Bayesian Kernel Methods for Natural Language Processing

Daniel Beck
Department of Computer Science
University of Sheffield
Sheffield, United Kingdom
debeck1@sheffield.ac.uk

Abstract

Kernel methods are heavily used in Natural Language Processing (NLP). Frequentist approaches like Support Vector Machines are the state-of-the-art in many tasks. However, these approaches lack efficient procedures for model selection, which hinders the usage of more advanced kernels. In this work, we propose the use of a Bayesian approach for kernel methods, Gaussian Processes, which allow easy model fitting even for complex kernel combinations. Our goal is to employ this approach to improve results in a number of regression and classification tasks in NLP.

1 Introduction

In the last years, kernel methods have been successfully employed in many Natural Language Processing tasks. These methods allow the building of non-parametric models which make less assumptions about the underlying pattern in the data. Another advantage of kernels is that they can be defined in arbitrary structures like strings or trees, which greatly reduce the need for careful feature engineering in these structures.

The properties cited above make kernel methods ideal for problems where we do not have much prior knowledge about how the data behaves. This is a common setting in NLP, where they have been mostly applied in the form of Support Vector Machines (SVMs). Systems based on SVMs have been the state-of-the-art in classification tasks like Text Categorization (Lodhi et al., 2002), Sentiment Analysis (Johansson and Moschitti, 2013; Pérez-Rosas and Mihalcea, 2013) and Question Classification (Moschitti, 2006; Croce et al., 2011). Recently, they were also employed in regression settings like Machine Translation Quality Estimation (Specia and Farzindar, 2010; Bojar et al., 2013) and structured prediction (Chang et al., 2013).

SVMs are a frequentist method: they aim to find an approximation to the exact latent function that explains the data. This is in contrast to Bayesian settings, which define a prior distribution on this function and perform inference by marginalizing over all its possible values. Although there is some discussion about which approach is better (Murphy, 2012, Sec. 6.6.4), Bayesian methods offer many useful theoretical properties. In fact, they have been used before in NLP, especially in grammar induction (Cohn et al., 2010) and word segmentation (Goldwater et al., 2009). However, only very recently kernel methods have been applied in NLP using the Bayesian approach.

Gaussian Processes (GPs) are the Bayesian counterpart of kernel methods and are widely considered the state-of-the-art for inference on functions (Hensman et al., 2013). They have a number of advantages which are very useful in NLP:

- Kernels in general can be combined and parameterized in many ways. This parameterization lead to the problem of model selection, which is difficult in frequentist approaches (mainly based on cross validation). The Bayesian formulation of GPs let them deal with model selection in a much more more efficient and elegant way: by maximizing the likelihood on the training data. This opens the door for the use of heavily parameterized kernel combinations, like multi-task kernels for example.

- Being a probabilistic framework, they are able to naturally encode uncertainty in the predictions, which can be propagated if the task is part of a larger system pipeline.

Besides these properties, GPs have also been applied sucessfully in many Machine Learning

Proceedings of the ACL 2014 Student Research Workshop, pages 1–9,
Baltimore, Maryland USA, June 22-27 2014.

tasks. Examples include Robotics (Ko et al., 2007), Bioinformatics (Chu et al., 2005; Polajnar et al., 2011), Geolocation (Schwaighofer et al., 2004) and Computer Vision (Sinz et al., 2004; Riihimäki et al., 2013). In NLP, GPs have been used only very recently and focused on regression tasks (Cohn and Specia, 2013; Preotiuc-Pietro and Cohn, 2013). In this work, we propose to combine GPs with recent kernel developments to advance the state-of-the-art in a number of NLP tasks.

2 Gaussian Processes

In this Section, we follow closely the definition of Rasmussen and Williams (2006). Consider a machine learning setting, where we have a dataset $\mathcal{X} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)\}$ and our goal is to infer the underlying function $f(\mathbf{x})$ that best explains the data. A GP model assumes a prior stochastic process over this function:

$$f(\mathbf{x}) \sim \mathcal{GP}(\mu(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')), \tag{1}$$

where $\mu(\mathbf{x})$ is the *mean* function, which is usually the $\mathbf{0}$ constant, and $k(\mathbf{x}, \mathbf{x}')$ is the kernel or *covariance* function. In this sense, they are analogous to Gaussian distributions, which are also defined in terms of a mean and a variance values, or in the case of multivariate Gaussians, a mean vector and a covariance matrix. In fact, a GP can be interpreted as an infinite-dimensional multivariate Gaussian distribution.

The full model uses Bayes' rule to define a posterior over f, combining the GP prior with the data likelihood:

$$p(f|\mathbf{X}, \mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{X}, f)p(f)}{p(\mathbf{y}|\mathbf{X})}, \tag{2}$$

where $\mathbf{X}$ and $\mathbf{y}$ are the training inputs and outputs, respectively. The posterior is then used to predict the label for an unseen input $\mathbf{x}_*$ by marginalizing over all possible latent functions:

$$p(y_*|\mathbf{x}_*, \mathbf{X}, \mathbf{y}) = \int_f p(y_*|\mathbf{x}_*, \mathbf{X}, f)p(f|\mathbf{X}, \mathbf{y})df. \tag{3}$$

where y_* is the predicted output. The choice of the likelihood distribution depends if the task is regression, classification or other prediction setting.

2.1 GP Regression

In a regression setting, we assume that the output values are equal to noisy latent function evaluations, i.e., $y_i = f(\mathbf{x_i}) + \eta$, where $\eta \sim \mathcal{N}(0, \sigma_n^2)$ is the added white noise. We also usually assume a Gaussian likelihood, because this able us to solve the integral in Equation 3 analytically. Substituting the likelihood and the prior in both Equations 2 and 3 and manipulating the result, we compute the posterior also as a Gaussian distribution:

$$y_* \sim \mathcal{N}(\mathbf{k}_*(\mathbf{K} + \sigma_n\mathbf{I})^{-1}\mathbf{y}^T, \\ k(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{k}_*^T(\mathbf{K} + \sigma_n\mathbf{I})^{-1}\mathbf{k}_*). \tag{4}$$

where $\mathbf{K}$ is the Gram matrix corresponding to the training inputs and $\mathbf{k}_* = [\langle \mathbf{x}_1, \mathbf{x}_* \rangle, \langle \mathbf{x}_2, \mathbf{x}_* \rangle, \ldots, \langle \mathbf{x}_n, \mathbf{x}_* \rangle]$ is the vector of kernel evaluations between the test input and each training input.

2.2 GP Classification

Consider binary classification using -1 and $+1$ as labels[1]. The model in this case use the actual, noiseless latent function evaluations $\mathbf{f}$ and "squash" them through the $[-1, +1]$ interval to obtain the outputs. The posterior over the outputs is then defined as:

$$p(y_* = +1|\mathbf{x}_*, \mathbf{X}, \mathbf{y}) = \\ \int_{f_*} \sigma(f_*)p(f_*|\mathbf{x}_*, \mathbf{X}, \mathbf{y})df_*, \tag{5}$$

where $\sigma(f_*)$ is a squashing function. Two common choices are the logistic function and the probit function. The distribution over the latent values f_* is obtained by integrating out the latent function:

$$p(f_*|\mathbf{x}_*, \mathbf{X}, \mathbf{y}) = \int_f p(f_*|\mathbf{x}_*, \mathbf{X}, f)p(f|\mathbf{X}, \mathbf{y})df. \tag{6}$$

Because the likelihood is not Gaussian, the resulting posterior integral is not analytically available anymore. The most common solution to this problem is to approximate the posterior $p(f|\mathbf{X}, \mathbf{y})$ with a Gaussian $q(f|\mathbf{X}, \mathbf{y})$. Two such approximation algorithms are the Laplace approximation (Williams and Barber, 1998) and the Expectation Propagation (Minka, 2001). Another option is to use Markov Chain Monte Carlo sampling methods on the true posterior (Neal, 1998).

2.3 Hyperparameter Optimization

The GP prior used in the models described above usually have a number of hyperparameters. The

[1] Extensions to multi-class settings are possible.

most important ones are the kernel ones but they can also include others like the white noise variance σ_n^2 used in regression. A key property of GPs is their ability to easily fit these hyperparameters to the data by maximizing the marginal likelihood:

$$p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta}) = \int_f p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta}, f)p(f), \qquad (7)$$

where $\boldsymbol{\theta}$ represents the full set of hyperparameters (which was suppressed from all conditionals until now for brevity). Optimization involves deriving the gradients of the marginal log likelihood w.r.t. the hyperparameters and then employ a gradient ascent procedure. Gradients can be found analitically for regression and by approximations for classification, using methods similar to the ones used for prediction.

2.4 Sparse Approximations for GPs

SVMs are naturally sparse models which use only a subset of data points to make predictions. This results in important speed-ups which is one of the reasons for their success. On the other hand, canonical GPs are not sparse, making use of all data points. This results in a training complexity of $O(n^3)$ (due to the Gram matrix inversion) and $O(n)$ for predictions.

Sparse GPs tackle this problem by approximating the Gram matrix using only a subset of m *inducing inputs*. Without loss of generalization, consider these m inputs as the first ones in the training data and $(n - m)$ the remaining ones. Then we can partition the Gram matrix in the following way (Rasmussen and Williams, 2006, Sec. 8.1):

$$\mathbf{K} = \begin{bmatrix} \mathbf{K}_{mm} & \mathbf{K}_{m(n-m)} \\ \mathbf{K}_{(n-m)m} & \mathbf{K}_{(n-m)(n-m)} \end{bmatrix},$$

where each block corresponds to a matrix of kernel evaluations between two sets of inputs. For brevity, we will refer $\mathbf{K}_{m(n-m)}$ as $\mathbf{K}_{mn}$ and its transpose as $\mathbf{K}_{nm}$. The block structure of $\mathbf{K}$ forms the base of the so-called Nyström approximation:

$$\tilde{\mathbf{K}} = \mathbf{K}_{nm}\mathbf{K}_{mm}^{-1}\mathbf{K}_{mn}. \qquad (8)$$

which result in the following predictive posterior:

$$\begin{aligned} y_* \sim \mathcal{N}(&\mathbf{k}_{m*}^T\tilde{\mathbf{G}}^{-1}\mathbf{K}_{mn}\mathbf{y}, \qquad (9) \\ &k(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{k}_{m*}^T\mathbf{K}_{mm}^{-1}\mathbf{k}_{m*} + \\ &\sigma_n^2\mathbf{k}_{m*}^T\tilde{\mathbf{G}}^{-1}\mathbf{k}_{m*}), \end{aligned}$$

where $\tilde{\mathbf{G}} = \sigma_n^2\mathbf{K}_{mm} + \mathbf{K}_{mn}\mathbf{K}_{nm}$ and $\mathbf{k}_{m*}$ is the vector of kernel evaluations between test input $\mathbf{x}_*$ and the m inducing inputs. The resulting complexities for training and prediction are $O(m^2n)$ and $O(m)$, respectively.

The remaining question is how to choose the inducing inputs. Seeger et al. (2003) use an iterative method that starts with some random data points and adds new ones based on a greedy procedure, in an active learning fashion. Snelson and Ghahramani (2006) use a different approach: it defines a fixed m a priori and use *pseudo-inputs* which can be optimized as regular hyperparameters. Later, Titsias (2009) also used pseudo-inputs but perform optimization using a variational method instead. Recently, Hensman et al. (2013) modified this method to allow Stochastic Variational Inference (Hoffman et al., 2013), which reduces the training complexity to $O(m^3)$.

3 Kernels

The core of a GP model is the kernel function. A kernel $k(\mathbf{x}, \mathbf{x}')$ is a symmetric and positive semi-definite function which returns a similarity score between two inputs in some feature space (Shawe-Taylor and Cristianini, 2004). Probably the most used kernel in general is the Radial Basis Function (RBF) kernel, which is defined over two real-valued vectors. Our focus in this work is on two different types of kernels which can be applied for NLP settings and allow richer parameterizations.

3.1 Kernels for Discrete Structures

In NLP, discrete structures like strings or trees are common in training data. To apply a vectorial kernel like the RBF, one can always extract real-valued features from these structures. However, kernels can be defined directly on these structures, potentially reducing the need for feature engineering. The string and tree kernels we define here are based on the theory of Convolution kernels of Haussler (1999), which calculate the similarity between two structures based on the number of substructures they have in common. Other approaches include random walk kernels (Gärtner et al., 2003; Vishwanathan et al., 2010) and Fisher kernels (Jaakkola et al., 2000).

3.1.1 String Kernels

Consider a function $\phi_s(\mathbf{x})$ that counts the number of times a substring s appears in $\mathbf{x}$. A string kernel

is defined as:

$$k(x, x') = \sum_{s \in \Sigma^*} w_s \phi_s(x) \phi_s(x'), \qquad (10)$$

where w_s is a non-negative weight for substring s and Σ^* is the set of all possible strings over the symbol alphabet Σ.

Usually in NLP, each word is considered a symbol, although some previous work also considered characters as symbols (Lodhi et al., 2002). If we restrict s to be only single words we end up having a *bag-of-words* (BOW) representation. Allowing longer substrings lead us to the Word Sequence Kernels of Cancedda et al. (2003), which also allow gaps between words.

One extension of these kernels is to allow soft matching between substrings. This is done by defining a similarity matrix $\mathbf{S}$, which encode symbol similarities. This matrix can be defined by external resources, like WordNet, or be inferred from data using Latent Semantic Analysis (Deerwester et al., 1990) for example.

3.1.2 Tree Kernels

Collins and Duffy (2001) first introduced Tree Kernels, which measure the similarity between two trees by counting the number of fragments they share, in a very similar way to string kernels. Consider two trees T_1 and T_2. We define the set of nodes in these two trees as N_1 and N_2 respectively. Consider also $\mathcal{F}$ the full set of possible tree fragments (similar to Σ^* in the case of strings). We define $I_i(n)$ as an indicator function that returns 1 if fragment $f_i \in \mathcal{F}$ has root n and 0 otherwise. A Tree Kernel can then be defined as:

$$k(T_1, T_2) = \sum_{n_1 \in N_1} \sum_{n_2 \in N_2} \Delta(n_1, n_2),$$

where:

$$\Delta(n_1, n_2) = \sum_{i=1}^{|\mathcal{F}|} \lambda^{\text{size}(i)} I_i(n_1) I_i(n_2).$$

Here, $0 < \lambda < 1$ is a decay factor that penalizes contributions from larger fragments cf. smaller ones.

Again, we can put restrictions on the type of tree fragment considered for comparison. Collins and Duffy (2001) defined Subtree kernels, which considered only subtrees as fragments, and Subset Tree Kernels (SSTK), where fragments can have non-terminals as leaves. Later, Moschitti (2006) introduced the Partial Tree Kernels (PTK), by allowing fragments with partial rule expansions.

Tree kernels were used in a variety of tasks, including Relation Extraction (Bloehdorn and Moschitti, 2007; Plank and Moschitti, 2013), Question Classification (Moschitti, 2006; Croce et al., 2011) and Quality Estimation (Hardmeier, 2011; Hardmeier et al., 2012). Furthermore, soft matching approaches were also used by Bloehdorn and Moschitti (2007) and Croce et al. (2011).

3.2 Multi-task Kernels

Kernels can also be extended to deal with settings where we want to predict a vector of values (Álvarez et al., 2012). These settings are useful in multi-task and domain adaptation problems. Kernels for vector-valued functions are known as *coregionalization* kernels in the literature. Here we are going to refer them as multi-task kernels.

One of the simplest ways to define a kernel for a multi-task setting is the Intrinsic Coregionalization Model (ICM):

$$\mathbf{K}(\mathbf{x}, \mathbf{x}') = \mathbf{B} \otimes k(\mathbf{x}, \mathbf{x}').$$

where $\otimes$ denotes the Kronecker product and $\mathbf{B}$ is the coregionalization matrix, encoding task covariances. We also denote the resulting kernel function as $\mathbf{K}(\mathbf{x}, \mathbf{x}')$ to stress out that its result is now a matrix instead of a scalar.

Cohn and Specia (2013) used the ICM to model annotator bias in Quality Estimation datasets. They parameterize $\mathbf{B}$ in a number of different ways and get significant improvements over single-task baselines, especially in post-editing time prediction. They also point out that the well known EasyAdapt method (Daumé III, 2007) for domain adaptation can be modeled by the ICM using $\mathbf{B} = \mathbf{1} + \mathbf{I}$, i.e., a coregionalization matrix with its diagonal elements equal to 2 and remaining elements equal to 1.

An extension of the ICM is the Linear Model of Coregionalization (LMC), which assume a sum of kernels with different coregionalization matrices:

$$\mathbf{K}(\mathbf{x}, \mathbf{x}') = \sum_{k_p \in P} \mathbf{B}_p \otimes k_p(\mathbf{x}, \mathbf{x}').$$

where P is the set of different kernels employed. Álvarez et al. (2012) argue that the LMC is much more flexible than the ICM because the latter assumes that each kernel contributes equally to the task covariances.

4 Planned Work

Our goal in this proposal is to employ GPs and the kernels introduced in Section 3 to advance the state-of-the-art in regression and classification NLP tasks. It would be unfeasible though, at least for a single thesis, to address all possible tasks so we are going to focus on three of them where kernel methods were already successfully applied.

4.1 Quality Estimation

The purpose of Machine Translation Quality Estimation is to provide a quality prediction for new, unseen machine translated texts, without relying on reference translations (Blatz et al., 2004; Bojar et al., 2013). A common use of quality predictions is the decision between post-editing a given machine translated sentence and translating its source from scratch.

GP regression models were recently successfully employed for post-editing time (Cohn and Specia, 2013) and HTER[2] prediction (Beck et al., 2013). Both used RBF kernels as the covariance function so a natural extension is to apply the structured kernels of Section 3.1. This was already been done with tree kernels by Hardmeier (2011) in the context of SVMs.

Multi-task kernels can also be applied for Quality Estimation in several ways. The model used by Cohn and Specia (2013) for modelling annotator bias can be further extended for settings with dozens or even hundreds of annotators. This is a common setting in crowdsourcing platforms like Amazon's Mechanical Turk[3].

Another plan is to use multi-task kernels to combine different datasets. Quality annotation is usually expensive, requiring post-editing or subjective scoring. Possibilities include combining datasets from different language pairs or different machine translation systems. Available datasets include those used in the WMT12 and WMT13 QE shared tasks (Callison-burch et al., 2012; Bojar et al., 2013) and others (Specia et al., 2009; Specia, 2011; Koponen et al., 2012).

4.2 Question Classification

A Question Classifier is a module that aims to restrict the answer hypotheses generated by a Question Answering system by applying a label to the input question (Li and Roth, 2002; Li and Roth, 2005). This task can be seen as an instance of text classification, where the inputs are usually composed of only one sentence.

Much of previous work in Question Classification largely used SVMs combined with structured kernels. Zhang and Lee (2003) compares String Kernels based on BOW and n-gram representations with the Subset Tree Kernel on constituent trees. Moschitti (2006) show improved results by using the Partial Tree Kernel and dependency trees instead of constituency ones. Bloehdorn and Moschitti (2007) combines a SSTK with different soft matching approaches to encode lexical similarity on tree leaves. The same soft matching idea is used by Croce et al. (2011), but applied to PTKs instead and permitting soft matches between any nodes in each tree (which is sensible when using kernels on dependency trees).

Our work proposes to address this task by employing tree kernels and GPs. Unlike Quality Estimation, this is a classification setting and our purpose is to find if this combination can also improve the state-of-the-art for tasks of this kind. We will use the TREC dataset provided by Li and Roth (2002), which assigns 6000 questions with both a coarse and a fine-grained label.

4.3 Multi-domain Sentiment Analysis

Sentiment Analysis is defined as "the computational treatment of opinion, sentiment and subjectivity in text" (Pang and Lee, 2008). In this proposal, we focus on the specific task of *polarity detection*, where the goal is to label a text as having positive or negative sentiment. State-of-the-art methods for this task use SVMs as the learning algorithm and vary between the feature sets used.

Polarity predictions can be heavily biased on the text domain. Consider the example showed by Turney (2002): the word "unpredictable" usually has a positive meaning in a movie review but a negative one when applied to an automotive review (in a phrase like "unpredictable steering", for instance). One of the first methods to tackle this issue is the Structural Correspondence Learning of Blitzer et al. (2007). Their method uses *pivot* words shared between domains to find correspondencies in words that are not shared.

A previous work that used structured kernels in Sentiment Analysis is the approach of Wu et al. (2009). Their method uses tree kernels on phrase dependency trees and outperforms bag-of-words

[2] Human Translation Error Rate (Snover et al., 2006).

[3] `www.mturk.com`

and word dependency approaches. They also show good results in cross-domain experiments.

We propose to apply GPs with a combination of structured and multi-task kernels for this task. The results showed by Wu et al. (2009) suggest that tree kernels on dependency trees are a good approach but we also plan to employ string kernels on this task. This is because string kernels have demonstrated promising results for text categorization in past work. Also, considering model selection is easily dealt by GPs, we can combine all those kernels in complex and heavily parameterized ways, an unfeasible setting for SVMs. We will use the Multi-Domain Sentiment Dataset (Blitzer et al., 2007), composed of Amazon product reviews in different categories.

4.4 Research Directions

In Section 2.3 we saw how the Bayesian formulation of GPs let us do model selection by maximizing the marginal likelihood. In fact, one of our main research directions in this proposal revolves around this crucial point: because we can easily fit hyperparameters to the data we have much more freedom to use richer kernel parameterizations and kernel combinations. Multi-task kernels are one example where we usually have a large number of hyperparameters because we need to fit all the elements of the coregionalization matrix. This number can get even larger if we have a LMC model, with multiple coregionalization matrices. Structured kernels can also be redefined in a richer way: tree kernels between constituency trees could have multiple decay hyperparameters, one for each POS tag. A more extreme example would be to treat all weights in a string kernel as hyperparameters. Thus, we plan to investigate these possibilities in the context of the three tasks detailed before.

As another research direction we also want to address the issue of scalability. Although GPs already showed promising results they can be slow when compared to other well established methods like SVM. Fortunately there has been a lot of advancements in the field of sparse GPs in the last years and we plan to employ them in our work. A key question is how to combine sparse GPs with the structured kernels we presented before. Although it is perfectly possible to select inducing points using greedy methods, it would be much more interesting to use the pseudo-inputs approach. However, it is not clear how to do that in conjunction with non-vectorial inputs, like the ones we plan to use in structured kernels, and this is a key direction that we also plan to investigate.

4.5 GP Toolkits

Available toolkits for GP modelling include GPML[4] (Rasmussen and Williams, 2006) and GPstuff[5] (Vanhatalo et al., 2013), which are written in Matlab. Our experiments will mainly use GPy[6], an open source toolkit written in Python. It implements models for regression and binary classification, including sparse approximations and many vectorial kernels. We plan to contribute to GPy by implementing the structured kernels of Section 3.1, effectively extending it to a GP framework for NLP.

5 Conclusions and Future Work

In this work we showed a proposal for advancing the state-of-the-art in a number of NLP tasks by combining Gaussian Process with structured and multi-task kernels. Our hypothesis is that highly parameterized kernel combinations allied with the fitting methods provided by GPs will result in better models for these tasks. We also detailed the future plans for experiments, including available datasets and toolkits.

Further research directions that can be explored by this proposal include the use of GPs in different learning settings. Models for ordinal regression (Chu and Ghahramani, 2005) and structured prediction (Altun et al., 2004; Bratières et al., 2013) were already proposed in the GP literature and a natural extension is to apply these models for their corresponding NLP tasks. Another extension is to employ other kinds of kernels. The literature on that subject is quite vast, with many approaches showing promising results.

Acknowledgements

This work was supported by funding from CNPq/Brazil (No. 237999/2012-9) and from the EU FP7-ICT QTLaunchPad project (No. 296347). The author would also like to thank Yahoo for the financial support and the anonymous reviewers for their excellent comments.

[4] www.gaussianprocess.org/gpml/code/matlab

[5] becs.aalto.fi/en/research/bayes/gpstuff

[6] github.com/SheffieldML/GPy

References

Yasemin Altun, Thomas Hofmann, and Alexander J. Smola. 2004. Gaussian Process Classification for Segmenting and Annotating Sequences. In *Proceedings of ICML*, page 8, New York, New York, USA. ACM Press.

Mauricio A. Álvarez, Lorenzo Rosasco, and Neil D. Lawrence. 2012. Kernels for Vector-Valued Functions: a Review. *Foundations and Trends in Machine Learning*, pages 1–37.

Daniel Beck, Kashif Shah, Trevor Cohn, and Lucia Specia. 2013. SHEF-Lite : When Less is More for Translation Quality Estimation. In *Proceedings of WMT13*, pages 337–342.

John Blatz, Erin Fitzgerald, and George Foster. 2004. Confidence estimation for machine translation. In *Proceedings of the 20th Conference on Computational Linguistics*, pages 315–321.

John Blitzer, Mark Dredze, and Fernando Pereira. 2007. Biographies, Bollywood, Boom-boxes and Blenders: Domain Adaptation for Sentiment Classification. In *Proceedings of ACL*.

Stephan Bloehdorn and Alessandro Moschitti. 2007. Exploiting Structure and Semantics for Expressive Text Kernels. In *Proceedings of CIKM*.

Ondřej Bojar, Christian Buck, Chris Callison-Burch, Christian Federmann, Barry Haddow, Philipp Koehn, Christof Monz, Matt Post, Radu Soricut, and Lucia Specia. 2013. Findings of the 2013 Workshop on Statistical Machine Translation. In *Proceedings of WMT13*, pages 1–44.

Sébastien Bratières, Novi Quadrianto, and Zoubin Ghahramani. 2013. Bayesian Structured Prediction using Gaussian Processes. *arXiv:1307.3846*, pages 1–17.

Chris Callison-burch, Philipp Koehn, Christof Monz, Matt Post, Radu Soricut, and Lucia Specia. 2012. Findings of the 2012 Workshop on Statistical Machine Translation. In *Proceedings of 7th Workshop on Statistical Machine Translation*.

Nicola Cancedda, Eric Gaussier, Cyril Goutte, and Jean-Michel Renders. 2003. Word-Sequence Kernels. *The Journal of Machine Learning Research*, 3:1059–1082.

Kai-Wei Chang, Vivek Srikumar, and Dan Roth. 2013. Multi-core Structural SVM Training. In *Proceedings of ECML-PHDD*.

Wei Chu and Zoubin Ghahramani. 2005. Gaussian Processes for Ordinal Regression. *Journal of Machine Learning Research*, 6:1019–1041.

Wei Chu, Zoubin Ghahramani, Francesco Falciani, and David L Wild. 2005. Biomarker discovery in microarray gene expression data with Gaussian processes. *Bioinformatics*, 21(16):3385–93, August.

Trevor Cohn and Lucia Specia. 2013. Modelling Annotator Bias with Multi-task Gaussian Processes: An Application to Machine Translation Quality Estimation. In *Proceedings of ACL*.

Trevor Cohn, Phil Blunsom, and Sharon Goldwater. 2010. Inducing tree-substitution grammars. *The Journal of Machine Learning*, 11:3053–3096.

Michael Collins and Nigel Duffy. 2001. Convolution Kernels for Natural Language. In *Advances in Neural Information Processing Systems*.

Danilo Croce, Alessandro Moschitti, and Roberto Basili. 2011. Structured Lexical Similarity via Convolution Kernels on Dependency Trees. In *Proc. of EMNLP*.

Hal Daumé III. 2007. Frustratingly easy domain adaptation. In *Proceedings of ACL*.

Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. 1990. Indexing by Latent Semantic Analysis. *Journal of the American Society For Information Science*, 41.

Thomas Gärtner, Peter Flach, and Stefan Wrobel. 2003. On Graph Kernels: Hardness Results and Efficient Alternatives. *LNAI*, 2777:129–143.

Sharon Goldwater, Thomas L. Griffiths, and Mark Johnson. 2009. A Bayesian framework for word segmentation: Exploring the effects of context. *Cognition*, 112(1):21–54, July.

Christian Hardmeier, Joakim Nivre, and Jörg Tiedemann. 2012. Tree Kernels for Machine Translation Quality Estimation. In *Proceedings of WMT12*, number 2011, pages 109–113.

Christian Hardmeier. 2011. Improving Machine Translation Quality Prediction with Syntactic Tree Kernels. In *Proceedings of EAMT*, number May.

David Haussler. 1999. Convolution Kernels on Discrete Structures. Technical report.

James Hensman, Nicolò Fusi, and Neil D. Lawrence. 2013. Gaussian Processes for Big Data. In *Proceedings of UAI*.

Matt Hoffman, David M. Blei, Chong Wang, and John Paisley. 2013. Stochastic Variational Inference. *The Journal of Machine Learning Research*.

Tommi Jaakkola, Mark Diekhans, and David Haussler. 2000. A discriminative framework for detecting remote protein homologies. *Journal of Computational Biology*, 7:95–114.

Richard Johansson and Alessandro Moschitti. 2013. Relational Features in Fine-Grained Opinion Analysis. *Computational Linguistics*, 39(3):473–509.

Jonathan Ko, Daniel J. Klein, Dieter Fox, and Dirk Haehnel. 2007. Gaussian Processes and Reinforcement Learning for Identification and Control of an Autonomous Blimp. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 742–747. Ieee, April.

Maarit Koponen, Wilker Aziz, Luciana Ramos, and Lucia Specia. 2012. Post-editing time as a measure of cognitive effort. In *Proceedings of WPTP*.

Xin Li and Dan Roth. 2002. Learning question classifiers. In *Proceedings of COLING*, volume 1, pages 1–7.

Xin Li and Dan Roth. 2005. Learning Question Classifiers: the Role of Semantic Information. *Natural Language Engineering*, 1(1).

Huma Lodhi, Craig Saunders, John Shawe-Taylor, Nello Cristianini, and Chris Watkins. 2002. Text Classification using String Kernels. *The Journal of Machine Learning Research*, 2:419–444.

Thomas P. Minka. 2001. *A family of algorithms for approximate Bayesian inference*. Ph.D. thesis.

Alessandro Moschitti. 2006. Efficient Convolution Kernels for Dependency and Constituent Syntactic Trees. In *Proceedings of ECML*.

Kevin P. Murphy. 2012. *Machine Learning: a Probabilistic Perspective*.

Radford M. Neal. 1998. Regression and Classification Using Gaussian Process Priors. *Bayesian Statistics*, 6.

Bo Pang and Lillian Lee. 2008. Opinion Mining and Sentiment Analysis. *Foundations and Trends in Information Retrieval*, 2(1–2):1–135.

Verónica Pérez-Rosas and Rada Mihalcea. 2013. Sentiment Analysis of Online Spoken Reviews. In *Proceedings of Interspeech*.

Barbara Plank and Alessandro Moschitti. 2013. Embedding Semantic Similarity in Tree Kernels for Domain Adaptation of Relation Extraction. In *Proceedings of ACL*, pages 1498–1507.

Tamara Polajnar, Simon Rogers, and Mark Girolami. 2011. Protein interaction detection in sentences via Gaussian Processes: a preliminary evaluation. *International Journal of Data Mining and Bioinformatics*, 5(1):52–72, January.

Daniel Preotiuc-Pietro and Trevor Cohn. 2013. A temporal model of text periodicities using Gaussian Processes. In *Proceedings of EMNLP*.

Carl Edward Rasmussen and Christopher K. I. Williams. 2006. *Gaussian processes for machine learning*, volume 1. MIT Press Cambridge.

Jaakko Riihimäki, Pasi Jylänki, and Aki Vehtari. 2013. Nested Expectation Propagation for Gaussian Process Classification with a Multinomial Probit Likelihood. *Journal of Machine Learning Research*, 14:75–109.

Anton Schwaighofer, Marian Grigoras, Volker Tresp, and Clemens Hoffmann. 2004. GPPS: A Gaussian Process Positioning System for Cellular Networks. In *Proceedings of NIPS*.

Matthias Seeger, Christopher K. I. Williams, and Neil D. Lawrence. 2003. Fast Forward Selection to Speed Up Sparse Gaussian Process Regression. In *Proceedings of AISTATS*.

John Shawe-Taylor and Nello Cristianini. 2004. *Kernel methods for pattern analysis*. Cambridge.

Fabian H. Sinz, Joaquin Quiñonero Candela, Gökhan H. Bakır, Carl E. Rasmussen, and Matthias O. Franz. 2004. Learning Depth from Stereo. *Pattern Recognition*, pages 1–8.

Edward Snelson and Zoubin Ghahramani. 2006. Sparse Gaussian Processes using Pseudo-inputs. In *Proceedings of NIPS*.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proceedings of AMTA*.

Lucia Specia and Atefeh Farzindar. 2010. Estimating machine translation post-editing effort with HTER. In *Proceedings of AMTA Workshop Bringing MT to the User: MT Research and the Translation Industry.*

Lucia Specia, Nicola Cancedda, Marc Dymetman, Marco Turchi, and Nello Cristianini. 2009. Estimating the sentence-level quality of machine translation systems. In *Proceedings of EAMT*, pages 28–35.

Lucia Specia. 2011. Exploiting objective annotations for measuring translation post-editing effort. In *Proceedings of EAMT*.

Michalis K. Titsias. 2009. Variational Learning of Inducing Variables in Sparse Gaussian Processes. In *Proceedings of AISTATS*, volume 5, pages 567–574.

Peter D. Turney. 2002. Thumbs Up or Thumbs Down?: Semantic Orientation Applied to Unsupervised Classification of Reviews. In *Proceedings of ACL*, number July, pages 417–424.

Jarno Vanhatalo, Jaakko Riihimäki, Jouni Hartikainen, Pasi Jylänki, Ville Tolvanen, and Aki Vehtari. 2013. GPstuff: Bayesian Modeling with Gaussian Processes. *The Journal of Machine Learning Research*, 14:1175–1179.

S. V. N. Vishwanathan, Nicol N. Schraudolph, Risi Kondor, and Karsten M. Borgwardt. 2010. Graph Kernels. *Journal of Machine Learning Research*, 11:1201–1242.

Christopher K. I. Williams and David Barber. 1998. Bayesian Classification with Gaussian Processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(12):1342–1351.

Yuanbin Wu, Qi Zhang, Xuanjing Huang, and Lide Wu. 2009. Phrase Dependency Parsing for Opinion Mining. In *Proceedings of EMNLP*, pages 1533–1541.

Dell Zhang and Wee Sun Lee. 2003. Question classification using support vector machines. In *Proceedings of SIGIR*, New York, New York, USA. ACM Press.