Stanford MLab at SemEval-2021 Task 1: Tree-Based Modelling of Lexical Complexity Using Word Embeddings

Erik Rozi*, **Niveditha Iyer***, **Gordon Chi**, **Enok Choe**, **Kathy Lee**,
Kevin Liu, **Patrick Liu**, **Zander Lack**, **Jillian Tang**†, and **Ethan A. Chi**†
Stanford University
{erikrozi, nivsiyer}@stanford.edu
{jiltang, ethanchi}@cs.stanford.edu

Abstract

This paper presents our system for the single- and multi-word lexical complexity prediction tasks of SemEval Task 1: Lexical Complexity Prediction. Text comprehension depends on the reader's ability to understand the words present in it; evaluating the lexical complexity of such texts can enable readers to find an appropriate text and systems to tailor a text to an audience's needs. We present our model pipeline, which applies a combination of embedding-based and manual features to predict lexical complexity on the CompLex English dataset using various tree-based and linear models. Our method is ranked 27 / 54 on single-word prediction and 14 / 37 on multi-word prediction.

1 Introduction

The rapid expansion of social media and other online channels has made readable information available at an astounding rate. However, the accessibility of this information is often limited by the complexity of this information, especially among readers with low literacy levels in the language of the text and those with reading disabilities. Furthermore, even to the average reader, specialized jargon found in governmental documents and scientific fields is often difficult to decipher.

Systems to guide these users may redirect readers to more easily comprehensible sources, convert the text to simpler wording, or provide additional information about what difficult words mean. The development of such systems is benefited by the ability to evaluate the complexity of sections of the text. While there is currently a large amount of available text data, very little of it is labeled with word complexity; automating the labelling process would make much more data available to aid the development of NLP systems in tasks such as text simplification.

Multiple features of a word can affect lexical complexity. In addition to a word's frequency, length and syllable count, the context in which a word is found is likely to affect its understandability. The additional factor of the reader's proficiency in a language makes this task complex as many words have a highly variable complexity.

In this paper, we describe our model that predicts single- and multi-word lexical complexity scores.

2 Background

2.1 Task Overview

All data was provided through SemEval Task 1 (Shardlow et al., 2021). Our dataset consists of an augmented version of the CompLex Corpus (Shardlow et al., 2020), which contains English sentences from three genres of corpora: the Bible, Europarl, and biomedical writing. From each sentence, both single- and multi-word tokens were selected and annotated by approximately 7 annotators. Each token was annotated on complexity from a scale of 1-5, though for this competition, complexity was normalized to a continuous scale between 0 and 1.

Token complexity can differ based on the complexity of the token both with and without context. For example, for one instance, the token *river* was rated to have a complexity of 0.0, while *jurisprudence* had a complexity of around 0.672 for another instance. However, token complexities can also change based on the context from which it came from. For example, the token *wisdom* was given a complexity of 0.125 when it was associated in the sentence *"The rod of correction gives **wisdom**, but a child left to himself causes shame to his mother."* However, the same token was given a significantly higher complexity score of 0.368 when associated with the sentence *"For in much **wisdom** is much*

*Co-first authors.
†Co-senior authors.

Proceedings of the 15th International Workshop on Semantic Evaluation (SemEval-2021), pages 688–693
Bangkok, Thailand (online), August 5–6, 2021.

grief; and he who increases knowledge increases sorrow."

Given that GloVe embeddings (Pennington et al., 2014) store semantic meaning of single words, we chose to use GloVe embeddings to represent both tokens and sentences. With this approach, we determine that despite contextual variation, inherent properties of the token itself are sufficient to explain much of the variance in lexical complexity.

2.2 Traditional Text Complexity Metrics

Many traditional metrics for calculating the complexity of text predict with syllable to word count ratios. For example, the Flesch-Kincaid Grade Level Formula [1] (Kincaid et al., 1975) calculates the complexity of a text with the formula

$$\text{GL} = 0.39\frac{\text{words}}{\text{sentence}} + 11.8\frac{\text{syllables}}{\text{word}} - 15.59.$$

Other models based on the grade level of a text, such as the Automated Readability Index and the SMOG Index (Kincaid et al., 1975), also exist. Our original hypothesis inferred that these indexes would be good indicators to predict the complexity of a token. However, through empirical analysis, we found that these indicators provided no marginal benefit compared to GloVe sentence embeddings and simpler handcrafted features. As seen in Table 1, we found that the correlation coefficients of traditional complexity metrics to dataset complexity values were low. To test this, we initially included these traditional metrics in our feature space for the following models. Our model reported an R score of 0.63 with the Flesch-Kincaid Grade and SMOG Index as additional features. We removed these features after observing little benefit or worse loss scores (in comparison to Table 2). This suggests that word complexity in context may be embedded in a deeper semantic level than simple word and syllable lengths.

Model	Pearson
Flesch-Kincaid Grade	0.07
Automated Readability Index	0.07
SMOG Index	0.03

Table 1: Pearson correlation between complexity metrics and true complexity values (single-word)

[1] https://github.com/shivam5992/textstat

3 System Overview

3.1 Single Word Complexity Score

3.1.1 Data Representation and Features

This system uses a combination of GloVe (Pennington et al., 2014) word embeddings and hand-crafted features as final features to predict complexity on. Pre-trained GloVe embeddings with a dimension of 300 for both the single-word token and each word in the context sentence were used. For the single-word embeddings, PCA with a final dimension of 100 was applied. Since the context sentences contained a variable number of words, we calculated the component-wise mean of all the word-vector representations in the context sentence. We found that sentence features had low mutual information, hence we decided to use a limited number of 10 PCA features to calculate the mean of the sentence features. This mean representation is concatenated with the GloVe embedding of the single-word token.

In other words, let $\mathbf{t}$ be the GloVe embedding of the single-word token, and $\mathbf{w}_i$ be the GloVe embedding for word i in the context sentence, with n words. We calculate the sentence representation $\mathbf{s}$ to be

$$\mathbf{s} = \frac{\sum_i^n \mathbf{w}_i}{n},$$

leading to features $\mathbf{r} = [\mathbf{t}, \mathbf{s}]$ with a dimensionality of 110 features.

On top of this representation, we include handcrafted features. Through manual tuning, we created a set of manual features:

- **NUMLETTERS**: the number of letters in the token
- **NUMCAPITALS**: the number of capital letters in the token
- **NUMSYLLABLES**: the number of syllables in the token
- **NUMDIGITS**: the number of digits in the token
- **ISFIRSTCAPITAL**: whether or not the first letter is capitalized (implying it is a subject or technical term)
- **NUMSENTWORDS**: the number of words in the context sentence
- **CORPUSTYPE**: the type of corpus the sentence is taken from

- **POS**: the part of speech of the token
- **IsInNER**: whether or not the token is in a named entity

The "POS" and "IsInNER" features are obtained from the Stanza NLP package (Qi et al., 2020).

Instead of relying on the frequencies of words in the text we were analyzing, we found that a more representative frequency metric could be obtained by counting word occurrences in all Wikipedia articles. Hence we decided to use frequencies of word as they appear in the English Wikipedia articles as of February 2019.[2] This feature was concatenated with all of the other handcrafted features and GloVe embeddings, leading to a final feature dimensionality of 126.

3.1.2 Learning Models[3]

Because the system primarily treats the input datapoints as sets of vectors and other numerical features, most of the models used were regressors made for data. As the baseline, we used linear regression with the GloVe embeddings for only the single-word token and obtained a baseline R of 0.7888 on the train set.

We explored the following machine learning models:

- **Ridge regression** is a linear least squares model with L2 regularization to reduce overfitting and variance. We use $\alpha = 0.00001$ as the regularization coefficient to prevent overfitting.
- **Support Vector Regression** is a Support Vector Machine for a regression task that tolerates errors within a certain degree ϵ. We use $\epsilon = 0.02$ as the distance within which no penalty is associated, and $C = 0.2$ as a regularization parameter to reduce overfitting.
- **Decision Tree Regression** creates a model as a series of decision rules. As a baseline, we created a decision tree with max_depth $= 6$, though other models use varying depths.
- **AdaBoost Regression** (Freund and Schapire, 1996) sequentially applies decision trees, with each tree placing more weight on data that previous trees did not fit well to. We use DecisionTreeRegressors with max_depth$= 10$ as the base estimator, with a total of $n_{\text{estimators}} = 20$ decision trees.
- **XGBoost Regressor** overcomes the inefficiency in gradient boosting of creating a single decision tree at a time by parallelizing tree building. We used max_depth$= 4$ and $\lambda = 2000$ as a regularization parameter. As λ is responsible for L2 regularization of weights, using a higher value would make the model more conservative by encouraging smaller weights.
- **LightGBM Regressor** [4] (Ke et al., 2017) is a framework that uses tree based learning algorithms for gradient boosting. Our model uses gain-based feature importance, with $\lambda = 50$ and $n_{\text{leaves}} = 40$ and a minimum of 100 datapoints per leaf. To avoid overfitting, we regularize with path smoothing of 1, set a maximum tree depth of 15, and trained using DART boosting.
- **Stacking** We also tested a stack of estimators with a final Ridge regressor to get an ensemble of predictions and reduce overfitting. We stacked five AdaBoost Regressors with $n_{\text{estimators}} = 50, 100$ estimators respectively, each with a base estimator of a Decision Tree Regressor with max_depth varying between 5, 7, and 9. On top of this, we stacked two Support Vector Regressors with $\epsilon = 0.01, 0.001$ and $C = 0.1, 0.01$ respectively. Finally, we stacked three LightGBM Regressors, each with $100, 50$, and 10 leaves respectively. This method was used with the theory that combining multiple models would result in better predictive power than one model alone.
- **Bagging** is an ensemble method involving training copies of a base model on independent random samples of the full dataset. We used an LGBM with $n_{\text{leaves}} = 40$, reg_lambda $= 100$, path_smooth $= 1$, max_depth $= 12$, and feature_fraction $= 0.75$ as our base model. We set $n_{\text{estimators}} = 10$, max_samples $= 0.8$, and max_features $= 0.75$ in order to reduce variance of the decision tree.

[2] https://github.com/IlyaSemenov/wikipedia-word-frequency

[3] All models were implemented using SKLearn (Pedregosa et al., 2012) unless otherwise mentioned.

[4] https://github.com/microsoft/LightGBM

- **BERT** We also explore context-dependent deep learning architectures: in particular, we fine-tune the pre-trained BERT model (Devlin et al., 2019). We leverage the pre-trained BERT neural network[5] by tokenizing each sentence, and providing the target word to the model as a second sentence. With 2-3 fully connected layers added on top of the pre-trained model, we fine-tuned this model to generate a numerical complexity prediction, by optimizing on the L2 Loss. All experiments were implemented using SKLearn (Pedregosa et al., 2012) and HuggingFace [6].

3.2 Multi-word Complexity Score

3.2.1 Data Representation and Features

Our multi-word data representation closely mirrored our single-word token representation. All hand-crafted features were crafted in the same way as the single word counterparts, except for the POS and NER features which were not included. For example, the feature **NumLetters** includes the number of letters from both words. The context sentence embeddings were calculated with the same methodology of applying PCA with dimension of 10 to the mean of the GloVe embeddings.

The key difference between the two models lies in the representation of the multi-word tokens themselves. The data provided was consistent in that each multi-word token consisted of two words. Therefore, to represent these tokens, we concatenated the GloVe representation of each word in the token, as well as the difference between both GloVe vectors. From there, we applied PCA of dimension 150 to this embedding, which was determined through experimentation, and concatenated this with the other hand-crafted and context sentence features mentioned previously.

More concretely, let $\mathbf{t}_1, \mathbf{t}_2$ be the GloVe embeddings of each word in the multi-word token. We found the new representation of a multi-word token $\mathbf{m}$ to be

$$\mathbf{m} = [\mathbf{t}_1, \mathbf{t}_2, \mathbf{t}_1 - \mathbf{t}_2].$$

This was concatenated with sentence representation $\mathbf{s}$ and handcrafted features for a final dimensionality of 174 features.

[5]We use tokenizers and pre-trained models from the HuggingFace transformers library: https://huggingface.co/transformers/model_doc/bert.html

[6]https://huggingface.com

3.2.2 Learning Models

Given the similarity of the multi-word representations versus the single-word representations (the only difference being the addition of a second token's GloVe embedding), we used the LightGBM Regressor outlined in section 3.1.2, as this model performed the best in the single word token setting. This proved to be an effective way to predict multi-word complexity.

4 Experimental Setup

The train and validation dataset splits provided were used in our experimental setup. In addition, we used K-fold validation to reduce overfitting. Using K-fold, we split the training set into k smaller sets arbitrarily, train using $k-1$ folds, and cross-validate with the remaining fold in the train set. This reduces leakage from the validation set into the model so that we can accurately validate our methods.

Task predictions were evaluated using Pearson correlation, though Spearman correlation, mean absolute error, mean squared error, and R-squared were also reported. We compared the performance of our own models using Pearson correlation to keep one consistent evaluation metric.

5 Results

5.1 Single Word Results

From Table 2, LGBMRegressor performs the best in terms of the Pearson metric. Therefore, we chose this model as our final model for submission.

We found that transforming the word frequencies to a logarithmic scale did not improve results across the models we tested. This is expected because tree-based regressors (Adaboost, LGBM, XGB) are invariant to monotonic scaling. Our results on the task evaluation metrics are shown in Table **??**.

We suspect Ensemble Stacking overgeneralized and did not perform effectively as a result, though other stacking methods could perform better. Surprisingly, the contextual deep learning approach of BERT did not perform well on the task, only approaching similar performance to the baseline linear regression on GloVe embeddings.

Though we scored 27th place out of 54 teams overall in the Pearson metric for single-words, the top score was only 0.03 points higher than our own evaluation score. We suspect that different methods of stacking regressors and using complex decision

Model	Pearson
Linear Regression	0.7888
BERT	0.7892
Ridge	0.7829
SVR	0.7945
DecisionTreeRegressor	0.7083
AdaBoost Regressor	0.7976
Ensemble Stacking	0.7578
XGBRegressor	0.7884
BaggingRegressor	0.8018
LGBMRegressor	**0.8056**

Table 2: Experimental results (single-word)

Metric	Score	Ranking
Pearson	0.7533	(27)
Spearman	0.7044	(34)
MAE	0.0653	(25)
MSE	0.0071	(29)
R2	0.5615	(26)

Table 3: Evaluation results (single-word)

trees would have created a model that predicts well with the CompLex dataset. However, whether this type of model will generalize to future datasets is a subject of investigation.

5.2 Multi-word Expressions Results

We note that our multi-word expression Pearson metric, as shown in Table **??**, performs better than our single word Pearson, and ranks 14th out of 37 teams. This is most likely because averaging the GloVe representations of the two tokens allows for more data points to be represented in the decision tree model.

6 Conclusion

In this paper we describe tree-based modelling of words in context to predict lexical complexity. We find that lexical complexity is already embedded in GloVe representations of words and that complex architectures provide some increase in predictive performance.

For future work, we suggest taking additional contextual features into account, such as the proximity of each neighboring word. We also suggest looking into newer transformer models to represent contextual embeddings.

As larger bodies of text become widely available to wide audiences for public consumption, we are hopeful that such systems will help readers identify suitable texts for their reading level and help build systems that can tailor text to varied reading levels, allowing for greater accessibility.

Metric	Score	Ranking
Pearson	0.8280	(14)
Spearman	0.8124	(18)
MAE	0.0711	(24)
MSE	0.0080	(24)
R2	0.6671	(14)

Table 4: Evaluation results (multi-word)

Acknowledgments

This research effort would not have been possible without the support of Stanford ACMLab. The authors thank Matthew Shardlow, Richard Evans, Gustavo Henrique Paetzold and Marcos Zampieri for organizing SemEval 2021 Task 1: Lexical Complexity Prediction. We also thank Yasmine Mitchell for helpful discussions.

References

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Yoav Freund and Robert E. Schapire. 1996. Experiments with a new boosting algorithm. In *Proceedings of the Thirteenth International Conference on International Conference on Machine Learning*, ICML'96, page 148–156, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. Lightgbm: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

J. Peter Kincaid, Robert P. Jr. Fishburne, Richard L. Rogers, and Brad S. Chissom. 1975. Derivation of new readability formulas (automated readability index, fog count and flesch reading ease formula) for navy enlisted personnel. *Institute for Simulation and Training*.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake VanderPlas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Edouard Duchesnay. 2012. Scikit-learn: Machine learning in python. *CoRR*, abs/1201.0490.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. 2020. Stanza: A Python natural language processing toolkit for many human languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*.

Matthew Shardlow, Michael Cooper, and Marcos Zampieri. 2020. CompLex — a new corpus for lexical complexity prediction from Likert Scale data. In *Proceedings of the 1st Workshop on Tools and Resources to Empower People with REAding DIfficulties (READI)*, pages 57–62, Marseille, France. European Language Resources Association.

Matthew Shardlow, Richard Evans, Gustavo Paetzold, and Marcos Zampieri. 2021. Semeval-2021 task 1: Lexical complexity prediction. In *Proceedings of the 14th International Workshop on Semantic Evaluation (SemEval-2021)*.