Reducing Sentence Complexity Through Machine Learning

By Phil Michaels, Daniel Haver & David Dunstan

Motivation:

In the past decade there has been an explosion of online open-source resources such as Wikipedia. With individuals increasingly relying on Wikipedia as a quick and convenient source, it is important to maintain the quality of the posts. While the content is peer-reviewed and monitored, the need for automated real-time feedback to users can further improve posts as they are created. To that end, we have obtained a dataset that includes over 400,000 sentences, each with an associated label of whether a reviewer thought they required simplification or not. The source of this text and provided labeling was Cristina Garbacea and was accessed via the SIADS 694/695 Kaggle page. While there are many examples of natural language processing (NLP) tasks, the majority of these involve the distillation of insight for applications such as sentiment analysis or machine translation. The problem of sentence simplification is unique in that the semantics and structure of the sentence as well as its grammatical correctness are central to the problem at hand. We herein describe a featurization, analysis and modeling of this data set to automatically predict whether or not a given sentence requires simplification.

Exploratory Data Analysis & Featurization:

When approaching the problem of sentence simplification through NLP, there are a multitude of techniques that can be applied. However, it is also important to generate some early understanding of the available data. To that end, our data set consists of 416768 rows and two columns, 'original_text', which is the original text of the sentence, and 'label' which is a binary value indicating whether the sentence requires simplification (1) or whether the sentence is acceptable (0). A few example rows are shown below.

	original_text	label
10946	However , in April 2005 he was cleared of all charges .	1
391560	Uerkheim is a municipality of the district of Zofingen in the canton of Aargau in Switzerland .	0
139792	The Gregorian calendar uses 1901 to 2000 .	1
270332	It is also the oldest stadium in Division I Football Bowl Subdivision .	0
251086	Some historians say that in 1607 , there was a slave rebellion in the Colombian town of Remedios .	0

The sentences also include punctuation and some artifacts from the decoding process such as -LRB- and -RRB- (left round bracket and right round bracket). Overall the data set is balanced with exactly 208384 of each class. To facilitate model evaluation and development we split the dataset into a training set, development set, and testing set of 80:10:10% of the data respectively. The split was confirmed to maintain the class balance of the original dataset with relative counts shown below.

	dataframe	positive class	negative class	total count
0	train	166548	166866	333414
1	dev	20946	20731	41677
2	test	20890	20787	41677

To predict the class from the original text, we first needed to construct additional features for the dataset and build an understanding of how these features might relate to sentence complexity. One common technique in NLP is the use of a Bag-of-Words (BOW) type vectorization, where each word is counted in the text and the sums are used to construct a vector. A related form of NLP featurization is the Term Frequency Inverse Document Frequency (TfIdf), which will construct a vector from the frequency of a word in the text relative to the word's frequency in the overall corpus. These words can also be constructed from short phrases in the case of an n-gram model, where sets of words are taken together. (Tfidf). For this data set, using Sklearn's TfIdf vectorizer, we obtain a sparse matrix of 85909 columns using n-grams of length 1-5 and a minimum frequency of 20. The TfIdf matrix was used to fit a logistic regression classifier using the training data. The prediction values and the model's probability of a text being in the simple class were added as features to the dataframe for use and comparison to later prediction tasks.

Given the domain knowledge of sentence structure and difficulty, it was important to explore simple sentence metrics such as word count, syllable count, character count. These were supplemented with part of speech (POS) tag counts from the Natural Language Toolkit (NLTK) part of speech tagging function. (POS) The part of speech for each word was extracted using the NLTK and then the count of each resulting POS was appended to the dataframe. In addition, named entities were extracted using the Spacy library (Spacy) and the named entity count was added to the dataframe.

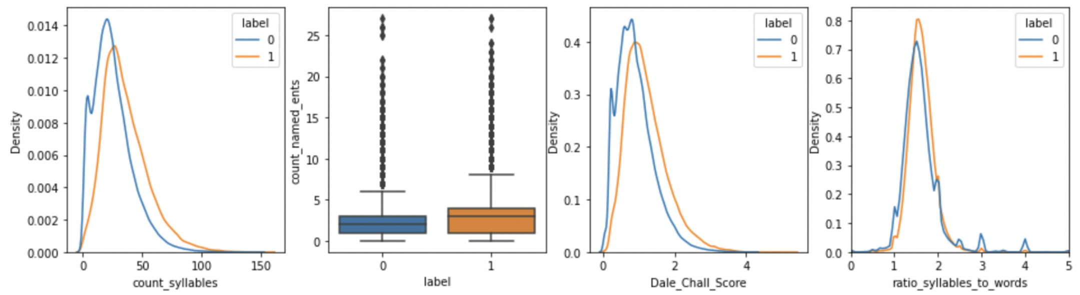
To supplement the original text with additional domain knowledge, we utilized Age of Acquisition (AOA) data, which captures the age at which a word is typically learned. The original AOA data source was obtained as a .csv file from AOA. The AOA data was transformed into a pandas dataframe and each word in the split original text was looked up in the dataframe. The age of acquisition, percent known were extracted and then the max, mean, median and standard deviation were added as features as well. To assess the correctness of the words, a Brysbaert correctness score was calculated. A .txt file containing the correctness scores was obtained (Brysbaert) and was joined to the dataframe as above. Lastly a difficulty score was calculated using the DaleChall approach (Readability), where each word was searched in the DaleChall list which was imported as a text file. While DaleChall scores were originally intended for entire text documents, a DaleChall score was calculated as below for our sentences.

$$\mathit{Fraction\ of\ Words\ Not\ in\ DaleChall}\ x\ 0.1579\ +\ 0.0496\ *\ \mathit{Number\ of\ Words\ in\ the\ Sentence}$$

The resulting features were then compiled into a dataframe for utilization in subsequent tasks. We also performed a cursory analysis of the features. While it does not capture non-linearly related features well, a simple correlation was performed between all the features and the label. The top 10 features are shown below. While none of these correlations are particularly strong, they suggest that the complexity of a sentence might be correlated with its length. The presence of the TfIdf model derived features also suggest that ensemble modeling might be a useful approach to this problem. Lastly, the Dale_Chall_Score, which is one of the engineered features is also present, however it is also highly correlated with the other length-based features (r=0.99 with num_words).

```
label                    1.000
tdif_prob_zero           0.521
tdif_preds               0.448
count_syllables          0.285
num_chars                0.283
count_chars              0.281
count_unique_items       0.275
count_unique_words       0.269
count_unique_words_ns    0.267
Dale_Chall_Score         0.264
```

Exploring the kernel density estimation plots as well as box plots of certain features is illuminating as it shows that there are differences between the means and the distribution of some of the features between the labels. For instance, if we explore syllable count and Dale-Chall score, these both show a shift towards higher means in the requires simplification class. Likely due to the large sample size, the difference in the means for the vast majority of features is statistically significant above the 95% level in a student's t-test. The number of named entities also follows a similar trend as shown in the box plot below. Finally, the ratio of syllables to words, a measure of how many syllables on average each word has in the sentence, shows a high degree of similarity between the means. However there are a number of smaller groups here that appear to show a higher representation in the simple sentences.



To explore the effect of punctuation, beyond simple counts, we explored ways to build a more complicated punctuation score. The API from Readability API was utilized to generate a punctuation based grade-level estimate. Unfortunately, the API did not allow for batch document submission and the speed of individual sentence requests was prohibitive to running this for the entire data set. From a test of 3000 samples we could observe a difference in the mean punctuation score values, but additional work to parallelize this API call would be needed to apply to the larger dataset and make it useful in practice.

K-means Clustering of TfIdf-Vectorized Text and LSA Feature Reduction:
As a first approach to unsupervised learning we had the idea to apply K-means clustering to our TfIdfvector transformed text. Given the complexity of this task, it seems like a linear model such as Logistic Regression may not be able to effectively divide data into two classes. A sentence may be considered complex for such a large number of reasons. K-means would be used to provide cluster labels as a new feature in the final supervised prediction model.

An important parameter of K-means is the k-value itself, representing the number of clusters the algorithm will find. One method of determining an ideal k-value is called the elbow method, in which you iterate through increasing values of k, and at each iteration record the inertia value. This inertia value is the within-cluster sum of squares calculation and minimizing this value means forming more coherent clusters, although it does make the assumption that clusters are uniform spheres.

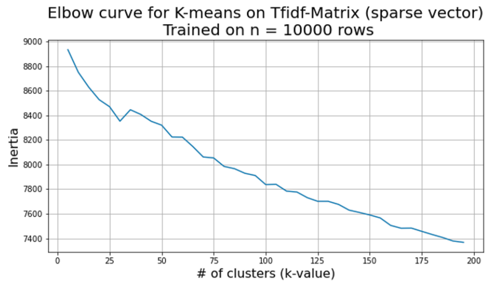
For the first attempt we fed a simple TfIdf sparse vector directly into the K-means algorithm at a few iterations. This method was incredibly slow and even with k-values between 5 and 200 at increments of 5, the elbow plot below showed that inertia continued to drop at an almost linear rate. Train time for this iterative k-means training with TfIdf vector data took nearly 1 hour, so as was done with many of the computationally expensive models, models were trained on a subset of the training data, 10,000 rows.

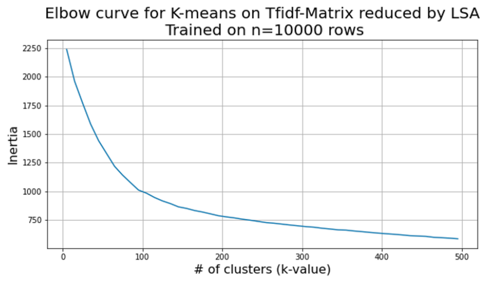
Further reading of the sci-kit learn documentation suggested that K-means is prone to the curse of dimensionality (K-means), and for data sets with many features it is recommended to first apply a feature reduction algorithm. Truncated Singular Value Decomposition (SVD) when applied to text is most commonly referred to as Latent Semantic Analysis (LSA) and is a common approach for dimensionality reduction. This approach is similar to PCA, but does not first center the matrix as PCA does, making it out-of-the box compatible with sparse matrices like the TfIdf matrix. This technique approaches the text analysis problem as a mathematical matrix decomposition problem, and has the ability to group similar words and meanings based on their usage in all provided examples (LSA). In this way, it also does a type of topic modeling, which we hope to be useful for the text complexity classification modes.

The scikit- learn documentation suggests using 100 components with TruncatedSVD for an LSA application, but we also looked at the explained variance of found components up to n = 500. Looking at an explained variance plot (similar to the elbow plot) showed a significant drop off in variance explained beyond 100 components, so we settled on using n_components = 100 for this application. Using this LSA approach we were able to reduce the simple TfIdf sparse matrix with 1293 features, to a matrix with only 100 features. We looked at the right singular vector matrix via TruncatedSVD.components_ to gather which terms had the highest values for each component, then sorted these terms by highest to lowest value and generated a dataframe to help understand how the LSA topics are forming.

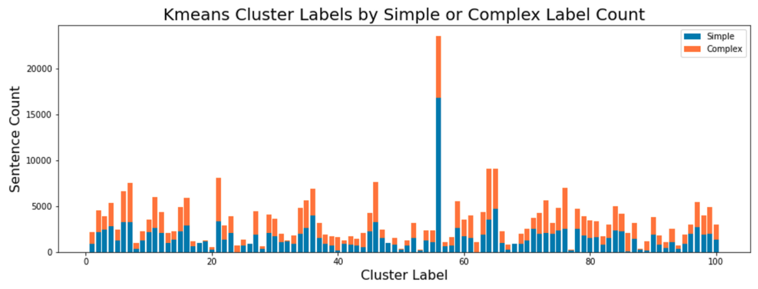
	Top_Term	2nd_Term	3nd_Term	4th_Term	5th_Term	6th_Term	7th_Term	8th_Term	9th_Term	10th_Term
Component_61	into	victoria	very	brand	fish	despite	until	process	dark	director
Component_60	olympic	fish	fiction	january	taken	process	1965	louis	1974	germany
Component_19	000	just	discovered	here	australian	english	has	put	sur	19
Component_97	national	king	germany	makes	footballer	discovery	ten	operating	january	studied
Component_49	dark	birth	brand	musical	articles	either	region	former	center	male

With the reduced feature matrix, we once again tried the elbow method of analysis with K-means. First we ran the same cluster increment range as was used for the TfIdf sparse vector. The models trained in significantly less time with the reduced feature matrix, and the elbow plot did appear to be leveling out, so we revamped the experiment to iterate between k values of 10 and 500 in increments of 10. This time an elbow shape did form indicating the best cluster number to inertia trade off to be between 100 and 200 clusters. It would make sense that this elbow location is influenced by the number of components chosen in the LSA reduction. The pre and post LSA reduction K-means elbow plots are shown below for comparison.





Once a successful K-means elbow plot was generated using the LSA matrix, it was decided to fit K-means with a k-value of 100 using the LSA transformed train split data. At this point it was possible to predict cluster labels, but we needed to also evaluate if any of these clusters would prove useful for the complexity labeling. Using a stacked bar plot in matplotlib, we looked at the number of complex and simple labels for each K-means cluster label. While most clusters seem to have a balanced mix of each label, it is possible to see in the plot below that some clusters are composed of almost entirely one label.



Evaluation of these clusters, as well as the clusters with nearly a 1:1 ratio of simple to complex class labels should help develop understanding of where the TfIdf or LSA features can be useful for classification, and where they are not. To help identify clusters with high or low relative population of one class compared to the other we calculated a relative_difference column in pandas for each cluster with formula **np.absolute(label 0 count - label 1 count) / (label 0 count + label 1 count)**. Using calculated relative differences in simple vs complex label counts we were able to determine the top 5 most differentiated clusters, clusters in comparison with 5 clusters with the least differentiation. The charts below show the top and bottom five K-means cluster labels sorted in descending (left chart) and ascending (right chart) order according to relative difference values with complexity/simplicity label counts for each of these clusters. This further demonstrates that indeed some clusters are highly differentiated.

label	0	1	relative_difference
cluster			
61	1	1065	0.998124
25	865	1	0.997691
23	2	729	0.994528
47	1021	11	0.978682
17	996	13	0.974232

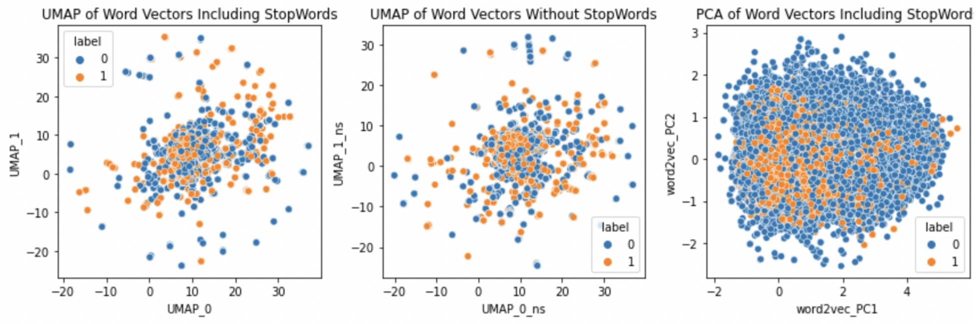
label	0	1	relative_difference
cluster			
48	796	796	0.000000
5	3309	3306	0.000454
96	2723	2729	0.001101
94	944	947	0.001586
19	251	250	0.001996

With new features engineered from both LSA and K-means models, we wanted to determine how these might affect a base level model score, so three default Random Forest Classifiers (RFC) were trained using TfIdf, LSA, and LSA with K-means labels as training data. For speed, only 10,000 training rows were used. Results for these exploratory models were interesting, showing that by f1_score and accuracy the TfIdf scored best, with the LSA model and LSA with K-means almost identical in 2nd and 3rd place. This is reflective of the fact that dimensionality reduction does lose some useful information and in the case of adding K-means labels can introduce additional noise, but the trade-off of computational efficiency and improved data storage may still make this worthwhile. We even took the idea of dimensionality reduction to an extreme by training a fourth model on just the K-means cluster labels alone and found that the result was still ~10% better than the uniform random “Dummy” model. LSA and K-means labels were added to feature datasets for use in supervised model development. With more time it might be possible to improve K-means cluster scores by bootstrapping the K-means model and taking the best cluster labels from the bootstrap results to account for the random nature of K-means.

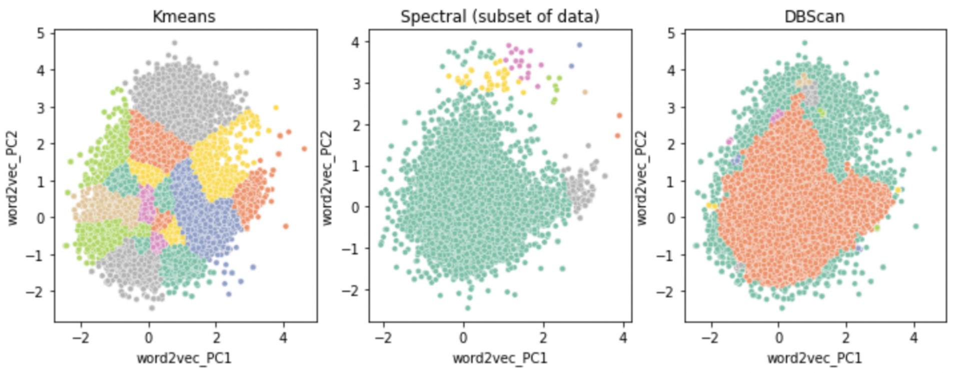
	Model - Features	Train Accuracy Score	Dev Accuracy Score	Train F1_Score	Dev F1_Score
0	RFC - TfIdf	0.656076	0.645224	0.645224	0.646775
1	RFC - LSA	0.636356	0.630492	0.630492	0.639817
2	RFC - LSA & Kmeans	0.636053	0.626725	0.626725	0.638083
3	RFC - Kmeans	0.574514	0.576265	0.576265	0.593031
4	Dummy - TfIdf	0.498446	0.494781	0.494781	0.496725

Word2Vec Application & Topic Extraction:
We also explored Word2Vec extraction of topics from the dataset to help build additional features for the supervised portions, as well as provide insight for the users. One could assume that simple and complex sentence structure might be perceived differently based on the field of study, and a resource such as Wikipedia would have broad exposure. To implement this, we deployed a Word2Vec based approach. Word2Vec involves utilization of a pre-trained external neural network model that associates each word with a vector. (Word2Vec) We utilized the Gensim package and selected the glove-wiki-gigaword-300 model. (Gensim) This model seemed like a good fit because it was trained on a Wikipedia corpus dating from 2014 onwards, so the sentence structure should be relevant to the dataset for this project. We also selected the 300 length vector model because the longer vector seemed likely to better pick-up differences between our relatively short sentences. It should be noted that there is prior work indicating it would be possible to fine tune the Word2Vec model on our dataset, however time limitations prevented us from exploring this further. (Word2Vec tuning)

To apply this to our data set, we adopted an approach based on an example by Dylan Castillo. (Dylan Castillo Word2Vec) First, each sentence was tokenized into individual words. The Word2Vec vector was then computed for each word in the sentence and the resulting word vectors were averaged. This process was repeated a second time, however any word present in the NLTK English stopwords list was removed to assess the effect of stop words on the results. The resulting vectors were then input into two dimensionality reduction techniques, Principal Component Analysis (PCA) and Umap to reduce them to two dimensions for visualization. For Umap, cosine similarity was selected because it made more sense in the semantic context since the angle between word vectors better captures their meaning than their length. Other Umap parameters including n_neighbors and min_distance were varied to produce a plot that demonstrated clusters, but hyperparameter testing was not exhaustive since the computational time required was significant and the parameters did not translate well from a sample dataset. PCA was significantly faster than Umap, and it preserved the overall structure as seen in the figure below. While Umap could likely produce similar or better results, this was reserved for future work since PCA worked well to advance the topic modeling. The use of stop words did not drastically affect the results from the visualization perspective, but did clean-up the topic words.



The word vectors were also utilized to explore clusters. Following the dimensionality reduction, the data was explored with K-means clustering, DBScan, and Spectral clustering. Spectral Clustering was too computationally intensive for use on the entire dataframe and did not appear to do as well at grouping the data points locally and many were classified as outliers in DBScan (shown as teal below). K-means produced a good balance of identifying locally similar points and producing more evenly sized clusters. As with the TfIdf cluster, a breakdown of the clusters both in terms of size and ratio of each data label was run, and the Word2Vec based K-means clusters demonstrated a similar ability to extract clusters that were class biased, suggesting they might be useful features for supervised portions.



To address the topics of the original text, the Word2Vec vectors for each sentence were assigned a cluster using the K-means algorithm. The vectors for each cluster were then averaged with each other, to produce a cluster vector. This vector was then utilized in the Word2Vec similar_by_vector function to determine the top 10 closest words to the cluster. Interestingly, some of the cluster topic words were seemingly similar, however others were quite distinct. As shown in the table below, there is a cluster with very few complex sentences that consists of "r*eferences, websites, articles*" , many of which have original text consisting of a single word. Another cluster is almost all complex sentences and involves several French words and phrases. In the largest clusters such as Cluster 0, the class balance is maintained and the similar words are common words, suggesting that additional stopword and cluster refinement could better produce topics.

	Cluster	Sample_Count	Class_Ratio	Top10_Similar_Words
22	12	2446	0.029845	[references, websites, pages, web, reference, articles, content, videos, text, descriptions]
23	7	1947	0.162815	[mangxamba, tom.fowler@chron.com, 3.6730, ____________________, brett.clanton@chron.com, zety, el1l, rungfapaisarn, http://www.kasparovchess.com, jiwamol]
1	2	8375	0.263403	[region, commune, france, part, in, ., department, northern, southern, southwestern]
0	15	13762	0.373783	[., ,, in, last, first, since, but, only, year, when]
12	0	19032	0.501208	[., ,, but, in, only, when, first, same, time, last]
18	8	15859	0.518255	[example, ., same, so, even, these, well, but, this, because]
11	18	7179	0.518735	[town, ., area, part, ,, region, in, where, located, northern]
14	17	4641	0.942254	[commune, france, department, region, île-de-france, département, pas-de-calais, north-eastern, north-western, alsace]

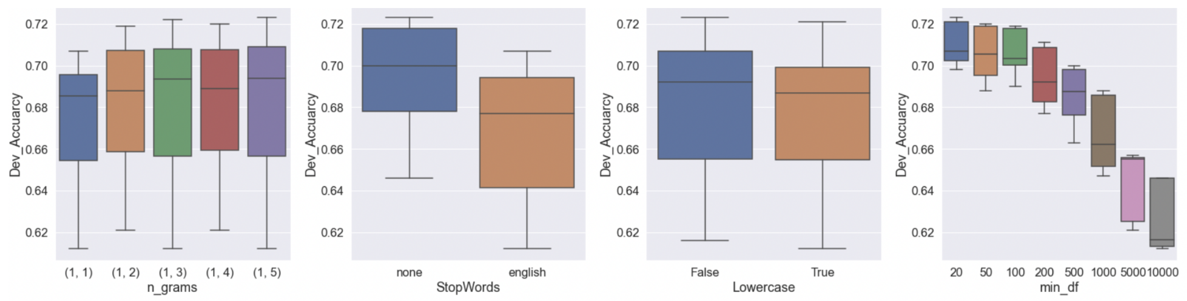
Comparing the original text, even for the unbiased clusters, does reveal some similarities between the sentences. This seems to indicate that further refinement of this approach could yield an effective way to model the topics of various sentences, or at the very least highlight key outliers such as the "references" group for exclusion from the simplification prediction. While this approach was not yet successful at genre classification, it has been shown elsewhere that training on a larger corpus of the text, such as the entire Wikipedia article, might be a more effective approach as the sentences in our original data might not always contain enough information to produce a clear topic. Some example sentences are listed in the Appendix to highlight the selected clusters, and overall these seem to align well with the extracted topic words, suggesting that the approach is effective for this dataset. While it was surprising that both the Word2Vec and TfIdf LSA approaches produced coherent topic results, further work could examine the effectiveness of combining these approaches for more differentiation among the results.

Model development & optimization:
To begin the supervised learning portion of the analysis, we first fit a Logistic Regression, Random Forest, and Stochastic Gradient Descent classifier on the TfIdf sparse matrix to serve as a baseline performance and to compare against other models. We chose to explore the Logistic Regression model due to its speed of training and superior accuracy relative to the other results below.

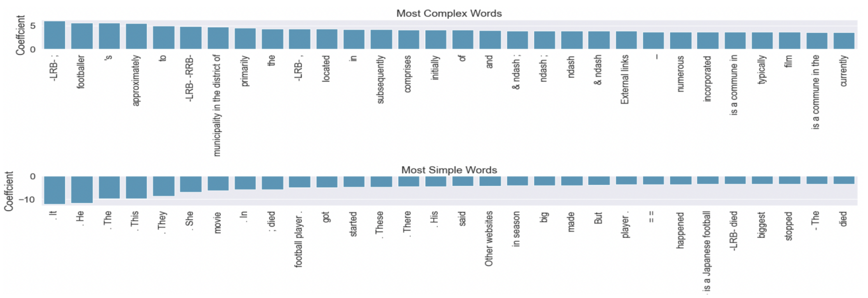
model	Uniform	Most_Frequent	Logistic Regression	Random Forest	SGDClassifier
accuracy_score	0.49	0.5	0.72	0.67	0.7
f1_score	0.5	0.0	0.72	0.67	0.71

We looked at a number of different parameters to optimize the TfIdf and LR model, however the best performance it was able to achieve on the development set was 72.3% with no stop words, n-grams of length 1-5 and min_df of 20, while the worst was 53.7% with English stop words, n-grams of 1-5 and min_df of 10,000. The best model was then applied to the test set and achieved 71.8% accuracy, which we found surprising given the complexity of the task and the relative sparsity of each of word features. It was also noted that this approach handily beat the dummy classification methods by over 20%.

In searching the parameters, the accuracy on the development set we explored lowercase words, stopwords, n-grams from 1-5 and minimum data frequency of 20 to 10,000. The parameter which showed the largest effect on accuracy was minimum data frequency, which provides evidence that eliminating too many words has a negative effect on performance, likely due to the short nature of our text documents. If too many words are eliminated then many sentences do not have sufficient information left to make a prediction. Interestingly, the removal of stopwords generally caused a decrease in performance as well, indicating that these contained useful information for this application. Somewhat surprisingly the n-gram range did not appear to have a general trend on performance, however the longer n-grams (3-5) counts did appear more frequently in the highest scoring models. These also showed more volatility in model performance as they produced vectors with more sparsely populated features than shorter n-grams.

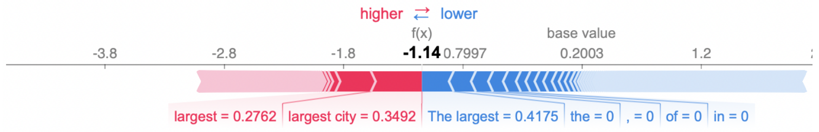


To explore words that attributed the strongest effect to the TfIdf logistic regression model, the model coefficients were calculated and then mapped to the features from the vectorizer and shown below. Interestingly there appear to be a number of punctuation related features including the above noted *LRB* and *RRB*, *";"* &, and *ndash*. There are also several adverbs: *approximately, primarily, subsequently, initially*. A few 4 to 5-grams also made the list "*municipality in the district of*" , "i*s a commune in*", and "*is a commune in the*", which appear to be tied to the French cluster identified in the Word2Vec modeling. On the other hand, the strongest negative coefficients, indicating simple phrases, seem to be simple pronouns combined with a preceding period, and a few references of simple verbs such as "*died, started, said & made*". This might suggest that pronouns and the use of simple verbs can help with reducing sentence complexity.

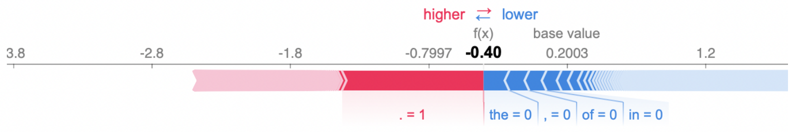


To explore the effect of the words on individual predictions, we utilized the SHAP library (SHAP) to add an additional level of interpretability to the TfIdf predictions. This library allows individual features to be scored providing unique insight into the model's predictions. In the two examples shown below, we can see that for a simple sentence, the model correctly attributes the stop words and indicates "The largest…" as a way to begin a simple sentence. On the other hand, the model is not as well able to correctly classify the somewhat ambiguous text regarding an archive date, likely due to the lack of overall context, so we decided to look at other text metrics to improve upon the Tfidf and Logistic regression result.

```
The largest city is Minneapolis .
Actual Label: 0
Predicted Label: 0
```



```
Archived 2009-10-31 .
Actual Label: 1
Predicted Label: 0
```

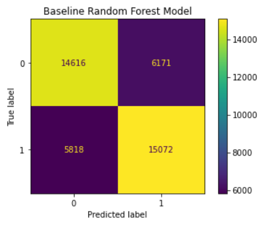


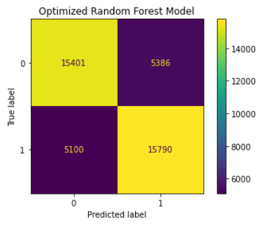
With the intent of improving performance and robustness, we included the expanded feature set into our Random Forest model and began to explore model hyperparameter settings. Random search cross validation with RandomizedSearchCV from the scikit-learn library was utilized to explore parameters using three cross-folds to help to protect against overfitting. The number of estimators included in the final model were varied along with setting for maximum tree depth in each model. The minimum samples needed in a node split and minimum samples needed in a leaf node were also varied. The number of iterations was set to ten to keep run times manageable and the parameter combinations were ranked based on the estimator's score method.

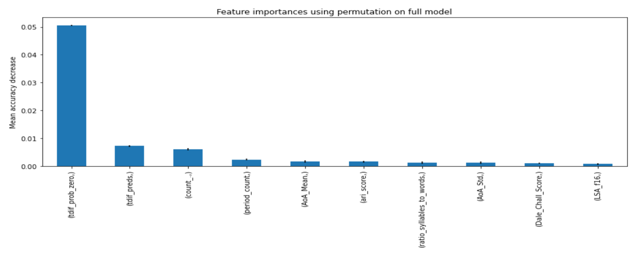
Recall is being used to measure optimal performance. In our use case, we intend to provide authors with a tool that allows them to improve the readability of their content. Identifying simple sentences as complex (false positives) provides more benefit to an author than mislabeling complex sentences as simple (false negative). Since reducing false negatives is fundamental to the scenario, recall is the best metric as it penalizes scores as the number of false positives increase.

After iterating through initial results, best model performance was observed using minimum leaf samples of 2, minimum split samples of 4, maximum tree depth of 30, and 50 estimators. Beyond 50 estimators, the performance on the training data continued to slightly increase but with little improvement to test data performance indicating the model may be starting to overfit. Based on the final parameters, the recall score for the training and test data were 0.981 and 0.756 respectively. The Random Forest model using default parameters produced scores of 0.744 and 0.715 for the same datasets. Additionally, we compared the effect of leaving out the Tfidf predictions as a feature, which resulted in a score of 0.745 on the test set indicating that the ensemble approach is adding value. Further, the relative importance of the features is quite similar here indicating that while not dramatically improving the score these additional features are adding value to the original TfIdf model.

The performance increase led to a 12.7% decrease in false positives (6171 false positives down to 5386) and a 12.3% decrease in false negatives (5818 down to 5100). To understand important features, permutation importance was used to overcome limitations of impurity-based importance such as bias toward high cardinality features and lower generalization of results (Permutation). In both models, the initial term frequency vectorizer results were most important. The importance scores were notably lower in the optimized model which may imply more robustness as the model is less dependent on a single feature. Additionally, the top twenty most important features in the optimized model included one of the Word2Vec principal components and additional features from the LSA dimensionality reduction.

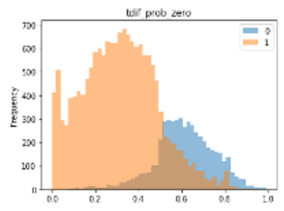


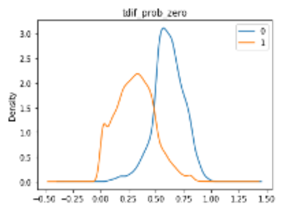


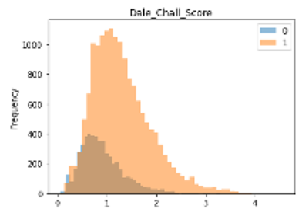


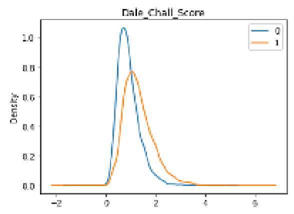
Error Analysis:

To understand false negative classification errors, we examined the distribution of false negative and true positives for the top features identified by the optimal Random Forest model. Histogram and kernel density estimation plots were created for each feature to examine differences between the classification labels.   The variation for the word vectorization predictor features were most distinct, likely indicating underlying discrepancies with the text classification labels. Differences in other features were more subtle but still showed some variation. The figures below show plots for the Tfidf word vectorizer predictor feature and the Dall Chall score.









The Dall Chall score is most sensitive to the ‘word complexity to word’ ratio in a sentence. The results show the mean distribution is slightly lower for false negatives. The sampled text below shows a false negative and true positive result that exemplifies the difference. Tuning the Dale Chall algorithm to provide consistent word complexity counts would likely minimize discrepancies. In addition, Dale Chall may not be particularly good at handling proper-nouns as evidenced by the second sentence.

	original_text	Dale_Chall_Score	rf_preds
8584	Coe was awarded with the first Prince of Asturias Award in sports category in 1987 .	0.817687	0
39013	Ségrie is a commune in the Sarthe department in the region of Pays-de-la-Loire in northwestern France .	1.033816	1

The word vectorization predictors were trained using 80% of the corpus. These features were consistently the top drivers in our classification model and show clear variations in distributions between false negatives and true positives which led us to investigate the original class labels. Two primary issues were identified, duplicate entries with conflicting labels and short word phrases identified as complex. In the training data there were a total of 4006 entries that were duplicates and had conflicting labels. Without knowing exactly how these training labels were generated, seeing the same text input with varying labels is very telling. It reveals a level of subjectivity to the labels which may mean that it would be impossible to get a perfect score for such a model. We reached a point in error analysis where it was not always clear to us as readers why one sentence might be marked as complex, while another is not. What is complex to one reader may not be complex to another. There were an additional 1466 single or dual word entries that appeared simple but were labeled as complex. To improve performance, a more consistent approach for text labeling should be implemented. Alternatively, the short word text entries could be removed prior to training as sentence complexity is likely not the primary issue.

	original_text	label
128762	A style of glassblowing	0
258773	A style of glassblowing	1
312209	A style of glassblowing	1

	original_text	label
2141	Biography from www.mikapohjola.com	1
105822	Biography from www.mikapohjola.com	0

	original_text	label
98514	.	1
24663	Grapefruit Juice	1
200498	500s	1
54774	Records .	1

Two examples of duplicate text entries with conflicting labels and an example of single/dual word entries classified as complex

Given the examples above, a number of other sentences might be ambiguously classified as well despite similar structure, word choice and semantics. In order to continue to improve our approach to sentence simplification it might be prudent to follow Andrew Ng's advice and be "more data centric and less model centric", meaning that data quality gains could outperform additional model optimization.(Andrew Ng)

Discussion and Conclusions:
There are also other directions that would be worth exploring in the future. While our team did attempt to make use of some deep learning architectures, our limited experience with setting up and applying these to NLP processing problems effectively limited the success of this result. Using an example from the Torch documentation (Pytorch) as a guide, a NN using torch text was fit, but was only able to achieve 68% accuracy. However, many of the most powerful NLP based models are currently deep learning based, so we believe that this could be an effective means to improve upon our model's accuracy. Another direction that would also be worth exploring in the future is whether this modeling approach could be evolved to include recommendations to simplify the sentence, instead of only classifying it. To that end, ongoing work at Stanford suggests using a RNN architecture and a paraphrase database that is able to take the input sentence and return a simplified sentence as the output. (Stanford RNN) This has the potential to not only flag a complex sentence to a user but also help them rectify the problem.

Looking towards the future, this work can help provide much needed editing to rapidly growing online works, however it does also come with some challenges. As discussed in the simplification section, the genre of the given articles may account for some of the variability in the perceived complexity of the text by the labelers, so a one-size fits all approach might be discriminatory towards certain areas such as math or science where longer and complex words are more commonplace and perhaps necessary to describe the topic. By encouraging these users to simplify their sentences, we could also contribute to an oversimplification of articles on complex tasks, resulting in a decrease in effectiveness of some Wikipedia reference articles. The idea of ethnic or cultural discrimination of the model might also be important to further evaluate, as the model might incorporate some bias towards a Western style of writing that might not be appropriate in all contexts, so caution should be utilized in its application. It might also be important to account for topic drift if this were to become a production model, so continued evaluation of performance would be necessary as language evolves over time. Further, as familiarity with certain topics increases in the general population this might impact the level of perceived complexity or Age of Acquisition of certain words. For the unsupervised portions, we should be somewhat cautious with the topic modeling, particularly if this is exposed to the users, as it has the potential to misclassify the topics which could be upsetting or confusing to users, particularly if the topics are sensitive in nature. There is also the potential for users to influence a topic model by flooding the open source system with certain words or phrases that could be inflammatory or target certain groups or cultures, so we should build in ways to monitor and stop the model if such an event arises.

While sentence complexity is a somewhat subjective problem, we feel that the above work has provided insight and a useful approach to solving the problem of sentence simplification for online article generation. To implement our approach, we would deploy a pipeline where sentences are first featurized and then transformed with the two unsupervised approaches. Next, both a logistic regression on the TfIdf features and a Random Forest on the resulting entire matrix would be used to make a classification prediction. Depending on the speed of this pipeline the featurization and unsupervised transformation steps could be tuned such that results are produced quickly. The end results would be delivered to the user in an underline type visualization that many users are familiar with from text processing software.

Appendix:

Examples from Select K-means Clusters from Word2Vec model.
Cluster_12 (mostly Simple text)

- Goldbachs conjecture
- Videos
- Other websites
- References
-

Cluster_7 (mostly Simple text)

- Plumeria clusioides
- Fitna
- Wombats
- Aubenas
-

Cluster_0 (Balanced between both classes)

- He and his horse May-Queen won the silver medal as member of the German team at the 1912 Summer Olympics . But he was 15th in the individual eventing competition .
- Winger is an American glam metal band formed in New York City that gained popularity during the late 1980s and early 1990s .
- In 1940 he was recalled for military service , but was discharged for medical reasons in 1942 .
- During the night of September 16 , all active watches and warnings were stopped .
-

Cluster_17 (Mostly Complex Text)

- Gonnehem is a commune in the Pas-de-Calais department in the Nord-Pas-de-Calais region of France .
- Gémenos is a commune east of Marseille in the Bouches-du-Rhône department in the Provence-Alpes-Côte d'Azur region in southern France .
- Gesnes-le-Gandelin is a commune in the Sarthe department in the region of Pays-de-la-Loire in northwestern France .
- Tilly-la-Campagne is a commune in the Calvados department in the Basse-Normandie region in northern France.

Sources:

1) https://en.wikipedia.org/wiki/Tf–idf (Accessed January 2022)
2) https://spacy.io/usage/linguistic-features (Accessed January 2022)
3) https://www.nltk.org/book/ch05.html (Accessed December 2021)
4) http://crr.ugent.be/archives/806 (Accessed January 2022)
5) http://crr.ugent.be/archives/1330 (Accessed January 2022)
6) https://en.wikipedia.org/wiki/Readability (Accessed December 2021)
7) http://api.dscovar.org/document/readability (Accessed January 2022)
8) https://scikit-learn.org/stable/modules/clustering.html#k-means (Accessed January 2022)
9) http://lsa.colorado.edu/papers/dp1.LSAintro.pdf (Accessed January 2022)
10) https://github.com/RaRe-Technologies/gensim-data (Accessed January 2022)
11) https://www.kaggle.com/rtatman/fine-tuning-word2vec/ (Accessed December 2021)
12) https://dylancastillo.co/nlp-snippets-cluster-documents-using-word2vec/ (Accessed January 2022)
13) https://shap.readthedocs.io/en/latest/example_notebooks/tabular_examples/linear_models/Sentiment%20Analysis%20with%20Logistic%20Regression.html (Accessed January 2022)
14) https://scikit-learn.org/stable/auto_examples/inspection/plot_permutation_importance.html (Accessed January 2022)
15) https://analyticsindiamag.com/big-data-to-good-data-andrew-ng-urges-ml-community-to-be-more-data-centric-and-less-model-centric/ (Accessed January 2022)

16) https://pytorch.org/tutorials/beginner/text_sentiment_ngrams_tutorial.html (Accessed December 2021)
17) https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1194/reports/custom/15842809.pdf (Accessed December 2021)
18) https://cocoxu.github.io/publications/tacl2016-smt-simplification.pdf (Accessed December 2021)

Deep Learning Sources:
19) https://github.com/scoutbee/pytorch-nlp-notebooks/blob/develop/3_rnn_text_classification.ipynb (Accessed December 2021)
20) https://blog.floydhub.com/gru-with-pytorch/ (Accessed December 2021)
21) https://www.youtube.com/watch?v=Apx_1erbQB4 (Accessed December 2021)

Statement of Work:

Phil: Explored the Tfidf clustering and unsupervised portion and contributed to POS tagging for the featurization. He looked at K-means clustering and efficient ways to find the optimal values for these methods as well as contributing unsupervised derived features for the supervised portions.

Daniel: Delivered many of the count based feature representations and named entity tagging, he also delivered the final scripts and portion on the random forest modeling and parameter optimization. He performed the random forest model error analysis and explored duplicate entries for the explanatory portions.

David: Built the external features (AOA etc.) into the final data set as well as delivered the final Tfidf supervised portion. He also contributed to the unsupervised portion of the report through the Word2Vec analysis and clustering. David also explored deep learning approaches to the project, however these were not performant enough to make the final report.

All: Contributed to the writing and editing of the final report.