Evolving Gaussian Process kernels for translation editing effort estimation

Ibai Roman[1], Roberto Santana[1], Alexander Mendiburu[1], and Jose A. Lozano[1,2]

[1]University of the Basque Country (UPV/EHU), San Sebastian, Spain, {ibai.roman, roberto.santana, alexander.mendiburu, ja.lozano}@ehu.eus
[2]Basque Center for Applied Mathematics (BCAM), Bilbao, Spain

Abstract

In many Natural Language Processing problems the combination of machine learning and optimization techniques is essential. One of these problems is estimating the effort required to improve, under direct human supervision, a text that has been translated using a machine translation method. Recent developments in this area have shown that Gaussian Processes can be accurate for post-editing effort prediction. However, the Gaussian Process kernel has to be chosen in advance, and this choice influences the quality of the prediction. In this paper, we propose a Genetic Programming algorithm to evolve kernels for Gaussian Processes. We show that the combination of evolutionary optimization and Gaussian Processes removes the need for a-priori specification of the kernel choice, and achieves predictions that, in many cases, outperform those obtained with fixed kernels.

Keywords— Evolutionary search, Gaussian Processes, Genetic Programming, Kernel Selection, Quality Estimation

1 Introduction

Gaussian Processes (GP) [30] have been extensively applied for function approximation. A GP is a model that lies on strong Bayesian inference foundations and can be updated when new evidence is available. In comparison to other regression methods, a GP provides not only a prediction of a given function but also estimates the uncertainty of the predictions. A GP requires a kernel function to be defined and adjusts its hyperparameters to the data. Usually, the kernel function is specified a-priori, and the search for the hyperparameters is posed as an optimization process. This optimization is an essential component of a GP model since it highly influences the quality of the approximation.

Within Natural Language Processing (NLP) literature, GPs have been applied for text classification [27], modeling periodic distributions of words over time [29], emotion or sentiment classification [1] and Quality Estimation (QE) [6, 34]. In these works, the choice of the GP kernel is made a-priori. For instance, for QE regression the most common kernel is the Squared Exponential (SE) or Radial Basis Function (RBF) kernel [6, 34]. For modeling text periodicities, the periodic (PER) and the Periodic Spikes (PS) kernels have been proposed as a sensible way to capture the periodicities of the function [29]. In addition to SE, two Matern kernels, Matern 32 (M32) and Matern 52 (M52), were evaluated in [1] for emotion analysis. There is a repertoire of kernel functions available in the literature [30, 10, 14], and selecting the best kernel for each problem requires an expert knowledge of the domain.

In this paper we propose to simultaneously optimize the kernel function itself, along with its hyperparameters. For this purpose, we propose the use of Genetic Programming (GenProg) [19]. In this work, kernels are encoded as a set of genes and optimized though an evolutionary process. As opposed to other proposals in the GP literature, we learn the initial kernels from scratch, without seeding human-designed kernels, allowing us to derive kernels that are not constrained by the prior knowledge while at the same time being optimized for the desired objective.

We focus on a practical NLP regression problem that is related to automatic translation of texts. In this domain, post-editing work is frequently required, and an estimation of the cost of the editing process (in terms of time, effort, and editing distance) is essential. In [36], Specia investigated the question of translation QE from different perspectives. In particular, a number of metrics describing translation quality were proposed and Support Vector Machine (SVM) regression models [39] were used to predict them from a set of pre-defined features.

Instead of manually defining the features, our approach relies on sentence embeddings, vector representations of the source and the automatically translated texts to predict the post-editing effort that leads to the final text. This allows a fully automated approach, where there is no need to to extract the features from the sentences, nor learn the kernel function. We investigate several methods to construct the sentence embeddings along with the evolution of the GP kernels, measuring the joint effect that these questions have in QE performance.

The remainder of the paper is organized as follows: The next section introduces the main concepts related to GP regression. Section 3 presents the addressed problem in the context of QA. The GenProg approach to evolve kernel functions is presented in Section 4 and a review on related work is provided. We describe the experimental framework used to validate our algorithm, along with the numerical results in Section 5. The conclusions of the paper and discussion of future work are presented in Section 6.

2 Gaussian Process Regression

A GP is a stochastic process, defined by a collection of random variables, any finite number of which have a joint Gaussian distribution [30]. A GP can be interpreted as a distribution over functions, and each sample of a GP as a function.

GPs can be completely defined by a mean function $m(\mathbf{x})$ and a covariance function, which depends on a kernel $k(\mathbf{x}, \mathbf{x}')$. Given that, a GP can be expressed as follows:

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

where we assume that $\mathbf{x} \in \mathbb{R}^d$. We also consider an a-priori equal-to-zero mean function ($m(\mathbf{x}) = 0$) to focus on the kernel search problem.

A GP can be used for regression by obtaining its posterior distribution given some (training) data. The joint distribution between the training outputs $\mathbf{f} = (f_1, f_2, ..., f_n)$ (where $f_i \in \mathbb{R}$, $i \in \{1, ..., n\}$ and $n \in \mathbb{N}$) and the test outputs $\mathbf{f}_* = (f_{n+1}, f_{n+2}, ..., f_{n+n_*})$ is given by:

$$\begin{bmatrix} \mathbf{f} \\ \mathbf{f}_* \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} K(X,X) & K(X,X_*) \\ K(X_*,X) & K(X_*,X_*) \end{bmatrix}\right) \tag{2}$$

where $N(\mu, \Sigma)$ is a multivariate Gaussian distribution, $X = (\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n)$ ($\mathbf{x}_i \in \mathbb{R}^d$, $i \in \{1, ..., n\}$ and $n \in \mathbb{N}$) corresponds to the training inputs and $X_* = (\mathbf{x}_{n+1}, ..., \mathbf{x}_{n+n_*})$ to the test inputs. $K(X, X_*)$ denotes the $n \times n_*$ matrix of the covariances evaluated for all the (X, X_*) pairs.

The predictive Gaussian distribution can be found by obtaining the conditional distribution given the training data and the test inputs:

$$\begin{aligned} \mathbf{f}_*|X_*, X, \mathbf{f} &\sim \mathcal{N}(\hat{M}(X_*), \hat{K}(X_*, X_*)) \\ \hat{M}(X_*) &= K(X_*, X)K(X,X)^{-1}\mathbf{f} \\ \hat{K}(X_*, X_*) &= K(X_*, X_*) - K(X_*, X)K(X,X)^{-1}K(X, X_*) \end{aligned} \tag{3}$$

2.1 Kernel functions

GP models use a kernel to define the covariance between any two function values [10]:

$$cov\,(f(\mathbf{x}), f(\mathbf{x}')) = k(\mathbf{x}, \mathbf{x}') \tag{4}$$

The kernel functions used in GPs are positive-definite kernels. According to Mercer's Theorem [23], any PSD kernel can be represented as an inner product in some Hilbert Space.

The best known kernels in GP literature are translation invariant, often referred to as stationary kernels. Among them, we focus on isotropic kernels, where the covariance function depends on the norm:

$$k(\mathbf{x}, \mathbf{x}') = \hat{k}(r) \text{ where } r = \frac{1}{\theta_l} \|\mathbf{x} - \mathbf{x}'\| \tag{5}$$

where θ_l is the length scale hyperparameter and $\hat{k}$ a function that guarantees that the kernel is PSD.

Table 1 shows four well-known kernels that have been previously applied to NLP applications [1, 36, 29]. The SE kernel, described as k_{SE} in the table, is known to capture the smoothness property of the objective functions. Matern class kernels, are denoted as $\hat{k}_{M32}$ and $\hat{k}_{M52}$ in the table, while the Periodic kernel is shown as $\hat{k}_{PER}$.

Kernel function expressions	
Squared Exp.	$\hat{k}_{SE}(r) = \theta_0^2 \; exp\left(-\frac{1}{2}r^2\right)$
Matern 32	$\hat{k}_{M32}(r) = \theta_0^2 \left(1 + \sqrt{3}r\right) exp\left(-\sqrt{3}r\right)$
Matern 52	$\hat{k}_{M52}(r) = \theta_0^2 \left(1 + \sqrt{5}r + \frac{5}{3}r^2\right) exp\left(-\sqrt{5}r\right)$
Periodic	$\hat{k}_{PER}(r) = \theta_0^2 \exp\left(-\frac{2\sin^2(\pi r)}{\theta_p^2}\right)$

Table 1: Well-known kernel functions. θ_0 and θ_p are the kernel hyperparameters called amplitude and period respectively.

2.2 Hyperparameter Optimization

The choice of the kernel function and its hyperparameters has a critical influence on the behavior of the model, and it is crucial to achieve good results in NLP applications of GPs [1]. This selection has been usually made by choosing one kernel a-priori, and then adjusting the hyperparameters of the kernel function so to optimize a given metric for the data. The most common approach is to find the hyperparameter set that maximizes the log marginal likelihood (LML):

$$log\, p\left(\mathbf{f}|X, \boldsymbol{\theta}, K\right) = -\frac{1}{2}\mathbf{f}^T K(X,X)^{-1}\mathbf{f} - \frac{1}{2} log\, |K(X,X)| - \frac{n}{2}\, log\, 2\pi \qquad (6)$$

where $\boldsymbol{\theta}$ is the set of hyperparameters of the kernel and n is the length of X.

3 Quality estimation and feature extraction

Following [36], we use three different metrics for QE. We assume the existence of an original text in the source language which is divided into sentences. For each sentence, an automatic translation to the target language is available. These translations are the subject of post-editing work. The metrics considered are, *post-editing effort*, *Human Translation Edit Rate* (HTER) and *post-edit time*.

In [36], translators were asked to post-edit each sentence and score the *post-editing effort* according to the following options:

1. Requires complete retranslation.

2. Requires some retranslation, but post-editing is still quicker than retranslation.

3. Very little post-editing needed.

4. Fit for purpose.

Another metric used to evaluate the translation quality is the edit distance between the automatic translation and its post-edited version. This is computed using the *Human Translation Edit Rate* (HTER) [35]. HTER is defined as $HTER = \frac{e}{pe_w}$, where e is the number of edits, which can be standard insertion, deletion and substitution of single words, and shifting of word sequences. pe_w is the number of words in the sentence.

Finally, the *post-edit time* was computed in [36] using the average number of seconds required by two translators to post-edit each word in the sentence. It is the number of seconds to post-edit the sentence, normalized by the number of words in that sentence.

For more details on the ways these metrics were defined, [36] can be consulted.

3.1 Sentence embeddings

There are a number of approaches to extract relevant information for QE [4]. In this domain, feature engineering to obtain informative features can be very labor-intensive [37]. In [36], 80 shallow and machine-translation independent features were extracted from the source sentences, their corresponding translations, and monolingual and parallel corpora. A selected set of these features, comprising only 17 features, is used in [37]. Syntactic information represented in tree fragments that contain complete grammar rules are used as feature representation in [2].

Here, we focus on sentence embeddings [24], a common text representation for NLP tasks, although less investigated for QE. We use word embedding dictionaries for the source and target languages (before post-edition). For each sentence, in each language, we compute the embedding representation of all the words. Words missing in the dictionary are assigned a zero-vector representation. In each corpus, for each sentence, a function that combines the embedding representations of all the words in a single vector (e.g., mean of the word vectors) is computed to generate the sentence embedding. Finally, for each pair of original and translated sentences, their sentence embeddings are concatenated to produce the vector representation that is used for QE.

We examine two questions related to the embedding representation. The embedding dimension and the way sentence embeddings are computed. In addition to the commonly applied *mean* function, which computes the mean of all word embeddings in a sentence, we use another two other functions: the maximum (*max*) of all word embeddings (maximum value of each embedding component across all word embeddings), and the standard deviation (*std*) of word embeddings. We hypothesize that variations in the words as captured by the maximum and standard deviation may provide a clue as to the difficulty of the translation.

4 Kernel function search

In this work, we automatically search for new kernel functions in order to better predict the translation effort. Specifically, our goal is to find the optimal $\hat{k}(r)$ as in Equation (5).

To guide this search, we propose using Genetic Programming (GenProg) [19]. In GenProg, computer programs are encoded as solutions using an evolvable representation. At each iteration, solutions are recombined and selected according to the program performance, until an optimal solution is found or a stop criterion is satisfied. In our case, the GP kernel function is the program that is encoded into an expression tree and its performance is evaluated in the context of the translation quality evaluation task.

We define a strongly-typed grammar [25] that specifies the possible combinations these kernels can be composed of, by breaking down the mathematical expressions present in the well-known kernels of Table 1 into basic operations (multiplication, square root, ...).

To randomly generate kernel expression trees that conform the initial population, we propose a grow method based on the work done in [19]. The initial expression tress are created by a recursive process where, at each step, a random terminal or operator is added.

GenProg also needs perturbation or variation methods to be defined in order to modify previous solutions to obtain new ones. A crossover operator, which combines two kernel functions to generate a new one that keeps some of the features of its parents, and a mutation operator, which introduces slight modifications to the original kernel to obtain a new individual are used. We propose a crossover operator that randomly selects a subtree from each kernel and combines them with the sum or the product operator. Furthermore, the mutation operator replaces, shrinks or inserts an elementary mathematical expression at a random position in the tree in a type-safe manner.

As in [11], we use the Bayesian Information Criterion (BIC) [32] as a quality metric for each kernel. This is the fitness function of our GenProg algorithm. As can be seen in Equation (7), this metric is similar to the LML, but a regularization term that penalizes the complexity of the kernels is added:

$$BIC(k_i) = -2\ log\ p\left(\mathbf{f}|X, k_i, \boldsymbol{\theta}_{i,best}\right) + q\ log\ n \tag{7}$$

where q is the number of hyperparameters of the kernel and n is the number of data points in X. $\boldsymbol{\theta}_{i,best}$ is the best hyperparameter set for the kernel k_i according to a given metric.

In contrast to other GenProg applications, the solutions in our approach do not encode all the necessary information to be evaluated. The optimal values of the hyperparameters, according to the LML, have to be determined. Thus, the performance of the solutions depends on the results of the hyperparameter optimization. Both search procedures, the selection of the best hyperparameters for each kernel and the selection of the best kernel given these hyperparameters, are illustrated in Figure 1.

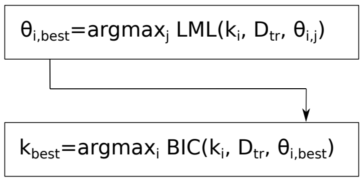

$\theta_{i,best}$=argmax$_j$ LML(k_i, D_{tr}, $\theta_{i,j}$)
k_{best}=argmax$_i$ BIC(k_i, D_{tr}, $\theta_{i,best}$)

Figure 1: Two nested search procedures: The selection of the best hyperparameters for each kernel is made according to LML and the selection of the best kernel according to the BIC.

In this paper, the hyperparameters are optimized by means of *Powell*'s local search algorithm [28]. As this algorithm is not bounded, the search space has to be constrained by penalizing non-feasible hyperparameter sets. Besides, as the function to optimize might be multi-modal, a multi-start approach was used, performing a random restart every time the stopping criteria of *Powell*'s algorithm are met, and getting the best overall result. During this hyperparameter search, a maximum number of 150 evaluations of the LML were allowed.

Note that, as a result of the inclusion of the randomized restarts, the hyperparameters found for a certain kernel in two independent evaluations may not be the same. In fact, this implies that the fitness function optimized by the GP algorithm is stochastic.

4.1 Related work

GPs are particularly suited to model uncertainty in the predictions and allow accurate prediction under noisy conditions. As such, there are diverse scenarios in which GP can be applied to NLP tasks [5]. In [34], they are used for feature selection for QE. Another property of GP, the possibility of extending them to model correlated tasks using multi-task kernels, is used in [6] to jointly learn a series of annotator and metadata specific models.

The most frequently used kernel for NLP tasks is the SE kernel. However, other kernels have shown a better performance than SE in specific tasks. In [29], frequencies of tweets are modeled using GP kernels specifically suited to capture periodicities. In the same paper, the PER and the PS kernels are shown to outperform non-periodic kernels and capture different periodic patterns. In [1], three different kernels are compared: the SE and two Matern kernels. The Matern kernels are reported to produce better results than SE. In addition to numerical kernels, structural-kernels (e.g., tree-kernels) have been also combined with GP. In [2], they are applied to emotion analysis and quality estimation.

A common characteristic of GP applications to NLP is that the choice of the GP kernel has to be made a-priori and does not depend on the quality of the function approximation. The focus is placed on hyperparameter optimization. However, research on evolutionary algorithms has shown that it is also possible to explore the space of kernel functions beyond the hyperparameter optimization. In the GP literature this has been done by combining known kernels [20, 11, 22]. Kernels have also been evolved for Support Vector Machines (SVMs) [17, 13, 8, 38, 9, 18] and Relevance Vector Machines (RVMs) [3]. Some of the SVM approaches are also based on combining the well-known kernels [38, 9], although in some other works the kernels are learned from simple mathematical expressions [17, 13, 8, 18].

In contrast to other works in the GP field, our approach is to evolve kernels from scratch. This method does not rely on previously proposed kernels, and thus, new kernels may naturally arise.

Word embeddings [24] are extensively applied to NLP tasks [21]. The usual approach when combining embeddings from words in a sentence is to compute the average. This is the procedure used in [1], where 100-dimensional Glove embeddings [26] are the representation of choice for mapping texts to emotion scores using GP regression. Word-embeddings have been also used together with GenProg in [31], but with the goal of solving the analogy task problem. This is a completely different problem and the solution presented in [31] does not consider the optimization of GPs or kernels.

5 Experiments

The objectives of the experimentation are twofold. On the one hand, we would like to know if the evolution of GP kernels can improve the results of well-known stationary kernels in translation post-editing effort estimation though sentence embeddings. On the other hand, we would also like to investigate the best solution to aggregate the word vectors to conform these sentence embeddings.

5.1 Datasets and embeddings

Our experiments consist of learning a GP regressor based on some combination of source and target embeddings, and use this combined representation to predict a particular metric. To carry out these experiment, we use the datasets originally proposed in [36]:

1. en-es news-test2010: First 1000 English news sentences and their translations into Spanish using Moses software. 800 sentences were used for training and 200 for testing.

2. fr-en news-test2009: 2525 French news sentences and their Moses translations into English. 800 sentences were used for training and the remaining ones for testing.

Punctuations were removed from the sentences, the text was tokenized, and also case ignored. For the "HTER" metric, an equal cost was used for all edits [36].

For each source language, we created two embedding representations with different vector dimensions. For the first dataset, we use English Glove embeddings [26] of dimensions 50 and 300. To represent sentences in the target language, we used Spanish embeddings of size 100 computed from the Spanish CoNLL17 corpus and available from the NLPL word embeddings repository[1]. For the second dataset, we used French embeddings of sizes 300 and 52. The 300-dimensional embeddings[2] were trained on Common Crawl and Wikipedia using fastText [15]. The 52-dimensional embeddings[3] were trained on tweets [7]. The 100-dimensional embeddings for the target language were those provided by Glove.

5.2 Experimental set-up

Learning a GP regressor implies optimizing the hyperparameters of the kernel. Since the local optimizer used for that optimization is a stochastic process, we run 30 executions of the fitting process using the training data. For traditional Gaussian kernels, this amounts to learning 30 different hyperparameter configurations of the same kernel (e.g., of SE). For the evolved kernels, this means obtaining 30 different kernels. Each regressor is then used to predict the corresponding metric on test data. Finally, for each kernel, the quality of the prediction is measured by computing the root mean squared error (RMSE) between the known true metric values and the predictions. In order to compare the different variants of the algorithms, we analyze the distribution of the RMSE and the mean RMSE values.

5.3 Results

In our first experiment, we compare the classical kernel models to the evolved kernels introduced in this paper. We also evaluate the effect of using embeddings of different dimensions to represent the source language text. Figure 2 shows the results. In the figure, *orig_vw_len* refers to the dimensions of the embeddings of the source data.

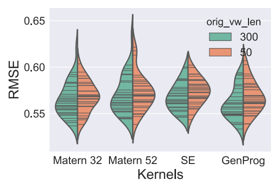
The first remarkable fact in Figure 2 is that GenProg kernels show a similar variance comparing to the classical kernels. This can be a sign of convergence at least in performance. On the other hand, it can be seen that the dimension of the source embeddings has a higher effect in the quality of the predictions than the type of kernel used. This is particularly evident for the *fr-en news-test2009* dataset for which all distribution shapes are clearly asymmetric, with embeddings of size 52 producing lower errors. It is important to take into consideration that, for this dataset, embeddings are different not only in terms of dimension but they also have been actually produced using different corpora and methods. This is not the case for *en-es news-test2010* dataset that uses 50- and 300-dimensional Glove embeddings. This fact may explain why differences for this dataset, particularly for the “time” and “HTER” metrics, are so small.

In terms of the class of kernels used, differences are only visible for the *fr-en news-test2009* and the “score” metric, for which the use of evolved kernels

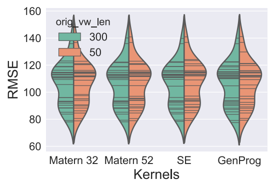
[1]`http://vectors.nlpl.eu/repository/`

[2]`https://fasttext.cc/docs/en/crawl-vectors.html`

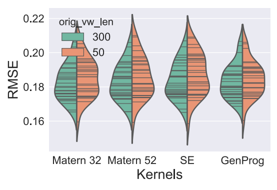
[3]`https://www.spinningbytes.com/resources/wordembeddings/`



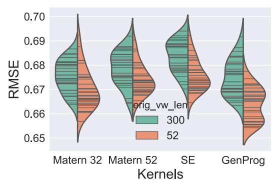
(a) score in *en-es*



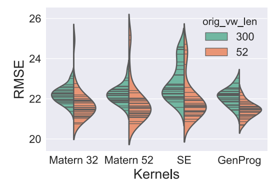
(b) time in *en-es*



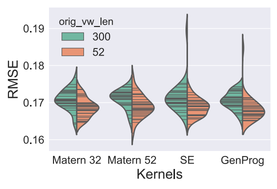
(c) HTER in *en-es*



(d) score in *fr-en*



(e) time in *fr-en*



(f) HTER in *fr-en*

Figure 2: Comparison of the kernel models. For all kernels, the *mean* sentence embedding has been used.

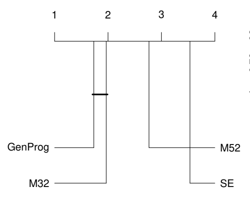
	Matern 32		Matern 52		SE		GenProg	
en-es	50	300	50	300	50	300	50	300
score	0.5686	0.5655	0.5731	0.5705	0.5742	0.5678	0.5679	**0.5609**
time	106.3459	105.5226	106.1876	**105.3945**	106.4699	105.4241	106.6932	105.4191
HTER	0.1851	**0.1827**	0.1857	0.1830	0.1872	0.1837	0.1846	0.1835
fr-en	52	300	52	300	52	300	52	300
score	0.6703	0.6755	0.6741	0.6786	0.6758	0.6834	**0.6638**	0.6748
time	21.7082	22.1893	21.8140	22.2194	21.9587	22.8207	**21.4360**	22.0969
HTER	0.1686	0.1708	0.1691	0.1710	0.1687	0.1714	**0.1684**	0.1706

Table 2: Comparison between the GP models (mean RMSE from 30 executions). The best results are shown in bold.

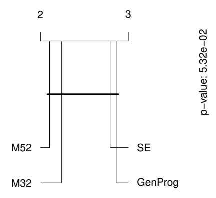
produces better predictions with a higher probability. The improvement over the Matern 52 and SE kernels is particularly clear. The results of the second experiment are summarized in Table 2.

We conducted a statistical test to assess the existence of significant differences among the kernels. For each dataset and metric, we applied Friedman's test [12] and we found significant differences in most comparisons (p-values can be seen in the figures). Then, for each configuration, we applied a post-hoc test based on Friedman's test, and adjusted its results with Shaffer's correction [33].

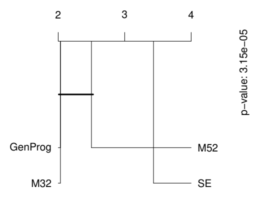
The results of the statistical tests are shown in Figure 3. The results confirm a coherent pattern where GenProg is the best performing kernel for most of the configurations. Particularly, in *fr-en news-test2009* dataset, it achieves significantly better results that the classical kernels for the "time" metric. However, according to this test, it can also be appreciated that for the rest of the configurations the differences between GenProg and M32 are not significant. In evaluating these results it is important to take into account that the kernels produced by GenProg have been generated completely from scratch, with no prior knowledge of the existing kernels. The algorithm is able evolve a well performing kernel starting from elementary mathematical components.



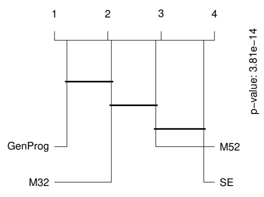
(a) score in *en-es*



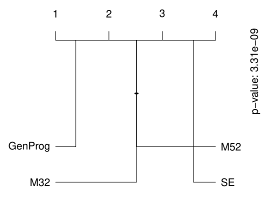
(b) time in *en-es*



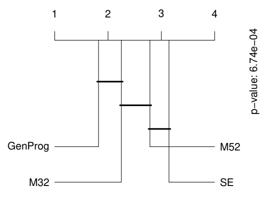
(c) HTER in *en-es*



(d) score in *fr-en*



(e) time in *fr-en*

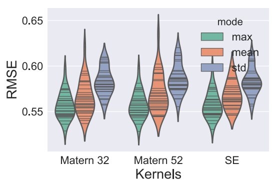


(f) HTER in *fr-en*

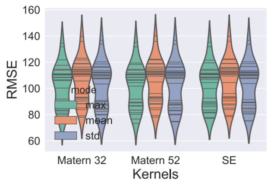
Figure 3: Critical difference diagrams. The kernels are ordered following the results in their ranking. The metrics with no significant differences among them are matched with a straight line. On the top, the results for the *en-es news-test2010* dataset with 50-dimensional word-vectors can be found. On the bottom of the figure, the results for *fr-en news-test2009* with 52-dimensional word-vectors are shown. For all kernels, the *mean* sentence embedding has been used.

Another important question is whether the different ways to compute sentence embeddings influences the quality of the prediction. In the next experiment, we investigate this issue. Figure 4 shows the violin plots [16] for RMSE as computed in the test set with the *max*, *mean*, and *std* functions of the word-vectors used. In these experiments, the 300-dimensional embeddings have been used. Each violin plot represents a histogram smoothened using a kernel density with Normal kernel. RMSE values for each of the 30 executions are shown as black vars.

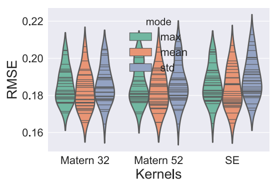
The analysis of Figure 4 reveals that there are not major differences in sentence embeddings for the "HTER" metric. However, for the "score" metric, *max* embeddings produce smaller errors in the predictions than the other two sentence embedding functions for the two datasets. This effect is more pronounced for the time metric in the *fr-en news-test2009* dataset, for which the *max* embeddings produce better predictions.



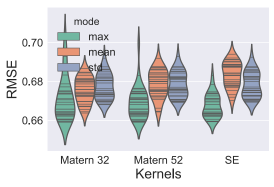
(a) score in *en-es*



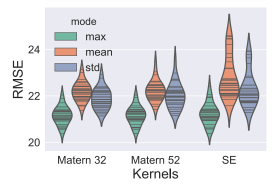
(b) time in *en-es*



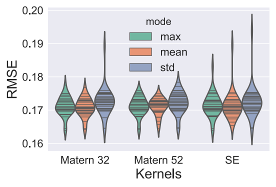
(c) HTER in *en-es*



(d) score in *fr-en*



(e) time in *fr-en*



(f) HTER in *fr-en*

Figure 4: Comparison of the three sentence embeddings: *max*, *mean*, and *std* on the two datasets, and three effort estimation metrics. For all kernels, 300-dimensional embeddings have been used.

6 Conclusions

Quality estimation of automatic translation has grown in interest in recent years. There are many factors that influence the quality of the final estimation. In particular, the type of text representation used and the class of regressor models are very relevant questions. In this paper, using different QE metrics, we have investigated how the joint determination of these factors influences the final QE. We have focused on GP models, evaluating evolutionary generated kernels, which can be more flexible than classical kernels.

Our results show that GenProg is the best performing kernel for most of the configurations, although in most cases no significant differences were found. For one particular metric, the "score" metric, evolved kernels produce better results than simpler classical models on average in both datasets. Moreover, in *fr-en news-test2009* dataset, significant differences with the classical kernels were found for the "time" metric. If the effort needed to train the model is not an issue, the GenProg approach is the recommended option. Alternatively, classical kernels, particularly the M32 kernel, are a good choice if a faster prediction is needed.

In terms of the sentence embeddings used, *max* embedding showed a slight advantage over extensively used *mean* embeddings. However, this effect seems to depend on the particular metric approximated since for the "HTER" metric we did not observe any difference between the sentence embeddings used. This may indicate that word embeddings in general are not a good feature representation for "HTER". The choice of the dimensionality of word embeddings produced a more marked effect in the quality of the predictions.

As future work we consider the further evaluation of the GP kernels using other features and more sophisticated approaches to compute sentence embeddings.

Acknowledgments

The research presented in this paper has been partially supported by the Basque Government (ELKARTEK programs), and Spanish Ministry of Economy and Competitiveness MINECO (project TIN2016-78365-R). Jose A. Lozano is also supported by BERC 2018-2021 (Basque Government), and Severo Ochoa Program SEV-2017-0718 (Spanish Ministry of Economy, Industry and Competitiveness).

References

[1] Beck, D.: Modelling Representation Noise in Emotion Analysis using Gaussian Processes. In: Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers). pp. 140–145. Asian Federation of Natural Language Processing, Taipei, Taiwan (Nov 2017), `https://www.aclweb.org/anthology/I17-2024`

[2] Beck, D., Cohn, T., Hardmeier, C., Specia, L.: Learning Structural Kernels for Natural Language Processing. Transactions of the Association for Computational Linguistics **3**, 461–473 (Dec 2015). https://doi.org/10.1162/tacl_a_00151, `https://doi.org/10.1162/tacl_a_00151`

[3] Bing, W., Wen-qiong, Z., Ling, C., Jia-hong, L.: A GP-based kernel construction and optimization method for RVM. In: 2010 The 2nd International Conference on Computer and Automation Engineering (ICCAE). vol. 4, pp. 419–423 (Feb 2010). https://doi.org/10.1109/ICCAE.2010.5451646

[4] Callison-Burch, C., Koehn, P., Monz, C., Zaidan, O.F.: Findings of the 2011 Workshop on Statistical Machine Translation. In: Proceedings of the Sixth Workshop on Statistical Machine Translation. pp. 22–64. WMT '11, Association for Computational Linguistics, Stroudsburg, PA, USA (2011), `http://dl.acm.org/citation.cfm?id=2132960.2132964`, event-place: Edinburgh, Scotland

[5] Cohn, T., Preotiuc-Pietro, D., Lawrence, N.: Gaussian processes for natural language processing. In: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics: Tutorials. pp. 1–3 (2014)

[6] Cohn, T., Specia, L.: Modelling annotator bias with multi-task gaussian processes: An application to machine translation quality estimation. In: Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). vol. 1, pp. 32–42 (2013)

[7] Deriu, J., Lucchi, A., De Luca, V., Severyn, A., Müller, S., Cieliebak, M., Hofmann, T., Jaggi, M.: Leveraging Large Amounts of Weakly Supervised Data for Multi-Language Sentiment Classification. In: Proceedings of the 26th International Conference on World Wide Web. pp. 1045–1052. WWW '17, International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, Switzerland (2017). https://doi.org/10.1145/3038912.3052611, https://doi.org/10.1145/3038912.3052611, event-place: Perth, Australia

[8] Diosan, L., Rogozan, A., Pecuchet, J.P.: Evolving kernel functions for SVMs by genetic programming. In: Sixth International Conference on Machine Learning and Applications (ICMLA 2007). pp. 19–24 (2007). https://doi.org/10.1109/ICMLA.2007.70

[9] Dioşan, L., Rogozan, A., Pecuchet, J.P.: Improving classification performance of Support Vector Machine by genetically optimising kernel shape and hyper-parameters. Applied Intelligence **36**(2), 280–294 (Mar 2012). https://doi.org/10.1007/s10489-010-0260-1, https://link.springer.com/article/10.1007/s10489-010-0260-1

[10] Duvenaud, D.: Automatic model construction with Gaussian processes. Thesis, University of Cambridge (2014), http://www.repository.cam.ac.uk/handle/1810/247281

[11] Duvenaud, D., Lloyd, J., Grosse, R., Tenenbaum, J., Zoubin, G.: Structure Discovery in Nonparametric Regression through Compositional Kernel Search. In: Proceedings of The 30th International Conference on Machine Learning. pp. 1166–1174 (2013), http://jmlr.org/proceedings/papers/v28/duvenaud13.html

[12] Friedman, M.: The use of ranks to avoid the assumption of normality implicit in the analysis of variance. Journal of the american statistical association **32**(200), 675–701 (1937)

[13] Gagné, C., Schoenauer, M., Sebag, M., Tomassini, M.: Genetic Programming for Kernel-Based Learning with Co-evolving Subsets Selection. In: Parallel Problem Solving from Nature - PPSN IX, pp. 1008–1017. Lecture Notes in Computer Science, Springer, Berlin, Heidelberg (2006). https://doi.org/10.1007/11844297_102, https://link.springer.com/chapter/10.1007/11844297_102

[14] Genton, M.G.: Classes of Kernels for Machine Learning: A Statistics Perspective. J. Mach. Learn. Res. **2**, 299–312 (Mar 2002), http://dl.acm.org/citation.cfm?id=944790.944815

[15] Grave, E., Bojanowski, P., Gupta, P., Joulin, A., Mikolov, T.: Learning Word Vectors for 157 Languages. In: Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC-2018). European Languages Resources Association (ELRA), Miyazaki, Japan (May 2018)

[16] Hintze, J.L., Nelson, R.D.: Violin Plots: A Box Plot-Density Trace Synergism. The American Statistician **52**(2), 181–184 (May 1998). https://doi.org/10.1080/00031305.1998.10480559, https://www.tandfonline.com/doi/abs/10.1080/00031305.1998.10480559

[17] Howley, T., Madden, M.G.: An Evolutionary Approach to Automatic Kernel Construction. In: Artificial Neural Networks – ICANN 2006. pp. 417–426. Lecture Notes in Computer Science, Springer, Berlin, Heidelberg (Sep 2006). https://doi.org/10.1007/11840930_43, https://link.springer.com/chapter/10.1007/11840930_43

[18] Koch, P., Bischl, B., Flasch, O., Bartz-Beielstein, T., Weihs, C., Konen, W.: Tuning and evolution of support vector kernels. Evolutionary Intelligence **5**(3), 153–170 (Sep 2012). https://doi.org/10.1007/s12065-012-0073-8, https://link.springer.com/article/10.1007/s12065-012-0073-8

[19] Koza, J.R.: Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT Press (1992)

[20] Kronberger, G., Kommenda, M.: Evolution of Covariance Functions for Gaussian Process Regression Using Genetic Programming. In: Computer Aided Systems Theory - EUROCAST 2013. pp. 308–315. Lecture Notes in Computer Science, Springer, Berlin, Heidelberg (Feb 2013). https://doi.org/10.1007/978-3-642-53856-8_39, https://link.springer.com/chapter/10.1007/978-3-642-53856-8_39

[21] Lampos, V., Zou, B., Cox, I.J.: Enhancing Feature Selection Using Word Embeddings: The Case of Flu Surveillance. In: Proceedings of the 26th International Conference on World Wide Web. pp. 695–704. WWW '17, International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, Switzerland (2017). https://doi.org/10.1145/3038912.3052622, https://doi.org/10.1145/3038912.3052622, event-place: Perth, Australia

[22] Lloyd, J.R., Duvenaud, D., Grosse, R., Tenenbaum, J., Ghahramani, Z.: Automatic Construction and Natural-Language Description of Nonparametric Regression Models. In: Twenty-Eighth AAAI Conference on Artificial Intelligence (Jun 2014), https://www.aaai.org/ocs/index.php/AAAI/AAAI14/paper/view/8240

[23] Mercer, J., A, B.: XVI. Functions of positive and negative type, and their connection the theory of integral equations. Phil. Trans. R. Soc. Lond. A **209**(441-458), 415–446

(Jan 1909). https://doi.org/10.1098/rsta.1909.0016, http://rsta.royalsocietypublishing.org/content/209/441-458/415

[24] Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed Representations of Words and Phrases and their Compositionality. In: Burges, C.J.C., Bottou, L., Welling, M., Ghahramani, Z., Weinberger, K.Q. (eds.) Advances in Neural Information Processing Systems 26, pp. 3111–3119. Curran Associates, Inc. (2013), http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf

[25] Montana, D.J.: Strongly Typed Genetic Programming. Evolutionary Computation **3**(2), 199–230 (Jun 1995). https://doi.org/10.1162/evco.1995.3.2.199, https://doi.org/10.1162/evco.1995.3.2.199

[26] Pennington, J., Socher, R., Manning, C.: Glove: Global vectors for word representation. In: Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP). pp. 1532–1543 (2014)

[27] Polajnar, T., Rogers, S., Girolami, M.: Protein interaction detection in sentences via Gaussian Processes: a preliminary evaluation. International journal of data mining and bioinformatics **5**(1), 52–72 (2011)

[28] Powell, M.J.D.: An efficient method for finding the minimum of a function of several variables without calculating derivatives. The Computer Journal **7**(2), 155–162 (Jan 1964). https://doi.org/10.1093/comjnl/7.2.155, http://comjnl.oxfordjournals.org/content/7/2/155

[29] Preoţiuc-Pietro, D., Cohn, T.: A temporal model of text periodicities using Gaussian Processes. In: Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing. pp. 977–988 (2013)

[30] Rasmussen, C.E., Williams, C.K.: Gaussian processes for machine learning. MIT Press (2006)

[31] Santana, R.: Reproducing and learning new algebraic operations on word embeddings using genetic programming. arXiv:1702.05624 [cs] (Feb 2017), http://arxiv.org/abs/1702.05624, arXiv: 1702.05624

[32] Schwarz, G.: Estimating the Dimension of a Model. The Annals of Statistics **6**(2), 461–464 (Mar 1978). https://doi.org/10.1214/aos/1176344136, https://projecteuclid.org/euclid.aos/1176344136

[33] Shaffer, J.P.: Modified Sequentially Rejective Multiple Test Procedures. Journal of the American Statistical Association (Mar 2012), https://amstat.tandfonline.com/doi/abs/10.1080/01621459.1986.10478341

[34] Shah, K., Cohn, T., Specia, L.: An investigation on the effectiveness of features for translation quality estimation. In: Proceedings of the Machine Translation Summit. vol. 14, pp. 167–174. Citeseer (2013)

[35] Snover, M., Dorr, B., Schwartz, R., Micciulla, L., Makhoul, J.: A study of translation edit rate with targeted human annotation. In: In Proceedings of Association for Machine Translation in the Americas. pp. 223–231 (2006)

[36] Specia, L.: Exploiting objective annotations for measuring translation post-editing effort. In: Proceedings of the 15th Conference of the European Association for Machine Translation. pp. 73–80 (2011)

[37] Specia, L., Shah, K., de Souza, J.G., Cohn, T.: QuEst - A translation quality estimation framework. In: Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics: System Demonstrations. pp. 79–84. Association for Computational Linguistics, Sofia, Bulgaria (Aug 2013)

[38] Sullivan, K.M., Luke, S.: Evolving Kernels for Support Vector Machine Classification. In: Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation. pp. 1702–1707. GECCO '07, ACM, New York, NY, USA (2007). https://doi.org/10.1145/1276958.1277292, http://doi.acm.org/10.1145/1276958.1277292

[39] Vapnik, V.N.: The Nature of Statistical Learning Theory. Springer-Verlag, Berlin, Heidelberg (1995)