Read Time Prediction

Abraham George, Ruchita Sinha, Badal Arun Pardhi
Carnegie Mellon University

Abstract

The goal of this project is to estimate the time it takes to read a given piece of text. This objective is achieved using traditional Machine Learning algorithms with a focus on feature extraction and modeling. To improve the accuracy of the results, various data analysis and data processing techniques were implemented, along with the introduction of complex features created from the input data. Using the processed data, the accuracy of shallow machine learning algorithms such as linear regression, polynomial regression, support vector regression, random forest regression, and linear regression with regularization, were calculated. Overall, the highest performing algorithm was support vector regression, achieving an outstanding 95.2% *test accuracy.*

1. Introduction

This project intends to accurately estimate the time it will take for an average reader to read a given piece of text, thereby solving a real world problem that the literature industry is facing in the modern era. Companies such as Newspaper Publishers and e-Book Publishers can benefit a great deal from the successful implementation of a read time prediction algorithm, with accurate read times for each article allowing their customers to their reading schedule more effectively.

The data for this machine learning implementation comes from audio books and text samples collected from a public repository. In total, approximately 500 data samples were collected, containing text scripts along with the read times associated with those scripts, in addition to significant metadata information such as genre, year of publication, and country of origin.

The collected data samples were then analyzed to extract additional features associated with the text scripts. The processed data was then fed to a variety of supervised machine learning algorithms to compare their performance. The focus of this project is to analyze how feature engineering techniques can affect the performance of different Machine Learning algorithms. To compare the results, absolute percentage error was used as an evaluation metric. Support Vector Regression was observed to have the lowest mean percent absolute error in predicting reading speed.

The final implementation of this model will take in a text script along with its genre, country of origin, and year of publication as input. The data will then be processed and fed to a support vector regression model to calculate the time it takes for an average reader to read that excerpt out loud.

2. Related Work

During the implementation of this idea, several research papers were referred to. Researchers have made significant progress in accurately predicting the read time it takes to read a single word. However, the accuracy reduces to a measurable extent if a text script is analyzed for it's reading time instead of a one single word. Further research pertaining to readability measures the difficulty of reading a text excerpt. A commonly used readability metric for English is the Flesch-Kincaid Score (FK Score). The FK Score uses the number of syllables and words present in a script to calculate the readability metric. A study titled “How many words do we read per minute?” compares the average reading speeds for different age groups for fiction and non-fiction text [1]. This also gives us an intuition on how the reading speed may vary if the text is read out loud or in mind (silent reading). Another study titled “You Don’t Have Time to Read This” explains the use of slightly complex features in a document such as font size, readability metrics, and subject matter to build a prediction model with increased data size [5]. Additionally the project also takes reference from an interesting study titled “Can AI Predict Reading Time Based on the Words in the Text?” [2]. The paper uses Linear Support Vector Machine to determine words in a text script that contribute the most and the least to the reading time.

This project uses similar readability metrics to influence the selection of features for the machine learning model implementation. In addition to the existing metrics it also tries to include the effect of three important metadata features, genre, year of publication, and the country of origin, that can significantly affect read time of a text script.

3. Data

3.1. Data collection

Because no existing data set could be found, the data was collected manually. The data set came from Lit2Go, a free online collection of public domain literary works. The site breaks down books by chapter, with each chapter given a webpage, featuring the text of that chapter, a public domain audio recording of the chapter being read, and information about the book the chapter is from, including year of publication, country of origin, and genre. In total, data was collected from 500 chapters.

For each data point (one chapter) the entire text of the chapter, the book meta-data, including the year of publication, country of origin, and the genre of the book, and the total read time of the chapter, which was measured using the provided audio recording, was collected. During the data collection process, the name of the book, the title of the chapter, and an index number were also recorded, to serve as a reference for each data point.

In order to ensure diversity and balance in the data set, the data collection was evenly divided across 50 books, with 10 chapters sampled from each. This split ensured that the data set was not skewed by one book or one narrator. Additionally, to ensure an even distribution of genres, 50 data points (5 books) were collected from each of the 10 most common genres on the Lit2Go website.

3.2. Feature extraction

Once the data was collected, feature extraction began. A python script was created to go through the text from each data point and calculate a series of features. The program calculates the number of paragraphs, number of words, number of characters, number of syllables, number of periods, number of exclamation points, number of question marks, number of dashes, number of commas, number of semicolons, and number of colons in the text.

In addition to counts, measurements of complexity were taken into account. First, the text's Flesch-Kincaid readability score was calculated, which is based on the number of words per sentence and the number of syllables per word in the text and provides a rating of how easy the text is to read [4]. Additionally, complexity is determine by calculating the percentage of words in the text that are from the 100, $1,000$, and $10,000$ most common English words, as determined by Google's Ngram Viewer[3]. The genre and country of origin data was also processed, being converted from labels to one-hot-encoded values. The date of publication was left unchanged.

4. Methods

4.1. Feature Engineering

4.1.1 Scaling

After feature extraction was complete, a serious issue arose with the data set: there were two distinct classes of features. Some of features, such as number of words, were a function of read time, (i.e. if the length of the text is doubled, the number of words doubles, and the read time doubles), while other features, such as the complexity score, were a function of read speed (i.e. if the length of text is doubled, the reading rate (words per minute) remains constant, and so does the complexity).

Another issue was that the data set included chapters of wildly different lengths, ranging from 2 minutes to 40 minutes. The solution to both of these problems was to fit the data for reading speed rather than reading time. This approach not only would convert all of the 'reading time' features into 'reading speed' features, achieving feature compatibility, but also would ensure all of the data was roughly the same size. To scale the data, a measurement of the length of the text was needed. Theoretically, the ideal approach would be to scale everything by read time, but read time is the dependent variable, so it cannot be used. Instead, a close approximation was used, specifically an independent variable that is strictly proportional to read time. For an independent variable to be proportional to the read time, [1] equation must hold true where C is a constant

$$Feature = C * ReadTime \tag{1}$$

Therefore, if a feature is proportional to read time, the value of C should remain constant across all of the data points. So, by finding which feature has the most constant value of C (the ratio of the feature value to read time) across all data points, the feature that is best suited to be used as the read time to read speed scalar can be identified. The coefficient of variation (standard deviation divided by the mean) of the C values for each of the proportional features across all data points was calculated and plotted as shown in figure 1. Looking at the results, the feature with the least variation is the number of syllables. Therefore, the features proportional to 'read time' were scaled by the number of syllables in each data point, as was the read time.

4.1.2 Z normalization

Once the scaling process was completed, the data was normalized using Z-normalization, and was finally ready for use in machine learning.

4.2. Machine Learning model selection

Once the data was fully processed, machine learning models were carefully chosen to use to fit the data. Details

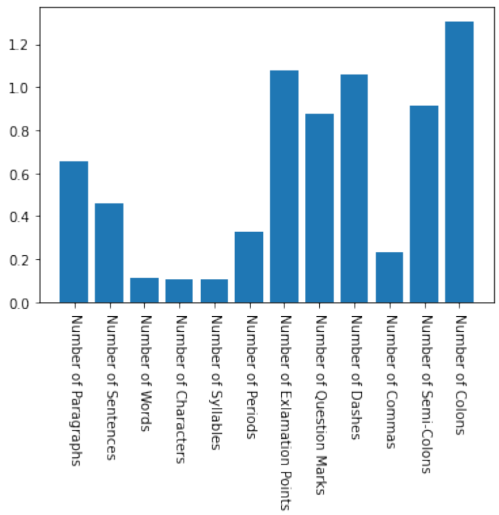

Figure 1: Coefficient of Variation of each Feature

on which models were chosen, along with the motivation behind why they were chosen, is presented in this section.

4.2.1 Linear Regression and Regularization

The first machine learning technique chosen for the project was linear regression, a statistical regression technique. Linear regression is the simplest regression machine learning model since it predicts a linear relationship between the independent features and the dependent feature, making it an effective model for supervised learning regression. To further improve the accuracy, $L1$ (lasso) and $L2$ (Ridge) regression were also included. These regularization techniques work by penalizing the weights associated with higher loss values. Because this project uses a linear regression model to predict a simple linear fit between the features and the label, $L1$ and $L2$ regularization are a useful tool to improve the prediction of linear regression.

4.2.2 Polynomial regression

This ML algorithm is similar to linear regression, however, the best fit is not linear but a curve of best fit, allowing the regression model to fit non-linear data sets. Polynomial regression was included in order to address any potential non-linearities in the data set.

4.2.3 Support Vector Regressor (SVR)

Support Vector Machine is typically used for classification problems. For regression problems, Support Vector Regression (SVR) is used. This works similar to linear regression model, where SVR is able to find a hyperplane fitting the training data and the error is defined by the margins. SVR is a more robust model for predicting the outcome for data with multiple features and hence gives a more accurate model than linear regression, making it a vital machine learning tool to include in the read time analysis.

4.2.4 Decision trees and Random forest regression

Decision trees can be used for classification and regression. Decision trees gives a more intuitive method to visualize the data and can work even with some missing data. Since decision trees are prone to overfitting, the random forest regressor reduces the error due to variance and bias. The project uses and then subsequently compares both the algorithms.

4.2.5 Multi Level Perceptron

MLP has been used to compare the performance of shallow machine learning algorithms with neural networks. Based on the amount of data that has been collected neural networks such as the MLP are expected to underperform in comparison to the other supervised machine learning algorithms.

4.3. Error calculation

Percent absolute error was used to determine the accuracy of the models. This metric was chosen instead of mean squared error because it helps to account for the wide range of readtimes in the dataset. Additionally, the percent error of the read time (seconds) and the percent error of the scaled output (seconds per syllable) are the same, since the number of syllables is an independent variable. This aspect is useful in the model evaluation process, because it means that the read time per syllable values do not have to be converted back to read time values in order to evaluate the results.

5. Experimentation

5.1. Implementation

After selecting these models, the next step was implementing them. The dataset was split into training and testing data in a $70 : 30$ ratio. Once the data was split, the training data was fed into each of the machine learning models that were selected. The models were implemented using the sklearn module in python. The percent absolute error of each model, for both the train and test datasets, was recorded. This process was repeated five times, each time with a different random train test split, and the results were averaged together to get a more robust accuracy measurement. Finally, the errors for each model were collected and compared using a bar chart, as explained in the results section.

5.1.1 Linear regression and regularization

Linear regression module in sklearn uses the ordinary least square method. This algorithm predicts a best fit line by minimizing the sum of square error (SSE) and therefore is a good fit given the small size of the dataset. Sklearn then fits a line through the training dataset to get a prediction and then tests the prediction with the testing set. To further improve the model, linear regression with regularization was also used. The first type of regularization used was Lasso or L1 regression, which modifies the ordinary least square method to penalize the terms with large SSE by minimizing the absolute sum of weights with high SSE. Similarly, Ridge or L2 regression was also implemented. This form of regularization modifies the ordinary least square method to penalize the terms with large SSE by minimizing the squared sum of weights with high SSE.

The regularization term has a hyper parameter, α, which adds a weight to the penalizing term. These α values were tuned for $L1$ and $L2$ regression to achieve the lowest possible testing error. Figures 2 and 3 show how the percent absolute error varies with α values. Figure 2 for $L1$ regression shows the error for training and testing data for tuning different α values. The testing error for alpha between $1.7e-05$ and $4.6e-04$ is fairly constant at about 0.07. Looking closely at the values, a slight dip before the increase in testing error can be seen, and the optimum value of α (corresponding to the lowest testing error) was found to be 0.00018. Similarly for $L2$ regression, figure 3 shows that the testing error for α between 0.1 and 10 is fairly constant at about 0.07, although the training error begins to rise. By closely examining the testing error, the minimum value was found to occur at an α value of 6.5.

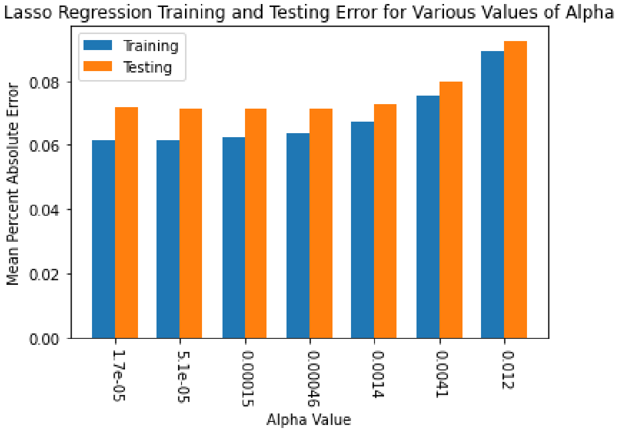

Figure 2: Alpha Optimization for Lasso Regression

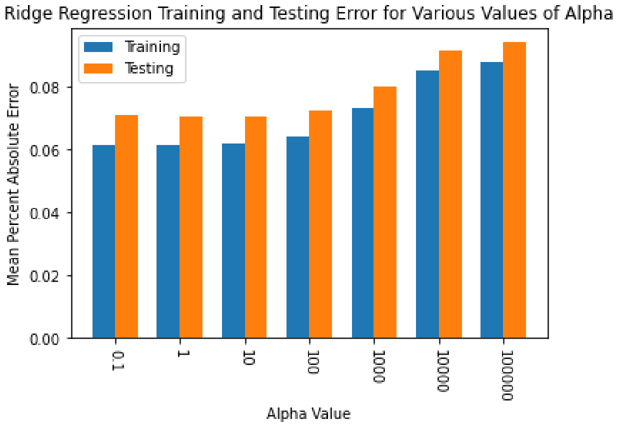

Figure 3: Alpha Optimization for Ridge Regression

5.1.2 Polynomial regression

Polynomial regression fits a curve through the training data to predict the best fit. Since there are 37 features, expanding them all into a polynomial expression will result in a very large number of terms. Fitting all of these features would not only overfit the model but also increase the computational time. To overcome these issues, the number of terms in the polynomial was reduced using principal component analysis (PCA). To determine how many features to reduce the data set to, the cumulative explained variance was plotted over the output dimension, as seen in figure 4. PCA dimensions were then selected so that 90% of the variance was explained. In this case, the number of output dimensions chosen was 17, meaning the polynomial regression only has 17 features to expand, instead of the the original 37 features.

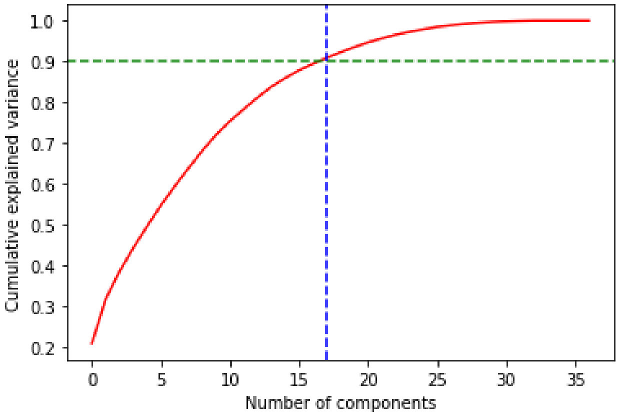

Figure 4: PCA Explained Variance

Similar to linear regression, polynomial regression in python is performed using the linear model in sklearn with the order of polynomials specified. Running and tuning the model shows that a fourth order polynomial is a good fit for the data.

5.1.3 Support Vector Regressor

Support vector regression was implemented using the SVM module in sklearn. SVR performs a $L2$ regularization to get the best hyperplane and computes the error margin.

Both of these tuning functions require hyperparameters. C is the penalty associated with $L2$ regularization which varies inversely with the strength of the regularization. C was tuned to 1. ϵ is the hyperparameter used to determine the size of the margins. Figure 5 shows how the absolute percentage error increases by increasing the ϵ. The testing error is fairly constant between $9.8e-04$ and $1.6e-02$ but the training error increases. An ϵ value of 0.01 was chosen to minimize the testing error in the model.

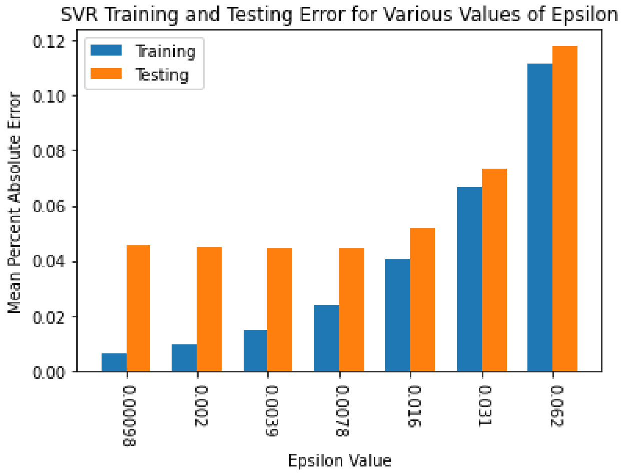

Figure 5: Epsilon Optimization for SVR

5.1.4 Decision tree regressor and Random forest regressor

Each of these algorithms use sklearn to implement the algorithm. Decision tree regressor uses the tree module in sklearn while random forest uses the ensemble module. Decision tree regressor and Random forest regressor both have feature importance which shows how important each feature is in predicting the read time.

5.1.5 Multi Layer Perceptron

Implementation of MLP involved 2500 neurons in the hidden layers which were activated by the Tanh function, optimized using the "Adam" solver, had a learning rate of 0.001, and maximum iterations was set to 200. These hyperparameters were tuned using trial and error methods.

5.2. Results

The eight machine learning algorithms were implemented on the scaled normalized data set in the manner discussed above, and the train and test accuracies for each method were plotted, and can be seen in figure 6. By analyzing this graph, insights into our data, the nature of the problem, and the characteristics of each machine learning algorithm can be found.

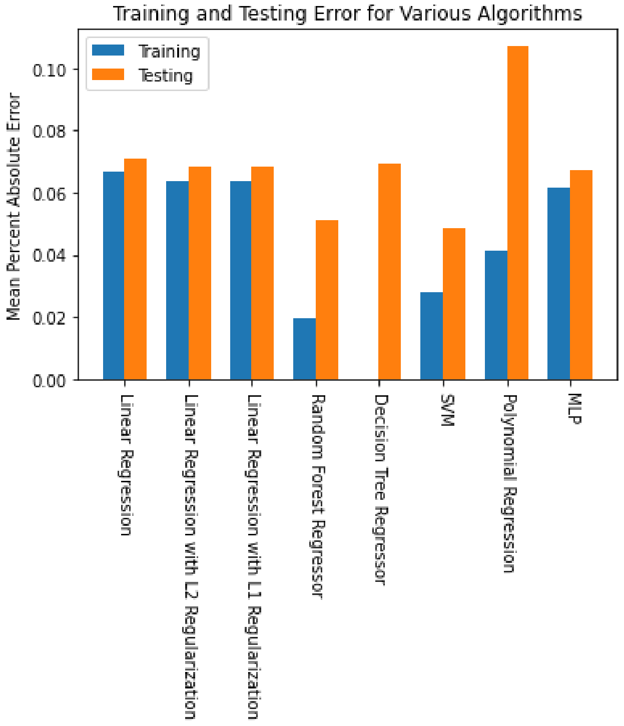

Figure 6: Model Evaluation

Beginning with linear regression, all three forms of linear regression performed quite well, with a test error of around 7%. Interestingly, the regularized models performed almost identically to the unregularized model, even though both regularization techniques had their alpha term optimized. This behavior suggests that even though there are 37 features used, the linear model is not overfitting, suggesting that the feature extraction was not overparameterized.

The other form of regression, polynomial regression, had the highest error out of all of the machine learning models. Even though the regression was optimized with $L2$ regularization, the model significantly overfit the data, with a much higher testing error than training error. In addition to the high overfitting, the abysmal error of the polynomial model can be traced back to the linearity of the data set. In exchange for overfitting, and a reduction in accuracy due to the use of PCA, the polynomial model gains the ability to fit nonlinear data. This means that if the data is linear in nature, then then polynomial regression makes these sacrifices without receiving any significant improvement in performance, which appears to be what happened in this case. Interestingly, the failure of the polynomial model sheds light on the nature of the read-time prediction problem. Specifically, these observations suggest that features are mostly linear in nature, which fits with how well linear regression performed.

Moving onto Random Forest, this model performed the second best, with a testing error of 5.1%. The random forest did overfit a significant amount, with the testing error measuring over twice the training error, but such overfitting is fairly common for decision tree based models. On the subject of decision trees, the single decision tree fit worked surprisingly well, with a test error similar to linear regression. However, because this model features only a single tree, the prediction completely overfit the data, with zero testing error. Once again, this level of overfitting was expected.

The best method overall was SVR, with a testing error of only 4.8%. This result is especially impressive when compared to the accuracy of the neural network model, which had a testing error of 6.7%, comparable to the linear regression models. The superiority of SVR to the neural network in this application is a great illustration of the limitations of deep learning models. In situations where the data size is relatively small (500 points in this case) the neural network simply does not have enough data to train on, and as such will return mediocre results and underperform shallow machine learning techniques.

Once the eight machine learning algorithms were evaluated, an additional question was explored: how important was the scaling of the data by the number of syllables? If the features were left in terms of read time instead of read speed, would there be a significant impact on accuracy of the models? To address these questions, the accuracy of a select few models was tested with both scaled and unscaled features. Z-normalization was performed on both datasets and the same hyper parameters were used. The results of this test can be seen in figures 7 and 8. Looking at the graphs side by side, the advantage of scaling can clearly be seen, with a decrease in all five models. Although this improvement isn't extreme, it is significant, with around a 20% reduction in error in the linear models, and similar improvements in the others. Clearly, scaling the 'read time' features by the number of syllables was an effective strategy.

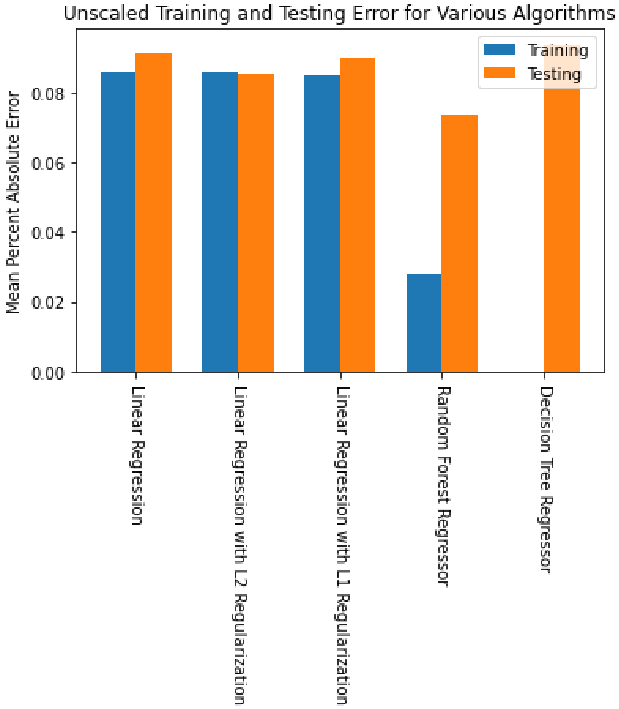

Figure 7: Model Evaluation for unscaled features

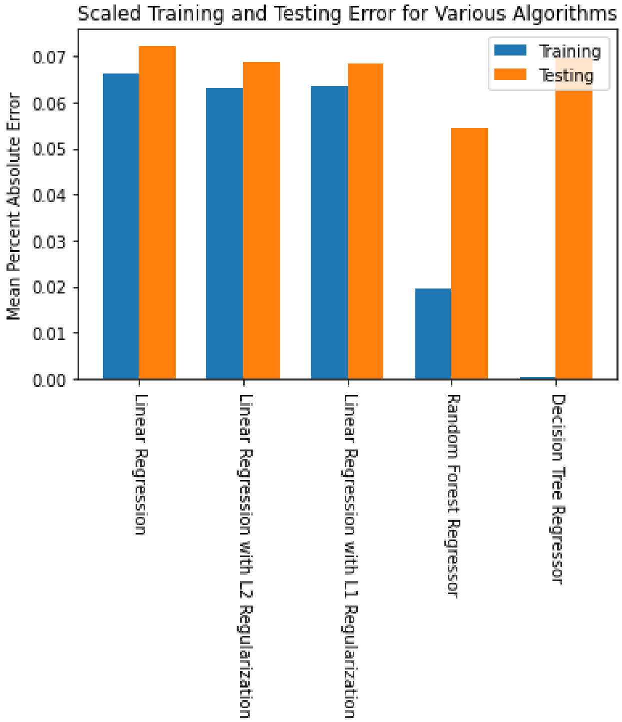

Figure 8: Model Evaluation for scaled features

6. Conclusion

After performing careful unbiased data collection, data analysis, feature extraction, and feature engineering, and testing out a variety of machine learning models, multiple algorithms have been successfully created to accurately predict the amount of time a piece of text will take to read. Of these models, the one based on support vector regression gives the best results, yielding a test accuracy of 95.2%. Given a sample of text, along with the metadata such as genre, country of origin, and year of publication as an input, the model will predict the time it takes for an average user to read that script.

7. Group member contribution

The work was divided between all members equally. There were 500 data points to be collected. Each member collected about 160 - 170 data points. Each member worked on feature extraction and then implemented 2 algorithms each.The other work including project proposal, presentations and report was divided evenly between all group members.

References

[1] M. Brysbaert. How many words do we read per minute? a review and meta-analysis of reading rate. *Journal of Memory and Language*, 109:104047, 2019. 1

[2] H. Buseyne. Can ai predict reading time based on the words in the text? an introduction to the use of artificial intelligence on text. *Twipe Digital Publishing*, 2018. 1

[3] J. B. D. M. Josh Kaufman, William Hingston. Google 10000 words. `https://github.com/first20hours/google-10000-english.git`, 2021. 2

[4] Readable. Flesch Reading Ease and the Flesch Kincaid Grade Level, url = https://readable.com/readability/flesch-reading-ease-flesch-kincaid-grade-level/,. 2

[5] O. Weller, J. Hildebrandt, I. Reznik, C. Challis, E. S. Tass, Q. Snell, and K. Seppi. You don't have time to read this: An exploration of document reading time prediction. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1789–1794, Online, July 2020. Association for Computational Linguistics. 1