TUM sebis at GermEval 2022: A Hybrid Model Leveraging Gaussian Processes and Fine-Tuned XLM-RoBERTa for German Text Complexity Analysis

Juraj Vladika*, **Stephen Meisenbacher***, **Florian Matthes**
Technical University of Munich
Department of Informatics
Garching, Germany
{juraj.vladika, stephen.meisenbacher, matthes}@tum.de

Abstract

The task of quantifying the complexity of written language presents an interesting endeavor, particularly in the opportunity that it presents for aiding language learners. In this pursuit, the question of what exactly about natural language contributes to its complexity (or lack thereof) is an interesting point of investigation. We propose a hybrid approach, utilizing shallow models to capture linguistic features, while leveraging a fine-tuned embedding model to encode the semantics of input text. By harmonizing these two methods, we achieve competitive scores in the given metric, and we demonstrate improvements over either singular method. In addition, we uncover the effectiveness of Gaussian processes in the training of shallow models for text complexity analysis.

1 Introduction

In this paper, we present a novel approach for the quantification of text complexity in the German language, as part of the Text Complexity DE Challenge 2022 (Mohtaj et al., 2022). Specifically, we emphasize a hybrid method for building a text complexity model, which combines a feature-based, shallow regression model with a fine-tuned XLM-RoBERTa model. In doing so, we hope to capture to the fullest both the linguistic aspects that contribute to text complexity, as well as the semantic factors. In the following Section 2, we briefly describe the task at hand, as well as the data used to train and test our models. Next, Section 3 introduces Gaussian Processes, which become central to our hybrid system. Likewise, fine-tuning RoBERTa for use in regression tasks is covered in Section 4. These concepts are brought together in 6, which describes our overall model architecture for the task. Before this, the feature set used to train both models is illustrated in Section 5. In Section 7, we present results from the training and validation phases, in which our model achieved the best score for this task's chosen metric. In the ensuing Section 8, we perform a qualitative analysis of our approach and lessons learned. Finally, Section 9 provides a few concluding remarks. The systems (and code used to create them) are publicly available under https://github.com/sebischair/Text-Complexity-DE-2022.

2 Dataset and Task

The dataset used in this task was first presented by Naderi et al. (2019). It consists of 1000 German language sentences sourced from 23 Wikipedia articles. These articles have been classified into three different genres. These sentences are annotated with ratings of 1-7 in the category of *Complexity*, *Understandability*, and *Lexical complexity*. These ratings are presented as a *Mean Opinion Score* (MOS), which is represented as the average scoring of the annotators. The sentences were scored by German language learners from A2 to B2 levels. For example, the sentence "*Eine Seifenblase entsteht, wenn sich ein dünner Wasserfilm mit Seifenmolekülen vermischt.*" is scored with 2.9.

For this challenge, one unified MOS score was given. How this score was derived from the original three scores is not explained. The original 1000 sentences are the same.

With this dataset, the task becomes to train a regression model that predicts the complexity of a sentence (i.e. MOS) from the sentence text. This score is intended to aid in the quantification of text complexity for language learners, as well as in the evaluation of the simplified text.

3 Gaussian Processes

A popular area in Machine Learning, particularly near the turn of the century, rested in the study of "kernel machines", which include the popular Support Vector Machines, but also the lesser known *Gaussian Process Models* (Rasmussen, 2003). At

*These authors contributed equally.

their core, Gaussian Processes (GPs) are powerful in the way they incorporate probabilistic thinking into kernel machines, making them a particularly suitable tool for supervised machine learning in small data settings (Urtasun and Darrell, 2007).

Literally, Gaussian processes are built upon multivariate Gaussian (normal) distributions, defined by a mean vector and covariance matrix , i.e.:

$$X = [X_1 \quad X_2 \quad \cdots \quad X_N] \quad N(\; ; \;)$$

This particular distribution has the useful property of being closed under marginalization (probability distribution of partitions) and conditioning (probability of one variable depending on another).

The main goal of Gaussian processes is to learn the underlying distribution of training data. Key to this process is the utilization of *Bayesian inference*, in which one assumes a prior and updates this hypothesis based upon new data. When modeling using Gaussian processes, a prior with dimensionality equal to that of the unseen points is chosen. A *kernel* is used to generate the covariance matrix by evaluating it on all training points.

In order to form the posterior distribution (i.e. train the model), the model observes training data, and conditions the current distribution based upon these new points. As new data comes in, the set of functions that the model can take is constrained, as only those functions exactly containing the new points are valid. The change in distribution induced by observing new points is reflected in an adjustment of the mean and standard deviation (achieved through marginalization). In addition, uncertainty in the data is modeled by adding an error term to the training points, modeled by $N(\; ; \; ^2)$.

Predictions from a trained Gaussian process model are made simply by sampling from the distribution of the model. In this way, Gaussian processes interestingly combine the ability to model (understand) the underlying distribution of the data at hand, as well as make accurate predictions from unseen instances. They, therefore, present a promising, powerful, and efficient method for tackling regression tasks (Williams and Rasmussen, 1995).

4 XLM-RoBERTa for Regression

Large pre-trained language models (PLMs) based on the transformer architecture (Vaswani et al., 2017) have achieved state-of-the-art performance on a wide array of common NLP tasks. Devlin et al. (2019) introduced BERT as a powerful language representation model that learns deep contextual representations of words. RoBERTa (Liu et al., 2019) is a robustly optimized extension of the BERT model. Both of these models work primarily on English text, so we decided to use the multilingual model XLM-RoBERTa (XLM-R), a variation of RoBERTa trained on data written in one hundred languages (Conneau et al., 2020). This model can recognize the language of an unseen textual input and achieves remarkable performance on a variety of non-English tasks, beating even the monolingual models optimized for specific languages.

The key benefit of using PLMs is the ability to load the already pre-learned contextual word embeddings and then to fine-tune them for the specific downstream task at hand. We tried out different pre-trained models for XLM-R and the best-performing ones for this task were the original *xlm-roberta-base*, *twitter-xlm-roberta-base*, and *xlm-roberta-base-wikiann-ner*. While PLMs like XLM-R are more commonly used for classification tasks, they can also be adapted for regression tasks. We achieved this by adding a new linear layer on top of the XLM-R. This linear layer had as its input the outputs of the final (12th) layer of XLM-R and learned what weights to assign to them.

Since our dataset contains only around 1000 examples, the process of fine-tuning had to be carried out carefully in order to prevent overfitting. Hyperparameters used were: number of folds 5, number of epochs 3, batch size 16, max. length of 100, no weight decay. The starting learning rate was 10^{-5}, after one third of all layers 5 10^{-5}, and after two thirds it was 10^{-4}. The idea behind this was that lower encoder layers can be understood as learning the lexical and syntactical features of the text, whereas higher layers model the semantic representation of it. For the task of text complexity, low-level features are more important so more emphasis was placed on them.

5 Feature Selection

The complete set of crafted features for model training is listed in Table 1. These features are separated into categories, followed a brief description, with supporting notes at the bottom. The character-, token-, and POS-based features were inspired by Falkenjack et al. (2013) and Chatzipanagiotidis et al. (2021). Before feature creation, preprocessing included stopword removal and lemmatization.

	Feature	Description
CHARACTER-BASED*		
	Avg_chars	Average number of characters per token in sentence
	Tokens_*N*	Number of tokens in sentence with length > N[1]
TOKEN-BASED*		
	Type_token	Distinct number of token *types*
	Carroll_TTR	Carroll's Corrected TTR measure
COMMON WORDS*		
	Num_common	Number of tokens found in the top 500 most common German words[2]
	Common_score	Cumulative score based upon rank in top 500 list
SENTENCE ATTRIBUTES*		
	Sentence_length	Length of sentence, i.e. number of tokens
	Longest_word	Length of the longest word in a sentence
	Commas	Number of commas in the sentence
	Parentheses	Number of (open) parentheses characters
	Digits	Number of numerical digits in the sentence
	Quotes	Number of quotation characters (ór ¨) in the sentence
	Avg_word_length	Average length of words in the sentence
	Wordrank_score	Overall score calculated from German Wiki frequency list[3]
POS TAGS*		
	POS_ratio	Ratio of (spaCy.pos_) POS Tags in sentence [4]
	TAG_ratio	Ratio of (spaCy.tag_) detailed POS Tags in sentence[5]
SPACY FEATURES*		
	Dep_length	Cumulative length (width) of dependencies in sentence
	Ne_length	Total length of all named entities in sentence
	Ne	Number of named entities in sentence
	L2_norm	L2 Norm of spaCy word vector representations
	Vec_exists	Number of sentence tokens for which a spaCy vector exists
SYNTAX TREE		
	Syn_height	Height of syntax tree
	Leaves	Number of leaves in syntax tree
	Subtrees	Number of subtrees in syntax tree
	Leaf_distance	Cumulative distance between the leave nodes in the sentence
SYLLABLES		
	Tot_syl	Total number of syllables in sentence
	Avg_syl	Average number of syllables per word
	Single_syl	Number of single syllable words in sentence
READABILITY[6]		
	Flesch	Flesch reading ease score
	Flesch_mod	Modified Flesch score
	Easy_words	Number of words in sentence with 2 syllables
	Hard_words	Number of words in sentence with > 2 syllables
	Gunning_fog	Gunning Fog readability index
	Mod_smog	SMOG readability index
	Mod_forcast	Forcast readability formula
	Ari	Automated readability index
	Linsear	Linsear write readability metric
WORDNET[7,†]		
	Synset_exists	Number of lemmas in sentence for which a synset exists
	Synset_depth	Cumulative maximum depth of all existing synsets in sentence
	Hyponyms	Cumulative number of hyponyms for existing synsets
	Senses	Cumulative number of word senses for existing synsets
	Syn_def	Total length of synset definitions for all existing synsets
	Avg_path	Average path length from one synset to the next, in sequential order
EXPERIMENTAL†		
	Scrabble_new	Scrabble score using the new German Scrabble point values
	Scrabble_old	Scrabble score using the old German Scrabble point values

[1] for $N \in \{2, 6, 7, 8, 10\}$
[2] https://www.thegermanprofessor.com/top-500-german-words/
[3] https://github.com/gambolputty/dewiki-wordrank
[4] for POS ∈ {'ADJ', 'ADP', 'ADV', 'AUX', 'NOUN', 'NUM', 'PRON','PROPN', 'VERB', 'X'}
[5] for TAG ∈{'ADJA', 'ADJD', 'ADV', 'APPR', 'ART', 'KON', 'KOUS', 'NN', 'PRELS', 'VAFIN', 'VVFIN', 'VVPP'}
[6] Where applicable, scores are modified for single sentences (denoted by *mod*)
[7] Using the *Open German WordNet*: https://github.com/hdaSprachtechnologie/odenet
* Features in these categories are calculated on the logarithmic scale, with either add-1 or add-0.1 smoothing, where necessary
† These features were not used in the final model (best test score)

Table 1: Feature Set

6 A Hybrid System

The development of the eventual final model took place in an iterative fashion. First, an array of popular shallow models were tested. In this process, the discovery of the effectiveness of Gaussian processes for this specific task led the authors to choose these particular models for tuning. The kernel used was the sum of Constant, Matern, and White kernels, optimized with 10 restarts. As the training of Gaussian process models seemed to hit a plateau, a deeper approach was pursued, namely using RoBERTa. This achieved good results (see Section 7), leading the authors to believe that some deep component was key to the task at hand.

Due to the documented success of stacking and ensemble methods (Pavlyshenko, 2018; Ganaie et al., 2021), the authors considered a third approach in which the best shallow and deep models (GPs and RoBERTa) were to be stacked. Concretely, the predictions of the two models could be harmonized in a way that combines the strengths of both. Traditionally, stacking is performed by training a "meta-model", which learns the optimal way to combine the outputs of the "level 0" models.

With this in mind, the authors took a simplified approach to stacking, in which the output predictions of the Gaussian process model and the fine-tuned XLM-RoBERTa were simply averaged. This resulted in the "meta" predictions, which were then used for submissions. In the development phase, this method proved to be the most effective, far outperforming both individual models. As such, a hybrid system was created, which was later utilized in the test phase. Results from the development phase are outlined in Section 7, where the performance of the individual and hybrid models are displayed.

7 Training and Results

In the following Table 2, we present the results from the development phase of the challenge. In particular, we include both the traditional Root Mean Squared Error (RMSE) for each model, as well as the *RMSE_Mapped* metric used for this specific task. Since the MOS of human annotators inherently includes subjective biases and offsets, some statistical uncertainty is always present in the scores (Yi et al., 2022). Therefore, a linear mapping function is applied to the RMSE in order to compensate for the possible variance between several subjective experiments. It should be noted that the specifics on how to calculate this mapped metric were not provided for this challenge.

Model	RMSE	RMSE_mapped
Lasso Regression	–	0.515
Ridge Regression	–	0.507
XGBoost Regression	0.520	0.490
Partial Least Sq. Regression	0.492	0.462
LightGBM Regression	0.465	0.434
Random Forest Regression	0.457	0.427
Gaussian Process Regression	0.453	0.401
w/ 50% train data	0.447	0.380
+ 20-dim PCA	0.442	0.377
+ noisy targets	0.427	**0.373**
XLM-RoBERTa (SQuAD 2.0)	0.443	0.442
XLM-RoBERTa (WikiAnn)	0.434	0.424
XLM-RoBERTa (Twitter)	0.434	0.420
w/ 70% train data	0.430	0.393
XLM-RoBERTa (Base)	0.426	0.415
w/ 150 features	0.438	0.403
w/ 20-dim PCA	0.440	0.399
w/ 70% train data	0.433	**0.384**
XLM-R (0.415) + GP (0.377)	0.395	0.349
XLM-R (0.399) + GP (0.377)	0.415	0.342
XLM-R (0.384) + GP (0.373)	0.397	0.331
XLM-R (0.384) + GP (0.377)	0.408	0.328
XLM-R (0.393) + GP (0.373)	0.394	0.328
XLM-R (0.393) + GP (0.377)	0.401	**0.324**

Table 2: Development Phase Results

In Table 2, bolded are the best-performing shallow and deep models, as well as the best-performing stacked model, which did not use either of the two best single models. The most effective shallow model was the Gaussian process regressor that used our handcrafted features described in Section 5 and Table 1 to learn the optimal distribution over the training data. Only 50% of training data was randomly selected and used to train the model since this provided the optimal performance, as measured by the mapped RMSE metric.

Model	RMSE	RMSE_mapped
XLM-R (Base, 70% train) + GP (20-dim PCA, 50% train)	0.514	0.489
XLM-R (WikiAnn, 70% train) + GP (20-dim PCA, 50% train)	0.518	0.488
XLM-R (Twitter, 70% train) + GP (20-dim PCA, 50% train)	0.518	0.465
XLM-R (Base + 20-dim PCA) + GP (20-dim PCA, 50% train)	0.473	0.459
XLM-R (Twitter, 60% train) + GP (20-dim PCA, 50% train)	0.485	**0.457**

Table 3: Final Phase Results

The best-performing deep model was the base model of XLM-RoBERTa. Although adding our handcrafted features to it improved the performance, the trick of using a reduced training data set again provided us with the best results (70% of

the training data). For the final stacked model, various combinations of GP and XLM-R models were tried out. The optimal combination turned out to be the XLM-RoBERTa fine-tuned on a Twitter dataset and the Gaussian Process using a 20-dimensional PCA representation of handcrafted features. This hybrid model achieved the mapped RMSE score of 0.324, which was the winning score (1st place) of the development phase of the competition.

Table 3 shows the results of the models submitted for the final phase of the competition. The authors decided to submit the best-performing models from the previous phase. The best-scoring model was later revealed to be the stacked combination of the XLM-RoBERTa pre-trained on Twitter and the Gaussian process with 20-dimensional PCA features, both models trained on reduced data. This came as no surprise as this was also the best-performing model in the previous phase. The model achieves the mapped RMSE score of 0.457, which resulted in 6th place in the final phase.

8 Discussion

Here, the authors reflect on lessons learned, useful findings, and possible future directions.

A useful and somewhat surprising finding came with the excellent performance of Gaussian processes, particularly during the development phase. In pondering why this occurred, one can look to the nature of GPs in conjunction with the specifics of the text complexity task. At their core, Gaussian process models aim to capture the underlying distribution of data that is complex. Furthermore, GPs seem to shine when the amount of training data is relatively small, e.g. under 1000 instances. As such, GPs may have been a logical choice for this task, which comprised of a quite small dataset and whose goal represents a quite complex regression task. Indeed, the quantification of text complexity proves to be challenging to reason about. Nevertheless, the effectiveness shown by GPs merits their consideration in future, related tasks.

Regarding XLM-RoBERTa, it cannot be denied that its inclusion greatly strengthened the final model. It is interesting, though, that the model performance quite significantly varied based upon the particular pre-trained model that was chosen. In this light, a potential future direction would involve further investigation into better models, as well as *why* these might be superior. Similarly, a more focused tuning of the hyperparameters for the fine-tuning process could have been performed. This likewise remains as future work.

In the creation of the hybrid model which was eventually used in the test phase, an interesting lesson was learned regarding how to "stack" our two best models. A more systematic way of doing so would be to learn a meta-model, i.e. a simple linear regression model, to stack on top of our GP + RoBERTa hybrid. As it turns out, learning such a model actually performed worse than a simple average of the two models' predictions. In this way, simplicity won the day in regards to the design of a well-performing hybrid regression model.

The analysis of our feature set also produces interesting insights. As a cursory analysis, heatmaps illustrating the correlation amongst features and to the target values are presented in Figures 1 and 2 in the Appendix. The question then becomes whether more features should have been included, particularly those with a firm linguistic foundation.

This notion is grounded in the authors' initial intuition that was used to produce the feature set. Interesting findings were gained here, too, such as the relatively high correlation of complex linguistic concepts (e.g. passive voice, genitive case, depth of syntax tree) to the MOS. We pose that such linguistic thinking is important for further improvements.

As already alluded to, the complexity of the data itself, or rather the ability to describe it effectively via features, proved to be a central problem in tackling the task at hand. In the process of studying the datasets, the authors noticed particular characteristics that could be crucial to feature creation. Of these, notable observations include the presence of rare-occurring symbols (e.g. §), as well as sentences that are clearly "simplified" sentences, rather than sourced directly from Wikipedia. Possible discrepancies between the dev and test sets were also observed. Accounting for such factors in the features is likely key to model performance.

9 Conclusion

In this paper, we discuss our novel approach to the quantification of text complexity in the German language. In particular, we present a hybrid model using Gaussian Processes and a fine-tuned XLM-RoBERTa. We also provide our full feature set, which to be best of the authors' knowledge includes features not previously presented in the literature. Finally, we discuss our results in the challenges and reflect upon their implications for future work.