Nine Models for Classifying Text

Cameron Grams

Motivation, Goals and Intent

I have a background teaching English as a Second Language so the goal of identifying difficult text is appealing. This project explores nine different techniques for text classification on a labeled dataset. The text classification problem was based on the question *“does a sentence need to be simplified in order to be more easily understood?*” This project was an opportunity to compare the different approaches and models for text classification. All of the code supporting this project can be found on my GitHub site. I did not see any ethical considerations surrounding this project since the dataset was publicly available and specifically provided for analysis.

With my background in education I brought with me some assumptions about what makes text difficult; such as that low frequency vocabulary will create difficulty, sentence length will add to difficulty, some types of sentence structure (passive use for example) will create difficulty, and vocabulary or topics specific to certain fields or occupations will create difficulty. This project focused primarily on vocabulary use and simple structure as the source of sentence difficulty.

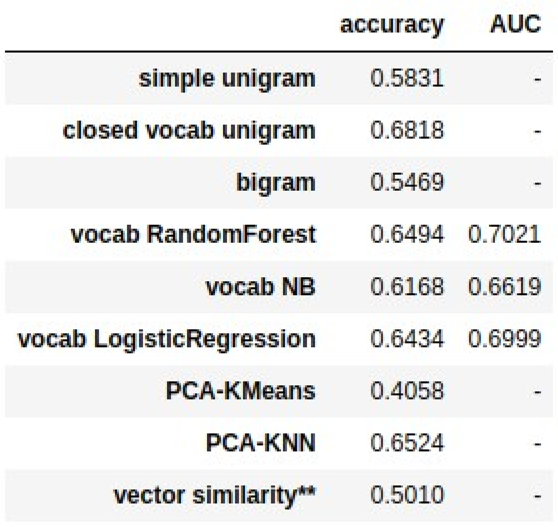
Two datasets were provided to support this project[1], a training dataset that has the sentences labeled as either needing (1) or not needing (0) simplification, and a test dataset. The test dataset was used in a class prediction task with the best performing model from the training phase. The results were uploaded to the Kaggle website to confirm the effectiveness of the classification from the model. The labels for the training dataset sentences were previously established.

I divided model development into four stages:

1. Stage One employed three standard Machine Learning models using features derived from the vocabulary of the dataset.
2. Stage Two employed three different probabilistic language models based on term distribution in the two classes; the assumption was that sentences that need to be simplified and sentences that do not need to be simplified will have different term distributions. This was born out by the models.
3. Stage Three employed Principal Component Analysis and KMeans clustering on the vocabulary model derived in Stage One to explore the underlying distribution of the two classes.
4. Stage Four employed a vector space model to attempt classification using the cosine similarity of the class vectors.

1 From the Kaggle site: https://www.kaggle.com/competitions/umich-siads-695-fall21-predicting-text-difficulty/data last accessed on 24 May 2022.

After Stage Four a classification task was performed on the test dataset with the best performing model. An overview of the accuracy scores from the models on the training dataset is as follows:



	accuracy	AUC
simple unigram	0.5831	-
closed vocab unigram	0.6818	-
bigram	0.5469	-
vocab RandomForest	0.6494	0.7021
vocab NB	0.6168	0.6619
vocab LogisticRegression	0.6434	0.6999
PCA-KMeans	0.4058	-
PCA-KNN	0.6524	-
vector similarity**	0.5010	-

** used 25% of the training data, details below.

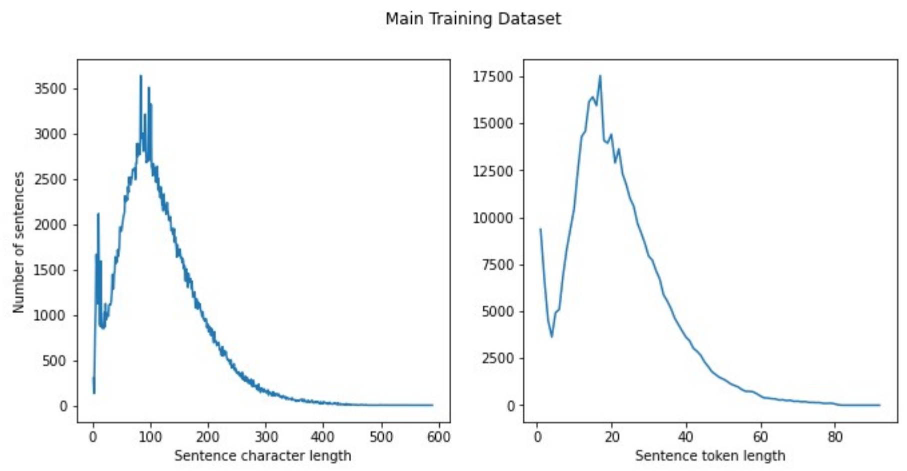
Data Exploration and Text Pre-Processing

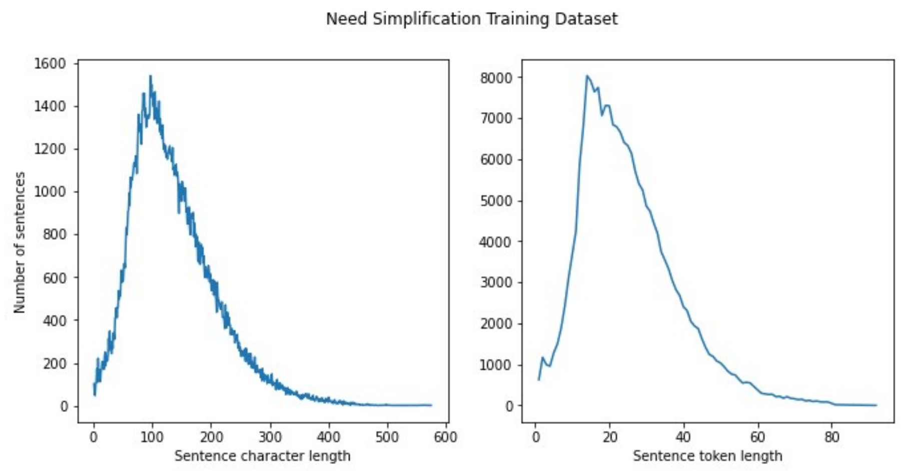
The first step was data exploration to get a sense of what was in the dataset and how the sentences were configured. There were 416,768 sentences in the training dataset, evenly divided between sentences needing simplification and those that did not (there were 208,384 in each class). Inspection of the head and tail of the dataset revealed that many sentences were single words or contained many non-text symbols. Of the 416,768 total sentences 69,412 were duplicate sentences.

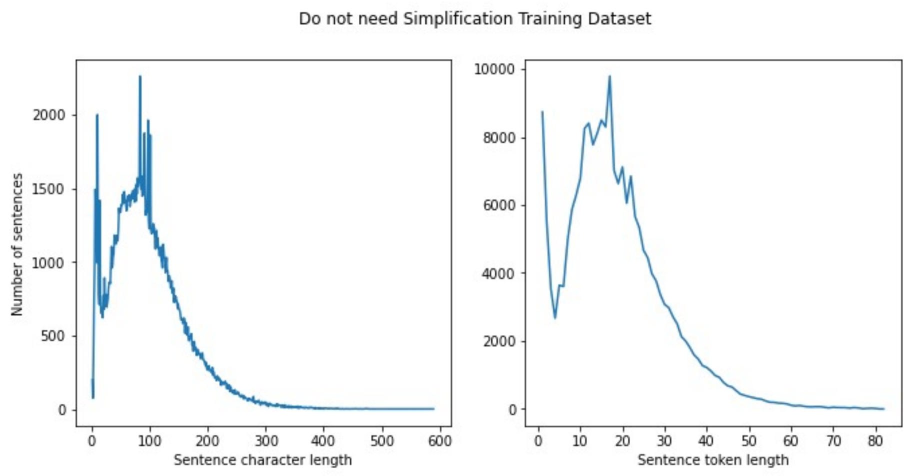
The sentences were pre-processed using the nltk library.[2] The goal was to take the single strings from the sentences as given and create lists that corresponded to the tokens in the sentences. The sentences were put into lowercase before tokenizing. No stopwords were removed; articles and other high frequency words may contribute to overall readability and therefore may be a component of how difficult the sentence was judged to be. An assumption was made that stopwords are a feature of the two classes.

Next I made a comparison of the distribution of sentence length by character and sentence length by token for the main training dataset, the class that needed simplification and the class that did not. A graphic representation of sentence length distribution follows:

2 https://www.nltk.org/ as accessed 26 May 2022.







The y-axis shows that the class that needs to be simplified (middle pair) has fewer short sentences, both by characters and token length. This is what would be expected; more difficult, wordier sentences are more likely to be judged as needing to be simplified.

It is significant that the sentences are all less than 80 tokens in length; the Flesch-Kincaid Readability measure works best for sentences that are longer than 100 words (or tokens), so the value of the Flesch-Kincaid Readability measures may be questionable.

Supervised Learning Techniques

Vocabulary Matrix
Stage One and Stage Three required placing the sentences in vector form. Each sentence was represented as a vector by defining the sentence as a collection of values derived from different assessments of the vocabulary in the sentence. There were eight features for each sentence in total.

In support of this project there were a number of English linguistics collections that were available to assist with determining word difficulty.[3] Each sentence vector had features that were a count of the number of words from the sentence that were present in each of those collections:

- **Concreteness**: the collection included words that had been rated for their concreteness and how well recognized the word was by evaluators.[4]
- **Dale-Challe**: a collection of 3000 elementary English words known by 80% of US children in the 5th grade.[5]
- **Age of Acquisition**: a collection of words coded for the estimated Age of Acquisition for the word and the percent of judges that recognized the word.[6]

Additionally each sentence was given scores based on the Flesch-Kincaid Readability formula for ease of reading and assessed grade level of the material. As mentioned this was likely an inaccurate measure due to the short length of the sentences, but the intent was to provide an additional feature to distinguish the two classes so these two measures were included.

These four techniques produced a matrix of the shape (416768, 8), where each feature represented a score for the sentence using one of these techniques.

Stage 1: Vocabulary Models
Using the vocabulary matrix and the assigned labels it was possible to compare conventional supervised learning models based on their class predictions using this matrix. Each model was primarily evaluated for accuracy, where the true labels were the focus. This score was considered most appropriate since it is most easily compared to the probabilistic models.

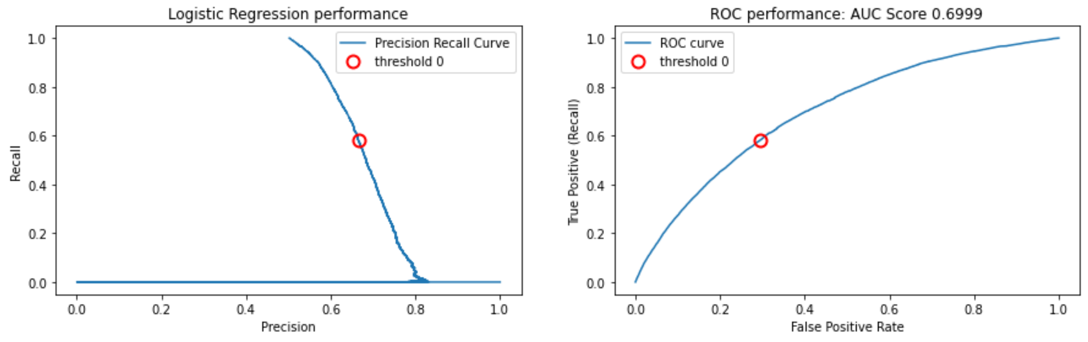
Each model had five fold cross validation performed on the data, where accuracy was assessed based on five different assignments to internal train and test splits in order to ensure that any

3 https://www.kaggle.com/competitions/umich-siads-695-fall21-predicting-text-difficulty/overview as accessed 26 May 2022.
4 [5, 6] As accessed on the Kaggle site, see references for original research.

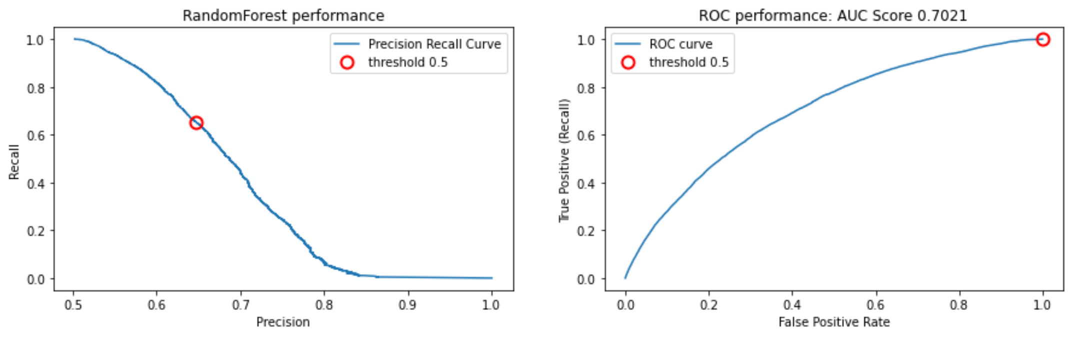
trends in the data did not affect model evaluation.   A mean of the five prediction values was used for the model accuracy score.

Additionally each model was assessed for the potential tradeoffs from adjusting the thresholds for precision and recall. Examining the curve defined by false-positive/true-positive decisions for the model thresholds produced the measure of what level of accuracy would be possible sacrificing a certain level of precision (expressed in the AUC Score). All models were compared against a Dummy Classifier that had an accuracy of 0.5 for this dataset. The results were as follows:

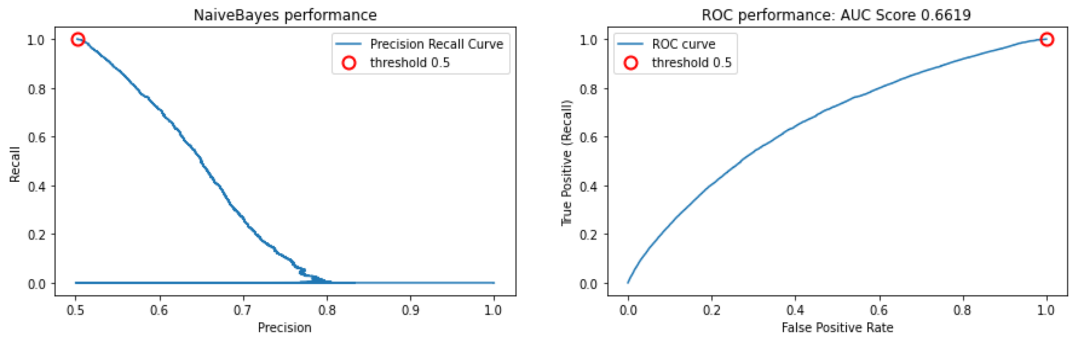
Logistic regression: 0.6434



RandomForest Classifier: 0.6494



Gaussian Naive Bayes Classifier: 0.6168



The AUC numbers indicate that with tuning or adjustment it would have been possible to achieve higher accuracy at the expense of greater false positive ratings for LogisticRegression Model.

Stage 2: Probabilistic Models

The next stage attempted to classify the sentences using probabilistic models based on the token distribution within each class. The text was pre-processed as before, but then the sentences were divided into the classes of those needing simplification and those that do not. For each class, class token probability was determined using one of three techniques. Once the token probability by class was determined it was possible to see which class had a higher probability for each sentence and classify the sentence accordingly.

The three techniques used for deriving token probability were as follows:

Simple Unigram Language Model: accuracy 0.5831
The first probabilistic language model looked at the raw counts of tokens in each class. The language model was built by determining individual token probability; by counting each token in the class and dividing by the total number of all tokens in the class. Sentence probability was determined by multiplying the probabilities of the individual tokens as determined by each class distribution. This calculation was performed using both class distributions for each sentence and the sentence was classified based on which distribution produced the higher probability. The results using his technique were slightly better than the 0.5 classification of the DummyClassifier.

Unigram Language Model with Closed Vocabulary: accuracy 0.6818
The second probabilistic model used the same principle for deriving unigram token probabilities, but in this model a minimum appearance threshold was established; if a given token did not appear in the class distribution a minimum number of times it was recoded as “<unk>”. This had the effect of reducing the number of individual tokens in each class, however the degree that this affected the classes proved to have a significant difference and contributed to an improvement in the ability of the language model to distinguish between the classes. Trial and error led to a threshold of 11 minimum appearances as producing the highest accuracy score.

Bigram Language Model with Laplacian Smoothing: accuracy 0.5469
The third probabilistic model used the bigram distribution of each class to predict sentence probability. Start (‘<s>’) and end (‘<e>’) tokens were added to the sentences at pre-processing. Laplacian smoothing was used to ensure that no zero probability bigrams altered overall sentence probability. Sentence probability was again assessed by multiplying the probabilities of the individual bigrams in the sentences. This had the worst performing classification accuracy of the three probabilistic models, but was still better than the Dummy Classifier. This could be due to the large number of short sentences in the dataset, but that is not immediately clear.

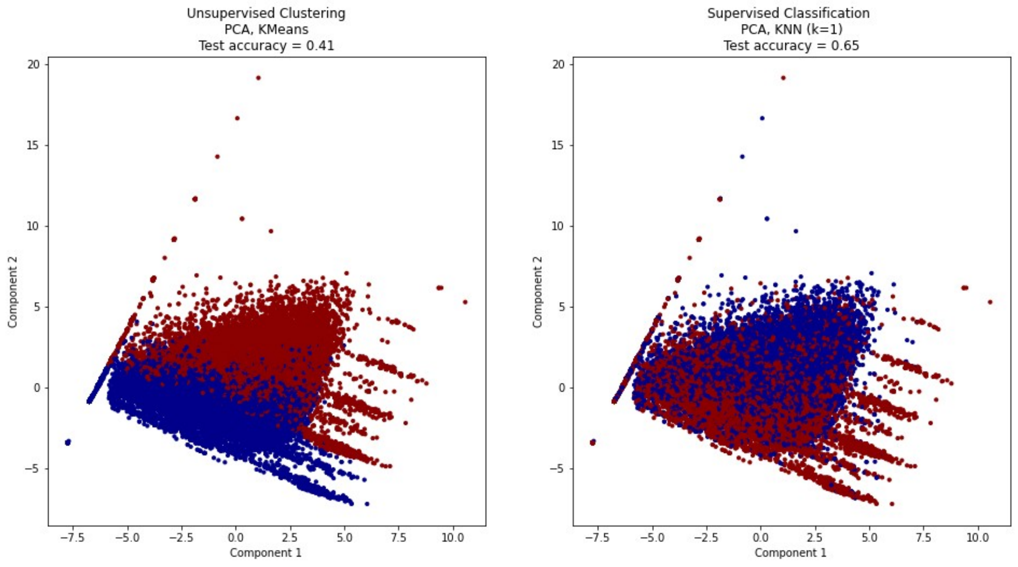
Unsupervised Learning Techniques

A question that we might want to ask is *“is there an underlying structure to the data, do the classes form clusters in the vector space?*” To explore this question I used Principal Component Analysis and KMeans Clustering to try and identify any groupings of associated sentences.

There are 8 features in the vocabulary matrix defined in Stage One. In order to plot any clusters I used Principal Component Analysis to reduce this feature space to 2 so that the sentences could be plotted and associations seen more clearly.

Stage 3: KMeans and KNearest Neighbors comparison

As an unsupervised learning technique KMeans Clustering makes an arbitrary division among the data points into the number of clusters we anticipate seeing. Since there are only the class of sentences that need to be simplified and the class of sentences that do not need to be simplified, assigning the KMeans a number of clusters as 2 was straightforward. After an arbitrary division, the algorithm iterates until the clusters are finally defined by having central points that represent the mean of each of the points (sentences in our case) in the cluster (in our case for two divisions). This assignment can be compared to the known classifications of the sentences to obtain an accuracy score for how well the sentences can be classified by location in the reduced feature space.



The accuracy obtained by defining the class of sentences by cluster position was worse than the DummyClassifier at 0.41.

This leads to a follow-on question, *“does location within the reduced feature space have any predictive value*”? Employing the supervised learning technique of KNearest Neighbors produced an accuracy of 0.65, the second best predictive value seen so far. In the reduced feature space assigning sentences the classification of their nearest known neighbor is an effective way to distinguish sentences that need simplification from those that don’t.

This raises the question, *“can the vector space defined by vocabulary be used more effectively for classification?”*

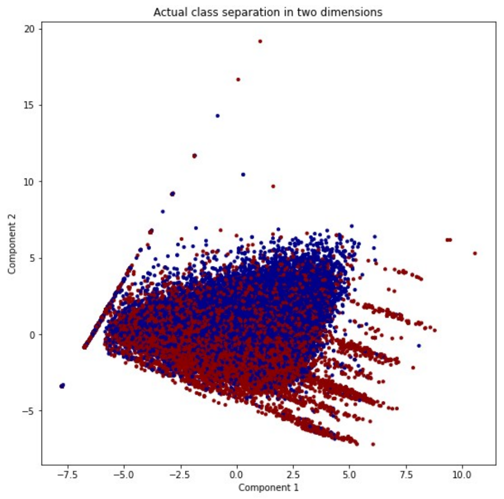
Vector Similarity Classification

To test the possibility of using the vector space for classification the text of the training dataset was placed into sparse vector form using scikit-learn's term frequency-inverse document frequency module TfidVectorizer. The result was a sparse vector of 142,341 terms for each of the sentences in the training dataset. Each class was separated and a mean vector was derived from all of the sparse vectors in each class.

Stage 4: Cosine Similarity Measure Model

The intent was to perform a cosine similarity comparison for each of the sparse vectors in the dataset with the two mean class vectors, and then assign the sparse vector the class that had the greater cosine similarity. This proved too computationally intensive for the entire dataset of 416,768 sparse sentence vectors. As a proof of concept 25% of the training dataset was sampled and classified using this technique. The process still required two hours and six minutes to complete.

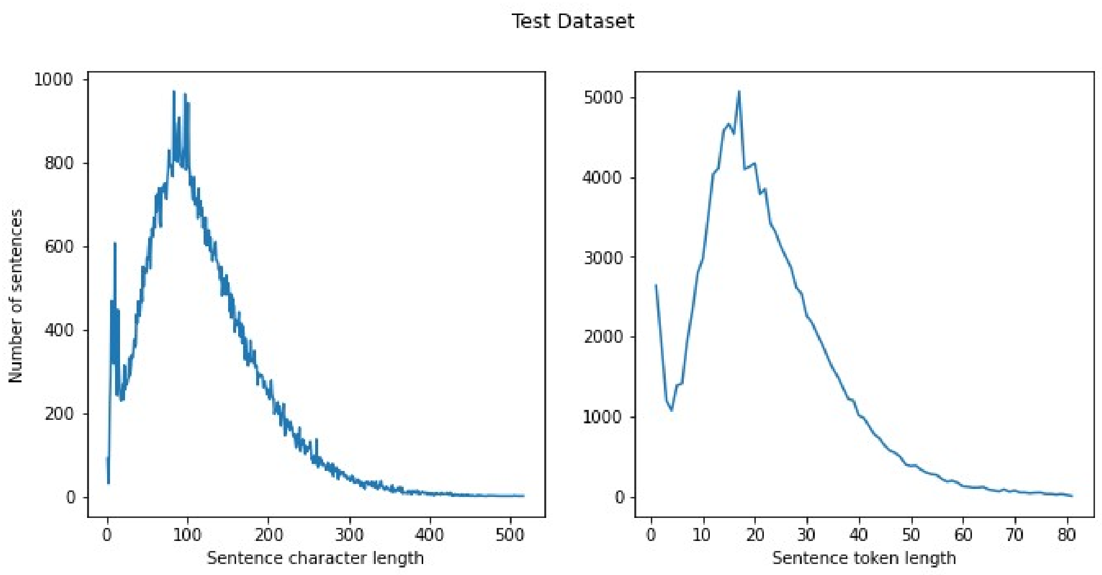
Unfortunately this technique proved little better than chance with an accuracy score of 0.501. When you examine the plot of the true values of the two classes in two dimensions the degree that they are intermingled is clear:

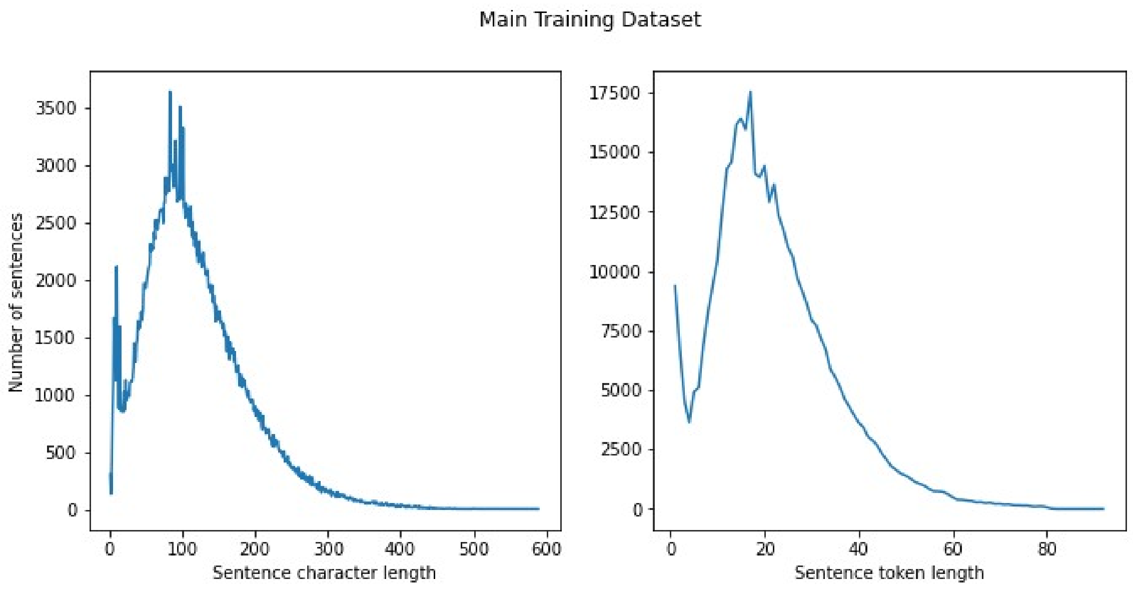


Average vectors using this technique did not produce the degree of discrimination needed to peform effective classification.

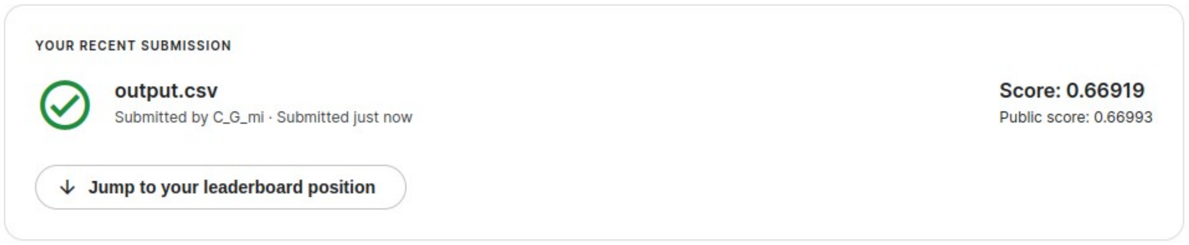
Model Winner and the Test Dataset

The test dataset is smaller than the training dataset, with 119,092 sentences. An exploration of the test dataset showed that the distribution of sentence lengths appeared similar and had a comparable number of duplicate terms proportional to the size of the training dataset. The training dataset was approximately 3.5 times larger.





Based on the comparison of the nine models with the training dataset the closed vocabulary unigram model with a threshold of 11 was chosen to classify the test dataset. As with the training dataset, the test dataset was pre-processed by placing the sentences in lowercase and tokenizing the terms. The probability distributions for the two classes derived from the training dataset were used to evaluate the probability of the tokenized sentences in the test dataset. The tokenized test dataset sentences produced a probability for each of the class probability distributions. Classes were assigned based on whichever distribution produced the higher probability. The results were uploaded to the Kaggle API to determine the final accuracy score:


YOUR RECENT SUBMISSION

output.csv
Submitted by C_G_mi · Submitted just now

Score: 0.66919
Public score: 0.66993

This result underperformed the language model in the training dataset by 0.0126, but it seems that that should be expected. More significantly this model underperformed the top of the leader-board by 0.11225, proving there is more to learn in this world.

Observations, conclusions, and recommendations

This project sought to compare different models for text classification and determining sentence similarity. The nature of the dataset made some of the models more effective than others. Some of the starting assumptions, that sentence length and word frequency would facilitate the ability to distinguish between the classes, seem to have been born out.

There is clearly room for improvement in all of the models that were explored. The AUC numbers indicate that the LogisticRegression model may be able to be practically tuned for higher accuracy. Additionally no neural nets or deep learning techniques were compared, so there are definitely additional areas to explore.

References

Brysbaert M, Warriner AB, Kuperman V. Concreteness ratings for 40 thousand generally known English word lemmas. Behav Res Methods. 2014 Sep;46(3):904-11. doi: 10.3758/s13428-013-0403-5. PMID: 24142837.

Kevyn Collins-Thompson and James P. Callan. 2004. A Language Modeling Approach to Predicting Reading Difficulty. In Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics: HLT-NAACL 2004, pages 193–200, Boston, Massachusetts, USA. Association for Computational Linguistics.

Kuperman, V., & Stadthagen-Gonzalez, H. (2012, April 16). Age-of-acquisition (AoA) norms for over 50 thousand English words. Crr.Ugent.Be. Retrieved May 28, 2022, from http://crr.ugent.be/archives/806

Müller, A., & Guido, S. (2016). Introduction to Machine Learning with Python: A Guide for Data Scientists (1st ed.). O'Reilly Media.

Scott, B. (ca. 2015). THE DALE-CHALL 3,000 WORD LIST FOR READABILITY FORMULAS. Readability Formulas. Retrieved May 28, 2022, from https://www.readabilityformulas.com/articles/dale-chall-readability-word-list.php