Feature 3: Emotions to capture the book start and book end

1) Milestone 1:

Capturing Emotions(Anger, Anticipation, Disgust, Fear, Joy, Sadness, Surprise, Trust)

Purpose of the feature: We believe that whilst the positive/negative sentiment provides an overview of the plot, emotions provide a more nuanced way of comparing two books. The degree of emotion felt by a reader whilst reading a book is often multi-faceted and we are trying to capture some of these facets as a way of identifying similar books based on the emotions that they draw out.

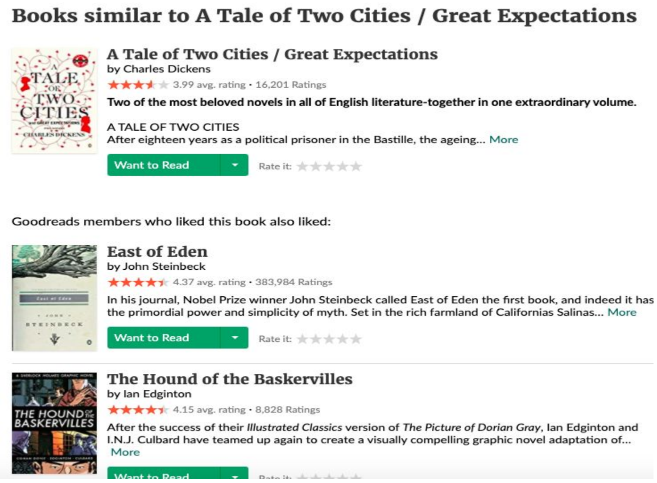
Why this method? -

- According to the latest research, language models and transformer techniques are the state of the art.
- We also researched on extracting Sarcasm and Irony from texts, But due to the nature and size of the corpus and availability of other labeled datasets, we have made some simplifying assumptions.
- We have used the NRC emotion lexicon(available for both English and German).

How its extracted? - The book list, book paths and the feature csv file path could be passed as parameters to a python program(shell script/Powershell script that calls python program). The program then reads each file and tokenize by words and applies stopword removal and lemmatizer. Depending on the book start and bookend percentage that should be considered for analysis, we filter out the tokens that are in the middle of the book. Remaining tokens are looked upon NRC emoticon which returns back a dictionary containing emoticon per token. The emotions are rolled up and normalized by the length of the token array of each book and also normalized using the sum of all emotions so that their combined sum could add up to one. Extracted features would be written in the feature csv file.

Example: Results contained some previously unseen combo of books such as A tale of two cities and The Hound of Baskervilles which when looked upon Goodreads were read by similar readers. Since it is based on a lexicon file it could be implemented in Java too depending on circumstance.

Main Libraries that would be used: Python - collections/ Java - Collections and NRC emoticon



Books similar to A Tale of Two Cities / Great Expectations

A Tale of Two Cities / Great Expectations
by Charles Dickens
3.99 avg. rating · 16,201 Ratings
Two of the most beloved novels in all of English literature-together in one extraordinary volume.

A TALE OF TWO CITIES
After eighteen years as a political prisoner in the Bastille, the ageing... More

Goodreads members who liked this book also liked:

East of Eden
by John Steinbeck
4.37 avg. rating · 383,984 Ratings
In his journal, Nobel Prize winner John Steinbeck called East of Eden the first book, and indeed it has the primordial power and simplicity of myth. Set in the rich farmland of Californias Salinas... More

The Hound of the Baskervilles
by Ian Edginton
4.15 avg. rating · 8,828 Ratings
After the success of their *Illustrated Classics* version of *The Picture of Dorian Gray*, Ian Edginton and I.N.J. Culbard have teamed up again to create a visually compelling graphic novel adaptation of... More

2) Milestone 2:

Implementation:

Preprocessing NRC emotion files and book list: We have separate English and German NRC files in a bit different formats, first we bought both of them into the same format and created a common emotion file for ease of use.

We also preprocess the book list excel file and pick up the pgid along with language and put it into a dictionary.

Books to Tokens: Each book's text would be read and its language would be retrieved from the booklist file using its pgid. A dictionary would be created consisting of pgid as key and language, text as value. We direct each text towards its respective language NLP pipeline (SPACY based) and to remove stopwords and lemmatize it. Before lemmatizing, we also cut off the middle part of the novel which is of no use for our calculation. This cutoff criterion is percentage based and dynamic.

Tokens to Emotion feature vector: Each token of the book is looked up in the dictionary and its emotion values are retrieved. Later each emotion vector is summed up and divided by the number of feature vectors in the book start/end respectively. We will also scale this normalized feature vector to some higher value by multiplying it with an empirical number so that we have a good spread of similarity.

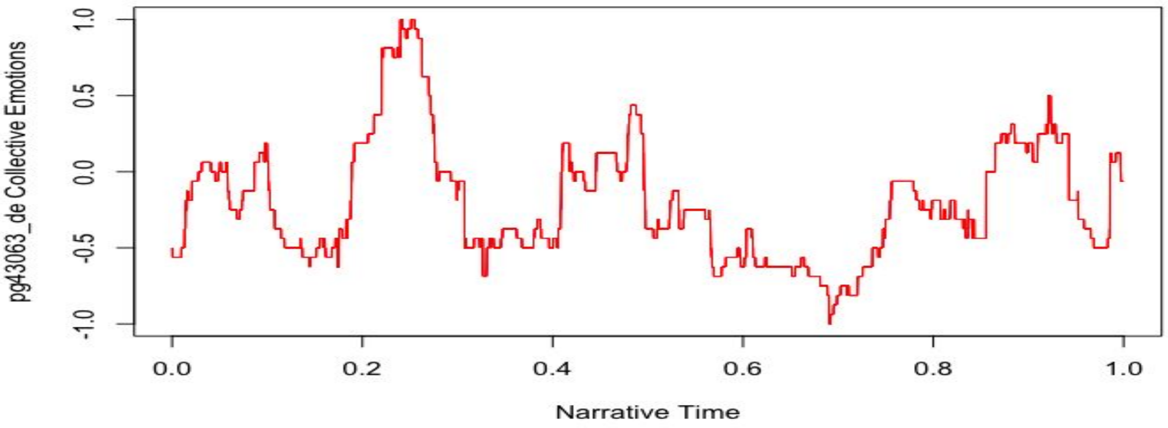
Similarity: After we have a normalized/scaled feature vector for book start and bookend for each book, we apply L2 similarity to get the similarity between book start and bookend of the same book. Cosine similarity is not used as it either provides a really high similarity or really low similarity. This similarity is saved in a file and later joined with the main feature file.

Explanations for English books: Similarity ranges have been color-coded Green - high, Yellow - Medium, Red - Low. A known set of books are in **bold**

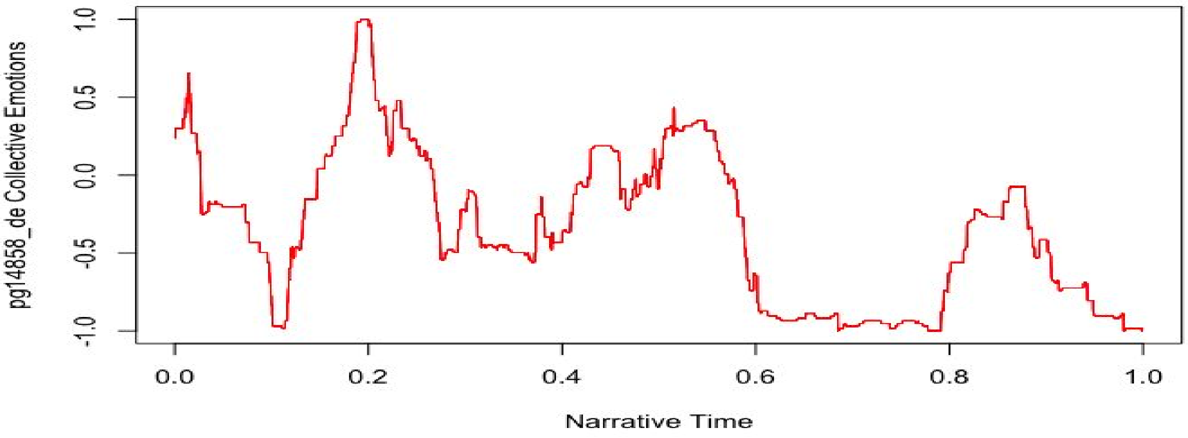
Book Id	**Book name - Genre**	score
pg43063	A Case in Camera - Detective Mystery	0.8663
pg5705	The queen of Sheba - Literature	0.8484
pg14253	Flames - Literature	0.8471
pg14858	The man thou gavest - American Literature	0.5505
pg5848	The Flyers - American Literature	0.5503
pg44872	The man who fell through the earth - Detective Mystery	0.548
pg29229	Andre - Drama Tragedy	0.1984
pg25565	The Newyork idea - Drama Playwright	0.18
pg26687	Black spirits and white - Ghost stories	0.1736
pg11	**Alice in Wonderland - Fantasy**	**0.39**
pg12	**Looking through the glass - Fantasy**	**0.62**
pg158	**Emma - Romantic Comedy**	**0.5741**
pg161	**Sense and Sensibility - Romance**	**0.602**

pg108	**Sherlock Holmes - Detective Mystery**	**0.7702**
pg98	**Tale of two cities - Historical Fiction/War**	**0.4254**
pg331	**The Mucker - Adventure**	**0.7156**

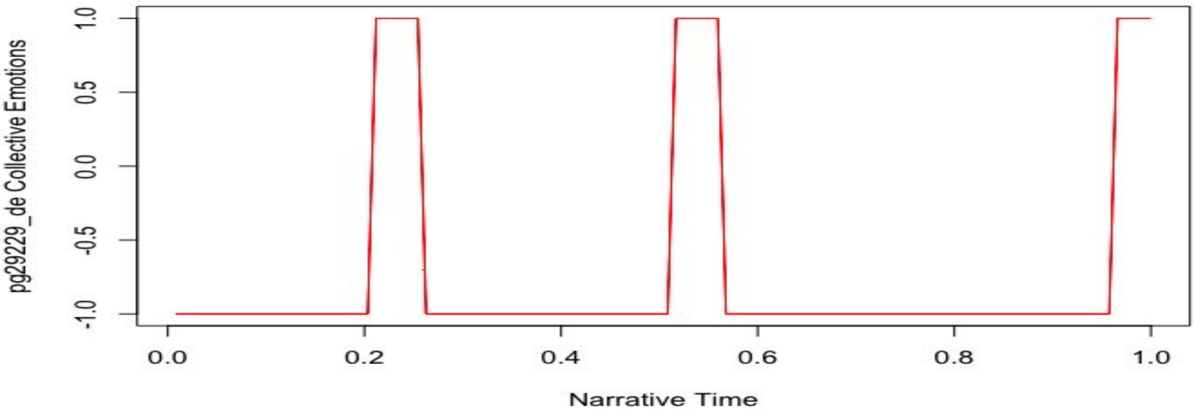
High similarity: pg43063 We can see the emotion plots between 0-0.2 and 0.8-1.0 (first and last 20% of books) that they look the same(start with peak then a trough and then peak again). Hence has high similarity.



Medium similarity: pg14858 We can see the emotion plots between 0-0.2 and 0.8-1.0 (first and last 20% of books) that they are not looking the same at all but when they are averaged out we get two vectors that have some similarity, hence they are in medium category.

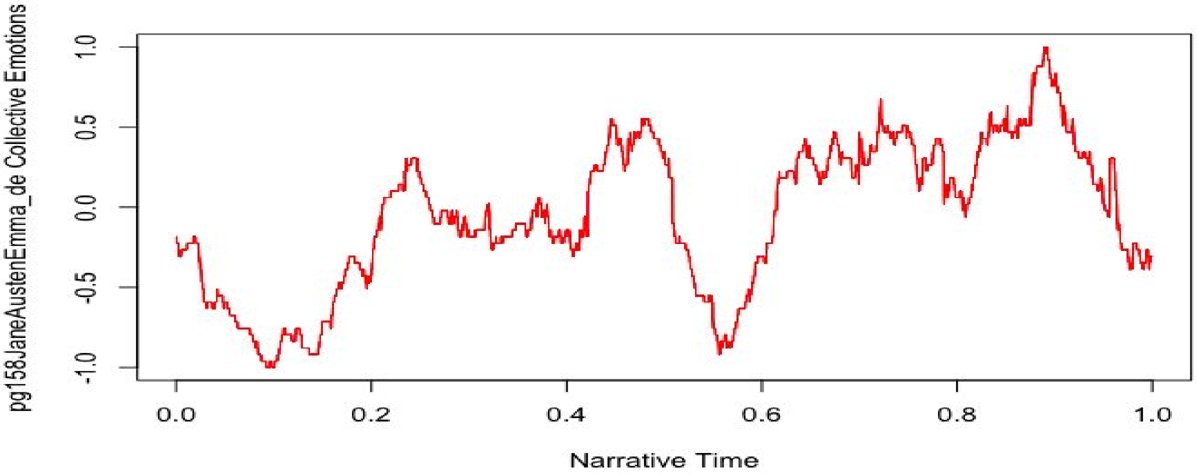


Low similarity: pg29229 We can see the emotion plots between 0-0.2 and 0.8-1.0 (first and last 20% of books) that they are completely dissimilar, hence low similarity score.

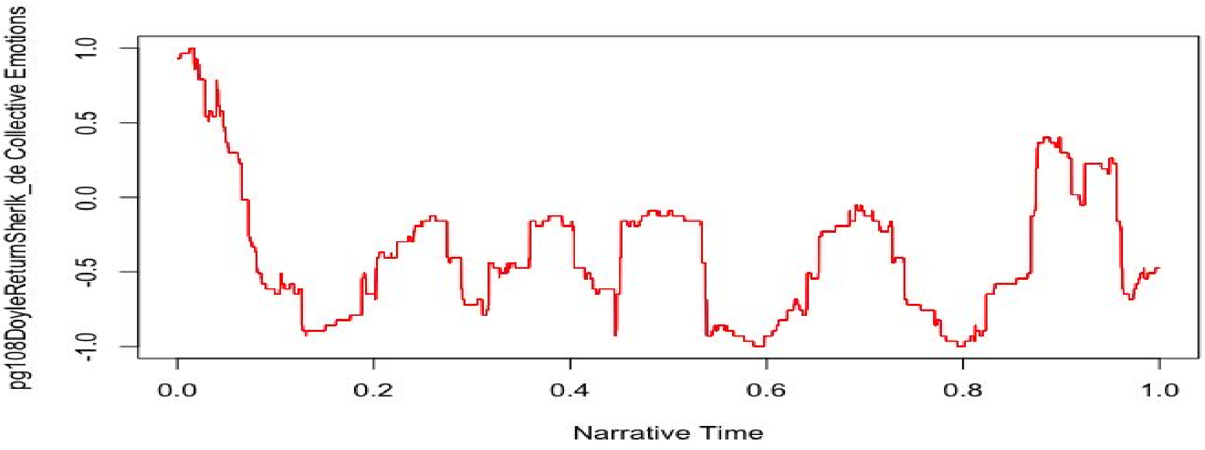


Known Books plots:

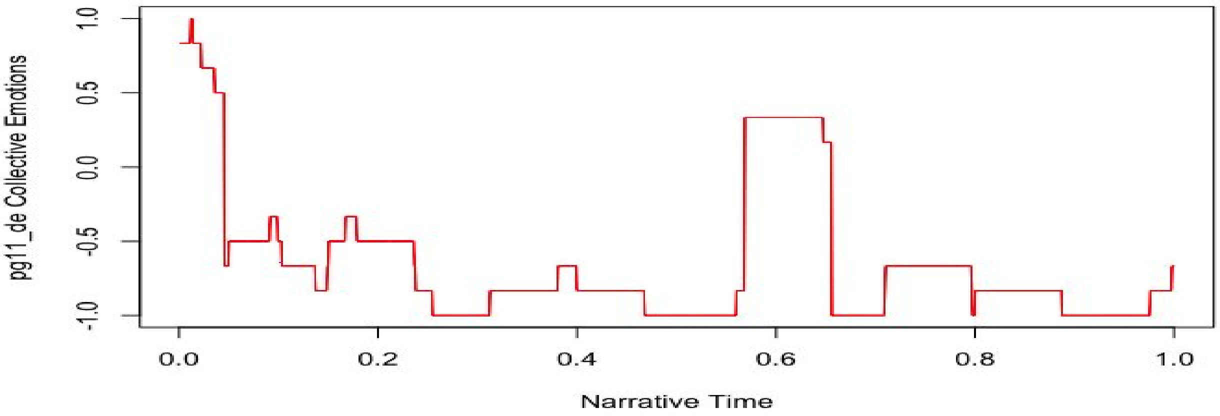
Emma: pg158 similarity = 0.57



Sherlock Holmes: pg108 similarity score: 0.77



Alice in Wonderland: pg11 similarity = 0.39



Observed Outliers: We see that most of the low similarity books tend to be small books, collections of stories, Plays, Sarcasm or History.

Small books might have to focus on only limited topics, hence their start and end are most likely not the same.

Collection Stories have multiple stories in them, hence similarity was medium-low in most cases.

Sarcasm/Irony is not captured, hence some of the witty humored books are not on high similarity.

Planned Refinements:

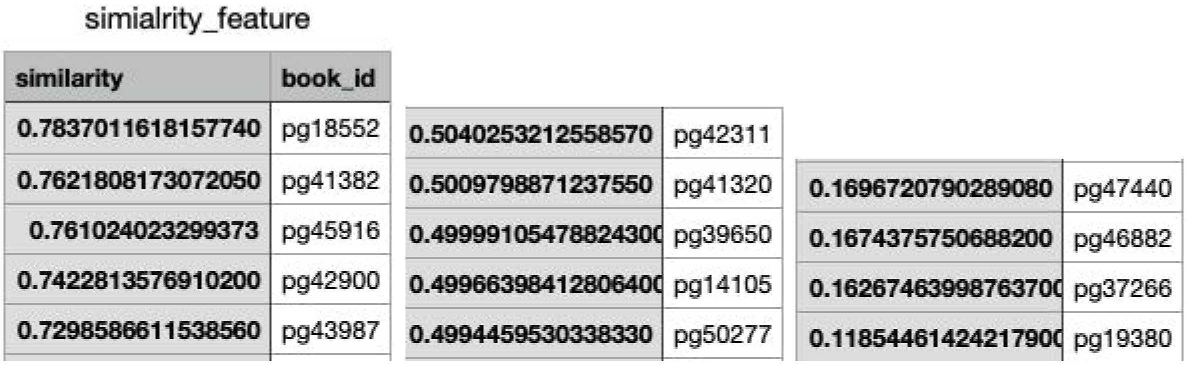
We can use the character map generated as an extended step of stop word list . This helps make the feature vector more dense.

Explanations for German books:

Below are some examples of feature vectors and the final similarity score for some German books

```
Book: pg3062 || Book start vector: [0.9761904761904762, 1.4285714285714288,
3.2380952380952386, 1.5714285714285716, 0.7142857142857144, 0.8571428571428572,
0.14285714285714288, 1.0714285714285716], Book end vector: [0.35714285714285715, 0.0,
4.285714285714286 5, 1.4285714285714286, 0.35714285714285715, 0.35714285714285715, 0.0,
3.2142857142857144]


Book: pg85 || Book start vector: [1.0083704819412622, 1.6575544128856288, 0.7710007610189097,
1.6836101528705885, 0.85246106820517, 1.0944069500103981, 0.9818836092656417,
1.950712563802401], Book end vector: [0.9629437488565452, 1.7311234857862816,
0.5354862905813994, 1.6789594525204696, 0.7605500011626496, 1.100649689983929,
0.44306014788351755, 2.7872271832252085]
```

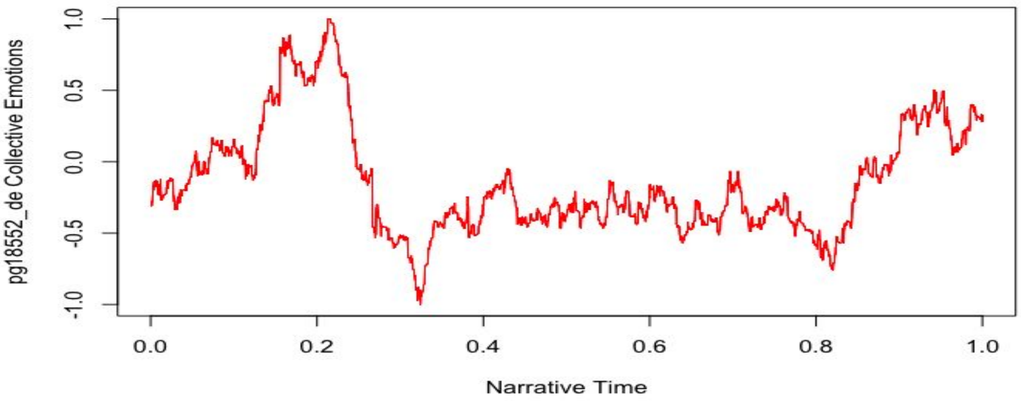


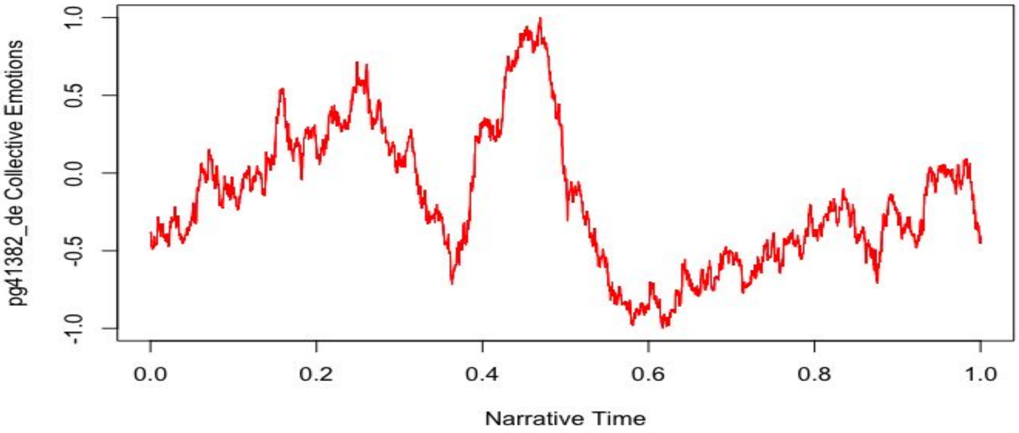
simialrity_feature

similarity	book_id
0.7837011618157740	pg18552
0.7621808173072050	pg41382
0.761024023299373	pg45916
0.7422813576910200	pg42900
0.7298586611538560	pg43987
0.5040253212558570	pg42311
0.5009798871237550	pg41320
0.4999910547882430	pg39650
0.4996639841280640	pg14105
0.4994459530338330	pg50277
0.1696720790289080	pg47440
0.1674375750688200	pg46882
0.1626746399876370	pg37266
0.1185446142421790	pg19380

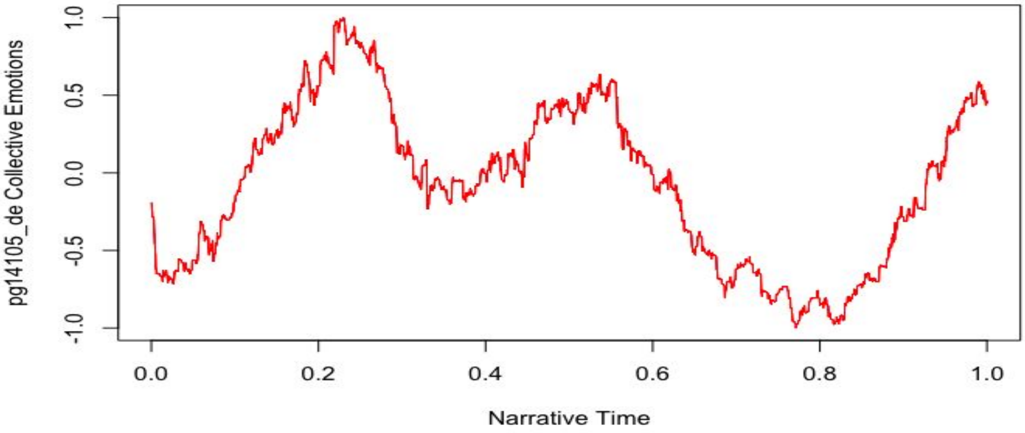
Below are some examples of the books with high, medium and low similarity

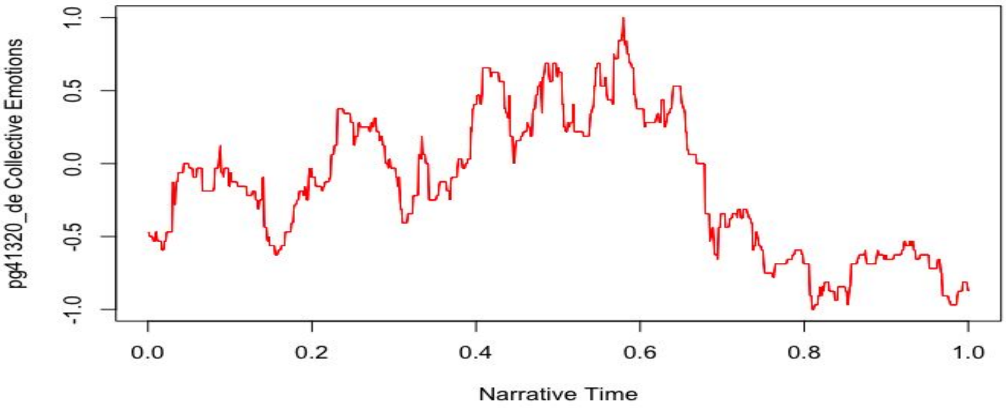
High similarity scorebooks: We can see by comparing their first 20% and last 20% of the plot that they almost look similar and going in the same direction. The similarity score was around 0.77 (pg18552)



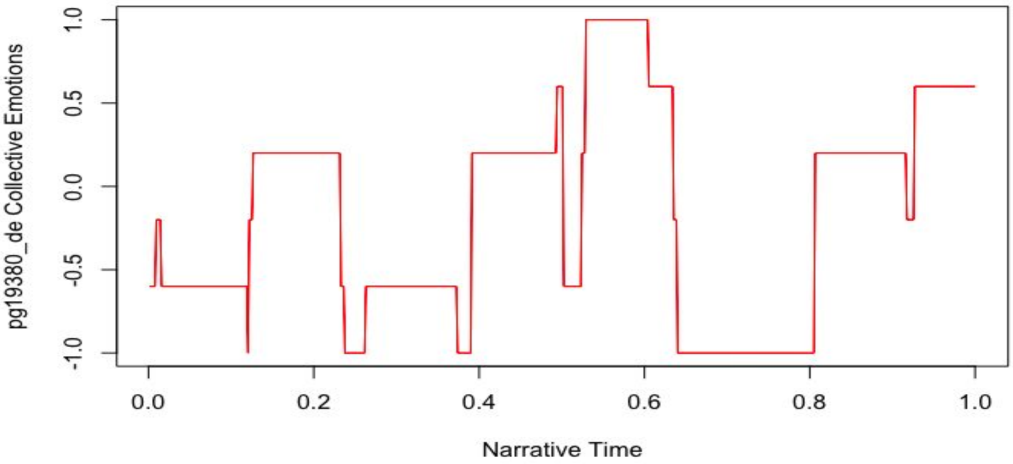


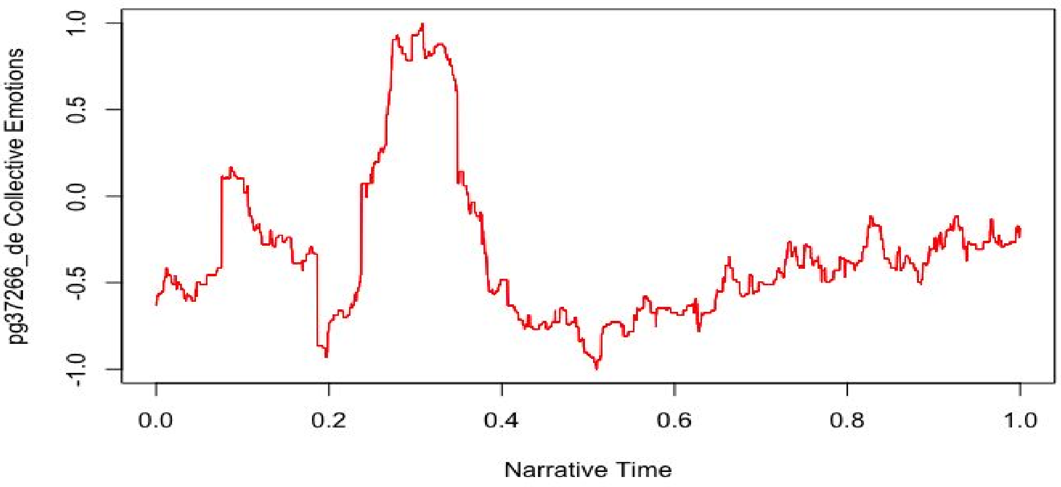
Medium similarity books: We can see that even though directions are almost the same but the magnitude is bit varying for the first 20 and the last 20% of the book. The similarity score was around 0.5 (pg14105)

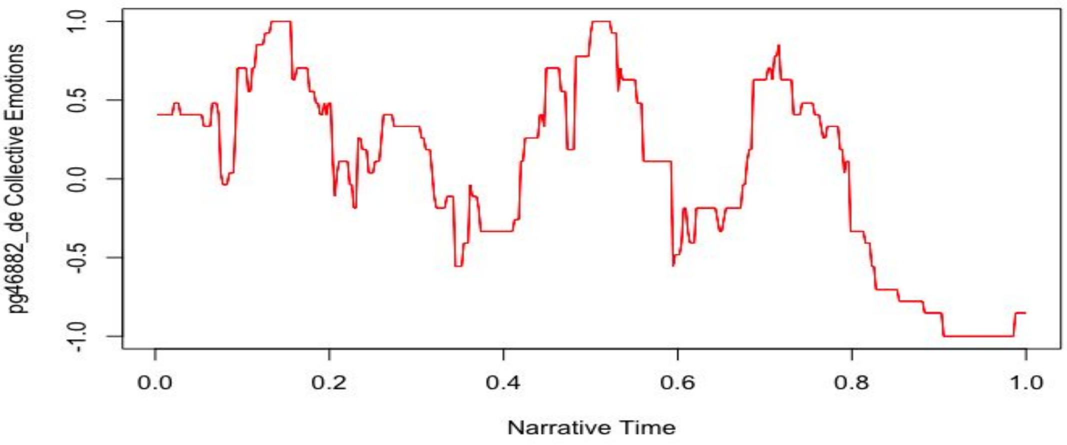




Low similarity books: We can clearly see that the collective emotions at the beginning and the end of the book are quite different and are pointing towards the opposite direction. The similarity score was around 0.18 (pg19380)







References:

1) An Evaluation of Lexicon-based Sentiment Analysis Techniques for the Plays of Gotthold Ephraim Lessing (*Thomas Schmidt et al*)

2) A Survey on Sentiment and Emotion Analysis for Computational Literary Studies (*Evgeny Kim et al*)

3) Story Comprehension for Predicting What Happens Next (*Snigdha Chaturvedi et al*)

4) Topic Modeling Literary Quality (*Kim Jautze et al*)

5) Subsentential Sentiment on a Shoestring: A Cross-lingual Analysis of Compositional Classification (*Michael Haas et al*)

6) Sentiment Analysis as a Tool for Understanding Fiction (*Matthias Landt*)

7) A review of sentiment computation methods with R packages (*Maurizio Naldi et al*)

8) **Fine-grained German Sentiment Analysis on Social Media** (*Saeedeh Momtazi*)

9) A novel automatic satire and irony detection using ensemble feature selection and data mining (*Kumar Ravi et al*)

10) **Irony and Sarcasm: Corpus Generation and Analysis Using Crowdsourcing** *(Elena Filatova)*