Aidan Jenkins

Prof. Kyle Gorman

Methods II

# Final Project - Tragedy Detection

1. Code samples: GitHub
2. Data Used:
    1. Oedipus.txt
    2. Scikit-learn
    3. BERT
3. Write-up:

In this project, I trained a Machine Learning Algorithm to detect whether a written play (in the form of a .txt file) was a tragedy. I started with research on what made a play a tragedy. The protagonist may die, the entire play could be somewhat tragic (Death of a Salesman), or perhaps mostly the ending (Oedipus Rex).

I would need to be able to detect who the protagonist is. I did this by writing a function that would create a list of all words in the text in uppercase, giving me a list of every speaker per line in the play (think “OEDIPUS:”). Using a counter function, the most frequent name in that list would be selected and set as the protagonist character.

Once I had the main character pulled, I began preprocessing and normalizing the text. This included converting it to lowercase, removing special characters, removing stop words, stripping the text, stemming, lemmatizing?, tokenizing, and so forth.

Next, I wanted the code to determine if the protagonist dies in the end, or whether death is occurring in proximity to the protagonist, (ie, a love interest or family member dying). Taking the tail end of the text, and analyzing it for mentions of death within a close range of the protagonist's name.

I did sentiment analysis of the play as a whole vs. the ending detecting whether it is sad (try VADER per Kyle's recommendation) (or comedic using BERT - at a later date). These values would be compared and determine the overall sentiment of the play. For the sake of this project I stuck mostly with determining whether it was tragic.

This would be an interesting tool for readers and writers, and for websites such as Goodreads.com. It could provide recommendations to readers based on their preference for sad or happy reads, it could weed out books involving themes that are either desirable or undesirable. Separately, tragedies are defined by a set of characteristics, and tragic plays are already classified as such. Because of this, the results are able to be objectively compared for accuracy and test the limits of sentiment analysis. Personal purposes included learning more about sentiment analysis, and enjoying tidbits of literature in the process. This would also be good for writers to gauge sentiment / balance of emotions and character appearances in their works in progress.

The following is a description of data used:

The user in this case can input the url of the play they wish to read, and the output would print a variety of information about the play such as whether it was tragic, if it has a tragic ending, or if there was any balance of comedy. Sci-kit Learn was used to check the sentences in the text for any form of sadness in the play, both in its entirety, and more importantly in its ending. A large jump between lack of tragedy and a tragic ending would be good indicators as well as the entire play, however I wanted the output to be clear on why it was determined as such. BERT is to be used to find humor in the play to let users know if the play would be just a cry fest, or whether it was a good balance. It would be great to continue working on this with other options rather than just "Is this a tragedy?" and more of "Is this play a comedy or tragedy?". For now the dataset was just used as another determining variable and detail in the output.

During this project, I experimented with many ways to gather the same information and used trial and error to find what would work the best, or at all.

Some **troubleshooting** needed:

- ☐ The cast is printing the first upper case word, but not the full name of the speaker). For example, “CHORUS” should be “CHORUS OF THEBAN ELDERS”. Something to work on the regex for. (See image below under results.)
- ☐ The dataframe was successfully created with 30 lines and 30 speakers, however some of the lines don’t line up with a speaker and this needs to be remedied for the results to be more accurate. It wouldn’t really affect the results of the sentiment analysis, but it is /sloppy/ and I want to figure out what’s wrong - I imagine it’s just something with the way the text is split or a blip in the regex. (See image below under results.)
- ☐ Protagonist mention label is not accurate as indicated in line 1 of the df.
- ☐ Dataframe only shows 6 samples of death related speech, there should be 65. The for loop needs adjustment to iterate fully without stopping. Ran into trouble running through each token instead of each line. There were 65 total words in the play relating to death, so I added breaks for now so it would skip to the next line when a token was found. I think the breaks are happening at the first find of one of the words in the set however.

The **results** of the evaluation:

- ➔ Code creates a list of speakers, creates a set of these speakers, and determines the most common one. The most frequent speaker is printed as “CREON” for the small_sample of the data. For the entire play it is printed as “OEDIPUS”, which is the accurate answer for the play.

```
Cast:  {'HERDSMAN', 'PRIEST', 'SUPPLIANTS', 'JOCASTA', 'SECOND', 'ISMENE', 'CREON', 'TEIRESIAS', 'MESSENGER',
'OEDIPUS', 'CHORUS', 'BYSTANDERS', 'ANTIGONE'}
Protagonist:  OEDIPUS
```

➔ The dataframe created is below:

| | Speaker | Line | Protagonist | Mention of P |
|---|---|---|---|---|
| 0 | I | spurned | 0 | 0 |
| 1 | PRIEST | yea oedipus my sovereign lord and king thou se... | 0 | 0 |
| 2 | Therefore | o king here at thy hearth we sit i and these c... | 0 | 1 |
| 3 | OEDIPUS | ah my poor children known ah known too well th... | 1 | 0 |
| 4 | PRIEST | thy words are well timed even as thou speakest... | 0 | 0 |
| .. | ... | ... | ... | ... |
| 424 | CREON | then they soon will grant thy plea | 0 | 0 |
| 425 | OEDIPUS | lead me hence then i am willing | 1 | 0 |
| 426 | CREON | come but let thy children go | 0 | 0 |
| 427 | OEDIPUS | rob me not of these my children | 1 | 0 |
| 428 | CREON | crave not mastery in all for the mastery that ... | 0 | 0 |

The columns are: row number, speaker of the line, line itself, label indicating whether the speaker is the protagonist determined and saved in a variable higher in the code, and a label of whether the line contains the name of the protagonist.

➔ Printed below are the samples of the line having a mention from the death related term dictionary.

| | Speaker | Line | Protagonist | Protagonist Mention | Death Mention |
|---|---|---|---|---|---|
| 41 | CHORUS | the oath thou profferest sire i take and swear... | 0 | 0 | 1 |
| 347 | OEDIPUS | the knave methinks will still prevaricate | 1 | 0 | 1 |
| 367 | OEDIPUS | ah me ah me all brought to pass all true o lig... | 1 | 0 | 1 |
| 389 | OEDIPUS | strophe | 1 | 0 | 1 |
| 407 | CREON | this had i done already but i deemed it first ... | 0 | 0 | 1 |
| 418 | CREON | weep not everything must have its day | 0 | 1 | 1 |

Five out of the six lines that are labeled as containing death related terms are in the last fifth of the play. This would add a point to the categorization of ‘Oedipus the King’ as a tragedy.

➔ Not sure what this means yet, but here are the results of the VADER tool so far.

| | scores | compound | comp_score |
|---|---|---|---|
| 0 | {'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound... | 0.0000 | pos |
| 1 | {'neg': 0.072, 'neu': 0.913, 'pos': 0.014, 'co... | -0.7579 | neg |
| 2 | {'neg': 0.023, 'neu': 0.826, 'pos': 0.151, 'co... | 0.9851 | pos |
| 3 | {'neg': 0.147, 'neu': 0.742, 'pos': 0.111, 'co... | -0.5416 | neg |
| 4 | {'neg': 0.0, 'neu': 0.739, 'pos': 0.261, 'comp... | 0.6486 | pos |

➔ The results of VADER:

```
425  {'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound...    0.0000        pos
426  {'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound...    0.0000        pos
427  {'neg': 0.375, 'neu': 0.625, 'pos': 0.0, 'comp...   -0.5574        neg
428  {'neg': 0.157, 'neu': 0.739, 'pos': 0.104, 'co...   -0.7009        neg

Score for the entire play:
 comp_score
pos    272
neg    157
Name: count, dtype: int64

Score for the first 4/5ths of the play:
 comp_score
pos    223
neg    121
Name: count, dtype: int64

Score for the last fifth of the play:
 comp_score
pos    49
neg    36
```

Based on these results, Oedipus would be considered a tragedy. VADER found the last fifth of the play to be 42% negative, the first four fifths 35% negative, and finally the entire play to be 37% negative. The ending is nearly halfway negative, which is a significant percent.

Referencing back to the final four lines in the play below (425-428 in the Vader results), I would say the score is accurately reflecting the neg/pos read on those lines as well:

CREON Come, but let thy children go.

OEDIPUS Rob me not of these my children!

CREON Crave not mastery in all,
For the mastery that raised thee was **thy bane and wrought thy fall**.

CHORUS Look ye, countrymen and Thebans, this is Oedipus the great,
He who knew the Sphinx's riddle and was mightiest in our state.
Who of all our townsmen gazed not on his fame with **envious** eyes?
Now, in what a sea of **troubles** sunk and **overwhelmed** he **lies**!
Therefore wait to see life's **ending** ere thou count one mortal blest;
Wait till free from **pain and sorrow** he has gained his **final rest**.

**Ideas** for later:

- ★ Comedy detection using BERT. Could compare the % of comedy to tragedy for better recommendations.
- ★ Search for things misaligned with a tragedy: marriage, happy sentiment, humor, celebration

- ★ Software - input for users to paste text / links / files , and have it analyzed for comedy or tragedy, balance of characters, etc. Writer tool, recommendation tool, so on.
- ★ Different outputs based off of some questions they would answer along with inputting the title. For example, the person could indicate preferences and the output could include not whether the play is comedic or tragic, but whether they would like it. There could be a randomizer of different phrases that would be fitting responses based on their choices to make it feel more “human”.
- ★ Column to mark characters by their frequency or label as protagonist or second most common to the protagonist, or some other character label (detecting love interest, foil, antagonist, etc.).
- ★ Label in df to indicate if sadness was indicated specifically in lines mentioning or spoken by the protagonist.