Week 8 Jupyter Notebook Assignment

Programming Historian: Introduction to stylometry with Python

Student Name: Muhammad Rakibul Islam

- Date:
- Instructor: Lisa Rhody
- Assignment due:
- Methods of Text Analysis
- MA in DH at The Graduate Center, CUNY

Preparing for this week - Conceptualization

This week's assignment is going to veer more toward the technical than other weeks. We are going to read about a type of sylometric analysis called authorship attribution, and we're going to work through one of the most common authorship attribution tests in DH.

The following notebook is much indebted to and grateful for Programming Historian, and in particular, François Dominic Laramée.

François Dominic Laramée, "Introduction to stylometry with Python," The Programming Historian 7 (2018), https://programminghistorian.org/en/lessons/introduction-to-stylometry-with-python.

The assignment

This week, you will be working through one of the most common authorship attribution activities: Assessing the likely authorship of the disputed essays in *The Federalist Papers*. As you do so, consider the readings that you have done and how they come into conversation with the methods in this activity. The assignment will work through three different approaches to the same question using 3 different assumptions about language and measuring. You may work together to run the cells of the notebooks, but individual assignments should be completed individually.

Your comments should reflect your consideration of what is happening functionally and theoretically throughout each experiment. Consider the following questions to help you get started:

- Why did the author pick this particular dataset?
- What about the selection of the dataset makes sense? What does the selection of data tell us about the kind of questions the authors expect users to have about their data?
- There are three different approaches to estimating the likely authorship of one of the disputed essays. What are the assumptions that underlie each of the approaches to authorship attribution?
- What are three different ways that "authorship" as a concept is measured?
- Are there gendered inflections to these approaches? How so or why not? (Be sure to point to a specific cell. Give evidence from a secondary source to support your point of view.)
- How does the result of each experiment "answer" the question at hand? What appeal does the article make toward the authority or correctness of the answer?
    - Another way to ask this question is: What are the features that seem to have the strongest correlation to authorship?
- Given our discussion of what a "text" is, what might these tests of authorship be missing?
- What are the advantages and/or disadvantages to using the *Federalist Papers* as the dataset we use to study ownership of language and authorship? What is assumed? What is ignored?

Response:

The author picked this particular dataset because of its rich political and social implication as one of the most important works on political philosophy that led a the constitution which is still relevant to this day and age.

But it makes it more interesting because while most of the papers have an attributed author, there are 12 papers that have disputed authorship and makes for an interesting case to test out the authorship attribution analysis.

Since there are articles with known authorship, few with shared authorship and then some with unknown/disputed authorship, it makes senseto use this dataset and the question that naturally comes to mind is: is there a way to attribute authorship of the disputed papers by comparing it to those we already know?

As to how each method works and the way they are measured, I already have notes below for each so not repeating here.

In the tests conducted by themselves in this particular experiement, there is no direct gender inflections. Since all the texts have already been attributed to or missing attribution for a male author, gender is not a case that affects this particular dataset in question.

However, it is important to note that in any experimentation where there are mixed gendered authors, it might be an important consideration to make because writing styles can be different

across gender and we have to make sure to not choose models that will be biased based on gender and ensuring that enough diversity of authors across genders are present in a training dataset to ensure that whatever models are later used are the least biased.

THe main advantages of using this dataset is that the question being asked is very specific and we already have a solid comparison data for authors who already havea attribution for particular pieces of text. Also this is from one general set of context which is on the political philosophy subject matter of the US constitution. All of this makes this a good dataset to work with.

One major disadvantage that comes to my mind is that we already know that some of the text is already colloborated on and we also know that two of the authors have been known to have a very similar style of writing. That brings up the challenge and question of how accurate the model can be, especially when they might have been written colloboratively.

Also this was a broad topic back then and I am sure it was being discussed with others and as such being influenced by others so there is no way to understand how those influences brought about the writing making it a challenge to truly understand if you can really completely attribute a partiular text to a particular person.

What is being assumed is that the disputed writings have been written by a partiuclar author and that the statistical models being used can differentiate between the patterns and styles of different authors. Both of which can be a flaw as well and hence a source of ignorance on its own,

Directions:

- To complete this lesson, go to the Programming Historian Lesson: Introduction to sylometry with Python by François Dominic Laramée using your web browser. Keep this notebook page open on one side of your monitor and follow along with the Programming Historian lesson on the other side of your screen/monitor.
- Download the zip file in the section titled The Dataset.
- Note: This Jupyter Notebook should be saved in the same folder as your Dataset folder.
- Add new cells above and below the cells you are annotating and type your annotations in mark down.

Historical Context

As Patrick Juola writes: "Authorship Attribution is about ownership of words." What is at stake when we embark upon a process of trying to assign ownership to words?

Response:

Well the first thing that comes to my mind while trying to put an authorship attribution particularly through code where the real author isn't known is misattribution particularly because in analysis our biases are being brought out whether as the scientist conducting the project or the biases in the training models themselves and that can cause the issue of misattributing authors based on such biases.

Also there is the whole ethical as well as legal considerations of what we are studying. Is it breaching any privacy concerns and do we have the consent of whoever actually wrote a particular text? What if the author chose to stay anonymous? Are we violating their right to privacy? There is also the legal (copyright and other) considerations that we need to make where we have to ensure we have the right to access the work in the first place and coming to conclusions from it.

Also literary work can be inspired from others, there might be portions that are quoted from other authors and all such complexities in which case it becomes a more difficult topic of how much of these "words" are "owned" by the author we are attributing it to.

Now these are the negative sides that are coming to my mind but on the other hand, it can also be a rewarding process. Authorship attribution gives a context to a text and that can help understand both a historical and cultural context of a piece of work. Not only that but it can also help us understand how an author's work has evolved over time if a particular time period of their work was missing and now we can use technology to attribute it back to them.

Data Import and Preparation

```
import nltk
nltk.download('punkt')
import matplotlib
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
papers = {
    'Madison': [10, 14, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48],
    'Hamilton': [1, 6, 7, 8, 9, 11, 12, 13, 15, 16, 17, 21, 22, 23, 24,
                 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 59, 60,
```

```
                61, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77,
                78, 79, 80, 81, 82, 83, 84, 85],
    'Jay': [2, 3, 4, 5],
    'Shared': [18, 19, 20],
    'Disputed': [49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 62, 63],
    'TestCase': [64]
}
```

Download [this zip file] onto your local computer, open it, and upload all of the files in the data folder to Google Colab. You will need to open the files menu on the left and put all the uploaded files into a new directory called "data."

```
from google.colab import files

uploaded = files.upload()

for fn in uploaded.keys():
  print('User uploaded file "{name}" with length {length} bytes'.format(
      name=fn, length=len(uploaded[fn])))
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable

```
# A function that compiles all of the text files associated with a single author into
def read_files_into_string(filenames):
    strings = []
    for filename in filenames:
        with open(f'/content/drive/MyDrive/FeministTextAnalysisS23/Islam_Weekly/data/f
            strings.append(f.read())
    return '\n'.join(strings)
```

Note: I changed directory as I had already uploaded files to the folder I am using for this course

```
# Make a dictionary out of the authors' corpora
federalist_by_author = {}
for author, files in papers.items():
    federalist_by_author[author] = read_files_into_string(files)
```

```
for author in papers:
    print(federalist_by_author[author][:100])
```

```
 10

The Same Subject Continued (The Union as a Safeguard Against Domestic
Faction and Insurrection)
 1
```

```
General Introduction

For the Independent Journal. Saturday, October 27, 1787


HAMILTON

To the
 2

Concerning Dangers from Foreign Force and Influence

For the Independent Journal. Wednesday, Oct
 18

The Same Subject Continued (The Insufficiency of the Present
Confederation to Preserve the Unio
 49

Method of Guarding Against the Encroachments of Any One Department of
Government by Appealing t
 64

The Powers of the Senate

From The Independent Journal. Wednesday, March 5, 1788.

JAY

To the
```

First Stylometric Test: Mendenhall's Characteristic Curves of Composition

What is being measured? How? What assumptions are translated into measurable functions?

Response:

Here word length is being counted.

As to how that is being done, here is the breakdown of what is happening from the programming historian lesson we are following:

- It invokes nltk's word_tokenize() method to chop an author's corpus into its component tokens, i.e., words, numbers, punctuation, etc.
- It looks at this list of tokens and filters out non-words

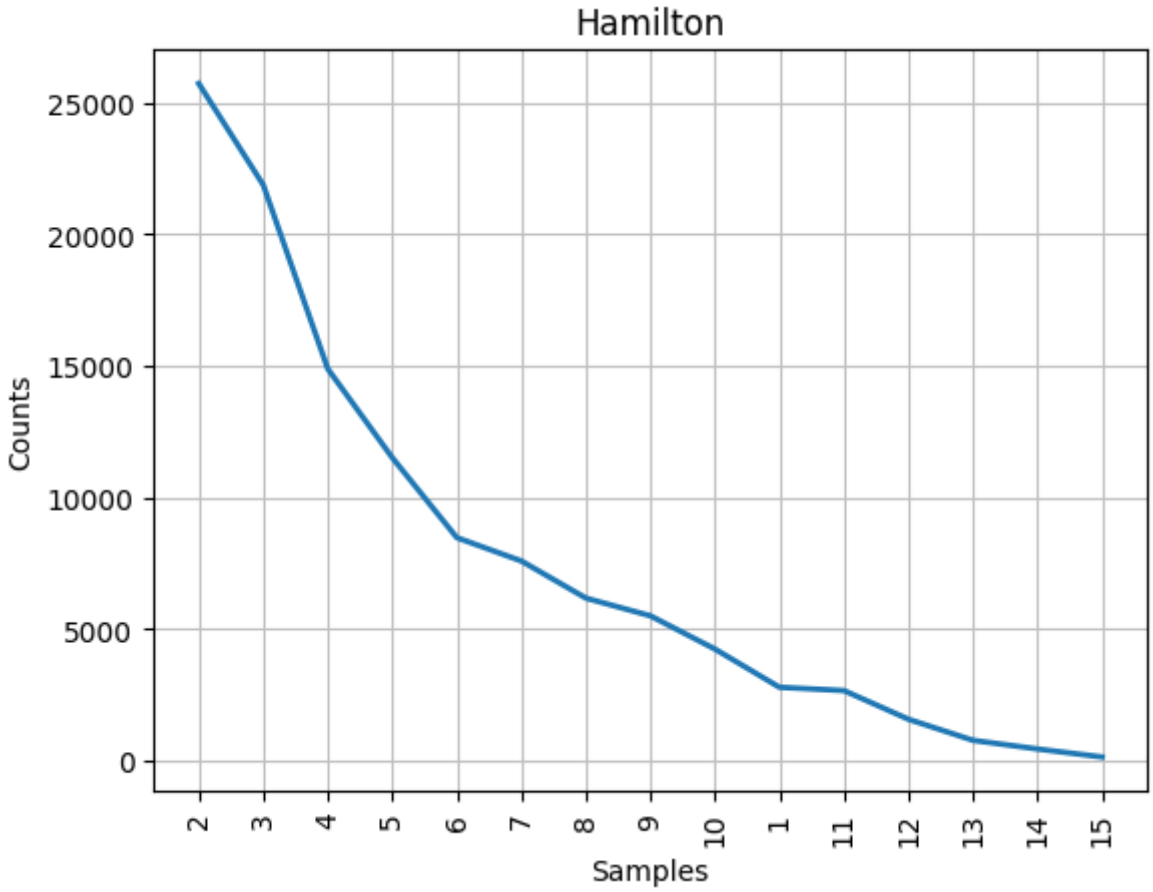
- It creates a list containing the lengths of every word token that remains
- It creates a frequency distribution object from this list of word lengths, basically counting how many one-letter words, two-letter words, etc., there are in the author's corpus.
- It plots a graph of the distribution of word lengths in the corpus, for all words up to length 15.

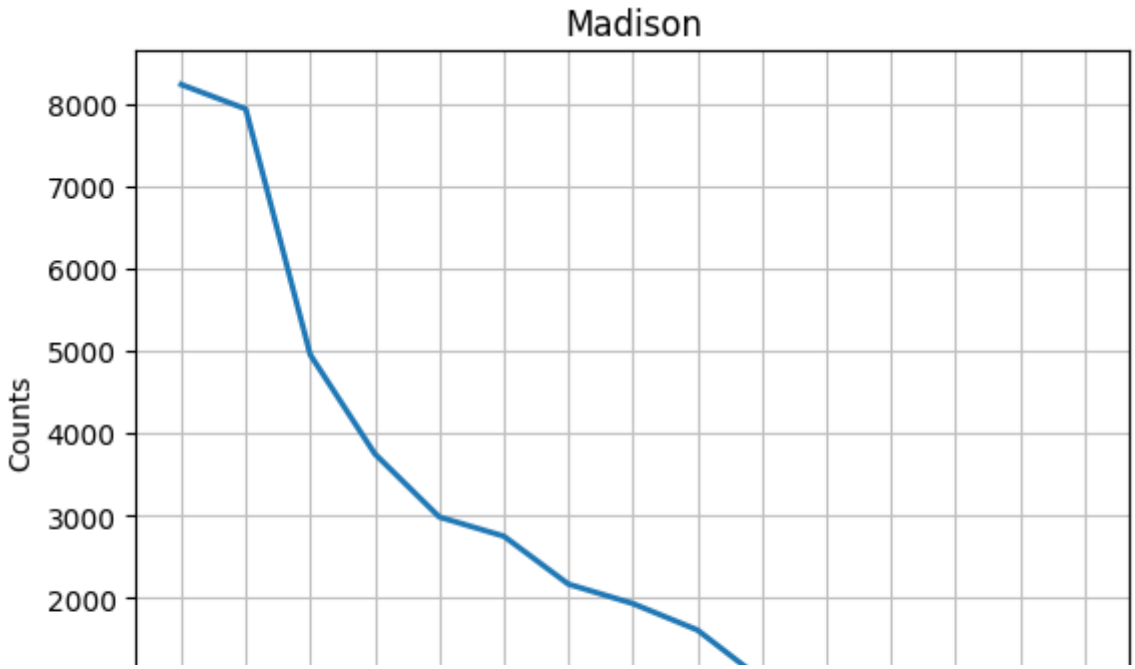
The assumption that we consider here is Mendenhall's theory that "an author's stylistic signature could be found by counting how often he or she used words of different lengths".

The comes up with its own set of challenge to me however because an author's own vocabulary and style can change over time. I am sure that my writing from 10 years ago is quite different than it is today so how do we consider that?

Also there could very well be two authors who have a quite similar writing style and vocabulary usage which would make things hard to distinguish especially if we try to take into context how much each deviates from each other and whether that would be statistically significant.

```
%matplotlib inline

# Compare the disputed papers to those written by everyone,
# including the shared ones.
authors = ("Hamilton", "Madison", "Disputed", "Jay", "Shared")

# Transform the authors' corpora into lists of word tokens
federalist_by_author_tokens = {}
federalist_by_author_length_distributions = {}
for author in authors:
    tokens = nltk.word_tokenize(federalist_by_author[author])

    # Filter out punctuation
    federalist_by_author_tokens[author] = ([token for token in tokens
                                            if any(c.isalpha() for c in token)])

    # Get a distribution of token lengths
    token_lengths = [len(token) for token in federalist_by_author_tokens[author]]
    federalist_by_author_length_distributions[author] = nltk.FreqDist(token_lengths)
    federalist_by_author_length_distributions[author].plot(15,title=author)
```





Second Stylometric Test: Kilgariff's Chi-Squared Method

What is a probability distribution? What does "distance" mean in the way that it is used here? Pay particular attention to the bulleted explanation of how the statistic is applied. In the following

section, the code is all written together. See if you can comment out what is happening at each stage and why it matters.

Response:

A probability distribution is a statistical tool that provides the likelihood (probability) of the occurence of different outcomes possible.

Here the word "distance" is used to calculate how "different" the usage of vocabularies are in two set of texts. The lower the distance between vocabularies of two text, the likelier that they are written by the same author and vice versa.

This assumes that people tend to have a pattern when it come to their vocabulary usage.

Note to self (possible source of bias):

No matter which stylometric method we use, the choice of n, the number of words to take into consideration, is something of a dark art. In the literature surveyed by Stamatatos2, scholars have suggested between 100 and 1,000 of the most common words; one project even used every word that appeared in the corpus at least twice. As a guideline, the larger the corpus, the larger the number of words that can be used as features without *running the risk of giving undue importance to a word that occurs only a handful of times*.

How many words is enough? Also is it a one size fit all model? What if different texts need different number of words to consider analyzing? Maybe some words really make a difference, but we do not notice and build a model without them?

```
# Who are the authors we are analyzing?
authors = ("Hamilton", "Madison")

# Lowercase the tokens so that the same word, capitalized or not,
# counts as one word
for author in authors:
    federalist_by_author_tokens[author] = (
        [token.lower() for token in federalist_by_author_tokens[author]])
federalist_by_author_tokens["Disputed"] = (
    [token.lower() for token in federalist_by_author_tokens["Disputed"]])

# Calculate chisquared for each of the two candidate authors
for author in authors:

    # First, build a joint corpus and identify the 500 most frequent words in it
    joint_corpus = (federalist_by_author_tokens[author] +
                    federalist_by_author_tokens["Disputed"])
```

```
joint_freq_dist = nltk.FreqDist(joint_corpus)
most_common = list(joint_freq_dist.most_common(500))

# What proportion of the joint corpus is made up
# of the candidate author's tokens?
author_share = (len(federalist_by_author_tokens[author])
                / len(joint_corpus))

# Now, let's look at the 500 most common words in the candidate
# author's corpus and compare the number of times they can be observed
# to what would be expected if the author's papers
# and the Disputed papers were both random samples from the same distribution.
chisquared = 0
for word,joint_count in most_common:

    # How often do we really see this common word?
    author_count = federalist_by_author_tokens[author].count(word)
    disputed_count = federalist_by_author_tokens["Disputed"].count(word)

    # How often should we see it?
    expected_author_count = joint_count * author_share
    expected_disputed_count = joint_count * (1-author_share)

    # Add the word's contribution to the chi-squared statistic
    chisquared += ((author_count-expected_author_count) *
                   (author_count-expected_author_count) /
                   expected_author_count)

    chisquared += ((disputed_count-expected_disputed_count) *
                   (disputed_count-expected_disputed_count)
                   / expected_disputed_count)

print("The Chi-squared statistic for candidate", author, "is", chisquared)
```

```
The Chi-squared statistic for candidate Hamilton is 3434.6850314768426
The Chi-squared statistic for candidate Madison is 1907.5992915766838
```

Third Stylometric Test: John Burrows' Delta Method (Advanced)

Note to Self (how Burrow's original algorithm works):

- Assemble a large corpus made up of texts written by an arbitrary number of authors; let's say that number of authors is x.
- Find the n most frequent words in the corpus to use as features.
- For each of these n features, calculate the share of each of the x authors' subcorpora represented by this feature, as a percentage of the total number of words. As an example, the word "the" may represent 4.72% of the words in Author A's subcorpus.

- Then, calculate the mean and the standard deviation of these x values and use them as the offical mean and standard deviation for this feature over the whole corpus. In other words, we will be using a mean of means instead of calculating a single value representing the share of the entire corpus represented by each word. This is because we want to avoid a larger subcorpus, like Hamilton's in our case, over-influencing the results in its favor and defining the corpus norm in such a way that everything would be expected to look like it.
- For each of the n features and x subcorpora, calculate a z-score describing how far away from the corpus norm the usage of this particular feature in this particular subcorpus happens to be. To do this, subtract the "mean of means" for the feature from the feature's frequency in the subcorpus and divide the result by the feature's standard deviation. Figure 7 shows the z-score equation for feature 'i', where C(i) represents the observed frequency, the greek letter mu represents the mean of means, and the greek letter sigma, the standard deviation.
- Then, calculate the same z-scores for each feature in the text for which we want to determine authorship.
- Finally, calculate a delta score comparing the anonymous paper with each candidate's subcorpus. To do this, take the average of the absolute values of the differences between the z-scores for each feature between the anonymous paper and the candidate's subcorpus. (Read that twice!) This gives equal weight to each feature, no matter how often the words occur in the texts; otherwise, the top 3 or 4 features would overwhelm everything else. Figure 8 shows the equation for Delta, where Z(c,i) is the z-score for feature 'i' in candidate 'c', and Z(t,i) is the z-score for feature 'i' in the test case.
- The "winning" candidate is the author for whom the delta score between the author's subcorpus and the test case is the lowest.

```
# Who are we dealing with this time?
authors = ("Hamilton", "Madison", "Jay", "Disputed", "Shared")

# Combine every paper except our test case into a single corpus
whole_corpus = []
for author in authors:
    whole_corpus += federalist_by_author_tokens[author]

# Get a frequency distribution
whole_corpus_freq_dist = list(nltk.FreqDist(whole_corpus).most_common(30))
whole_corpus_freq_dist[ :10 ]
```

```
[('the', 17745),
 ('of', 11795),
 ('to', 6999),
 ('and', 5012),
 ('in', 4383),
 ('a', 3957),
 ('be', 3770),
 ('that', 2739),
```

```
        ('it', 2492),
        ('is', 2177)]
```

```
# The main data structure
features = [word for word,freq in whole_corpus_freq_dist]
feature_freqs = {}

for author in authors:
    # A dictionary for each candidate's features
    feature_freqs[author] = {}

    # A helper value containing the number of tokens in the author's subcorpus
    overall = len(federalist_by_author_tokens[author])

    # Calculate each feature's presence in the subcorpus
    for feature in features:
        presence = federalist_by_author_tokens[author].count(feature)
        feature_freqs[author][feature] = presence / overall
```

```
import math

# The data structure into which we will be storing the "corpus standard" statistics
corpus_features = {}

# For each feature...
for feature in features:
    # Create a sub-dictionary that will contain the feature's mean
    # and standard deviation
    corpus_features[feature] = {}

    # Calculate the mean of the frequencies expressed in the subcorpora
    feature_average = 0
    for author in authors:
        feature_average += feature_freqs[author][feature]
    feature_average /= len(authors)
    corpus_features[feature]["Mean"] = feature_average

    # Calculate the standard deviation using the basic formula for a sample
    feature_stdev = 0
    for author in authors:
        diff = feature_freqs[author][feature] - corpus_features[feature]["Mean"]
        feature_stdev += diff*diff
    feature_stdev /= (len(authors) - 1)
    feature_stdev = math.sqrt(feature_stdev)
    corpus_features[feature]["StdDev"] = feature_stdev
```

```
feature_zscores = {}
for author in authors:
    feature_zscores[author] = {}
```

```
    for feature in features:

        # Z-score definition = (value - mean) / stddev
        # We use intermediate variables to make the code easier to read
        feature_val = feature_freqs[author][feature]
        feature_mean = corpus_features[feature]["Mean"]
        feature_stdev = corpus_features[feature]["StdDev"]
        feature_zscores[author][feature] = ((feature_val-feature_mean) /
                                            feature_stdev)
```

```
# Tokenize the test case
testcase_tokens = nltk.word_tokenize(federalist_by_author["TestCase"])

# Filter out punctuation and lowercase the tokens
testcase_tokens = [token.lower() for token in testcase_tokens
                   if any(c.isalpha() for c in token)]

# Calculate the test case's features
overall = len(testcase_tokens)
testcase_freqs = {}
for feature in features:
    presence = testcase_tokens.count(feature)
    testcase_freqs[feature] = presence / overall

# Calculate the test case's feature z-scores
testcase_zscores = {}
for feature in features:
    feature_val = testcase_freqs[feature]
    feature_mean = corpus_features[feature]["Mean"]
    feature_stdev = corpus_features[feature]["StdDev"]
    testcase_zscores[feature] = (feature_val - feature_mean) / feature_stdev
    print("Test case z-score for feature", feature, "is", testcase_zscores[feature])
```

```
Test case z-score for feature the is -0.5905131029403456
Test case z-score for feature of is -1.815053228501068
Test case z-score for feature to is 1.080722357197572
Test case z-score for feature and is 1.0546002678666273
Test case z-score for feature in is 0.7425341727883432
Test case z-score for feature a is -0.7962692057857793
Test case z-score for feature be is 1.0279650702511498
Test case z-score for feature that is 1.9604023041278147
Test case z-score for feature it is 0.21265791060592468
Test case z-score for feature is is -0.8792324482592065
Test case z-score for feature which is -2.059010144513673
Test case z-score for feature by is 1.2185982163454618
Test case z-score for feature as is 4.556093784647465
Test case z-score for feature this is -0.651311983665639
Test case z-score for feature not is 0.8424621292127045
Test case z-score for feature would is -0.8419452065894578
Test case z-score for feature for is -0.84301315697513
Test case z-score for feature have is 2.3422900648666367
Test case z-score for feature will is 1.504662365589896
```

```
Test case z-score for feature or is -0.24024581137684636
Test case z-score for feature from is -0.5012009119505166
Test case z-score for feature their is 0.8623445543648857
Test case z-score for feature with is -0.04867218094628638
Test case z-score for feature are is 7.827980222511478
Test case z-score for feature on is -0.028883899479019082
Test case z-score for feature an is -0.7141594856498873
Test case z-score for feature they is 5.360237198048704
Test case z-score for feature states is -0.7201541151578137
Test case z-score for feature government is -2.043489634686769
Test case z-score for feature may is 0.9954644116572955
```

```
for author in authors:
    delta = 0
    for feature in features:
        delta += math.fabs((testcase_zscores[feature] -
                            feature_zscores[author][feature]))
    delta /= len(features)
    print( "Delta score for candidate", author, "is", delta )
```

```
Delta score for candidate Hamilton is 1.7560432408322548
Delta score for candidate Madison is 1.5981882978381434
Delta score for candidate Jay is 1.5159420162682575
Delta score for candidate Disputed is 1.535744690035478
Delta score for candidate Shared is 1.9064655212964878
```

Conclusions and Further Reading and Resources

Consider one of the "Interesting case studies" at the end of the lesson. What are the opportunities / stakes that authorship attribution raises in each case? Are there cases when authorship attribution may not make sense to do? Are there ethical implications? How might/could authorship attribution participate in cultural or archival recovery projects?

Response:

The question says to consider one of the case first and then it talks about questions for each case so I am a bit confused if we are supposed to analyze just one or give examples separately but I have come up with the following response:

I personally really enjoyed the idea in the following case study:

"Moshe Koppel and Winter Yaron propose the “impostor method”, which attempts to determine whether two texts have been written by the same author by inserting them into a set of texts written by false candidates."

That sounds really cool because it not only is an interesting project but can test the limits of the analytical tool. Like what if there was a way that you could fool the authorship attribution method used and that can be a valid limitation. On the other hand, if the model worked well, you are proving just how accurate it can be.

About ethical implications I felt most concerned with Marcelo Luiz Brocardo, Issa Traore, Sherif Saad and Isaac Woungang's project because email and tweets are very private personal property and it seems to be to be an attack on privacy and I personally do not support any such methods because people have a right to privacy and anonymity. This project is not only problematic by itself but can be a potential dangerous solution that other people like stalkers on the web can use to cause harm.

As for the authorship attribution in cultural or archival recovery projects, it can have great implications where lost texts and those with disputed authorship or no authorship attribution at all could be clarified and it can provide better cultural and historical context of the period the works were from, providing valuable insight.