Introduction to Computational Social Science assignment 4:

The emotional arc of stories: Shakespeare - Caesar

Mihaly Hanics, MS in Social Data Science

This report is about analyzing blocks of consecutive words in the digitalized copy of Shakespeare's tragedy of Julius Caesar and creating an "emotional arc". The analysis method is based on the paper: "The emotional arcs of stories are dominated by six basic shapes" by Reagan, A. J., et al.

Technical details: I used the Project Gutenberg library to access the book's digitalized format, but imported from the Python Natural Language Toolkit (NLTK)'s corpora. The crowdsourced sentiment score dictionary I choose is the labMT-simple one. For using the dictionary effectively, I mostly used the code given as example in the documentation (here). I also used the code here for inspiration, mostly for pre-processing the words. LLMs: I used GitHub Copilot for code completion. I did not prompt it, as I write code, it has the ability to continue it, for example, this snippet was written by Copilot:

```
for i in range(0, len(tokens), block_size):
    block = tokens[i:i+block_size]
    blocks.append(block)
```

But this is just "trivial" code completion to save time and not for coming up with ideas to tackle the problem. I did ask ChatGPT (3) for some inspiration of how to design my code optimally, I asked it what should be my functions, etc. But I did not use the code it recommended, only the inspiration. The *prepocess_text* function is strikingly named the exact same way as mine, true, but note how different the function is, much more similar to the text analysis GH example. Other codes given by GPT-3 are not effective. The prompt: I just copied details fast from the assignment, and tailored it, and asked a question:

"My task will be to analyze the emotional arc of a story by calculating the time series of word sen- timent in a digitized book (like from Project Gutenberg) where 'time' is a word window that slides through the entire text. Computa- tional Story Lab (CSL): Reagan, A. J., et al. "The emotional arcs of stories are dominated by six basic shapes." EPJ Data Science 5.1 (2016): 31. I have to pick a crowdsourced dictionary that gives each word in the text a numerical sentiment score (in a range of i.e. 1–9). The dictionary could come from the ANEW study, CSL's LabMT dataset, or any other source. I then have to compute the emotional arc of the book. Picking a word window (i.e. T = 100 words), the average sentiment in the window has to be calculated, and slide the window throughout the whole text. The result should be a time series of average word sentiment. Pre-processing the text as much as needed (tokenize, disregard words etc.) Have to try different T values.

In this project, what would you define as functions, macros, and so on?"

The answer:

In the context of your project, you can structure your code using functions to encapsulate different tasks and promote modularity. Here's a suggested breakdown:

1. **Pre-processing Functions:**
    - **`preprocess_text(text)`**: Tokenizes and cleans the input text, converting it to lowercase, removing punctuation, and handling any additional pre-processing steps.

```
import nltk
from nltk.tokenize import word_tokenize
```

```
def preprocess_text(text):
    text = text.lower()
    tokens = word_tokenize(text)
    # Additional pre-processing steps if needed
    return tokens
```

2. **Sentiment Analysis Functions:**
    - **`load_sentiment_dictionary(dictionary_path)`**: Loads the sentiment dictionary from a specified file.
    - **`calculate_average_sentiment(window, sentiment_dict)`**: Calculates the average sentiment in a given word window.
    - **`compute_emotional_arc(text, window_size, sentiment_dict)`**: Computes the emotional arc of the book by iterating through the text with a sliding window and calculating average sentiment.

```
def load_sentiment_dictionary(dictionary_path):
    sentiment_dict = {}
    with open(dictionary_path, 'r') as file:
        for line in file:
            word, score = line.split()
            sentiment_dict[word] = float(score)
    return sentiment_dict

def calculate_average_sentiment(window, sentiment_dict):
    valid_words = [word for word in window if word in sentiment_dict]
    if not valid_words:
        return None
    average_sentiment = sum(sentiment_dict[word] for word in valid_words) / len(valid_words)
    return average_sentiment


def compute_emotional_arc(text, window_size, sentiment_dict):
    tokenized_text = preprocess_text(text)
    time_series = []

    for i in range(0, len(tokenized_text) - window_size + 1):
        window = tokenized_text[i:i + window_size]
        average_sentiment = calculate_average_sentiment(window, sentiment_dict)
        if average_sentiment is not None:
            time_series.append(average_sentiment)

    return time_series
```

3. **Main Execution:**
    - Use a script or main function to orchestrate the execution of the different steps.

```
4. if __name__ == "__main__":
5.     # Load text, sentiment dictionary, and call compute_emotional_arc with different
       window sizes
6.     book_text = load_book_text()  # You need to implement a function to load your book
       text
7.     sentiment_dictionary = load_sentiment_dictionary('labmt_dict.txt')
8.
```

```
9.      emotional_arc_t100 = compute_emotional_arc(book_text, window_size=100,
    sentiment_dictionary)
10.     emotional_arc_t50 = compute_emotional_arc(book_text, window_size=50,
    sentiment_dictionary)
```

This structure helps in organizing your code, making it more readable, modular, and easier to maintain. Each function focuses on a specific task, promoting code reuse and clarity.

As you can see, I almost completely ignored what it recommended. I should have probably told it two things:

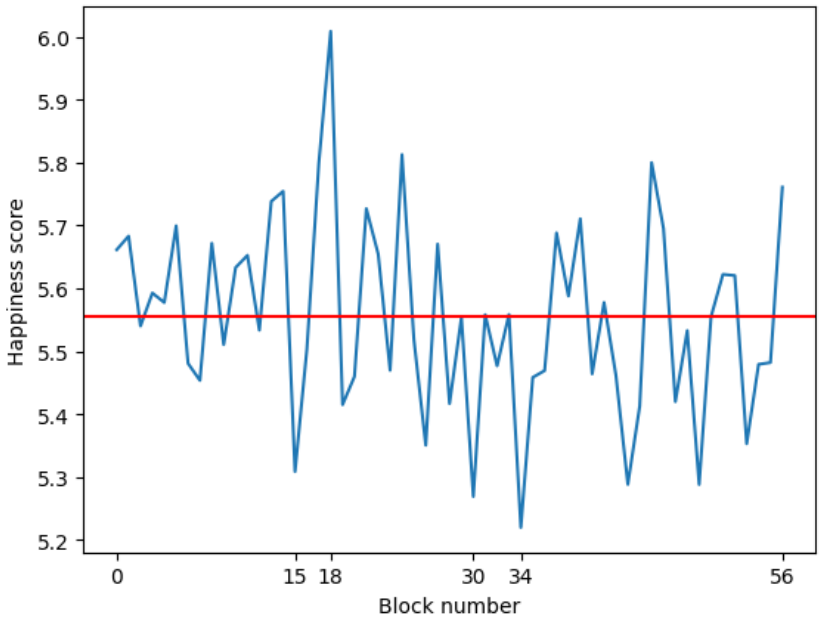
-don't include code (more on point, and probably more accurate)
-I am proficient in Python (the main part is for amateurs)

However, Copilot did help a lot in writing the code faster, and also probably I'd make more mistakes without these code completions.

Note: I couldn't choose a book that I really like. Those books include mostly scientific and documentary books, and I couldn't find one that I read digitalized. So mind me on my lack of knowledge of the story. I did look into the story however upon writing this.

Preprocessing: I filtered out the stopwords, the non-alphabetic-characters, and words not in the labMT wordlist. This resulted in less than 5800 processable words, so I tried processing the words with Porter-Stemmer and then filtering them if they're not in the labMT wordlist, but it only gave me about a 5% increase in words, and I found that it's not worth it, should not have much impact on the results directly. I found that the labMTsimple library contains not very many words. I also tried non-filtering, see results below.

I used a blocksize of 100 initially. Let's see first the plot of the happiness levels: (red line representing the average score)



It seems like right before the peak of happiness, there was a great sad part, with score 5.31. Looking more into it, the deep point is Act II, Scene I, Rome, Brutus' orchard. That particular block contained Brutus' talking about the suffering of Romans (under Caesar). The text (more than 100 words, but only 100 are not filtered):

BRUTUS: ... So let high-sighted tyranny range on, Till each man drop by lottery. But if these,

As I am sure they do, bear fire enough To kindle cowards, and to steel with valour

The melting spirits of women; then, countrymen, What need we any spur but our own cause

To prick us to redress? what other bond Than secret Romans, that have spoke the word,

And will not palter? and what other oath Than honesty to honesty engag'd,

That this shall be, or we will fall for it? Swear priests and cowards, and men cautelous,

Old feeble carrions, and such suffering souls That welcome wrongs; unto bad causes swear

Such creatures as men doubt; but do not stain The even virtue of our enterprise,

Nor th' insuppressive mettle of our spirits, To think that or our cause or our performance

Did need an oath; when every drop of blood That every Roman bears, and nobly bears,

Is guilty of a several bastardy, If he do break the smallest particle

*Of any promise that hath pass'd from him.**

CASSIUS. But what of Cicero? Shall we sound him? I think he will stand very strong with us.

CASCA. Let us not leave him out.

CINNA. No, by no means.

METELLUS. O, let us have him, for his silver hairs Will purchase us a good opinion,

And buy men's voices to commend our deeds. It shall be said, his judgement rul'd our hands;

Our youths and wildness shall no whit appear, But all be buried in his gravity.

BRUTUS.

O, name him not; let us not break with him; For he will never follow anything That other men begin.

CASSIUS. Then leave him out.

CASCA. Indeed, he is not fit.

DECIUS. Shall no man else be touch'd but only Caesar?

CASSIUS. Decius, well urg'd. I think it is not meet, Mark Antony, so well belov'd of Caesar,

Should outlive Caesar: we shall find of him A shrewd contriver; and you know, his means,

If he improve them, may well stretch so far As to annoy us all; which to prevent,

Let Antony and Caesar fall together.

BRUTUS. Our course will seem too bloody, Caius Cassius, To cut the head off, and then hack the limbs,

Like wrath in death, and envy afterwards; For Antony is but a limb of Caesar. Let...

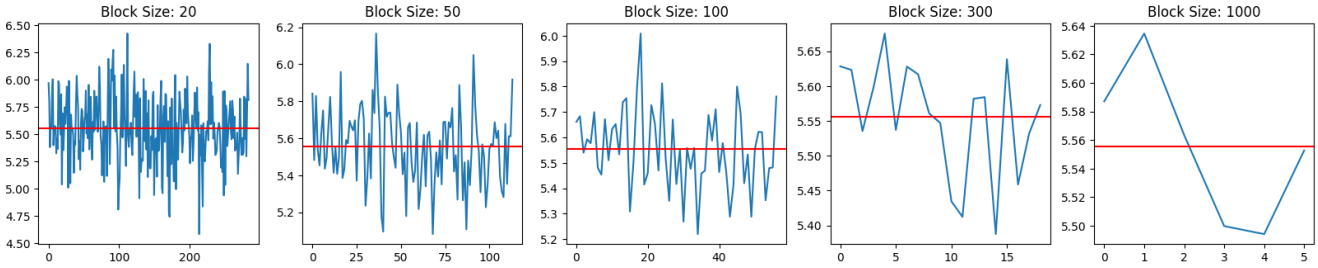
As you can see, some “good” words got into the text, which may have rose the happiness level by a tenth. But does a happiness score of 5.31 mean “sad”? I tried to find words with such happiness score, “median” is 5.32, words like “random” are around 5.1-5.3. I took the mean of the scores in the labMT dataset which came out to be around 5.3 aswell. Well, it is safe to say that it’s disappointing that one of the most dramatic parts of a tragedy is only “average”, not sad. So I took only the text of Brutus’ lengthy monologue and fed it into the emotion evaluator, which gave the value of 5.18. I took the even more “sad”, bloody monologue of Brutus’ saying how only Caesar shall be murdered, but not Mark Antony, as that’d be “butchering”, which came with score 5.14. These results are better, but still not in the “sad” category, so I thought I’d filter the stopwords and such from them. Surprisingly, the results were much higher, 5.31 and 5.32! That is unexpected, that is not what is supposed to happen! I believe a lot of the phrases actually have a positive score, which bring up the score, and the “average” stopwords helped keep the scores be more “average” (lower). Like how “bloody spirit” is supposed to be very negative, but spirit has a positive value, so it comes off quite average. I checked then the general results without preprocessing. All blocks seem to be about 0.15-0.25 lower in happiness score. So it turns out that our preprocessing was mostly only worth a “lift”.

Interestingly, the happiest part of the novel is Brutus arguing with her wife. Her wife knows something bothers him, and wants to know what’s the matter, but he feds her off. The words wife, beauty, kneel, bond, marriage, gentle, face, heart, etc. all increase the happiness level in this block however. This is a false positive. Later on, to prove her trustworthiness, Portia stabs herself in front of her husband.

The lowest happiness score is not much after the murder of Caesar in Act III, Scene II, when Mark Antony describes that he should do wrong with Brutus and Cassius, but as they’re honorable men, he shall not do it. The citizens of Rome want to hear the will of Caesar, but Antony is afraid to read it as he fears the “daggers that stabbed Caesar”. I’d say this should not be sadder than the death itself, or the planning. But it indeed has strong words.

If I could assume, the results are poor because there are too many average words, that bring the scores closer to the average. I mean the difference between the highest and lowest point is less than 0.7… To tackle that, maybe the “average” words need to be filtered, or the “extreme” words need higher weight.

I have tried other T values for the analysis:



Here as expected, smaller T values can find the happier and sadder blocks. Now lengthy monologues of about 50 words indeed get the “sadder than average” mark. With block size 300, already the span is less than 0.25m which is already likely a bad measure on which

parts aresad and happy. Indeed, the plans of killing Caesar get mixed with happy parts, and get near the average score of the story. With block size 1000, we'd indicate a "happy beginning, sad ending" part, so a rise/fall story. But in fact, the beginning acts describe unhappy situations already, and the plans to murder shall not be considered happy either. Thus I'd categorize this as a "fall" story, as it only describes the fall and death of Caesar, not the rise. I could not compare with what is shown on the Hedonometer website, as the slides, visualizations, explorables are not available anymore (on Edge, Mozilla at least).