Integrating Gaussian Processes with Word-Sequence Kernels for Bayesian Text Categorization

Maryam Mahdaviani, Sara Forghanizadeh and Giuseppe Carenini
Department of Computer Science
University of British Columbia
{*maryam,forghani,carenini*}@*cs.ubc.ca*

Abstract

We address the problem of multi-labelled text classification using word-sequence kernels. However, rather than applying them with Support Vector Machine as in previous work, we chose a classifier based on Gaussian Processes. This is a probabilistic non-parametric method that retains a sound probabilistic semantics while overcoming the limitations of parametric methods. We present the empirical evaluation of our approach on the standard Reuters-21578 datasets.

1 Introduction

Text Categorization is the problem of classifying documents to a set of pre-specified categories. Filtering a stream of news, indexing documents for later retrieval, content analysis and spam filtering are just a few applications of text categorization. In order to classify texts, we need to identify a set of features by which we can correlate the documents to specified classes. Individual words, subsequence of terms, latent semantic indices are some widely used features for texts. However, in the space generated by such features classification is often impractical because the boundaries between classes are not linear.

By using $kernel\ \ trick$ in Kernel-based methods (Aizermanet al. , 1964) this problem can be effectively addressed. Essentially, documents are implicitly mapped to a feature space in which the classification becomes a simpler problem. In this work, we used non-contiguous sub-sequences of words as features of documents and computed the kernel matrix based on them. We focus on bigram non-contiguous subsequences of words as well as bag-of-words and averaged them with different weights. We penalized the gaps in non-contiguous subsequences by a decay factor.

In previous work, word-sequence kernels have been applied with Support Vector Machine (Cancedda et al., 2003). In contrast, we propose to use these kernels with a Gaussian Processes classifier. This is a probabilistic non-parametric method that retains a sound Bayesian probabilistic semantics while overcoming the limitations of parametric methods (Manning and Sch, 1999).

The main motivations behind our choice are both empirical and theoretical. Empirically, Gaussian Processes have been shown to outperform Support Vector Machine in text classification based on bag-of-words (Chai et al., 2002). On the theory side, by having a sound Bayesian probabilistic semantics, Gaussian Processes provide a confidence measure in the predications that can be used for instance to abstain on certain classification decisions (Altunet al. , 2004). To our knowledge, the integration of Gaussian Processes with word-sequence kernels is novel and it represents the main contribution of this paper.

Computing the similarity of two documents based on word-subsequences is not practical even for short documents. Therefore, we adopted and extended the dynamic programming approach of (Lodhi et al., 2002) to recursively compute the similarity.

In order to provide a fair evaluation, we test the performance of our methods on the Reuters-21578 dataset that has been used in previous related work. This dataset contains 21578 documents from Reuters newswire stories in 1987. The documents have been classified by human and have been compiled for public use. We used the same test/train split as the work of (Lodhi et al., 2001; Lodhi et al., 2002; Cancedda et al., 2003). Similarly to (Cancedda et al., 2003), we limited ourselves to the 10 most frequent categories rather than all 672 categories available for Reuters-21578.

As a preview of the paper, in section 2 we motivate our approach in the context of related work in text categorization. In section 3, we describe word-sequence kernels and how they are used to compute document similarity. After that, in section 4 we introduce Gaussian Processes. In section 5, we explain our evaluation including details of the dataset, the balancing of the data and the experiment settings. The outcomes of the experiments are discussed in section 6.

2 Related Work

In the past few years, a number of Machine Learning techniques have been applied to ranking and classification problems in Natural Language Processing, and text categorization is not an exception.

Similar to other complex classifications, text classification not only requires a strong classifier but also a set of features that represent the document. Bag-of-words models have been widely used in classification tasks (e.g., (Joachims, 1998)). The assumption behind bag-of-words representation is that the relative position of terms has little impact in Information Retrieval.

However, recent work in text categorization indicates that, in order to obtain a better classification, taking the order of terms is essential.

String Kernels are an effective way to take order into account. In this approach, the similarity between two documents is measured by the number of similar non-contiguous subsequence of items that the documents share. Non-contiguous sub-sequences are penalized with the number of gaps they have. The items considered can be, in order of increasing linguistic complexity, characters, words or concepts (i.e., senses)

Among the many different available classifiers, Support Vector Machine (SVM) and Boosting have been widely applied recently to text classification. (Joachims, 1998) has shown that SVM outperforms competing methods, such as decision trees and neural networks using bag-of-words as features. In (Chai et al., 2002), SVMs have been compared against Gaussian Processes (GPs from now on) also using bag-of-words. Results show that GP outperforms SVM on the Reuters-21578 dataset. Finally, in (Lodhi et al., 2002) and (Cancedda et al., 2003) SVM were successfully applied on the same dataset with kernels of character sequences and word sequences respectively.

Their results are currently the best reported in the literature. All this previous work motivated us to move a step further by integrating GPs with word sequence kernels.

Figure 1 shows our approach along with previous works in the context of the decision space for applying machine learning to text classification. At the top, the three most popular algorithms are considered (i.e., Neural Networks (NN), Kernel Methods (KM) and Decision Trees (DT)). The second level further specializes only KMs for which a classifier and a kernel have to be selected. Previous works and our approach are then pointed to by the corresponding classifier and kernel.

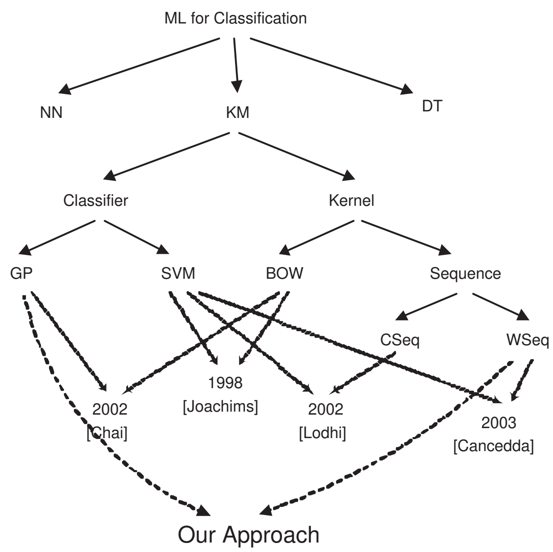

Figure 1: Previous work and our approach in the context of the decision space for applying machine learning to text categorization.

2.1 Word-Sequence Kernels

2.2 Representation and Explicit Computation of Document Similarity

Word Sequence Kernels (Cancedda et al., 2003) is a special case of string kernels. In String Kernels (Lodhi et al., 2001; Lodhi et al., 2002), one of the first departures from bag-of-words models, documents are not represented by word frequencies but by all possible ordered subsequences of characters. Following their notation, let Σ be a finite alphabet, and $s = s_1 s_2 ... s_{|s|}$ a sequence over Σ. Let $\mathbf{i} = [i_1\, i_2, ...\, i_n]$, where $1 \leq i_1 < i_2 < ... < i_n \leq |s|$, be a subset of the indices in s. We also write $l(\mathbf{i}) = i_n - i_1 + 1$. Then the similarity of two strings of characters s and t over Σ is defined as:

$$k_n(s,t) = \Sigma_{u \in \Sigma^n} \Sigma_{\mathbf{i}:s[\mathbf{i}]=u} \Sigma_{\mathbf{j}:t[\mathbf{j}]=u} \lambda^{l(\mathbf{i})+l(\mathbf{j})} \quad (1)$$

where $\lambda \in [0,1]$ is a decay factor for penalizing the gaps. Equation (1) defines a valid kernel function since in fact it is performing an inner product in a feature space with one feature per ordered subsequences $u \in \Sigma^n$ with value: $\Phi_u(s) = \Sigma_{\mathbf{i}:s[\mathbf{i}]=u} \lambda^{l(\mathbf{i})}$. The intuition behind this formulation is that we match all possible subsequences of length n, penalizing for the gaps in these subsequences. The longer the subsequence, the larger the power of λ will be. Their results show that character sequence kernels perform comparable to traditional bag-of-words methods. But (Cancedda et al., 2003) argue that since gaps within the sequences are allowed string kernels could also pick up parts of stems of consecutive words. As a result, (Cancedda et al., 2003) introduced Word-Sequence kernels in which Σ is a finite alphabet of all possible words rather than characters and $\mathbf{\Phi_u(s)}$ is the sum over lengths of all subsequences of document s in which feature u, a subsequence of words of length n, occurs. Table (2.2) shows examples of calculating Φ for two short documents s and t after POS-tagging and Lemmatization are applied and stop-words are removed. *s = China bought tons of U.S. wheat under the export enhancement program.*, and *t = China exported tons of wheat from US.*

Features	$\mathbf{\Phi}_u(s)$	$\mathbf{\Phi}_u(t)$
China	λ	λ
buy	λ	0
ton	λ	λ
U.S.	λ	λ
wheat	λ	λ
export	λ	λ
enhance	λ	0
program	λ	0

Table 1: Mapping documents s and t to the feature space of word subsequences of length 1. Note that by lemmatization for example “tons” and “exported” have been mapped to “ton” and “export” respectively, and that stop-words such as “of” and “under” have been eliminated

Similarly documents can be mapped to the feature spaces of word subsequences of higher lengths. Bi-gram subsequences contain all possible sequences of two words in two documents. For example u =*“export program”* is a non-contiguous subsequence that is seen in s and not in t. Therefore in feature space, the value of $\mathbf{\Phi}_u$ for t is 0 and for s is λ^3 where 3 is the length of “*‘export enhance program”*.

The similarity of each pair of documents s and t can then be computed by the inner product of their coordinates in feature space:

$$\mathbf{k}(s,t) = \Sigma_u \Phi_u(s).\Phi_u(t) \quad (2)$$

As an example, for the two sample documents mentioned above, we have $\mathbf{k}_1(s,t) = 6\lambda^2$, and $\mathbf{k}_2(s,t) = 2\lambda^9 + \lambda^8 + \lambda^6 + 2\lambda^5$ (corresponding to 1-grams and 2- grams respectively).

In order to keep the kernel values comparable for different values of n and independent of the length of sub-sequences, the normalized version of the above formula was used:

$$\hat{\mathrm{k}}(s,t) = \frac{\mathbf{k}(s,t)}{\sqrt{\mathbf{k}(s,s).\mathbf{k}(t,t)}} \quad (3)$$

Taking all n-grams into consideration, we will have the following formula which gives the similarity between two documents:

$$\bar{\mathrm{k}}_n(s,t) = \Sigma_{i=1}^{n} \mu^{l-i} \hat{\mathrm{k}}_i(s,t) \quad (4)$$

As we saw above, kernel values for different subsequence lengths are normalized independently before being combined. In this way it is possible to control the relative weight given to different subsequence length using the parameter μ.

2.3 Implicitly Computing Kernel Matrix

The proposed solution for computing the similarity of documents based on all possible subsequences is extremely expensive and impractical even for short documents. To solve this problem, (Lodhi et al., 2002) proposed a more efficient dynamic-programming-based approach to implicitly expand the feature space and recursively compute the similarity between two documents. The recursive formulation is based on the following reasoning: if we already know the value of the kernel for two strings s and t, we can compute the kernel for sx and t, where x is a word, by considering all word subsequences common to s and t and all new matching subsequences ending in x which occur in t and whose $(n-1)$-symbol prefix occur in s. They formulate this reasoning in equation (5), where $\mathbf{k'}_{n-1}(s,t)$ counts matching subsequences of $n-1$ symbols, but instead of discounting them according to the length of the subsequence; it discounts them according to the distance from the first symbol in the subsequence to the end of the complete sequence.

$$\mathbf{k}_n(sx,t) = \mathbf{k}_n(s,t) + \Sigma_{j:t_j=x}\lambda^2\mathbf{k'}_{n-1}(s,t[1:j-1]) \quad (5)$$

In order to make the kernel computation even more efficient, (Lodhi et al., 2002) introduced $\mathbf{k}''_i(sx,t)$ to recursively compute $\mathbf{k}'_i(sx,t)$. Intuitively $\mathbf{k}''_i(sx,t)$ stores the sum of the discounted masses of matches of length $(i-1)$ ending just before x. For further information on this implicit computation, the reader can refer to the appendix of (Cancedda et al., 2003).

While adopting this dynamic-programming technique in our approach, we observed that in longer documents, those subsequences with large power of λ can be ignored without effecting the accuracy of the performance. Therefore to enhance the efficiency of computation, we introduce parameter b to be the size of the window for considering subsequences. Therefore in feature-space any values smaller than λ^b is set to 0. By this modification, we managed to reduce the complexity of the system from $O(n|s||t|)$ to $O(nmax(|s|,|t|))$, where n is the gram and $|s|$ and $|t|$ are the length of documents s and t.

3 Prediction by Gaussian Processes (GPs)

As we mentioned in previous sections, (Lodhi et al., 2001; Lodhi et al., 2002; Cancedda et al., 2003) who devised and applied sequence kernels for text categorization, used Support Vector Machines as their kernel methods. In contrast, we propose to use GPs, which are non-parametric techniques for performing Bayesian inferences. Like SVMs, their non-parametric characteristic enables us to work in high and possibly infinite dimensional spaces. Although GPs are well-known for being computationally expensive, recent work on approximation methods (Smola and and Scholkopf, 2000; Seegaret. al., 2003) is addressing this problem. Furthermore, in our approach (like in (Cancedda et al., 2003)) the cost of the prediction step is much less that the cost of computing similarity of documents. As a result, GPs not only represent a promising computational alternative to SVMs, but they can bring up additional advantages because of their sound probabilistic nature.

GPs are a generalization of a finite Gaussian distributions and an instance of the class of stochastic processes. Similar to multivariate Gaussian distributions, they operate on functions instead of vectors. These processes are classical non-parametric methods. In recent years they have attracted a great deal of interest in machine learning (Gray, 2004). Formally, a GP is a distribution:

$$p(\mathbf{t}|\mathbf{K},\{x_n\}) = \frac{1}{\mathbf{Z}}exp(-\frac{1}{2}(\mathbf{t}-\mu)\mathbf{K}^{-1}(\mathbf{t}-\mu)),$$

where $\mathbf{t}$ is a set of random functions indexed by $\{x_n\}$, $\mathbf{t} = \{t(x_1), t(x_2), ...\}$, and matrix $\mathbf{K}$ is the covariance matrix of these functions. With a Gaussian Process model we can predict the value of t_{N+1} given x_{N+1} and a set of observations on variables. GPs is a promising non-linear interpolation tool, but they can be modified to produce efficient classifiers (Lang, 2004). In the classification task the goal is to predict the labels of test variables based on their similarity to labeled training data. The prediction can be approximated (Williams and Barber, 1998; Williams and Seeger, 2000) and formulated in the form of matrix vector multiplication:

$$\mathbf{y}^{(u)} = \mathbf{K_{ul}}\mathbf{K_{ll}}^{-1}\mathbf{y}^{(l)}. \quad (6)$$

where $\mathbf{y}^{(u)}$ and $\mathbf{y}^{(l)}$ represent vector of labels for test and train variables respectively, $\mathbf{K_{ll}}$ is the covariance matrix for labeled variables and $\mathbf{K_{ul}}$ is the similarity matrix between train and test variables. We can measure the similarity between variables with different kernel functions as long as they produce a valid kernel matrix. As described in the previous section, in our approach we use word-sequence Kernels which is a variation of polynomial kernels for measuring the similarity of documents. Therefore, the similarity between documents x_i and x_j is measured by $\bar{\mathrm{k}}_n(x_i,x_j)$, defined in equation (4), where x_i and x_j can be train or test documents.

Note that entries in matrix $\mathbf{K_{ll}}$ represent the similarity for all pairs of train variables (i.e., documents) $x_i^{(l)}$ and $x_j^{(l)}$. As a result matrix $\mathbf{K_{ll}}$ is a valid kernel matrix and consequently symmetric positive definite. Entries of matrix

$\mathbf{K_{ul}}$, $k(x_i^{(u)}, x_j^{(l)})$, represent the similarity between unlabeled variables $x_i^{(u)}$'s and labeled variables $x_j^{(l)}$'s. Therefore, $\mathbf{K_{ul}}$ and $\mathbf{K_{ll}}$ are $N\mathbf{x}M$ and $M\mathbf{x}M$ matrices where N is the size of unlabeled test variables and M represents the size of the training set. With our choice of kernel function, higher similarity values will be assigned to points that are closer to each other in feature space. We can extend binary classification with Gaussian Process to multi-class classification. In other words we can introduce a matrix of labels $\mathbf{Y}^{(u)}$ and $\mathbf{Y}^{(l)}$ instead of vector of labels. Consequently, equation (6) can be written as:

$$\mathbf{Y}^{(u)} = \mathbf{K_{ul}}\mathbf{K_{ll}}^{-1}\mathbf{Y}^{(l)}. \tag{7}$$

where now $\mathbf{Y}^{(u)} \in \mathbf{R}^{N\mathbf{x}T}$, $\mathbf{Y}^{(l)} \in \mathbf{R}^{M\mathbf{x}T}$ and T is the number of classes including the background class.

$\mathbf{Y}_{uj}$ is a vector of responses for test documents for class j. Entry i in this vector corresponds to the chance that document i belongs to class j. $\mathbf{Y}_{lj}$ is a vector of binary values of 0 and 1 for each classified document. A value of 1 is assigned to the label of a document if that document belongs to class j. $\mathbf{K}_{ul}$ is just a kernel matrix in which entries are the similarities between all pairs of unlabeled and labeled documents. Similarly, $\mathbf{K}_{ll}$ holds the similarity measure between all labeled documents. After computing $\mathbf{Y}_u$ for all classes, for each document i we have a set of responses, which are correlated to the likelihood of having document i in each class. For further information on this process, the reader may refer to David Mackay's tutorial (MacKay, 1997) and Mark Gibbs' thesis (Gibbs, 1997).

Since we know that all documents belong to at least one class, we take the topic corresponding to maximum response, as the first class of the document. Next, a document is also assigned to all additional classes for which the response is above a given threshold. We determined this threshold by cross-validation on a development dataset.

4 Evaluation

4.1 Dataset of Documents: Reuters-21578

In order to provide a fair evaluation, we need to test the performance of our methods on the same dataset that has been used in previous related work. Most text classification methods in the recent years have been tested on the Reuters-21578 dataset. This dataset contains 21578 documents from Reuters newswire stories in 1987. The documents have been classified by human and have been compiled for public use.

There is a so-called "ModApte split" in Reuters, which uses a total number of 12,902 documents: 9,603 documents for training and 3,299 for testing. In all recent work on word sequence kernels, only the documents in the ten most frequent categories from this set have been used. For the sake of comparison, we focused on the same ten categories, which left us with 7,193 documents for training, and 2,787 documents for testing. Table (4.1) shows the ten categories that were used.

Category	# of test	# of training	Neg/Pos
earn	2877	1087	3
acq	1650	719	6
money	538	179	18
grain	433	149	22
crude	389	189	25
trade	369	117	26
interest	347	131	28
ship	197	89	49
wheat	212	71	45
corn	181	56	53

Table 2: Number of positive examples in the ten most frequent categories of the Reuters-21578 corpus (ModApte split)

4.2 Balancing the Data

Similar to many other real-world classification tasks, in this classification task we are facing unbalanced data. To reduce the impact of unbalanced data in their tasks, (Cancedda et al., 2003) set the parameter of SVM^{light} to the integer closest to the ratio of Negative/Positive examples for each category. However using GPs, we need to use a different strategy. Since we are performing multi-labeled classification (rather than binary classification for each topic), we would like to have a relatively similar Negative/Positive ratio for all categories of train documents. To achieve this goal we can either under-sample the majority classes and leave the minority classes unchanged or we can over-sample the minority classed and leave the majority classes unchanged. Each of these approaches have their own advantages and disadvantages. On the one hand, by under-sampling we only use a randomly chosen subset of examples from majority classes. The problem with this approach is that we may miss a number of good training examples and as a result the accuracy of supervised prediction decreases. On the other hand, by over-sampling minority classes, we can duplicate the minority classes. And the problem with this solution is that as a result the kernel matrix will have dependent rows, becoming ill-conditioned (Kueck, 2004). Our solution is a compromise of these two approaches. We randomly split the majority classes in a way that each split contains between 180-220 documents. We then randomly picked a split from each category and formed 7 sets of training documents. With each of these training sets, we calculated the labels for test documents. At the end we averaged the labels obtained by each training set.

4.3 Experiments

In all experiments we fixed the value of decay factor, λ, and the weight factor, μ, to 0.5. (Cancedda et al., 2003) had extensively experimented with different values of λ, μ and n. They suggested to define λ_{match} for rewarding matches and λ_{gap} for penalizing gaps. They claimed that their best results were achieved by varying λ_{match} and λ_{gap}. However, we limited ourselves to fixed decay factors, λ, without introducing any parameter for rewarding matches. So, we compare our results only with the results of their approach with fixed parameters. As for the parameter n, the length of subsequences of words, we also followed (Cancedda et al., 2003). (Cancedda et al., 2003) reported that by considering subsequences longer than 2, although the precision increases slightly, the recall drops significantly. As a result, we also set n to 2, (i.e. we also consider only unigrams and bigrams).

We performed two sets of experiments. In the first set of experiments, we initially performed POS-tagging by applying $java\ QTag$. Next we lemmatized all documents by applying the $Stanford\ JavaNLPlemmatizer$. Then, we computed the similarity of documents represented by sequences of lemmas except for stop words. This pre-processing of documents is the same as that of (Cancedda et al., 2003). Therefore, GPs can be compared to SVMs under similar conditions, and both using word-sequence kernels.

In the second set of experiments, we again performed POS-tagging with the same java tagger. But then we only kept the nouns contained in the documents without any further processing. After that, we computed the similarity of documents represented by sequences of nouns. The purpose of the second set of experiments was to explore the accuracy of document-classifications when documents are represented only with some parts of speech. Since computing string kernels are expensive, we are looking for ways of improving the efficiency without considerably losing accuracy.

In all our experiments, to test the effectiveness of our approach, we compared our predicted class labels with real labels of test documents to compute the true positives TP (number of documents the model correctly identifies as belonging to the target class), the false positives FP (number of documents the system falsely identifies as belonging to the target class) and the false negatives FN (number of documents the model fails to identify as belonging to the target class). The performance measures will be:
$Precision$: the ratio of true positives among all retrieved documents, $p = \frac{TP}{TP+FP}$
$Recall$: The ratio of true positives over all positives, $r = \frac{TP}{TP+FN}$
F-score: The harmonic mean of precision and recall, $F_1 = \frac{2pr}{p+r}$.
We report our results by computing the micro and macro averages of these three performance measures as it is standard in text categorization.

5 Results

We compared the accuracy of our approach against the best results reported in the literature (Cancedda et al., 2003). The outcome of our first set of experiments when we consider all lemmas except for stop words are shown in Table 3 and Table 4. In particular, Table 3 shows that our approach, GPs with word-sequence (GP-WK) outperforms SVMs with the same kernels (SVM-WK) when compared with respect to macro-average. Since the authors of (Cancedda et al., 2003) shared with us their results for each category, we could also run a paired two-tailed t-test on performance of (Cancedda et al., 2003), SVM-WK, and our approach, GP-WK, for each category. The differences for accuracy, precision and F_1 measure were all statistically significant ($p - value < .009$).

In contrast, as shown in Table 4, SVM-WK outperforms GP-WK when compared with respect to micro-average. However, this difference is

Table 3: Macro-Average: All lemmas (stop-words removed)

method and kernel	*Macro-Average*		
	p	r	F_1
GP-WK	**90.58**	**89.69**	**90.13**
SVM-WK	85.37	76.71	80.64

Table 4: Micro-Average: All lemmas (stop-words removed)

method and kernel	*Micro-Average*		
	p	r	F_1
GP-WK	88.73	85.85	87.26
SVM-WK	**93.01**	**88.09**	**90.52**

much less pronounced (3% vs. 10%) than the one for macro-average. So, it appears that GP-WK should be the approach of choice unless in the target application micro-average performance is substantially more critical than the macro-average one. As for statistical significance of the micro-average differences, since we did not have access to raw data about multiple runs of the SVM-WK approach, we cannot provide a definite estimate. However, given the size of the differences and the size of the population, we hypothesize the differences in micro-average also to be significant.

With respect to the second set of experiments, notice that, as shown in Tables 6 and 5, even when only nouns are used, we obtain rather accurate results for both micro and macro averages. This confirms that nouns carry most of the information needed for text categorization and since considering nouns only is computationally more efficient in some domains this can represent in practice the most satisfactory solution.

6 Conclusions and Future Work

Text categorization is a key task in NLP. All the most popular machine learning algorithms have been applied to this task, with kernel methods being the most successful. Most combinations of different kernels and classifiers have been tested in previous work, but to our knowledge the integration of Gaussian Processes with word-sequence kernels is novel and it represents the main contribution of this paper.

Table 5: Macro-Average: All Nouns

method and kernel	*Macro-Average*		
	p	r	F_1
GP-WK	87.35	84.77	86.04

Table 6: Micro-Average: All Nouns

method and kernel	*Micro-Average*		
	p	r	F_1
GP-WK	85.64	79.26	82.32

In our experiments, we compared GPs with word-sequence kernels against the best approach reported in the literature, namely SVMs with word-sequence kernels. These experiments indicate that while GPs outperform SVMs with respect to macro-average, they are outperformed by SVMs with respect to micro-average. However the dominance of GPs on macro-average is much more pronounced than the dominance of SVMs on micro-average. Furthermore, GPs provide an additional critical advantage when compared to SVMs. Since GPs have a sound Bayesian probabilistic semantics, they can be effectively used in Bayesian decision making. For instance, they provide a confidence measure in the predications that can be used to abstain on certain classification decisions.

The next step in our research is to combine GPs with concept-sequence kernels. In practice, instead of computing similarity of documents by exact matching of subsequences of words, we plan to consider a form of soft matching in which similarity scores are based on the semantic distance of the constituent words. Initially, we intend to experiment with measures of semantic distance based on Wordnet.

7 Acknowledgements

We would like to thank Nando de Freitas and Kevin Murphy for their helpful insights and advice. We also thank Nicola Cancedda, Eric Gaussier, Cyril Goutte and Jean-Michel Renders for providing us with their experimental results.

References

M Aizerman, E Braverman and L Rozonoer. 1964. *Theoretical foundations of the potential function method in pattern recognition learning*. Automation and Remote Control,volume 25, pages 821-837.

Y Altun, T Hofmann and A Smola. 2003. *Gaussian*

Process Classification for Segmenting and Annotating Sequences. 21th International Conference on Machine Learning.

L Cai, T Hofmann. 2005. *Text Categorization by Boosting Automatically Extracted Concepts*. the 26th Annual International ACM SIGIR onference on Research and Developement in Information Retrieval, Toronto, Canada.

N Cancedda, E Gaussier, C Goutte, J Renders. 2003. *Word-Sequence Kernels*. Journal of Machine Learning Research, volume 3, pages 1059-1082 .

K M A Chai, H T Ng, H L Chieu. 1997. *Bayesian Online Classifier for Text Classification and Filtering*. 25th ACM International Conference on Research and Developement in Information Retrieval, Tampere, FL.

E Gabrilovich, S Markovitch. 2004. Text Categorization with Many Redundant Features: Using Aggressive Feature Selection to Make SVMs competetive with C4.5. *the Twenty-First International Conference on Machine Learning*.

T Joachims. 1998. Making Large Scale SVM Learning Practical. *Bernhard Scholkopf, Chris Burges, and Alex Smola, editors, Advances in Kernel Methods-Support Vector Learning, MIT Press, 1999*.

D MacKay. 1997. *Introduction to Gaussian Processes*. http://www.cs.toronto.edu/ mackay/gpB.ps.gz.

M N Gibbs. 1997. *Bayesian Gaussian Processes for Regression and Classification. PhD Thesis, University of Cambridge*.

E Gabrilovich, S Markovitch. 2004. Text Categorization with Many Redundant Features: Using Aggressive Feature Selection to Make SVMs competetive with C4.5. *the Twenty-First International Conference on Machine Learning*.

A G Gray. 2004. Fast Kernel Matrix-Vector Multiplication with Application to Gaussian Process Learning. *Technical Report, Carnegie Mellon University, CMU-CS-04-110,*.

H Kueck. 2004. Bayesian Formulations of Multiple Instance Learning with Applications to General Object Recognition. *MSc Thesis, Department of Computer Science, University of British Columbia*.

D Lang. 2004. Fast Methods for Inference in Graphical Models and Beat Tracking the Grapgical Model Way. *MSc Thesis, Department of Computer Science, University of British Columbia*.

H Lodhi, C Saunders, J Shaw-Taylor, N Cristiani, C Watkins. 2001. *Text Classification using String Kernels*. Advances in Neural Information Processing Systems, 13. MIT Press

H Lodhi, C Saunders, J Shaw-Taylor, N Cristiani, C Watkins. 2002. *Text Classification using String Kernels*. Journal of Machine Learning Research, 2, pages 419-444.

C D Manning and H Schutze 1999. Foundations of Statistical Natural Language Processing. *Cambridge, Massachusetts*

M Seegar, N D Lawrence and R Hebrich. . Fast Gaussian Sparse Methods: The informative vector machine. *Advances in Neural Information Processing Systems*.

A J Smola and B Scholkopf. 2000. Sparse Greedy Matrix Approximation for Machine Learning. *17th International Conference in International Conf. on Machine Learning*, pages 911-918.

C K I Williams and D Barber. 1998. Bayesian Classification with Gaussian Processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 20, pages 1342-1351

C K I Williams and M Seeger. 2000. Using the Nystrom Method to Speed-uo Kernel Machines. *Advances in Neural Information Processing Systems*.

Y Yang and X Liu. 1999. A re-examination of text categorization methods. *Proceeding of the 22nd ACM SIGIR Conference on Research and Developement in Information Retrieval*, pages 42-49.