Prediction of text difficulty

Notebook Folder Link , Data Folder Link1, Data Folder Link2, App Folder Link

Team 10, Sharadwata Ganguli (sganguli@umich.edu), Panini Mohan Mokrala (mokrala@umich.edu)

Abstract

Background: For editors of **simple Wikipedia** articles, it would be quite valuable to know which parts of an article's text to simplify before spending actual effort in simplifying the text. A **text difficulty prediction algorithm** would be effective for this task

Approach / Methods: We used a **multi-step approach** starting from Feature Engineering to Unsupervised & Supervised Learning, to create the **text difficulty prediction algorithm**. We used the same Wikipedia text labeled training corpus for both unsupervised and supervised components

Findings: Throughout our exercise, we found some interesting features in the Wikipedia text corpus. While the **length of text** that requires simplification is longer, this attribute isn't significant enough to improve the accuracy by a lot. Our unsupervised learning algorithms helped us get the '**denser**' feature space, that reveals a vastly similar distribution across training labels highlighting the need for us to understand how the labelling had been done. A simple illustration through UMAP illustrated that there is no clear distinct clustering structure. This overlapping nature of the sentence features across labels means that the classification algorithms will find it hard to codify the inherent patterns of sentences that needs simplification and vice-versa. With the above-mentioned limitations, we are able to achieve a **reasonable accuracy of 70% in validation dataset and 72% accuracy in test dataset**. Some interesting observations supporting this argument are present in Failure analysis.

Conclusion: The **Machine Learning algorithms** that we have used to identify clusters or predict difficulty **apply the same rule to each text consistently**. However, the **ground truth labels** of text difficulty in the training corpus were presumably crowd sourced - this would mean that there could be **inconsistency** in the training labelling approach due to various **biases of the labeling person(s)**. Thus, the ground truth labels could be **subjective** and using it to train models could end up **replicating the biases** of the human labelers. A potential next step could be to identify **approaches to make the ground truth labels more consistent** and thus reduce the subjectivity in the labeling

1. Introduction

For **editors of simple Wikipedia** articles, it would be **valuable to be provided suggestions** as to which parts of a text needs to be simplified. Given this context, we have developed a **text difficulty prediction model** (the **supervised part**) whose objective will be to accurately **predict 'need for simplicity'** of any input sentence. We explored **setting up an app** that can take real world examples and highlight if the sentence needs simplification. Simultaneously, an **unsupervised part** was used to **augment the features** for the supervised part as well as **identify actual clusters** in the text. We have used **google colab & google drive** platforms for coding, collaboration & data storage.

We have used a set of Notebooks (Folder link) -these were run in the following sequence to follow the below Process flow:

1. Step1_Feature_Engineering_Part1 (Notebook1)
2. Step1_Feature_Engineering_Part2 (Notebook2)
3. Step2_Unsupervised_Learning (Notebook3)
4. Step3_Supervised Learning (Notebook4)
5. Step4_FastApi App/Python application (Notebook5)

We followed a 4-step approach for this project– as shown below. We will discuss the steps in detail.

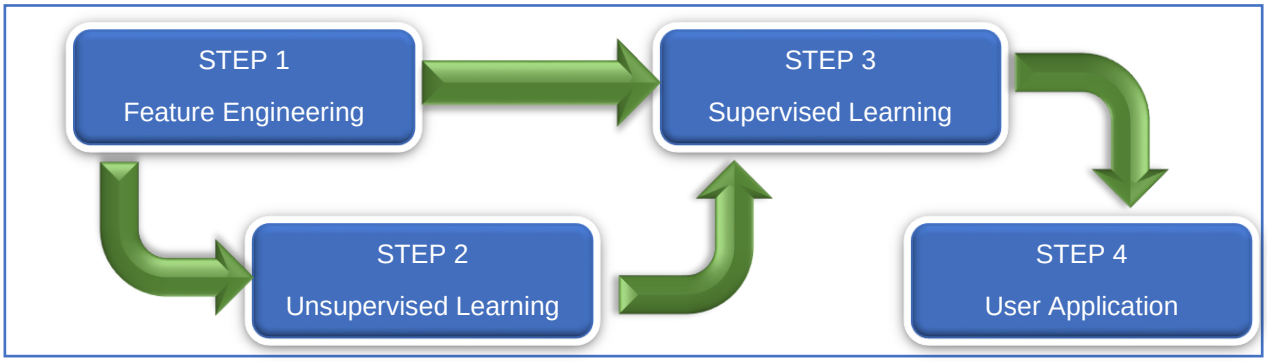


Figure 1. Project process flow

2. Feature Engineering

2.1 Motivation:
We kept the Feature Engineering task of the Project as a separate initial component as we believe it's useful to create and evaluate a set of features before implementing them in Machine Learning later. We used 2 notebooks for this step (Notebook1 link, Notebook2 link).

2.2 Data sources:
The data source was https://www.kaggle.com/c/umich-siads-695-fall21-predicting-text-difficulty/data , from where we downloaded a zipped folder containing the following files:

File Name	File Format	File Size	Key variables	Number of records / Notes
WikiLarge_Train.csv	Csv	50,084 KB	original_text, label	416,768 (347,356 unique text)
WikiLarge_Test.csv	Csv	14,923 KB	Id, original_text,	119,092 (110,166 unique text)
AoA_51715_words.csv	Csv	3,443 KB	Word, AoA_Kup_lem	51,715 unique words
Concreteness_ratings_ Brysbaert_et_al_BRM.txt	txt	1,569 KB	Percent_known	40,000 lemma words
dale_chall.txt	Txt	19 KB	NA	List of 3000 words
additional_resource_ file_readme.txt	Txt	3 KB	NA	NA
sampleSubmission.csv	Csv	1,055 KB	id, label	Submission Format for predicted Test labels

Table 1. Input file details

2.3 Approach / Methods:

We used *green font italics* to denote the features in this Report. Key steps are as follows:

1. *Initial Feature Generation* (Notebook1 link, Notebook2 link)

We generated 5 sentence level features of character count, token count, characters per token, character count of largest token and Part of Speech Tag counts in Notebook2. These features cover how long a sentence is, does it contain big word or multiple Parts of Speech etc.

A few features were generated in a separate notebook (Notebook1 link). In this notebook, we used external data sources that are provided through Kaggle to us. Quick summary on the features –

1. Dale Chall Words: The hypothesis we had was‘ sentences that needs simplification will have less words overlapping with Dale Chall words’. We found that on an average, ~56% of the words in a given sentence are easy to interpret irrespective of the label. This is not unsurprising as English is an ever-evolving language, more rapidly in today’s communication and digital age. Hence, the Dale Chall list developed in the 90s may not be relevant for this more recent corpus
2. Average Age of Acquisition: The hypothesis we had here is ‘sentences that needs simplification will have higher learning age compared to other sentences’. We found out a slightly **counter intuitive insight** – sentences that needed simplification had similar ‘Age of Acquisition’ hence we did not proceed with it. We further hypothesized that easy sentences are easy to interpret and hence classifiable as ‘Needed simplification.
3. Concreteness ratings: We hypothesized that ‘sentences that needs simplification may have limited Concreteness and in general greater ambiguity of definition’. While the result was partly true, due to statistical significance issues, we didn’t go further with this feature set.
4. We generated Named Entity Recognition features using spacy and found that there is a differential distribution of NERs across the 2 labels. Hence, NER features seem to be a good candidate for predicting the text difficulty labels. However, a high rate of missing values resulted in not using these features. The missing values problem is occurring as we used a pre-trained model from Spacy

Kindly refer to the appendix section, in case you are interested in looking through these inferences

1. *Exploratory data analysis to guide the feature generation step* (Notebook2 link)
   We found the training labels to be balanced at 50% each. This would indicate that a **simple prediction accuracy would be good enough** to evaluate various models -no need for F1 score. We identified that the **most common words are stop words** – removing them would help reduce unnecessary dimensions from bag of word count or tf_idf weight features.

   Now, when we evaluated *token attribute count* or *POS tag count* (generated in the initial feature generation stage), we found that while they do have a difference across labels -but this difference is small. This is evident in below plot for word count (*token attribute count* feature) values across the 2 labels (other features show a similar pattern).

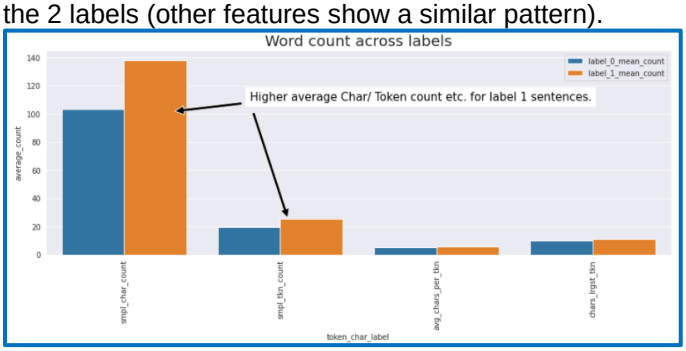


Figure 2. Word count across labels (link)

The reason could be that sentences being considered difficult (label 1) have **slightly** bigger, presumably complex longer words or sentences with a **varied richer grammatical structure**

consisting of more Parts of Speech. This difference, though not large indicates that these features can be considered for difficulty prediction.

2. *Final Feature Generation using bag of word Counts, tf_idf weights and word 2vec* (Notebook2 link)
   In the final Feature Generation stage, we derive the dev and test versions of the initial features. We also developed *bag of word counts* and *tf_idf weights* training features (sparse features) -3 options each including a bigram option as well a 100-dimensional *word2vec* dense feature. We subsequently generated the dev and test versions of these new features too.

2.4 Evaluation & Conclusion (Notebook2 link)**:**
Along with a uniform random classifier as a baseline, we fitted a default **LR Classifier** (basic **supervised learning**) and identified the following features with the highest training and dev accuracy as well as lower dimensions (we also plotted the dimensions of the feature as a line using a log scale as the sparse features of *bigram bag of word counts* and *bigram tf_idf weights* have 10^5 dimensions):

1. Tfidf_vector bigram (*bigram tf_idf weights)*, sparse feature
2. Count_vector bigram (*bigram bag of word counts)*, sparse feature
3. word & pos tag count (*token attribute count + POS tag count*), dense feature
4. pos tag count (*POS tag count*), dense feature

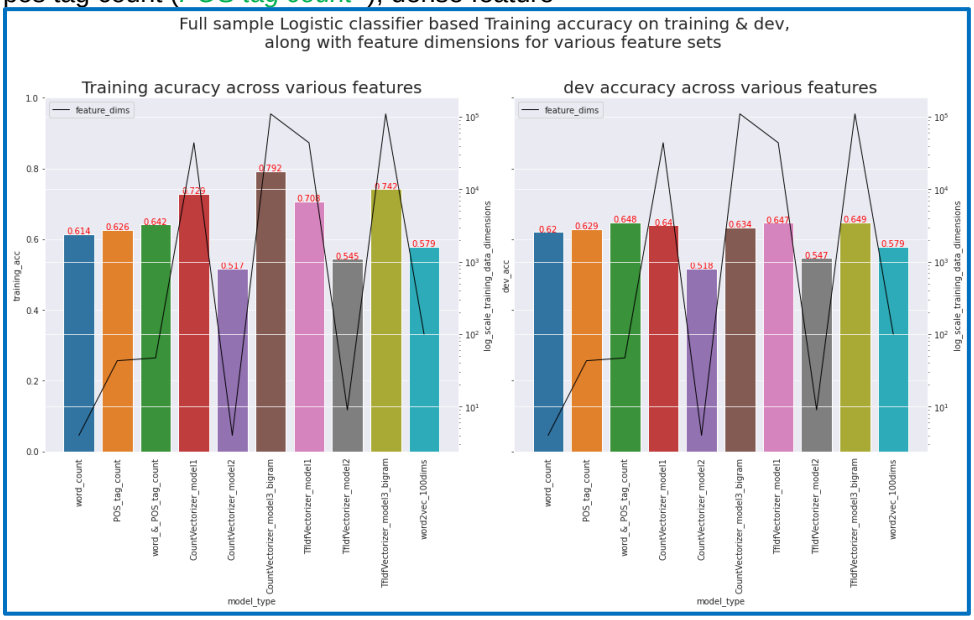


Figure 3. Training and dev accuracy evaluation for generated features *(link)*

While the dev accuracy on Logistic Classifier is mostly ~60% accurate for these features, **we hope to achieve higher dev accuracy when using these features with more complex models**.

We also explored combining a dense feature (*token attribute count + POS tag count*) to the sparse features (*bigram tf_idf weights*, *bigram bag of word counts*) and dense vector (*word2vec*) respectively. When we run the more complex Logistic Classifier, we find a consistent better performance for the *combined bigram bag of word*, *combined bigram tf_idf weights*, *combined word2vec* and *token attribute count + POS tag count* -on training as well as dev accuracy. These 4 features and rest generated features were saved in a pickle file, avoiding recreation of features.

2.5 Caveat
A point worth noting is that the features generated at this step were evaluated on simple models NB and simple Logistic classifier-at a later stage of the pipeline, when using more complex models **we may end**

up tweaking these features or using some other features based on the insight available at that later stage.

3. Unsupervised Learning (Part B)

3.1 Motivation:

We envisaged Unsupervised Learning as a step in between the steps of Feature Engineering and Supervised Learning to achieve the following outcomes:

1. Reduce dimensions of the generated features, assuming reducing dimensions of best performing features will enable quicker Machine Learning without loss of accuracy.
2. Clustering the sentences to understand the **actual structure of the sentences in feature space** and understand if the label prediction task could have a high accuracy for specific feature.

3.2 Data sources:

The data source for the Unsupervised step is the **pickle file** containing the features generated in the previous step of Feature Engineering (Step2). We picked the dense features with the best accuracy (*combined word2vec* and *token attribute count + POS tag count*) and the sparse features with the best accuracy (*combined bigram bag of word*, *combined bigram tf_idf weights*). We also kept the *word2vec* dense feature as a reference.

3.3 Unsupervised Learning Methods:

We used 1 notebook for this step-Notebook3.

Key Steps included:

1. *Initial Visualization of Structure*
   When we executed basic 2-dimensional PCA on the normalized dense features, we found that the data points were concentrated in a central mass instead of being in roughly 2 separate clusters (as we had 2 training labels), as shown below:

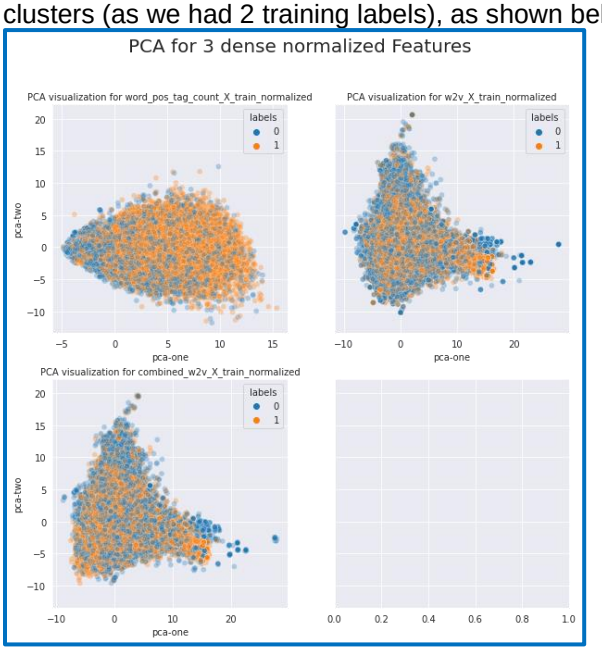


Figure 4. Basic PCA for 3 dense features *(link)*

A reason for this could be that the 1st 2 Principal components only explained ~20% of the variance, or the sentences are very close on these features! We then identified the exact cut-off required to explain 80% of the variance which would be useful in actual dimension reduction.

Even when using MDS or tSNE (**using lower sample due to computational limitations**) -we got a similar central overlapped mass as the PCA. We also observed this central overlapped mass using a full sample default UMAP.

When we ran Truncated SVD on the *combined bigram bag of word*, *combined bigram tf_idf weights* features, we found that the 1st SVD component was taking up ~99% of the variance - making any dimension reduction meaningless. This could be due to the fact that these were a combination of dense and sparse arrays. When we ran the Truncated SVD on the original sparse bigram features (*bigram bag of word*, *bigram tf_idf weights*) – we got a reasonable spread of variance across the SVD components.

2. *Dimensionality Reduction*
In this step we ran a basic PCA on the normalized versions of the 3 dense arrays, using the number of components identified earlier to explain 80% of the variance in the data. A key point is that while the highest accuracy combined dense + sparse features cannot be dimensionally reduced -we still have the better performing dense features whose dimension we can reduce. **Thus one of our objectives of Unsupervised Learning was satisfied!**

4. *Dimensionality Reduction Evaluation*
We found that the reduced versions of dense features *reduced combined word2vec* and *reduced token attribute count + POS tag count* were similar to the Training Accuracy of their original versions on NB classifier . Hence these 2 reduced features were taken forward to clustering stage.

7. *Clustering*
We implemented 2 different clustering solutions -a simple KMeans and a GMM clustering which can provide us soft clustering probabilities -these can be used as features. However we found that the GMM cluster probabilities are not that great on Training accuracy , possibly as the single digit GMM cluster labels may not have sufficient information to allow for robust clustering. **UMAP** visualization was conducted using combinations of hyperparameters of **n_neighbors** and **min_dist** to identify a **global topography with local nuance**. We then evaluated the KMeans and GMM labels by overlapping them on this UMAP structure and comparing the cluster labels vs the training labels -as shown below for the smaller dimension *reduced token attribute count + POS tag count* :

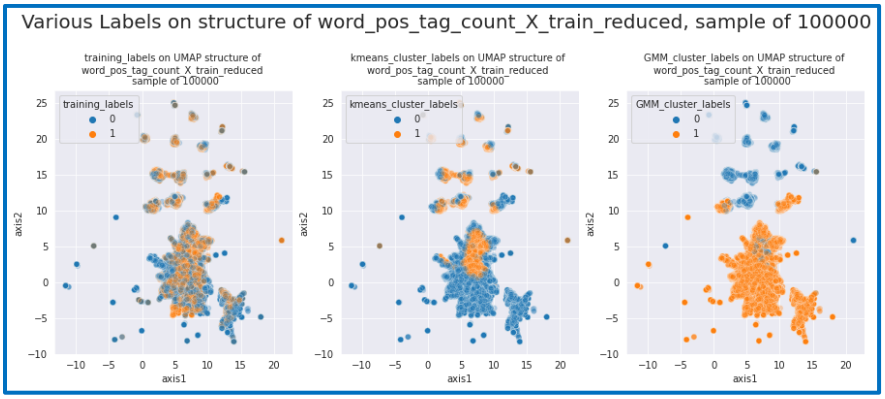


Figure 5. Training and Cluster labels on structure of *reduced token attribute count + POS tag count (link)*

From above, we can see that the KMeans 2 cluster labels are more similar to the binary Training labels. The clusters for *reduced combined word2vec* feature have less overlap with training labels though. **Does this mean that the 2 cluster K means solution is the better cluster ?**

8. *Clustering Evaluation*
   We now evaluated the 4 clusters -2 cluster Kmeans , 2 cluster GMM for *reduced token attribute count + POS tag count* and 5 cluster KMeans and 5 cluster GMM for *reduced combined word2vec* on 3 non ground truth cluster metrics of Silhouette score, Davies Bouldin score and Calinski-Harabasz score. Additionally as we have the training labels, we also used the ground truth based cluster quality metrics of Cluster completeness, Cluster homogeneity and Adjusted Rand Index. We found that the 2 cluster KMeans cluster for *reduced token attribute count + POS tag count* is the best cluster solution amongst the 4 clusters we evaluated. It does better on 4 out of 6 evaluation metrics (except silhouette score and ARI). The *reduced token attribute count + POS tag count* feature has a natural cluster of 2 which aligns with the training labels -also it has reasonable structural overlap with the training labels. Would this be the feature with the best performance in Supervised Learning ?

3.4 Unsupervised Evaluation / Insights:

Central overlapped mass
The **overlapped central mass** observed in PCA, MDS, tSNE and UMAP indicate that the **sentences having different labels lie very close to each other in the feature space** (for the 2 dense features evaluated)– we had observed a similar pattern for some of these dense features in the Feature Engineering stage too. This indicates that differently labelled sentences may not be very different -this could impact prediction accuracy!

Optimum clustering
The KMeans 2 cluster solution for the *reduced token attribute count + POS tag count* feature had some overlap with the binary Training labels as well a better clustering quality. As its clustering was better aligned with the training labels, this could be a better predicting feature.

3.5 Unsupervised Caveat:

A key challenge that we faced in this step is **computational resource limitations** which often meant we used a lower (often around 1/3 of training sample =100,000 instances) to visualize using UMAP or run Cluster Evaluation. **We assume that the distribution for this 1/3 of training data is same as full sample**. If this is not the case -then our cluster insights would not hold! Also, the accuracy of reduced dimension features was evaluated on simple NB classifier and the accuracy may not hold on complex models.

4. Supervised Learning (Part A)

4.1 Motivation: With richer understanding of the data through Feature engineering and Unsupervised learning, we tried to triage these learnings and create a classification engine that differentiates if a particular sentence needs simplification or not. We approached this section as follows –

a. Build a baseline model and check its performance with Dummy classifier
b. Utilizing the Feature engineering that we performed in the earlier sections, compare the baseline classifier with different feature spaces
c. After arriving at the best model, try building a Word2Vec based classifier and see if it can achieve higher accuracy

Note to the reader: We didn't utilize the reduced vectors obtained from various unsupervised learning exercises because these vector representations had lower accuracy on complex models, owing to the large proportion of unexplained variance in the data when using complex classifiers.

4.2 Data sources: We utilized the Wikipedia text corpus (divided as Training & Testing) to build a base classifier. Through our Feature engineering steps, we already have access to key features like *token attribute count + POS tag count* , *bigram tf_idf weights)* and *bigram bag of word counts*. To build the Word2Vec based model, we downloaded pre-trained Genism model - fasttext-wiki-news-subwords-300.

4.3 Approach / Methods: As highlighted in the 4.1 Section, we divided the entire supervised learning into 3 stage process. Across this section, we will elaborate the methodology and the decision points that made us follow the structure –

Stage – 1: Feasibility Checks

a. First, we built an As-Is Logistic Regression (LR) model plainly with the given text corpus (No corrections) and compared it with the Dummy classifiers. We observe that, the plain text classifier has an accuracy of 64% - A lift of 15% vs the Dummy classifier
b. As-Is model is doing better than a Dummy classifier, we can conclude that the Supervised learning for this problem space is worth investigating. We further investigated 3 scenarios:
    a. Removed stop words and prepared a Tf_idf vectorizer
    b. Removed stop words, lemmatized the tokens and prepared a Tf_idf vectorizer
    c. Lemmatized the tokens and prepared a Tf_idf vectorizer

   The rationale here is to see if stop words and lemmatizations helps in training accuracy. The results that we have are summarized in Figure 6. The lemmatized tokens improve the accuracy of classification tremendously – implying the **sentences that needed simplification mostly are to do with how a same word is represented in different lemmas, this was possibly missed by the other features like text length, bag of words and tf_idf weights.**

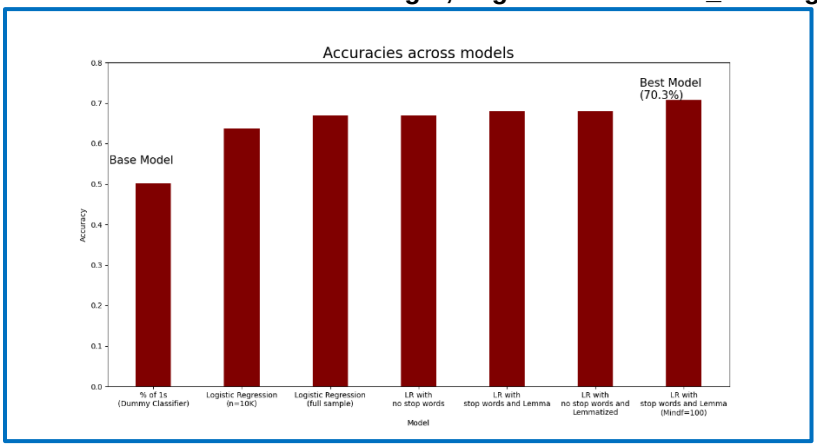


Figure 6: Baseline models compared with various versions – Best model is LR model with stop words and Lemmatization *(link)*

c. We finalized the Lemmatization based feature as it had a test accuracy of 72.1%.

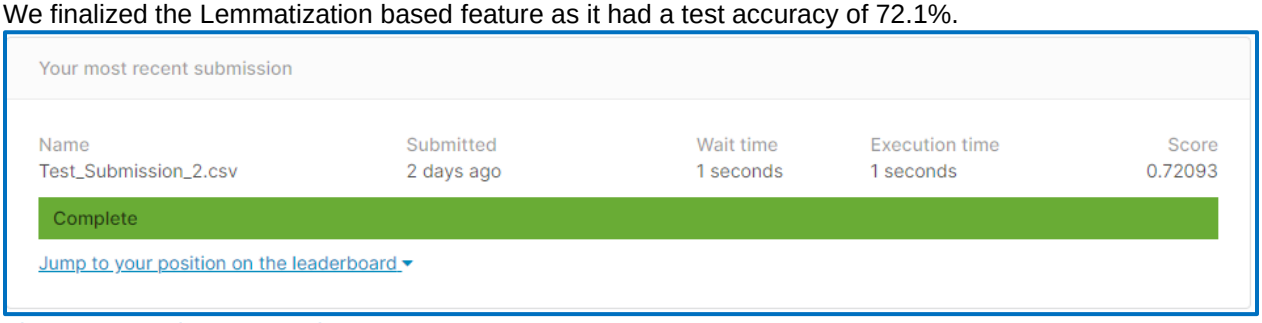


Figure 7: Kaggle Test result

Stage – 2: Compare with other feature spaces

a. The strategy here is, take every feature space, train them on a variety of models (Ridge Classifier, Perceptron, Passive aggressive classifier, Logistic regression, Naive bayes, Linear Support Vector Classifier and Stochastic gradient descent). Evaluate these models on the basis of the validation accuracy. Select the model with lower training time if accuracy similar for multiple models.
b. The table here summarizes the result for various feature spaces that we selected, top model and the resulting accuracy. The resulting validation accuracies of various models can also be found in the notebook link –

Feature space	Best fit model	Accuracy
Lemmatized text	Logistic regression	70.3%
Count vectorizer (*bigram bag of word counts)*	Logistic regression	68%
TF_IDF tags (*bigram tf_idf weights)*	Logistic regression	65%
Word + PoS tags (*token attribute count + POS tag count)*	Linear SVC	65%

Table 2: summary of best fit model and accuracy

Stage – 3: Word2Vec from pre-trained corpus: We utilized the Gensim pre-trained word2vec vectors (**fasttext-wiki-news-subwords-300**). For every token in the document, we extracted the vector notation and averaged out the vectors to represent each document. The method resulted in a training accuracy of only 64%. Hence, we went ahead with the **Lemmatized text LR with stop words and Lemma (mindf = 100)** model as it had a higher validation accuracy of 70.3%.

4.4 Failure Analysis: We looked into the examples where the model is not classifying correctly. We divided the data into 4 cohorts – 1. **Correct ones** - Correct classification as 1; 2. **Correct Zeroes** - Correct classification as 0; 3. **Wrong Ones** - Incorrect classification as 1; and 4. **Wrong Zeroes** - Incorrect classification as 0. We found the following problems with the current model framework:

- The Classifier is assuming that the length of tokens / characters for 'Sentences that needs simplification (label 1)' is higher. However, there are few instances where the converse is true - this possibly explain the rather moderate prediction accuracy, due to this **lack of consistency**.

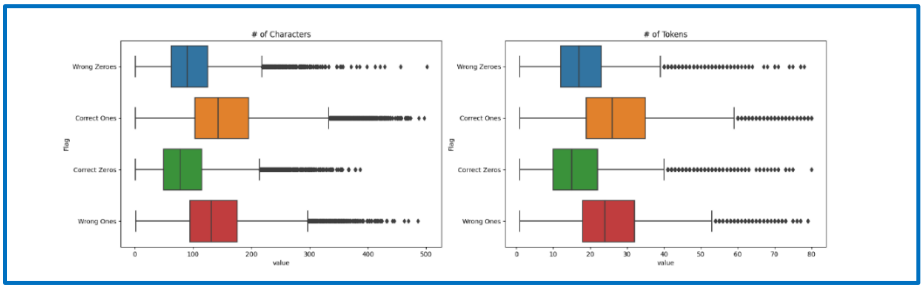


Figure 8 : Characters and tokens distribution across 4 groups of documents *(link)*

While these are good findings, the improvements that needs to be done can be a) treat the sentence as sequence instead of a vector b) check for coreferences in the document and feed into the classifier to ensure that the sequence is captured.

5. User Application

A fast api based application was created for a tech savvy user to utilize the model and see how the model flags for the sentence simplification. The app is not very user friendly and requires certain permissions to host in a public environment.

The instruction to run the application are present in the *App folder*. A caution to the App user- this may not work as per expectations due to the specific environment in which the codes are run.

6. Discussion and next steps

After a comprehensive feature engineering exercise followed by unsupervised learning, the supervised model accuracy of 70% on Validation and 72% on Test Dataset depicts the following insights –

1. **Feature space creation**: A simple vectorizer choice of TF_IDF vector on lemmatized words is yielding better results compared to the complex vectorization choices: There are high chances that complex vectorization choices and not restricting the document frequency is underfitting the model.

2. **Unsupervised learning choices**: The corpus that is flagged as needing simplification and not needing simplification have very similar syntactic structure, **a surprise**. As a result, the unsupervised algorithms couldn't capture the differentiation among the labels. Having said that, we would like to try advanced algorithm like LDA / Semi-supervised learning models to see if they can explain the inherent structure of sentences. This also highlights that there may be some bias in labelling a text corpus as at the end of the day, they are human labeled.

3. **Supervised learning choices**: In spite of trying various algorithms, the model is working well with a simple logistic regression. While we expected the complex models like Random forest, XGBoost to improve the accuracy, due to computational resource limitations, we didn't proceed that route - we **had to trade off model complexity for quicker evaluation**. Given a platform with higher resource limits, we would have explored the unsupervised structure visualization and clustering using full sample and would also have also implemented more demanding supervised learning algorithms like Random forest, XGBoost, Neural Networks.

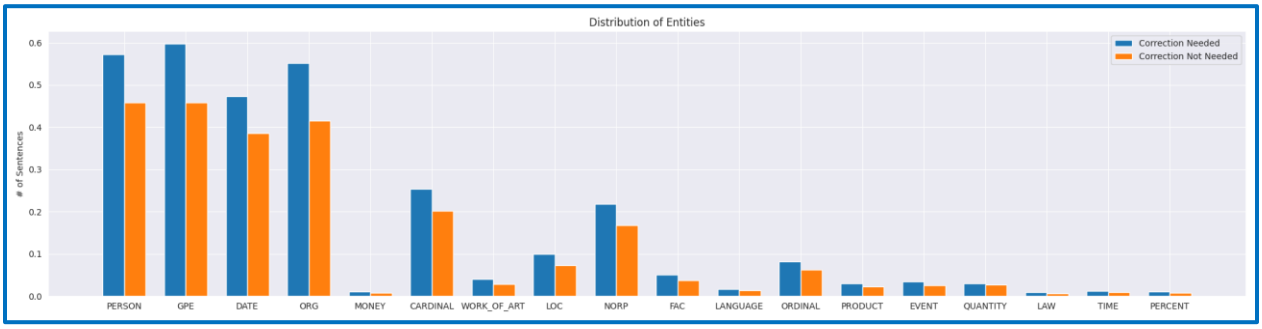
4. **Ethical Issues** : Wikipedia text may carry biases, which could increase the **historical disadvantage** of **protected groups** like race, gender etc. For example, the text could associate specific groups with **negative stereotypes**. Hence, we had planned to include a **bias identification step** in the **supervised learning** section, to identify and flag these potential negative stereotypes about protected classes in the text provided for classification. Due to limited time we could not implement this step though.

8. Statement of work

Sharadwata Ganguli
Sharad focused on drafting and finalizing the Proposal and the final Project Report. He led the Feature Engineering and Unsupervised Steps.

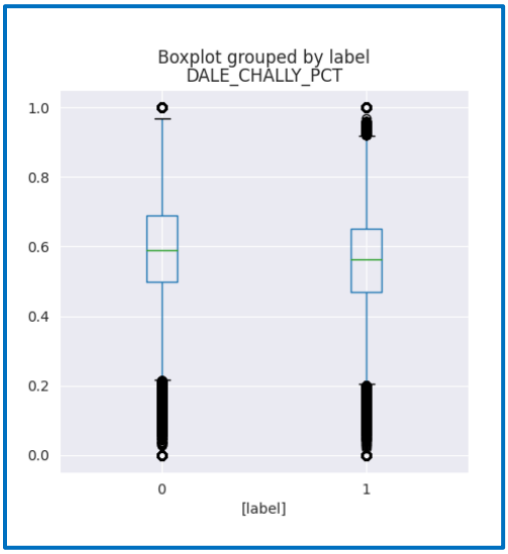
Panini Mohan Mokrala
Panini reviewed the Proposal and the Final Project. He also contributed to Feature Engineering, led the Supervised Learning component as well as developed the Web App.

9. Appendix



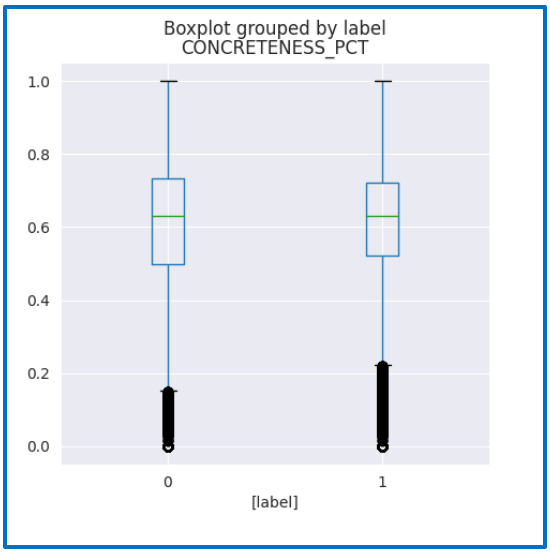
Appendix Figure 1. NER Comparison (*link)*

Label 1 text corpus has higher proportion of Named entities compared to other label suggesting that more references were made in Label 1 – This is in line with our insights on the token length



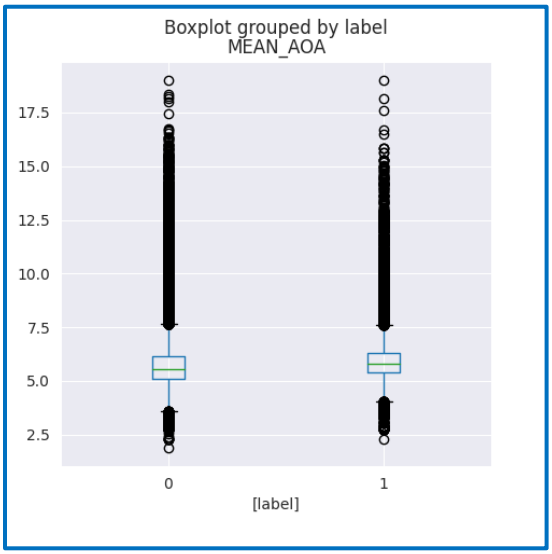
Appendix Figure 2. Dale Chall text (*link)*

Label 1 and Label 0 have a similar proportion of Dale Chall words.



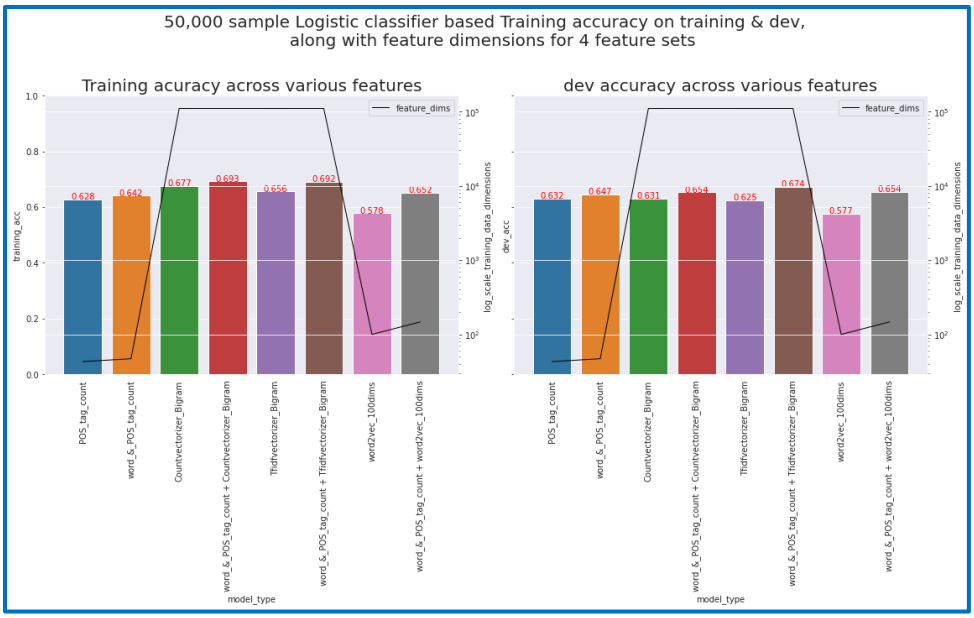
Appendix Figure 3. Concreteness words (percentage) (*link)*

Label 1 and Label 0 have a similar proportion of Concreteness percentage



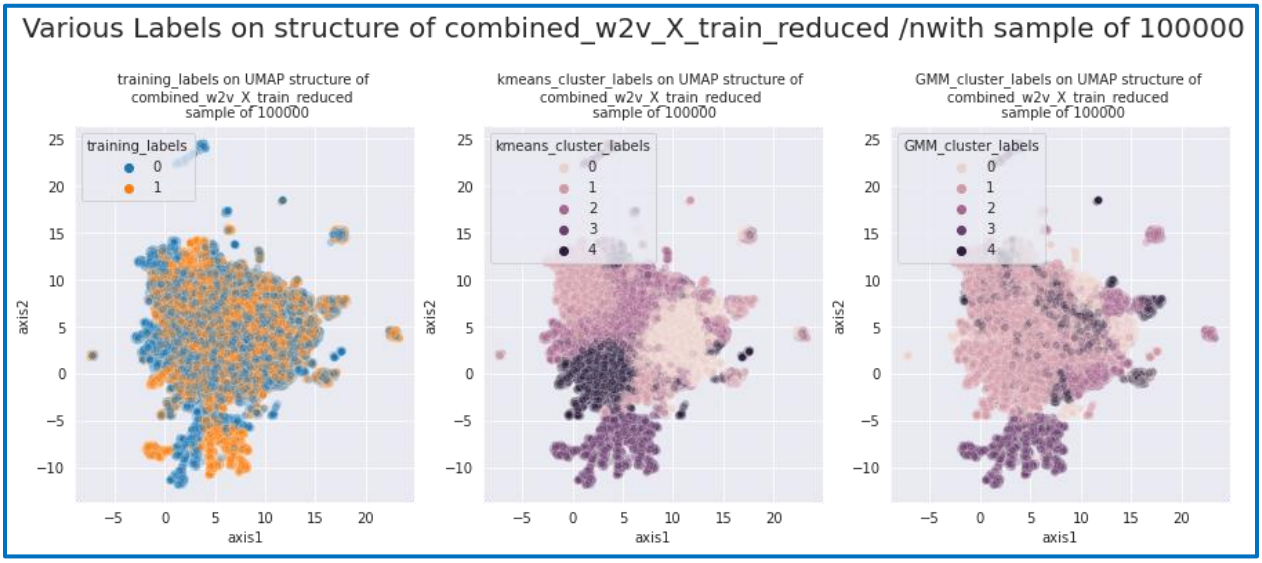
Appendix Figure 4. Mean Average Age of Acquisition for words in a sentence (*link)*

Label 1 and Label 0 have a similar age of acquisition



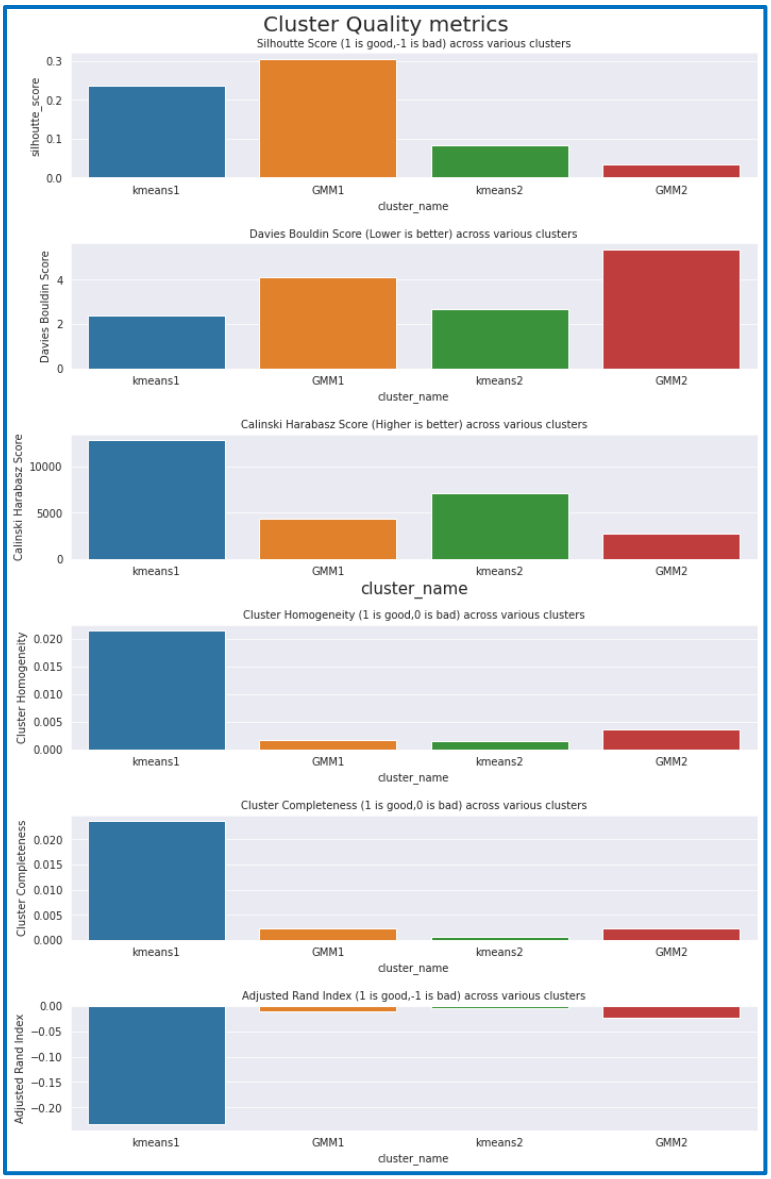
Appendix Figure 5. Final engineered Features Evaluation (*link)*

The best performing features on the final 50,000 sample Logistic classifier were *combined bigram bag of word*, *combined bigram tf_idf weights*, *combined word2vec* and *token attribute count + POS tag count*



Appendix Figure 6. Training label overlap for cluster structure of *reduced combined word2vec* (*link)*

The clusters for *reduced combined word2vec* feature have less overlap with training labels though



Appendix Figure 7. Cluster Evaluation (*link*)

KMeans1 is the best cluster solution amongst the 4 clusters we evaluated. It does better on 4 out of 6 evaluation metrics (except Silhouette score and Adjusted Rand Index)