{18} - Final Report

Authors: Lim Yu Rong, Samuel, Goh Shau Cher Shaun , Spencer Tan Wen Hao
Emails: e0310716@u.nus.edu, e0175574@u.nus.edu, e0148700@u.nus.edu

Abstract

Stylometry is a crucial aspect of book recommendation systems. In this project, we explore various models that are capable of extracting an author's writing style from a text document. We aim to compare style similarity between Victorian-era authors in order to investigate three different approaches: document-level embedding with Doc2vec, Feature Engineering using Logistic Regression, and function word frequency k-NN, as well as ensemble approaches combining them. When testing shorter excerpts of text we found that models that mostly work on localised vocabulary (Naive Bayes) and text characteristics (Logistic Regression) were not as effective as more holistic document-level embeddings (Doc2vec). However, with longer test excerpts, most models were able to identify the correct author amongst the small pool of authors.

Introduction

Since most recommendation systems in current use employ metadata features such as genre, author, publication date and subject matter keywords and page count etc., there appears to be a gap in research in stylometry based recommendation. This approach to book recommendation could yield more relevant recommendations to readers, especially in generating recommendations of different authors from a source author.

This project studies fictional novels written by a set of prolific authors. Our goal is to explore the feasibility of style extraction and comparison, given a sample excerpt of text. Our target model's stylometric measurements of similarity between texts can be used to make style-based recommendations of different authors based on input excerpts of text from an author that you enjoy.

Specifically, we intend to address the following questions:

1. Can an author's writing style be extracted from their novels and represented within a model?
2. Can a model then identify the original author and similar authors when given a sample text input of a work that was not part of the training data?
3. How long does the text input need to be for a model to achieve good performance in its identification?

Literature Review

Stylometry is a relatively older field in Natural Language Processing. We have chosen to focus on vocabulary-centric methods, such as higher-level text embeddings. Doc2vec was referenced for document-level embeddings[1], in favour of sentence-level embeddings like SentenceBERT due to the faster execution time and greater accuracy of the former[2].

We also identified relevant vocabulary features for our Feature Engineering and k-Nearest Neighbors approaches, such as the frequency of an author's use of stopwords and punctuation[3]. We also focused on feature extraction techniques, such as rank-distance of function words, a crucial identifier of style. M. Popescu and LP. Dinu utilised rank-distance in the hierarchical clustering of function word vectors[4][5], which proved superior to simple frequency analysis. However, due to implementation difficulties, a different function word set was used instead, with k-Nearest Neighbors.

Datasets

The datasets consist of novels of Victorian-era authors and similar novels from other eras. This reduces the vocabulary difference between each novel due to varying genres. These were sourced from Project Gutenberg, a large online repository of eBooks that is available to the public. Text that is unrelated to the book's story (eg. author's notes, chapter titles, headers/footers) was removed before use.

Some books were used for testing only while others were used for both training and testing. The following table details the data used:

Author	No. of books	Training word count	Test word count	Usage
Charles Dickens	10	829,510	1,771,288	Training + Testing
Fyodor Dostoevsky	5	697,234	97,970	Training + Testing
Mark Twain	7	365,199	199,459	Training + Testing
Jane Austen	7	0	739,925	Testing
John Steinbeck	1	0	29,675	Testing

Table 1. Standardized dataset used for training and testing of all models

These 5 specific authors were chosen to expose a model's efficacy at extracting an author's style.

In regards to the authors chosen to train the models, Charles Dickens and Mark Twain are known to have many similarities between their writing styles[6]. On the contrary, Fyodor Dostoevsky was a Russian novelist whose works explored more psychological themes, and whose style is relatively distant in comparison. Seeing the model's classification of these 3 author's texts would reveal whether it truly understood their individual writing styles. For example, a model who could accurately classify Dostoevsky's works but is more undecided between Twain and Dickens can be judged to be more successful than one who could classify each of them decently but constantly misclassified Dostoevsky as one of the other two authors.

With respect to the remaining two authors, Jane Austen is often classified to have a similar style to Dickens[7], whereas John Steinbeck is commonly compared to Twain. Observing how a model classifies these authors without training on their novels would show if it can identify which author is most similar to them, and thus, a successful extraction of an author's writing style.

As for book selection, we tried to keep the content similar, to avoid having the models simply classifying them based on thematic differences. Thus, books that veered too far off in terms of themes such as "Personal Recollections of Joan of Arc" by Twain which had heavy medieval themes were not included.

Models

Naive Bayes (Baseline)

Being the baseline model, only simple preprocessing was conducted, case folding and stopwords removal. The list of stopwords were taken from NLTK. The TF-IDF value of each remaining word was computed and passed to the Naive Bayes model.

This model is effective as a baseline, since it establishes the limit of what a general vocabulary count-based model can plausibly do, without any further intervention. If style is an emergent property of text as a whole, then a more sophisticated model would be required. Improvements on this baseline would most likely require a deeper understanding of style extraction.

Doc2vec (Document-level embeddings)

The Doc2vec model is an extension of the Word2vec model. Apart from standard word-level embeddings, it also trains a document-level embedding vector, which is then used to represent an entire document[1]

A sentence-level embedding model in SentenceBERT[2] was initially considered, however, difficulties arose due to the difficulty in extending SentenceBERT to very long training data elements (e.g. entire novels, or novel chapters). It was also time-prohibitive to run SentenceBERT on much smaller segments of the dataset, so therefore the faster Doc2vec model was chosen instead.

The Doc2vec model follows the following process for training and testing:

1. Embed each training element into a vector, tagged with the author ID.
2. Embed each testing element into a vector.
3. Categorize the testing element by identifying the author ID whose training elements had the largest cosine similarity.

Key hyperparameters used when testing were the number of dimensions in the embedding vector, and the number of epochs used to train the model. Final decisions of the hyperparameters were **200** dimensions for the embeddings, and **100** epochs used in training. Some further analysis of the Doc2vec model can be found in the Results and Discussion section.

Feature Engineering (Logistic Regression)

A total of 17 features are extracted from text and passed into a multinomial Logistic Regression model. L2 regularization is used with the inverse of regularization strength as 0.8. This results in the weights shown in Figure 4a. If L1 regularization is used instead, some weights end up being very large while others disappear as can be seen in Figure 4b, causing worse performance. The model uses a variant of stochastic average gradient descent.

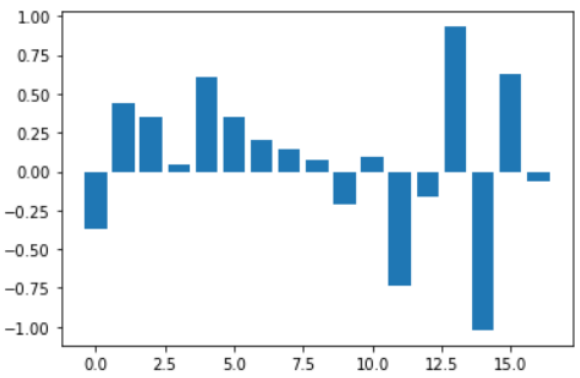

Figure 1a. Coefficients for Mark Twain in LR with L2 Regularization

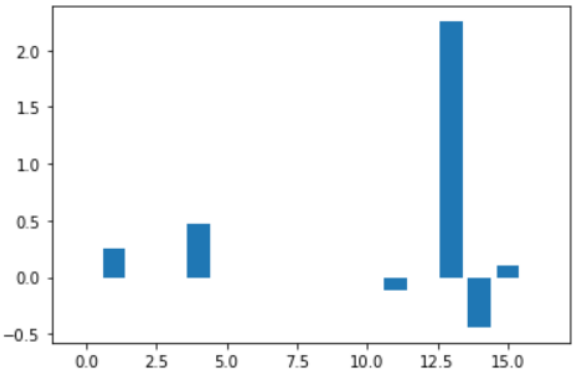

Figure 1b. Coefficients for Mark Twain in LR with L1 Regularization

The input text is POS-tagged to obtain information on the sentence structure.

List of features used:

- Personal pronoun POS tags per sentence
- Determiner POS tags per sentence
- Adjective POS tags per sentence
- Adverb POS tags per sentence
- Verb, past tense POS tags per sentence
- Any noun POS tags per sentence
- Interjection POS tags per sentence
- Modal POS tags per sentence
- Foreign word POS tags per sentence
- Stopword count per sentence (stopwords from NLTK)
- Capitalized words per sentence (to capture the number of named entities used, excludes word at start of sentence)
- Dashes per sentence
- Commas per sentence
- Italicized words per sentence
- Dialogue chunks per sentence
- Average number of words per sentence
- Average word length

Other features such as counting the number of contractions, exclamation marks and question marks were explored but did not improve performance. This seemed to be due to their common usage across all the authors, causing it to make differentiating them harder if it was used as a feature.

Analyzing the author's vocabulary was also attempted, but the huge difference in length between a training sample and a test sample made it incomparable. Training samples, which are longer in length, would have a significantly higher number of unique words used compared to test samples.

k-Nearest Neighbors (using Function Word rank)

Function word frequency is a feature commonly used in stylometric evaluation, as the use of function words (determiners, pronouns etc.) is not likely under the conscious control of authors[4]. However, instead of training on frequency directly, ranking function words according to frequency is more effective, as we are more concerned about the relative frequency of function words among authors.

In our implementation, we used a standardised set of 160 function words derived from the 'english' stopword set provided by NLTK. Case-folding was performed on both the NLTK stopword set and the training texts, in order to accurately track the use of function words.

1. Split and separately tokenize each novel.
2. For each novel segment, count and rank pre-determined function words from the standardised set.
3. Compute a rank vector for each segment, with each dimension being the frequency rank of a function word.

Order of dimensions follows the function word order given by the standardised set. For function words that do not appear in the segment, they are given the lowest possible rank.

4. Use the L1 distance metric to generate a k-Nearest Neighbors model from the rank vectors, with k = 17.
5. Select the author with the shortest total distance to the test-case rank vector.

Ensemble Model

The 3 non-baseline models were also combined into an overall ensemble model. The hyperparameters used in each of the models are the same as those used in the final form of each model. When test cases are received by this model, classification responses would be obtained from each of the 3 models. The final response was derived as follows:

- If there is majority agreement (i.e. 2 or 3 models select the same classification), then choose that response.
- Else, select the model with the highest 'confidence score'.

The 'confidence score' is output by all models for each classification, and is obtained as follows:

- **Naive Bayes**: Calculated using the probability score that is output by the model along with its prediction.
- **Doc2vec**: Calculated as the difference in cosine similarity scores between first and second choices.
- **Feature Engineering**: Calculated using the probability value output by the model via a softmax function.
- **k-Nearest Neighbors**: Computed using the formula `1 - (distance1 / distance)`, where `distance1` is the total distance of the selected author among the k shortlisted points, while `distance2` is the distance of the nearest point belonging to a different author from the selected author.

The confidence scores are normalized across the confidence scores of each model to the standard normal distribution, so that the resultant scores all follow the same distribution. This allows for direct comparisons to be made between the confidence scores of each model.

Testing Methodology

Evaluation Metrics

The main metric used in evaluating the performance of the models was the weighted F1 score. The F1 score was chosen because a good balance of precision and recall was desired, and the weighted score was chosen in the case that there were any imbalances in the sizes of the classes in the testing sets.

To further elucidate any tendencies of the models, the confusion matrices of each test run were also analyzed. These would be used to verify that the models were performing as expected.

Type of Test Data

Two test suites were constructed, with their test cases being obtained from novels that fell under one of two categories:

- 'Same Authors': Novels written by authors in the training dataset, but which were not used in training.
- 'Different Authors': Novels written by authors not in the training dataset, but whose writing styles were considered similar.

The test cases were parsed via a Python script, which selected lines of each novel to extract as test cases. The script extracted test cases deterministically, so multiple runs with the same parameters would lead to the exact same results.

The lines chosen were intentionally distributed evenly within each novel, such that if 10 test cases were parsed from a novel, then they would be evenly spaced. Since the writing style within a novel could vary depending on which section of the novel the test case was extracted from, the even spacing of the test cases would help to smoothen out any local variations in the writing style.

Additionally, for each author, the same number of test cases was taken per novel, to ensure that there was an equal representation of each author's novels within the test suites.

Test cases were not strictly kept to the exact same length, although a minimum character count was enforced, which could be varied as wished. Character count was used in favour of word count because it provided smoother measures of test case length. This feature of parsing different test case lengths would help to allow the exploration of performance improvements with greater test case lengths. Character count is approximately 5 times of word count, so the two will be used interchangeably with this value, for ease of reference.

Types of Tests Conducted

The 4 models, as well as the ensemble model, were each tested on the two test suites. The main variation made between tests was the length of each test case. This was done by adjusting the parameter in the Python script for the minimum character length per test case. A side effect of this change was that fewer test cases were parseable, since the novels did not have sufficient content to accommodate a large number of long test cases.

Another variation used in testing was by changing the type of training data used. To address the unbalanced amount of training data, novels were split into similarly-sized segments for training where each author had a similar number of segments. Although different authors' segments would have different lengths, normalization would reduce most effects that arise from it.

Other tests focused on specific models and their confusion matrices, such as analysis on the effectiveness of ensembling, the ability of some models to generalize to the 'Different Authors' dataset, and an ablation study of the best-performing model. The results of these various tests can be seen in the section below.

Testing Results and Discussion

Overview

These are the final results of each model on each test suite, using test cases of about 200 words in length, and 111 training samples divided among 41 samples from Charles Dickens, 34 samples from Fyodor Dostoevsky and 36 samples from Mark Twain.

Model	'Same Authors'	'Different Authors'
Naive Bayes (NB)	0.769	0.833
Doc2vec (D2V)	0.878	0.958
Feature Engineering (LR)	0.624	0.251
k-Nearest Neighbors (kNN)	0.333	0.330
Ensemble	0.820	0.529

Table 2. Headline performance of each model on both test suites

Doc2vec was our best performing model on both test suites, with F1 scores of 0.878 and 0.958 as underlined in the table. k-Nearest

Neighbors did not perform well for both test suites, mainly due to the low test case length of 200 words. Naive Bayes did exceptionally well despite being the baseline model, which we attribute to its vocabulary-based approach. The Feature Engineering model performed well in initial tests and did decently for the 'Same Authors' test suite, but very badly for the 'Different Authors' test suite, showing that it cannot generalize well. While the ensemble model did have a fairly good performance on the 'Same Authors' test suite, it did significantly worse for the 'Different Authors' test suite, likely being affected by the Feature Engineering and k-Nearest Neighbors models' bad performances for that test suite.

Our findings will be further elaborated on in the following sections.

Model Performance with Varying Training Sample Count

The models were tested with different training data length and number of samples.

To increase the number of samples and keep the length of samples similar within each author, each novel was segmented into smaller chunks. Each author's novels were split into similarly sized chunks. To maintain the fidelity of the text, sentences were not broken up and each sample's text belongs to only one novel.

The following table shows the different tests conducted. The best performing model's F1 score has been bolded and underlined.

Total no. of samples	Sample length (shortest - longest)	F1 scores ('Same Authors' test suite)
Total: **11** C.D: 4 F.D: 3 M.T: 4 (original samples)	C.D: 135,000 - 353,000 F.D: 202,000 - 253,000 M.T: 68,000 - 117,000	NB: 0.613 D2V: **0.698** FE: 0.547 KNN: 0.404 Ensem.: 0.638
Total: **61** C.D: 22 F.D: 19 M.T: 20	C.D: 36,000 - 46,000 F.D: 35,000 - 41,000 M.T: 17,000 - 22,000	NB: 0.784 D2V: **0.824** FE: 0.600 KNN: 0.333 Ensem.: 0.769
Total: **87** C.D: 32 F.D: 27 M.T: 28	C.D: 25,000 - 27,000 F.D: 25,000 - 27,000 M.T: 12,000 - 14,000	NB: 0.767 D2V: **0.847** FE: 0.618 KNN: 0.402 Ensem.: 0.800
Total: **73** C.D: 32 F.D: 27 M.T: 14	C.D: 25,000 - 27,000 F.D: 25,000 - 27,000 M.T: 23,000 - 34,000	NB: 0.587 D2V: **0.824** FE: 0.587 KNN: 0.333 Ensem.: 0.691
Total: **111** C.D: 41 F.D: 34 M.T: 36	C.D: 20,000 - 23,000 F.D: 20,000 - 21,000 M.T: 10,000 - 11,000	NB: 0.769 D2V: **0.878** FE: 0.624 KNN: 0.333 Ensem.: 0.820
Total: **150** C.D: 55 F.D: 46 M.T: 49	C.D: 15,000 - 16,000 F.D: About 15,000 M.T: 7,500 - 8,000	NB: 0.798 D2V: **0.851** FE: 0.653 KNN: 0.333 Ensem.: 0.822

Table 3. Test sets with varying sample text lengths and corresponding performance of each model

The first row in the table above shows the models' performances on the unaltered original datasets. As can be seen, splitting the novels into chunks improved performance significantly.

The k-Nearest Neighbors model is the only exception, with a fluctuating performance of between 0.3 to 0.4. This can be ascribed to the insufficient test case length of 200 words. It is possible that the limiting factor for performance in the kNN model is not the number of training data samples, but rather the length of each test case used in testing.

Additionally, it was discovered that keeping the number of samples similar among the different authors was more important than keeping the sample length similar across different authors. This is shown in the 2 tests highlighted yellow in the table above, where Mark Twain's samples were set to a similar sample length as the other 2 authors. Performance was lower for all the models, with the Naive Bayes and Feature Engineering models having a drastic drop. This could be due to under-representation of the author with a lower number of samples. Furthermore, the different sample length is addressed through normalization and its impact is much smaller.

However, increasing the number of samples at the cost of sample length can also negatively affect a model's performance. This is observable in the performance decrease in Doc2vec when the number of samples increased from 111 to 150 in the last test. The respective F1 scores are highlighted purple. Thus, since Doc2vec is the best performing model with a score of 0.8756, we decided to work with the 111 training samples.

Model Performance with Varying Test Case Length

Unlike document-level embedding models like Doc2vec, our tests with k-NN have demonstrated its weakness with shorter test case lengths of below 1000 words. However, for test cases of greater than 2000 words, k-NN performance gradually improves to become comparable to Doc2vec. Below are some confusion matrices demonstrating the effect of varying test case lengths on the performance of k-NN. All tests are conducted on the identical 17-NN model with the same weighted-distance decision metric and internal hyperparameters. The model is started off with test case lengths of 2000 words, and the number of words in each test case is gradually reduced, to 1000, and finally to 200.

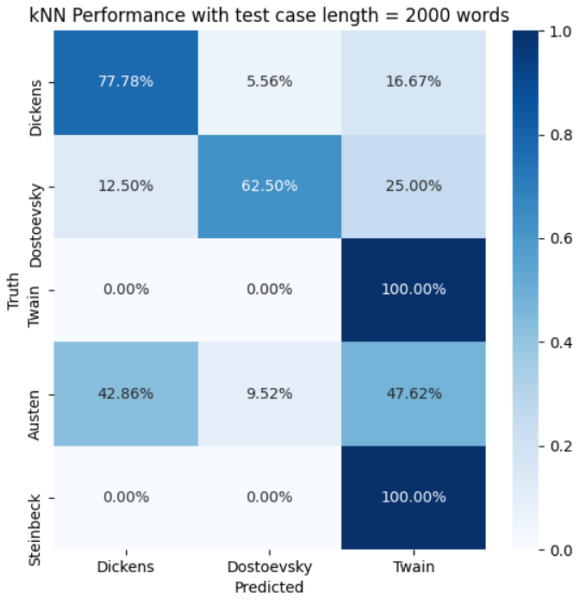

Figure 2a. Confusion matrix for kNN with test case length = 2000 words

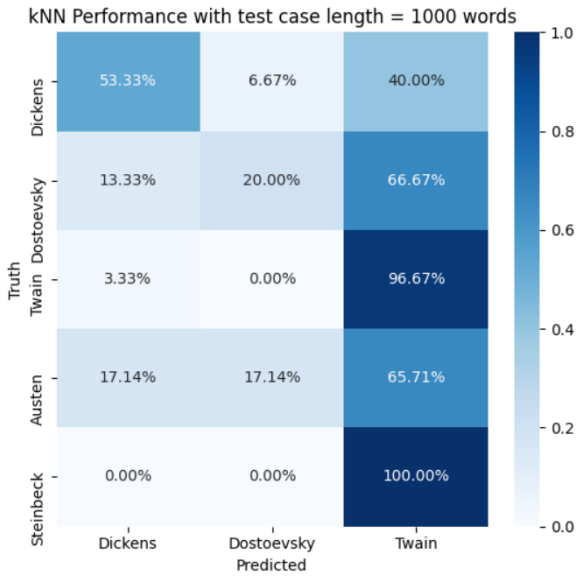

Figure 2b. Confusion matrix for kNN with test case length = 1000 words

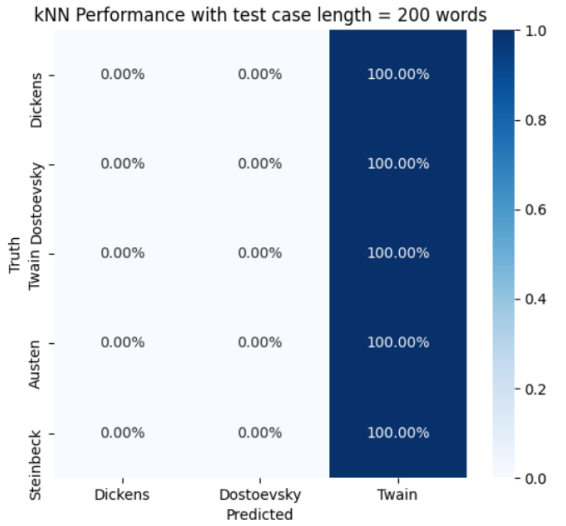

Figure 2c. Confusion matrix for kNN with test case length = 200 words

Examining the confusion matrices, we can observe that the 17-NN model becomes increasingly biased towards Mark Twain as performance degrades. While training samples for Mark Twain had a lower average word count than Charles Dickens and Fyodor Dostoevsky, the link between a lower word count and the bias towards is not clear.

Overall, we observe that accuracy increased with increasing test case length, from 0.331 for 200 words, to 0.533 for 1000 words, then finally to 0.824 for 2000 words.

Generalizability of Models to 'Different Authors' Tests

To test our models' ability to generalize to unseen authors, we included two unseen authors in our test suite: Jane Austen and John Steinbeck. The confusion matrices are shown below.

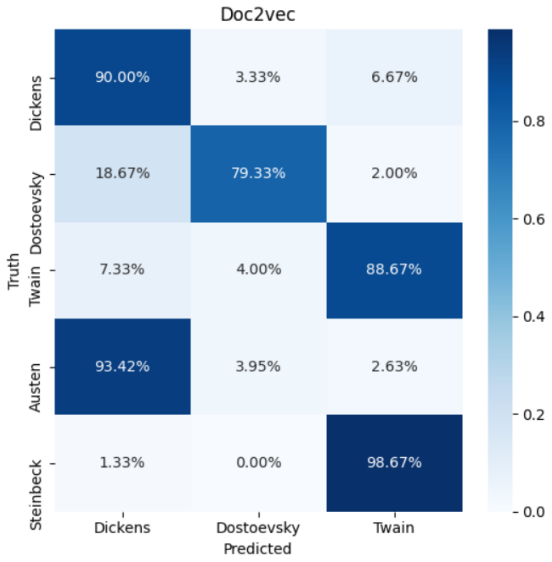

Figure 3a. Confusion matrix for Doc2vec with 150 training samples

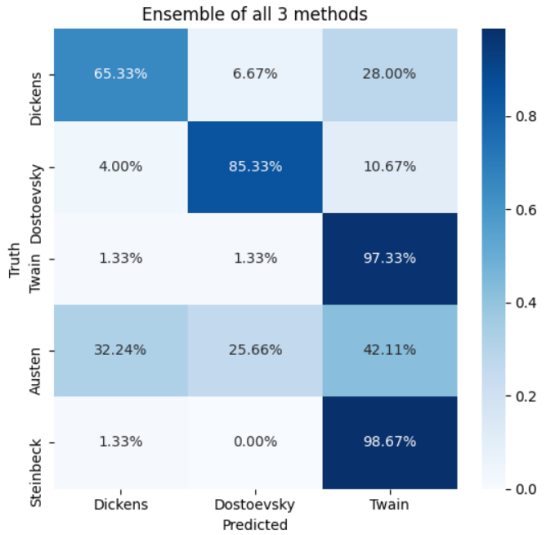

Figure 3b. Confusion matrix for Ensemble with 150 training samples

These models have a relatively similar F1 score of 0.851 for the Doc2vec model, to 0.822 for the ensemble model. However, with reference to Figure 2a, the Doc2vec model classifies most of Austen's samples to Dickens and Steinbeck's to Twain, which are the expected results. However, the ensemble model is undecided on Austen's samples, classifying only 32.24% of the samples to Dickens, and classifies a larger proportion of 42.11% to Twain. However, performance is better for Steinbeck, where almost all test cases are correctly attributed. This shows that the ensemble model is not able to understand that Austen and Dickens's styles are similar and that it is not as capable as the Doc2vec model in generalizing to unseen authors.

Effectiveness of Ensembling Method

The ensemble method performed worse than the Doc2vec model in general, but performed better than the other 3 models used. This is primarily because the Doc2vec model significantly outperforms the other 2 models, and therefore does not benefit from them.

However, in some specific situations, a modified ensemble model performs better than any individual model. For example, in the following confusion matrices, the Doc2vec, Feature Engineering, and Ensemble models are tested on the 'Same Authors' test suite with **test case length of 1000 characters** and **11 samples used for training**. Note that the 'Ensemble' method in this case combines the

Doc2vec and Feature Engineering models only. Also, notice that these were **not** the final models used.

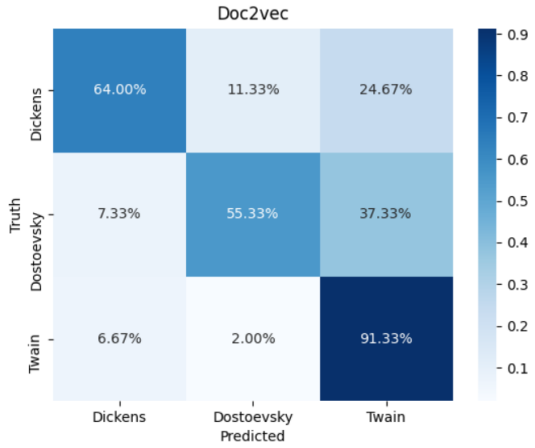

Figure 4a. Confusion matrix for Doc2vec

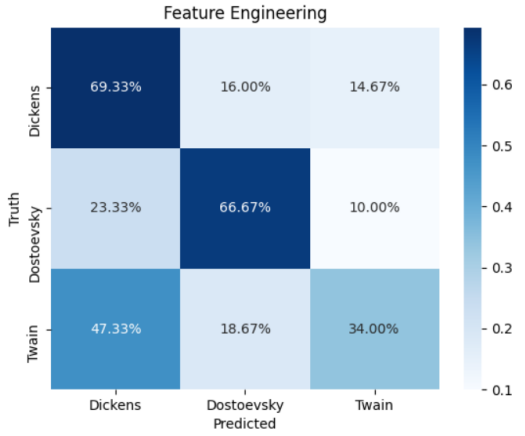

Figure 4b. Confusion matrix for Feature Engineering (Logistic Regression)

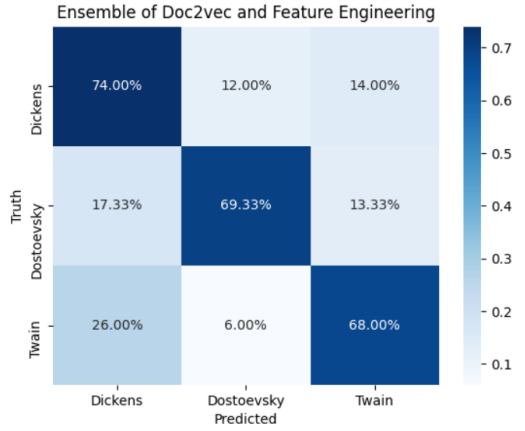

Figure 4c. Confusion matrix for Ensemble

FIgures 4a and 4b depict the individual performances of the Doc2vec and Feature Engineering models, and FIgure 4c depicts the performance of the ensemble model that used both.

Comparing the individual models, Doc2vec tended to predict Mark Twain, and Feature Engineering tended not to predict Mark Twain. Therefore, when combining the two into an ensemble, the resultant performance was an improvement, since the two preferences cancelled out. Specifically, the performance of the ensemble on Fyodor Dostoevsky and Charles Dickens was better than either model alone. All three authors are also predicted with a more even frequency in the ensemble model, and the overall accuracy is higher.

Effectiveness of Baseline Model

The baseline model performed surprisingly well in general. It consistently outperformed the Feature Engineering and k-Nearest Neighbors models, but was itself outperformed by the Doc2vec model. With the current training and testing datasets, it appears that vocabulary still plays a key role in the identification of writing style. Despite this, it can be seen that the baseline model fails to capture some element of writing style, since the Doc2vec model performed significantly better.

Nevertheless, if this baseline model is considered the minimum that can be achieved through simple text processing, then it might be the case that only the Doc2vec model is considered successful.

Ablation Study of Best-Performing Model

An ablation study was carried out on the best performing model (Doc2vec) to examine which factors contributed the most to its performance. The results are summarized in the table below.

Parameters used	**F1 score (weighted)**	
	'Same Authors'	'Different Authors'
Default*	0.878	0.958
11 training samples	0.698	0.668
50 word test cases	0.689	0.789
50 dimensions	0.866	0.933
20 epochs	0.831	0.967

Table 4. Ablation study results - weighted F1 score for the 'Same Authors' and 'Different Authors' test sets

*The ‘default’ parameters used are: 111 samples used in training, test length of approximately 200 words, vector dimensional count of 200, training epochs of 100. For all other combinations of parameters, only one parameter was varied at a time.

In general, performance was markedly weaker when the number of training samples was decreased by a factor of 10, and also when the length of test cases was reduced by a factor of 4. Possible reasons for these are discussed in the sections above on the effects of the number of training samples and the length of test cases used.

Performance in general was similar when the hyperparameters of the Doc2vec model (i.e. the dimensional count of the embedded vector, and the number of epochs used in training) were tweaked. For this reason, it is suggested that the form of the training and test data is much more important than the specific values used for the hyperparameters of the model.

Conclusion

Style is an abstract and hard-to-quantify concept. Simple attempts at comparing engineered vocabulary characteristics such as in Naive Bayes and Feature Engineering were not as effective as more holistic document-level embeddings such as Doc2vec. However, with longer test case lengths (around 2000 words), most models were able to identify the correct author amongst the small pool of authors. This result was achieved even after controlling for vocabulary and genre differences in the datasets. Therefore, for successfully extracting ‘style’ from novels to make recommendations to users, we suggest a document-level embedding model similar to Doc2vec. Future research may focus on expanding the training and testing datasets to more authors from different genres and time periods, to examine if the trends discovered in this paper continue to apply.

Acknowledgments

Group 18 would like to sincerely extend our gratitude to our lecturer, Prof. Kan Min-Yen, and our TA Rahul Baid, for their helpful feedback. Their guidance was indispensable in refining our research direction.

References

[1] Q. Le and T. Mikolov. Distributed Representations of Sentences and Documents (PDF). Retrieved 15 April 2021

[2] I. Gurevych and N. Reimers, “Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks,” 2019

[3] M. A. Thornton, “Analyzing stylistic similarity amongst authors,” 11-Aug-2015. [Online]. Available: http://markallenthornton.com/blog/stylistic-similarity/. [Accessed: 15-Apr-2021].

[4] LP. Dinu and M. Popescu. 2008. Rank distance as a stylistic similarity. In Coling 2008. pages 91–94.

[5] M. G. Kendall, F. Mosteller, and D. L. Wallace, “Inference and Disputed Authorship: The Federalist,” *Biometrics*, vol. 22, no. 1, p. 200, 1966.

[6] S. Yuan, “A Comparative Analysis of Charles Dickens and Mark Twain,” *Academic Journal of Humanities & Social Sciences, 3(7).*, vol. 3, no. 7, pp. 154–159, 2020.

[7] “Authors Like Jane Austen,” *Jane Austen - English Author*, Jan-2021. [Online]. Available: https://www.janeausten.org/authors-like-jane.php. [Accessed: 22-Apr-2021].

Appendix A

The repository containing our source code for this project can be found at the link here. Specific information about the files used in the repository can be found in the README.md file.