ALEXANDRU IOAN CUZA
UNIVERSITY of IAŞI

FACULTY OF
COMPUTER SCIENCE

Lexical Complexity Prediction Using Machine Learning Techniques

Nistor Marian-Sergiu
Popescu Calin
Sescu Alexandra

Coordinator:
Dr. Gîfu Daniela

Abstract

The purpose of the conducted research consists of improving the existing techniques in predicting lexical complexity on single-word and multi-word expressions, using statistical techniques and supervised machine learning.

Keywords: lexical complexity, machine learning, supervised learning, single-word, multi-word, statistical

Table of Contents

Abstract **1**

1 Introduction **1**
1.1 State of the art 1

2 Technical Aspects **3**
2.1 Datasets 3
2.1.1 WordNet 3
2.1.2 CompLex 3
2.2 Architecture 5

3 Implementation and Results **6**
3.1 Word Embedding 6
3.1.1 FastText 6
3.2 Complexity Prediction 7
3.2.1 Histogram-based Gradient Boosting Regression Tree 7

4 Conclusion and Future work **9**
4.1 Conclusion 9
4.2 Future work 9

References **13**

Appendices **14**

A Additional machine-learning algorithms training results **14**

Chapter 1

Introduction

Our proposed implementation for a statistical model for predicting lexical complexity makes use of Histogram-based Gradient Boosting Regression Trees [1], in combination with FastText [2] [3] [4], as the word embedding algorithm.
We used the CompLex Dataset [5] in order to train the Histogram-based Gradient Boosting Regression Tree. The token embeddings generated with FastText [2] [3] [4] will serve as features, while the complexities annotated in the CompLex Dataset [5] will serve as labels, when training the Histogram-based Gradient Boosting Regression Tree.
The word embedding algorithm(FastText [2] [3] [4]) uses WordNet [6] as its training corpus.
Implementations of these specific machine learning algorithms were taken from libraries such as scikit-learn [7] and Gensim [8].

1.1 State of the art

Previous attempts at predicting lexical complexity have also used various methods of embedding the words from the corpus, using GloVe [9] [10], or Word2Vec [11]. We have chosen to use FastText [2] [3] [4] as a mean of generating word embeddings, as it can embed out-of-vocabulary words, if trained on a big

enough corpus, due to its use of n-grams [12] [13].
As for the statistical/machine learning methods used to train and compute the complexity of a given word, there are implementations using ensemble learning/meta-classifiers, with methods such as decision trees [14], multi-layer perceptrons [15], linear SVMs [16], logistic regressors [16], polynomial regressors [16], linear regressors [16], random forest regressors [17]. We will study the performance of Histogram-based Gradient Boosting Regression Trees on the given task. Past papers suggest the use of features, such as word probability [14] [15], word length [14] [15] [17], sentence length(taking context into consideration), conditional and joint probabilities of the given word to occur(considering the previous word and its surrounding words) [14], multi-word expression compounds [15] [17], maximum complexity [15], mean complexity [15], text genre [15], number of sentences per text [16], average and maximum number of words per sentence [16], average number of characters per word [16], average number of syllables per words [16] [17], type-token ratio(ratio for the number of unique word tokens to the total number of word tokens in a text) [16], PoS [16] [17], lexical variation(type-token ratio of lexical items) [16], lexical density(ratio of content words and function words) [16], proportion of academic vocabulary words in text [16], n-grams [16] [17], suffix length [17], gender(for nouns) [17], degree of polysemy [17], degree of homonymy [17], topic distribution [17].
Our approach is oriented towards n-grams and their linkage with a given word's complexity.

Chapter 2

Technical Aspects

2.1 Datasets

We have used two datasets in order to train the FastText model and the Histogram-based Gradient Boosting Regression Tree: WordNet corpus [6] and CompLex dataset [5].

2.1.1 WordNet

Table 2.1: WordNet statistics

Words	Synsets	Word-Sense Pairs
155,327	175,979	207,016

2.1.2 CompLex

Table 2.2: CompLex statistics [5]

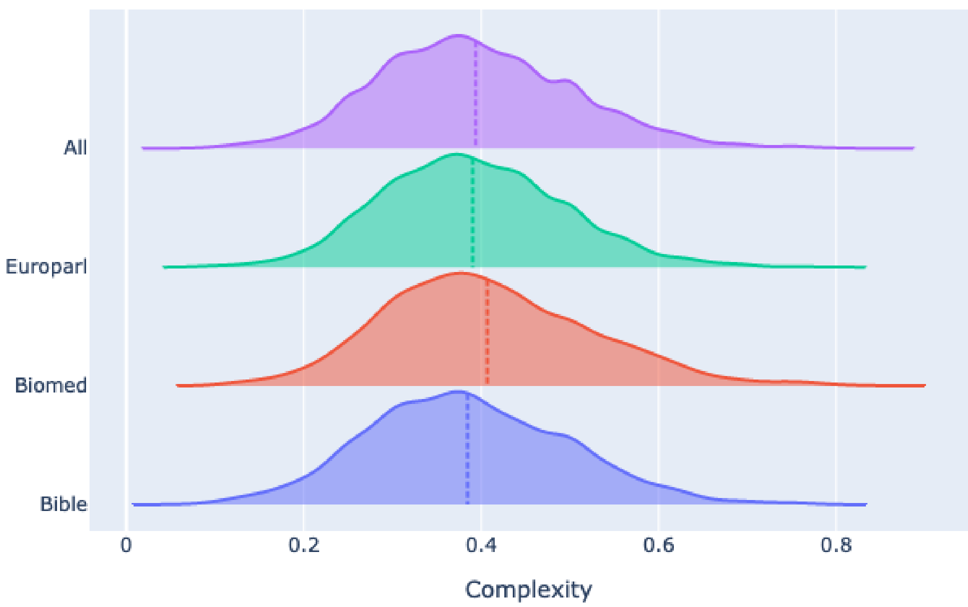
Source	Contexts	Unique Words	Complexity Mean	Complexity Standard Deviation
All	9,476	5,166	0.394	0.110
Europarl	3,496	2,194	0.390	0.101
Biomed	2,960	1,670	0.407	0.115
Bible	3,020	1,705	0.385	0.112

Table 2.3: CompLex statistics for single-word expression [5]

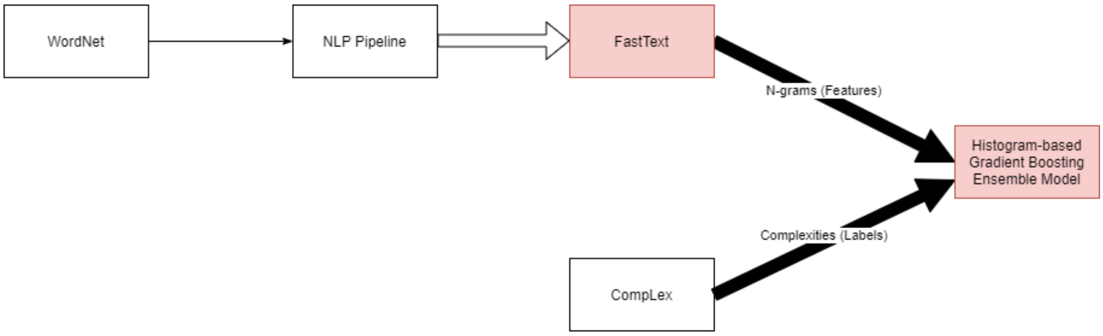
Source	Contexts	Unique Words	Complexity Mean	Complexity Standard Deviation
All	7,974	3,903	0.385	0.108
Europarl	2,896	1,693	0.381	0.100
Biomed	2,480	1,250	0.395	0.112
Bible	2,600	1,362	0.379	0.111

Table 2.4: CompLex statistics for multi-word expression [5]

Source	Contexts	Unique Words	Complexity Mean	Complexity Standard Deviation
All	1,500	1,263	0.442	0.105
Europarl	600	501	0.433	0.091
Biomed	480	420	0.470	0.109
Bible	420	343	0.442	0.112



2.2 Architecture



The model architecture consists of two distinct models: FastText and Histogram-based Gradient Boosting Regression Tree.

The FastText model trains on the WordNet corpus [6] in order to generate the n-grams for the respective tokens. The Histogram-based Gradient Boosting Regression Tree uses the word embeddings generated by the FastText model as features, and the complexities that are present in the CompLex dataset [5] as labels, after generating n-grams for the token associated with a given complexity in the dataset.

Chapter 3

Implementation and Results

As presented in the Architecture section, we have used a word embedding model and then the embeddings were used as features when training the prediction model.
In the following paragraphs we are going to discuss the approach we took for each model.

3.1 Word Embedding

3.1.1 FastText

In order to transform the tokens into n-grams, we have trained a FastText [2] [3] [4] model on the WordNet [6] corpus.
The hyperparameters used to train the model consisted of an initial learning rate of 0.1, which would linearly drop the 0.0001 during the training process, in order for the model to eventually reach a loss closer to the local minimum. We have also used hierarchical softmax [18] and CBOW and a minimum n-gram size of 1(unigram). The model was trained for 10 epochs.

3.2 Complexity Prediction

The n-grams that are being produced via the FastText model(features) are used to train the machine learning model, in combination with the complexities(labels) from the CompLex dataset. The training results can be seen in the table below. The models were tested on unseen data.

Table 3.1: Training results

Regressor	Mean MSE	Single-word expressions MSE	Multi-word expressions MSE
HistGradientBoostingRegressor	0.015212	0.013706	0.022718
ExtraTreesRegressor	0.016234	0.014948	0.022644
GradientBoostingRegressor	0.016256	0.014446	0.025278

3.2.1 Histogram-based Gradient Boosting Regression Tree

Using the Histogram-based Gradient Boosting Regression Tree, we achieved a minimum mean MSE of 0.015212 on both single and multi-word expressions. The single-word expressions MSE was 0.013706, while the multi-word expressions MSE was 0.022718.

The hyperparameters that we chose for the boosting regression tree were the following: a learning-rate of 0.1, on 100 maximum iterations, a maximum of 31 maximum leaf nodes number, without any regularization applied.

After using a PCA for dimensionality reduction on the test dataset, reducing the input variables dimensionality from 100 to 1, we have plotted the predictions achieved by the Histogram-based Gradient Boosting Regressor, as shown in 3.1.

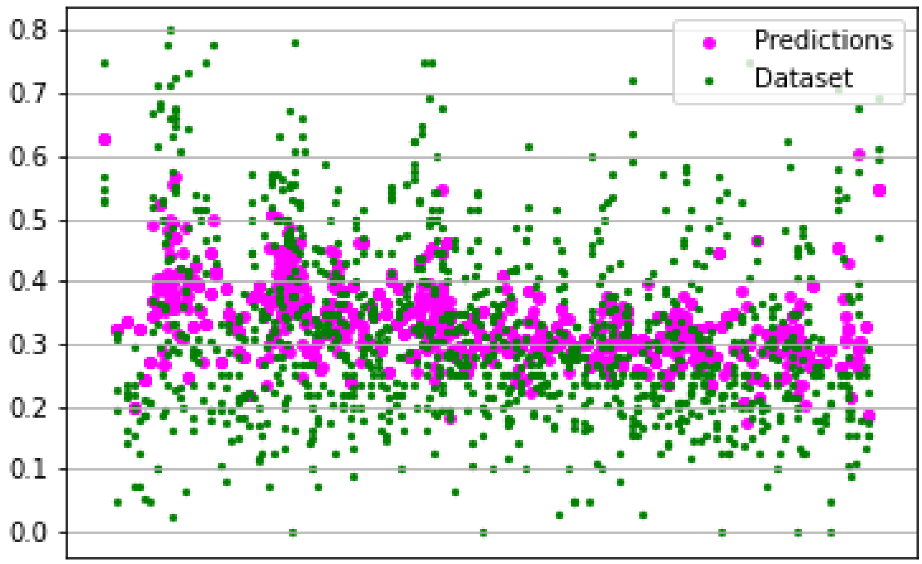


Figure 3.1: Histogram-based Gradient Boosting Regressor predictions

Chapter 4

Conclusion and Future work

4.1 Conclusion

To conclude, the study shows the effectiveness of chaining n-gram generation algorithms with machine learning methods, in order to predict lexical complexity for certain words.
The FastText model was an alternative for the Word2Vec, as the latter is incapable of generating n-grams for out-of-vocabulary words.
Also, our research serves as a proof that classic machine learning algorithms(e.g. HistGradientBoostingRegressor, ExtraTreesRegressor, RandomForestRegressor) provide better approximations for this particular task, when compared to neural networks. The HistGradientBoostingRegressor managed to achieve the optimal mapping between word n-grams and complexity.

4.2 Future work

Due to the small number of entries in the CompLex dataset, we suggest the usage of bigger datasets, in future studies. This way, it is likely that better generalization is going to be achieved. An issue with the current data was the fact that complexities were centered around a mean value of 0.4, following a normal distribution. Am uniform distribution might be a better choice for this particular task. Our results showed the fact that the regressors inclined towards predicting a value close to 0.4.

The previously stated fact could also reflect in the unfavorable loss reached during the training of the neural network. Even though we tried several architecture for the model, neither one of them could surpass a mean MSE value of 0.0237736.

References

[1] Aleksei Guryanov. Histogram-based algorithm for building gradient boosting ensembles of piecewise linear decision trees. In *International Conference on Analysis of Images, Social Networks and Texts*, pages 39–50. Springer, 2019.

[2] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*, 2016.

[3] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*, 2016.

[4] Armand Joulin, Edouard Grave, Piotr Bojanowski, Matthijs Douze, Hérve Jégou, and Tomas Mikolov. Fasttext.zip: Compressing text classification models. *arXiv preprint arXiv:1612.03651*, 2016.

[5] Matthew Shardlow, Michael Cooper, and Marcos Zampieri. 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, May 2020. European Language Resources Association.

[6] Christiane Fellbaum. *WordNet: An Electronic Lexical Database*. Bradford Books, 1998.

[7] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vander-

plas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[8] Radim Řehůřek and Petr Sojka. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May 2010. ELRA.

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

[10] Sian Gooding and Ekaterina Kochmar. Complex word identification as a sequence labelling task. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1148–1153, Florence, Italy, July 2019. Association for Computational Linguistics.

[11] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space, 2013.

[12] Drew Schmidt and Christian Heckendorf. ngram: Fast n-gram tokenization, 2017. R package version 3.0.4.

[13] Drew Schmidt and Christian Heckendorf. *Guide to the ngram Package: Fast n-gram Tokenization*, 2017. R Vignette.

[14] Shervin Malmasi and Marcos Zampieri. MAZA at SemEval-2016 task 11: Detecting lexical complexity using a decision stump meta-classifier. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 991–995, San Diego, California, June 2016. Association for Computational Linguistics.

[15] Sian Gooding, Shiva Taslimipoor, and Ekaterina Kochmar. Incorporating multiword expressions in phrase complexity estimation. In *Proceedings of*

the 1st Workshop on Tools and Resources to Empower People with REAding DIfficulties (READI), pages 14–19, Marseille, France, May 2020. European Language Resources Association.

[16] Menglin Xia, Ekaterina Kochmar, and Ted Briscoe. Text readability assessment for second language learners. In *Proceedings of the 11th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 12–22, San Diego, CA, June 2016. Association for Computational Linguistics.

[17] David Alfter and Elena Volodina. Towards single word lexical complexity prediction. In *Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 79–88, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

[18] Hao Peng, Jianxin Li, Yangqiu Song, and Yaopeng Liu. Incrementally learning the hierarchical softmax function for neural language models. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI'17, page 3267–3273. AAAI Press, 2017.

[19] C. Zhang and Y. Ma. *Ensemble methods: Methods and applications*. 01 2012.

Appendix A

Additional machine-learning algorithms training results

During our study, we have tested 45 total regressors for lexical complexity approximation. Table A.1 shows the results(MSE loss) on unseen data.

As it can be noticed, the best results were obtained by ensemble algorithms. In general, ensembles(HistGradientBoostingRegressor, ExtraTreesRegressor, RandomForestRegressor, GradientBoostingRegressor) can achieve better performances, when compared to singular models, as they reduce the dispersion of the predictions, as they rely on multiple components' predictions [19]. SVR's(SVR, NuSVR) were second best, when using a radial basis function kernel, while LinearSVR, which uses a linear kernel, performed much worse. Such differences in performance are caused by the shape of the dataset that is being used for training. The function associated with the data behaves similarly to a radial basis function. Figure A.1 shows the differences between SVR kernels, on a randomly generate dataset [7].

Linear models(RANSACRegressor, PassiveAggressiveRegressor, LinearSVR) came to the worst results, due to the same idea as previously stated, as the function that they are supposed to learn is not of linear nature. The only exception is the TheilSenRegressor, probably because of its lack of sensitivity to outliers. The lowest accuracy was achieved by the PLSCanonical, as its main purpose is dimensionality reduction by cross decomposition, and it is not supposed to obtain good results for predictions.

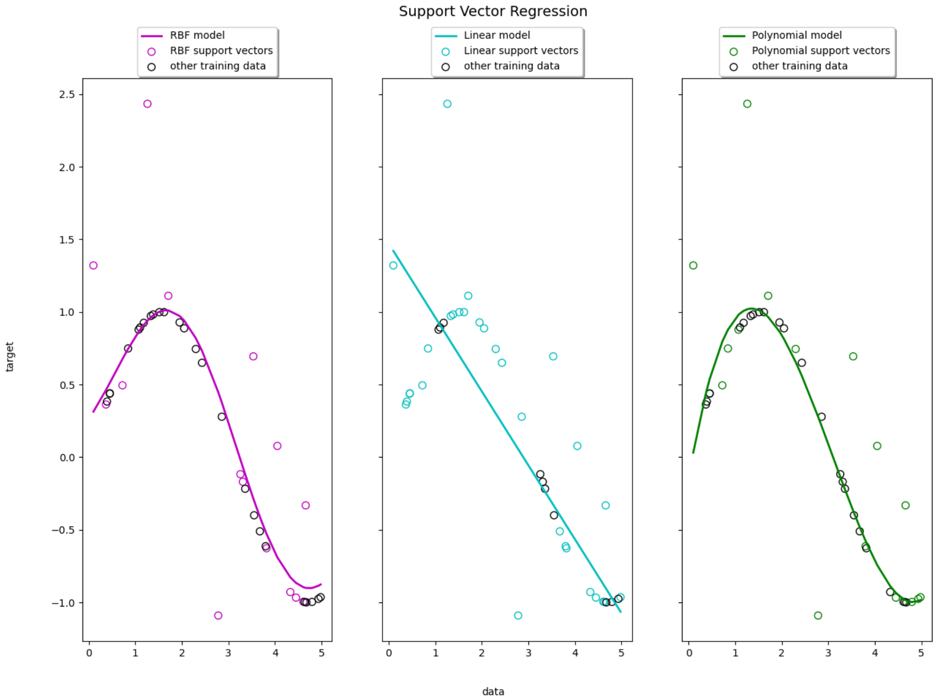


Figure A.1: SVR Kernels comparison [7]

Table A.1: Additional training results

Regressor	Mean MSE	Single-word expressions MSE	Multi-word expressions MSE
HistGradientBoostingRegressor	0.015212	0.013706	0.022718
ExtraTreesRegressor	0.016234	0.014948	0.022644
GradientBoostingRegressor	0.016256	0.014446	0.025278
RandomForestRegressor	0.016412	0.014893	0.023985
SVR	0.017308	0.015463	0.026501
NuSVR	0.017383	0.015485	0.026844
BaggingRegressor	0.018269	0.017102	0.024083
LassoLarsIC	0.018766	0.016749	0.028819
ARDRegression	0.018772	0.016758	0.02881
BayesianRidge	0.018798	0.01676	0.028957
OrthogonalMatchingPursuit	0.018876	0.016595	0.030244
LinearRegression	0.018942	0.016994	0.028643
TransformedTargetRegressor	0.018942	0.016994	0.028643
LassoLarsCV	0.019183	0.016727	0.031422
LassoCV	0.019197	0.016819	0.031047
ElasticNetCV	0.019206	0.016829	0.031054
RidgeCV	0.019266	0.016876	0.03118
LarsCV	0.019331	0.016674	0.032573
HuberRegressor	0.019359	0.017132	0.030456
OrthogonalMatchingPursuitCV	0.019365	0.017055	0.030877
MLPRegressor	0.019386	0.016871	0.031918
Ridge	0.019446	0.016799	0.032634
AdaBoostRegressor	0.019672	0.018782	0.024108
SGDRegressor	0.01973	0.016736	0.034653
DummyRegressor	0.019739	0.016816	0.034305
ElasticNet	0.019739	0.016815	0.034304
Lasso	0.019739	0.016815	0.034304
LassoLars	0.019739	0.016816	0.034305
PoissonRegressor	0.019739	0.016816	0.034305
RadiusNeighborsRegressor	0.019739	0.016816	0.034305
TweedieRegressor	0.019739	0.016816	0.034305
LinearSVR	0.019778	0.016512	0.036054
Lars	0.019808	0.017757	0.030027
PLSRegression	0.020013	0.017764	0.031223
Neural Network	0.02002	0.0176079	0.032042
TheilSenRegressor	0.020356	0.019563	0.024311
KNeighborsRegressor	0.021856	0.020649	0.027872
KernelRidge	0.025424	0.019771	0.053595
CCA	0.028219	0.027899	0.029816
DecisionTreeRegressor	0.032261	0.029591	0.045565
ExtraTreeRegressor	0.033883	0.031579	0.045363
PassiveAggressiveRegressor	0.222653	0.235524	0.158509
RANSACRegressor	0.312742	0.342366	0.165106
PLSCanonical	0.771532	0.829382	0.483223