Calculating Latin Readability Scores Using Linear Regression

Amanda Lin
High Technology High School
MehtA+Tutoring

Surya Kolluri
Dublin Jerome High School
MehtA+Tutoring

Dylan Sheehan
Lyme Old Lyme High School
MehtA+Tutoring

August 4, 2021

Abstract

Readability scores are important in order to help students and teachers determine which texts are best for them to read. Although many such scores exist for English, there are no commonly used scored for Latin. Using a labeled dataset with associated LexR scores, machine learning and linear regression is used to predict the readability scores for additional Latin texts. The four different linear models tested all produced similar accuracy scores and had comparable results.

1 Introduction

Readability metrics have been used by teachers to select the best texts for students to read. Metrics such as Lexile scores are commonly used with English texts to determine difficulty. However, such scores do not translate across the language barrier between English and Latin. As such, there are currently no widely-used metrics that can help an instructor determine which Latin texts are best suited for a particular student.

2 Related Work

Machine learning, including the usage of SVM models has been previously used to determine the reading level of an English text [1, 2]. Additionally, classification methods for the reading levels of English texts have been shown to have some applicability for French texts [3]. However, none of these previous studies have dealt with the Latin language and its associated eccentricities.

3 Methodology

3.1 Dataset

The dataset that was used was part of the Bridge corpus. It was provided by Professor Bret Mulligan of Haverford College. It included 275 Latin texts, textbooks, and vocabulary lists. There was also a spreadsheet with the LexR scores of 631 texts along with the titles of the texts.

The texts were stored as Python files within a Github repository. Each Python file contained a list of tuples, with each tuple containing the data associated with one word. Index 0 of each tuple was the lemmatized word, index 1 was the location of the word within the text, and index 2 was the inflected word itself, as it appeared in the text. A word is lemmatized when all inflected forms of the word are grouped together, in order to be analyzed properly.

One challenge that we faced with the dataset was that the data provided with the associated word in the tuples was inconsistent across the texts. While all of the tuples had 6 indices, some of the texts had empty strings in the tuples. We only utilized the 3 indices mentioned above, as those appeared consistently across those texts.

Another challenge was that the titles of the texts on the spreadsheet did not always match up to the Python file names. Additionally, not all files had LexR scores, and not all texts were within the Github repository.

The Github repository was cloned, and all of the files were imported. The file names and the imported modules were stored in a dictionary, with the file names as the key. Some files had periods within the file names, and that caused problems upon importing, and as a result, had to be imported separately. The periods within the file names were replaced with underscores.

3.1.1 Preprocessing

There was a plethora of preprocessing associated with the dataset. It was necessary to extract the text and any important information from the Python files, link the titles associated with the LexR scores to the file names, and extract the relevant features from the text files.

Python Files The Python files in the folder were iterated through and the list of tuples that contained the texts was accessed. The LamonPy library was utilized to find the parts of speech and other related information such as cases, verb tenses, and the genders of a noun or adjective. The words in each text, along with the data provided by LamonPy, were stored as Pandas dataframes and saved as CSV files.

Feature Extraction Key features from each of the CSV files associated with the texts were extracted and stored as a Pandas dataframe. These features included average word length, the distribution of the parts of speech, the cases of nouns and adjectives, and verb tenses. Additionally, the usage of common words was found by using the Diederich Frequency List for Latin, which was one of the files within the provided Github repository. The Pandas dataframes for all of the texts were concatenated and saved as a CSV file.

LexR Scores The spreadsheet with the LexR scores was processed, and any duplicates were removed. Each title was parsed and converted into a set of words. Previously, the words in the titles were separated by underscores, capital letters (camel case), or a combination of the two. This made it easier to process the text. The file names of the CSV files were also parsed. The sets of words were compared, and if there was a high similarity between the sets, or one set was a subset of the other, then the CSV file names and the titles of the texts were linked to each other with a dictionary. Then, the dictionary was manually checked to ensure that the file names and titles were linked properly. A Pandas dataframe was created with the title of the CSV file and the LexR score associated with it, which was then saved as a CSV file. There were, in total, 104 texts with LexR scores.

3.2 Model

The LexR scores were used as labels, and the features extracted from the CSV files were used as the features in the creation of our models. The data was split 80/20 into training and testing data. The random state for this split was 25.

Different types of linear regression models from the Scikit-Learn Python library were used in order to determine which one had the highest accuracy. These models include:

- Ordinary least squares linear regression
- Ridge regression
- ElasticNet
- ElasticNetCV

4 Results

To optimize accuracy scores, the parameters of the models were modified. It was discovered that a smaller alpha value corresponded to a greater accuracy score. This can be seen in Table 1. The alpha value is the regularization strength, with larger values meaning a larger regularization strength, which would reduce the variance of the predicted values.

Additionally, for the two ElasticNet models, the l1 ratio and the maximum number of iterations were changed. This changed the combination of the l1 and l2 regularization used. The maximum iterations were changed because it was found that a fewer number of iterations did not allow the models to converge.

As per Table 2, the accuracy scores of the different models were quite similar, with only small differences between

Table 1: Varying Alpha Values with Ridge Regression Accuracy Scores

Alpha Value	**Accuracy Score**
10^{-1}	0.5050
10^{-2}	0.6291
10^{-3}	0.6450
10^{-4}	0.6932
10^{-5}	0.7219
10^{-6}	0.7276
10^{-7}	0.7282
10^{-8}	0.7283

them. The only outlier is the ElasticNetCV model. This can also be seen in the predicted values for the LexR scores, as shown in Table 3. The Linear regression, Ridge regression, and ElasticNet models all output similar LexR values for the selected text, Apuleius Florida, that are very close to the actual LexR value, whereas the ElasticNetCV model remains an outlier.

Table 2: Accuracy Scores of Various Models

Model	**Accuracy Score**
Linear Regression	0.728
Ridge Regression	0.728
ElasticNet	0.725
ElasticNetCV	0.667

Table 3: LexR Values of Apuleius Florida

Method	**LexR Value**
Linear Regression	7.997
Ridge Regression	7.997
ElasticNet	7.972
ElasticNetCV	6.377
Actual	8.040

5 Conclusion and Future Work

In general, the models all produced very similar results. The accuracy scores and the LexR values were consistent across all of the models, with the exception of the ElasticNetCV model. The linear regression and the ridge regression models, with the alpha value that was chosen, were the same across all of the metrics we measured.

Additional research, such as adding more features or testing different models, should be done in order to better implement linear regression with the readability scores of Latin texts. This could potentially increase accuracy scores. Features that could be added include more information on word frequency.

Additionally, models for the readability scores of different languages could be compared to visualize the similarities between languages. Similar languages potentially would have similar models for readability scores.

6 Division of Labor

We divided the work as follows:

- Researching Related Work: Amanda L, Dylan S
- Importing Python Files: Surya K
- Preprocessing Python Files: Amanda L, Dylan S
- Preprocessing LexR scores: Amanda L, Surya K

- Feature Extraction: Amanda L, Dylan S, Surya K
- Model: Amanda L, Surya K
- Poster: Amanda L, Dylan S, Surya K
- Paper: Amanda L, Dylan S, Surya K

7 Acknowledgements

We would like to acknowledge Haverford College, including the team working on Bridge and Professor Bret Mulligan; MehtA+ Tutoring and the instructors, Haripriya Mehta, Bhagirath Mehta, Mohammad Sharafat, Andrea Jaba, and Marwa Al Alawi; and our friends and family who support our work.

References

[1] Sarah E. Petersen and Mari Ostendorf. A machine learning approach to reading level assessment. *Computer Speech & Language*, 23(1):89–106, 2009.

[2] Sarah E. Schwarm and Mari Ostendorf. Reading level assessment using support vector machines and statistical language models. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, page 523–530, USA, 2005. Association for Computational Linguistics.

[3] Kevyn Collins-Thompson and James P. Callan. A language modeling approach to predicting reading difficulty. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics: HLT-NAACL 2004*, pages 193–200, Boston, Massachusetts, USA, May 2 - May 7 2004. Association for Computational Linguistics.