Final Iteration Implementation Summary

Beau Carlborg, Clara Richter, Sunny Chiu,

First steps: Refining our data processing pipeline

During this iteration, almost right out of the gates, we were able to get simple baseline models working using a lexical summary statistics. For a given text, these statistics represent simple metrics such as the average word length, sentence length, and the simple vocabulary richness of the work itself. While we were able to get our model working using these statistics, we desperately wanted to be able to use the n-grams and part of speech ngrams to make our model better.

Eliminating Columns from the data:

In order for us to effectively use the n grams and part of speech ngrams, we needed to perform significant data reduction in order to reduce the size of our matrix. Without adding any form of reduction, generating up to tri-grams for our data on multiple authors and multiple books generated matrices which had nearly a million columns. In order to have a more manageable data set, we needed to create a method which would allow us to reduce the number of columns in our data set. Sunny began the initial steps of this project by creating more effective means for our group to process the large data sets. Using Sunny's initial code, Clara and Beau were able to continue on to use create two functions which could be used to remove columns of insignificant data. Beau constructed the first of these functions is used to eliminate columns based on sparsity of data. This function evaluates each column of data, and ensures that the column meets some minimum threshold for the percent of rows which contain data. We chose a 50% threshold for our data columns. Ensuring that for a single feature, every author has that feature present in at least half of their works. Clara constructed the second of these two functions which eliminated columns which did not display a significant difference in their means or had too large of standard deviations. Our hopes in constructing this function was that our columns would represent works which had larger difference in the metrics they captured.

Creating our Training and Test Data

After creating our preprocessing techniques for the data. A crucial step in our process was to actually compute the processed data. Because of the large scale of our data, this posed a very significant challenge for our group. In order to effectively preprocess, we needed to refactor many elements of our code to use more memory efficient techniques to compute the data. Every group member contributed immensely to this part of the project. In order to effectively run our program, we each needed gain a much more in depth understanding of the ways in which various data structures operate in terms of their memory efficiency. This became essential when we were processing such large amounts of data. Small improvements in our code efficiency relative to reading and writing our data would provide enormous practical improvements in the practical runtime of our project. For example, at one point, choosing to use a python dictionary as opposed to pandas series allowed us to cut the runtime of our algorithm by a factor of the number of books we provided as input to our function.

Running the Model

After we created the datasets, we were able to run our model and evaluate the results as we tweaked various factors. We began by implementing a simple linear classifiers that we could use to evaluate if our model was “beating the monkey.” To our surprise. Our model performed significantly better than the monkey simply using lexical features. Upon utilizing the ngrams and parts of speech grams, we were able to see even more significant improvements in our data by incorporating the ngrams and parts of speech grams. Initially, for two authors, we observed success rates as high as 90%. We continued testing our model using various combinations of number of authors, number of grams, and testing mechanisms.

We were able to test our data by running a logistic regression and a support vector machine. We chose to perform k-fold testing for our data set in order to sure that there was an appropriate amount of test code. Sunny constructed the models while Beau ran the various data processing programs in order to generate the data tables required for the analysis. Once the processing was completed. We created various graphs to display the differences between the logistic regression and the support vector machine.

Running our code

The entirety of our data preprocessing library can be found in the second iteration folder within our github repository. In this folder three jupyter notebooks represent the entirety of our data preprocessing pipeline.

1. **data_retrieval.ipynb** -- Allows us to obtain and consolidate available authors works.
2. **feature_extraction.ipynb** -- This file allows us to obtain the various features from the data such as lexical summary statistics like average word length, as well as n-grams and parts of speech grams.
3. **feature_reduction.ipynb** -- This file allows us to limit the column size of our matrix which we use for training.
4. **Modeling.ipynb** -- this file contains our test and trial runs as well as our visualizations for the effectiveness of our model.