# Predicting Text Difficulty

SIADS 694/695 Milestone II Project Report
by James Mete & Matt Dannheisser
Kaggle Team: JM_MD_Team9

## Introduction:

This work aims to classify the textual complexity of sentences from the Simple English Wikipedia edition. We built a machine learning pipeline that used an evolving set of supervised models aided by processes of feature engineering and unsupervised learning to classify sentences into whether or not they require simplification. Such a task is useful in domains such as Simple Wikipedia that aims to present simplified articles that are easier to understand, especially for those who have "limited abilities"[1] in comprehension of the English language.

We utilized the "Predicting Text Difficulty" challenge on Kaggle as a framework for structuring our process and setting goals of our research. Our primary goal is to reach a 0.8 accuracy score proven possible by previous iteration's submissions.

Our key findings were the following:

1. Visual exploration of data revealed that the training data labels are perfectly balanced.
2. PCA transformation revealed the training data can be generally separated into simple and complex sentences, but heavy overlap in clusters exists.
3. Topic modeling revealed similar topics throughout the data, regardless of complexity label. Topics alone were not a significant indicator of text difficulty.
4. Clustering analysis revealed that a 2-cluster KMeans model performed best which correlates to our goal of binary classification. Our KMeans labels were 61% accurate compared to the training labels which suggests that our model is picking up on a true signal considering the dataset is balanced.
5. We trained multiple machine learning models **, and observed best performance from BERT fine-tuned on our training data & engineered features which achieved an accuracy score of 77.6% on the test dataset** provided by Kaggle which is the second highest score on the private leaderboard.
6. Deep learning models were quick to overfit and sensitive to hyper parameter tuning.

## Data Exploration:

Our major dataset were the files presented by the Kaggle challenge, but we added additional sources to help our feature engineering efforts. The data sources mentioned were used for both the unsupervised and supervised sections of the project. Specifically, the files we used were:

**WikiLarge_Train.csv** [Kaggle]:
Training dataset containing 416,768 sentences from Wikipedia that also contains labels. 0 refers to sentences that are sufficiently simple, and1 refers to sentences that require review for simplification. **Our exploration of the training data revealed that the data is balanced in terms of labels**. This makes classification accuracy more reasonable, but we decided to use both accuracy and F1 scores into consideration for extra information in model evaluation.

[1] Limited abilities is defined as "people with different needs, such as students, children, adults with learning disabilities, and people learning English" (Simple English Wikipedia, 2009).

**WikiLarge_Test.csv** [Kaggle]:
Test dataset with 119,092 sentences without labels. Predicting the labels of this dataset is the main evaluation task based on the Kaggle challenge. The evaluation metric used is classification accuracy.
**Dale_chall.txt** *[Kaggle]***:**
List of 3,000 words that are understood in reading by at least 80% of 5th graders. A sentence primarily composed of these words is most likely to be a simple sentence.
**SAT_words.csv** *[External]***:**
This file is an external CSV resource we collected to complement the existing files. This file contains 6,000 SAT words which are considered more difficult. Our intuition was that words with more SAT words would be more likely to be considered complex and thus require simplification.
**AoA_51715_words.csv** *[Kaggle]***:**
This file contains Age of Acquisition (AoA) estimates for about 51k lemmatized English words with corresponding metrics about the average age the lexical meaning is acquired. We selected the age of acquisition (AoA) of the lemmatized word and the frequency of use in general English as features. The former represents the average perceived difficulty of understanding the word and the latter represents the obscurity of the word through lack of use. We were selective of which features to train the model on as we were attempting to avoid adding too much noise.
**Concreteness_ratings_Brysbaert_et_al_BRM.txt** *[Kaggle]***:**
This file contains "concreteness ratings" for 40 thousand English lemmatized words known by at least 85% of raters. We selected the mean concreteness rating to demonstrate the ambiguity associated with these known words. The concreteness word list was a narrower and better-known list of words compared to the Age of Acquisition list. The concreteness score thus was focused on different types of ambiguity which added quality to our features.

## Feature Engineering:

Our text data alone is not enough to generate highly robust classifiers, at least using traditional machine learning models. Furthermore, our unsupervised methods revealed heavy overlap in terms of clusters and topics which increased the need for better features to help separate the simple from the complex sentences. Thus, **we engineered 30 features utilizing all of our available resources.**

We used a machine learning cycle framework to generate our features. We would explore the data, develop unsupervised or supervised models to test the data further, and then repeat the cycle by exploring and adding new features. We also researched external papers on sentence complexity to gather more feature ideas. Such a cycle helped us generate more useful features over time, such as readability scores and sentence embeddings. Our features were saved into .pkl files to increase efficiency and allow us to test more models. Features used include:

**Dale Chall:**

**Dale Chall Sum:** Number of Dale Chall words in the sentence.
**Dale Chall Percent:** Percentage of Dale Chall words to the total number of words in the sentence.

**SAT Words:**

**SAT Sum:** Number of SAT words in the sentence.
**SAT Percent:** Percentage of SAT words to the total number of words in the sentence.

**Age of Acquisition (AoA):**

**AoA_Freq:** Sentence average of word frequency scores. This score measures the frequency of a given word in common english.
**AoA_Mean_Age:** Sentence average age of acquisition for words in the sentence that matched.
**Max_AoA_Age:** Sentence maximum age of acquisition in the sentence.
**Min_AoA_Age:** Sentence minimum age of acquisition in the sentence.
**Sqrd_AoA_Mean_Age:** Sentence squared average age of acquisition of words in the sentence.

**Concreteness Ratings:**

**Conc.M:** Sentence average of word concreteness ratings.
**Percent_Known:** Sentence average of word percent know scores demonstrating the percent of 5th grade that had understanding of the word.

**Text_stats from Textacy Library:**

**Automated_readability_index:** Readability score from Textacy library. While all readability scores are useful, this one turned out to be the most useful.

**Coleman_liau_index:** Readability score from Textacy library.

**Flesch_kincaid_grade_level:** Asses the reading grade level.

**Flesh_reading_ease:** Assesses the ease of readability in a document.

**Gunning_fog_index:** Assesses the reading grade level.

**Lix:** Readability score from Textacy library.

**Perspicuity_index:** Readability score from Textacy library.

**Smog_index:** Readability score normalized on 30 sentence samples.

**Word Embeddings (FastText):**

**Average_Embed:** Average value of 50 dimensional sentence embedding via FastText. (Mikolov Et al.)

**Sentence_Embed:** 50 dimensional sentence embedding via FastText

**Word_Embed:** List of 50 dimensional word embedding per word in the sentence via FastText.

**Other:**

**Passage Sum:** Total number of words in the sentence.

**TTR:** Type-Token Ratio. (Number of Unique words / Number of Words) * 100. Values closer to one represent more complex and rich text.

**Pronoun Count:** Number of NLTK tagged pronouns in the sentence.

**Pronoun Percent:** Percentage of pronouns to total number of words in the sentence.

**LRB_RRB_matched:** Boolean value to check if wikipedia text tags ("-LRB-|-RRB-") are present. This boolean (and the next two) were added after failure analysis revealed numerous incorrect predictions involving sentences with these words / tokens.

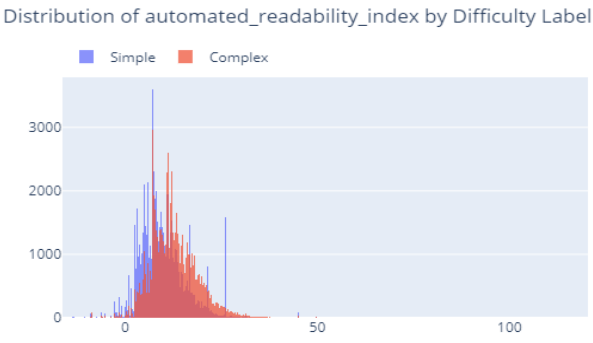
**Commune_matched:** Boolean value to check if the word “commune” is present.

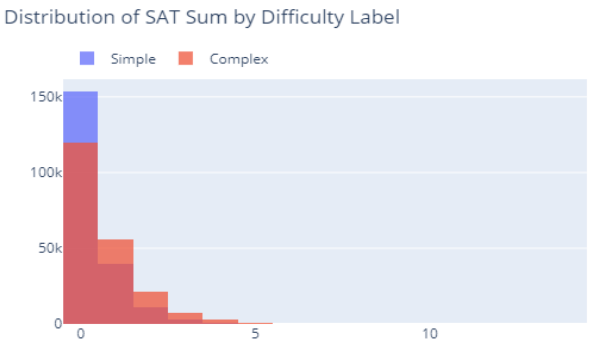
**Football_matched:** Boolean value to check if the word “football” is present.

**KMeans Label:** Label generated by our unsupervised KMeans model with 2 clusters.

**Methodology of Feature Selection:**

While developing our features, we explored them visually in multiple ways such as exploring them using other visualizations such as histograms to see if their distribution differed depending on the label class. The best features would ideally have less overlap in their distributions by label. We observed that the automated readability index and the AoA frequencies have quite stable but noticeable differences between the labels despite heavy overlap. On the other hand, features like SAT Sum had extreme overlap and were highly skewed towards lower values regardless of label.

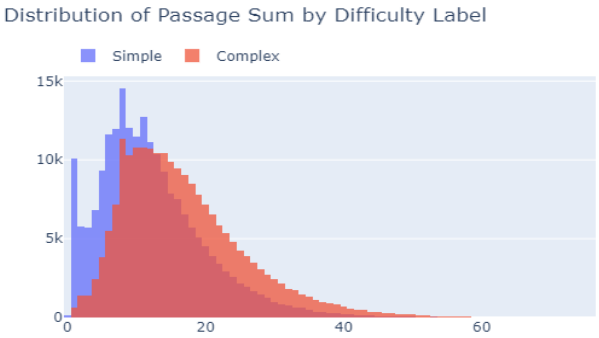


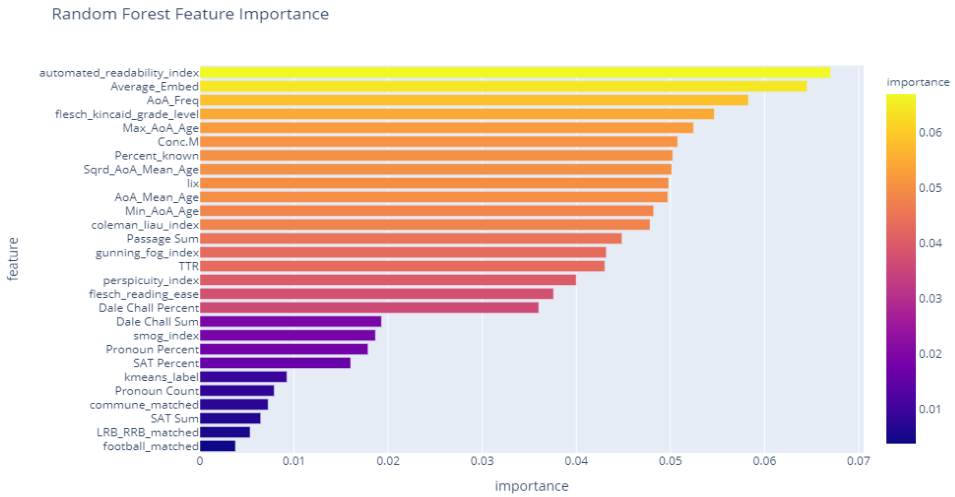




We were able to deduce which of the features were acting as a word complexity scorer. Take the Passage Sum feature where the lower values are more likely to be simple sentences and the higher values are more likely to be complex sentences, as demonstrated by the blue bars and orange bars. This seems to suggest that a change in the

complexity occurs when sentences have more than 16-18 words. This same trend occurs in the various AoA Age features and the Automated Readability Index.

Furthermore, we used the feature importance values generated by fitting our Random Forest models to check for the most useful features. We can see that certain features contribute a lot more to the classification process than others. In essence, this helped confirm our histogram analysis with the readability scores having relatively high feature importance and others such as SAT Sum having lower values. Feature importance analysis also helped us understand the task more intuitively. For example, we noticed higher importance for Max AoA age rather than the min or average. This makes sense in practice because a single pivotal but complex word in a sentence can be very difficult to understand and can often benefit from simplification.




Overall, our feature engineering and importance analysis revealed that there are multiple successful approaches to understanding complexity with the three main categories being:

1. **Readability Scores:** More complex formulas to quantify complexity based on research. These are more nuanced statistical methods that can give a more holistic view of the text difficulty. Full details of readability scores can be found in the Textacy documentation.
2. **Standard & Historical Methods**: This includes counting words or features in a more basic manner and/or matching them with existing resources of known complexity such as Dale Chall, SAT Words, or AoA.

3. **Embedding Vectors:** Our research highlighted the power of semi-supervised methods such as Sentence2Vec to generate a contextual vector in high dimensional space. Interestingly enough, even a single average value of the embedding was highly influential. While the full 50-dimensional vector did increase our accuracy by 1% in our Random Forest evaluation, the resource costs of doing so are high and most likely not worth it in practice. However, future research is warranted to further test the effect of higher dimensional embeddings on complexity classification.

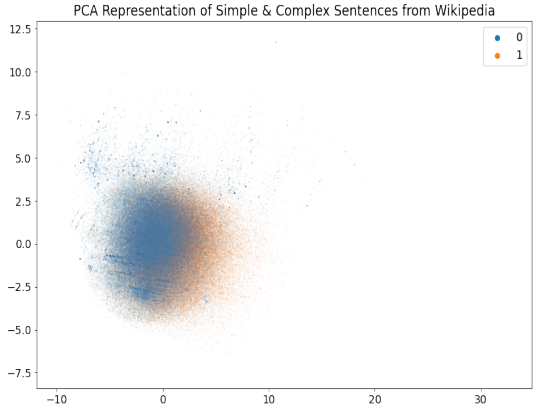
## Unsupervised Learning:

Our main goal for the unsupervised portion of our project was to aid in improving our classification accuracy of the downstream application which involves supervised learning models.

This could be a direct benefit such as producing effective cluster features that could be used as extra features, as well as indirect benefits such as interesting visualizations or data representations such as PCA and Topic Models that would reveal insights about the data that could inspire pipeline choices or new features to be created.

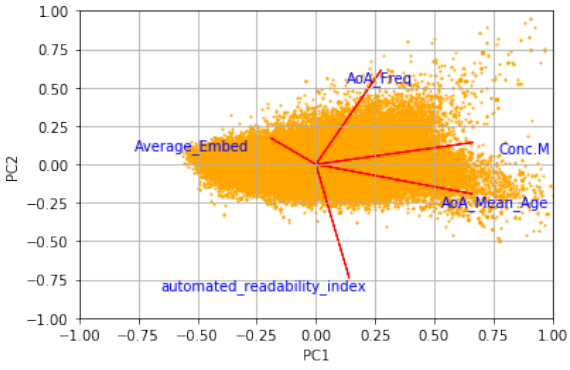
### Dimensionality Reduction & Data Visualization:

Visualizing our data using unsupervised methods was a key part of our project. We projected the data with a focus on our engineered features into different feature spaces using both PCA and UMAP, both of which are relatively efficient and able to handle large datasets.




We projected the data into PCA space first using the first two principal components to focus on the highest variance as well as make it easier to plot and visualize. PCA revealed an interesting structure of the data with a noticeable shift from left to right between our labels based on the training data. While there is heavy overlap in the labels, it gave us confidence that our features were producing noticeable differences between the labels.

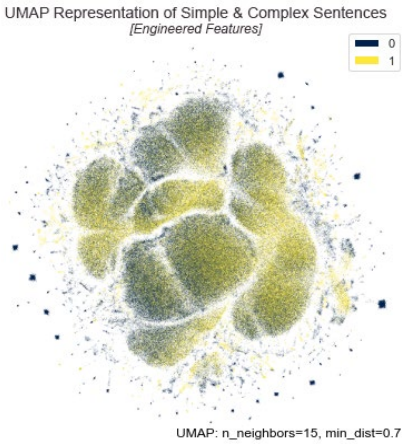
We also used a PCA representation to plot our feature vectors against the two major principal components. This helped us better understand our features in relation to each other and helped us check if any features were redundant.




PCA represented data also gave us performance benefits in terms of speed due to reduced dimensions, but the effect on downstream accuracy varied. PCA worked fine with KMeans, but our machine learning models lost accuracy under using PCA representations. For example, our Random Forest went from 75% accuracy to 68%

accuracy under PCA. This is expected due to loss of information, but still unfortunate.

For UMAP, we also observed an interesting phenomenon where it segregated the data using our features into multiple clusters. However, the clusters UMAP found did not correlate with either label. In fact, most clusters were split equally between both labels. This may suggest that UMAP is finding other common themes. However, it is likely that UMAP is finding local clustering elements and not necessarily interesting global structure. Furthermore, the UMAP representation is highly dependent on parameters such as neighbors and minimum distance. We looped through different parameter options to check for interesting trends, but the sensitivity can be an issue. It is better than T-SNE but still vulnerable. Overall, the UMAP representation was visually interesting but less practically useful.




## Clustering:

Our PCA results inspired us to attempt to cluster the data into 2 clusters which could potentially match our intended complexity labels. However, we also tested different numbers of clusters to see what matched our features best.

It should be noted that different metrics are evaluated in different ways. We tested a variety of metrics and observed that a cluster size of 2 was an optimal cluster size in general using KMeans. It was only beaten during our testing by clusters of 5 & 6 when considering the Davies-Bouldin Score. This may be a sign of the heavy overlap in clusters since Davies-Bouldin is the "average similarity measure of each cluster with its most similar cluster."

We also tested our clustering results by fitting a 2-cluster KMeans model on our features (without labels) and compared the predicted labels against the training labels. Our KMeans labels were 61% accurate compared to the training labels. Furthermore, the F1 score was 0.58 which indicates that our model picked up on actual variance or signal in the features considering the training dataset is balanced. This gave us confidence that our features were working and that the KMeans labels may be useful as a feature for our downstream supervised classification task.

| clusters | adjusted_rand_scores | completeness_scores | v_measure_scores | davies_bouldin_scores | calinski_harabasz_scores |
|---|---|---|---|---|---|
| 2 | 0.050836 | 0.038090 | 0.037862 | 1.933924 | 91968.000067 |
| 3 | 0.037042 | 0.025556 | 0.031317 | 2.059652 | 70530.292067 |
| 4 | 0.026783 | 0.025043 | 0.033205 | 2.071049 | 58386.026692 |
| 5 | 0.026164 | 0.024015 | 0.032544 | 1.819474 | 53302.231331 |
| 6 | 0.028761 | 0.021082 | 0.028900 | 1.706343 | 49985.610747 |

## Topic Modeling:

A major question we had going into this project was if the need to be simplified was correlated with certain topics. If labels were biased towards different topics, that could be a useful research area or opportunity to generate new features to help in our classification task.

We separated the data into two sets based on their training labels, and then performed topic modeling on those sets to observe the differences. Specifically, we used a TF-IDF text representation of the data that was fed through Non-Negative Matrix Factorization (NMF) to discover topic vectors. We chose NMF due to its efficiency and more coherent topics. We then took the top-5 topics of each set to check for the most reasonable topics. Focusing on a smaller but pivotal set of topics improves readability. understanding and should help reduce noise.

However, we observed nearly identical topics between both sets of simple and complex passages. Furthermore, the overall topics between the training and test data were similar. This discovery combined with our other observations such as the heavy overlap in PCA transformed data is an indicator that while the task may be feasible, it is definitely challenging and topics alone would not be a significant contributor to classifying sentences as needed.

**Top-5 Simple (0) Topic Vectors**

| # | Label | Word 1 | Word 2 | Word 3 | Word 4 | Word 5 |
|---|---|---|---|---|---|---|
| 1 | French département: Aisne | france | department | region | aisne | picardie |
| 2 | Football | rrb[2] | lrb | born | football | player |
| 3 | Websites | references | websites | notes | pages | heading |
| 4 | French région: Nord-Pas-de-Calais | pas | calais | nord | north | departme nt |
| 5 | Iowa | united | states | city | iowa | state |

**Top-5 Complex (1) Topic Vectors**

| # | Label | Word 1 | Word 2 | Word 3 | Word 4 | Word 5 |
|---|---|---|---|---|---|---|
| 1 | Communes in Aisne, France | france | commune | department | northern | aisne |
| 2 | Births | rrb | lrb | born | calendar | year |
| 3 | Communes in Nord-Pas-de-Calais, France | pas | calais | nord | region | commune |
| 4 | Iowa | united | states | county | city | iowa |
| 5 | Football | football | born | league | player | national |

*Note: lrb and rrb are text encodings for right and left parenthesis commonly found in Wikipedia passages. They were intentionally left in the data to be used as a matching feature.*

### Failure Analysis - Unsupervised:

The largest failure we noticed in terms of unsupervised learning was the lack of unique topic vectors. Furthermore, labeling topics remains a challenge. We relied on human intuition which we felt was reasonable due to the small number of topics and the more coherent topics produced by NMF.

Our KMeans label turned out to have some importance as can be seen by its own accuracy achievements, but it did not end up being very useful for our downstream classification task. Adding or removing the feature did not change the accuracy or F1 score from what we observed. Our cluster evaluations above as well as a silhouette score of 0.17 for 2 clusters show that our clusters are not clearly separated. Furthermore, the feature importance was low. This is most likely due to the fact that the majority of the information is held in other features that are overpowering it. Further research is needed to see if our KMeans label's importance is statistically significant.

Another issue we faced was related to resource limitations. Our training data was very large dimensionally, both in terms of rows and columns. When trying certain methods such as Kernel PCA or MDS, our localized (or cloud) environments would crash due to memory issues. Similar issues plagued certain clustering algorithms such as DBSCAN. We performed such tests on a smaller sample, but we felt they were not appropriate for a large-scale analysis or realistic deployment scenario. We would like to explore these methods in the future if possible.

## Supervised Learning:

Supervised learning was a major part of our project with our main goal of achieving a high classification accuracy directly linked to our supervised learning models.We tested multiple different models utilizing our features and performing both manual testing of models using sci-kit learn as well as an automated model evaluation using PyCaret. For reproducibility reasons, we set the same seed values throughout as much as possible to limit fluctuations due to randomization. This aided us in comparing models while tuning hyper-parameters.

## Dummy Classifier:

We started with a dummy classifier to set a "random" baseline. In our results, the accuracy was 50%. This is expected due to the balanced nature of the dataset.

## SVM Classifier:

SVM classifiers performed decently but accuracy remained around 57%-65% depending on parameters and sample data. The rbf kernel performed the best during our testing, but within a single percentage point of the others overall. SVM faced severe challenges in scaling to the large dataset required and is not practical due to time constraints.

## Logistic Regression Classifier:

Logistic regression had the advantage of being fast, being able to perform a 10-fold cross validation cycle on our training data of engineered features within 20 seconds. However, our accuracy remained at 66%-67% overall for the mean cross validation score.

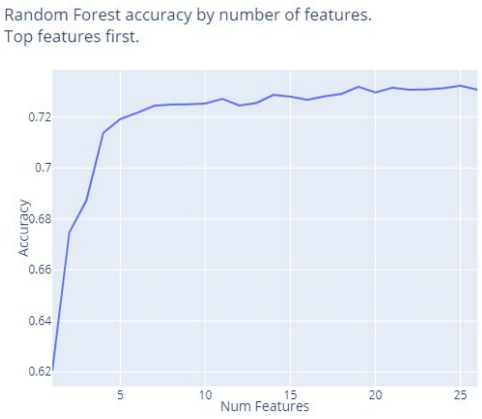
## Random Forest Classifier:

The Random Forest performed well, achieving the best score out of any traditional machine learning models. We achieved a 75% cross-validation score on the training data using our regular features, and 76% if we used a full 50-dimensional sentence vector instead of a single average embedding value as features. However, the computational cost of doing so was high both in terms of memory and time.

The only metric where the Random Forest did not perform the best was in recall but it remained competitive. It should be noted that the models performing better in recall were worse overall and suffered in precision. Thus, the Random Forest was able to balance the precision-recall tradeoff which can also be seen by its leading F1 score.

Random Forests are an ensemble model that are a staple of traditional machine learning. They are robust to different features, and often perform very well with little tuning required. Furthermore, they are fast, easy to interpret and explain due to the binary checkpoint nature of their structure acting like a complex flow-chart. Interpretability and overall performance are the major advantages of Random Forests which aids in both deployment, explaining results to stakeholders, and understanding the model to aid in feature engineering and parameter tuning. We used the feature importance values to aid our feature engineering efforts.

Furthermore, feature importance can be used to see the practical contribution of our features to our actual accuracy score. To do so, we retrained our Random Forest model with fixed parameters, but added in one extra feature per iteration starting from the most important features to the least and plotted the resulting accuracy of the model. We observed that our top features contributed heavily to the accuracy score, with performance hitting a slight plateau between 8-10 top features. This shows us that good feature engineering is vital to good performance.



## Neural Networks & Deep Learning:

We tested multiple different types of neural networks from basic MLPs suitable to tabular data on our engineered features, as well as more advanced models such as an LSTM (AWD-LSTM) and Transformer models (BERT) that could be run on the raw text.

We were specifically interested in the potential of the deep learning models, but they are computationally expensive.. In order to get around the resource limitations, we used a pre-trained BERT transformer model that we fine-tuned on our training data. A semi-supervised pre-trained BERT model has a lot of potential and uses all aspects of machine learning. We used libraries such as Fastai and HuggingFace as they provide useful wrapper functions and pre-trained models. We saved all checkpoints and final models for efficiency purposes.

The MLP performed the worst at around 68% but could probably achieve better with more parameter tuning or architecture changes. The language models could easily achieve 75% accuracy with relatively little training or tuning. We developed our own wrapper functions to speed up our process of iteration to download different pre-trained models, tokenize our data, train the model, and then evaluate it on a specified percentage of data as a validation set. We used a test-set percentage of 5% - 10% in our testing. In terms of evaluation metrics, we tracked training loss, validation loss, accuracy, and F1 score.

Hyper-parameter tuning was essential to good performance. We limited the maximum sequence length from 512 as default to 64. This helped improve memory and time efficiency, and restricted the information the model could learn which helped prevent overfitting which greatly boosted our accuracy score. We also modified learning rates and found that 3e-5 worked the best. We increased weight decay values from 0.01 to 0.1 to help prevent overfitting by regularizing the model more. Furthermore, BERT has other regularization steps in their architecture such as dropout.

To help improve accuracy, we concatenated our engineered numerical features to the end of the text passages joined by [SEP] tokens. We rounded off any numerical features to make them less unique since they would be used as tokens. This helped give extra data and information to the model.

All of these changes helped us boost our performance from around 73% to 77.2% accuracy / 0.79 F1 score during training. **Our best performing model was a deep learning model (BERT) fine-tuned on our training data & engineered features which achieved an accuracy score of 77.6% (0.77635) on the test dataset.**

We believe that with proper resources, larger models, and precise hyperparameter tuning, the deep learning models have the best chance at achieving the highest accuracy on this task. However, it should be noted that the Random Forest achieved effective performance with much less resources. The Random Forest did require a lot of manual feature engineering though, which is a challenge by itself. We believe both models have their potential.

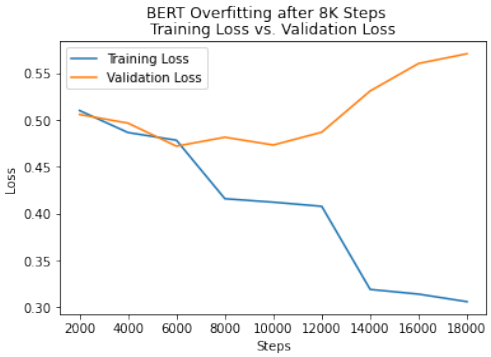
## Failure Analysis - Supervised:

Our Random Forest (and other traditional machine learning) models required heavy usage of feature engineering and other preprocessing steps, yet were outperformed by the BERT deep learning model pointing to the need for better features. An interesting aspect of the deep learning models we observed is that they are much less sensitive to the data. Normalization, imputation, and extra kmeans labels did not have a noticeable impact on our metrics. This might be explained by the structure of BERT where features, including nan values, are being treated as text tokens.

The deep learning models performed well with little pre-processing of data, but faced a downside in terms of extreme performance costs, including high memory, processor, and time requirements. We helped tackle these by

making sure to fine-tune models, using higher batch sizes, and off-loading training to a GPU when possible. However, despite these changes, models still took multiple hours to train and inference was also slow.

Unfortunately, we noticed a trend where the deep learning models would start to overfit the data. There is a rather consistent point where the model begins to overfit where the training loss drops from 0.40 to 0.32 but the validation loss then increases by a few points. Furthermore, the accuracy will begin to drop after that as well. This might be caused by our initialization steps, but it warrants further investigation. Better parameter tuning may solve this issue.




Another aspect of failure analysis is to observe what predictions went wrong. Luckily, our models can output their predictive score as well as a label. We filtered our resulting table to show rows where the predicted label and the training label (on a validation set) did not match (incorrect predictions). We focused on the "most confident" incorrect predictions and noticed some interesting trends. The majority of the top 50 incorrect predictions were sentences about football, about regions containing the word commune, or having the specific tags used in Wikipedia such as -LRB- or -RRB-. This brings us back to the beginning of our machine learning cycle where we used this information to adjust our feature engineering to create boolean values for them in hopes that it would improve classification accuracy. It also made us examine the topic vectors since the incorrect lines also correlated with the top topics, although that may also be due to frequency or prevalence of certain topics. However, further research is needed to see the statistical significance of our added features. It is possible that the difference is rather negligible and the major differences came from parameter tuning of the model itself.

## Conclusion & Next Steps:

We believe our research has given us more insight into what it takes to identify a sentence that needs to be simplified. We ran a continuous cycle of data exploration, feature engineering, unsupervised learning, supervised learning, deep learning, and continually iterated to improve our accuracy step by step. We achieved a 77.6% accuracy score on the test data which is a good result. However, we feel that with more time and resources, we could do better.

We believe that more effective features would be beneficial. For example, one of our features was the Type-Token Ratio, but our extended research shows that there are alternatives such as Guiraud's corrected TTR (GTTR) or Caroll's corrected TTR (CTTR) which show more importance in terms of complexity text classification. We would like to explore the Dynamic Support of Contextual Vocabulary Acquisition for Reading API that was recommended by Professor Kevyn Collins-Thompson to generate interesting readability scores.

Furthermore, we were very impressed by the relatively easy win of the deep learning model. It was able to exceed the traditional machine learning models on the text alone. We believe that with more resources and time to train bigger or more complex models such as GPT-3 variants that we could achieve even greater results. We also believe that further investigation and tuning of the deep learning models or preprocessing of the data is needed to help overcome the overfitting issue we encountered. In terms of improvements or refinements, we believe that further statistical analysis is needed to truly quantify how significant our results are, especially in regards to the fine-grained differences in feature importance or model selection. For example, our KMeans label showed importance greater than 0, but the minute values require further investigation.

There are potential risks or ethical concerns with running an automated classifier on such a nuanced topic. For example, if cultural biases are inherent in the training data, then our model may end up unfairly rating various cultural topics. This may cause a forced simplification at the expense of youth learning about other cultures. Our model may also overly classify text as needing simplification which could be detrimental to properly understanding important concepts. Despite these concerns though, we believe that with proper deployment and consideration of stakeholders, a model such as the one we trained would be beneficial. Overall, we hope our results can be used in the future to simplify sentences as needed in an ethical and effective manner.

## Statement of Work:

We worked together on most areas of the project with multiple calls and collaboration sessions to code together. While we did have different focuses, it was a highly collaborative effort as we refined our total pipeline and research together. Our focuses included:

- **James Mete:** Unsupervised & Supervised Learning, Deep Learning, Visualizations, Feature Importance.
- **Matt Dannheisser:** Feature Engineering, Pipeline Construction, Refactoring of Code.

## Works Cited:


Dale E; Chall J (1948). "A Formula for Predicting Readability". Educational Research Bulletin. 27: 11–20+28. (n.d.).

Bansal, S., & Aggarwal, C. (2021, September 5). *textstat 0.7.2*. Retrieved from PyPi: pypi.org/project/textstat/

Brysbaert, M. W. (2021, September 22). *Concreteness ratings for 40 thousand English lemmas*. Retrieved from Crr.ugent.be: http://crr.ugent.be/papers/Concreteness_ratings_Brysbaert_et_al_BRM.xlsx

Transfer learning in text. fastai. (n.d.). Retrieved September 14, 2021, from https://docs.fast.ai/tutorial.text.html.

Kotani , K., Yoshimi , T., & Isahara, H. (n.d.). A Machine Learning Approach to Measurement of Text Readability for EFL Learners Using Various Linguistic Features . 26 September 2021; David Publishing. P 771.

Kuperman, V., & Stadthagen-Gonzalez, H. (2021, September 22). *Age-of-acquisition (AoA) norms for over 50 thousand English words*. Retrieved from Crr.urgent.be: http://crr.ugent.be/archives/806

Kurdi, M. Z. (2020, July 29). Text complexity classification based on linguistic information: Application to intelligent tutoring of ESL. arXiv.org. Retrieved September 26, 2021, from https://arxiv.org/abs/2001.01863.

Rockikz, A. (2020, December 25). How to fine tune bert for text classification using Transformers in python. Python Code. Retrieved September 15, 2021, from https://www.thepythoncode.com/article/finetuning-bert-using-huggingface-transformers-python.

Rosewood Media. (2021, August 10). *SAT Vocabulary 1.0*. Retrieved from Most Comprehensive SAT Word List on the Web.: https://satvocabulary.us/INDEX.ASP?CATEGORY=6000LIST

Simple English Wikipedia. (2009).

T. Mikolov, E. Grave, P. Bojanowski, C. Puhrsch, A. Joulin. Advances in Pre-Training Distributed Word Representations

## Reproducibility:

We have provided three jupyter to reproduce our code corresponding to our major areas of feature engineering, unsupervised learning, and supervised learning. More files can be found in our shared google drive link here: https://drive.google.com/drive/folders/1zxM27RLrTujaU7I80T4IjV8hqGoIVS98?usp=sharing

Please follow the instructions and comments inside the notebooks. Major libraries used include Numpy, Pandas, Sci-kit Learn, FastText, UMAP, HuggingFace, PyTorch, Fastai, Pickle, Plotly, Seaborn, Matplotlib, PyCaret, NLTK, Spacy, and Textacy.

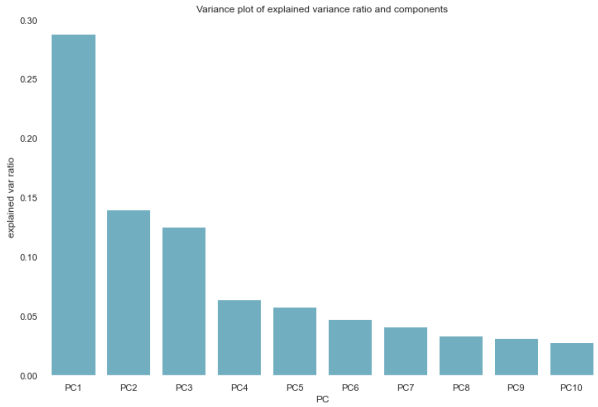
Kaggle files can be found at: https://www.kaggle.com/c/umich-siads-695-fall21-predicting-text-difficulty/overview
Please note that our models require the full datasets to be trained to stated performance levels in this report.

## Appendix:

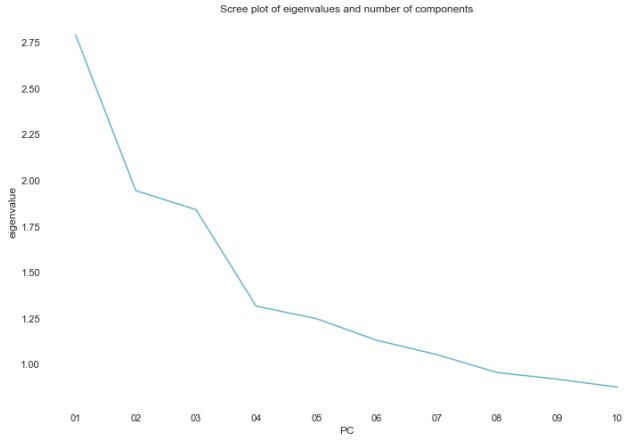
**Model performance chart (via PyCaret)**

| | Model | Accuracy | AUC | Recall | Prec. | F1 | Kappa | MCC | TT (Sec) |
|---|---|---|---|---|---|---|---|---|---|
| **rf** | Random Forest Classifier | 0.7177 | 0.8044 | 0.7309 | 0.7123 | 0.7215 | 0.4354 | 0.4356 | 23.5730 |
| **et** | Extra Trees Classifier | 0.7146 | 0.8005 | 0.7310 | 0.7080 | 0.7193 | 0.4293 | 0.4295 | 16.5150 |
| **lightgbm** | Light Gradient Boosting Machine | 0.6818 | 0.7512 | 0.7134 | 0.6712 | 0.6917 | 0.3637 | 0.3644 | 1.4070 |
| **gbc** | Gradient Boosting Classifier | 0.6700 | 0.7357 | 0.6968 | 0.6615 | 0.6787 | 0.3400 | 0.3405 | 29.6410 |
| **lr** | Logistic Regression | 0.6648 | 0.7166 | 0.6676 | 0.6640 | 0.6658 | 0.3295 | 0.3295 | 10.0050 |
| **ridge** | Ridge Classifier | 0.6644 | 0.0000 | 0.6577 | 0.6668 | 0.6622 | 0.3288 | 0.3289 | 0.1270 |
| **lda** | Linear Discriminant Analysis | 0.6644 | 0.7165 | 0.6575 | 0.6670 | 0.6622 | 0.3289 | 0.3289 | 0.3900 |
| **ada** | Ada Boost Classifier | 0.6621 | 0.7198 | 0.6855 | 0.6551 | 0.6699 | 0.3243 | 0.3246 | 6.2210 |
| **dt** | Decision Tree Classifier | 0.6587 | 0.6591 | 0.6586 | 0.6589 | 0.6588 | 0.3174 | 0.3174 | 2.0180 |
| **knn** | K Neighbors Classifier | 0.6465 | 0.7011 | 0.6692 | 0.6403 | 0.6544 | 0.2930 | 0.2933 | 2.9030 |
| **nb** | Naive Bayes | 0.6347 | 0.6929 | 0.7324 | 0.6129 | 0.6673 | 0.2694 | 0.2747 | 0.1120 |
| **svm** | SVM - Linear Kernel | 0.5758 | 0.0000 | 0.5603 | 0.6336 | 0.5095 | 0.1515 | 0.1906 | 4.8700 |
| **qda** | Quadratic Discriminant Analysis | 0.5704 | 0.6752 | 0.9177 | 0.5417 | 0.6813 | 0.1406 | 0.1955 | 0.2920 |

**PCA Variance Plot of Explained Variance Ratio and Components**
**Based on code from SIADS 543**




**PCA Scree Plot**
**Based on code from SIADS 543**




**PCA Magnitude Chart of Features and Top 2 Components**
**Based on code from SIADS 543**