Word Complexity Prediction Algorithm
Machine Learning

2nd Semester of 2024-2025

Tudorache Andrei-Theodor
andrei-theodor.tudorache@s.unibuc.ro

Gheorghe Andrei-Bogdan
andrei-bogdan.gheorghe@s.unibuc.ro

Abstract

This paper presents a machine learning algorithm used to predict lexical complexity in Romanian, focusing on a hybrid methodology that combines handcrafted linguistic features with contextual embeddings from RoBERT. We explored multiple regression models and also fine-tuned a transformer model for this task.

1 Introduction

Lexical complexity prediction consists in estimating how difficult a given word is to understand, depending on the sentence in which it appears. We tackled this problem for the Romanian language by combining traditional linguistic features with contextual embeddings from RoBERT.

We worked together throughout the entire project, either in person or online. All major decisions, feature ideas, and implementation details were discussed and developed collaboratively.

Experiments tried:

- We started with a basic set of traditional features such as word length and frequency, which offered a simple but interpretable foundation.
- Gradually, we extended the feature set to include syllable count, part-of-speech tags, and syntactic details extracted using spaCy, aiming to capture more nuanced linguistic patterns.
- Once the traditional pipeline was solid, we integrated contextual embeddings from RoBERT, using mean pooling over token representations to obtain sentence-aware vector features.
- To evaluate performance, we experimented with multiple regression models including Ridge, SVR, LightGBM, XGBoost, and CatBoost, comparing results and tuning hyperparameters for each.
- Finally, we fine-tuned RoBERT directly for regression using the HuggingFace Trainer and combined its predictions with those from traditional models using ensemble averaging to boost robustness and performance.

We worked on this project because we were interested in learning more about the field of Natural Language Processing and gaining experience with techniques used in this domain.

2 Approach

In this section, we present the full process we followed to develop our word complexity prediction algorithm. This method combines classic linguistic features with embeddings, comparing multiple models to find the best-performing one.

GitHub: https://github.com/curs-ia-2025/complex25-andreig

Tools used: Python, scikit-learn, LightGBM, XGBoost, CatBoost, spaCy, wordfreq, HuggingFace Transformers, PyTorch.

Training time:

- Traditional ML models: 2-3 minutes on CPU.
- RoBERT fine-tuning: 10 minutes on NVIDIA RTX3050 GPU.

Machine Learning pipeline:

- Feature extraction: 300-dimensional spaCy embeddings, linguistic features, including: length, frequency, POS, etc.
- RoBERT embeddings: 768-dimensional contextual sentence representations - mean-pooled.
- Combined features: All features were concatenated and used to train ensemble-friendly regressors such as GradientBoosting and LightGBM. Model selection and tuning were performed using GridSearchCV.

- RoBERT fine-tuning: The RoBERT model was also fine-tuned directly for regression using the HuggingFace Trainer API, with R^2 as the main evaluation metric.

Tricks used: We used mean pooling to convert token embeddings from RoBERT into fixed-size sentence vectors. Features were normalized with StandardScaler to make sure all data had a similar scale, which helped the models learn better. We tuned model hyperparameters with GridSearchCV to find the best settings and avoid overfitting. Finally, we combined predictions from different models by averaging them, which improved the overall accuracy and stability of our results.

In addition, we consulted ChatGPT to gather suggestions on potential improvements and additional features we could implement. This helped us discover and implement the use of BERT embeddings, as well as the idea of comparing different models to select the best-performing ones.

Evaluation: All models were evaluated using the R^2 metric. The R^2 metric measures the proportion of variance explained by the model, with values closer to 1 indicating better fit. Optimized GradientBoosting reached R^2 0.97, while the fine-tuned RoBERT model achieved slightly better performance ($R^2 > 0.97$). The final submission used an average of both predictions to improve generalization and stability.

3 Conclusions

We really enjoyed working on this project and learned a lot, not only about NLP concepts, but also about collaborating effectively as a team, with each of us bringing our own strengths to the table.

Andrei-Theodor Tudorache:

I enjoyed working at this project and it helped me develop a better understanding of Machine Learning, especially working with a real-like scenario. I got to play with different base models and see how they work, learned how to do a fine-tunning for an already trained model and how to manage and manipulate datasets to train a NLP.

Andrei-Bogdan Gheorghe:

Through this project, I gained practical knowledge in both transformer fine-tuning and classic ML techniques. However, I personally enjoyed working more on segmentation and image-related tasks during the semester project than on NLP. I found it relatively challenging to fully understand the requirements for this NLP task, even after doing some research.

Suggestions

One suggestion for the course would be to place more emphasis on the project itself, clearly outlining what is expected, as the instructions felt a bit ambiguous. Also, starting the project earlier in the semester would help students manage their time and progress more smoothly.

References

[1] C. Tănase and S. D. Dumitrescu. RoBERT – Romanian BERT Model.

[2] S. Nisioi. Machine Learning Course Repository. https://github.com/senisioi/curs-ia