Modeling Characteristics of Writing with Markov Chains

Zack Barnes, Stephen Hung, Brian Kang, Daniel Pham

June 15, 2019

Contents

1 Introduction

For our project, our group models the pattern of an author's writing by analyzing the distances between the Markov Chain transition matrices of their texts. The states of these transition matrices are the length of the words. The purpose of this model is to be able to identify authorship to a text by measuring the distance of an author's transition matrix to that of another authors' transition matrix. We expand from academic journal and novels found on *Project Gutenburg* text to modern media such as Tweets, a software User manual, and a group member's Facebook message. We investigate if these transition matrices can encapsulate the speech pattern of a given author. Multiple distance metrics are used to see how this metric affects the notion of distance.

2 Background

Our group was interested in Markov Chains for their ability to model transition of states for discrete phenomena. Initially, we wanted to center our project around modeling music composition; however, after further discussion and advise, we realized that coming up with a meaningful way of describing possible transition matrices and their states could be difficult. We then decided to change to modeling how people write/speak. Specifically, we decided to model authors by creating transition matrices for their books.

1. There is a wide source of online text that is available. The abundant amount of resources and data leads to more 'full' transition matrices, that is, matrices with less zero entries. Obtaining less zeroes is ideal because it allows us to fully identify similarities or differences between authors, and our matrices become closer to being ergodic.

2. It did not have complicated data structures that we did not have prior exposure to, for example, MIDI or ABC files.

After we solidified our project topic, we realized that there still was a wide variety of ways in which we could analyze text. Initially, we thought of just having entire word(s) themselves be the states for our transition matrix; however, we realized that could result in millions of words transitioning to millions of other words, creating extremely large transition matrices with mostly zeroes. This would not be feasible for a meaningful model to describe these transition matrices. We discussed other potential options to be the states, such as, subsets of words with different levels of complexity or latin/greek root words underlying the foundations of vocabulary, but these would lead to sparse matrices as well. We then decided to the have the length of individual words be the states of the transition matrix because this will produce less sparse matrices with meaning for us to interpret.[1]

2.1 History

Markov chains were introduced by Russian mathematician, Andrei Andreyevich Markov, in 1906.[2] Markov chains are a mathematical system that describes the probability for an independent sequence of events in a discrete state space. Markov used his mathematical process to calculate the letter sequences in the Russian language in 1913. He expanded the horizon of the Markov process by implementing it with the law of large numbers of dependent events. From this interest, the stochastic Markov process was born, and its significance lies within the fact that the Markov property, which says that the past, present, future are independent, holds for certain random processes in order to accurately build a stochastic model. For this project, using transition matrices will suffice to represent the seemingly random process. Transition matrices are a common way of formatting the entirety of the chain. Figure 1, the state diagram, shows how a transition for predicting weather patterns

[1]Molly L. Lewis and Michael C. Frank. "The length of words reflects their conceptual complexity". In: *Cognition* 153 (Aug. 2016), pp. 182–195. DOI: https://doi.org/10.1016/j.pmedr.2016.04.003.

[2]Guy Leonard Kouemou. "History and Theoretical Basics of Hidden Markov Models". In: *Hidden Markov Models*. Ed. by Przemyslaw Dymarski. Rijeka: IntechOpen, 2011. Chap. 1. URL: http://cdn.intechweb.org/pdfs/15369.pdf.

might be formatted.

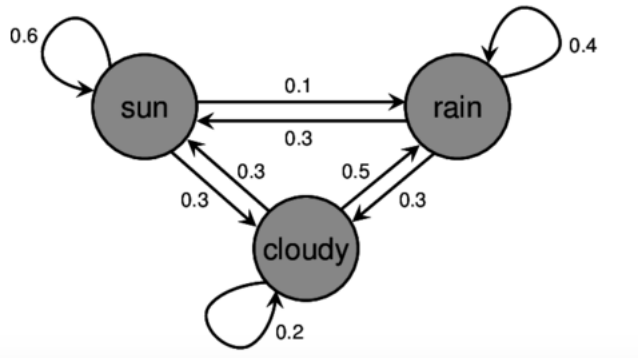

Figure 1: Markov chain for weather prediction.

The transition matrix for that chain is the matrix P below, where the columns are the states for each weather phenomena and the rows are the state that the weather can transition to. Each entry in the transition matrix corresponds to the probability of transitioning from each state in the columns to each state in the rows. Then, the sum of the probabilities in each column equals to one, but the sum of each row need not equal to one. Row one and column one refer to a sunny state; row two and column two refer to a rainy state; row three and column three refer to a cloudy state.

$$P = \begin{bmatrix} 0.6 & 0.3 & 0.3 \\ 0.1 & 0.4 & 0.5 \\ 0.3 & 0.3 & 0.2 \end{bmatrix}$$

2.2 Application: Identification

Aside from predicting weather patterns, Markov chains can be used to describe writing styles of authors called stylometry,[3] which in turn can be used in the field of forensic linguistics for matters of authorship.[4] While there are also other methods (besides Markov

[3] *Stylometry.* URL: `https://en.wikipedia.org/wiki/Stylometry` (visited on 03/07/2019).
[4] Lewis and Frank, "The length of words reflects their conceptual complexity".

Chains) in which authorship of writing could be studied,[5] we decided to use Markov chains because we were interested in possible transition patterns that may emerge. A study in 2001 had a concept of using Markov chains where the states were the individual letter of a text.[6] They pre-processed their data by removed all punctuation and formatting. They also had better results by removing all proper nouns and initial words in sentences based off of a prior study.[7] We will explore this methodology in one of our extensions for our model.

3 Model

For our project, we are going to create a distance model for comparing transition matrices of different authors. We gathered our data from the *Gutenberg Project*, an online repository of novels and other texts.

We decided to make the states of our transition matrix be the length of a word. We include certain punctuation as length 1 words because we found that including punctuation in text analysis is useful for context interpretation.[8] Row i and column j of the transition matrix represent the number of letters a word can have. Each cross-sections (indices) of the transition matrix represent the probability of a word of length j be followed by a word of length i. For example, the sentence below would have the following matrix.

[5]Conrad Sanderson and Simon Günter. "On Authorship Attribution via Markov Chains and Sequence Kernels". In: vol. 3. Jan. 2006, pp. 437–440. DOI: 10.1109/ICPR.2006.899.

[6]Dmitry V. Khmelev and Fiona J. Tweedie. "Using Markov Chains for Identification of Writer". In: *Literary and Linguistic Computing* 16.3 (Sept. 2001), pp. 299–307. DOI: https://doi.org/10.1093/llc/16.3.299.

[7]D.V. Khmelev. "Disputed Authorship Resolution through Using Relative Empirical Entropy for Markov Chains of Letters in Human Language Texts". In: *Journal of Quantitative Linguistics* 7.3 (2000), pp. 201–207. DOI: 10.1076/jqul.7.3.201.4108.

[8]Bernard E. M. Jones. "Exploring the role of punctuation in parsing natural text". In: *Proceedings of the 15th conference on Computational linguistics* - (1994). DOI: 10.3115/991886.991960.

I love math and Markov chains but not the cold snow

$$P = \begin{bmatrix} 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.2 & 0.1 & 0.0 & 0.1 \\ 0.1 & 0.0 & 0.1 & 0.2 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.1 & 0.0 & 0.0 & 0.1 \end{bmatrix}$$

This matrix is rather sparse because the text is rather small in size. The data we acquired has 100,000's of word transitions per a text.

3.1 Set-Up for Data

We implemented Python to read in and process text files downloaded from Project Gutenberg and to construct the transition matrices. The code for constructing the these transition matrices can be found in the appendix, Section 7.1. The code takes the text file as an input such that the program could read it word by word, or token by token. Then the code processes each of the words' lengths l and stores a count for each different word length following the initial word length l in a matrix. For instance, *Math is fun!* would have the following representation.

		from word length			
		1	2	3	4
to word length	1	0	0	1	0
	2	0	0	0	1
	3	0	1	0	0
	4	0	0	0	0

Then, the code normalizes the matrix by the total number of transitions per word length in the text file so that each transitional probability is computed. The resulting transition matrix is in the form such that the element in the ith column and jth row, x_{ij}, is the probability that a word with length i is followed by a word with length j.

We decided to create each of our transition matrices with state of maximum 25 letters. This is because in some texts, words up to length 23 were used such as "Geschlechtsverhaltnisse" in Darwin's book *Different Forms of Flowers*. Even though many of the texts did not have up to 25 letters, the transition matrices must have the same dimensions for computing the distance among and between them.

We chose a total of 5 different authors from *Project Gutenberg* and ran all of their available texts (excluding text written in non-English languages) through our program. Below is a table for the number of texts that we used for each author.

Author	**Number of Texts**
Jane Austen	7
Charles Darwin	9
Charles Dickens	21
Sir Arthur Conan Doyle	63
William Shakespeare	11

We also used two children's stories for the purpose of cross-checking the authors' transition matrices, utilizing the knowledge that children stories likely do not contain words with long length. This means that we would expect a large distance between the transition matrices of authors' novels and those of the children stories.

The purpose of running through all of the authors texts was to encapture their writing style to the best extent possible. For each of the authors, we created a "centroid" transition matrix which was the averaged sum of all of the transition matrices of authors' texts.

3.2 Computing the Distance

Our first model for computing the distance between the transition matrices was using the Euclidean distances between the centroids of the authors. We define centroid as the mean value of all transition matrices of an author. We first computed the centroid of the authors by using this mathematical expression,

$$\sum_{k=1}^{n}\sum_{j=1}^{25}\sum_{i=1}^{25} x_{ij,k}/T_w = Cw$$

where,

1. n = the number of texts the author wrote
2. x_{i_j} = the probability of an i length word to lead to a j length word of text k
3. T_w = the total number of word transitions in all of author w's texts
4. C_w = the centroid of author w

Then, we computed the distances between authors by comparing the centroids of each author. For one author's centroid X where $x_{ij} \in X\ \forall i, j \in [1, 2, ..., 25]$ and another author's centroid Y where $y_{ij} \in Y\ \forall i, j \in [1, 2, ..., 25]$.

$$\sum_{i=1}^{25}\sum_{j=1}^{25} |x_{ij} - y_{ij}|$$

We computed the centroid for each of the 5 authors (and also for the children's stories) and created a histogram of the overall word-length frequency of the respective authors' centroid to see if there were any noticeable differences.

The code that generated these histograms can be found in the appendix, Section 7.5.

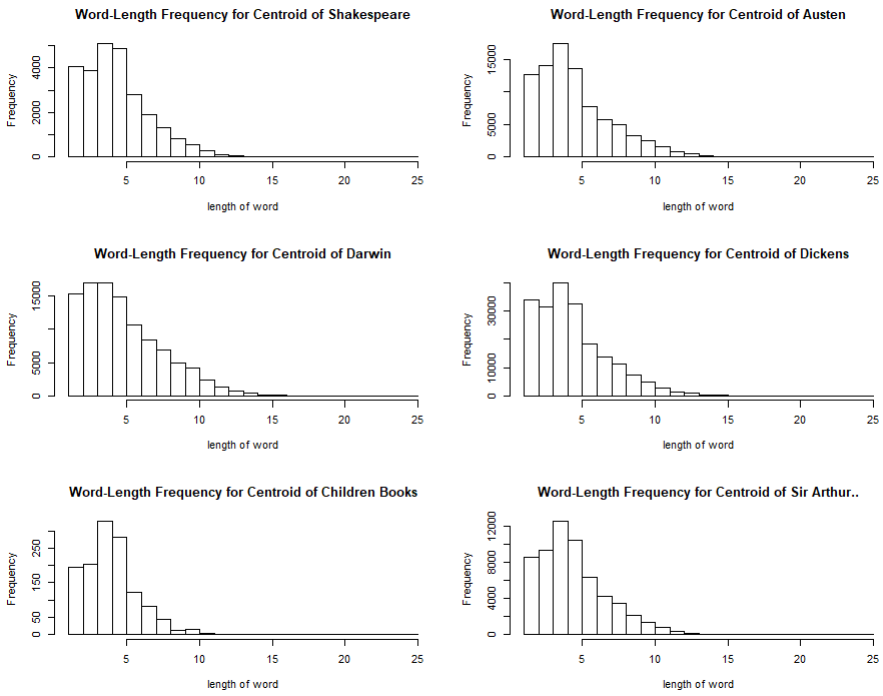

Figure 2: Frequency of word lengths for the authors' centroid

3.3 Assumptions of Our Model

For our model, we took the following assumptions...

1. All the text taken from *Project Gutenberg* was written by the author. This means everything was transcribed word for word, character by character.

2. Punctuation was the author's choice and was thus included in our model as length one words.

3. A one-chain Markov is enough to encapsulate the stylistic writing of an author.

4. Only the lengths of words that the author wrote will give meaningful information even though it has removed many properties of words such as meaning, parts of speech, etc.

4 Results

We ran our python code on a Macbook Pro Retina 13in. 2015 model. It took 0m30.040s.

We constructed heat map images of the centroid of each of the authors. From these images, we are able to distinguish which word-length probabilities were the most prevalent for the author. Below is a sample of the heat map images. The darker the color, the higher the probability for that row index to column index word transition. The rest of the authors heat maps can be found in the Appendix Section 7.2. Note that the indices began at 0, however this is due to the code format for the heat map. In reality each of the indices are +1 since there are no words of length 0.

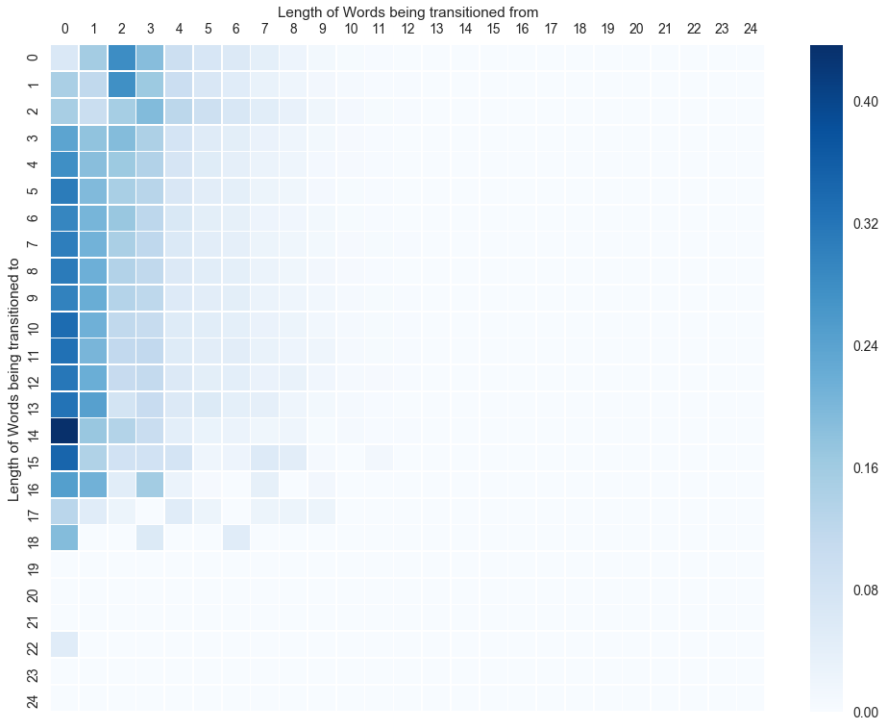

Figure 3: Heat Map for Dickens

We found the authors distances, seen in Figure 5. We can see from the results, Jane Austen, Charles Dickens, and Arthur Doyle had the most similar and smallest distance of approximately 3. Interestingly, Shakespeare was not close to any of the other authors with

Figure 4: Distance comparison between the centroids of different authors

his distances ranging from around 5 to around 8.

A result that we anticipated was that the distance from the authors to the Children's Stories should be large since Children's Stories most likely use shorter and more simple words children can understand. Moreover, the authors we chose such as Charles Darwin and Shakespeare were not writing for an audience of children.

From our heatmap, there is an interesting trend in that there is a high probability that any length word is transitioned from a one length word. We printed some of these occurrences in Charles Dickens' books and found that this occurred because we took punctuation into account. Since we treated punctuation as a one length word, any word after punctuation is treated as a transition from a one length word to another length word.

We also compared the distance between each author's books/plays with every author's centroid. We then took the average of the distance from each author's books/plays to every

author's centroid. There is a small snapshot of the results in the figure below. It has the average distance from all of Jane Austen's books to every author's centroids including her own. It also has the average distance from all of Charles Dicken's books to every author's centroids including his own. The full image can be seen in Appendix 7.4. We found that the distance between each author's texts and the centroids for every author was roughly the same (in a range from 2 to 4). Since this is the case, we cannot conclude that using the transition matrices for word lengths can identify authors.

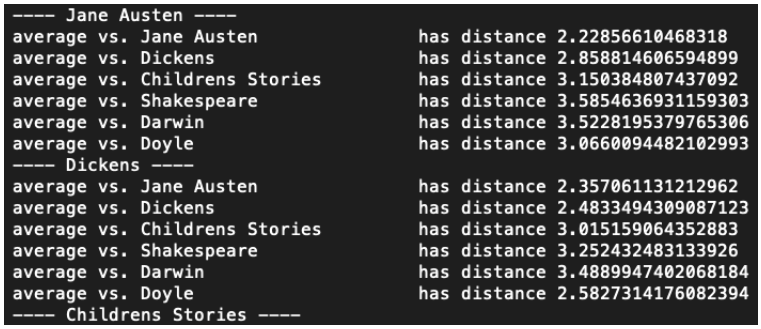

Figure 5: Average Distance from Dickens and Jane Austen's books to all author's centroids.

While we cannot gauge our model because there have not been previous studies done on the word-length transitions of these authors, we can note that it encapsulates the idea that of authors having different writing patterns.

5 Model Adjustment and Variations

5.1 Adjustment

After completing our initial model, we focus on how we can expand our analysis. We take several variations of the model itself, along with trying several different distance metrics on each of the variations.

Our first variation is to include different types of media, aside from Gutenberg texts. The reasoning behind this variation is that our speech or way of writing can change depending

on the environment one is in. The three forms of modern-day media chosen were tweets from Donald Trump, a user manual for computer software, and the Facebook messages of one team member.

Our second variation is to remove all capitalized words and remove all proper nouns and first words of a sentence. We utilize this variation because of a study by Dmitri V. Khmelev (2000) that found this technique to improve rates of correct authorship identification.[9]

Finally, we repeat the previous variations and our original model with three distinct distance metrics. First, we use two different variations of Minkowski distance, the distance between two variables in an n-dimensional space, X and Y,

$$\sum_{i=1}^{n} (|X_i - Y_i|^p)^{\frac{1}{p}}$$

.

In our first case, we use the Minkowski distance where $p = 1$. In our second case, we use $p = 2$, also known as the "Manhattan" distance. We attempt to find the distance when $p \to \infty$, however all distances returns as zero so this metric is not used.

Our last distance metric use the Minkowski distance with $p = 1$, however, we add a conditional statement in the distance calculation method that will only consider entries of the transition matrices that are larger than 0.05. Our reasoning behind the implementation of this metric is that we only want to consider the most prominent transitions, and omit uncommon transitions.

The two additional distance metrics, Minkowski $p = 2$ and Conditional Minkowski will be explained further in Variation 3 and 4.

[9]Khmelev, "Disputed Authorship Resolution through Using Relative Empirical Entropy for Markov Chains of Letters in Human Language Texts".

5.2 Variation Results

5.2.1 Variation 1: Different Media Types

The range of the average distance of author centroids between Gutenberg texts range between 5 and 6, with the exception of childrens book with an average distance of over 10. The Facebook messages have an average distance of 8.03 from the Gutenburg texts − slightly higher than the range of books and academic papers from Darwin. The User manual presents the next largest average distance from the Gutenburg texts of 9.062. Finally, Trump Tweets have the largest average distance from the Gutenburg texts of 18.91. The distance for the Trump Tweets seem to be quite large, and this led us to our eventual modification our distance metric.

5.2.2 Variation 2: Removing Capitalized Words

In order to implement this variation, we modify the code so that it ignores all capitalized words in the initial construction of the transition matrix..[10] Afterwards, we compute the distance between each author's centroid using the three distance measures defined above (Minkowski $p = 1$, $p = 2$, and when probability ≥ 0.05). The results are in the appendixes.

Furthermore, we compute the difference in each author's centroid when capitalized words were kept versus when they were removed. Using the Minkowski distance of $p = 1$, the results are as follows:

[10]Khmelev, "Disputed Authorship Resolution through Using Relative Empirical Entropy for Markov Chains of Letters in Human Language Texts".

Author	Centroid Distance
Doyle	0.6880
Charles Dickens	0.8467
Jane Austen	0.9688
Darwin	1.2951
Children's Stories	1.1304
Shakespeare	1.5199

The centroid distance is within our assumptions, since removing capital words should not change the centroid of an author greatly.

Interested in what word transitions made such a big difference, we visualize the difference in a heat map where the larger the difference in an element of the transition matrices, the darker that element become. Let us look at the difference for Shakespeare as it has the largest difference in centroids. (The indices represent the word length minus 1 , so index 0 is word length 1).

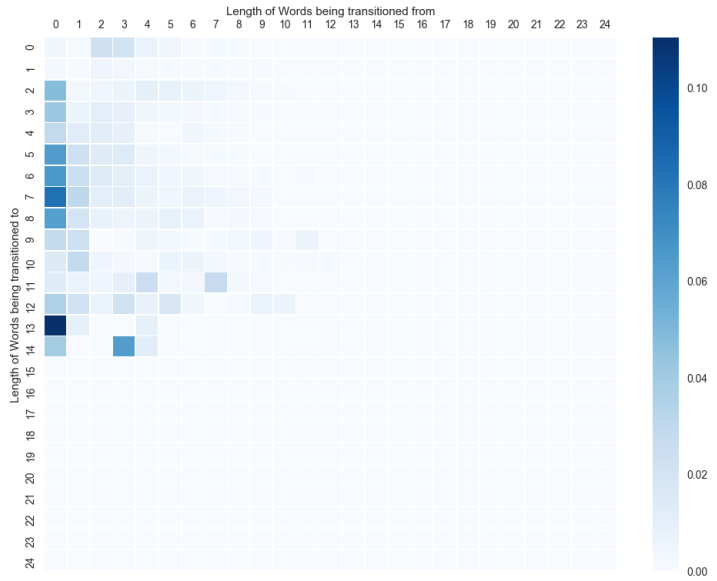

Figure 6: Difference Centroid for Shakespeare

As we can see, there is a relatively small difference between Shakespeare's centroid containing capitalized words and Shakespeare's centroid without capitalized words primarily around transitioning from one to four length words.

The three highest difference in probabilities and their different probabilities can be seen below:

	W/ out Capital Words	With Capital Words	Difference
Transitioning from length 1 to length 14	0.4242	0.5346	0.1104
Transitioning from length 1 to length 8	0.3289	0.4118	0.0829
Transitioning from length 1 to length 7	0.3096	0.3754	0.0658

5.2.3 Variation 3: Minokowski $p = 2$

In order to implement this variation, we add that the difference of each of the elements in the centroid will rise to the power of 2, then in the return statement we take the square root. This metric provides similar results as the absolute difference in our original model, however, results are scaled down. The range of results also reduce to values between 0.5 and 2. We believe this metric does not provide utility as it seems to decrease distances of distinct media.

5.2.4 Variation 4: Conditional Minokowski

In order to implement this variation, we add a conditional statement in our initial matrix model so that only probabilities ≥ 0.05 are counted. While this mainly scales down the transition matrices for all authors, it can be seen that the distance between some of the authors increase/decrease.

Below is a table of the furthest distance that each text has with this new model.

Author	Furthest From	Distance w/ $p \geq 0.05$	Original Distance
Jane Austen	Trump Tweets	5.089	15.5896
Charles Darwin	User Manual	5.218	10.2294
Charles Dickens	Trump Tweets	4.941	14.4409
Sir Arthur Conan Doyle	User Manual	4.677	8.1055
William Shakespeare	User Manual	5.362	7.9897
Children's Stories	Trump Tweets	4.151	19.9460
Facebook	User Manual	1.78	9.3784
User Manual	Trump Tweets	5.914	18.3215

6 Conclusion

From our investigation, we have found that Markov chains are not useful in the modeling of characteristics in writing styles. Our results show different distances for distinct types of writing found in novels, academic journals, and childrens books have a similar range of distances depending on the type of text, however, when we measured each text against every centroid which resulted in similar distances for all authors to every text. If this was an accurate model, we would expect to see nearly no difference between books of the same other, and a larger difference otherwise. We believe that the model cannot fully encapsulate all writing characteristics as writing is much more rich and complex than merely length of transitions from one word to another.

During our investigation, we found that there is many ways in which that distance could be measured, and depending on this metric we can change our notion of "close" and "far". We found that restructuring how we collect the data (i.e., removing all the words that started with a capital letter) to show larger results for measuring distance. This is ideal for a distance model to able distinguish differences between texts.

We also found that different forms of writing can lead to having different transition matrices, as the text files from Gutenberg (mostly compromised of novels) were closer in distance to each other than to any of the three forms of media we used *see appendix 7.3*. The Trump tweets in particular were furthest distance to the other forms of text in almost

all variations that we ran; this could be due to the fact that Tweets are a form of text communication that is meant to be short (Tweets are limited to 280 characters or less).

In conclusion, it is important for distance models to explicitly identify and recognize the data of which it is studying; if there is academic novels, such as Charles Darwin's use of "Geschlechtsverhaltnisse," it could be ideal to take measures of removing words with outlier word-lengths. It is also important to test different measures of distance like in our variation 4, where we apply conditional formatting when computing the distance. While it did remove some of our data, it allows for the isolation of more probable transitions in the an author's writing.

While we learned a lot from using Markov Chains to look at how an author writes, there are other measures besides word length that have significant importance. By reducing the text to just the letter count of a word, a lot of other data (such as meaning) was lost. Because Markov chains work better with more ample datasets, we would most likely use another technique to explore authors writing so we could explore other aspects of text.

References

Jones, Bernard E. M. "Exploring the role of punctuation in parsing natural text". In: *Proceedings of the 15th conference on Computational linguistics* - (1994). DOI: 10.3115/991886.991960.

Khmelev, Dmitry V. and Fiona J. Tweedie. "Using Markov Chains for Identification of Writer". In: *Literary and Linguistic Computing* 16.3 (Sept. 2001), pp. 299–307. DOI: https://doi.org/10.1093/llc/16.3.299.

Khmelev, D.V. "Disputed Authorship Resolution through Using Relative Empirical Entropy for Markov Chains of Letters in Human Language Texts". In: *Journal of Quantitative Linguistics* 7.3 (2000), pp. 201–207. DOI: 10.1076/jqul.7.3.201.4108.

Kouemou, Guy Leonard. "History and Theoretical Basics of Hidden Markov Models". In: *Hidden Markov Models*. Ed. by Przemyslaw Dymarski. Rijeka: IntechOpen, 2011. Chap. 1. URL: http://cdn.intechweb.org/pdfs/15369.pdf.

Lewis, Molly L. and Michael C. Frank. "The length of words reflects their conceptual complexity". In: *Cognition* 153 (Aug. 2016), pp. 182–195. DOI: https://doi.org/10.1016/j.pmedr.2016.04.003.

Sanderson, Conrad and Simon Günter. "On Authorship Attribution via Markov Chains and Sequence Kernels". In: vol. 3. Jan. 2006, pp. 437–440. DOI: 10.1109/ICPR.2006.899.

Stylometry. URL: https://en.wikipedia.org/wiki/Stylometry (visited on 03/07/2019).

7 Appendixes

7.1 Python Code for Generating the Transition Matrices

```
# Markov Chain for regular text files.

import sys, os, json, random
import copy

import random
import numpy as np
import re
import math

# size of matrix (N x N)
N = 25

# Transition Matrix class used to create markov chain. (can ignore)
class TransitionMatrix:

    def __init__(self, init_name=None):
        self.matrix = {}
        self.norm_matrix = None
        self.fname = init_name if init_name else 'data'
        self.curr_state = None
        self.choice_matrix = None

    def add_transition(self, prev_state, next_state):
        if prev_state in self.matrix.keys():
            if next_state in self.matrix[prev_state]:
                self.matrix[prev_state][next_state]+=1
```

```
28              else:
29                  self.matrix[prev_state][next_state]=1
30
31          else:
32              self.matrix[prev_state] = {}
33              self.matrix[prev_state][next_state]=1
34
35      def normalize(self):
36          new_matrix = {}
37          for prev_freq in self.matrix.keys():
38              curr_freq_data = self.matrix[prev_freq]
39              total = sum(curr_freq_data.values())
40              new_dict = {}
41              for key in curr_freq_data.keys():
42                  new_dict[key] = curr_freq_data[key]/total
43              new_matrix[prev_freq] = new_dict
44
45          self.norm_matrix = new_matrix
46
47      # Save the data into a file
48      # Returns either the file name for the frequencies,
49      # or the file name for the normalized probabilities
50      def save(self, norm=False):
51          self.normalize()
52          frequency_data = json.dumps(self.matrix)
53          normalized_data = json.dumps(self.norm_matrix)
54          open(self.fname + '.json', 'w').write(frequency_data)
55          open(self.fname + '_norm.json', 'w').write(normalized_data)
56          if not norm:
57              return self.fname + '.json'
58          return self.fname + '_norm.json'
```

```
59
60      # Add data from another existing matrix into the current object.
61      # Combines and sums the new data with the data in the structure
62      def load_data(self, filepath):
63          if os.path.isfile(filepath):
64              prob_raw = open(filepath)
65              prob_raw_data = prob_raw.read()
66              new_matrix = json.loads(prob_raw_data)
67              for key in new_matrix.keys():
68                  if key in self.matrix:
69                      for transition in new_matrix[key].keys():
70                          if transition in self.matrix[key]:
71                              self.matrix[key][transition] += new_matrix[key][
    transition]
72                          else:
73                              self.matrix[key][transition] = new_matrix[key][
    transition]
74                  else:
75                      self.matrix[key] = new_matrix[key]
76              self.normalize()
77
78      def initialize_chain(self):
79          self.normalize()
80          self.curr_state = random.choice(list(self.matrix.keys()))
81          self.choice_matrix = copy.deepcopy(self.norm_matrix)
82          for key in self.choice_matrix:
83              acc = 0
84              for outcome in self.choice_matrix[key]:
85                  acc += self.choice_matrix[key][outcome]
86                  self.choice_matrix[key][outcome] = acc
87
```

```
88      def get_next_outcome(self):
89          rand = random.random()
90          if not self.curr_state in self.choice_matrix:
91              self.curr_state = random.choice(list(self.choice_matrix.keys()))
92          for key in self.choice_matrix[self.curr_state]:
93              if self.choice_matrix[self.curr_state][key] >= rand:
94                  self.curr_state = key
95                  return key
96
97  # Load in text file and create a transition matrix
98  # INPUT:
99  #   filename    String of file's name.
100 # OUTPUT:
101 #   maxLengthWord     length of longest word - used in compareMatrices to
        create best size matrix.
102 def loadTextFile(filename, matrix):
103     maxLengthWord = 0
104     startReading = True
105     file = open(filename,encoding="utf8")
106     for line in file:
107         if startReading:
108             words = re.findall(r"[\w']+|[.,!?;]", line)
109             for i in range(0, len(words) - 1):
110                 lenInputWord = len(words[i])
111                 lenNextWord = len(words[i+1])
112                 if(lenInputWord > maxLengthWord):
113                     maxLengthWord = lenInputWord
114                 elif(lenNextWord > maxLengthWord):
115                     maxLengthWord = lenNextWord
116                 matrix.add_transition(lenInputWord,lenNextWord)
117     return maxLengthWord
```

```
118
119 # convert TransitionMatrix into a 2D array
120 # OUTPUT:
121 #   arr2    2D array representation of TransitionMatrix
122 def convertTo2D(tMatrix):
123     tMatrix.normalize()
124     arr = np.zeros((N,N))
125     for key in tMatrix.norm_matrix:
126         for nextKey in tMatrix.norm_matrix[key]:
127             NumNextKey = tMatrix.norm_matrix[k
128             ey][nextKey]
129             arr[key-1][nextKey-1] = NumNextKey
130     return arr
131
132 # Compare 2D arrays (transition matrices) via distance formulas and return
    distance.
133 # INPUT:
134 #   arr1    first transition matrix - result from convertTo2D
135 #   arr2    second transition matrix - result from convertTo2D
136 #   length  int of largest word length - used to make distanceMatrix best size
    .
137 # OUTPUT:
138 #   distance  result of distance comparison between the two transition matices
139 def compareMatrices(arr1, arr2, length):
140     distanceMatrix = np.zeros((length,length))
141     distance = 0
142     for i in range(0,length):
143         for j in range(0,length):
144             distanceMatrix[i,j] = abs(arr1[i][j] - arr2[i][j]) # rounding to 2
    nd decimal point
145             distance+= abs(arr1[i][j] - arr2[i][j]) # Abs value difference
```

```
146             #distance+= math.pow((arr1[i][j] - arr2[i][j]),2) # square root
    distance (1) square difference
147     #return distance**(0.5) # square root distance (2) square root of total
148     print('\n'.join([''.join(['{:10}'.format("%.4f" % item) for item in row])
    # Prints out distance matrix
149       for row in distanceMatrix]))
150     return distance
151
152 # General steps for creating transition matrix
153 # (1) init TransitionMatrix
154 # (2) load text file and get the length
155 # (3) convert matrix to 2D
156
157 maxLength = 0
158 distance = 0
159 matrixArray = {}
160 counter = 0
161 os.chdir("ja")
162 for file in glob.glob("*.txt"):
163     counter+=1
164     matrixArray[file] = TransitionMatrix()
165     length = loadTextFile(file, matrixArray[file])
166     arr = convertTo2D(matrixArray[file])
167     maxLength = max(length, maxLength)
168     matrixArray[file] = arr
169     print("----  ARR {}  ----".format(counter))
170     print('\n'.join([''.join(['{:10}'.format("%.4f" % item) for item in row])
    # Prints out distance matrix
171           for row in arr]))
172     print("maxlength = ", maxLength)
173     print("-------")
```

```
174
175 # compare matrices
176 distanceMatrix = {}
177 centroidMatrix = np.zeros((maxLength+1,maxLength+1))
178 print(maxLength)
179 print("----  DISTANCE  ----")
180 for key in matrixArray:
181     centroidMatrix = centroidMatrix + matrixArray[key]
182     for key2 in matrixArray:
183         if(key2 != key):
184             distance = compareMatrices(matrixArray[key], matrixArray[key2],
    maxLength)
185             distanceMatrix["{} vs. {}".format(key,key2)] = distance
186 centroidMatrix = centroidMatrix / counter
187
188 for key in distanceMatrix:
189     print("{key:<50} {value:>10}".format(key=key,value="%.4f" %distanceMatrix[
    key]))
190 #distance = compareMatrices(arr1,arr2, maxLength)
191 print("-------")
192
193 print("Distance between matrices =", distance) # prints out difference between
     arr1 and arr2
194
195 #print(arr)
196 #print("----------")
197 #print(arr2)
```

7.2 HeatMaps for Initial Distance Model

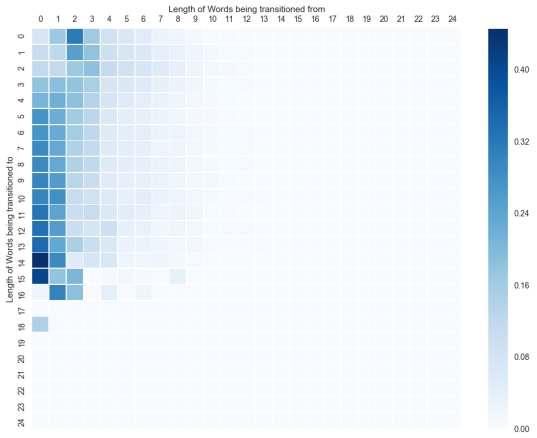

Figure 7: Initial Distance Model Heat map for Austen

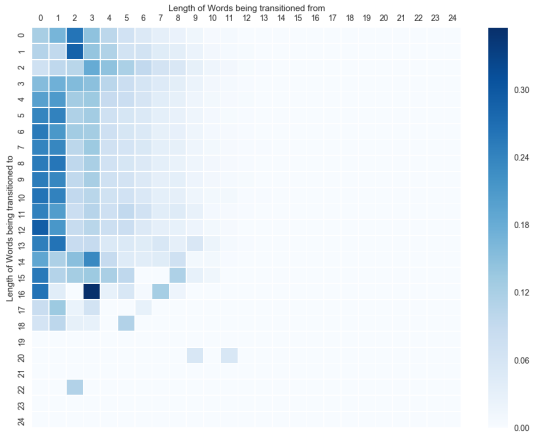

Figure 8: Initial Distance Model Heat map for Darwin

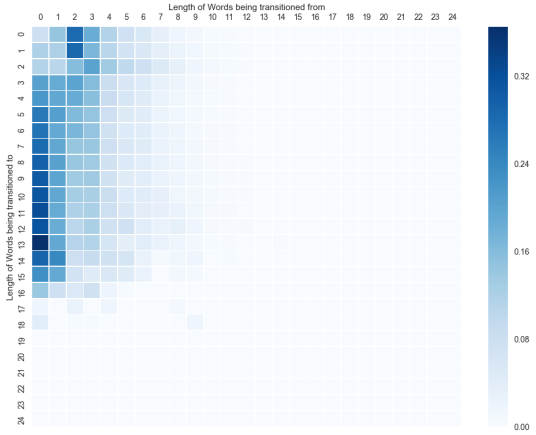

Figure 9: Initial Distance Model Heat map for Doyle

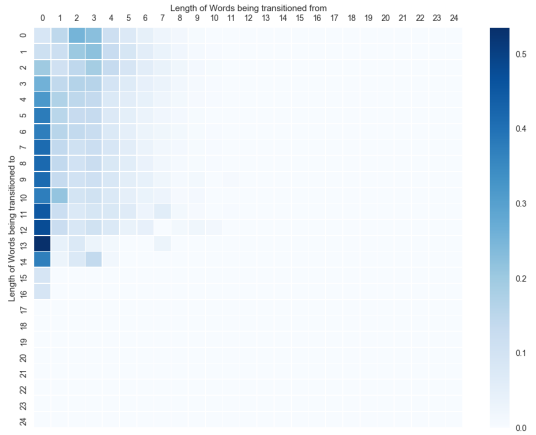

Figure 10: Initial Distance Model Heat map for Shakespeare

7.3 Table for Max and Min for Authors through the Variations

Below are comparison graphs for each of the authors throughout the 3 variations/adjustments we did.

Jane Austen	**Variation 2**	**Variation 3**	**Variation 4**
Closest to	Doyle (3.4)	Doyle (0.5)	Dickens (1.8)
Furthest From	Trump (15.6)	User Manual (1.7)	Trump (5.1)

Charles Dickens	**Variation 2**	**Variation 3**	**Variation 4**
Closest to	Doyle (3.1)	Doyle (0.4)	Austen (1.9)
Furthest From	Trump (14.4)	User Manual (1.6)	Trump (4.9)

Children Stories	Variation 2	Variation 3	Variation 4
Closest to	Facebook (8.0)	Doyle (1.1)	Dickens (2.6)
Furthest From	Trump (19.9)	User Manual (1.8)	Trump (4.1)

Shakespeare	Variation 2	Variation 3	Variation 4
Closest to	Doyle (5.1)	Doyle (0.6)	Children's (2.9)
Furthest From	Trump (16.8)	Trump (1.7)	User Manual (5.3)

Darwin	Variation 2	Variation 3	Variation 4
Closest to	Doyle (5.6)	Doyle (0.6)	Austen (3.0)
Furthest From	Trump (13.9)	Trump (1.5)	User Manual (5.2)

Doyle	Variation 2	Variation 3	Variation 4
Closest to	Dickens (3.1)	Dickens (0.4)	Dickens (1.9)
Furthest From	Trump (13.9)	Trump (1.5)	User Manual (4.7)

Facebook	Variation 2	Variation 3	Variation 4
Closest to	Doyle (6.6)	Doyle (0.9)	Children's (3)
Furthest From	Trump (16.5)	User Manual (1.8)	Shakespeare (4.1)

User Manual	Variation 2	Variation 3	Variation 4
Closest to	Shakespeare (8.0)	Darwin (1.4)	Children's (2.9)
Furthest From	Trump (18.3)	Trump (2.1)	Trump (5.9)

Trump	Variation 2	Variation 3	Variation 4
Closest to	Darwin (13.9)	Darwin (1.5)	Facebook (3.6)
Furthest From	User Manual (18.3)	User Manual (2.1)	User Manual (5.9)

7.4 Minkowski Distance $p = 1$ comparison between all text transition matrices and each author's centroid

Take the average distance between each transition matrix from each author's text files used and compare it to all author's centroids. As we can see, the distances are roughly the same with a range from 2 to 4. Furthermore, one author's average is not necessarily the closest to that author's centroid. For example, Charles Dickens has an average distance of around 2.48 from all of his texts to his own centroid but has an average distance of around 2.357 to Jane Austen.

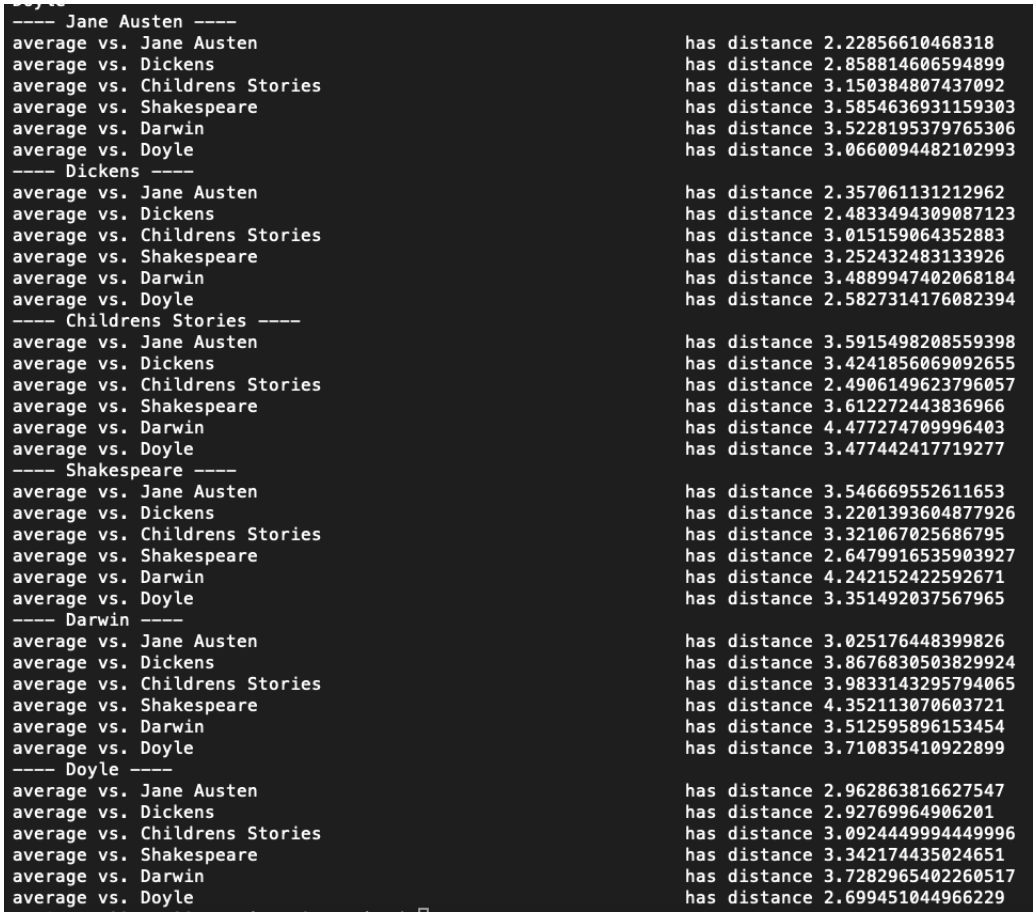

Figure 11: Average distance between each author's text and all author's centroid.

7.5 The Code Used to Generate the Histograms

```
# -*- coding: utf-8 -*-
"""
Created on Thu Mar  7 11:25:23 2019

@author: danny
"""
import matplotlib.pyplot as plt
import os
import glob
import re
```

```
# remember to remove the brackets in r studio
num = 25
centroid = [0] * num
counter = 0
author = "bs" #put author name here
os.chdir(author)
f = open("{}HIST.txt".format(author),"w+")
f.write('size <- 1:25\n')
f.write('par(mfrow=c(3,3))\n')
for file in glob.glob("*.txt"):
    f.write("{} <- c(".format(file))
    counter = counter + 1
    filename = open(file,encoding="utf8")
    matrix = [0] * num # change to the size of whatever we are doing
    for line in filename:
        words = re.findall(r"[\w']+|[.,!?;]", line)
        for i in range(0, len(words) - 1):
            if(len(words[i]) < num):
                #should I -1 for the words length? for some reason there is no
    1 length
                matrix[len(words[i])] = matrix[len(words[i])] + 1
                centroid[len(words[i])] = centroid[len(words[i])] + 1
    f.write('{}'.format(matrix))
    f.write(") \n")
    f.write('table <- table(c(size), c({}))\n'.format(file))
    #change overall condition if we need to a different range
    f.write("df <- as.data.frame(cbind(Overall.Cond= 1:25, {}))\n".format(file
    ))
    f.write("df.freq <- as.vector(rep(df$Overall.Cond, df${}))\n".format(file)
    )
```

```
38      f.write('hist(df.freq, xlab = "length of word", main = "Word-Length
        Frequency of {}", breaks = 1:25)\n'.format(file))
39      #f.write('\n')
40  newCentroid = [x / counter for x in centroid]
41  f.write("centroid <- c({}".format(newCentroid))
42  f.write(") \n")
43  f.write('table <- table(c(size), c(centroid))\n')
44      #change overall condition if we need to a different range
45  f.write("df <- as.data.frame(cbind(Overall.Cond= 1:25, centroid))\n")
46  f.write("df.freq <- as.vector(rep(df$Overall.Cond, df$centroid))\n")
47  f.write('hist(df.freq, xlab = "length of word", main = "Word-Length Frequency
        of centroid", breaks = 1:25)\n')
```

The python code above was used to generate the text (code) that was inputted into R-Studio.

```
1  size <- 1:25
2  par(mfrow=c(3,2))
3
4  centroid <- c(0.0, 4078.4166666666665, 3902.5833333333335, 5102.75,
       4870.916666666667, 2802.5833333333335, 1905.25, 1312.75,
       828.0833333333334, 531.9166666666666, 262.3333333333333,
       108.58333333333333, 42.916666666666664, 20.916666666666668,
       5.583333333333333, 2.4166666666666665, 0.25, 0.16666666666666666, 0.0,
       0.0, 0.0, 0.0, 0.0, 0.0, 0.0)
5  table <- table(c(size), c(centroid))
6  df <- as.data.frame(cbind(Overall.Cond= 1:25, centroid))
7  df.freq <- as.vector(rep(df$Overall.Cond, df$centroid))
8  hist(df.freq, xlab = "length of word", main = "Word-Length Frequency for
       Centroid of Shakespeare", breaks = 1:25)
9
10 centroid <- c(0.0, 12612.75, 13987.0, 17404.875, 13502.875, 7634.125, 5696.25,
       4893.125, 3166.5, 2509.125, 1492.625, 748.125, 414.5, 204.0, 57.625,
```

```
    19.75, 6.125, 2.5, 0.625, 0.5, 0.0, 0.0, 0.0, 0.0, 0.0)
table <- table(c(size), c(centroid))
df <- as.data.frame(cbind(Overall.Cond= 1:25, centroid))
df.freq <- as.vector(rep(df$Overall.Cond, df$centroid))
hist(df.freq, xlab = "length of word", main = "Word-Length Frequency for
    Centroid of Austen", breaks = 1:25)

centroid <- c(0.0, 15335.3, 17034.5, 17040.3, 14926.1, 10635.8, 8452.5,
    6860.1, 4899.4, 4183.7, 2419.0, 1234.0, 750.0, 333.9, 152.4, 36.4, 6.0,
    3.0, 1.1, 1.2, 0.0, 0.2, 0.0, 0.1, 0.0)
table <- table(c(size), c(centroid))
df <- as.data.frame(cbind(Overall.Cond= 1:25, centroid))
df.freq <- as.vector(rep(df$Overall.Cond, df$centroid))
hist(df.freq, xlab = "length of word", main = "Word-Length Frequency for
    Centroid of Darwin", breaks = 1:25)

centroid <- c(0.0, 33917.90476190476, 31365.95238095238, 40199.19047619047,
    32757.190476190477, 18384.52380952381, 13770.47619047619,
    11108.285714285714, 7448.047619047619, 5008.9047619047615,
    2840.904761904762, 1427.5714285714287, 798.5238095238095,
    411.42857142857144, 104.9047619047619, 42.19047619047619,
    6.523809523809524, 2.4285714285714284, 0.5238095238095238,
    0.5238095238095238, 0.0, 0.0, 0.0, 0.047619047619047616, 0.0)
table <- table(c(size), c(centroid))
df <- as.data.frame(cbind(Overall.Cond= 1:25, centroid))
df.freq <- as.vector(rep(df$Overall.Cond, df$centroid))
hist(df.freq, xlab = "length of word", main = "Word-Length Frequency for
    Centroid of Dickens", breaks = 1:25)


centroid <- c(0.0, 195.33333333333334, 204.33333333333334, 327.6666666666667,
```

```
    282.6666666666667, 123.0, 81.0, 44.0, 12.333333333333334, 15.0, 4.0,
    0.6666666666666666, 0.3333333333333333, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
     0.0, 0.0, 0.0, 0.0, 0.0)
table <- table(c(size), c(centroid))
df <- as.data.frame(cbind(Overall.Cond= 1:25, centroid))
df.freq <- as.vector(rep(df$Overall.Cond, df$centroid))
hist(df.freq, xlab = "length of word", main = "Word-Length Frequency for
    Centroid of Children Books", breaks = 1:25)

centroid <- c(0.0, 8549.333333333334, 9337.079365079366, 12490.174603174602,
    10419.587301587302, 6349.746031746032, 4280.015873015873,
    3479.2380952380954, 2116.809523809524, 1376.4126984126983,
    766.5714285714286, 337.8253968253968, 179.42857142857142,
    79.22222222222223, 25.07936507936508, 6.428571428571429,
    1.7301587301587302, 0.6190476190476191, 0.09523809523809523,
    0.12698412698412698, 0.0, 0.0, 0.0, 0.0, 0.0)
table <- table(c(size), c(centroid))
df <- as.data.frame(cbind(Overall.Cond= 1:25, centroid))
df.freq <- as.vector(rep(df$Overall.Cond, df$centroid))
hist(df.freq, xlab = "length of word", main = "Word-Length Frequency for
    Centroid of Sir Arthur..", breaks = 1:25)
```