Shakespeare Sentiment Analysis

Thomas Sato

ENGL 341 Shakespeare
Professor Mike Chasar
Willamette University
May 2023

Contents

1 Introduction

Natural Language Processing (NLP) is a field at the intersection between computer science, linguistics, and artificial intelligence that deals with how computers interpret spoken/written human language. Historically, researchers have been solving natural language problems since the 1950s, investigating computer tasks ranging from automatic machine language translation to the development of ”chatbots” that respond to human commands. Today, developments in NLP have become accessible and mainstream, from things like the Stable Diffusion model[1], which generates images based on user input of text, to tools like ChatGPT, which is a text-generation chatbot that can write essays, code, and responses to just about any human prompt that it receives.

In particular, this paper will explore the subdomain of NLP called Sentiment Analysis, and how Shakespeare's plays can be analyzed from a computational perspective. This field concerns the assignment of emotion, or sentiment, to text data, in order to determine whether the emotional tone of the text is positive or negative. The application of sentiment analysis can be very useful in the business/marketing fields in particular. For example, businesses take in large amounts of textual data in the form of customer reviews, and manually analyzing these reviews can exhaust lots of resources. With sentiment analysis, the process is automated and easily summarizable, and the analysis can remove some of the human bias involved in interpreting reviews.

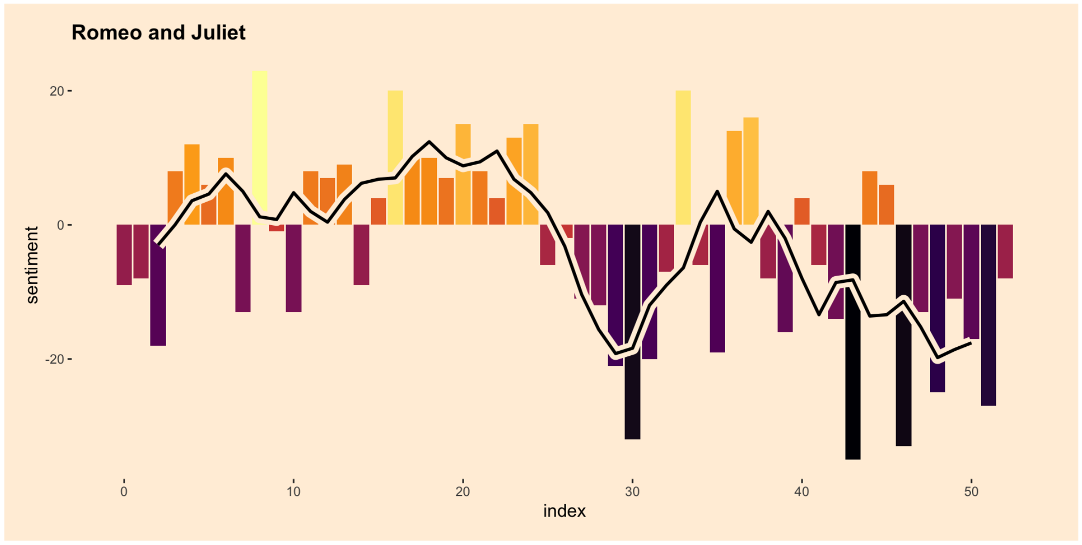


Figure 1: Distribution of positive and negative sentiment throughout Shakespeare's *Romeo and Juliet*. (Image taken from Christensen)[2]

Undoubtedly, sentiment analysis is very useful for businesses, however, the application in the literary sphere is much less common. In one online blog post by Peer Christensen[2], a simple exploratory text analysis of Shakespeare's texts was showcased (Figure 1). The author used the *tidytext*[3] package in the R programming language to create visualizations of the distributions of sentiment across all of Shakespeare's tragedies. While this article is a fantastic showcase of the tools that can be used in R to perform text analyses, the goal of this paper is to expand upon the data analysis in a way that can lend insight to a literary analysis.

This idea of this project is to program a model that will take two of Shakespeare's plays as input, and use sentiment analysis to compare the characters in both plays with each other, matching characters based on the similarity in their sentiments. I will first make a hypothesis of my own based on two plays of my choosing, and then I will compare the results to each other. The idea is that by using a computer to understand how similar characters are based on the language they use may provide a higher-level understanding of how each character uses language to convey emotion. A computer analysis may be limited in that it will not sense the nuances between how a character relates to its own environment within the play, whereas a human analysis will be able to read more closely into the emotional distance between characters in different plays. However, sentiment analysis provides a more "objective" perspective of the similarity of characters between texts, and this perspective may reveal some relationships that may not be immediately clear to the human perspective.

The rest of this paper will first outline my hypothesis and the reasoning behind the choices that were made in the process of creating it. Then, the design process for creating a computer algorithm will be explored and described in detail, and the code that goes along with the algorithm will be addressed as well. The results from the model will then be compared to the hypothesis, and any unusual findings will be reported. This paper will conclude with the limitations of the methods that were used, and some further work that could potentially be developed with the model.

2 Project Hypothesis

The two plays of Shakespeare's I will look at are *Measure for Measure* and *The Taming of the Shrew*. I will match ten characters together based on how I believe they are most similar in terms of how they interact with each other within their respective plays. It is not a perfect isomorphism, of course, and if there did exist a perfect mapping from the characters of each play, the question in this paper would not be very interesting. Therefore, it is important to note that the list in Figure 2 below is from my own understanding and interpretation of the texts.

I will now give my reasoning for choosing the matches that I did. To begin, looking at where Christopher Sly and the Lord from the Induction in *The Taming of the Shrew* fit into *Measure for Measure*, I thought that both the Duke and Lucio were characters that seemed somewhat outside of the play in certain ways. The Duke was a character that was always omniscient in a way, dressed in disguise for most of the play and listening in on conversations between the other characters, like in the prison where Claudio was kept in Act III, Scene 1. Lucio, on the other hand, felt out of the play in the same way when he talks to Isabella and encourages her to be bold to Angelo at the end of Act I. And in the same way that the Lord in the Induction has the power to play a prank on Sly, the Duke at the end of *Measure for Measure* exercises a strange power over Lucio when he passes judgement on all the other characters in the play.

Since Katherine and Bianca are the only two female characters in *The Taming of the Shrew*, I put Bianca with Mariana and did not include Juliet in the chart. I chose to match characters by gender just to simplify my own decisions in formulating my hypothesis. I think that Mariana is a unique character with regard to those in *The Taming of the Shrew*,

Measure for Measure	*The Taming of the Shrew*
Angelo	*Petruchio*
Claudio	*Lucentio*
Duke	*Lord*
Elbow	*Gremio*
Escalus	*Grumio*
Isabella	*Katherine*
Pompey	*Hortensio*
Lucio	*Christopher Sly*
Mariana	*Bianca*
Provost	*Tranio*

Figure 2: How I matched characters in *Measure for Measure* to *The Taming of the Shrew.*

because although she plays an important part in *Measure for Measure*, she has relatively few lines and her character is completely mysterious, besides the reading that her love for Angelo is so strong that no matter how he treats her she will still beg for mercy for his love and his life.

The way that Angelo and Isabella interact with each other in *Measure for Measure* struck me as quite similar to the way that Petruchio and Katherine interact in *The Taming of the Shrew.* Both Angelo and Petruchio are very forceful and assertive in their own particular ways, and Isabella and Katherine respectively are subject to this imposing of their wills onto them. Notably, however, there are the differences in character as Angelo desires Isabella for her virtue, whereas Petruchio seemingly desires Katherine for the opposite! The portrayal of Isabella's character is of this unbreakable sense of morality, but Katherine is portrayed as too aggressive for her own good, in the the world of the play. I rather believe that either Katherine or Bianca would be suitably similar to Isabella in their own ways. It turned out to be a tougher decision than I thought for who to match Isabella with, because my feeling is that in terms of how characters are related to each other within the plays, Angelo and Petruchio are similar to Isabella and Katherine. However, I believe that the personalities of Bianca and Isabella would be more similar than that of Katherine and Isabella. I would venture to suggest that perhaps a human would be more likely to match Katherine and Isabella, but, based on language and emotion, that a computer would be more likely to match Bianca to Isabella.

To continue the comparison between Angelo and Petruchio, I matched Escalus with Grumio, since both are second-in-command to each of the respective rulers. Since Pompey and Elbow are both comedic relief characters, I looped them in with Hortensio and Gremio, the suitors to Bianca. In certain ways, Hortensio and Gremio are rivals to each other just as Pompey and Elbow are in the brief scene that includes both of them. Pompey seems to me to have more control and a more significant role in the play than Elbow has, just as Hortensio has a more significant role than Gremio.

Lastly, the Provost and Claudio are rather minor characters in *Measure for Measure*, but I notice that in many of the scenes, the Provost and Claudio will be together in the prison.

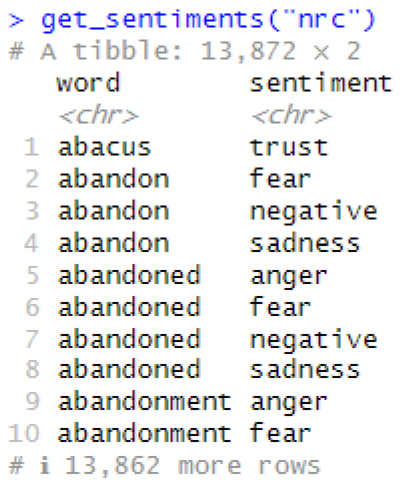
Although there may not be any direct relationship between the Provost and Claudio, there is with Lucentio and Tranio, as Lucentio is the main suitor to Bianca, and Tranio is the servant and friend to Lucentio. In certain ways both Claudio and Lucentio have their own trials with regards to love.

Again, there are plenty of other valid comparisons that could be made between the two texts, and this one is not perfect, but for the purposes of this project, it will be sufficient to compare the computer results to one human result.

3 Design Process

3.1 The NRC Lexicon

For a simple sentiment analysis project like this, the first step is deciding what kind of lexicon to use. The *tidytext* package in R provides four lexicons, which are the lists of words and their associated valence (positive, negative) (Figure 3). Rather than comparing characters based solely on the positive or negative sentiments in their texts, the NRC[4] (National Research Council of Canada) lexicon provides eight distinct categories of emotion, as follows: “Trust,” “Fear,” “Sadness,” “Anger,” “Surprise,” “Disgust,” “Joy,” and “Anticipation.” This model is based off of Plutchik’s Wheel of Emotions, where joy and sadness, trust and disgust, fear and anger, and anticipation and surprise are all diametrically opposed to each other. The lexicon also provides positive and negative sentiment for each word, but this paper will focus on the emotion categories.



```
> get_sentiments("nrc")
# A tibble: 13,872 × 2
   word        sentiment
   <chr>       <chr>
 1 abacus      trust
 2 abandon     fear
 3 abandon     negative
 4 abandon     sadness
 5 abandoned   anger
 6 abandoned   fear
 7 abandoned   negative
 8 abandoned   sadness
 9 abandonment anger
10 abandonment fear
# ℹ 13,862 more rows
```

Figure 3: First ten rows of the NRC dataset. Each word has an associated emotion and valence.

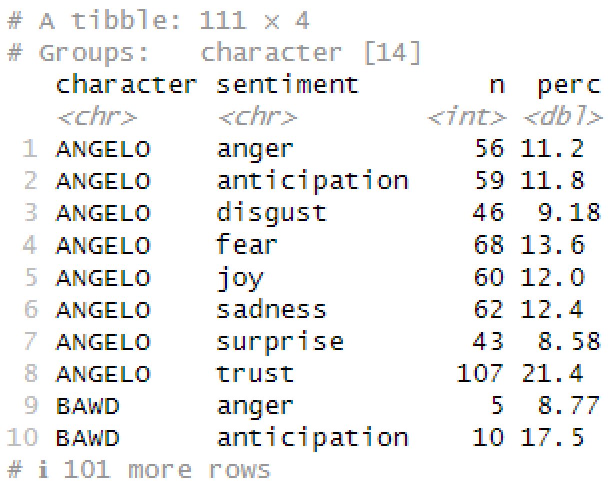
Before using it for an analysis, it is important to understand where this lexicon comes from and if it is appropriate to integrate into the project. Some lexicons are manually created by individuals who go through each word in the given lexicon and assign each word a positive or negative sentiment. However, I would suggest that the NRC lexicon is more appropriate to use, because in the study in which it was compiled, a crowdsourcing method was used to gather and group together input data from lots of different sources.

The researchers used Amazon’s Mechanical Turk service as a platform to gather mass amounts of input data. The service allows users to get paid for doing tasks like answering surveys. The survey consists of an extensive questioning about the associations between each word and the eight emotions to annotate. They found in a pilot study that annotators were more likely to agree when the survey asked whether the words were *associated* with the emotions, rather than if the words *evoked* certain emotions. In the end, over 8,000 individual terms were tagged, with an average of about 4.4 annotators for each word.

Obviously, there are issues with using a public method like this, when people may choose to not answer faithfully, but in the study controls were implemented to attempt to catch and nullify these responses to the survey. On a larger scale, computational models that aim to capture human emotion will always be subject to human biases, since emotion is an inherently subjective topic that will contain lots of variability from person to person. These limitations will be discussed as well at the end of the paper, but it is important to keep in mind that analyses with emotion data like this are not perfectly objective. Although a sample size of 4.4 annotators per word is relatively small, the variety of categories lends well to a similarity analysis like the one in this paper, and the crowdsourcing method does limit bias compared to a model that is created by only one individual.

3.2 The Mathematics

Now that we have the lexicon that we are going to use, and the texts that we want to analyze, the goal is to figure out a way to mathematically understand what it means to match characters with each other. When the words in Shakespeare's plays are joined together with their associated sentiments, the output is every character in the play, the eight categories associated with each character, and how many times a word appears in that character's text, that is associated with that emotion. For example, in *Measure for Measure*, throughout Angelo's text in the play, 68 words that were spoken were associated with fear in the NRC lexicon. There were 501 tags in Angelo's text that were associated with words in the lexicon, meaning a given category was counted towards Angelo's character 501 times. A given word may have multiple emotions associated with it.



```
# A tibble: 111 × 4
# Groups:   character [14]
   character sentiment        n  perc
   <chr>     <chr>        <int> <dbl>
 1 ANGELO    anger           56 11.2
 2 ANGELO    anticipation    59 11.8
 3 ANGELO    disgust         46  9.18
 4 ANGELO    fear            68 13.6
 5 ANGELO    joy             60 12.0
 6 ANGELO    sadness         62 12.4
 7 ANGELO    surprise        43  8.58
 8 ANGELO    trust          107 21.4
 9 BAWD      anger            5  8.77
10 BAWD      anticipation    10 17.5
# ℹ 101 more rows
```

Figure 4: Example output of sentiment analysis for *Measure for Measure*

Because some characters in a given play speak more words than others, it makes sense to take the proportion of each category. A data table is given in Figure 4 in order to understand better what the output will look like.

Mathematically, what we have is a collection of probability vectors, which are a collection of numbers whose components sum up to one. For example, the probability vector for Angelo is the vector that consists of the probabilities

$$< 11.2, 11.8, 9.18, 13.6, 12.0, 12.4, 8.58, 21.4 >,$$

each associated with a particular emotion. Additionally,

$$11.2 + 11.8 + 9.18 + 13.6 + 12.0 + 12.4 + 8.58 + 21.4 \approx 100,$$

when accounting for some rounding errors. For each character in each play, we have eight probabilities, and the goal is to find a way to compare the vectors to each other. After searching for the most appropriate similarity metric to compare the vectors, and after consulting with the math department (thanks to Professor Otto!), what was determined to be most appropriate for finding the distance between probability vectors is the **Total Variation Distance**, defined by

$$||\mu - \nu||_{TV} = \frac{1}{2} \sum_{i \in S} |\mu(i) - \nu(i)|,$$

where μ and ν are probability distributions defined on the same state space S. The distance is between two probability vectors, one for each character we would like to compare, and the state space is the eight emotional categories defined by the NRC lexicon. So for each emotion, the absolute value of the difference in each emotion is added together, then divided by two at the end. For a given pair of characters, a lower total variation distance is associated with more similarity, and a higher total variation distance with less similarity.

Now, once we have the total variation distance for every pair of characters in the two plays we are examining, one logical way to view each of these pairs is in a matrix, which is just a collection of numbers in a grid-like format. Each row in the matrix will correspond to a character in one play, and each column will correspond to a character in the other. Every entry in the matrix will be the total variation distance between the characters in the corresponding row and column. As a made-up example in Figure 5, take this matrix between two characters, each in one of the plays:

$$\begin{array}{r|cc|} & \text{Katherine} & \text{Petruchio} \\ \text{Angelo} & 7.50 & 5.00 \\ \text{Isabella} & 15.50 & 25.00 \end{array}$$

Figure 5: Example of a matrix that has the total variation distance between characters, which I will call the total variation distance matrix.

In this example, the two characters who are most similar are Angelo and Petruchio, and the total variation distance between them in terms of the emotion in the language they use is 5.00. Isabella and Petruchio are the least similar.

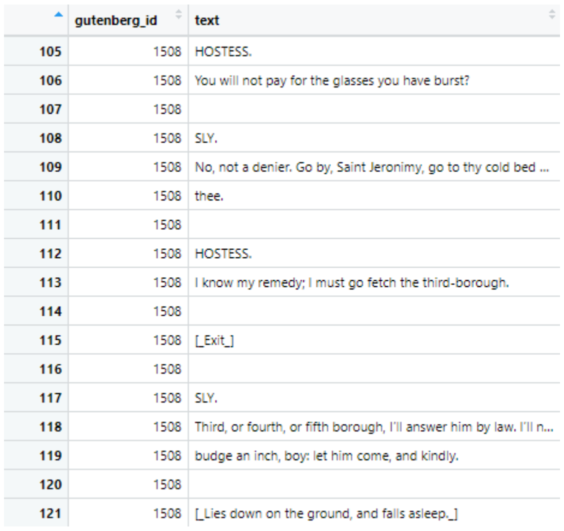
Once we have the total variation distance matrix for every character in each of the two plays, the last step is to match the characters together based on the values in the matrix. There are multiple methods and algorithms to complete this task, but the simplest way is to find the lowest value in the matrix, which are the two most similar characters, and delete the row and column associated with those two characters. Then repeat the process for the resulting matrix until there is only a 1×1 (one row by one column) matrix with the last possible match.

By using this algorithm, the most similar characters are matched first. This method works well because it does not matter which play corresponds to the rows and which one corresponds to the columns. If there was no particular order that the minimum total variation distance was found, there may be arbitrary results. For example, in Figure 5, Katherine is most similar to Angelo going by column, but Angelo is most similar to Petruchio, going by row. As a side note, another advantage of the matrix format in this context is that, given any individual character in a play in the rows, the most similar character in the other play is the one corresponding to the column associated with the lowest value in that row.

In the end, however, the result is as desired: A match for each character in each play based on a similarity metric, based on the sentiments in the language that they use.

4 Model Implementation

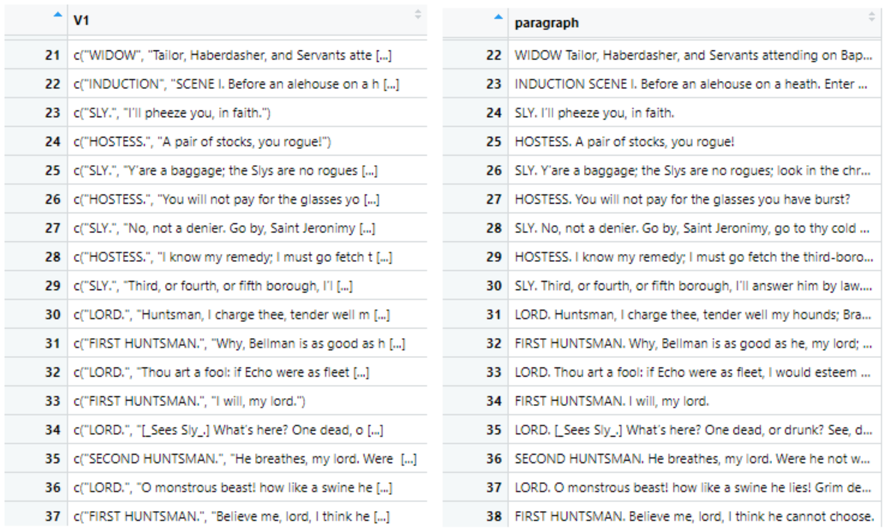
This section will now examine the code that was used to implement the algorithms described in the previous section. Like most data science projects, the part of the project that took the most time and effort to program was the data wrangling, which is getting the data in a format that is easily usable for analysis. Shakespeare's plays are more easily accessible in R in the *gutenbergr*[5] package, which gives access to many full length books and texts. Each line in Shakespeare's play is an observation in a column (Figure 6).



	gutenberg_id	text
105	1508	HOSTESS.
106	1508	You will not pay for the glasses you have burst?
107	1508	
108	1508	SLY.
109	1508	No, not a denier. Go by, Saint Jeronimy, go to thy cold bed ...
110	1508	thee.
111	1508	
112	1508	HOSTESS.
113	1508	I know my remedy; I must go fetch the third-borough.
114	1508	
115	1508	[_Exit_]
116	1508	
117	1508	SLY.
118	1508	Third, or fourth, or fifth borough, I'll answer him by law. I'll n...
119	1508	budge an inch, boy: let him come, and kindly.
120	1508	
121	1508	[_Lies down on the ground, and falls asleep._]

Figure 6: Part of the dataset from *The Taming of the Shrew* downloaded from *gutenbergr*.

If the reader is familiar with the R programming language, I will refer to Appendix A from now on when discussing specific chunks of code that serve a particular purpose in the analysis. For all of the code in the full functional program, see Appendix B.



	V1
21	c("WIDOW", "Tailor, Haberdasher, and Servants atte [...]
22	c("INDUCTION", "SCENE I. Before an alehouse on a h [...]
23	c("SLY.", "I'll pheeze you, in faith.")
24	c("HOSTESS.", "A pair of stocks, you rogue!")
25	c("SLY.", "Y'are a baggage; the Slys are no rogues [...]
26	c("HOSTESS.", "You will not pay for the glasses yo [...]
27	c("SLY.", "No, not a denier. Go by, Saint Jeronimy [...]
28	c("HOSTESS.", "I know my remedy; I must go fetch t [...]
29	c("SLY.", "Third, or fourth, or fifth borough, I'l [...]
30	c("LORD.", "Huntsman, I charge thee, tender well m [...]
31	c("FIRST HUNTSMAN.", "Why, Bellman is as good as h [...]
32	c("LORD.", "Thou art a fool: if Echo were as fleet [...]
33	c("FIRST HUNTSMAN.", "I will, my lord.")
34	c("LORD.", "[_Sees Sly_.] What's here? One dead, o [...]
35	c("SECOND HUNTSMAN.", "He breathes, my lord. Were [...]
36	c("LORD.", "O monstrous beast! how like a swine he [...]
37	c("FIRST HUNTSMAN.", "Believe me, lord, I think he [...]

	paragraph
22	WIDOW Tailor, Haberdasher, and Servants attending on Bap...
23	INDUCTION SCENE I. Before an alehouse on a heath. Enter ...
24	SLY. I'll pheeze you, in faith.
25	HOSTESS. A pair of stocks, you rogue!
26	SLY. Y'are a baggage; the Slys are no rogues; look in the chr...
27	HOSTESS. You will not pay for the glasses you have burst?
28	SLY. No, not a denier. Go by, Saint Jeronimy, go to thy cold ...
29	HOSTESS. I know my remedy; I must go fetch the third-boro...
30	SLY. Third, or fourth, or fifth borough, I'll answer him by law....
31	LORD. Huntsman, I charge thee, tender well my hounds; Bra...
32	FIRST HUNTSMAN. Why, Bellman is as good as he, my lord; ...
33	LORD. Thou art a fool: if Echo were as fleet, I would esteem ...
34	FIRST HUNTSMAN. I will, my lord.
35	LORD. [_Sees Sly_.] What's here? One dead, or drunk? See, d...
36	SECOND HUNTSMAN. He breathes, my lord. Were he not w...
37	LORD. O monstrous beast! how like a swine he lies! Grim de...
38	FIRST HUNTSMAN. Believe me, lord, I think he cannot choose.

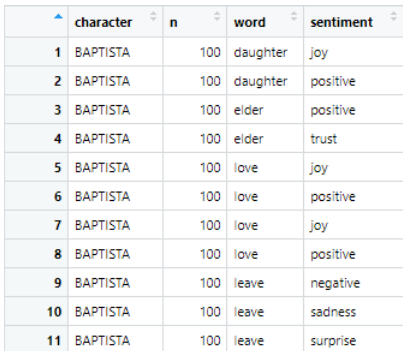
Figure 7: Vector of every paragraph in *The Taming of the Shrew* (left), and the same vector expanded and cleaned up (right)

The first task in this project is to group all of the text in a play by the character that is speaking. This was made much easier using the *gutenbergr* package because whenever a new character starts to speak, the name of the character is displayed on a separate line in all capital letters, with a period at the end, which is the same format in all of the Shakespeare plays. This standardized format made it a lot easier to automate the grouping of the text together.

The way I chose to group the text together was to go through each line of the text, and label each line by paragraph (Figure A.2). Every time the program noticed that it was a character speaking, it increased the paragraph number by one, so that every line in the play had a corresponding paragraph number. Before labeling by paragraph, however, it was necessary to remove all of the stage directions, since they were not spoken by characters, and remove the empty lines in the text as well, which are just rows in the dataset that contain empty strings (this just means no text data). The process of removing the stage directions and white space is in Figure A.1.

In order to group by character, first I needed to group by paragraph number (Figure A.3). This will give me a vector of every paragraph in the play, which looks something like Figure 7. Once each entry in the vector is collapsed into a single string of characters, it can be separated into two columns: One for the character, and one for the paragraphs themselves (A.4)

At this point, finally, we can use the *unnest_tokens* function from the *tidytext* package on the dataset, which lengthens the dataset even more by splitting the text column into one observation for every word in the paragraph (Figure A.5). Then, since the data is in the format of one column for the character, and an associated column for every word that is spoken by the character, we can now join together the NRC lexicon in order to have a complete dataset of characters and their associated sentiments for the words that they speak (Figure 8).



	character	n	word	sentiment
1	BAPTISTA	100	daughter	joy
2	BAPTISTA	100	daughter	positive
3	BAPTISTA	100	elder	positive
4	BAPTISTA	100	elder	trust
5	BAPTISTA	100	love	joy
6	BAPTISTA	100	love	positive
7	BAPTISTA	100	love	joy
8	BAPTISTA	100	love	positive
9	BAPTISTA	100	leave	negative
10	BAPTISTA	100	leave	sadness
11	BAPTISTA	100	leave	surprise

Figure 8: First 10 rows of the joined dataset from *The Taming of the Shrew*

Once we have the complete dataset, we can group by the character name in the "character" column, then calculate the sum and percentage that each category in the "sentiment" column appears. Once this is done, there is one more step of data transformation to get a probability vector for every character before constructing the total variation distance matrix (Figure A.6).

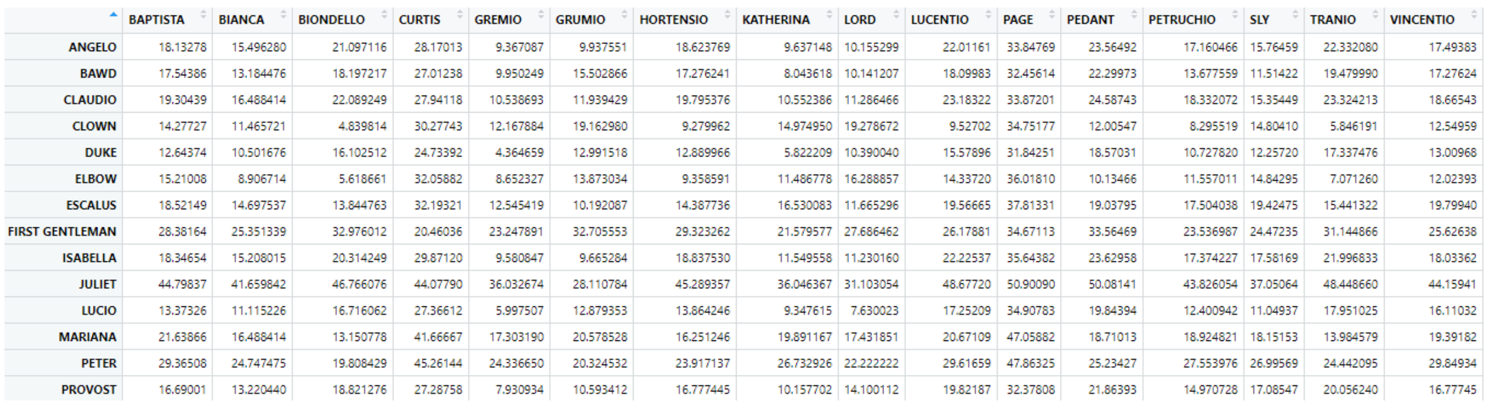
Once the total variation distance matrix is created, removing the columns and rows for the smallest total variation distance and matching the characters becomes quite simple (Figure A.7)! One thing, however, is that the number of characters in one play may be different than the number of characters in another. The algorithm for determining the matches is best defined on a square matrix, that is, a matrix with the same number of rows as columns. In practice, however, we can run the algorithm until the play with less characters has no more left to match. That way, the least similar characters in the play with more of them will not be matched, and the more similar ones will.

5 Results

5.1 Character Matches

By working with the datasets from the *gutenbergr* package, the data cleaning/transformation process is able to be automated. That means that all of the code that is included in Appendix B is able to run with not only *Measure for Measure* and *The Taming of the Shrew*, but all of Shakespeare's plays in the *gutenbergr* library, and in fact, any play that shares the same formatting as the plays in *gutenbergr*. The *text_transformation* function that was produced over the course of this project takes as input the name of any play in the format described above, and returns a dataset with all of the characters and the proportions of each sentiment categogry. The total variation distance function is also defined on two probability vectors, *mu* and *nu*. Then, the *shakespearecomparison* function will calculate the total variation distance matrix.

Figure 9 is the total variation distance matrix comparing *Measure for Measure* to *The Taming of the Shrew*. The total variation distances between characters range between 4.365 and 50.901 percent. Some considerations to be made in this example in particular is that some characters like the First Gentleman in *Measure for Measure* and the Pedant in *The Taming of the Shrew* are very minor characters, and have very few lines, yet are included



	BAPTISTA	BIANCA	BIONDELLO	CURTIS	GREMIO	GRUMIO	HORTENSIO	KATHERINA	LORD	LUCENTIO	PAGE	PEDANT	PETRUCHIO	SLY	TRANIO	VINCENTIO
ANGELO	18.13278	15.496280	21.097116	28.17013	9.367087	9.937551	18.623769	9.637148	10.155299	22.01161	33.84769	23.56492	17.160466	15.76459	22.332080	17.49383
BAWD	17.54386	13.184476	18.197217	27.01238	9.950249	15.502866	17.276241	8.043618	10.141207	18.09983	32.45614	22.29973	13.677559	11.51422	19.479990	17.27624
CLAUDIO	19.30439	16.488414	22.089249	27.94118	10.538693	11.939429	19.795376	10.552386	11.286466	23.18322	33.87201	24.58743	18.332072	15.35449	23.324213	18.66543
CLOWN	14.27727	11.465721	4.839814	30.27743	12.167884	19.162980	9.279962	14.974950	19.278672	9.52702	34.75177	12.00547	8.295519	14.80410	5.846191	12.54959
DUKE	12.64374	10.501676	16.102512	24.73392	4.364659	12.991518	12.889966	5.822209	10.390040	15.57896	31.84251	18.57031	10.727820	12.25720	17.337476	13.00968
ELBOW	15.21008	8.906714	5.618661	32.05882	8.652327	13.873034	9.358591	11.486778	16.288857	14.33720	36.01810	10.13466	11.557011	14.84295	7.071260	12.02393
ESCALUS	18.52149	14.697537	13.844763	32.19321	12.545419	10.192087	14.387736	16.530083	11.665296	19.56665	37.81331	19.03795	17.504038	19.42475	15.441322	19.79940
FIRST GENTLEMAN	28.38164	25.351339	32.976012	20.46036	23.247891	32.705553	29.323262	21.579577	27.686462	26.17881	34.67113	33.56469	23.536987	24.47235	31.144866	25.62638
ISABELLA	18.34654	15.208015	20.314249	29.87120	9.580847	9.665284	18.837530	11.549558	11.230160	22.22537	35.64382	23.62958	17.374227	17.58169	21.996833	18.03362
JULIET	44.79837	41.659842	46.766076	44.07790	36.032674	28.110784	45.289357	36.046367	31.103054	48.67720	50.90090	50.08141	43.826054	37.05064	48.448660	44.15941
LUCIO	13.37326	11.115226	16.716062	27.36612	5.997507	12.879353	13.864246	9.347615	7.630023	17.25209	34.90783	19.84394	12.400942	11.04937	17.951025	16.11032
MARIANA	21.63866	16.488414	13.150778	41.66667	17.303190	20.578528	16.251246	19.891167	17.431851	20.67109	47.05882	18.71013	18.924821	18.15153	13.984579	19.39182
PETER	29.36508	24.747475	19.808429	45.26144	24.336650	20.324532	23.917137	26.732926	22.222222	29.61659	47.86325	25.23427	27.553976	26.99569	24.442095	29.84934
PROVOST	16.69001	13.220440	18.821276	27.28758	7.930934	10.593412	16.777445	10.157702	14.100112	19.82187	32.37808	21.86393	14.970728	17.08547	20.056240	16.77745

Figure 9: Total variation distance matrix for *Measure for Measure* (rows) and *The Taming of the Shrew* (columns).

in the analysis. This result from the model considers characters that have more than 15 sentiments assigned to their texts, which is a value that can be tuned depending on how limited the user wishes the model to be in terms of the number of characters. In this case, the data may be inaccurate, as some characters like the Page, who may be significant enough to warrant wanting him to be included in the analysis, have no observations in the "anger" category, which will not show up in the result from the *text_transformation* function.

Now we will compare the results from the computer analysis to the human results from my own analysis. Below is both of the matches for characters side-by-side (Figure 10). The computer analysis is listed first, and it is listed in order of which matches were determined to be most similar.

Computer Analysis		*Human Analysis*	
Measure for Measure	*The Taming of the Shrew*	*Measure for Measure*	*The Taming of the Shrew*
Duke	*Gremio*	*Duke*	*Lord*
Pompey	*Biondello*	*Pompey*	*Hortensio*
Elbow	*Tranio*	*Elbow*	*Gremio*
Lucio	*Lord*	*Lucio*	*Christopher Sly*
Mistress Overdone	*Katherine*	*Mariana*	*Bianca*
Isabella	*Grumio*	*Isabella*	*Katherine*
Provost	*Bianca*	*Provost*	*Tranio*
Escalus	*Hortensio*	*Escalus*	*Grumio*
Claudio	*Christopher Sly*	*Claudio*	*Lucentio*
Angelo	*Petruchio*	*Angelo*	*Petruchio*

Figure 10: Comparison between the computer analysis of characters' sentiments and my analysis of characters' intertextual emotional relationships.

From this computer analysis, there are many surprising results. I will now go through

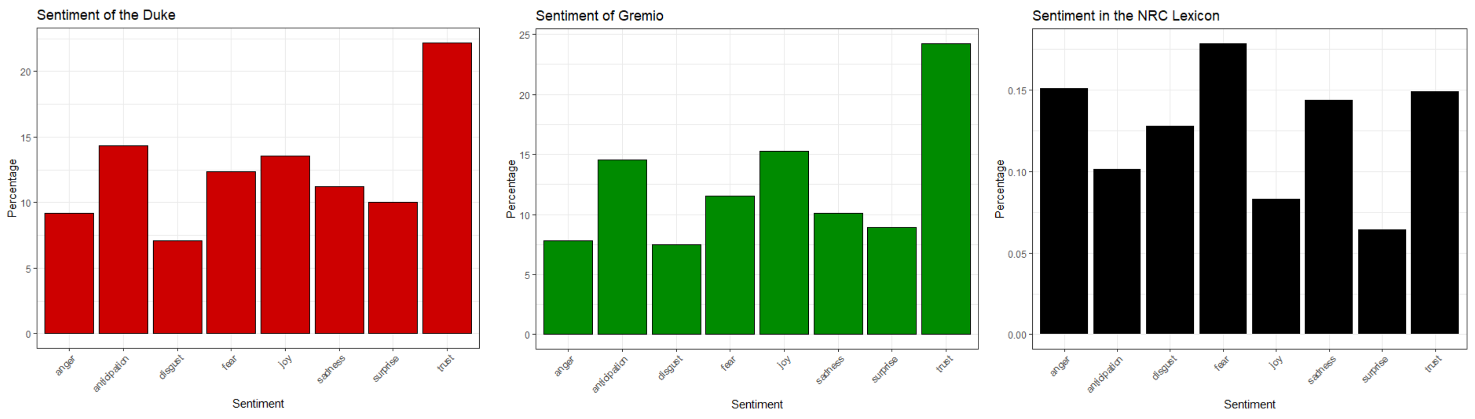


Figure 11: Distributions of sentiment categories for the Duke, Gremio, and the NRC lexicon.

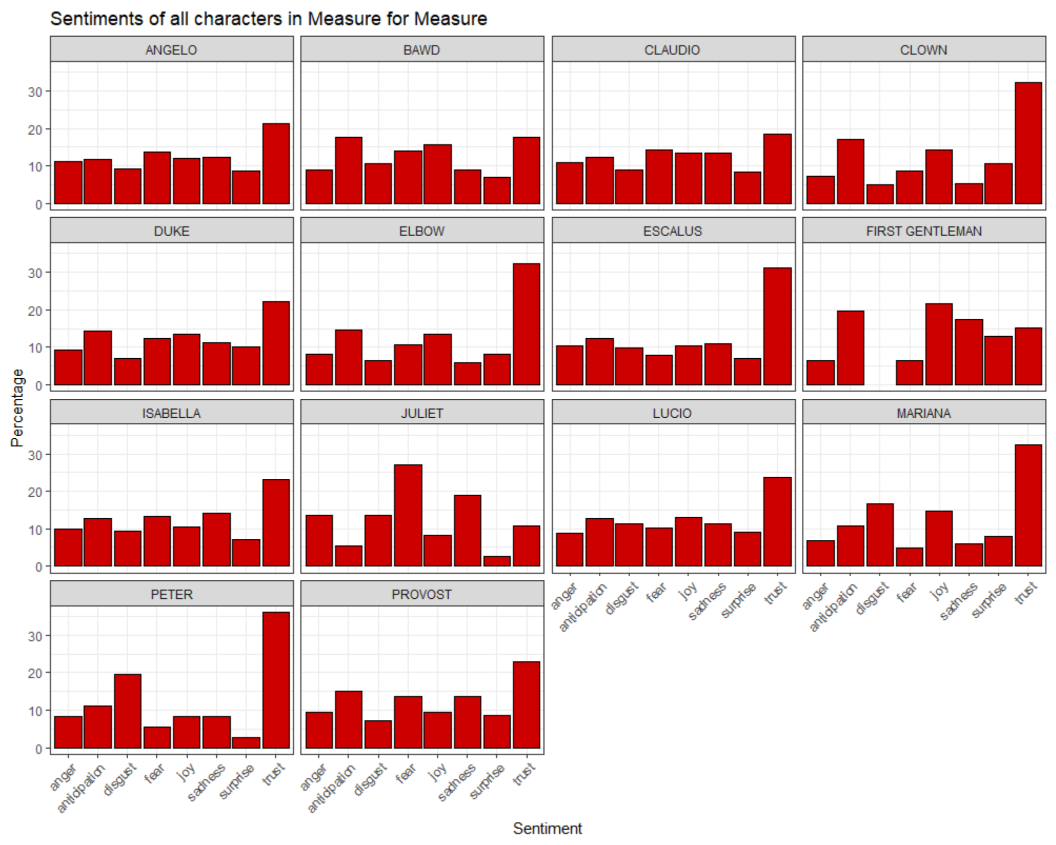
each pairing and attempt to understand this computer analysis from a human perspective.

5.2 Results Interpretation

The first of the results that was most surprising to me was that the Duke and Gremio were the two most similar characters according to the model. Below the two distributions of category are compared, along with the distribution of categories in the NRC lexicon (Figure 11). While both the Duke and Gremio have disproportionate amounts of language associated with trust, this also holds true for nearly every character that has many lines. My suspicion is that words that happen to be associated with trust in the lexicon also happen to be more common words. The word "lord" appears 81 times in *Measure for Measure*, and it is associated with trust in the NRC lexicon, for example. This phenomenon also occurs in Christensen's visualizations of Shakespeare's tragedies as well[2].

Visually speaking, the distributions of sentiments of the Duke and Gremio are very similar. However, what I notice looking at the distributions of categories of all of the characters in *Measure for Measure* and *The Taming of the Shrew* is that both the Duke and Gremio have higher proportions of language involving anticipation than the other characters in their respective plays, with a little under 15% for both of them, which may lend towards their having a higher total variation distance with respect to other characters. Thinking about the images of either character, I think it makes sense that Gremio would have lots of language associated with anticipation, as he is portrayed as kind of a creepy old man who is making lots of plans with Hortensio in order to court Bianca. The anticipation that Gremio may feel throughout the story may be caused by his association with the schemes he makes with Hortensio, and the anticipation of these schemes reflected in the language that he uses. The Duke, on the other hand, also is a character that seems to be involved in these odd schemes, when the Duke disguises himself as a friar in order to keep watch over Angelo. I can imagine these similarities between the anticipatory language when constructing schemes is what the analysis picks up on. In this way, it is interesting to note the similarities between the relatively minor character of Gremio and the quite significant character of the Duke.

Now, one interesting pairing of characters in that of Pompey and Elbow being paired with Biondello and Tranio, respectively. What is interesting to me is noting the relationship that these characters have between each other within the plays. In *Measure for Measure*,



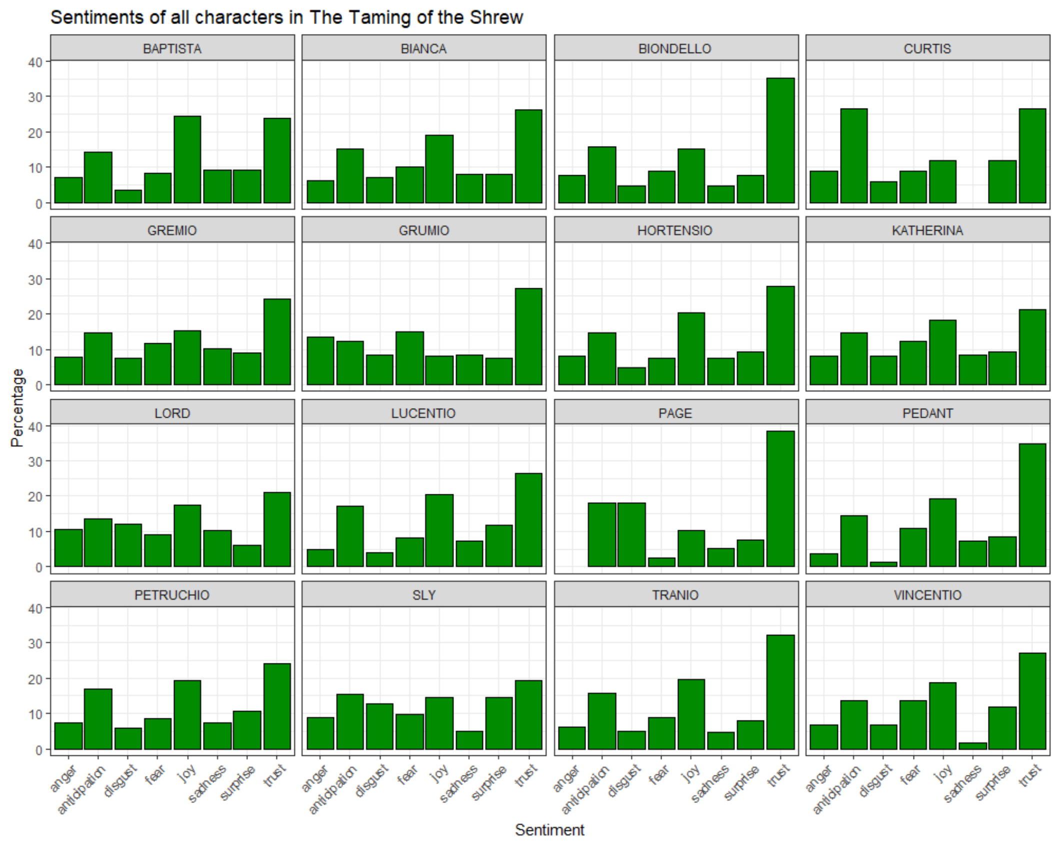


Figure 12: Distributions of sentiment for every character in *Measure for Measure* and *The Taming of the Shrew*

Pompey and Elbow show up for the most part in only one scene in which they interact with each other. Pompey being the clown that is caught working for the hostess of a brothel, and Elbow the constable that brings him to Angelo for the crime he has committed. Biondello and Tranio in *The Taming of the Shrew* are both servants to Lucentio, but they have a more long-term role in the play, not quite that of comic relief in one scene like Pompey and Elbow. Biondello serves as a sort of messenger throughout the play, coming in and out of the scenes, whereas Tranio disguises as Lucentio in order for Lucentio to disguise himself as a schoolmaster. It is difficult to see the connection between these two sets of characters. I suspect that part of why Biondello is matched with Pompey is because Biondello has very few lines compared to Pompey, which may give more weight to the sentiments of the few significant words that Biondello does have, making it more likely that Biondello matches with Pompey just by random chance alone. The same reasoning can be applied to Elbow and Tranio, as Elbow has very few lines as well. Despite the few lines that both Biondello and Elbow have, they are memorable enough characters while reading the plays. The computer analysis, however, only considers the quantifiable data from the text that is available to it. This is another limitation of this model, and generally speaking of many machine learning and artificial intelligence models as well, that with few lines there is not enough data to perform a thorough analysis, and the model will be susceptible to inaccurate results by underfitting of the data it is trained on.

The Provost and Bianca is another pairing which I believe falls into the category of perhaps it was due to random chance that these characters were paired together. It is my determination that Bianca and the Provost do not have enough in common to warrant a matching, in the human sense. As a second-in-command to the Duke, the Provost does not have many lines that would indicate a strong emotional inclination or character development within the context of the play, whereas Bianca plays a very significant symbolic role in *The Taming of the Shrew* as the submissive/desirable counterpart to Katherine.

From how I understood the texts, Lucio and the Lord from the Induction are a match that makes a lot more sense, though different from how I made the matches in my own hypothesis. Both the Lord and Sly are troublemakers in their own ways, in that Sly causes trouble breaking glasses in the first few lines of the play, and the Lord wants to play a prank on Sly for doing so. This is where I see the similarity between the Lord and Lucio. Lucio also seems like a character who wants to cause trouble, by whispering in the ear of Isabella, along with lying both to the Duke to his face, and to the Duke disguised as a friar. What I notice is that both the Lord and Lucio have relatively uniform distributions of category save for the ”trust” category, meaning that each category has the same proportion of sentiments. For the Lord, I would suppose that, in the Induction when he is directing his huntsmen to trick Sly, he is using very elaborate and descriptive language, which would mark lots of sentiments for every category. For Lucio, I would suppose that the positive emotions in the first parts of the play balance out the negative language in Act V, when he is punished.

Now, for another surprising result from how the female characters were matched, looking at the charts for Mistress Overdone and Katherine, both characters have relatively little language associated with trust, compared to the other characters in the plays, and both had higher associations with anticipation and joy. For the few lines that Mistress Overdone has in the play, she is worried about the new laws that are being enforced regarding her brothels, which may explain the anticipation. As for Katherine, I can imagine the stress from

Petruchio would cause one to become very worried in their language, but again, it is difficult to discern similarities between the two characters. The similarity between Katherine and Mistress Overdone perhaps may come from the way they are treated in each play. Mistress Overdone is dismissed mostly for her role as a hostess of a brothel, and Katherine is dismissed as being "shrewish" by the male characters in the play. It may be a stretch, but perhaps the anticipatory language that each character uses in reacting to this kind of dismissal is picked up by the computer analysis, which explains the similarity between the two. What is more mysterious is the high proportion of joy-related sentiments from each of the characters. One would expect especially for Katherine that the proportion of anger-associated language would be much higher than it is, based on how the other characters perceive her within the play.

Isabella being matched with Grumio is another surprising result from the computer analysis. Looking at the similarities in the distributions for Isabella and Grumio, Grumio has a higher proportion of sentiment associated with fear, with joy, sadness, and surprise being lower. Being Petruchio's main servant and subject to the abuse in *The Taming of the Shrew*, this makes sense. The similarity between Grumio and Isabella comes from this position of subservience and victimhood. Grumio suffers abuse from Petruchio much in a similar way that Isabella suffers abuse from Angelo, though perhaps not as directly. In some sense, it may be reasonable to match Isabella and Grumio by their language use.

Next, Escalus and Hortensio were matched together. At this point in the analysis, some of the diffreences in distribution become clearer, as Hortensio is a character who has a much higher proportion of language associated with joy than Escalus. Both of the characters, however, have quite a high proportion of sentiment associated with trust. This would likely make sense for Escalus, as a merciful counterpart to Angelo, and perhaps Hortensio as well when he attempts to court Bianca. There is a parallel to be made in this matching between Escalus and Angelo in *Measure for Measure* and Hortensio and Petruchio in *The Taming of the Shrew*. Escalus, as Angelo's second-in-command, is in a way similar to Hortensio as Petruchio's friend and the one who introduces him to Katherine. Then, he acts as a student to Petruchio's "taming school" later on in the play. Otherwise, there are many differences between the characters of Escalus and Hortensio, as again, Escalus is a character who has a strong sense of justice, and Hortensio is one who has no strong sense for anything, it seems, as he gives up easily on marrying Bianca despite all of the effort he goes to court her, and decides to marry a rich widow instead.

Claudio and Sly are similar in the sense that both are trapped in some sort of way. Claudio is imprisoned for getting Juliet with child before marriage, and Sly is tricked into thinking that he himself is a lord, and that he has control over the situation, when in reality he is at the will of the Lord who is playing a prank on him to punish him. In both cases, Claudio and Sly are subject to the whims of authority in some way. To compare the characters in terms of sentiment, both of them have fairly even distributions of proportions of sentiments. One difference is that Sly has quite the high proportion of surprise as opposed to the rest of the characters in *The Taming of the Shrew*, and compared to Claudio.

Lastly, the one match that the computer analysis agreed with me on is Angelo and Petruchio. I think that in the computer analysis world, Angelo and Petruchio go surprisingly well with Isabella and Grumio, in the sense that, as stated before, Angelo and Petruchio both exercise the power that they wield over Isabella and Grumio respectively. Angelo's distribution of sentiments is, perhaps unsurprisingly, very uniform save for the trust category, but in contrast, Petruchio's is quite high in anticipation and joy, which also seems fitting. I believe that Angelo is meant to be someone who thinks he is very objective and emotionless, and perhaps Angelo would like to think that this distribution of sentiments of his is uniform. Petruchio is portrayed as a very outgoing character, to say the least, and I think that is also reflected in his distribution of sentiments.

Although this method of deleting rows and columns is sufficient for constructing a one-to-one mapping of characters from each play, other insights can be gleaned from the matrix as well. As mentioned previously, by looking for the minimum value across a given row and finding the column that corresponds with the lowest total variation distance, you will find the character from that column that is most similar to the character in the row. The same applies to the columns rather than the rows. Looking across the characters in *Measure for Measure*, Angelo, the Duke, Claudio, Isabella, Lucio, and the Provost are all most similar to Gremio in *The Taming of the Shrew.* The only explanation I can think of is that Gremio is the closest to average character in terms of sentiments compared to the characters in *Measure for Measure*, and therefore is very similar to many characters in the play.

Looking across columns, Hortensio is most similar to Pompey the clown, as I had suspected in my determination of matches. While the Duke is most similar to Gremio, Katherine is most similar to the Duke, which I find to be particularly of interest. The shape of their distributions are very similar, and I can imagine that the aggressive language that Katherine uses in *The Taming of the Shrew* is matched by the Duke's judgement at the end of Act V of *Measure for Measure.*

While Lucio is most similar to the Lord in *The Taming of the Shrew*, Sly is most similar to Lucio, which matches with my determination. Although Sly is most similar to Lucio, Mistress Overdone and the Duke are close behind. Sorting by the total variation distance in Sly's column, it seems to me that the more virtuous the character is, the higher the distance that character is from Sly, where the bottom half of the characters consist of, in increasing order, the characters the Provost, Isabella, Mariana, Escalus, First Gentleman, Peter (the friar), and lastly, Juliet. Pompey, Elbow, Claudio, and Angelo consist of the rest of the first half.

Juliet seems to be the least similar to any of the characters in *The Taming of the Shrew.* She is most similar to Grumio with a distance of 28.11%. Noticably, in her distribution of sentiment, the fear category has a significantly higher proportion of sentiment than in any other character. I suspect this has to do with her few lines with the Duke disguised as a friar concerning the execution of Claudio and her repentance for her crime. She also seemingly has the highest proportion of sadness sentiment and the lowest of trust, which I believe both reflect on how her character was presented in the play.

6 Conclusion

Overall, there were a lot of results from these matches that I did not expect. Perhaps the most surprising result was the Duke and Gremio being the two characters between the two plays that were the most similar.

The process of thinking about how characters in Shakespeare's plays relate to each other between plays has changed the way I think about and read Shakespeare. Noticing how similar any two of Shakespeare's plays are is a valuable tool to have when analyzing the texts from a literary perspective, and seeing these patterns can really help make sense of how Shakespeare writes about the world, in a general sense. In *Measure for Measure* and *The Taming of the Shrew*, Shakespeare draws from themes common to both plays, for example the role of authority and power to not only civil subject, but real people in individual relationships as well. The role of sex and desirability in human relationships are also a big theme in both of these plays. By thinking about how these themes are presented, and the parallels between how they are presented in different works makes interpreting and thinking about these themes more accessible.

We will conclude this paper with some limitations of the theoretical model, and then some further work that could be implemented to improve the practical model that was developed over the course of this paper.

6.1 Limitations

When designing a model, it is always important to consider the potential limitations to the model. As mentioned earlier in the paper, one of the biggest limitations is the introduction of human bias into the dataset, which is something that happens often with sentiment analysis models. Because human emotion is inherently subjective, the lexicons that are constructed will never be correct for anyone who uses them. However, the introduction of crowdsourcing methods will reduce this bias, which will get closer to an average of how humans understand the association of emotions to words. And although the crowdsourcing methods may be an improvement on single individuals assigning sentiments to the words, there is also response bias introduced since all of the sentiments in the NRC lexicon are assigned by people who use the Mechanical Turk service. From an experimental design standpoint, the construction of the NRC lexicon was not done from a random sample, which would improve the accuracy of the model if the goal is to approach an average association of words with their emotions. In the end, it is good to keep in mind that assigning emotions to words is not as simple as it may seem.

6.2 Further Work

There are many ways in practice to improve on the comparison model and algorithm that was implemented. One potential addition, for example, considering that oftentimes there are more characters in one play than in another, is creating a way to match multiple characters in the play with more characters to one character in the play with less characters, which doesn't leave any characters out of the comparison.

Another consideration to think about, also mentioned before, is to find a better way to integrate characters that have no observation in a given category. As it stands now, because there are no observations, the total variation distance will be inaccurate when subtracting vectors, so there is an additional data wrangling step in order to fix that bug in the program. However, for the purposes of this paper, the matches in *Measure for Measure* and *The Taming of the Shrew* were made with characters that all have at least one observation in each category.

Given more time, one idea/improvement that I think would be more significant in the results of the project is normalizing the probability values for each character based on how often the category appears in the whole NRC lexicon. As was noted previously, trust was a category that, for every character, shared a greater proportion of the sentiments, but based on the rightmost graph in Figure 11, trust would be expected to show up about 15% of the time. To add onto this, data normalization could be implemented using most common words, where more common words have less weight for their sentiments, for example. This is a half-baked idea, and the reasoning behind the normalization will have to be developed more rigorously in order for this to be a good idea to implement. However, this project did complete my goal as a first pass on comparing sentiments using computer analysis.

Again, although the results were surprising, there are many unexplored avenues that the field of Natural Language Processing can take, especially within a literary context. For me, this was a great opportunity to introduce myself to sentiment analysis, and apply it to a field that interests me. Throughout the course of this project, I discovered that there are a lot more parallels between Shakespeare's texts than I had thought about previously, and that these themes are prevalent in the language that the characters use, and the relationships that those characters have to each other. Even completing a brief analysis in this paper on only two of Shakespeare's plays, the cohesiveness of these relationships is prevalent.

References

[1] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, “High-resolution image synthesis with latent diffusion models,” pp. 10 684–10 695, June 2022.

[2] P. Christensen, “Fair is foul, and foul is fair: a tidytext sentiment analysis of shakespeare’s tragedies,” June 2018. [Online]. Available: https://peerchristensen.netlify.app/post/fair-is-foul-and-foul-is-fair-a-tidytext-entiment-analysis-of-shakespeare-s-tragedies

[3] J. Silge and D. Robinson, “Tidytext: Text mining and analysis using tidy data principles in R,” *J. Open Source Softw.*, vol. 1, no. 3, p. 37, Jul. 2016. [Online]. Available: http://dx.doi.org/10.21105/joss.00037

[4] S. M. Mohammad and P. D. Turney, “Crowdsourcing a word-emotion association lexicon,” pp. 436–465, 2013. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-8640.2012.00460.x

[5] M. Johnston and D. Robinson, *gutenbergr: Download and Process Public Domain Works from Project Gutenberg*, 2022, r package version 0.2.3. [Online]. Available: https://CRAN.R-project.org/package=gutenbergr

A R Implementation and Code

```r
lgl_stage_dir <- str_detect(play$text, "^\\[.*\\]$")

play <- mutate(play, row = row_number())
indices <- which(lgl_stage_dir)

for (index in indices) {
    play <- subset(play, !(row == index))
}

play <- filter(play, text != "")
```

Figure A.1: Using RegEx to identify all of the stage directions and filter out blank space

```r
index <- 0
for (i in 1:length(play$text)) {
    if (!str_detect(play$text[i], "[[:lower:]]")) {
        index <- index + 1
    }
    play[i, 4] <- index
}
names(play)[4] <- "paragraph"
```

Figure A.2: Numbering each paragraph.

```r
play <- t(pivot_wider(play[, -c(1, 3)],
                        names_from = paragraph,
                        values_from = text))

play_text <- data.frame(lapply(list(play),
                               stri_join_list,
                               sep = " "))
play_text <- data.frame(lapply(play_text,
                               trimws,
                               which = "left"))
names(play_text)[1] <- "paragraph"
```

Figure A.3: Grouping by paragraph, then joining the lists together to get a nice character vector.

```r
play_paragraphs <- data.frame(play_vector)
play_paragraphs <- play_paragraphs %>%
    separate(col = play_vector,
             sep = "\\. ",
             into = c("character", "text"),
             extra = "merge") %>%
    na.omit()

play_paragraphs <- filter(play_paragraphs, !str_detect(character, "^ACT"))
```

Figure A.4: Separating paragraphs into two columns: One for the character name, and one for the text itself. Then, filtering out the lines that start with ”ACT”

```r
play_tokens <- unnest_tokens(play_paragraphs, word, text)
play_sentiment <- inner_join(play_tokens, get_sentiments("nrc"),
                             relationship = "many-to-many")

n_words <- play_sentiment %>%
    group_by(word, character) %>%
    summarize(n = n()) %>%
    group_by(character) %>%
    summarize(n = n())

play_sentiment <- inner_join(filter(n_words, n > 15), play_sentiment)
```

Figure A.5: Unnesting the paragraphs, then joining together the result with the NRC lexicon.

```
shakespearecomparison <- function(play1, play2) {
  suppressWarnings({ suppressMessages({

    text1 <- text_transformation(play1)
    text2 <- text_transformation(play2)

    tvdmatrix <- data.frame()

    charactersentiments1 <- pivot_wider(text1[, c(1, 4)],
                                        names_from = character,
                                        values_from = perc)
    charactersentiments2 <- pivot_wider(text2[, c(1, 4)],
                                        names_from = character,
                                        values_from = perc)

    for (i in 1:length(unique(text1$character))) {
      for (j in 1:length(unique(text2$character))) {
        tvdmatrix[i, j] <- TVD(unlist(charactersentiments1[[i]]),
                               unlist(charactersentiments2[[j]]))
      }
    }

    colnames(tvdmatrix) <- colnames(charactersentiments2)
    rownames(tvdmatrix) <- colnames(charactersentiments1)

    return(tvdmatrix)

  }) })
}
```

Figure A.6: This fucntion takes the two plays as input, transforms the data, and constructs each entry of the total variation distance matrix with the nested for loops.

```
charmatrix <- shakespearecomparison("Measure for Measure", "The Taming of the Shrew")

if (nrow(charmatrix) > ncol(charmatrix)) {
  charmatrix <- t(charmatrix)
}

reduction <- charmatrix
matches <- c()
for (i in 1:(nrow(charmatrix) - 1)) {
  mat <- which(reduction == min(reduction), arr.ind = TRUE)
  matches[i] <- paste(rownames(reduction)[mat[, 1]], "|", colnames(reduction)[mat[, 2]])
  reduction <- reduction[-mat[, 1], -mat[, 2]]
}

matches # FINAL RESULT
```

Figure A.7: Storing the matches in a vector and deleting the rows and columns each time. Before this, the matrix is transposed so that the number of rows is less than the number of columns

B Full Functional R Program

Imports.

```r
library(tidyverse)
library(tidytext)
library(gutenbergr)
library(stringi)
```

The heavy lifting and data transformation is done in this function– takes the name of a play as a parameter.

```r
shakespeare <- gutenberg_works(author == "Shakespeare, William")

text_transformation <- function(play) {

  ID <- shakespeare[shakespeare$title == play,]$gutenberg_id
  play <- gutenberg_download(ID)

  lgl_stage_dir <- str_detect(play$text, "^\\[.*\\]$")

  play <- mutate(play, row = row_number())
  indices <- which(lgl_stage_dir)

  for (index in indices) {
    play <- subset(play, !(row == index))
  }

  play <- filter(play, text != "")

  index <- 0
  for (i in 1:length(play$text)) {
    if (!str_detect(play$text[i], "[[:lower:]]")) {
      index <- index + 1
    }
    play[i, 4] <- index
  }
  names(play)[4] <- "paragraph"

  play <- t(pivot_wider(play[, -c(1, 3)],
                        names_from = paragraph,
                        values_from = text))

  play_text <- data.frame(lapply(list(play),
                                 stri_join_list,
                                 sep = " "))
  play_text <- data.frame(lapply(play_text,
                                 trimws,
                                 which = "left"))
  names(play_text)[1] <- "paragraph"

  row <- ""
  play_vector <- play_text$paragraph
  while (!str_detect(row, "^ACT")) {
    play_vector <- play_vector[-1]
    row <- play_vector[1]
  }

  play_paragraphs <- data.frame(play_vector)
  play_paragraphs <- play_paragraphs %>%
    separate(col = play_vector,
             sep = "\\. ",
             into = c("character", "text"),
             extra = "merge") %>%
    na.omit()

  play_paragraphs <- filter(play_paragraphs, !str_detect(character, "^ACT"))

  play_tokens <- unnest_tokens(play_paragraphs, word, text)
  play_sentiment <- inner_join(play_tokens, get_sentiments("nrc"),
                               relationship = "many-to-many")

  n_words <- play_sentiment %>%
    group_by(word, character) %>%
    summarize(n = n()) %>%
    group_by(character) %>%
    summarize(n = n())

  play_sentiment <- inner_join(filter(n_words, n > 15), play_sentiment)

  play_sentiment <- play_sentiment %>%
    filter(!(sentiment %in% c("positive", "negative"))) %>%
    group_by(character) %>%
    count(sentiment) %>%
    mutate(perc = 100 * n / sum(n))

}
```

Total Variation Distance function.

```r
TVD <- function(mu, nu) {
  0.5 * sum(abs(mu - nu))
}
```

Automated in one big function! Takes two parameters– both are strings, and the titles of the plays to be compared.

```r
shakespearecomparison <- function(play1, play2) {
  suppressWarnings({ suppressMessages({

    text1 <- text_transformation(play1)
    text2 <- text_transformation(play2)

    tvdmatrix <- data.frame()

    charactersentiments1 <- pivot_wider(text1[, c(1, 4)],
                                        names_from = character,
                                        values_from = perc)
    charactersentiments2 <- pivot_wider(text2[, c(1, 4)],
                                        names_from = character,
                                        values_from = perc)

    for (i in 1:length(unique(text1$character))) {
      for (j in 1:length(unique(text2$character))) {
        tvdmatrix[i, j] <- TVD(unlist(charactersentiments1[[i]]),
                               unlist(charactersentiments2[[j]]))
      }
    }

    colnames(tvdmatrix) <- colnames(charactersentiments2)
    rownames(tvdmatrix) <- colnames(charactersentiments1)

    return(tvdmatrix)

  }) })
}
```

```r
charmatrix <- shakespearecomparison("Measure for Measure", "The Taming of the Shrew")

if (nrow(charmatrix) > ncol(charmatrix)) {
  charmatrix <- t(charmatrix)
}

reduction <- charmatrix
```

```r
matches <- c()
for (i in 1:(nrow(charmatrix) - 1)) {
  mat <- which(reduction == min(reduction), arr.ind = TRUE)
  matches[i] <- paste(rownames(reduction)[mat[, 1]],
                      "|",
                      colnames(reduction)[mat[, 2]])
  reduction <- reduction[-mat[, 1], -mat[, 2]]
}

matches # FINAL RESULT
```