# Application of Markov Chain in Linguistic Analysis

Cloris Li, Irene Li, Hongda Li, Khoa Vo

March 2020

# 1 Introduction

In this paper, our objective is to model novels written by different authors through Markov chain with letter to letter transition matrix. We will focus on two authors: Charles Dickens and Mark Twain, which are chosen based on group members' knowledge and preference on English writers. We consider the above two authors are well-known to English readers, and they have quite different writing styles. Charles Dickens was an British writer from 1812 to 1870 with style more poetic and focuses on Victorian romanticism[1], while Mark Twain was an American writer from 1835 to 1910 known for keen satire and careful choice of diction[28].

Our purpose is to investigate whether the two authors can be represented by two separate clusters of transition matrices generated from their novels or there is an overlap between their transition matrices under various assumptions and conditions. We will also examine the closeness of author's writing to his own profile. This is analyzed through calculating distances of each novel's transition matrix to his own author's profile matrix, as well as the cross distances of each novel's transition matrix to the other author's matrix. Transitional matrix is first be defined by letter by letter base and later modified to two letter by two letter sequence. The effect of changing assumption on syntactic elements such as handling proper noun and emphasizing rarely used words is studied. We also investigated the effect of using different distance measures for comparison.

# 2 Background

Our group project was interested in the real-world application of Markov Chain. We first started with the idea of using Markov Chain to model the texts of a few authors as trained data to identify the author of a text. After some discussions, we decided to narrow our project down to investigate the novels of two authors, Charles Dickens and Mark Twain, to see whether transition matrices generated by the novels of each author has a "cluster" that is distinctly theirs or there is a large overlap between their works.

Markov chains had been a commonly technique applied in literature analysis. Since each author has their own writing style, the literature written by different authors have specific syntactic

elements that could be detected and traced back to its author. Using the transition matrix of the Markov Chain, we could construct a profile for the author's writing and compare it to other author's writings. The application of authorship attribution was commonly used in plagiarism detection, deducing the writer of threatening or harassing e-mails and other problems with unclear authorship.[16]

A study on this topic was done and published in 2014 by Osipovs, Rinkevics, Kuleshova and Borisov. Their research is focused on using Markov chains in the task of text authorship identification. In the research, they created a model using Markov chain transition matrix to represent the author's writing style. This model is constructed based on the texts for which the authorship of the targeted author is known. The result produced from the model should indicate whether a new text, whose author is unknown, is written by the target author or not. They used texts with more than 26000 words to make sure it's large enough to contain information about specifics of writing styles. They also implement various ways to pre-process the texts such as dealing with punctuation and short words to see if yields different results. Results of this study indicate that most useful information about the author's style is contained in long words, and removal of short words and punctuation marks does not affect the model.[15] With this reference, we will ignore the effect of punctuation by considering them as space that separates words, and detail will be explained later in model assumption.

A paper published in 2006 by Sanderson, Conrad, and Simon Guenter used similar techniques in relatively short texts (300 to 5000 words) authorship attribution. They worked on a large dataset which includes 50 authors with each author covering several topics. Because they wanted to avoid the probability that the model is based on topics rather than the author's writing styles. The completeness of their model was verified by matching testing texts to the correct authors. In conclusion, they claimed that analyzing texts using character and word Markov chains is less useful when dealing with short texts. A good performance of the model should be constructed base on texts with more than 5000 words.[16] This is consistent with the later research mentioned above, where long texts are used as data for text analysis. With these reference in mind, we decide to focus on analyzing novels from selected authors since the long texts will provide enough information to build models that represents unique writing style for the two chosen author. We

also use different ways of pre-processing our texts in adjustment section to see how results changes with or without proper nouns.

Authorship identification is not only applicable to book literature, but also social media text post. A research by Daniel Freeman aim to predict the author of an unknown post on the social media network Twitter character Markov chains. Freeman chooses to analyze Twitter posts since Twitter posts are different with formal writings in both length and writing standard, such as the use of hashtags. Markov chain transition matrix for this model is constructed based on tokens of characters and also tokens of words. A collection of Markov chain transition matrices from different authors are generated. Freeman calculates the probability that a new text could be represented by the each existing Markov chains, and the highest probability corresponds to the authorship predicted by this model. The author found that it is easier to identify authorship with Twitter posts using character as tokens, since there are more unique transitions between characters in informal short texts that captures writer's individuality.[12] Our project is intended to perform data modeling by tokens of characters on long novels, on which the technique mentioned in this paper did not work best. We consider authors can variate their writing style greatly in novels, which capture more individuality compare to formal literature like academic writings. This is interesting because with different types of writing, we will need to model our data differently.

A paper written by Jagtap, Avinash, and Limbore has many similarities with our project. We create model by using Markov chains that based on the probabilities of subsequent letters, and all punctuation and formatting are removed. Also, the capital letters in the texts are converted into lower cases. Thus, the transitional matrices in both papers have 27 state spaces which corresponds to 26 letters and space. However, the difference between their model and ours is that they calculate the average of the elements of the matrices from all the texts by each author and then consider the probability of new text being generated by each of the transition matrices. Whereas we have different matrices for each text by the same author.[14] Instead of attributing testing texts to the correct authors, we verify our model by illustrating the clouds of target authors, which is constructed by the matrices from their novels, are not overlapped. In order to show the relationship of the text clouds, we calculate and analyze the distances between matrices. The

procedure of creating our model and analyzing the distances are described in more details below.

# 3 Model

In this model, we will generate transition matrices for novels written by Charles Dickens and Mark Twain that capture their writing styles. In addition, a centroid of each author will also be generated using the aggregated texts, which will be specified below. We will compare the distances of each author's literature to their own centroid as well as distance between the two author's writing.

## 3.1 Data

We will be using 10 novels written each author of our interest. Names of the literature are specified in the table below.

| **Charles Dickens** | **Mark Twain** |
|---|---|
| A Tale of Two Cities | A Connecticut Yankee in King Arthur's Court |
| Bleak House | Adventures of Huckleberry Finn |
| David Copperfield | Personal Recollections of Joan of Arc |
| Dombey and Son | Life on the Mississippi |
| Great Expectations | The Adventures of Tom Sawyer |
| Little Dorrit | The American Claimant |
| Olive Twist | The Gilded Age: A Tale of Today |
| Our mutual friend | The Mysterious Stranger |
| The Life And Adventures Of Nicholas Nickleby | The Prince and the Pauper |
| The Pickwick papers | The Tragedy of Pudd'nhead Wilson |

The text files of these novels can be found and downloaded from Project Gutenberg. These data files are pre-processed by deleting all the text related to Project Gutenberg so that only writing by the author is considered when generating the transition matrix.

## 3.2 Model Assumptions

### 3.2.1 Transition matrix

When constructing the transition matrix, we assumed that an author's writing style can be characterized by the sequence of individual letter and space used in his/her writing. Thus, we will be considering the probability of one character following another character in the transition matrix of our model.

### 3.2.2 Syntactic elements in text pre-processing

Regarding syntactic elements, we assume the following when pre-processing the text files.

- Difference between upper and lower case letters are ignored. Upper case letters are generally used as start of the sentence or proper nouns, and we consider changing the upper cases to lower cases has no effect on individual author's style.
- Apostrophes are removed. We believe it usually does not separate two meaningful word. For example, it is not meaningful to separate the word "don't" into "don" and "t".
- Punctuation is consider to have the same function as space, which is to separate words.
- A space followed by a space has no useful information on apprehension of author's writing style. So consecutive spaces are treated as a single space in the texts.

By employing the above assumptions, our model will only be focused on authors' writing in terms the way they use English words. More specifically, we believe author's writing feature can be captured by the sequences of letters and spaces used in their literature. We do realize that by omitting the difference in capitalized and lower case letters, we might have a frequently occurred sequence of the book character's name. We will be caution with this and will later address in result interpretation and alter in adjustment.

To implement the assumptions above, all alphabetical letters are converted to lower case after the above process. Characters other than the alphabetical letters(i.e. punctuation) are replaced with space character, with the exception of apostrophe, which is removed. All consecutive spaces are collapsed into one single space. We will pre-process all texts used by this method before generating our transition matrices.

### 3.2.3 Measure of distance

There are many different ways to compare how similar two authors' writings are by determining the "difference" between two transition matrices. One of those is to measure the distance between two transition matrices created from the writings of each author. In our model, we will calculate the Euclidean distance between two transition matrices (i.e. Frobenius or 2 norm of the difference of matrices). The formula to calculate the distance is:

$$\text{dis}(A, B) := ||A - B||_F = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n} |a_{ij} - b_{ij}|^2}$$

where $A, B$ are 2 transition matrices with $n$ states under comparisons, and $a, b$ are entries of matrices. Using this distance measure, two identical matrices will have a distance of 0; matrices with different entries will have a distance greater than 0, and the more different the entries, the greater the distance.

### 3.2.4 Measure of author metric space(centroid and cluster)

To compare the similarity of two authors, we will define a centroid as the profile of the author. The centroid is a $\mathbb{R}^{27\times 27}$ transition matrix generated from aggregating all 10 texts from the author(i.e. appending all 10 texts together). The cloud of an author would be interpreted by calculating the distances of 10 transition matrices of individual novels to the author's centroid. The distance between two authors will be the Euclidean distance of two centroids of the authors.

## 3.3 Generate Transition Matrices

By our assumptions, the transition matrix of our model will be in $\mathbb{R}^{27\times 27}$, with the first 26 states corresponding to the 26 letters in English alphabet, and the last state representing space that separates the characters. We generate the transition matrix using Python. We iterate through the entire text to count the frequencies of letter $\alpha$ following letter $\beta$, where $\alpha$ and $\beta$ are the 27 English letters and space character. The probability of letter $\alpha$ following letter $\beta$ is then calculated by dividing the frequency of letter $\alpha$ following letter $\beta$ by the total frequency of letter $\beta$ occurs.

The matrix $\mathbf{P}_{\text{tm27}}$ can be represented as:

$$\mathbf{P}_{\text{tm27}} = \begin{bmatrix} \mathbb{P}(a|a) & \mathbb{P}(b|a) & \dots & \mathbb{P}(z|a) & \mathbb{P}(*|a) \\ \mathbb{P}(a|b) & \mathbb{P}(b|b) & \dots & \mathbb{P}(z|b) & \mathbb{P}(*|b) \\ \vdots & & & & \\ \mathbb{P}(a|z) & \mathbb{P}(b|z) & \dots & \mathbb{P}(z|z) & \mathbb{P}(*|z) \\ \mathbb{P}(a|*) & \mathbb{P}(b|*) & \dots & \mathbb{P}(z|*) & \mathbb{P}(*|*) \end{bmatrix}$$

Where $\mathbb{P}(a|b)$ denotes the probability of letter "a" following by a letter "b" in the given text, etc. The character "*" in the matrix represents the space character.

# 4 Results

Using the method described above, we generate 10 transition matrices for the novels of each author, and 1 transition matrix as the profile matrix (i.e. centroid) of each author. Python 3.8 is used to construct the model, and code can be found in the appendix section. It takes approximately 1 minute to calculate all distances on macOs system with processor 2.4 GHz Quad-Core Intel Core i5. By calculating the distances of each transition matrices to the authors' centroids, we can analyze the "cloud" information of each author indicated in the table below.

For Charles Dickens and his 10 novels, the distances from his novels to his centroid and cross distances of his novels to Mark Twain's centroid are shown in the table below.

| **Charles Dickens' Novels** | **Distance to Centoid** | **Distance to Twain's Centoid** |
|---|---|---|
| A Tale of Two Cities | 0.3646 | 0.4526 |
| Bleak House | 0.2511 | 0.4212 |
| David Copperfield | 0.2416 | 0.3723 |
| Dombey and Son | 0.3304 | 0.4780 |
| Great Expectations | 0.3721 | 0.4796 |
| Little Dorrit | 0.2710 | 0.4357 |
| Oliver Twist | 0.3789 | 0.5340 |
| Our mutual friend | 0.4407 | 0.6524 |
| The Life And Adventures Of Nicholas Nickleby | 0.2519 | 0.4071 |
| The Pickwick papers | 0.3772 | 0.4724 |

From the table above, we find that all 10 books from Charles Dickens are closer to his own profile than to Mark Twain's profile. The average distance of the 10 novels to Charles Dickens' centroid is approximately 0.3274 with standard deviation of 0.0655. To visualize how the distances are distributed, we plot a histogram of the distances of Charles Dickens' novels to his centroid, with a blue vertical line indicating the average distances.

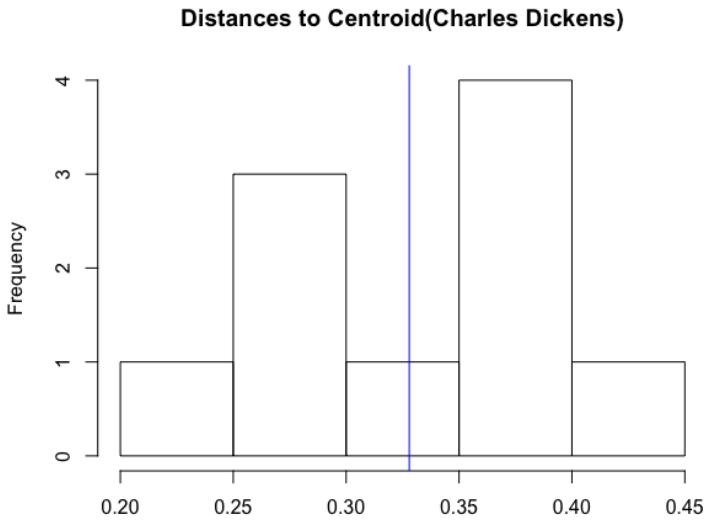


Figure 1: Histogram of Euclidean distances of Dickens' novels to own centroid in original model.

We can see that distribution of distances is quite symmetry around the the average distance from centroid, with two peaks close to the mean and low frequencies at two tails. The thin tails of histogram imply that we have 1 novel really close to the author's profile and 1 novel really different from Dickens' profile. If we imagine the transition matrices of the author lying in a higher than 3 dimensional space, it would be a cloud of matrices where most of the matrices would lie with a distance around 0.25 to 0.4 (i.e. roughly the average distance to centroid) around the centroid, with just a couple matrices much closer or much further away from the centroid.

Similarly, the distances from Mark Twain's novels to his own profile and cross distances of his novels to Charles Dickens' profile are shown in following table.

| **Mark Twain's Novels** | **Distance to Centoid** | **Distance to Dickens' Centoid** |
|---|---|---|
| A Connecticut Yankee in King Arthur's Court | 0.2466 | 0.4514 |
| Adventures of Huckleberry Finn | 0.7091 | 0.9068 |
| Personal Recollections of Joan of Arc | 0.4697 | 0.5084 |
| Life on the Mississippi | 0.2173 | 0.4241 |
| The Adventures of Tom Sawyer | 0.2820 | 0.5176 |
| The American Claimant | 0.2895 | 0.4600 |
| The Gilded Age: A Tale of Today | 0.2648 | 0.4208 |
| The Mysterious Stranger | 0.3999 | 0.5800 |
| The Prince and the Pauper | 0.4282 | 0.4207 |
| The Tragedy of Pudd'nhead Wilson | 0.4677 | 0.6441 |

Unlike Charles Dickens, most of Mark Twain's novels have a smaller distance to own centroid than cross distance to Twain's centroid with the exception of *The Prince and the Pauper*. The average distance of the 10 novels to Mark Twain's centroid is approximately 0.3775 with a standard deviation of 0.1421, and we can visualize the distribution of distances by plotting a histogram.

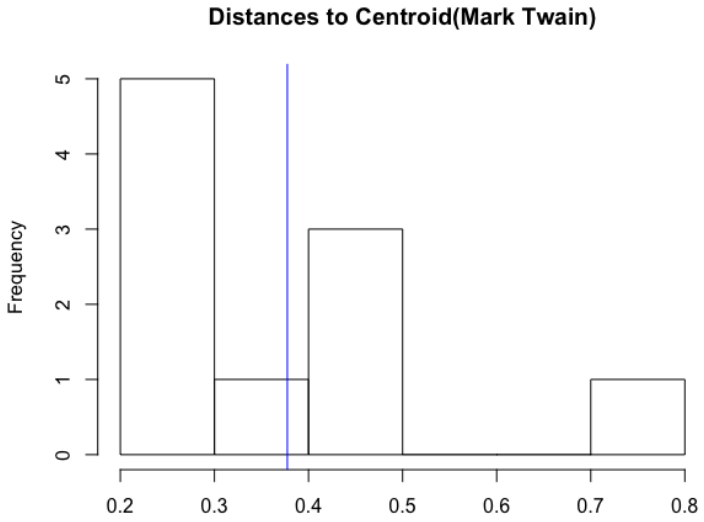


Figure 2: Histogram of Euclidean distances of Twain's novels to own centroid in original model.

Unlike Charles Dickens' distance distribution, the distances of Mark Twain's texts to his centroid seems to be skewed left, with more novels similar to his profile and 1 novel really different from the profile. This can be represented by cloud of matrices with most of the matrices lied with a distance of 0.2 - 0.5 to the centroid, and 1 matrix almost twice the distance away from the centroid. We also notice that the standard deviation of the distances is greater than that of Charles Dickens, and this could imply a greater variation in Mark Twain's writing styles. The majority of that variation can possibly explained by the novel with furthest distance to centroid.

From the table of distance above, we observe that the novel with furthest distance(however not neccesarily an outlier) to Mark Twain's profile is *Adventures of Huckleberry Finn*, which is surprising because many people consider it as his most iconic work. We then take a closer look into the transition matrix of *Adventures of Huckleberry Finn* and compare it to the Mark Twain's centroid. A possible explanation is the effect of special nouns in this book, such as character names. In the adjustment section, we remove the proper nouns to see if this explanation is reasonable. To further investigate this, we generate the difference matrix by taking their difference: $A - B$, where $A$ is the transition matrix for *Adventures of Huckleberry Finn* and $B$ is the transition matrix for Twain's centroid. A color coded matrix is shown in the following figure.

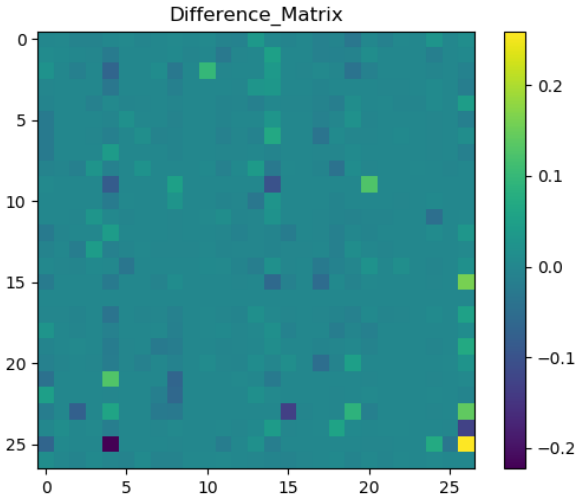


Figure 3: Difference matrix of *Adventures of Huckleberry Finn* and Mark Twain's centroid.

As we can observe that the largest difference occurs in row 26 and column 27 of the difference matrix(index in figured is shifted by 1 due to programming technicality). This corresponds to the sequence "z[space]", which means a significant difference in the use of words ending with letter "z" in *Adventures of Huckleberry Finn*. We look up main characters name in *Adventures of Huckleberry Finn*, and none of those ends with letter "z". Thus, proper noun does not seem to be a factor that cause the large distance from centroid. This imply that *Adventures of Huckleberry Finn* has relatively distinct writing style compare to Twain's other literature.

We also calculate the Euclidean distance between the two centroids of Charles Dickens and Mark Twain, which is 0.3595. This distance between two authors is greater than 0.3274, which is Charles Dickens' cloud "radius", but smaller than Mark Twain's cloud "radius" of 0.3775, where "radius" is the average distance of novels to own author's centroid. Since the distance between two authors' centroids does not seem significantly larger than the authors' clouds "radius", we cannot make a conclusion based on this information. However, we can plot a comparative box-plot of novel's distances to own author's centroid and their cross distances to the other author's centroid and observe if there exist any significant difference.

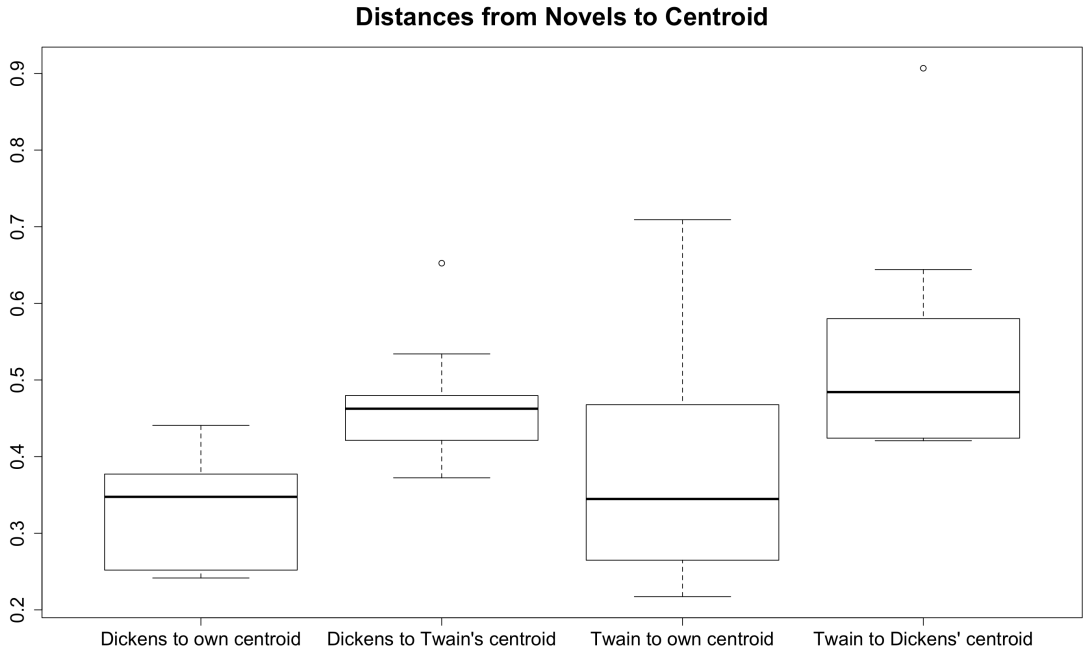


Figure 4: Comparative box-plot of novel's distances to own author's centroid and their cross distances to the other author's centroid in original model.

The left two boxes show the distances from Charles Dickens' 10 novels to his centroid and to Mark Twain's centroid. We conclude that these 10 texts are closer to Charles Dickens's centroid since the second box is higher than the first one without overlapping. The spreads of these two groups of data have big difference. The second box is much smaller than the first box means the variation of the cross distances of Charles Dickens' novels is small. We could make a guess that the ten books are spread around Charles Dickens's centroid but all of them are far from Mark Twain's centroid. Thus, we say most of Charles Dickens's ten texts were written in his own writing style compared to Mark Twain's styles. We can see that there is overlap between cross distances and distances of Mark Twain's texts, meaning it is hard for us to make a clear conclusion about his cloud, even though the median distance from Mark Twain' s texts to Charles Dickens' centroid is greater than these texts to their own author's centroid. Thus, because of the overlap of the box areas, we do not have enough evidence to conclude that the two authors have their own clusters

without overlapping with each other. One thing to notice is that both cross distances box-plots have extreme high values. Twain's *Out Mutual Friend* is much further away from Dickens' profile, and Dickens' *Adventures of Huckleberry Finn* is exceptionally further from Twain's profile. This could potentially indicate that those two novels have very distinct style to another author's writings. However, we do aware that it could potentially be the effect of repeating proper noun, and we will adjust the model later to see if these outliers still present.

# 5 Adjustments and Extensions

In the previous section, we create transition matrices to represent texts and analyze authors' writing styles. However since the matrices live in a rather high dimensional space, we cannot easily conclude that based on the distances we have calculated so far. The distances used in the previous model is calculated based on one of the ways to compare matrices. Some interesting results occurs in original model results due to possible effects of special nouns or just difference in writing styles. To study how model results changes as the above condition changes, we make several adjustment and extensions in this section.

## 5.1 Omitting Proper Nouns

In our original model, we convert the capitalized letters to lowercase so that we don't lose any written information(i.e. all letters and space) from the author. However, characters' name or names of the locations in a particular novel usually have uncommonly used letter sequences. This could have noticeable effect when generating character by character transition matrix, which could potentially bias the results. For example, if we change the main character's name from *Adventures of Huckleberry Finn* from "Huck" to "Jack", the distance between these two texts will not be zero, while syntactically they are identical.

To prevent this situation, we can remove all repeated proper nouns designated to a particular novel. However, the implementation to remove just those proper noun is complicated. For the sake of simplicity, we will remove all the words with capitalized first letter when pre-processing the texts files. We are aware of the fact that words at the start of sentences would also be removed. Since the novels we choose are long texts, the rest of the texts will still capture enough

syntactic information for us to analyze with capitalize words removed. We then analyze the two clouds of authors with the new adjustment of ignoring proper noun.

The distances of Charles Dickens to each author's centroid with new assumption is shown in the table below.

| **Charles Dickens' Novels** | **Distance to Centoid** | **Distance to Twain's Centoid** |
|---|---|---|
| A Tale of Two Cities | 0.2030 | 0.3670 |
| Bleak House | 0.1352 | 0.3939 |
| David Copperfield | 0.1737 | 0.4002 |
| Dombey and Son | 0.1330 | 0.4567 |
| Great Expectations | 0.1783 | 0.3609 |
| Little Dorrit | 0.1465 | 0.4125 |
| Oliver Twist | 0.2098 | 0.4063 |
| Our mutual friend | 0.1673 | 0.4556 |
| The Life And Adventures Of Nicholas Nickleby | 0.1768 | 0.4233 |
| The Pickwick papers | 0.2234 | 0.3430 |

We can see all novels are closer to Dickens' own centroid than to Twain's, which is consistent with original model result. We also visualize the distribution of distances in a histogram.

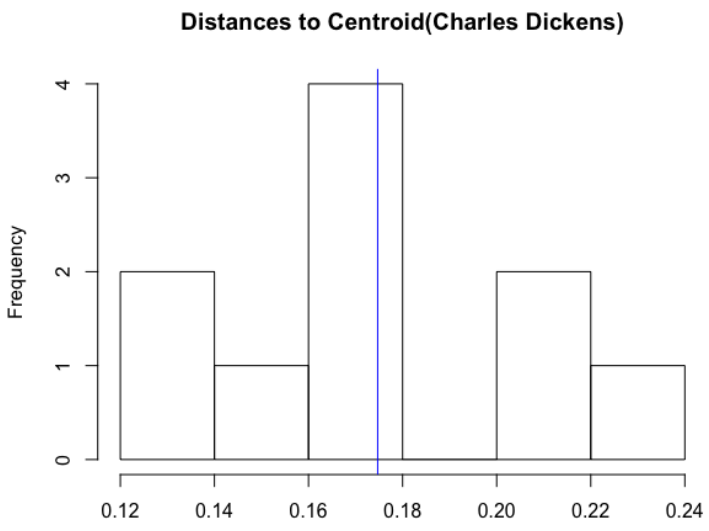


Figure 5: Histogram of Euclidean distances of Dickens' novels to own centroid in Adjustment 1(omit special noun).

The distribution of distances also seems symmetric and roughly normal. The new average distance of Dickens' novels to the centroid is 0.1746, which is much smaller than 0.3274, the average distance when capitalized words are not removed. This indicates that the clouds of Charles Dickens' novels to his profile is less spread when the proper nouns are not considered as his writing style. The new standard deviation of distances is 0.0293, which roughly reduces the original standard deviation 0.0655 by half. This implies that when we exclude the affect of proper nouns in Charles Dickens' writing, the variation on writing is much lower.

For Mark Twain, the table of distances and histogram are shown below.

| Mark Twain's Novels | Distance to Centoid | Distance to Dickens' Centoid |
|---|---|---|
| A Connecticut Yankee in King Arthur's Court | 0.1476 | 0.3637 |
| Adventures of Huckleberry Finn | 0.6298 | 0.8827 |
| Personal Recollections of Joan of Arc | 0.1735 | 0.4013 |
| Life on the Mississippi | 0.1829 | 0.3701 |
| The Adventures of Tom Sawyer | 0.2152 | 0.4945 |
| The American Claimant | 0.2338 | 0.3833 |
| The Gilded Age: A Tale of Today | 0.1613 | 0.3137 |
| The Mysterious Stranger | 0.2440 | 0.4858 |
| The Prince and the Pauper | 0.3972 | 0.3719 |
| The Tragedy of Pudd'nhead Wilson | 0.2188 | 0.3471 |

After we ignoring the special nouns, most novels from Mark Twain are still closer to his own profile except for *The Prince and the Pauper*. Since special noun is no longer a factor for the difference, this imply that the writing style of *The Prince and the Pauper* is quite ambiguous between Dickens and Twain.

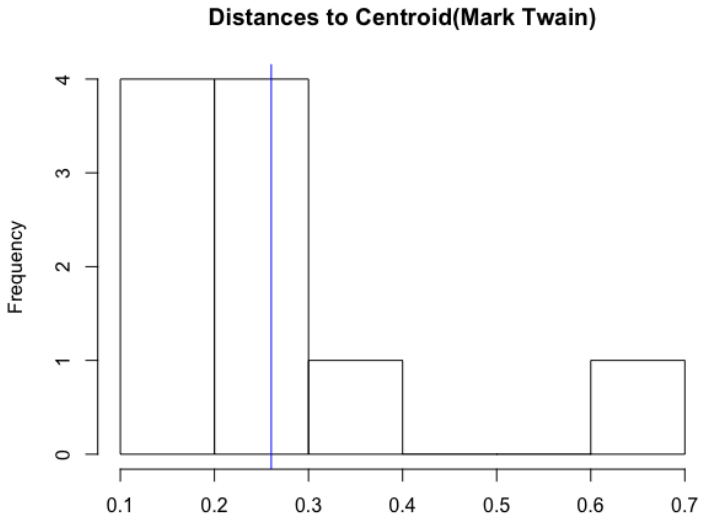


Figure 6: Histogram of Euclidean distances of Twain's novels to own centroid in Adjustment 1(omit special noun).

We can see that distribution of Mark Twain's novels to his centroid is still skewed left as in original model, but with a mean distance 0.2604 smaller than the original mean of 0.3775. This also indicates that removing the special nouns makes Twain's cloud of novels closer to the centroid. The new standard deviation is 0.1400, which is slightly less than 0.1421 in the original model. This is a potential indicator that special nouns in Twain's literature is not a significant source of his variations in writing style, which is consistent with original model result.

To better compare the two authors clusters, a comparative box-plot of the distances and cross distances is shown below.

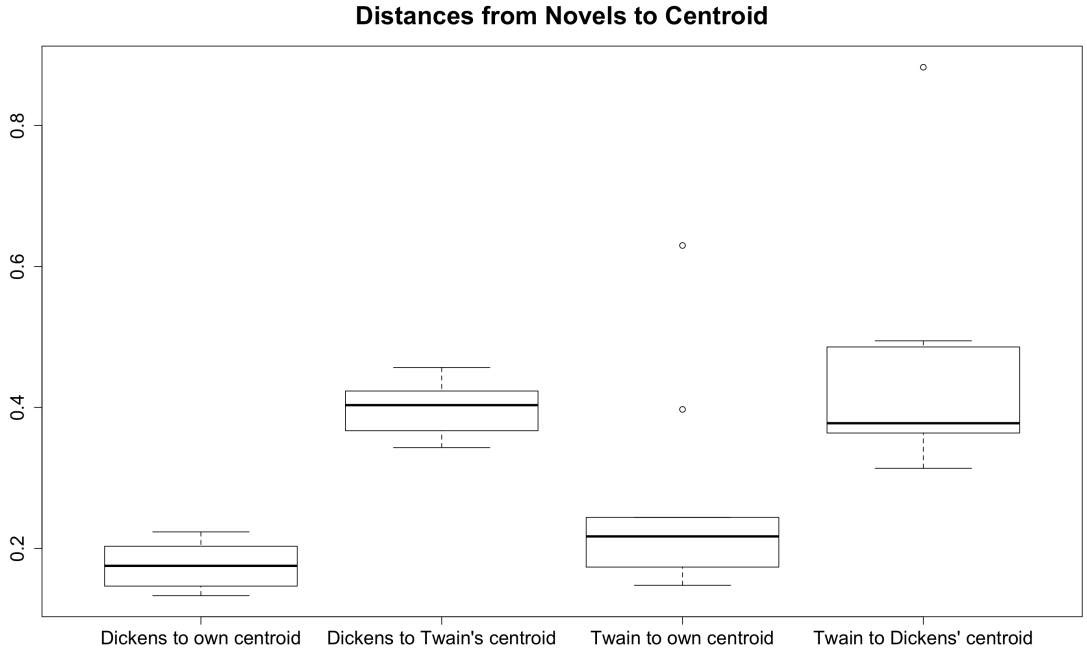


Figure 7: Comparative box-plot of novel's distances to own author's centroid and their cross distances to the other author's centroid in Adjustment 1(omit special noun).

After ommiting special nouns, the box-plots of each author's books are much less overlapped,

which indicates that ignoring proper nouns better separates the two authors works and now all books are closer to the center of their writers. The two boxes of cross distances are much higher than the two boxes of distances to author's own centroid, with a couple exceptions. This suggest that the two authors novels are very likely to form their own cluster that are separated from the other author. Moreover, all four boxes are smaller than those in the original model, which implies that removing the special nouns create less variations and make the texts closer to each other. The changes allow us to be able to avoid meaningless effects caused by special nouns on transition matrices, and thus reveal two dense clouds that are relatively away from each other.

Two outlier distances in Mark Twain novels to own centroid, which in decreasing distance order are *Adventures of Huckleberry Finn* and *The Prince and the Pauper*. The existence of *Adventures of Huckleberry Finn* as an outlier is consistent with our conclusion in original model results, where we found out the entry with largest difference is not one of the repeated character names. We look up the novel *Adventures of Huckleberry Finn* and find out that one special characteristic in this novel is Mark Twain's heavy use of vernacular. This novel is important in the history of American literature since it was the first time introducing dialect in public reading. In order to show the background of a character, Huck, Mark Twain used a number of dialects in his words, which include: "the Missouri "Negro" dialect, the backwoods Southwestern dialect, the Pike County dialect, and a few others". Moreover, contractions are also frequently used in this novel, such as ain't, they'd, you've, wouldn't, they'll, it', I'll, etc. At that time, using contractions in books was not a popular writing style.[13] We also find many use of the phrase "wuz" and "uz" in the writing, which is rarely seem in English writing. Thus, the large distance between *Adventures of Huckleberry Finn* and the centroid can be explained by the specially use of vernacular. *The Prince and the Pauper* was also a outlier but less far away from the centroid might be due to the fact that it is written in 1881, which is recent to 1884 when *Adventures of Huckleberry Finn* first published, so they might have similar use of language.

In sum, by omitting proper noun better the two authors novel form a denser cluster with less overlap than when proper noun is include. *Adventures of Huckleberry Finn* has distance significantly far away from Twain's profile, and it reflectse distinct writing style of using special dialect for this novel.

## 5.2 Two-letter by Two-letter Transition Matrix

We suspect that one letter to one letter sequence might not carry enough syntactic information of individual author, so the difference between two matrices is relatively small. With a goal to capture more syntactic elements of the author, we now build our transition matrix based on the probability of two-letter sequence following by another two-letter sequence. In this case, we will have $27^2 = 729$ states indicating the 2-character permutations from the 26 English letters and a space character. Matrix generated by this method is referred as the second order transition matrix.

The second order transition matrix is a $\mathbb{R}^{729\times 729}$ matrix with states of the matrix being a list of ordered tuple produce by the cross product from the states of thec"$\mathrm{P}_{tm27}$" matrix. For example, the first 4 states in the markov chain is: $aa$, $ab$, $ac$, $ad$... Here is how the "$\mathrm{P}_{2ndtm}$" looks like when written out:

$$\mathbf{P}_{\text{2ndtm}} = \begin{bmatrix} \mathbb{P}(aa|aa) & \mathbb{P}(ab|aa) & \dots & \mathbb{P}(*z|aa) & \mathbb{P}(**|aa) \\ \mathbb{P}(aa|ab) & \mathbb{P}(ab|ab) & \dots & \mathbb{P}(*z|ab) & \mathbb{P}(**|ab) \\ \vdots & & & & \\ \mathbb{P}(aa|*z) & \mathbb{P}(ab|*z) & \dots & \mathbb{P}(*z|*z) & \mathbb{P}(**|*z) \\ \mathbb{P}(aa|**) & \mathbb{P}(ab|**) & \dots & \mathbb{P}(*z|**) & \mathbb{P}(**|**) \end{bmatrix}$$

For this type of transition matrix, there could be some rows that are entirely zeros if a given 2 characters sequence is never observed in the written text from the author. This is absolutely true for the last row of the $\mathbf{P}_{\text{2ndtm}}$ due to the way text are pre-processed. Two consecutive spaces never appear in the text after pre-processing. While having a row consists of entirely zeros breaks the definition of a transition matrix, it keeps the size of transition matrices consistent for all the written works from different authors, making the comparison using matrix norm possible for all authors.

Then, we combine the Adjustment 5.1, which is to ignore the proper nouns in the novels with this new method of generating transition matrices to analyze two clouds of the authors. The following tables shows the distances of two authors' novels to their own centroid and cross distances to

other author's centroid with the new adjustment.

| **Charles Dickens' Novels** | **Distance to Centoid** | **Distance to Twain's Centoid** |
|---|---|---|
| A Tale of Two Cities | 8.8439 | 9.9892 |
| Bleak House | 7.8239 | 10.0659 |
| David Copperfield | 7.5429 | 9.8883 |
| Dombey and Son | 7.5891 | 9.9286 |
| Great Expectations | 8.3164 | 9.9106 |
| Little Dorrit | 7.4535 | 9.6967 |
| Oliver Twist | 8.4592 | 9.9044 |
| Our mutual friend | 7.8250 | 9.6872 |
| The Life And Adventures Of Nicholas Nickleby | 7.3731 | 10.0668 |
| The Pickwick papers | 7.4124 | 10.3332 |

| **Mark Twain's Novels** | **Distance to Centoid** | **Distance to Dickens' Centoid** |
|---|---|---|
| A Connecticut Yankee in King Arthur's Court | 8.1742 | 10.6459 |
| Adventures of Huckleberry Finn | 10.5916 | 12.7403 |
| Personal Recollections of Joan of Arc | 8.9146 | 10.3789 |
| Life on the Mississippi | 7.8095 | 10.4310 |
| The Adventures of Tom Sawyer | 8.8183 | 10.7295 |
| The American Claimant | 9.0189 | 10.7002 |
| The Gilded Age: A Tale of Today | 8.2194 | 9.8532 |
| The Mysterious Stranger | 9.9974 | 11.0395 |
| The Prince and the Pauper | 9.5853 | 10.4050 |
| The Tragedy of Pudd'nhead Wilson | 9.4333 | 10.8971 |

The comparative box-plot of the distances is shown below.

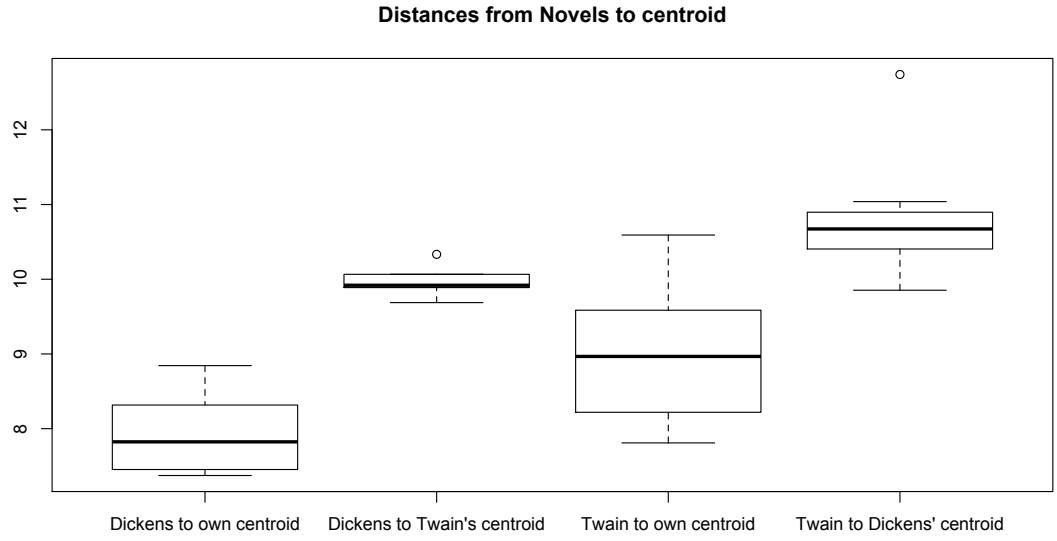


Figure 8: Comparative box-plot of novel's distances to own author's centroid and their cross distances to the other author's centroid in Adjustment 2(second order matrix omitting special noun).

The first noticeable point is that the distances from each author's novels to his centroid is significantly greater than the distances in the previous models, the Charles Dickens' and Mark Twain's mean distances to their own centroids are 7.8639 and 9.0562 respectively. This is because we summing differences between two $\mathbb{R}^{729\times729}$ matrices, which is has much more entries than the $\mathbb{R}^{27\times27}$ matrices in the previous models. Two letter by two letter sequence is also more restrictive on the letter sequence and thus captures more syntactic information from writings, which could potentially reflect the different writing style better.

Secondly, Figure 8 shows that the distances between each author's novels to his own centroid is smaller than the distances between his novels to the other author's centroid. The distance between two centroids in this model is 9.8827, which is larger than the average distances from Charles Dickens' and Mark Twain's novels to their own centroids, which are 7.8639 and 9.0562. This indicates that each author are likely to have his own cluster that are separated from the

other author. The 2 box-plots of cross distances being higher than distances to own centroid with little overlap also support this conclusion.

A minor fact to notice is that all Twain's novels are now closer to his own centroid than to Dickens', even *The Prince and the Pauper* which was ambiguous in previous models. The outlier of *Adventures of Huckleberry Finn* is still present, and it is shown to be significantly far away from Dickens' centroid. This is also consistent with Twain's use of dialect and idioms, which is not an characteristic of Dickens' writing.

## 5.3 Other Distance Measures

In all of the models above, Euclidean distance(i.e. 2-Norm of matrix) are used as the measure of distance between two matrices. In this section, we want to investigate how author's clusters will change under different distance measure: matrix one norm and entry-wise one norm.

### 5.3.1 One Norm

To use matrix one norm, we first calculate the difference matrix of probability of a two letter sequence following by another two letter sequence. We then sum up the probability difference in absolute value of each column, and use the maximum column sum as the distance measure. The formula to calculate one norm is

$$\text{dis}(A, B) := ||A - B||_1 = \max_{1 \le j \le n} \sum_{i=1}^{n} |a_{ij} - b_{ij}|$$

where $A, B$ are two second-order transition matrices with $n = 729$ states under comparisons, and $a, b$ are entries of matrices.

The intuition behind using one norm is that if there is a significant difference in a specific sequence of words used, it will be reflected in distance without being weighted out by other entries, especially when differences are less than 1. For example, if one novel use excessive amount of the word "quad" while the other use none, we should expect transitioning entry of "qu" to "ad" to be significantly large in the difference matrix, which will very likely to make the column with that entry to have maximum sum. Following adjustment 2, we calculate the one norm distances

using the two letter by two letter transition matrix for two authors, and results are shown in tables below.

| **Charles Dickens' Novels** | **Distance to Centoid** | **Distance to Twain's Centoid** |
|---|---|---|
| A Tale of Two Cities | 7.1986 | 9.0822 |
| Bleak House | 6.7416 | 9.1862 |
| David Copperfield | 6.3498 | 10.2326 |
| Dombey and Son | 6.4366 | 8.8756 |
| Great Expectations | 7.4247 | 8.8523 |
| Little Dorrit | 5.9755 | 8.3802 |
| Oliver Twist | 7.6053 | 10.1967 |
| Our mutual friend | 5.8113 | 8.5028 |
| The Life And Adventures Of Nicholas Nickleby | 6.5210 | 10.5184 |
| The Pickwick papers | 6.1336 | 10.7013 |

| **Mark Twain's Novels** | **Distance to Centoid** | **Distance to Dickens' Centoid** |
|---|---|---|
| A Connecticut Yankee in King Arthur's Court | 6.1060 | 10.3811 |
| Adventures of Huckleberry Finn | 9.9373 | 13.4310 |
| Personal Recollections of Joan of Arc | 7.7319 | 9.0927 |
| Life on the Mississippi | 5.2345 | 9.5424 |
| The Adventures of Tom Sawyer | 8.2151 | 10.4789 |
| The American Claimant | 7.9103 | 11.1265 |
| The Gilded Age: A Tale of Today | 6.3769 | 9.5817 |
| The Mysterious Stranger | 8.1830 | 10.6664 |
| The Prince and the Pauper | 8.8797 | 11.0327 |
| The Tragedy of Pudd'nhead Wilson | 8.5975 | 11.3223 |

All novels are closer to their author's centroid and further away from another author's centroid, and this is consistent throughout all the models so far. This again is an evidence that Charles Dickens and Mark Twain's novels form their own clusters. The comparative box-plots of one norm distances is shown below.

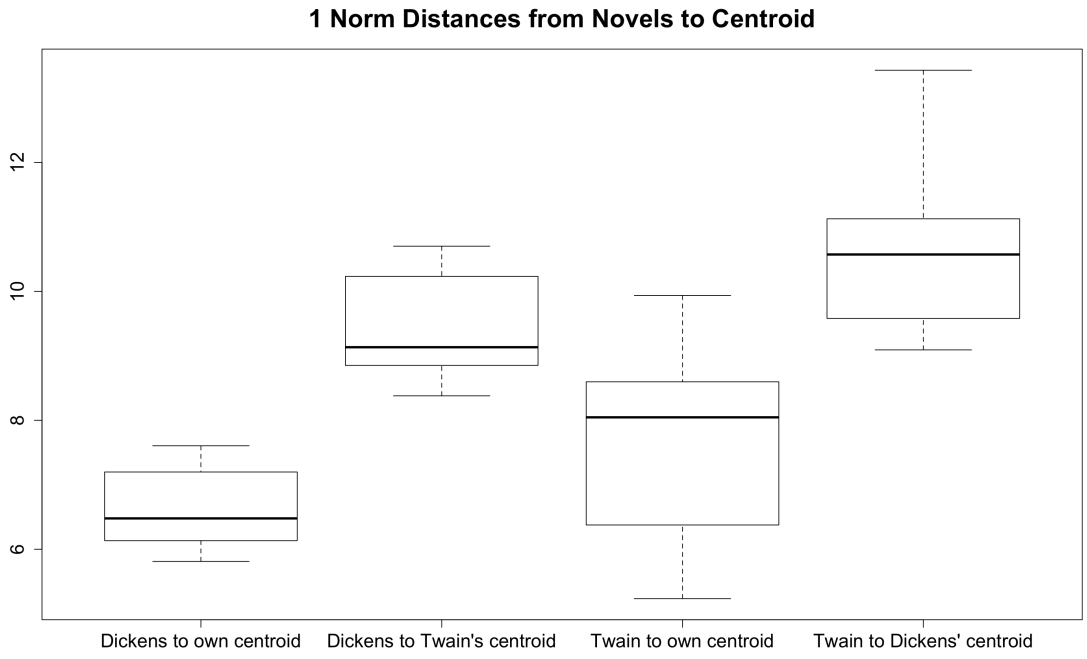


Figure 9: Comparative box-plot of novel's one norm distances and cross distances in Adjustment 3.1(second order matrix omitting special noun).

Similar to previous model, both author have their writings closer to their own centroid. The cross distances box-plots being higher than distances to own centroid allow us to make the conclusion that two clusters constructed by the texts from different authors are separate. Moreover, we observe the distances in this model have bigger spreads than previous models. This is because of the feature of one norm, which uses the absolute value while computing the results. As a result of increased spread, although we still have large distances for novels at the upper tail, the extreme values are no longer considered as outliers.

### 5.3.2 Entry-wise Matrix One Norm

To calculate entry-wise matrix one norm, we use the formula

$$\text{dis}(A, B) := ||vec(A - B)||_1 = \sum_{i=1}^{n} \sum_{j=1}^{n} |a_{ij} - b_{ij}|$$

The intuition of using this "entry-wise" one norm is that most of the probabilities are less than 1 and so are their differences in absolute value. Using two norm in previous models, we square individual differences which makes then even smaller. Distances calculated by entry-wise one norm and the comparative box-plot are shown below.

| **Charles Dickens' Novels** | **Distance to Centoid** | **Distance to Twain's Centoid** |
|---|---|---|
| A Tale of Two Cities | 272.7793 | 341.2822 |
| Bleak House | 213.3771 | 334.3947 |
| David Copperfield | 202.8842 | 335.7517 |
| Dombey and Son | 200.9683 | 339.2411 |
| Great Expectations | 247.7411 | 335.9497 |
| Little Dorrit | 198.0833 | 323.7918 |
| Oliver Twist | 258.7954 | 348.5776 |
| Our mutual friend | 209.2619 | 326.3303 |
| The Life And Adventures Of Nicholas Nickleby | 199.5594 | 343.6294 |
| The Pickwick papers | 223.1247 | 363.6674 |

| **Mark Twain's Novels** | **Distance to Centoid** | **Distance to Dickens' Centoid** |
|---|---|---|
| A Connecticut Yankee in King Arthur's Court | 250.8516 | 373.8739 |
| Adventures of Huckleberry Finn | 402.8773 | 511.7869 |
| Personal Recollections of Joan of Arc | 276.2573 | 366.0022 |
| Life on the Mississippi | 251.0494 | 366.0901 |
| The Adventures of Tom Sawyer | 304.5456 | 390.6068 |
| The American Claimant | 306.2761 | 381.4775 |
| The Gilded Age: A Tale of Today | 253.7137 | 324.3947 |
| The Mysterious Stranger | 346.5795 | 406.7008 |
| The Prince and the Pauper | 329.9942 | 375.6150 |
| The Tragedy of Pudd'nhead Wilson | 319.8175 | 396.9573 |

Using entry-wise one norm, the distance measures are significantly larger than any of the distance measure in previous models. All novels are closer to own author's centroid than others, which

is consistent with previous model. A comparative box-plot of distances is shown in the following figure.

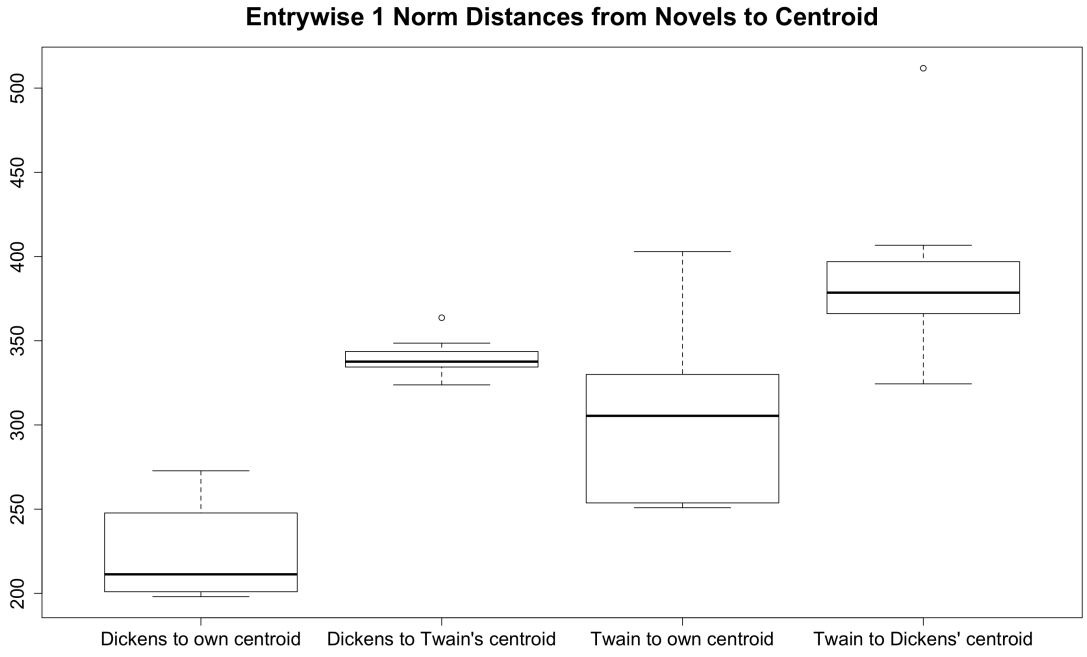


Figure 10: Comparative box-plot of novel's entry-wise one norm distances and cross distances in Adjustment 3.2(second order matrix omitting special noun).

Similar with the previous models, the box-plots supports that there are two separate clouds, since the distances of texts to their authors' profile is much less than the cross distances. We also observe that the cross distances are generally higher than distances from novels to own centroid while having a much smaller spread, especially Dickens' novels to Twain's centroid. The smaller spread of cross distances is a result of increasing magnitude of distance value. Under entry-wise one norm formula, all distances to centroid are in larger value and so is the distance between two centroids. Imagine in a high dimensional space($\mathbb{R}^{729\times 729}$), it would be two clouds of matrices closely located around a centroid but clouds are far away from each other. The distances within each cloud are relatively small while the two clouds are far away, so the cross distances from one cloud to the other's centroid be large but with small spread.

## 5.4 Log Scaled Transition Matrix Accounting for Uncommon Sequence

We are aware that some sequence of English letters are very rarely used, such as "az" or "xe", or even the case of "wuz" mentioned earlier in *Adventures of Huckleberry Finn*. We think the frequencies of using these rare words might be an important factor of an author's writing style. In previous model, we fail to capture this factor since the probability of these transitions are very small. As a result, it does not carry much weight when we calculated the distance between two matrices(i.e. norm of difference matrix).

In order to make our matrices having an emphasis on the use of rare words, we will take the *log* of the count of one character following another character before calculating the probability. Rarely used sequence with smaller counts would now weight a little more than when without taking *log*. We are aware that there are sequences that might not be used at all by the authors or do not form a word in English, which would have a count of 0. In that case, we keep the count of those sequences as 0 instead of taking *log*, since $log(0)$ is undefined. We generate the log scaled second order transition matrices for each author(proper noun omitted), and the Euclidean distances are shown in the following table.

| **Charles Dickens' Novels** | **Distance to Centoid** | **Distance to Twain's Centoid** |
|---|---|---|
| A Tale of Two Cities | 9.5910 | 9.5923 |
| Bleak House | 8.1064 | 9.3101 |
| David Copperfield | 8.4421 | 9.3244 |
| Dombey and Son | 8.0977 | 8.7139 |
| Great Expectations | 9.2536 | 9.5502 |
| Little Dorrit | 8.5623 | 8.8847 |
| Oliver Twist | 9.3095 | 9.4043 |
| Our mutual friend | 8.2005 | 9.0116 |
| The Life And Adventures Of Nicholas Nickleby | 8.1721 | 9.0656 |
| The Pickwick papers | 7.9420 | 9.2946 |

| Mark Twain's Novels | Distance to Centoid | Distance to Dickens' Centoid |
|---|---|---|
| A Connecticut Yankee in King Arthur's Court | 8.7710 | 10.1071 |
| Adventures of Huckleberry Finn | 10.1208 | 11.5048 |
| Personal Recollections of Joan of Arc | 8.7629 | 10.0970 |
| Life on the Mississippi | 8.0998 | 9.7903 |
| The Adventures of Tom Sawyer | 9.5983 | 10.8898 |
| The American Claimant | 9.8571 | 11.0426 |
| The Gilded Age: A Tale of Today | 8.6106 | 9.8611 |
| The Mysterious Stranger | 10.6961 | 11.2969 |
| The Prince and the Pauper | 9.7923 | 10.6779 |
| The Tragedy of Pudd'nhead Wilson | 9.9615 | 10.8796 |

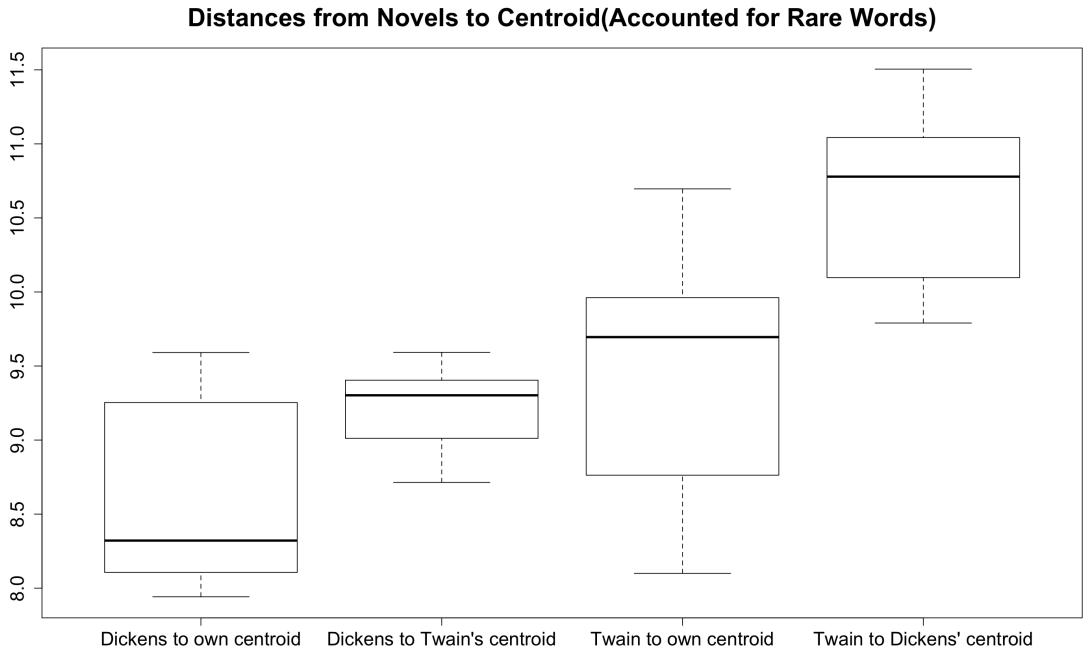


Figure 11: Comparative box-plot of novel's Euclidean distances and cross distances in Adjustment 4 (log scaled second order matrix omitting special noun).

From figure 11, we can see that the pair box-plot of Dickens novels to itself and the cross distance to Twain's centroid is very overlapped, meaning we cannot conclude that Dickens have a cluster for his profile even though the average cross distance is much higher than average distance to his own centroid. For Mark Twain, there is little overlap on the two box-plot on the right. Cross distances of Twain's novels to Dickens' centroid are shown to be greater than distances to his own centroid, meaning Twain is very likely to have his small cluster of literature relatively far from Dicken's profile.

Comparing the box-plot to those from previous adjusted models, log scaling the counts to emphasize rarely used sequence does not seem to be very effective in defining clouds of authors that are separated, especially for Dickens novels in this case. A possible reason could be that those rarely used words are just not an important factor of the author's writing style. For example, if the word "za" is used a couple times in one of Dickens' writing, it does not necessarily imply using "za" should be consider as a characteristic of his writing. Thus, it is not always reasonable to emphasize the use of rarely used sequence. It might work for author who are known for the use of uncommon diction to generate better clusters with smaller "radius" and far from other cluster. In our case, rarely used sequence does not seem to play an important role in Dickens and Twain's writing.

# 6 Conclusion

After the above analysis on transition matrices generated from two authors novels, we can conclude that using Markov transition matrix can captures syntactic elements or writing style of authors. We do have evidence showing that transition matrices generated from novels of Charles Dickens and Mark Twain form two distinct clusters as we use different distance measure and impose more restrictive assumption such as omitting proper noun or using second order transition matrix. This is consistent with real world phenomenon, because the two authors we have chosen are generally known to have different writing style.

When defining transition matrix using probability of letter by letter sequence and with proper noun intact, we cannot conclude two authors have distinct cluster separated from each other since

there is overlap in Twain's cross distances to other centroid with distances to own centroid. After omitting proper noun, overlap is much more reduced which implies two denser cluster of authors' novel relatively away from each other.

As we move to second order transition matrix define by two letter by two letter probabilities, distance values are significantly greater. No overlap between cross distances and distances to own centroid is observed and the spread of distances seem to be reduced, which further supports the conclusion on two separate clouds.

When we use one norm and entry-wise one norm measure used for distance comparison, same conclusion on two separate cluster still holds. However, we observe the spread of one norm cross distances being slighter larger than Euclidean distance, and the spread of entry-wise one norm being smaller than Euclidean distance. This imply that using entry-wise one norm seem to define two distinct cluster better for the two chosen authors.

Using log scaled transition matrices for the intention to emphasize use of rare words does not support previous conclusion of separate clusters, since there is overlapped in Dickens' cross distances to other centroid and distances to own centroid. We believe that emphasizing rare words might not be reasonable to use for the two chosen author, since it is not one of the characteristic of their writing style.

An observation consistent though out all models is that the texts from Charles Dickens are always closer to his centroid with smaller spread than those of Mark Twain. This imply that Charles Dickens had a more stable writing style for his literature than Mark Twain. This can also be supported by the fact that *Adventures of Huckleberry Finn* occur as an outlier for several models. As mentioned previously, Twain's frequent use of special dialect and idioms is very distinct from both Dickens' and his own writings, which contribute to the large variation in Twain's novels. This specialty is not captured the the models using one norm distance measure and log scaled matrix.

There are many aspect of the model we can consider for future modifications. If the author have a

large profile of literature, aggregating all writings could be challenging. We can change the centroid measure to the average of individual transition matrices of author's writing. When comparing two authors distance to each other using just one value, we can use the distance between two closest matrices from two author instead of distance between centroid to give us a sense of how close can the two potential clouds be.

# References


[1] *Charles Dickens*. Charles Dickens - The Greatest Literature of All Time, www.editoreric.com/greatlit/authors/Dickens.html

[2] Dickens, Charles. *A Tale of Two Cities*. Project Gutenberg. N.p. N.d. EPUB. Downloaded Feb 25, 2020. https://www.gutenberg.org/ebooks/98

[3] Dickens, Charles. *Bleak House*. Project Gutenberg. N.p. N.d. EPUB. Downloaded Feb 25, 2020. http://www.gutenberg.org/ebooks/1023

[4] Dickens, Charles. *David Copperfield*. Project Gutenberg. N.p. N.d. EPUB. Downloaded Feb 25, 2020. https://www.gutenberg.org/ebooks/766

[5] Dickens, Charles. *Dombey and Son*. Project Gutenberg. N.p. N.d. EPUB. Downloaded Feb 25, 2020. https://www.gutenberg.org/ebooks/821

[6] Dickens, Charles. *Great Expectations*. Project Gutenberg. N.p. N.d. EPUB. Downloaded Feb 25, 2020. https://www.gutenberg.org/ebooks/1400

[7] Dickens, Charles. *Little Dorrit*. Project Gutenberg. N.p. N.d. EPUB. Downloaded Feb 25, 2020. http://www.gutenberg.org/ebooks/963

[8] Dickens, Charles. *Oliver Twist*. Project Gutenberg. N.p. N.d. EPUB. Downloaded Feb 25, 2020. https://www.gutenberg.org/ebooks/730

[9] Dickens, Charles. *Our mutual friend*. Project Gutenberg. N.p. N.d. EPUB. Downloaded Feb 25, 2020. http://www.gutenberg.org/ebooks/883

[10] Dickens, Charles. *The Life And Adventures Of Nicholas Nickleby*. Project Gutenberg. N.p. N.d. EPUB. Downloaded Feb 25, 2020. http://www.gutenberg.org/ebooks/967

[11] Dickens, Charles. *The Pickwick papers*. Project Gutenberg. N.p. N.d. EPUB. Downloaded Feb 25, 2020.
http://www.gutenberg.org/ebooks/580

[12] Freeman Daniel, *Predicting the author of Twitter posts with Markov chain analysis.*. (2017). Honors Theses.
https://scholar.utc.edu/cgi/viewcontent.cgi?article=1126&context=honors-theses

[13] *Huck's Vernacular throughout The Adventures of Huckleberry Finn.*. Living Language, 29 Apr. 2009,
livingwithlanguage.wordpress.com/2009/04/03/huck's-vernacular-throughout-the-adventures-of-huckle

[14] Jagtap, Avinash S, and Jaya L Limbore. *Authorship Attribution Using Markov Chain.*. International Journal of Latest Trends in Engineering and Technology. Vol.(9) Issue(3), pp.119-125.
http://dx.doi.org/10.21172/1.93.21

[15] Pavels Osipovs, Andrejs Rinkevics, Galina Kuleshova and Arkady Borisov. *Markov Chains in the Task of Author's Writing Style Profile Construction.*. Information Technology and Management Science Volume 17 Issue 1 (2014). Information Technology and Management Science. Sciendo, March 11, 2015.
https://doi.org/10.1515/itms-2014-0018

[16] Sanderson, Conrad, and Simon Guenter. *Short Text Authorship Attribution via Sequence Kernels, Markov Chains and Author Unmasking: An Investigation.*. ACL Anthology. Empirical Methods in Natural Language Processing 2006, pages 482–491, July 2006.
https://www.aclweb.org/anthology/W06-1657

[17] Twain, Mark. *A Connecticut Yankee in King Arthur's Court*. Project Gutenberg. N.p. N.d. EPUB. Downloaded Feb 25, 2020.
https://www.gutenberg.org/ebooks/86

[18] Twain, Mark. *Adventures of Huckleberry Finn*. Project Gutenberg. N.p. N.d. EPUB. Downloaded Feb 25, 2020.
https://www.gutenberg.org/ebooks/76

[19] Twain, Mark. *Life on the Mississippi*. Project Gutenberg. N.p. N.d. EPUB. Downloaded Feb 25, 2020.
https://www.gutenberg.org/ebooks/245

[20] Twain, Mark. *Personal Recollections of Joan of Arc Vol.1*. Project Gutenberg. N.p. N.d. EPUB. Downloaded Feb 25, 2020.
https://www.gutenberg.org/ebooks/2874

[21] Twain, Mark. *Personal Recollections of Joan of Arc Vol.2*. Project Gutenberg. N.p. N.d. EPUB. Downloaded Feb 25, 2020.
https://www.gutenberg.org/ebooks/2875

[22] Twain, Mark. *The Adventures of Tom Sawyer*. Project Gutenberg. N.p. N.d. EPUB. Downloaded Feb 25, 2020.
https://www.gutenberg.org/ebooks/74

[23] Twain, Mark. *The American Claimant*. Project Gutenberg. N.p. N.d. EPUB. Downloaded Feb 25, 2020.
https://www.gutenberg.org/ebooks/3179

[24] Twain, Mark. *The Gilded Age: A Tale of Today*. Project Gutenberg. N.p. N.d. EPUB. Downloaded Feb 25, 2020.
https://www.gutenberg.org/ebooks/3178

[25] Twain, Mark. *The Mysterious Stranger*. Project Gutenberg. N.p. N.d. EPUB. Downloaded Feb 25, 2020.
https://www.gutenberg.org/ebooks/3186

[26] Twain, Mark. *The Prince and the Pauper*. Project Gutenberg. N.p. N.d. EPUB. Downloaded Feb 25, 2020.
https://www.gutenberg.org/ebooks/1837

[27] Twain, Mark. *The Tragedy of Pudd'nhead Wilson*. Project Gutenberg. N.p. N.d. EPUB. Downloaded Feb 25, 2020.
https://www.gutenberg.org/ebooks/102

[28] *Writing Style - Mark Twain Author Study.*. Google Sites, sites.google.com/site/marktwainauthorstudy3/home/writing-style

# Appendix

## Code to print the result

```
from core import *


AUTHORS_FOLDER = "data"
from os import listdir
from os.path import isfile, isdir
from typing import Callable, Type

class Authors:
    """
        This class models a bunch of authors.
        self.__AuthorList:
            This is an array of authors created from the root folder, it contains
            all instance of the author, with generating function and a
            ignore case option specified by the self.prepare() function.
    """
    def __init__(self, MatrixGeneratingFunction: Callable,
                 IgnoreSpecialNoun:bool=False):
        self.__AuthorList = None
        self.prepare(MatrixGeneratingFunction, IgnoreSpecialNoun)
        self.__IgnoreSpecialNoun = IgnoreSpecialNoun

    def prepare(self, MatrixGeneratingFxn: Callable, IgnoreSpecialNoun: bool):
        """
            This function set up all the authors in the folders.
            * It buffers all the text for each of the authors.
            * Specifies Matrix generationfunction and Ignore special nouns option.
            * Please repeatedly call the function cause that will me it slower than
    necessary.
            * This function must be called in the __init__ to establish the field
            of the class, such as the transition matrix generating function, ignore
    case option, and the
            author list.

        :param MatrixGeneratingFxn:
```

```
            This is the matrix function you want to use
            to generate the transition matrices of the authors.
        :param IgnoreSpecialNoun:
            This is set to true if you want to remove
            all special nouns in the text.
        :return:
            None
        """
        AuthorsFolders = []
        for d in listdir(AUTHORS_FOLDER):
            d = AUTHORS_FOLDER + "/" + d
            if isdir(d):
                AuthorsFolders.append(d)
        assert len(AuthorsFolders) > 0, f"Author's Folder is empty: AUTHORS_fOLDER:{AUTHORS_FOLDER}"

        self.__AuthorList = []
        self.__IgnoreSpecialNoun = IgnoreSpecialNoun
        self.__GeneratingFunction = MatrixGeneratingFxn
        for AuthorDir in AuthorsFolders:
            self.__AuthorList.append(Author(AuthorDir, MatrixGeneratingFxn, IgnoreSpecialNoun=IgnoreSpecialNoun))

    def produce_results(self,
                CentroidType:Type[CentroidOption]=CentroidOption.AggregateMatrix,
                Mmtric:Type[MatrixMetric]=MatrixMetric.TwoNorm,
                Ametric:Type[AuthorMetric]=AuthorMetric.CentroidDis,):
        """
            This function takes a series of options on metric of the authors and the matrices and
            centroid type.
            * I will return a string, which is a verbal description of all the
            author's works relative to each other.
        :param CentroidType:
            An instance of the class: CentroidType
        :param Mmtric:
            An instance of the class: MatrixMetric
        :param Ametric:
            An instance of the class Ametric.
```

```
        :return:
            A string that is a report of all the works of the authors.
        """
        # Specify all the metrics for all the authors
        Author.CentroidType = CentroidType
        Author.MMetricType = Mmtric
        Author.AMetricType = Ametric
        NameList = [Author.name() for Author in self.__AuthorList]
        MaxNameLen = max(len(n) for n in NameList)
        NameList = [Name.ljust(MaxNameLen) for Name in NameList]
        Res = f"||{'|'.join(NameList)}||\n"

        DistanceTriangle = []
        for I, EachAuthor in enumerate(self.__AuthorList):
            Info1, Info2 = EachAuthor.author_cloud()
            Row = [Info1[0]]
            for J in range(I + 1, len(self.__AuthorList)):
                Row.append(dis_between_authors(EachAuthor, self.__AuthorList[J]))
            DistanceTriangle.append(Row)

        for I, EachRow in enumerate(DistanceTriangle):
            EachRow = [f"{{:<{MaxNameLen + 1}.4f}}".format(Value) for Value in
    EachRow]
            Res += "  " + " "*I*(MaxNameLen + 1) + "".join(EachRow) + "\n"

        Res += "".join(str(Author) for Author in self.__AuthorList)
        Res += "="*50 + "\n"
        return Res


    def __repr__(self):
        s = ""
        for author in self.__AuthorList:
            s += str(author)
        N = len(self.__AuthorList)
        A = self.__AuthorList
        s += "--------------ALL AUTHORS TOGETHER----------------------\n"
        s += "Author's Distance matrix with triangular part printed:\n"
        s += " | " + " | ".join(self.__AuthorNameList) + " |\n"
```

```
        for I in range(N - 1):
            for J in range(I + 1, N):
                s += '{:10.4f}'.format(dis_between_authors(A[I], A[J]))
            s += "\n"
        s += f"The matrix metric is: {Author.MMetricType}\n"
        s += f"The Author metric is: {Author.AMetricType}\n"
        s += "All special nouns are IGNORED\n" if self.__IgnoreSpecialNoun else ""
        s += "======================================================"
        return s


def print_experiment(
        MatrixGeneratingFxn: Callable,
        IgnoreSpecialNouns: bool,
        CentroidType: CentroidOption,
        MatrixMetric: MatrixMetric,
        AuthorMetric: AuthorMetric
):
    """
        Given all the options, this function will print out everything and run the
    experiment based on that option.
    :param MatrixGeneratingFxn:
    :param IgnoreSpecialNouns:
    :param CentroidType:
    :param MatrixMetric:
    :param AuthorMetric:
    :return:
    """
    TheInstance = Authors(MatrixGeneratingFunction=MatrixGeneratingFxn,
                          IgnoreSpecialNoun = IgnoreSpecialNouns)
    print(TheInstance.produce_results(CentroidType=CentroidType,
                Mmtric=MatrixMetric,
                Ametric=AuthorMetric))
    return


def print_cross_compare_experiement(Author1Directory:str,
                                    Author2Directory:str,
                                    MatrixGeneratingFxn: Callable = get_tm27,
```

```
                                    IgnoreSpecialNouns: bool = True,
                                    MatrixMetric: MatrixMetric = MatrixMetric.
TwoNorm):
    """
        This function takes in 2 instances of author and cross compare then and then
 print all
        the report on each of the individual works, average distances to the
opposite centroid.
    :param Author1Directory:
        A string representing the folder that contains all the text files of the
author
    :param Author2Directory:
        A string, representing another author.
    :return:
        None
    """
    assert Author2Directory != Author1Directory, "You should not cross compare the
same author. "

    Author.MMetricType = MatrixMetric

    Author_A, Author_B= Author(Author1Directory, MatrixGeneratingFxn,
IgnoreSpecialNouns), \
                      Author(Author2Directory, MatrixGeneratingFxn,
IgnoreSpecialNouns)
    Author1, Author2 = Author_A.cross_distance_stats(Author_B)
    def print_author(A, B):
        for K, V in A[0].items():
            print(f"{K}: {V}")
        print("-"*40)
        print(f"The average distance is: {A[1]}, The Standard Deviation is: {A[2]}")

    print("Here is a list of distances when we compare each individuals works of "
         f"{Author_A.name()} to {Author_B.name()}'s centroid")
    print("+"*40)
    print_author(Author1, Author2)
    print("Here is a list of distances when we compare each individuals works of "
          f"{Author_B.name()} to {Author_A.name()}'s centroid")
    print("+"*40)
```

```
    print_author(Author2, Author1)
    print("="*40)
    print("Here is a list the cloud info for both author: ")
    print("="*40)
    print(Author_A)
    print(Author_B)
    return


if __name__ == "__main__":
    print_cross_compare_experiement("data/Charles Dickens", "data/Mark Twain")
```

## Code to define functions needed to generate matrices

```
"""
Group 4, this is for the Project 2.


"""
import numpy as np
import matplotlib.pyplot as plt

from string import ascii_letters
from typing import List, Callable, Type
import re
import enum
import math
from os import listdir
from os.path import isfile

__all__ = ["Author", "dis_between_authors", "get_tm27", "get_2ndtm","
    get_2ndlogarithm",
           "CHARLES_DICKENS",
           "MARK_TWAIN", "CentroidOption", "MatrixMetric", "AuthorMetric", "dis"]

# A list of authors' directory:
CHARLES_DICKENS = "data/Charles Dickens"
MARK_TWAIN = "data/Mark Twain"
ENCODING = "utf8"
```

```
PLOT = "plot"

np.set_printoptions(threshold=10, precision=2, linewidth=1000)


def process_text(filepath: str):
    """
    Function reads lines from the file, with the new line character stripped off
    from
    the line.
    :param filepath:
        The file path.
    :return:
        A array of string. Each string is a line in the file.
    """
    with open(filepath, 'r', encoding=ENCODING) as f:
        return [l.strip() for l in f.readlines()]


def trim_line(s:str, IgnoreCapitalzedWord=False):
    if IgnoreCapitalzedWord:
        s = ' '.join([word for word in s.split() if word[0].islower()])
    Res = ""
    for c in s:
        if c == "\'":
            continue
        Res += c if c in ascii_letters else (" " if len(Res) >= 1 and Res[-1] != ' '
     else "")
    return Res.lower()


def get_tm27(lines: List[str], ignoreSpecialNoun=False):
    """
        Function takes the path of a file and returns the transition
        matrix based on the 26 letters in the alphabet,
        The last states of the matrix is space.
        * All spaces are collapsed.
        * All punctuations are ignored.
        * The apostrophe is stripped off from the text.
```

```
    :param lines:
        An array of lines that is in the file.
    :return:
    """
    matrix = np.zeros((27, 27))
    characters = ascii_letters[0:26] + " "
    for line in lines:
        trimmed_line = trim_line(line, IgnoreCapitalzedWord=ignoreSpecialNoun)
        line = list(trimmed_line)
        for i, c2 in enumerate(line[1:]):
            c1 = line[i]
            indx1 = characters.find(c1)
            indx2 = characters.find(c2)
            if (indx1 == -1 or indx2 == -1):
                continue  # Skip non alphabetical characters.
            matrix[indx1, indx2] += 1
    for i in range(27):
        s = np.sum(matrix[i])
        if s > 0:
            matrix[i] /= s
    return matrix


def get_2ndtm(lines: List[str], skipSpecialNoun=False):
    """
        Given the content of the file separated by lines, this function will return the
        26^2 by 26^2 transition matrix.
        * It's a second order transition matrix based on the letters of the alphabet.
        * Spaces will be included as the last states of the matrix.
    :param lines:
        The content of the file represented in the an array of lines.
    :return:
        The np matrix.
    """
    Alphabet = ascii_letters[0:26] + " "
    l = len(Alphabet)
    n = l ** 2
```

```
    npmatrix = np.zeros((n, n))

    def s(letter):
        return Alphabet.find(letter)

    for Line in lines:
        Line = trim_line(Line, IgnoreCapitalzedWord=skipSpecialNoun)
        for I in range(len(Line) - 3):
            i = s(Line[I]) * l + s(Line[I + 1])
            j = s(Line[I + 2]) * l + s(Line[I + 3])
            npmatrix[i, j] += 1

    for i in range(npmatrix.shape[0]):
        s = np.sum(npmatrix[i])
        if s > 0:
            npmatrix[i] /= s
    return npmatrix


def get_2ndlogarithm(lines: List[str], skipSpecialNoun=False):
    """
        Takes the logarithm after counting the frequency,
        THis is a transition matrix that will amplifies the
        occurences of rare sequence compare to traditional sequences.
    :param lines:
        All the lines in the file
    :param skipSpecialNoun:
        All special nouns are ignore if this is set to true.
    :return:
        A numpy matrix
    """
    Alphabet = ascii_letters[0:26] + " "
    l = len(Alphabet)
    n = l ** 2
    npmatrix = np.zeros((n, n))

    def s(letter):
        return Alphabet.find(letter)

```

```
    for Line in lines:
        Line = trim_line(Line, IgnoreCapitalzedWord=skipSpecialNoun)
        for I in range(len(Line) - 3):
            i = s(Line[I]) * l + s(Line[I + 1])
            j = s(Line[I + 2]) * l + s(Line[I + 3])
            npmatrix[i, j] += 1
    for i in range(npmatrix.shape[0]):
        for j in range(npmatrix.shape[1]):
            if npmatrix[i][j] > 0:
                npmatrix[i][j] = np.log(npmatrix[i][j])
    for i in range(npmatrix.shape[0]):
        s = np.sum(npmatrix[i])
        if s > 0:
            npmatrix[i] /= s
    return npmatrix


class CentroidOption(enum.Enum):
    """
    An enum class to represent the options for center of the author cloud.
    """
    AggregateMatrix = 1 # Taking the average among all works of the author.
    AverageMatrix = 2 # Treating all works as one single block of text.


class MatrixMetric(enum.Enum):
    """
    An enum class to represent the options of measuring distance between matrices.
    """
    OneNorm = 1 # Matrix 1 norm
    TwoNorm = 2 # Matrix Euclidean distance
    # WeightedNorm = 3 # Matrix weighted by PD matrix.
    # HighPower2Norm = 4 # Raising matrix to high power and take the 2 norm.
    Vectorized1Norm = 5


class AuthorMetric(enum.Enum):
    """
    THis is a enum class consists of method that can be used to determine the
```

```
    distance
    between the cloud of an author to a transition matrix.
    ! This option only specifies how to determine the distance of an author with a
    transition
    matrix!
    """
    MinimumDis = 1 # The distance between the author and a given transition matrix
    is the minimum distance
    # of any work of the author to that transition matrix.

    AverageDis = 2 # Taking the average distance of the given transition matrix with
     respect to
    # All the matrices of the author.


    CentroidDis = 3 # This metric take the matrix norm on the difference of 2
    centroids of the author.


MM = Type[MatrixMetric]
CO = Type[CentroidOption]
def dis(Matrix1, Matrix2, Metric:MM):
    """
        This function returns the distance between 2 matrices, given
        the type of Metric space and the weights.
    :param Matrix1:
        A numpy matrix.
    :param Matrix2:
        A numpy matrix.
    :param WeightVec1:
        A numpy vector
    :param WeightVec2:
        A numpy vector.
    :return:
        A float.
    """
    if Metric == MatrixMetric.OneNorm:
        return np.linalg.norm(Matrix1 - Matrix2, 1)
    elif Metric == MatrixMetric.TwoNorm:
```

```
211         return np.linalg.norm(Matrix1 - Matrix2)
212     # elif Metric == MatrixMetric.WeightedNorm:
213     #     raise RuntimeError("WeightedNorm not yet implemented. ")
214     # elif Metric == MatrixMetric.HighPower2Norm:
215     #     return np.linalg.norm(Matrix1**10 - Matrix2**10)
216     elif Metric == MatrixMetric.Vectorized1Norm:
217         return np.linalg.norm(np.matrix.ravel(Matrix1) - np.matrix.ravel(Matrix2),
    1)
218     else:
219         raise RuntimeError("Invalid Matrix metric space. ")
220
221
222 class Author:
223     """
224     Files for an author and transitional matrix for the author.
225     * Transitional Matrix classified by each files in the folder
226     * For each file, there will be several transitional matrices for parts of the
    files.
227     * All the matrices will be in the same order as the list of works.
228     """
229     CentroidType = CentroidOption.AggregateMatrix
230     MMetricType = MatrixMetric.TwoNorm
231     AMetricType = AuthorMetric.CentroidDis
232
233     def __init__(self, dir: str, matrixfunction: Callable = get_tm27,
    IgnoreSpecialNoun=False):
234         """
235             Create an instance of an author by specifying:
236                 * A directory containing all text files written by the author.
237         :param dir:
238             The directory of the folder.
239         :param matrixfunction:
240             A function you want to use to genereate the transition matrices for the
    authors.
241         """
242         FilePathList = []
243         for filename in listdir(dir):
244             filepath = dir + "/" + filename
245             if isfile(filepath):
```

```
246                 FilePathList.append(filepath)
247
248         assert len(FilePathList) > 0, f"There is no file under the directory: {dir}"
249         self.__IgnoreSpecialNoun=IgnoreSpecialNoun
250         FilePath2Lines = {}
251         for f in FilePathList:
252             FilePath2Lines[f] = process_text(f)
253         # A map that maps the file path to array of lines containing the content of
    the file.
254         self.__FilePathToLines = FilePath2Lines
255         self.__AuthorName = dir.split("/")[-1]
256         self.__TMFunction = matrixfunction
257         self.__NpMatrices = None  # a list of np matrices for each works of the
    author
258         self.__AggregateMatrix = None  # Instance of transition matrix.
259         self.__AuthorItems = list(self.__FilePathToLines.items())
260
261     def get_fp2lines(self):
262         """
263             A map where the key is the path of the file of an author's work, and the
264             value is a list of string, representing the raw content of the
265             work written by the author.
266             * The text in the line is un-processed.
267         :return:
268         """
269         return self.__FilePathToLines
270
271     def list_of_works(self):
272         return [work.split("/")[-1] for work in list(self.__FilePathToLines.keys())]
273
274     def list_of_works_content(self):
275         return list(self.__FilePathToLines.values())
276
277     def work_matrix_dict(self):
278         """
279             Give a dictionary that maps the name of the works to the
280             transition matrices.
281         :return:
282         """
```

```
283         return dict(zip(self.list_of_works(), self.get_matrices()))
284 
285     def name(self):
286         return self.__AuthorName
287 
288     def get_matrices(self):
289         """
290             This function returns a transition matrix for each work of the author.
291             each work of the author is a file in the author's folder.
292         :return:
293             A list of np matrix.
294         """
295         res = None
296         if self.__NpMatrices is not None:
297             res = self.__NpMatrices
298         else:
299             self.__NpMatrices = [self.__TMFunction(lines, self.__IgnoreSpecialNoun)\
300                                  for lines in self.list_of_works_content()]
301             res = self.__NpMatrices
302         return res
303 
304     def __aggregate_matrix(self):
305         """
306             combine all the lines in the folder into one single work.
307             Then create the transitional matrix for this author,
308             treating all his works as one text.
309             * Results are stored after first time computing it.
310         :return:
311             An npmatrix
312         """
313         if self.__AggregateMatrix is not None:
314             return self.__AggregateMatrix
315         alllines = []
316         for writing in self.list_of_works_content():
317             alllines += writing
318         self.__AggregateMatrix = self.__TMFunction(alllines, self.
    __IgnoreSpecialNoun)
319         return self.__AggregateMatrix
320 
```

```
321     def __average_matrix(self):
322         """
323             This function returns the centroid matrix.
324             * A centroid matrix is the average for each of all the matrices
325             from this authors.
326             * It's not necessarily a transition matrix anymore.
327         :return:
328             A numpy matrix.
329         """
330         TransitionMatrices = self.get_matrices()
331         N = len(TransitionMatrices)
332         CentroidMatrix = np.zeros(TransitionMatrices[0].shape)
333         for TM in TransitionMatrices:
334             CentroidMatrix += TM
335         CentroidMatrix /= N
336         return CentroidMatrix
337 
338     def get_center(self):
339         """
340             Returns a matrix representing the center of the author,
341             The type of matrix depends on the global options for the authors.
342         :return:
343             An Np matrix.
344         """
345         Center = None
346         if Author.CentroidType == CentroidOption.AggregateMatrix:
347             Center = self.__aggregate_matrix()
348         elif Author.CentroidType == CentroidOption.AverageMatrix:
349             Center = self.__average_matrix()
350         else:
351             raise RuntimeError("Unspecified Centroid Option.")
352         return Center
353 
354     def work_distances(self):
355         """
356             The function will calculate the distance for all the transition matrix
357             from each file.
358         :param mode:
359             mode == 1:
```

```
                Using the centroid as the center of this author.
            mode != 1:
                Using the aggregate matrix as the center of this author.
        :return:
            A map; the key is the name of the file and the
            value is the distance from teh centroid matrix.
        """
        DistanceMap = {}
        Center = self.get_center()
        for Writing, Matrix in zip(self.list_of_works(), self.get_matrices()):
            DistanceMap[Writing] = dis(Matrix, Center, Metric=Author.MMetricType)
        return DistanceMap

    def author_cloud(self):
        """
            * The average distance.
            * the standard deviations of the distance.
            * A map describing the distance from the center, mapping
            author's works to a distance represented in a float value.
        :return:
            2 items:
            1. [<average distance>, <standard deviation>], distancelist
            2. A map, string to float.
        """
        DistanceList = self.work_distances()
        Sum = 0
        Squaresum = 0
        L = len(DistanceList)
        for Distance in DistanceList.values():
            Sum += Distance
            Squaresum += Distance**2
        Sum/=L
        Squaresum/=L
        return [Sum, math.sqrt(Squaresum - Sum**2)], DistanceList

    def distance_to(self, m2):
        """
            The function return the distance of this author to a given transition
            matrix of the same size.
```

```
            * How the distance is calculated depends on the input
            parameters.
        :param norm:
            The matrix to norm to calculate the distances.
        :param mdoe:
            mode==1:
                Distance from the centroid of the author.
            mode != 1:
                Distance from the aggregate matrix of the author.
        :return:
            A float.
        """
        centroid = self.get_center()
        if Author.AMetricType == AuthorMetric.CentroidDis:
            return dis(centroid, m2, Metric=Author.MMetricType)
        temp = [dis(m1, m2, Metric=Author.MMetricType) for m1 in self.get_matrices()
    ]
        if Author.AMetricType == AuthorMetric.MinimumDis:
            return min(temp)
        if Author.AMetricType == AuthorMetric.AverageDis: # Not sure if symmetry
    property is preserved.
            return sum(temp)/len(temp)
        raise("Invalid Author Metric. ")

    def cross_distance_stats(self, AnotherAuthor):
        """
            * This function will compare each of the works of THIS author
            to another author using the matrix metric.
            * This function will return detailed statistics about the
            distance for all the works from this author to that author, and THAT
            author to THIS author.

        :param AnotherAuthor:
            An in stance of another author that is not this author.
        :return:
        - w lists are returned, each is a packed information about all information.
        [<Distance Dict>, Avg_distance, SD],
        [<Distance Dict>, Avg_distance, Sd],
        """
```

```
436         def sd_avg(Arr):
437             ArrSquared = list(map(lambda x: x**2, Arr))
438             return sum(Arr)/len(Arr), math.sqrt(sum(ArrSquared)/len(ArrSquared) - (
    sum(Arr)/len(Arr))**2)
439 
440         def cross_compare(Matrices, WorkList, Centroid):
441             Distances = [dis(M, Centroid, Metric=Author.MMetricType) for M in
    Matrices]
442             DistancesDict = dict(zip(WorkList, Distances))
443             Avg, SD = sd_avg(Distances)
444             return [DistancesDict, Avg, SD]
445 
446         ThisCompareToThat = cross_compare(self.get_matrices(), self.list_of_works(),
     AnotherAuthor.get_center())
447         ThatCompareToThis = cross_compare(AnotherAuthor.get_matrices(),
    AnotherAuthor.list_of_works(), \
448                                           self.get_center())
449         return ThisCompareToThat, ThatCompareToThis
450 
451 
452     def __repr__(self):
453         s = "-------------------AUTHOR INFO---------------------\n"
454         s += f"Author's Name: {self.__AuthorName} \n"
455         s += "Distances of his works from the center:\n"
456         Cloud, DistanceList = self.author_cloud()
457         s += f"Average Distance from the center: {Cloud[0]}\n"
458         s += f"Standard deviation of the distances: {Cloud[1]}\n"
459         TitleMaxLength = max(len(W) for W in DistanceList.keys())
460         for Work, dis in DistanceList.items():
461             s += f"{(Work+':').ljust(TitleMaxLength)} {'{:10.4f}'.format(dis)} \n"
462         s += f"Matrix Norm used: {Author.MMetricType}\n"
463         s += f"Centroid Matrix is: {Author.CentroidType}\n"
464         s += f"Function used to generate transition matrix: {self.__TMFunction.
    __name__}\n"
465         return s
466 
467 
468 def dis_between_authors(author1, author2):
469     """
```

```
        This function returns 1 number to represent the distance between 2 author's
        works.
    :param author1:
        An instance of an Author class.
    :param author2:
        An instance of an Author class.
    :param metric
        A type of author metric.
    :return:
        a float.
    """
    metric = Author.AMetricType
    if metric == AuthorMetric.CentroidDis:
        return author2.distance_to(author1.get_center())
    temp = [author1.distance_to(Author2Works) for Author2Works in author2.
    get_matrices()]
    if metric == AuthorMetric.AverageDis:
        return sum(temp)/len(temp)
    if metric == AuthorMetric.MinimumDis:
        return min(temp)
    raise RuntimeError("Invalid AuthorMetricOption")


def main():
    """
            Adventures of Huckleberry Finn: tm27, 2ndtm, and their centroid.
       """
    print("Producing and print...")

    def plot_matrix_savefig(m, filedir: str):
        plt.imshow(m)
        plt.colorbar()
        plt.savefig(filedir)
        plt.clf()

    def save_nparray(m, name):
        np.savetxt(X=m, fname=name)
    global Author

```

```
    Mark27 = Author(MARK_TWAIN, get_tm27)
    Mark2nd = Author(MARK_TWAIN, get_2ndtm)
    Mark27_ignore = Author(MARK_TWAIN, get_tm27, IgnoreSpecialNoun=True)
    Mark2nd_ignore = Author(MARK_TWAIN, get_2ndtm, IgnoreSpecialNoun=True)
    filenames = ["Mark27", "Mark2nd", "Mark27_ignore", "Mark2nd_ignore"]
    authors = [Mark27, Mark2nd, Mark27_ignore, Mark2nd_ignore]
    for Author, Filename in zip(authors, filenames):
        for WorkName, Matrix in zip(Author.list_of_works(), Author.get_matrices()):
            if "Huckle" in WorkName:
                save_nparray(Matrix, Filename + " Huckle.txt")
        save_nparray(Author.get_center(), f"{Filename} Centroid.txt")
    print("Ok, we are going to save some centroid for both of the authors now: ")


def save_matrices_forall_data():
    """
        This function will save all the works of the authors's transition
        matrices as a text files, this is for the concerns of correctness of
        the codes, so that people can just take all the text file and confirm the
        correctness by themselves.
    :return:
        None
    """
    global Author
    def save(NpMatrix, dir:str, filename:str):

        np.savetxt(fname=filename, X=NpMatrix)
        return

    def generate_all_authors():
        AllMatrices = [get_tm27, get_2ndtm, get_2ndlogarithm]
        FileLocations = [CHARLES_DICKENS, MARK_TWAIN]
        ListofAuthors = []
        for L in FileLocations:
            for G in AllMatrices:
                ListofAuthors.append(
                    Author(dir=L,
                           matrixfunction=G,
                           IgnoreSpecialNoun=True))
```

```
                ListofAuthors.append(
                    Author(dir=L,
                           matrixfunction=G,
                           IgnoreSpecialNoun=True))

        return ListofAuthors

    # TODO: FINISH THIS SHIT.
    for Aut in generate_all_authors():
        AuthorName = Aut.name()
        for Work, Matrix in zip(Aut.list_of_works(), Aut.get_matrices()):
            save(Matrix, "")
        pass

    pass


if __name__ == '__main__':
    main()
    pass
```

Github Repository for the codes:

https://github.com/iluvjava/Math-381-Project-2