernel from this distance function we will use M2 from Section 2.1. Consider the inner product space $\Re^\Omega$ of functions $f : \Omega \rightarrow \Re$, the inner product of two functions f and g is defined as:

$$< f, g >= \sum_{x \in \Omega} f(x).g(x) \quad (7)$$

Next, let $\phi_{SOK} : \Omega \rightarrow \Re^\Omega$ with

$$\phi_{SOK}(a) = \frac{1}{\sqrt{2}} \delta_a \quad (8)$$

be a feature mapping to the feature space F. The function δ_a is defined as $\delta_a(x) = 1$ when $x = a$ and 0 otherwise (see Equation 4). Starting from the features and the above defined feature mapping we will now work out the inner product to come to the corresponding kernel $k_{SOK} : \Omega \times \Omega \rightarrow \Re$:

$$\begin{aligned} k_{SOK}(a, b) &= < \phi_{SOK}(a), \phi_{SOK}(b) > \\ &= < \tfrac{1}{\sqrt{2}} \delta_a, \tfrac{1}{\sqrt{2}} \delta_b > \\ &= \tfrac{1}{2} \textstyle\sum_{x \in \Omega} \delta_a(x).\delta_b(x) \\ &= \tfrac{1}{2} \delta_a(b) \end{aligned}$$

We don't have to proof that the kernel k_{SOK} satisfies the Mercer conditions because this follows naturally from the definition of the inner product. We started from the features, defined a mapping ϕ_{SOK} on the features and we worked out the inner product between them, see M2 from Section 2.1. However, to show that the kernel really corresponds to the distance function given in Equations 3 and 4 we have to verify the distance formula for kernels given in Section 2.1:

$$\begin{aligned} \delta(a, b) &= \sqrt{k_{SOK}(a, a) - 2k_{SOK}(a, b) + k_{SOK}(b, b)} \\ &= \sqrt{\tfrac{1}{2}(\delta_a(a) - 2\delta_a(b) + \delta_b(b))} \\ &= \sqrt{\tfrac{1}{2}(2 - 2\delta_a(b))} \\ &= 1 - \delta_a(b) \end{aligned}$$

The last step is justified by the fact that $1 - \delta_a(b)$ is either 0 or 1 [Ramon, 2002].

4.2 Extending the SOK to n Dimensions

In this section we consider vectors $\mathbf{a}, \mathbf{b} \in \Omega^n$, the distance between these vectors is measured by the distance d_{SOM} from Equation 2. Now consider the inner product space $\Re^{\Omega^n}$ (the space of functions $f : \Omega^n \rightarrow \Re$) comparable to the one in Equation 7, with the following inner product:

$$< f, g >= \sum_{\mathbf{x} \in \Omega^n} f(\mathbf{x}).g(\mathbf{x}) \tag{9}$$

We begin by defining a mapping $\Phi_{SOK} : \Omega^n \rightarrow \Re^{\Omega^n}$:

$$\Phi_{SOK}(\mathbf{a}) = \frac{1}{\sqrt{2}}\delta_{\mathbf{a}}$$

We will again use M2 from Section 2.1 to derive a kernel for d_{SOM}. We work out the inner product between the feature vectors $\Phi_{SOK}(\mathbf{a})$ and $\Phi_{SOK}(\mathbf{b})$ to come to the corresponding kernel $K_{SOK} : \Omega^n \times \Omega^n \rightarrow \Re$:

$$\begin{aligned} K_{SOK}(\mathbf{a}, \mathbf{b}) &= < \Phi_{SOK}(\mathbf{a}), \Phi_{SOK}(\mathbf{b}) > \\ &= \tfrac{1}{2} \textstyle\sum_{\mathbf{x} \in \Omega^n} \delta_{\mathbf{a}}(\mathbf{x}).\delta_{\mathbf{b}}(\mathbf{x}) \\ &= \tfrac{1}{2}\delta_{\mathbf{a}}(\mathbf{b}) \\ &= \tfrac{1}{2} \textstyle\sum_{i=1}^{n} \delta_{a_i}(b_i) \end{aligned}$$

Here the function $\delta_{\mathbf{a}}$ is defined as $\delta_{\mathbf{a}}(\mathbf{x}) = \sum_{i=1}^{n} \delta_{a_i}(x_i)$ with $\delta_{a_i}(x_i)$ like in Equations 4 and 8. Notice that this corresponds to the inner product between the vectors $\tilde{\mathbf{a}} = (\phi_{SOK}(a_1), ..., \phi_{SOK}(a_n))$ and $\tilde{\mathbf{b}} = (\phi_{SOK}(b_1), ..., \phi_{SOK}(b_n))$. So, alternatively we can write $K_{SOK} : \Omega^n \times \Omega^n \rightarrow \Re$:

$$K_{SOK}(\mathbf{a}, \mathbf{b}) = \sum_{i=1}^{n} k_{SOK}(a_i, b_i) \tag{10}$$

We don't have to proof that the kernel K_{SOK} is a valid kernel as this follows naturally from the definition of the inner product. However, we again have to show that it is a kernel that corresponds to the distance function d_{SOM} from Equation 2. In the following we will show that the kernel K_{SOK} in fact corresponds to the square root of d_{SOM}, we will call this distance function d_r:

$$\begin{aligned} d_r(\mathbf{a}, \mathbf{b}) &= \sqrt{K_{SOK}(\mathbf{a}, \mathbf{a}) - 2K_{SOK}(\mathbf{a}, \mathbf{b}) + K_{SOK}(\mathbf{b}, \mathbf{b})} \\ &= \sqrt{(1 - \delta_{a_1}(b_1)) + ... + (1 - \delta_{a_n}(b_n))} \\ &= \sqrt{\textstyle\sum_{i=1}^{n} \delta(a_i, b_i)} \\ &= \sqrt{d_{SOM}(\mathbf{a}, \mathbf{b})} \end{aligned}$$

However, this does not impose any problems for the kernel we are aiming to develop.

4.3 Adding Information Gain to the SOK

Next, we will introduce IGW into the kernel. We will construct a kernel based on the distance function d_{IG} from Equation 6. We have to define new mappings ϕ and Φ to cope with the weights:

$$\Phi_{IG} : \Omega^n \rightarrow \Re^{\Omega^n} : \Phi_{IG}(\mathbf{a}) = \frac{\mathbf{w}}{\sqrt{2}}\delta_{\mathbf{a}} \tag{11}$$

$$\phi_{IG} : \Omega \rightarrow \Re^{\Omega} : \phi_{IG}(a_i) = \frac{w_i}{\sqrt{2}}\delta_{a_i} \tag{12}$$

We will start again by working out the inner product for the 1-dimensional case $k_{IG} : \Omega \times \Omega \rightarrow \Re$:

$$\begin{aligned} k_{IG}(a_i, b_i) &= < \phi_{IG}(a_i), \phi_{IG}(b_i) > \\ &= < \tfrac{w_i}{\sqrt{2}}\delta_{a_i}, \tfrac{w_i}{\sqrt{2}}\delta_{b_i} > \\ &= \tfrac{w_i^2}{2} \textstyle\sum_{x \in \Omega} \delta_{a_i}(x)\delta_{b_i}(x) \\ &= \tfrac{w_i^2}{2}\delta_{a_i}(b_i) \end{aligned}$$

The function k_{IG} is guaranteed to be a valid kernel, this follows naturally from the definition of the inner product (see M2 from Section 2.1). In the same way as for K_{SOK} we can show that K_{IG} corresponds to the inner product between the vectors $\tilde{\mathbf{a}} = (\phi_{IG}(a_1), ..., \phi_{IG}(a_n))$ and $\tilde{\mathbf{b}} = (\phi_{IG}(b_1), ..., \phi_{IG}(b_n))$, yielding the following result for the kernel $K_{IG} : \Omega^n \times \Omega^n \rightarrow \Re$:

$$K_{IG}(\mathbf{a}, \mathbf{b}) = \sum_{i=1}^{n} k_{IG}(a_i, b_i)$$

Also, in the same way as we did before we can show for k_{IG} that it corresponds to a distance function $\delta_{IG} : \Omega \times \Omega \rightarrow \Re$ with $\delta_{IG}(a_i, b_i) = w_i\delta(a_i, b_i)$ with δ as defined in Equation 3. Furthermore, in the same way as we did for K_{SOK} in Section 4.2 we can show that the kernel K_{IG} corresponds to the square root of the distance function d_{IG} from Equation 6.

4.4 A Polynomial Information Gain Kernel

In this section, we will increase the dimensionality of the feature space F by making use of a polynomial kernel K_{poly}. We will start this section by giving an example [Cristianini and Shawe-Taylor, 2000]:

Example 2 *(Polynomial kernel): Consider a two dimensional input space X, with vectors $\mathbf{x}, \mathbf{y} \in X$:*

$$\begin{aligned} < \mathbf{x}, \mathbf{y} >^2 &= \left(\textstyle\sum_{i=1}^{2} x_i y_i\right)^2 \\ &= \textstyle\sum_{(i,j)=(1,1)}^{(2,2)} (x_i x_j)(y_i y_j) \\ &= K_{poly}(\mathbf{x}, \mathbf{y}) \end{aligned}$$

which is equivalent to an inner product between feature vectors of the form

$$\Phi_{poly}(\mathbf{x}) = (x_1^2, x_2^2, 2x_1x_2)$$

We call the kernel K_{poly} a polynomial kernel because it maps the vectors from the input space X to the feature space F of all monomials of degree 2.

Next, we will derive the *Polynomial Information Gain (PIG)* kernel K_{PIG}, making use of M1 from Section 2.1. We will do this based on the closure property mentioned in Section 2.1. Consider the feature mapping Φ_{IG} from Equation 11, the kernel K_{PIG} will be defined by a combination of this feature mapping and the polynomial kernel K_{poly} from the above example.

$$K_{PIG}(\mathbf{a}, \mathbf{b}) = K_{poly}(\Phi_{IG}(\mathbf{a}), \Phi_{IG}(\mathbf{b})) \tag{13}$$

with

$$\Phi_{IG} : \Omega^n \rightarrow \Re^{\Omega^n}$$

as defined in Equation 11 and

$$K_{poly} : \Re^{\Omega^n} \times \Re^{\Omega^n} \rightarrow \Re$$

From the closure properties of kernels, it follows naturally that K_{PIG} indeed is a valid kernel which calculates the inner product of two vectors transformed by a feature mapping $\Phi_{PIG} : \Omega^n \rightarrow \Re^{\Omega^m}$.

The mapping Φ_{PIG} maps vectors $\mathbf{a} \in \Omega^n$ to the space $\Re^{\Omega^m}$ with $m > n$. The features of the vectors in $\Re^{\Omega^m}$ are combinations (inner products) of the following form: $< \phi_{IG}(a_i), \phi_{IG}(a_i) >$ and $2 < \phi_{IG}(a_i), \phi_{IG}(a_j) >$. In this way we aim to capture more relevant information that might be present in such combinations. Notice that the inner product here above is the one defined in Equation 7.

5 Experimental Results

In the following section, we will give some experimental results we obtained with the kernels K_{IG} and K_{PIG}. First we will describe the PP attachment disambiguation problem in more detail, followed by the experimental set up (Section 5.2). Finally, we will present the results and compare them to other methods that have been applied on the same data set (Section 5.3).

5.1 PP attachment disambiguation problem

If it is uncertain in a sentence whether the preposition attaches to the verb or to the noun then we have a prepositional phrase (PP) attachment problem. For example, in sentence 1 of Example 1, **with** modifies the noun **shirt** because **with pockets** (PP) describes the **shirt**. In contrast, in sentence 2, **with** modifies the verb **washed** because **with soap** (PP) describes how the shirt is **washed** [Ratnaparkhi, 1998].

Example 3 *For the purpose of PP attachment disambiguation, sentences 1 and 2 from Example 1 will be reduced to vectors* $\mathbf{a}$ *and* $\mathbf{b} \in \Omega^4$ *as follows:*

1. $\mathbf{a}$ = *(bought, shirt, with, pockets)*
2. $\mathbf{b}$ = *(washed, shirt, with, soap)*

In fact, we only keep those words that are of any importance to the PP attachment problem, i.e. (V(erb),N(oun),P(reposition),N(oun)).

In the case where sentences are reduced to quadruples as illustrated in Example 3, the human performance is approximately 88.2% [Ratnaparkhi *et al.*, 1994]. This performance rate gives us an acceptable upper limit for the maximum performance of a computer because it seems unreasonable to expect an algorithm to perform much better than a human. As we will show in our experimental results the kernel K_{IG} achieves a classification accuracy up to 82,9%, see Section 5.3. However, in [Zavrel *et al.*, 1997] the IB1-IG attains a maximum classification accuracy of 84.1%, this is a good indication of the classification accuracy that should be possible to obtain with a kernel based on the distance defined in Equation 6.

5.2 Experimental Setup

The experiments have been done with LIBSVM, a C/C++ and Java library for SVMs [Chih-Chung and Chi-Jen, 2002]. The machine we have used is a Pentium III with 256MB RAM memory, running Windows XP. We choose to implement the kernels K_{IG} and K_{PIG} in Java.
The type of SVM learning we have used is C-SVM [Chih-Chung and Chi-Jen, 2002]. The parameter C controls the model complexity versus the model accuracy and has to be chosen based on the complexity of the problem.

The experiments in this section are conducted on a simplified version of the full PP attachment problem (see Example 3). The data consists of four-tuples of words, extracted from the Wall Street Journal Treebank [Marcus *et al.*, 1993] by a group at IBM [Ratnaparkhi *et al.*, 1994][1].

The data set contains 20801 training patterns, 3097 test patterns and an independant validation set of 4093 patterns for parameter optimization. All of the models described below were trained on all training examples and the results are given for the 3097 test patterns. For the benchmark comparison with other models from the literature, we use only results for which all parameters have been optimized on the validation set. For more details concerning the data set we refer to [Zavrel *et al.*, 1997].

5.3 Results

The table below gives the results obtained with the kernels K_{IG} and K_{PIG} and compares them to other methods taken from the literature:

Method	Percent correct
K_{IG}	82.9%
K_{PIG}	**84.8%**
IB1	83.7%
IB1-IG	84.1%
C4.5	79.7%
Maximum Entropy	77.7%
Transformations	81.9%
Back-off model	84.1%
Late Closure	59.0%
Most Likely for each P	72.0%

The results for K_{IG} and K_{PIG} were obtained after optimization of the parameter C on the validation set. For K_{IG} this optimal value was $C = 0.8$ and for K_{PIG} this optimal value was $C = 0.55$. The scores of IB1, IB1-IG, C4.5, Late Closure and Most Likely for each P are taken from [Zavrel *et al.*, 1997], the scores from Maximum Entropy are taken from [Ratnaparkhi *et al.*, 1994] and the scores of Transformations are taken from [Collins and Brooks, 1995].

The fact that the kernel K_{IG} performs worse than IB1-IG, although it is equipped with the very same distance metric, may seem somehow surprising. We believe this is because SVMs perform linear separation in the feature space F. The decision boundary of IB1-IG on the other hand is non-linear. Due to the linearity of the decision boundary of the SVM, some points get misclassified. The number of misclassifications is controlled by the parameter C. Choosing a larger

[1] This data set is freely available by ftp from **ftp://ftp.cis.upenn.edu/pub/adwait/PPattachData/**.

value for C forces the SVM to avoid classification errors. However, choosing a value for C that is too high will result in overfitting. Increasing the dimensionality of F makes linear separation easier (Cover's theorem). This can be seen by comparing the classification accuracies between the kernels K_{IG} and K_{PIG}. The results also show that our kernel K_{PIG} outperforms all other methods in the comparison.

6 Conclusion

In this paper, we have proposed a method for designing kernels for SVM learning in natural language settings. As a starting point we took distance functions that have been used in memory-based learning. The resulting kernel K_{PIG} achieves high classification accuracy compared to other methods that have been applied on the same data set. In our approach we started from the vectors of the input space, defined a mapping on those vectors and worked out their inner product in the feature space. All necesary kernel conditions follow naturally from the definition of the inner product. We used the distance formula for kernels to show that the kernels are actually based on the distance functions from Section 3. The experimental results of the kernel K_{PIG} show that increasing the dimensionality in the feature space yields better results. We increased the dimensionality of the feature space by making combinations of the features Φ_{IG} through a polynomial kernel. In this way, we do not only take into account the similarity between corresponding features of vectors, but we also take into account the similarity between non-corresponding features of the vectors. In fact this comes down to taking into account, to a greater extend, the context in wich a word occurs.

We used the *resolvement of PP attachment ambiguities* as a case-study to validate our findings, but it is our belief that the kernel K_{PIG} can be used for a wide range of natural language problems. In the future we will apply our kernels in more complex natural language settings and compare the results to other methods that have been applied on the same problems. At the moment we are already running experiments on *language independent named-entity recognition* [Sang, 2002]. The first results look promising, but it is too early to draw any significant conclusions. We will also try to further extend the kernels proposed in this paper to achieve even higher accuracies. Moreover, there are still many distance functions reported in the literature that have not yet been investigated to see whether they are applicable in SVM learning, we intend to investigate such distance functions and if possible derive a kernel from them.

References

[Aha *et al.*, 1991] D. Aha, D. Kibler, and M. Albert. Instance based learning algorithms. *Machine Learning*, 6:37 – 66, 1991.

[Brown *et al.*, 1999] Michael P.S. Brown, William Noble Grundy, David Lin, Nello Cristianini, Charles Sugnet, Manuel Ares, and David Haussler. Support vector machine classification of microarray gene expression data, 1999.

[Chih-Chung and Chi-Jen, 2002] Chang Chih-Chung and Lin Chi-Jen. Libsvm: a library for support vector machines, 2002.

[Collins and Brooks, 1995] M.J Collins and J. Brooks. Prepositional phrase attachment through a backed-off model. In *Proceedings of the Third Workshop on Very Large Corpora, Cambridge*, 1995.

[Cost and Salzberg, 1993] Scott Cost and Steven Salzberg. A weighted nearest neighbour algorithm for learning with symbolic features. *Machine Learning*, 10:57 – 78, 1993.

[Cristianini and Shawe-Taylor, 2000] Nello Cristianini and John Shawe-Taylor. *An Introduction to Support Vector Machines and other Kernel-based Learning Methods.* Cambridge University Press, 2000.

[Daelemans *et al.*, 2002] Walter Daelemans, Jakub Zavrel, Ko van der Sloot, and Antal van den Bosch. Timbl: Tilburg memory-based learner, version 4.3. Technical report, Tilburg University and University of Antwerp, 2002.

[Marcus *et al.*, 1993] M. Marcus, Santorini B, and M.A. Marcinkiewicz. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19(2):313 – 330, 1993.

[Mitchell, 1997] Tom Mitchell. *Machine Learning*. The McGraw-Hill Companies, Inc., 1997.

[Ramon, 2002] Jan Ramon. *Clustering and instance based learning in first order logic*. PhD thesis, K.U. Leuven, Belgium, 2002.

[Ratnaparkhi *et al.*, 1994] Adwait Ratnaparkhi, J. Reynar, and S. Roukos. A maximum entropy model for prepositional phrase attachment. In *Proceedings of the ARPA Workshop on Human Language Technology, Plainsboro, NJ*, 1994.

[Ratnaparkhi, 1998] Adwait Ratnaparkhi. *Maximum Entropy Models for Natural Language Ambiguity Resolution*. PhD thesis, University of Pennsylvania, Philadelphia, PA, 1998.

[Sang, 2002] Erik F. Tjong Kim Sang. Introduction to the conll-2002 shared task: Language-independent named entity recognition. In *Proceedings of CoNLL-2002, Taipei, Taiwan*, pages 155 – 158, 2002.

[Stetina and Nagao, 1997] Jiri Stetina and Makoto Nagao. Corpus based pp attachment ambiguity resolution with a semantic dictionary., 1997. Kyoto University, Yoshida Honmachi, Kyoto 606, Japan.

[Vapnik, 1998] Vladimir Vapnik. *The Nature of Statistical Learning Theory.* Springer-Verlag, New York, 1998.

[Veenstra *et al.*, 2000] Jorn Veenstra, Antal van den Bosch, Sabine Buchholz, Walter Daelemans, and Jakub Zavrel. Memory-based word senese disambiguation. *Computers and the Humanities*, 34:1-2:171 – 177, 2000.

[Zavrel *et al.*, 1997] Jakub Zavrel, Walter Daelemans, and Jorn Veenstra. Resolving pp attachment ambiguities with memory-based learning. In *Proceedings CoNNL, Madrid*, pages 136 – 144. Computational Linguistics, Tilburg University, 1997.