Text Complexity with Wikipedia Data

By Laura Stagnaro and Sean Cafferty

SIADS 694/5 - The University of Michigan

TABLE OF CONTENTS

PART A: Supervised Learning

1. Supervised Learning Methods

1.1 - WORKFLOW OF SOURCE CODE

Our dataset consists of a collection of sentences (and sentence fragments) from Wikipedia articles. The objective of our project was to build a classifier to differentiate entries based on whether they are written in Simple English or Standard English. In practice, such a classifier might be used to identify Wikipedia articles that would benefit from linguistic simplification, thereby making them more readable for young readers and L2 English speakers. In order to approach this task, we began by creating a feature engine (**feature_extractor.py**) as well as a visualization engine (**viz_engine.py**) that would allow us to analyze potential feature representations. After establishing the viability of some features (**1_data_exploration.ipynb**), we proceeded to train models on the data in order to establish promising approaches to the classification task (**3_train_classification_models.ipynb** / **4_deep_learning**). After systematically analyzing the successes and failures of various models, feature representations, and evaluation methods, we identified our best performing model and feature combinations and established a pipeline that we believe is suitable for the challenge at hand.

1.2 - CHOSEN LEARNING METHODS

We chose to explore the viability of both deep learning and standard supervised machine learning techniques. We trained both **LSTM** and **CNN** models to classify the data as well as **logistic regression**, **random forest**, and **multinomial naive Bayes** models. Ultimately, the random forest and logistic regression models proved to be the most effective.

1.3 - DATA EXPLORATION AND FEATURE REPRESENTATIONS

In order to make decisions about feature representations, we created numerous potential candidates for both numeric and text representations to engineer into the dataset. We began by preprocessing the text and searching for ways in which the text data could be represented numerically (i.e. features that exist independently of semantic content). Such features allow us to embellish the data with more mathematical texture, thereby allowing for more effective classification.

The most intuitive option for numeric features is the **length** of sentences insofar as more complex sentences are likely to be relatively longer. After examining the data, we concluded that length was indeed a salient difference between the data in the Standard English and Simple English components of the corpus (*See Appendix - Fig. 1*) . In addition to computing the length of sentences, we also included counts of **punctuation**, **parts of speech,** as well as **named entities**. We also used the *Age of Acquisition* (AoA) and *Concreteness Rating* (CR) datasets provided by the Kaggle competition to create columns with both the count and ratio of **'advanced'** and '**hard**' words in a given sentence. Words were categorized as 'advanced' if less than 92% of people in the "concreteness" dataset knew the word, or if the word was acquired after the age of 8 according to the AoA dataset. Meanwhile, **'hard'** words included any words

that were *not* represented in either the CR dataset or the Dale-Chall[1] list of words, as well as words acquired after the age of 12 according to the AoA dataset. We used both the **'hard'** and **'advanced'** data representations to allow for multiple gradations of vocabulary difficulty to be incorporated into the feature representations. Although these features may contribute to the classifier's ability to distinguish class membership, they could also simply be a function of sentence length. As such, it became necessary to engineer features that weren't necessarily associated with the length of a given entry.

Unlike the features that include raw counts, **average word length** is less contingent on the length of the sentence. Given the likelihood that more difficult vocabulary contains more characters on average, we computed the average length of tokens within each entry. Moreover, though part-of-speech counts may provide some information about class membership, these counts might also be a function of length *(See Appendix - Fig. 2)*. As such, we decided it would be more effective to measure each part of speech (according to the nltk tag schema) as a proportion of the number of words in a given entry *(See Appendix - Fig. 3)*.

In addition to creating numeric features from the text, we also sought to engineer features that recognized semantic distinctions. Though features such as average word length and punctuation counts are useful, such measures ignore the actual content of the text. As such, we began to explore possible ways in which to more thoroughly incorporate semantic meaning into the data. Before considering precisely which text representation technique to choose, we first undertook an examination of the semantic content of the data in order to determine whether such text features would meaningfully enrich our analysis.

Simple word clouds give a cursory but effective snapshot of the vocabulary within a given corpora. At first glance, it appears that the Simple Wikipedia and Standard Wikipedia do not differ significantly in terms of vocabulary *(See Appendix - Fig. 4)*. More refined techniques were thus needed to understand the semantic differences across the Simple and Standard English sections of the corpus.

In order to examine the text data with more granularity, we created separate word embeddings for both the Standard and Simple English entries in order to gain a better understanding of the content. In order to better visualize the differing relationships between words in the text data, we decided to create matrices of cosine similarities between vocabulary found within the corpus. The use of colored matrix representations of the relationship between vocabulary is perhaps a useful way to quickly compare the coherence of semantic information within two datasets. While inputting common vocabulary, the datasets do not appear to be particularly different *(See Appendix - Fig. 5)***.** If we input words that are overrepresented or underrepresented in either class of the corpus, however, the semantic differences between the two classes of the dataset become much more pronounced *(See Appendix - Fig 6)***.** As such, in addition to generating numeric features for our dataset, it seemed reasonable to engineer text features as well. In order to take advantage of the differing semantic texture of the two classes of data, we used both TFIDF vectors and pre-trained GloVE word embeddings to represent the text in our pipelines.

[1] "Dale-Chall list of ~3000 elementary English words that are typically familiar to 80% of American 4th grade students"

1.4 - CHOOSING PARAMETERS

Prior to tuning any of the basic classification models, we first ran all the models with different feature representations using the baseline hyperparameter settings. We then picked several of the best initial models for tuning to see how much we could improve our accuracy. In addition to looking at the accuracy of the test data, we also looked at the accuracy of the training data (**'score'**) to determine whether the model was overfitting. We also referred to metrics like precision, recall, and AUC in order to assess how changes to the model yielded differing results. Given that accuracy was the target metric for this classification task, we ultimately sought to obtain the best accuracy possible.

In order to tune parameters, we initially attempted to use scikit's GridSearchCV and RandomizedSearchCV. Unfortunately, both of these methods took a considerable amount of time and processing power, especially when tuning a model such as random forest. Thus, instead of relying on these built-in grid search methods, we decided to use a more manual approach. This manual approach included looking at the initial score and accuracy to assess if the model was overfitting or underfitting, and then individually tuning the hyperparameters that would best address the model's performance problems. For example, when tuning the random forest model, we chose to focus on *n_estimators* and *max_depth*—the former modulating the number of decision trees and the latter determining the maximum depth of each tree. Both hyperparameters are useful in preventing overfitting and improving model accuracy. We thus tuned each hyperparameter separately by determining a range of several values to try, retraining our model using each value, and then evaluating the metrics. We then chose the value that returned the best accuracy before tuning the next hyperparameter. Similarly, when tuning the logistic regression model we focused on the regularization hyperparameters *'penalty'* and *'C'*.

1.5 - CHALLENGES

As we began testing our models, we were struck by the seeming inefficacy of our deep learning models. For most of our DL approaches, we encoded the text data into Stanford's GLoVe word embeddings and also created a dual pipeline to include numeric features. Despite our best efforts to tune hyperparameters to find that perfect balance, the best accuracy score our models managed to achieve was slightly above 70 percent—and only after several hours of training (**4_deep_learning.ipynb**). Though more patience and computing power might have allowed us to grid-search our way to an ideal DL model, this experience was an excellent lesson in the shortcomings of deep learning. Namely, binary text classification tasks with extremely noisy data are unlikely to yield promising deep learning models. We thus chose to invest our efforts into generating features that add numeric texture to the data set, thereby rendering the data more digestible for conventional ML algorithms.

Though we eventually settled on standard machine learning techniques, the challenge of tuning hyperparameters persisted. As such, the search for more effective feature representations was also complicated by the need for consistency in hyperparameters. In order to determine whether better performance was the result of modified features or differing hyperparameters, we were obliged to approach the problem in a manner that was slow, but systematic, in order to make causal assertions about our results. The choice of features and models thus turned out to be a complex task that required a predetermined standard of procedure in order to avoid becoming mired in conflicting results.

2. Supervised Learning Evaluation

2.1 - MODEL EVALUATION

After experimenting with numerous combinations of models and feature representations, we compared baseline performance and settled on three potential candidates for our final model: 1) a random forest model with the engineered features, 2) a logistic regression model with the engineered features and TFIDF text representations, and 3) a random forest model with the engineered features and GloVe word embeddings.

The random forest model with the 45 engineered features achieved a score of 0.989 and an accuracy of 0.712. We were able to evaluate the importance of each feature by using the *feature_importances_* attribute available for sklearn's random forest model. This attribute measures importance based on which features are associated with higher average decreases of impurity across the nodes in all the trees. After ranking the features by importance, we then retrained the model on subsets of the most important features in increments of five (e.g. 5 most important features, 10 most important features, etc.). Ultimately, the model's accuracy performed best after being trained on the 30 most important features, achieving a score of 0.714. Though this improvement in performance was minimal, it enabled us to remove redundant or ineffective features, thereby increasing the efficiency of model training. After adjusting the hyperparameters (namely, *max_depth* and *n_estimators*), the accuracy increased to 0.719.

Although we were attempting to optimize our model's accuracy score, we also assessed metrics such as AUC score, precision, and recall. While training our first random forest model, we achieved slight improvements across these different metrics. Even so, such improvements were minuscule and suggest that we should not rely exclusively on feature reduction and hyperparameter tuning to optimize our model.

We also trained a second random forest model, which in addition to the engineered features included word embedding representations of the original text. For this model our baseline was 95 features, and we achieved a score of 0.992 and an accuracy of 0.716. This model performed best while using only the top 20 features, achieving a score of 0.989 and accuracy of 0.717. Again, this feature selection process did not lead to a huge increase in accuracy, but it did slightly decrease how much the model was overfitting and led to a more efficient training process. After conducting hyperparameter tuning the accuracy increased to **0.722**, the highest accuracy we attained with any model. There was also a slight increase in AUC and recall, but also a very slight decrease in precision. Though the slight improvements were encouraging, feature reduction and hyperparameter tuning again proved to be an only mildly effective means to improve model performance.

Our third and final candidate model was the logistic regression model with TFIDF text representations and the engineered features. We began with over 8,000 features, and achieved a score of 0.720 and an accuracy of 0.700. As with random forest models, we first tried to optimize our model by assessing which features were most important. Initially, we wanted to use traditional dimensionality reduction techniques such as SVD or PCA, but found these methods to be computationally inefficient given the number of dimensions we would have to keep in order to retain a large majority of the variance in the data. For

example, when reducing the data to 100 features, we only retained about 26% of the variance. This reduction led to a significant decrease in performance, and when we attempted to reduce the features to around 1000 we ran into computational problems. We also thought about reducing the features manually, as we did for the random forest models, but this seemed inefficient given the number of features.

Fortunately, logistic regression actually has a built-in hyperparameter called '*penalty*' which when set to L1 can reduce the number of features in a model. An L1 penalty, or lasso regularization, pushes the coefficients of less important features to zero, essentially removing them from the model. We found that the L1 penalty reduced the number of features to around 6,100 which did produce a slight increase in accuracy. We then focused on tuning the *C* hyperparameter that controls regularization and can be used to prevent underfitting or overfitting. Given that our model appeared to be underfitting since the accuracy on the training data was very similar to the test data, we tried to increase the model's complexity by increasing the value of *C*. In doing so, we did manage to decrease underfitting, though at the expense of accuracy. Perhaps the complexity of our data prevents the *C* hyperparameter from having a significant effect on performance. After tuning the model, we ended up with a score of 0.719 and an accuracy of 0.701, a very minimal performance increase. As for other metrics, a small increase in AUC and precision was paired with a decrease in recall. As with the random forest models, hyperparameter tuning did not lead to significant performance gains.

In comparing all of the models and the feature representations, it is clear that the random forest models outperformed the logistic regression and multinomial naive Bayes models, regardless of feature representations. In general, random forest models tend to outperform these other models on classification tasks because they essentially train multiple different decision trees, all of which make their own predictions based on how the majority of the decision trees classify a given datapoint. This ensemble method allows for more error tolerance in that some decision trees may wrongly classify a datapoint, yet the model itself may still arrive at the correct conclusion so long as the majority of decision trees choose the proper category. This error tolerance does come with some downsides, such as computational cost, since hundreds or even thousands of individual decision trees might need to be trained. Also, as we noted in our evaluation, random forest models tend to overfit the training data. Unlike random forests, models such as logistic regression and multinomial naive Bayes have 'one shot' to make a correct prediction. Even so, such models are computationally more efficient and tend not to overfit as much as random forests. They do, however, also have their downsides because they may make certain assumptions about the data which could lead to relatively worse performance. Since logistic regression is a linear model, one of its primary assumptions is that the classes are linearly separable. Based on our results, it is clear that our data is not linearly separable given the difficulty of achieving results above 0.70. As such, models that are better suited to linearly inseparable data such as Kernelized SVM might have performed well with our data. Though we attempted to work with SVM models early on in the project, the amount of time needed to train these models was prohibitively burdensome. Given such constraints, the random forest approach appears to be the best option, shortcomings notwithstanding.

MODEL / FEATURE REPRESENTATIONS	***Accuracy Score***
Random Forest with GloVe Embeddings and Numeric Features	**0.722**
Logistic Regression with TFIDF and Numeric Features	0.720
Random Forest with Numeric Feature	0.719
Multinomial Naive Bayes with Numeric Features	0.619

As for feature representations, the engineered numeric features combined with the word embeddings performed the best. However, given that the addition of word embeddings led to only a 0.003 improvement in the accuracy score, it is clear that semantic content is not the primary driver of class membership. This finding is logical in that word embeddings are extremely useful for differentiating topic classes in large data sets, though not necessarily effective while differentiating classes based on text complexity. Indeed, our analysis of feature importances indicated that our models depended largely on non-semantic features such as sentence length, average word count, and the punctuation ratio.

Designing prediction models inevitably involves tradeoffs. During our project, we were obliged to choose between model performance and computational efficiency. Although models like Kernelized SVMs and Gradient Boosted Trees may have led to better performance, both of these models were much too time-consuming and computationally costly. Moreover, if we had had more time and computer power, we would have been able to take advantage of GridSearchCV while tuning our hyperparameters.

The sensitivity of our results to various model specifications was noticeable, but we did not witness any significant swings in accuracy scores based on either parameters or feature representations as long as several of the most important features were included. During our preliminary analysis, we found a few data representations that led to poor performance. Moreover, when the number of features is reduced to the top five ‘most important’ features, there was a large drop in accuracy. And though the performance improved with the inclusion of more features, the diminishing returns on the addition of more features quickly became noticeable at around 10 features. This pattern of model sensitivity suggests that there is a threshold at which additional features do not lead to significant changes in performance. Meanwhile, the tuning of hyperparameters does not seem to have a significant impact on the models’ performance so long as such parameters remain within a reasonable range. This suggests that feature representation is much more important than hyperparameter tuning and leads us to believe there are probably other features which would have improved the performance of our model.

2.2 - FAILURE ANALYSIS

As discussed, we had high hopes for deep learning models, but our initial failures to optimize such a pipeline prompted us to abandon the deep learning paradigm altogether. Though ultimately unsuccessful, our misadventures in deep learning were quite informative about the potential situations in which such models would be the preferred option.

In addition to abandoning deep learning, we were disappointed by the results of TFIDF features. We trained a multinomial naive Bayes, random forest and logistic regression model with just TFIDF features and only achieved around 0.60 for accuracy. The reason this failed is because TFIDF is a way of weighting the actual words within the sentence. Words that are more important in the sentence will be weighted higher than words which are more frequent throughout the entire corpus of words. Had we been trying to classify sentences based on the actual content, this may have been a good approach. As we discussed earlier though, the distinction between the two classes actually has much more to do with the structure of the sentences than the content. For this reason, using TFIDF as the only feature turned out to be a failure.

Moreover, we found that using pre-processed text for the TFIDF representations actually performed slightly worse than using the unprocessed data to make text representations. Typical NLP techniques such as lemmatization and stop word removal are thought to reduce 'noise' and improve model performance. In this case, however, we actually found the opposite to be true because the noise may actually be part of the signal. Including stopwords or differing verb tenses is perhaps quite important for distinguishing Simple from Standard English text. Thus, the marginal improvements in the model attributable to TFIDF representations and word embeddings might result from counterintuitive approaches to preprocessing data—namely, retaining verb morphology and stop words. Though we experienced some setbacks during our analysis, these setbacks significantly improved our understanding of the data and will hopefully inform our practices in the future.

Part B. Unsupervised Learning

For the unsupervised-learning portion of our project, we decided to use the same dataset from the supervised-learning portion in order to deepen our understanding of topic modelling techniques such as Latent Semantic Analysis and Latent Dirichlet Allocation. The objective was to find coherent topics within the text data in order to better infer its provenance. We used the text from the WikiLarge_Train.csv file of 400,000+ Wikipedia sentences in order to conduct our topic modeling. Although we used the same dataset as the supervised-learning portion of the project, we chose to focus on the global structure of the dataset as opposed to classification tasks. The distinction between Simple and Standard English thus became irrelevant.

1. Unsupervised Learning Methods

We pre-processed the data in order to render it suitable for input into our models. For both the LSA and LDA approaches, it was necessary to use document-term matrices in order to generate such features. We first conducted a latent semantic analysis, and though our results yielded some interesting patterns, human observers would likely not consider these topics particularly coherent. Nonetheless, one topic appeared to contain a large number of words pertaining to France, while another topic seemed to largely pertain to both America and sports *(See topic_modeling.ipynb)*.

Given that the results of the LSA were not particularly robust, we sought to analyze the results of Latent Dirichlet Allocation techniques as well. Again, it appeared that most topics were incoherent, all except

one pertaining to France and another (albeit more vague) that seemed to pertain to 'American' themes (*See Appendix - Fig. 7*)**.** In order to achieve the most mathematically coherent number of topics based on measures of perplexity and log likelihood, we decided to conduct a grid search of LDA parameters in order to find the optimal number of clusters. Ultimately, the grid search yielded only two topics that were deemed mathematically coherent. Though few topics were present, it seems that both topics are rather logical. Again, one topic refers to France while the other topic somewhat refers to what might be considered 'American Sports' themes *(See topic_modeling.ipynb)*.

The Final Two - Grid Search Results from LDA

	Word 0	Word 1	Word 2	Word 3	Word 4	Word 5	Word 6	Word 7	Word 8	Word 9
Topic 0	rrb	lrb	bear	state	unit	footbal	play	american	nation	year
Topic 1	use	franc	commun	depart	region	make	north	peopl	citi	south

In addition to the LSA and LDA, we also decided to explore the data using Word2Vec, which might be considered a semi-supervised learning technique. We used gensim's Word2Vec to create custom word embeddings from text, and then used Networkx and Plotly to generate a network of common interrelated vocabulary within the data. Our final product was an interactive network graph that allowed us to manually explore the relationships between words. **(***See Figures 8 & 9***)** This data exploration technique revealed that there are indeed clusters of common vocabulary related to the topics identified by the unsupervised learning techniques. Thus, although few robust topics are present in the data, there does seem to be a semantic skew towards France- and America-related topics, leading us to conclude that articles regarding these topics are rather overrepresented in the dataset.

2. Unsupervised Evaluation

As mentioned above, it proved to be quite challenging to cluster human-coherent topics within the data set. This is perhaps due to limitations in computational capacities in that the ability to grid search through higher numbers of topics (say, 200) may have resulted in an elegantly logical set of topics. Moreover, such challenges in topic modeling might result from the high levels of noise within the data itself. Even so, both the LDA topic modeling and the visual exploration of the dataset revealed a notable geographic component to the text data. Indeed, the LDA analysis varyingly revealed categories pertaining to 'France' and 'America', and sometimes 'England'. Moreover, the word cloud analysis revealed a notable overrepresentation of geographic terms *(See Appendix - Fig. 10)*.

After we had conducted the LSA and LDA, we were questioning the extent to which this pattern of geographic terms was the residue of the English language as it appears in Wikipedia (e.g. decidedly Euro-centric), or if the Franco-American themes were simply a feature of this subset of Wikipedia data. This motivated our decision to use the interactive network graph of common words. The manual analysis of the word embedding network graph revealed several identifiable clusters of semantic information buried within the text dataset. Most notably, we found that there was a cluster of names of French localities. Their appearance in the graph suggests that these terms are well-represented in the text. Moreover, upon closer examination of the word cloud, we noticed that the words 'region', 'department',

and ‘commune’ were thus likely not referring to their English meanings, but rather their French meanings. That is, France’s 13 *régions* are subdivided into 96 *départments* which are eventually subdivided into one of 36,529 *communes*. We certainly would not have noticed this relationship had we not undertaken a multi-pronged approach to modeling topics in this noisy data. Accordingly, it seems reasonable to surmise that in addition to containing texts regarding numerous regions of the world (especially the English-speaking world), much of the text in our datasets was retrieved from Wikipedia articles regarding France. The most coherent topic to be found in the dataset both mathematically and subjectively thus seems to be France-related. *(See Appendix - Fig. 11)*

Discussion

1. Takeaways

One of our biggest takeaways from the supervised-learning portion of this project was the extent to which models and feature representations allow for differing solutions to the same problem. Moreover, although we were quite eager to use deep learning models, we learned through trial and error that such models are not nearly as effective with noisy data wherein the semantic content itself is of limited importance. In addition, we were impressed by the performance of the humble random forest model, as well as other models that might be deemed more pedestrian. This project certainly instilled us with a greater appreciation for the strengths and weaknesses of various models.

If we had had more time to experiment with feature engineering, we would surely have added a collection of additional numeric features to the data—such as counts of the dependencies and word shape, both of which can be produced with the SpaCy library. There are surely numerous possible numeric features that would further enrich the dataset with numeric texture. Moreover, apropos text features, it would be interesting to create TFIDF representations of n-grams that may provide information about the structural difficulty of the text. In addition, though we did make functions to identify overrepresented words in either class, it proved somewhat complicated to incorporate these features into the pipeline within the given timeframe. In terms of models, it would have been interesting to determine if a character-level approach (as opposed to a word-level approach) would have enhanced our deep learning models. Finally, if we had been able to harness more computing power, perhaps we would have been able to better utilize grid search techniques in order to find a more optimal solution.

During the unsupervised-learning portion of the project, we were struck by the extent to which human language could be wrangled by mathematics. As neither of us have backgrounds in computational linguistics, the depth and possibilities of the field were quite inspiring. Moreover, we were surprised that such robust word embeddings could be trained on a relatively small amount of data. Both the topic modeling techniques and the Word2Vec network thus greatly deepened our appreciation for the power of distributional semantics and helped us better grasp the role of NLP in fields like AI.

If we had more time and resources at our disposal, we certainly would have mined the dataset for more topics and patterns. All of our techniques were computationally costly and would perhaps have yielded more interesting results if we had more computing power. As mentioned, we would have liked to conduct a grid search of the topic models to allow for a greater number of potential topics. Perhaps then we would have been able to find a much tidier set of topics than we ultimately achieved. Moreover, it would have been quite fascinating to create a word embedding network graph with a complete vocabulary. The sheer number of edges and nodes required, however, meant that our visualization did not scale well. We were thus obliged to limit our network graphs to the 1000 most common words—anything significantly larger would not function appropriately. If we had been able to comfortably dart around a rendering of a much larger network, we may have been able to gain a more nuanced understanding of some patterns in our data. Instead, we tended to focus on the hyper-salient patterns—namely, the French geographic terms.

2. Ethics in Data Science

The experience of a supervised-learning competition underscored the extent to which the workflow of data scientists creates ample opportunity for ethical lapses. When data scientists become singularly focused on achieving commendable accuracy or precision scores, they might find themselves sacrificing best practices in the pursuit of 'good numbers'. In the case of our Kaggle competition, a few instances of data leakage led us to become momentarily overconfident in our models. We were so enamoured with our 'good numbers' that we were more susceptible to making improper inferences about our data. If we had not audited our code or reflected on our models' performance, our predictions would have ultimately been quite distorted. Luckily, we became aware of our mistakes early on in the project. Though the stakes in a Kaggle competition are quite low, real-world algorithms can have a significant impact on people's lives. There are surely many instances in which data scientists' attachment to 'good numbers' has led to the deployment of faulty models—and in industries such as finance and healthcare, faulty models might result in costly mistakes. This project thus helped us internalize the extent to which the pursuit of 'good numbers' must be undertaken within a framework of best practices as well as an awareness of potential societal costs.

As for the unsupervised-learning portion of our project, we noted the extent to which observer bias is capable of undermining an effective analysis. Namely, the fact that we initially misidentified French words as their English cognates exemplified the degree to which observers are steeped in cultural expectations. Though confusing 'department' with '*département'* is a mundane example of observer bias, there is ample potential for such biases to have a deleterious impact in the real world. For instance, individuals of differing cultural backgrounds may communicate symptoms differently when describing ailments to, say, an NLP-based medical service. If the engineers behind such applications fail to account for linguistic variability, patients that differ from the 'data mainstream' might experience worse health outcomes. In order to address such observer biases, data science practitioners should continuously improve their cultural literacy alongside their technical skills. Indeed, because the practice of data science is inherently embedded in social, economic, and political realities, data scientists should encourage each other to approach their work with a high degree of reflexivity.

Statement of Work

- Laura focused on optimizing the models and organizing the pipeline for the Kaggle competition.
- Sean worked on data exploration visualizations and managed the version control.
- Both Laura and Sean engineered feature representations and designed viable models for further testing. Overall, this project was very collaborative.

APPENDIX

Figure 1 (back to section)

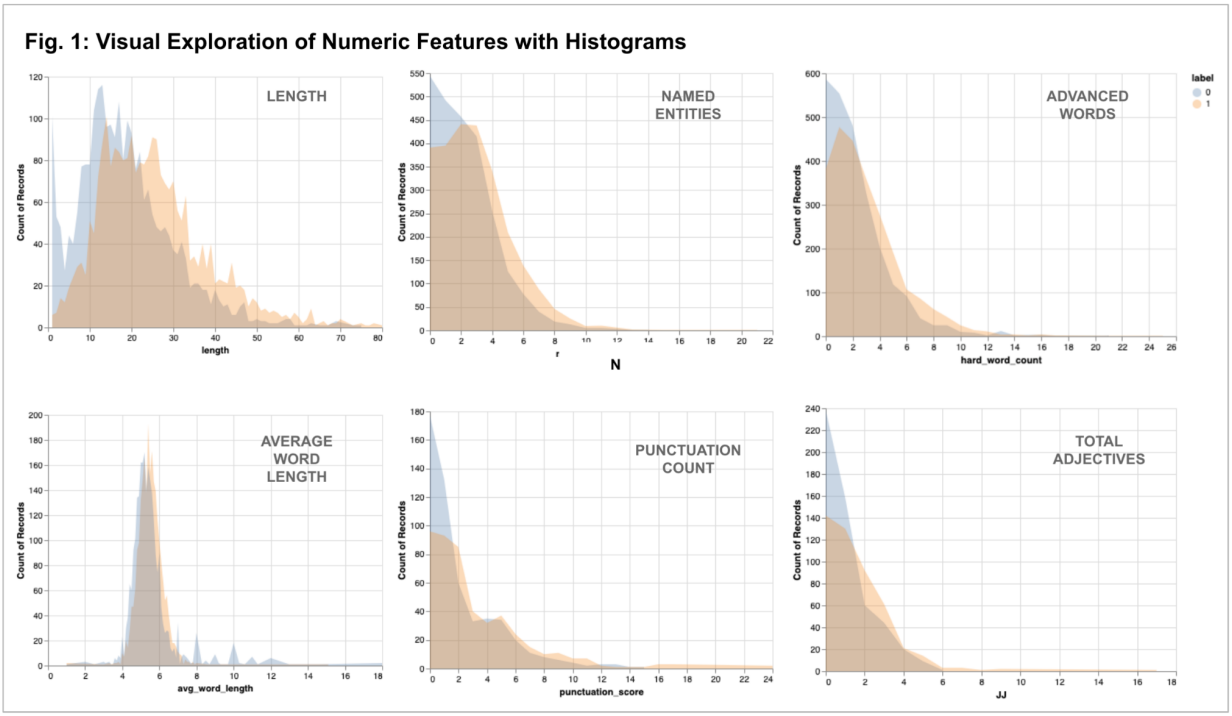

Figure 2 (back to section)

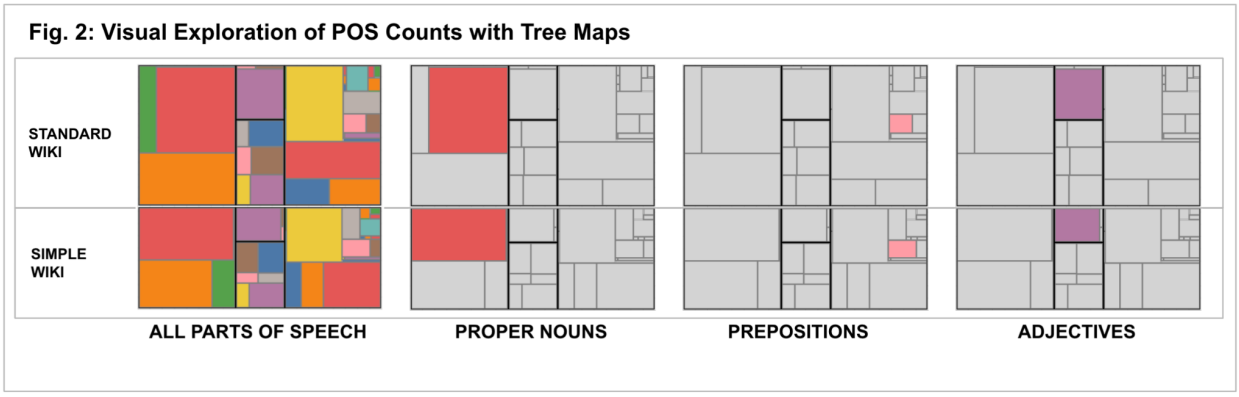

Figure 3 (back to section)

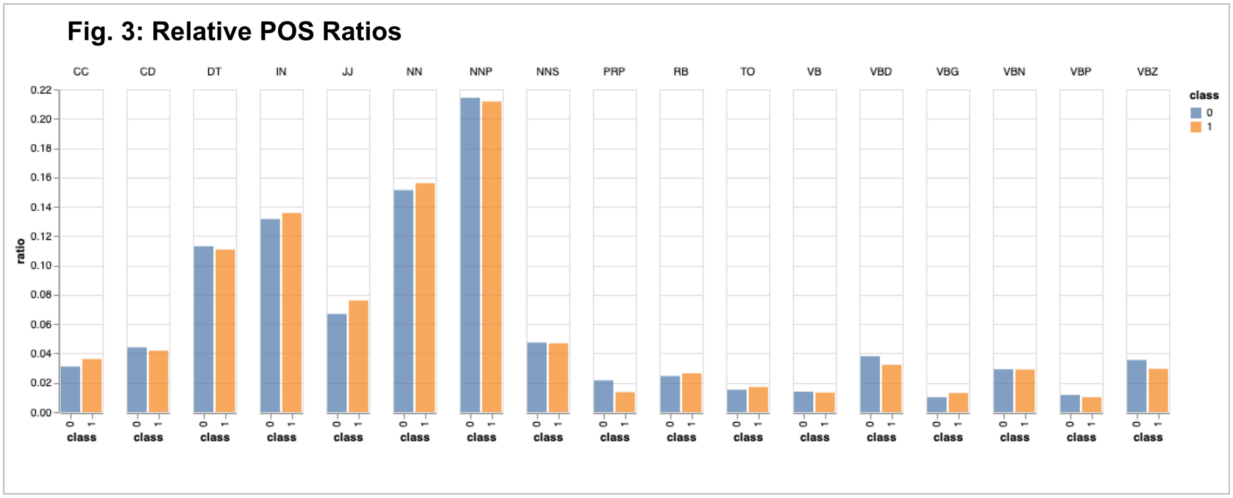

Figure 4 (back to section)

Figure 5 (back to section)

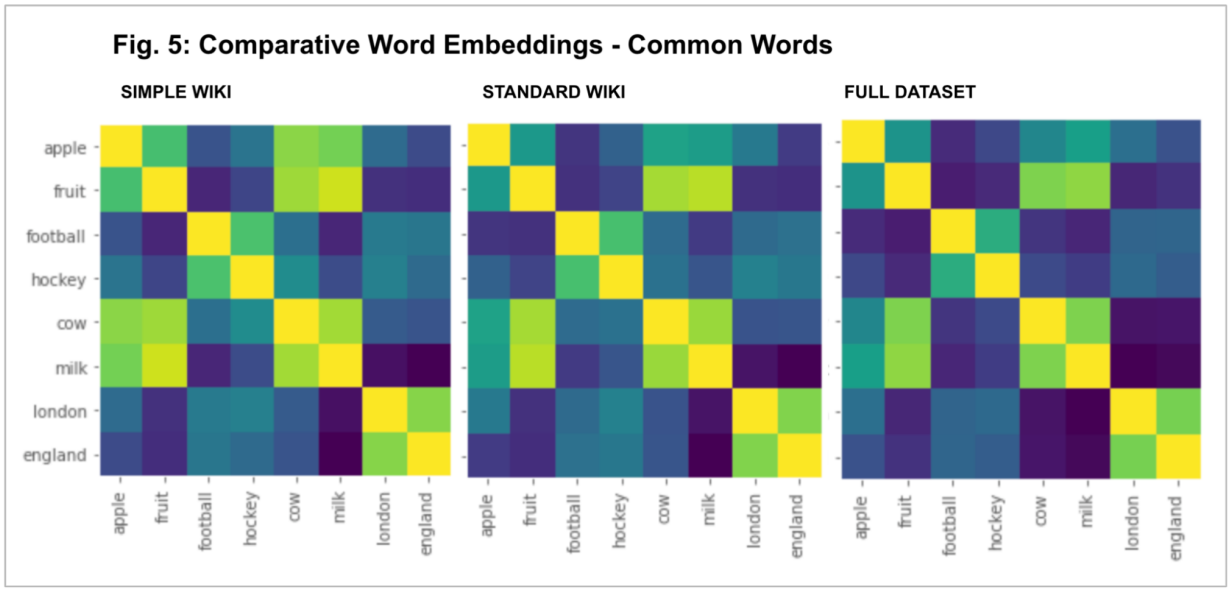

Figure 6 (back to section)

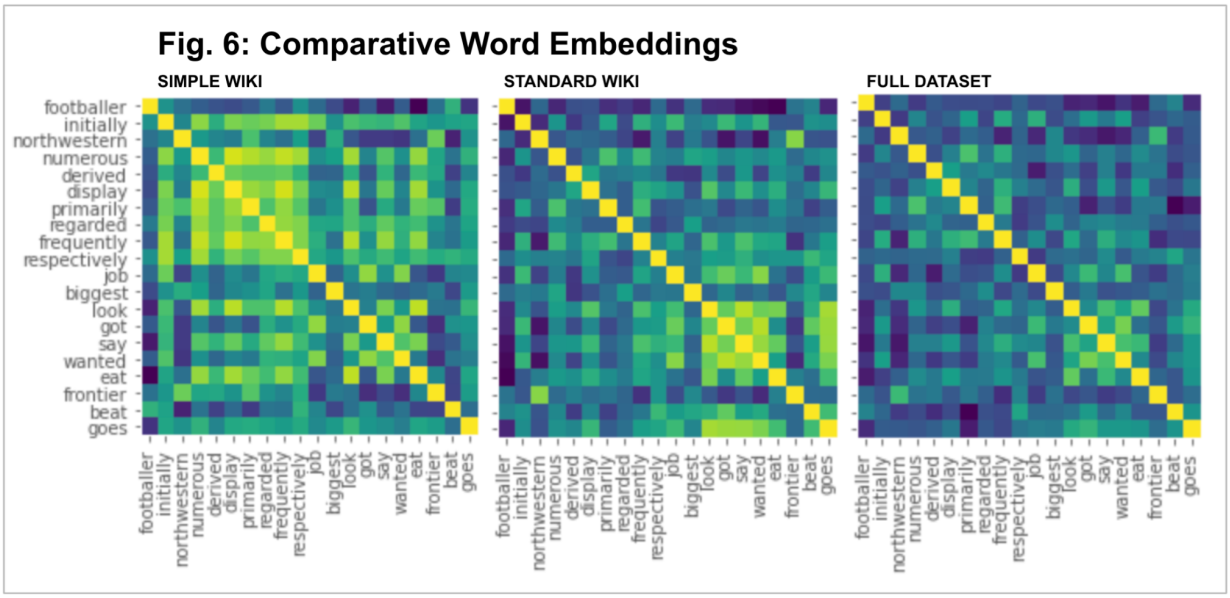

Figure 7 (back to section)

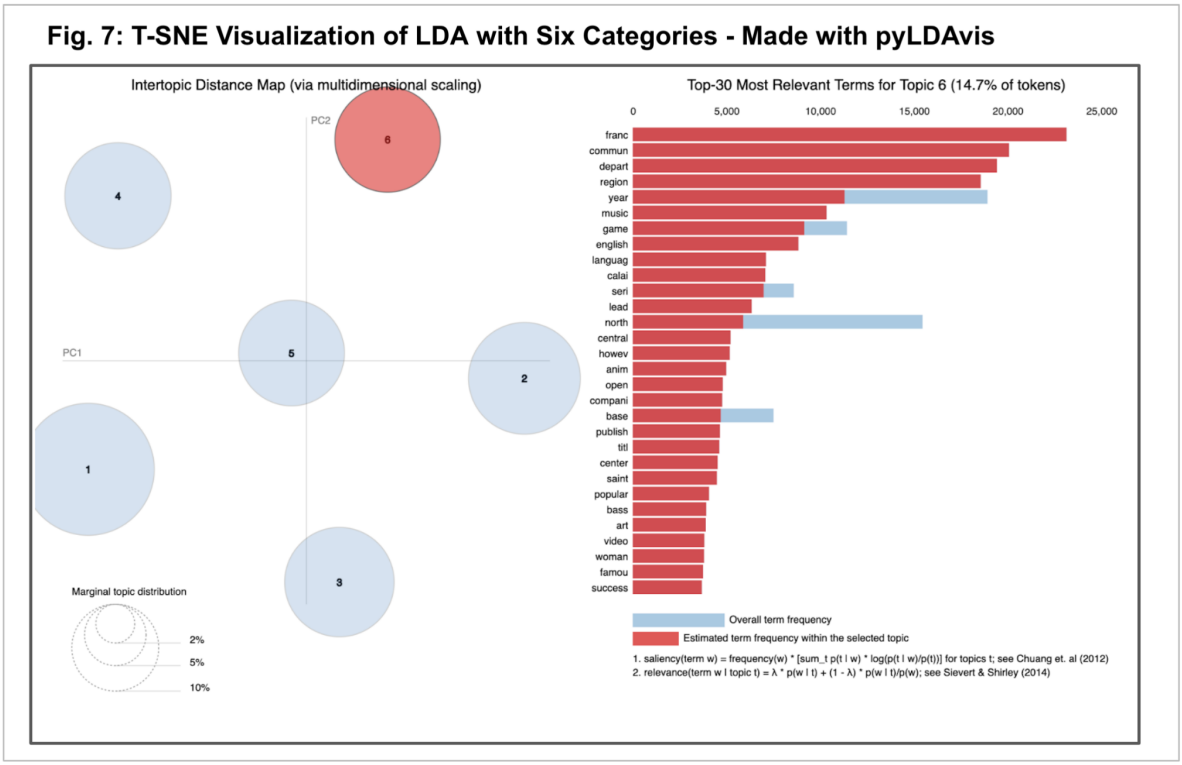

Figure 8 (back to section)

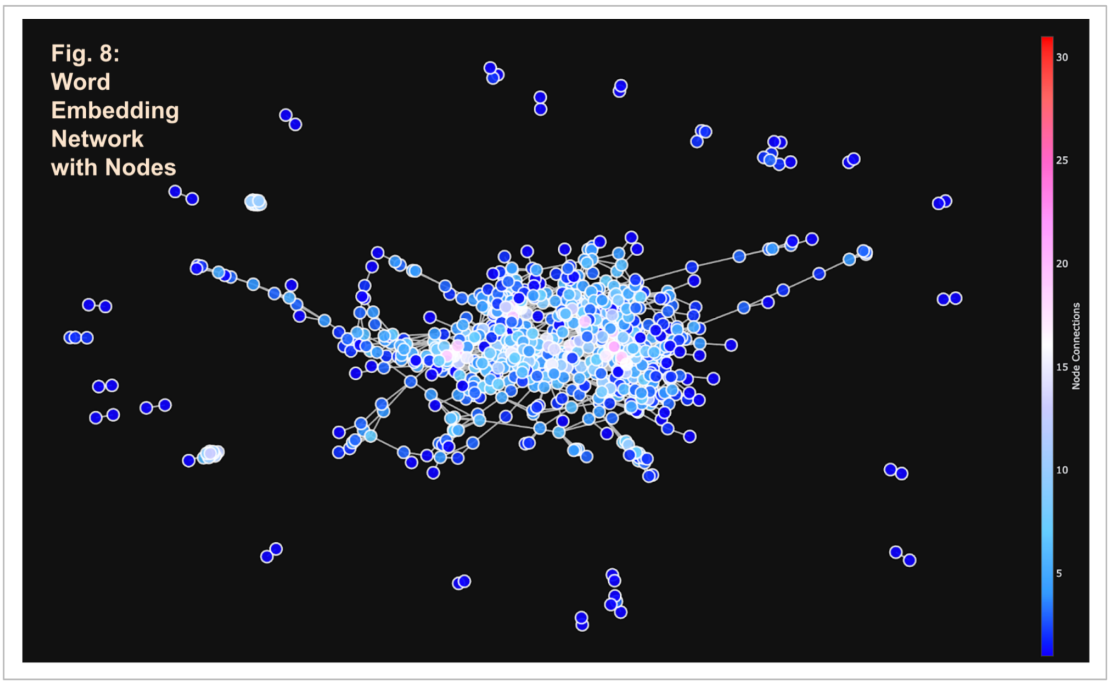

Figure 9 (back to section)

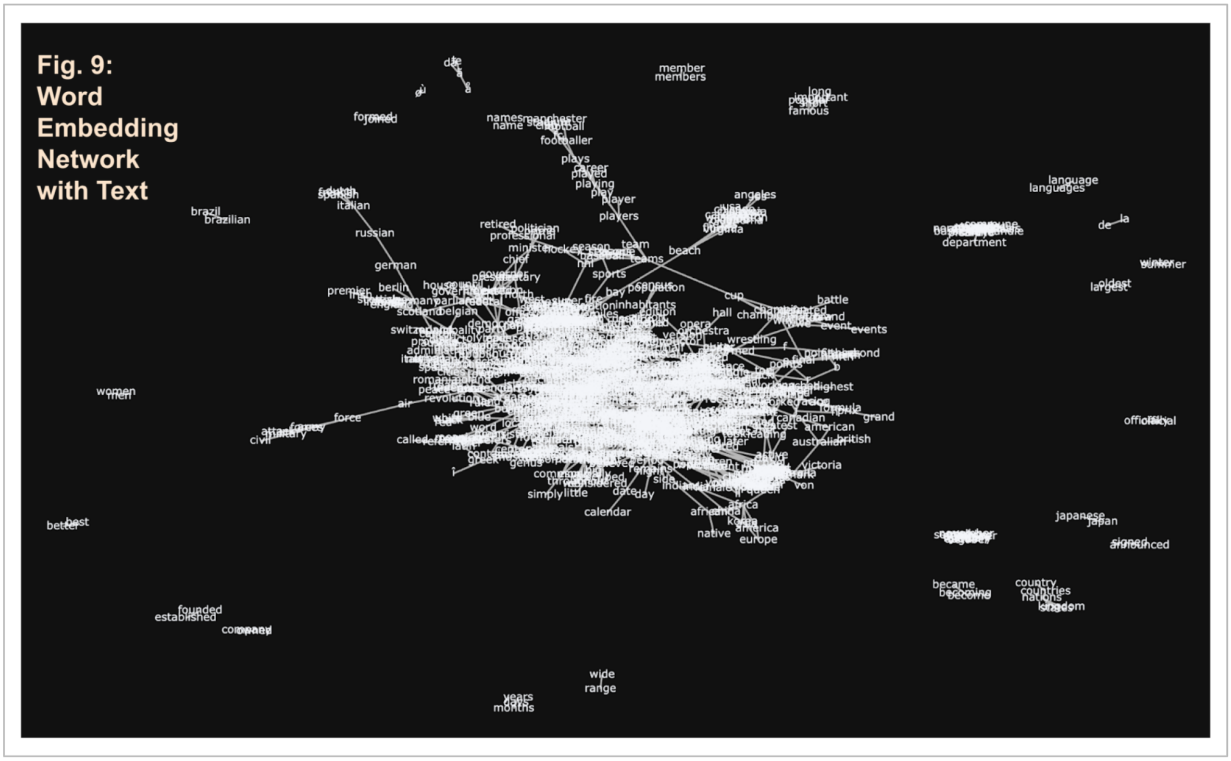

Figure 10 (back to section)

Fig. 10: Simple Word Cloud

Figure 11 (back to section)