Emotional Analysis Internship Report (Final)

SRIP-2020

IIIT Allahabad

Parth Parikh

Supervisor: Punit Singh

Under Prof. U. S. Tiwari

Professor

INDIAN INSTITUTE OF INFORMATION TECHNOLOGY, ALLAHABAD

(A UNIVERSITY ESTABLISHED UNDER SEC.3 OF UGC ACT, 1956 VIDE NOTIFICATION NO. F.9-4/99-U.3 DATED 04.08.2000 OF THE GOVT. OF INDIA)

A CENTRE OF EXCELLENCE IN INFORMATION TECHNOLOGY ESTABLISHED BY GOVT. OF INDIA

Topic

We are building a recommender system that matches the mood of a person and suggests a book from Project Gutenberg. The book suggested should be such that after reading the book, it improves the mood of the reader.

This project can be divided into 3 parts:

1. Finding the emotions of a set of books from Project Gutenberg. The emotions are mapped to a 3-dimensional space called VAD space. Here, V stands for *valence* (i.e. positiveness/negativeness of the text), A stands for *arousal* (i.e. calmness/excitement present in the text), D stands for *dominance* (i.e. submissiveness/dominance present in the text).
2. Finding the mood of the readers (before reading) by asking them some psychological questions using a questionnaire.
3. Recommending a chapter/text from books such that the recommended book's sentiment improves the reader's mood.

In Week 1, our primary focus was to get the relevant datasets, read literature, and sketch out a work plan. In the rest of this report, we will be delineating these goals.

Dataset

Our project requires different kinds of datasets for a variety of tasks:

- Lexicons: We are using Lexicons to find out the VAD mapping of the word. The lexicons we are using are -
    - The NRC Valence, Arousal, and Dominance (VAD) lexicon **[1]** assigns valence, arousal, and dominance scores to 20,000 words.
    - VAD Lexicons by Warriner, et. al **[2]**. The original Warriner lexicons are in 1-9 VAD space. JEmAS (Jena Emotion Analysis System) **[3]**, a model which we are studying transforms these lexicons to `[-4, 4]` space.

  The below-mentioned Lexicons were developed primarily using Amazon Mechanical Turk (crowdsourcing website). Usually, human labeling for emotion analysis tasks is a two-step process **[4]**:

1. To ensure that the annotators were judging the correct sense of the word, they first answered a multiple-choice synonym question that primed the correct sense of the word.
2. For each word, the annotator was then asked to rate how associated that word is in VAD space.

- Corpus: We could find three corpora that map sentences/stories to the VAD space. These are:
    - 4000 stories with sentiment analysis dataset **[5]**. This dataset presents 4,000 short stories that have been classified in terms of their emotional content and semantic structure. The emotional content was calculated using the valence, arousal, and dominance norms in Warriner et al. (2014).
    - EmoBank **[6]**, a large-scale text corpus manually annotated with emotion according to the psychological VAD scheme. This dataset contains sentences that are of roughly the same length as a tweet.
    - Affective Norms for English Text (ANET) **[7]**. The Affective Norms for English Text (ANET) provides normative ratings of emotion (pleasure, arousal, dominance) for a large set of brief texts in the English language for use in experimental investigations of emotion and attention.

- Miscellaneous:
    - Project Gutenberg Dataset: We have downloaded 1728 fictional books from Project Gutenberg. We chose the books used by Reagan et al **[8]** for their research. Metadata of the books were downloaded from a project by Hugo van Kemenade **[9]**. The uncompressed size of the Gutenberg corpus curated is 733 MB.
    - Dataset for mean and standard deviations (SD) of different emotions on VAD dimensions is from Verma, Gyanendra & Tiwary, Uma Shanker (2015) **[10].**

Apart from the above-mentioned datasets, to classify VAD dimensions to emotions (like anger, joy, etc.) we are using Dataset for mean and standard deviations (SD) of different emotions on VAD dimensions **[10]** to synthetically generate our data.

Literature Survey

In week 1, our literature review will primarily focus on finding the emotions present in a particular text. As mentioned in the Dataset section, human labeling can primarily be used to find out the emotions of English words. However, it is not easy to scale this to sentences, that is, asking humans to read books and scale their emotions would be a very long and tedious process.

To overcome this issue, we searched for unsupervised learning techniques which, given some sentence, can predict the Valence, Arousal, and Dominance of the emotions. As mentioned earlier we are going to map values from VAD space for the recommendation task. A rule-based approach mentioned by Buechel, et al **[3]** is as follows:

$$EV_{d_i} := \frac{\sum_{k=1 \,\wedge\, \exists lex_q \in \pi_1(VAD):\, SEQ(lex_q, t_{i,k})}^{n} \lambda_{t_{i,k}} \times \langle v_q, a_q, d_q \rangle}{\sum_{k=1 \,\wedge\, \exists lex_q \in \pi_1(VAD):\, SEQ(lex_q, t_{i,k})}^{n} \lambda_{t_{i,k}}}$$

Fig 1: In the above equation, Emotion Value (EV) of a document d_i can be measured by - for each term t_k in document i, if the term's VAD are present in a particular lexicon like Warriner **[2]** or NRC **[1],** then the sum of all such terms' TF-IDF values multiplied by the VAD space tuple becomes the numerator space. The denominator is then the sum of the TF-IDF terms. As an alternative lambda can be Term Frequency.

We coded the above paper in Python, and the RMSE on the Emobank **[6]** corpus was:

Valence	Arousal	Dominance
0.545715751868221	0.49471129595381275	0.39740000815828447

Apart from the above rule-based method, we trained a doc2vec model on Emobank's text using the Gensim library. We first trained Distributed Memory (DM) and Distributed Bag of Words (DBOW) models **[11].** We then performed Mean Normalization and Principal Component Analysis (keeping 95% of the variance) on the vectors generated. After this, we performed supervised learning on the trained vectors of each sentence and its corresponding label in the Emobank corpus. The learning algorithms we tried were Multilayer Perceptron Model, Linear Regression, Support Vector Regression, and Nu Support Vector Regression. We used the Scikit-learn library to train our models.

We found that both SVR and Nu SVR gave the best performance, followed by MLP and finally Linear Regression.

We also noticed that with mean normalization and PCA, the convergence was faster and accuracy was better. The RMSE for each model was roughly around:

Emotion	MLP	SVR	NuSVR
Valence	0.33975230	0.33887909	0.34122344
Arousal	0.24657109	0.2443844	0.24528608
Dominance	0.20402400	0.2024209	0.20322541

Note: The above optimal RMSE values were obtained with the following hyperparameters:

For MLP - Hidden layer units = (10, 5)
Alpha = 1e-4
Learning rate = 0.0001
Activation = Logistic
Optimization = Adam

For SVR - Kernel = Radial Basis Function
C=0.8
epsilon=0.2

Unfortunately, we noticed that on training the above ML models on Emobank, it was not able to generalize well. We tried basic regularization techniques such as L1 and L2 regularization but it did not affect the performance. We are in the process of applying Dropout regularization. Also, Lemmatization had little to no effect on our models.

Some of the charts from our trained model:

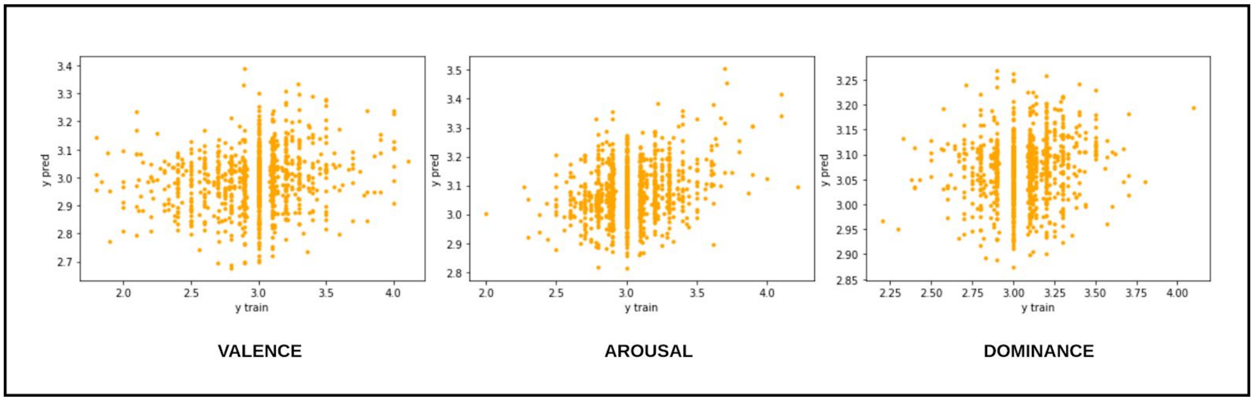


Fig 2: The above chart shows a scatter plot of predicted Y values and actual Y values on Emobank's **validation set** after training on VAD space.

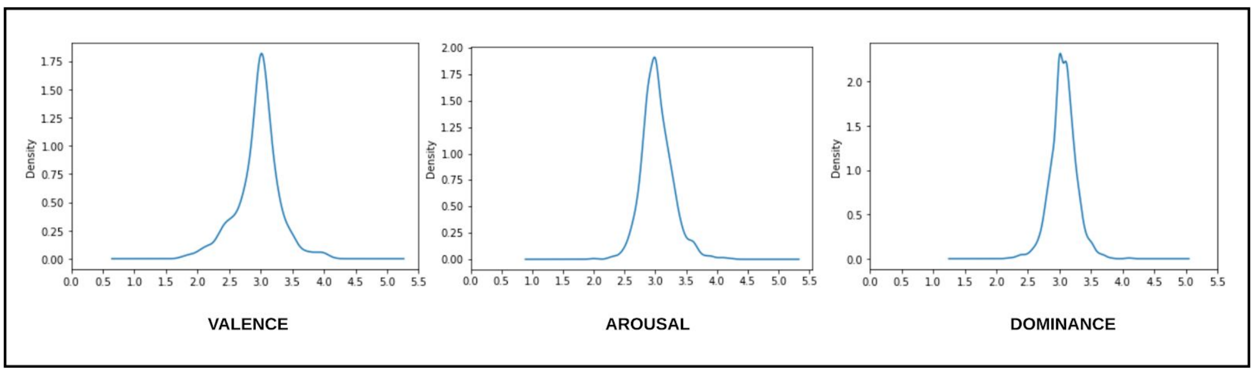


Fig 3: The above chart shows the density curves of the Valence, Arousal, and Dominance space in Emobank's validation set (Y). It can be observed that emotions are primarily around 2.5 to 3.5 space.

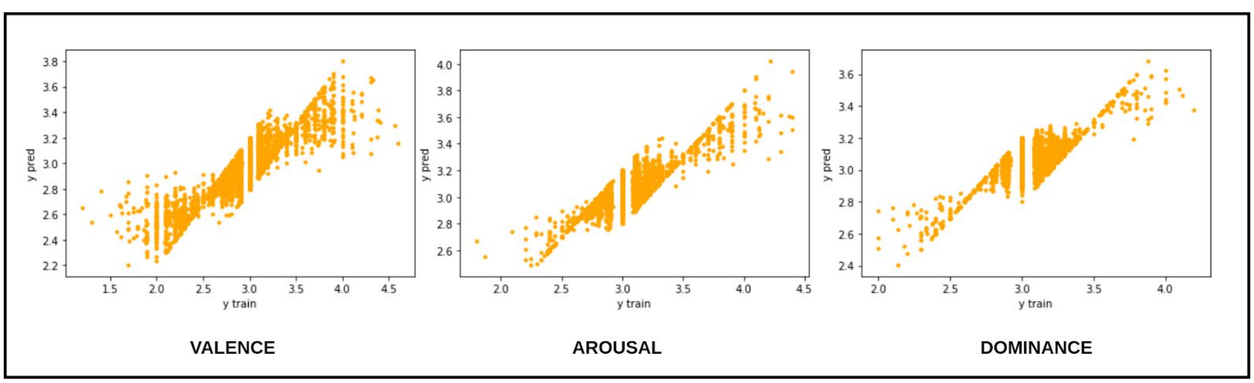


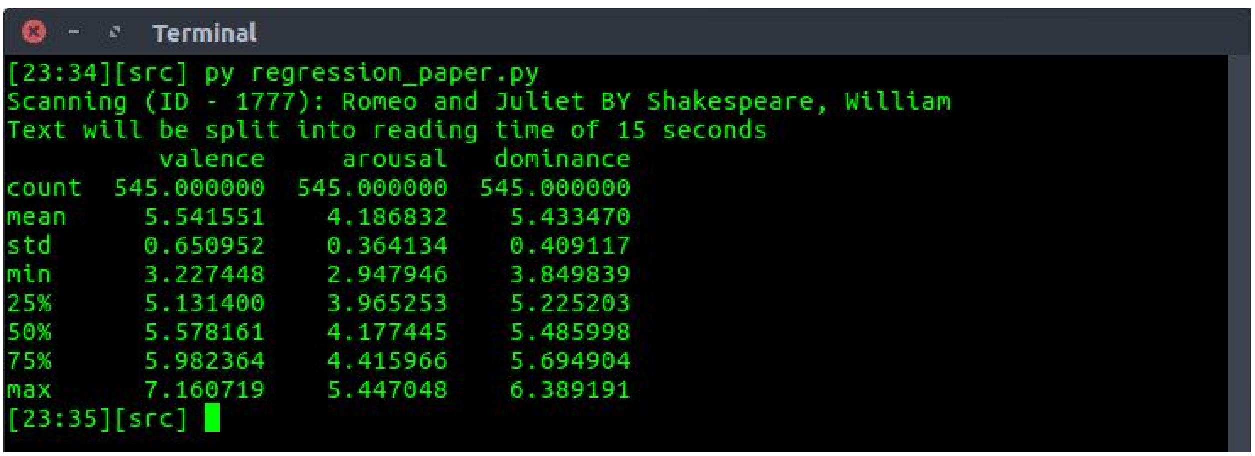
Fig 4: The above chart shows a scatter plot of predicted Y values and actual Y values on Emobank's **training set** after training on VAD space.

We then tried to train the rule-based model (Fig 1.) on books from Project Gutenberg. We have chosen to divide the entire book into paragraphs, such that each paragraph has a reading time of 15 seconds. This was done using the algorithm:

```
words = word_tokenize(text)
reading_minutes = int(len(words)/200)
reading_seconds = int((len(words)/200 - reading_minutes)*60)
```

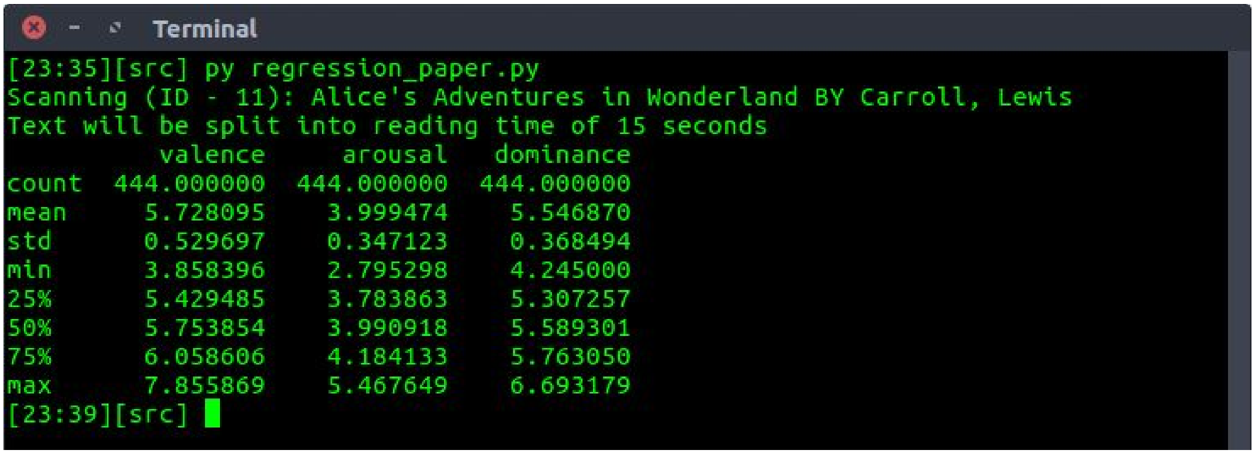
The above algorithm is based on the conclusion by James McNair **[12]**, that an average human has a reading rate of 200 words per minute.

After dividing the book into 15-second paragraphs, we then try to predict the point in VAD space corresponding to that text. Here are some of the results:



```
Terminal
[23:34][src] py regression_paper.py
Scanning (ID - 1777): Romeo and Juliet BY Shakespeare, William
Text will be split into reading time of 15 seconds
         valence     arousal   dominance
count  545.000000  545.000000  545.000000
mean     5.541551    4.186832    5.433470
std      0.650952    0.364134    0.409117
min      3.227448    2.947946    3.849839
25%      5.131400    3.965253    5.225203
50%      5.578161    4.177445    5.485998
75%      5.982364    4.415966    5.694904
max      7.160719    5.447048    6.389191
[23:35][src]
```

Fig 5: The above output shows the VAD space (1-9 range) statistics for Shakespeare's Romeo and Juliet (which is a tragedy by nature).



```
Terminal
[23:35][src] py regression_paper.py
Scanning (ID - 11): Alice's Adventures in Wonderland BY Carroll, Lewis
Text will be split into reading time of 15 seconds
           valence     arousal   dominance
count  444.000000  444.000000  444.000000
mean     5.728095    3.999474    5.546870
std      0.529697    0.347123    0.368494
min      3.858396    2.795298    4.245000
25%      5.429485    3.783863    5.307257
50%      5.753854    3.990918    5.589301
75%      6.058606    4.184133    5.763050
max      7.855869    5.467649    6.693179
[23:39][src]
```

Fig 6: Above output shows the VAD space (1-9 range) statistics for Lewis Carroll's Alice's Adventures in Wonderland (which is a fantasy by nature).

Note: Our results are in the preliminary stage and we have yet to explore this space. Also, we observed that as we increased the paragraph's reading time to 30 seconds or higher, it was unable to capture the emotions of the text and would be stuck at the mean (4.5 to 6 space in a 1-9 VAD space).

For the questionnaire part, we are currently reading the literature by Mayer, J. D., et al **[13]** and D. von Zerssen **[14]**.

To debug the Emobank dataset and understand it better, we trained a model to predict the emotions present in each text of the Emobank dataset. This was done using a dataset of mean and standard deviations (SD) of different emotions on VAD dimensions **[10]**. We used this dataset to synthetically generate our data by finding truncated normal continuous random variables using mean and standard deviations of each emotion. The lower bound was kept at (mean - standard deviation), whereas the upper bound was at (mean + standard deviation). So the values generated were from the highly-dense 68% space (Gaussian graph analogy).

We trained this synthetically generated training set N different times on N K-nearest neighbor models. Some of the results obtained are as follows:

```
[00:29][src] py
Python 3.6.10 (default, Dec 19 2019, 23:04:32)
[GCC 5.4.0 20160609] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import random
>>> import pandas as pd
>>> df = pd.read_csv(open("model.csv"))
>>>
>>> example_1 = random.randint(0, len(df))
>>> df.iloc[example_1][["V","A","D"]]
V      5
A    4.8
D    4.8
Name: 3510, dtype: object
>>> df.iloc[example_1]["text"]
'Holding Dvorov's directions, Nepthys took us around the corner of the building into a skinny alleyway.'
>>> df.iloc[example_1]["emotions"]
"{'hate': 17, 'shock': 25, 'sentimental': 28, 'joy': 7, 'depressing': 13, 'terrible': 10}"
>>>
>>> example_2 = random.randint(0, len(df))
>>> df.iloc[example_2][["V","A","D"]]
V    6.56
A    5.22
D    5.88
Name: 802, dtype: object
>>> df.iloc[example_2]["text"]
'"I did a lot of walking, got an interview and got a job and an opportunity of a lifetime,"'
>>> df.iloc[example_2]["emotions"]
"{'happy': 52, 'lovely': 12, 'fun': 24, 'joy': 5, 'exciting': 2, 'love': 5}"
>>>
```

Fig 7: The above output shows emotions predicted by the model on two indices. To prevent any bias, two randomly generated indices were chosen. While the first example shows mixed emotions such as hate, shock, and sentimental, example 2 clearly shows happy and fun emotions. The emotions are mapped using VAD values predicted by Emobank's human labelers.

To familiarize with Emotion Analysis, we found Speech and Language Processing [4] to be a great resource. Chapter 21 introduces concepts such as terminologies like valence, arousal, dominance; Plutchik's wheel of emotion; the Scherer typology of affective states; General Inquirer, NRC (EmoLex), and other lexicons; Creating Affect Lexicons by Human Labeling; Semi-supervised techniques such as Semantic Axis Methods and Label Propagation; Potts diagram; Dirichlet Prior, etc.

Methodology

In this section, the code along with the various parameter hypertuning is delineated. The entire code is available on: https://github.com/pncnmnp/Affective-Analysis. As this is a private repository, do write me a mail on parthparikh1999p@gmail.com for access.

As observed in the *Literature Survey* section, the rule-based methods along with the doc2vec based implementation (Gensim) gives a high RMSE error for

Valence, Arousal and Dominance on Emobank. To scale it from Emobank to Gutenberg corpus, we first extract the doc2vec vectors of each Gutenberg corpus. This is done by first parsing each book from Gutenberg and stripping unwanted text such as licenses, metadata, header, date of publication, and alternate links. The parsed text is then *paragraph tokenized* in such a manner that each paragraph has an estimated reading time of M seconds. The detailed explanation for the same is in the *Literature Survey* section.

These paragraphs are then passed through a custom pipeline whose first step is cleaning each paragraph for doc2vec. In this phase, we strip whitespaces, convert text to lowercase, remove escape sequences (such as `\n` and `\t`), separate out each punctuation, and split out each word. The end result is a list of lists, with each sub-list containing the list of processed words. The number of sublists is the same as the number of tokenized paragraphs.

The next stage of pipelining consists of labelizing the lists (from text cleaning). During the training and validation phase, labels TRAIN and VAL are used however when parsing the Gutenberg corpus for vectors, a TEST label is provided. We then obtain the Distributed Memory (DM) and Distributed Bag of Words (DBOW) models of doc2vec using the following configuration **[15]**:

1. min_count: It ignores all words with frequency less than min_count. Tuned to min_count=1.
2. window: The maximum distance between the current and predicted word within a sentence. Tuned to window=4.
3. sample: Threshold to downsample high frequency words. Tuned to 0.001.
4. Feature vector size was tuned to 300.

With the above configuration, we built a vocab for each book in Gutenberg corpus and trained it for 20 epochs.

The doc2vec vectors obtained were then used to predict valence, arousal and dominance for tokenized paragraphs of each book. This was done by using the ML models trained on Emobank for predicting the book's doc2vec vectors.

Although the model was able to generalize well to Emobank's corpus, we observed that the above trained pipeline was not scaling to the Gutenberg corpus. To overcome this, we tried training a BERT model [18].

The BERT model was again trained on Emobank and Pytorch transformers were primarily used for this task. We used DistilBERT which is a lighter version of BERT developed by Huggingface. It primarily consumes less memory and is faster than BERT. Pretrained weights - `distilbert-base-uncased` were used for obtaining the embeddings. Pytorch's DistilBertTokenizer was used to tokenize each sentence of Emobank for training.

In *Tokenization* step, each text from Emobank is broken into tokens, [CLS] (classification) tokens are added at the start of the first sentence and [SEP] tokens are added between two sentences of the same text. These tokens are then substituted with their individual ids. To process all the training examples in one batch, padding is required for each training example. For this, the length of the largest training example is obtained, which in case of Emobank is `153`. All the training examples are then padded with ids of value `0` till they match the length of the largest training example. To inform BERT of the padding, we create an Attention mask, where position is `0` if id is `0`, else `1`. We then process the entire Emobank model by first creating tensors for training examples and attention masks, and then training the model on these tensors. Hidden states corresponding to the entire model are then returned after training.

As we were performing the above on Google Colab, we observed that it was facing memory issues processing the entire Emobank model (`~10000`) as a batch. We observed that creating sub-batches of `100x100` was an optimal alternative. However, even with this, it took approximately `4-5` hours to generate all the `10000` hidden states. These hidden states were flattened and the output corresponding to the CLS token (the first token, which can be thought of as providing embeddings for the entire sentence) were obtained for all the `10000` training examples. These embeddings were then split into training, validation and test sets and trained using SVR (support vector regression) corresponding to the Emobank's valence, arousal and dominance values. We primarily used SVR as it provided optimal and consistent performance on rule-based and Gensim implementations. The results obtained on Emobank are:

```
{'model': Pipeline(memory=None,
         steps=[('standardscaler',
                 StandardScaler(copy=True, with_mean=True, with_std=True)),
                ('svr',
                 SVR(C=0.8, cache_size=200, coef0=0.0, degree=3, epsilon=0.2,
                     gamma='scale', kernel='rbf', max_iter=-1, shrinking=True,
                     tol=0.001, verbose=False))],
         verbose=False), 'score_train': 0.7355896455313307, 'score_val': 0.4410507437568624, 'rmse_val': 0.2553075189350901, '
rmse_train': 0.17947462985272425}
```

Fig 8: The above figure shows the **valence** scores and RMSE values obtained using the BERT embeddings. As mentioned in the *Literature Survey* section, the epsilon, C, and kernel type remains unchanged. Both the training and validation RMSE values are **significantly** lower than those obtained using Gensim. While Gensim was providing `rmse_val=0.339`, BERT provides `rmse_val=0.255`.

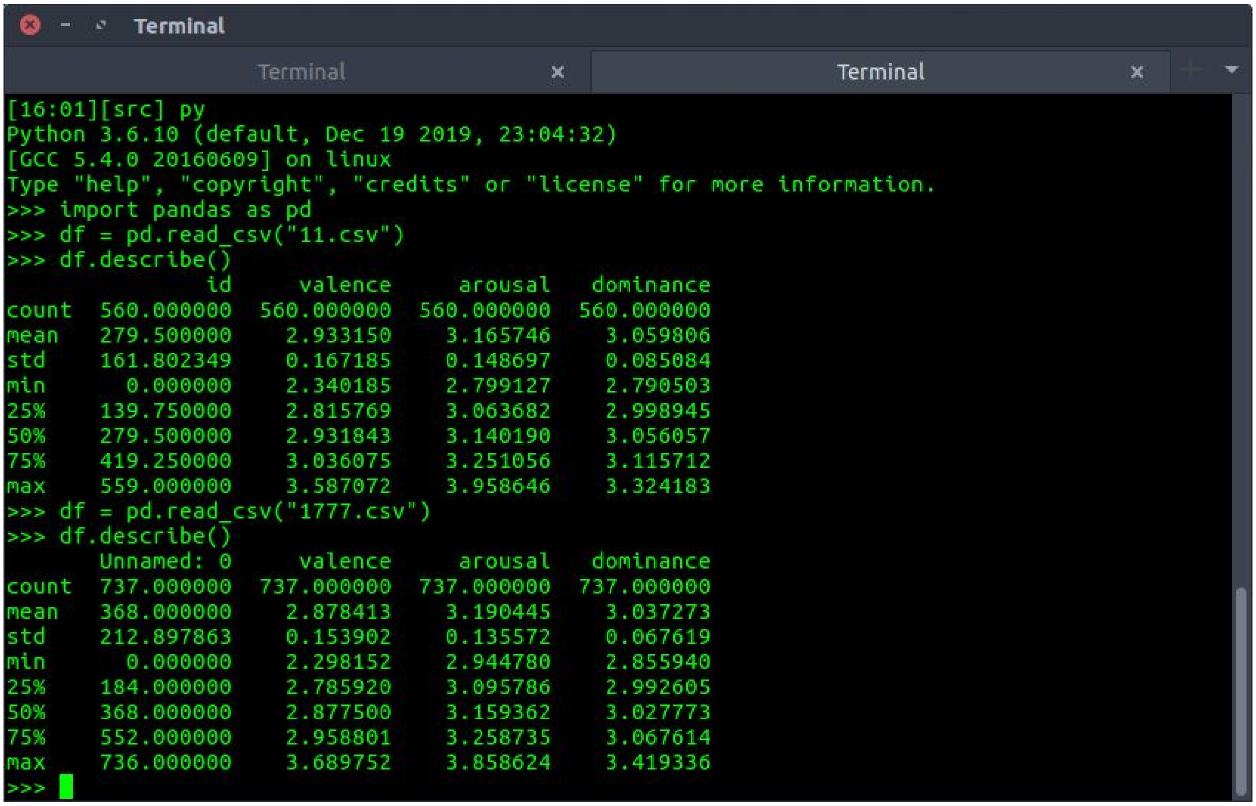
```
{'model': Pipeline(memory=None,
         steps=[('standardscaler',
                 StandardScaler(copy=True, with_mean=True, with_std=True)),
                ('svr',
                 SVR(C=0.8, cache_size=200, coef0=0.0, degree=3, epsilon=0.2,
                     gamma='scale', kernel='rbf', max_iter=-1, shrinking=True,
                     tol=0.001, verbose=False))],
         verbose=False), 'score_train': 0.6197145886598122, 'score_val': 0.3351597215142541, 'rmse_val': 0.2174676998428308, '
rmse_train': 0.15946283769480094}
```

Fig 9: The above figure shows the **arousal** scores and RMSE values obtained using the BERT embeddings. As mentioned in the *Literature Survey* section, the epsilon, C, and kernel type remains unchanged. Both the training and validation RMSE values are lower than those obtained using Gensim. While Gensim was providing `rmse_val=0.244`, BERT provides `rmse_val=0.217`.

```
{'model': Pipeline(memory=None,
         steps=[('standardscaler',
                 StandardScaler(copy=True, with_mean=True, with_std=True)),
                ('svr',
                 SVR(C=0.8, cache_size=200, coef0=0.0, degree=3, epsilon=0.2,
                     gamma='scale', kernel='rbf', max_iter=-1, shrinking=True,
                     tol=0.001, verbose=False))],
         verbose=False), 'score_train': 0.494491490335783, 'score_val': 0.07534107336064744, 'rmse_val': 0.19160460024338652,
'rmse_train': 0.1496677505418944}
```

Fig 10: The above figure shows the **dominance** scores and RMSE values obtained using the BERT embeddings. As mentioned in the *Literature Survey* section, the epsilon, C, and kernel type remains unchanged. Both the training and validation RMSE values are lower than that obtained using Gensim.

To scale these models to Gutenberg corpus, each book is *paragraph tokenized* as mentioned earlier. However we do not perform the preprocessing as observed in the Gensim model, we perform the above mentioned BERT preprocessing which consists of tokenization, padding, and masking. The BERT embeddings of each paragraph are then obtained using the same method as above. It takes around 30 to 60 minutes to obtain the BERT embeddings. Once the embeddings are obtained, the above mentioned SVR-BERT model (trained on Emobank) is used to predict these paragraph embeddings for valence, arousal, and dominance dimensions.



```
Terminal
Terminal        Terminal
[16:01][src] py
Python 3.6.10 (default, Dec 19 2019, 23:04:32)
[GCC 5.4.0 20160609] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import pandas as pd
>>> df = pd.read_csv("11.csv")
>>> df.describe()
               id     valence     arousal   dominance
count  560.000000  560.000000  560.000000  560.000000
mean   279.500000    2.933150    3.165746    3.059806
std    161.802349    0.167185    0.148697    0.085084
min      0.000000    2.340185    2.799127    2.790503
25%    139.750000    2.815769    3.063682    2.998945
50%    279.500000    2.931843    3.140190    3.056057
75%    419.250000    3.036075    3.251056    3.115712
max    559.000000    3.587072    3.958646    3.324183
>>> df = pd.read_csv("1777.csv")
>>> df.describe()
       Unnamed: 0     valence     arousal   dominance
count  737.000000  737.000000  737.000000  737.000000
mean   368.000000    2.878413    3.190445    3.037273
std    212.897863    0.153902    0.135572    0.067619
min      0.000000    2.298152    2.944780    2.855940
25%    184.000000    2.785920    3.095786    2.992605
50%    368.000000    2.877500    3.159362    3.027773
75%    552.000000    2.958801    3.258735    3.067614
max    736.000000    3.689752    3.858624    3.419336
>>>
```

Fig 11: The statistics of valence, arousal and dominance scores obtained after fine-tuning using SVR-BERT model and the book's BERT embeddings. Book ID 11 is *Alice in Wonderland by Lewis Carroll*, and Book ID 1777 is *William Shakespeare's Romeo and Juliet*. The range is from 1-5. The paragraph tokenization was done with reading time of 10 seconds.

As observed in Fig. 11, the median (valence) of Romeo and Juliet is 0.054 less than that of Alice in Wonderland. However, based on our understanding, it should be significantly less considering that Romeo and Juliet is a tragedy, whereas Alice in Wonderland is a children's fantasy.

From the *Literature Review* section, to debug the Gutenberg dataset and understand it better, we trained a KNN-based model to predict the emotions present in each *tokenized paragraph*. This was done using a dataset of mean and standard deviations (SD) of different emotions on VAD dimensions **[10]** as mentioned in Fig 12.

```
Terminal
Terminal                                   Terminal
[16:21][lemma] py
Python 3.6.10 (default, Dec 19 2019, 23:04:32)
[GCC 5.4.0 20160609] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import pandas as pd
>>> df = pd.read_csv("emotions.csv")
>>> df
        emotion  V_mean   V_std  A_mean   A_std  D_mean   D_std
0           fun  6.8571  1.3015  5.8571  2.1993  6.0000  1.5584
1      exciting  5.9286  2.0516  6.9286  1.9808  5.5000  2.4128
2         happy  7.1429  1.1867  4.8571  1.4569  5.2143  1.3190
3           joy  6.9333  2.3228  6.4667  1.9276  5.8000  2.0067
4      cheerful  5.9286  1.7914  3.3571  1.3420  4.9286  1.7914
5          love  6.5714  1.3997  4.2143  2.5122  5.4286  2.0603
6        lovely  6.4667  1.3597  4.0000  1.7889  4.9333  1.9137
7   sentimental  4.2000  1.4236  3.7333  1.8062  3.9333  1.9482
8    melancholy  3.3333  1.1926  4.4667  1.9956  3.2000  1.3760
9           sad  3.3333  1.3499  2.9333  1.6918  4.6667  2.1499
10   depressing  4.2000  1.6411  3.6000  1.2000  4.6000  1.8184
11       mellow  4.2000  1.7963  3.0000  1.5055  3.3333  1.7764
12     terrible  3.6667  1.4907  5.4667  2.0613  4.6000  1.7436
13        shock  4.6667  1.4907  6.4000  1.9253  4.9333  1.5691
14         hate  3.9333  2.0483  6.1333  2.0934  5.5333  1.8927
>>>
```

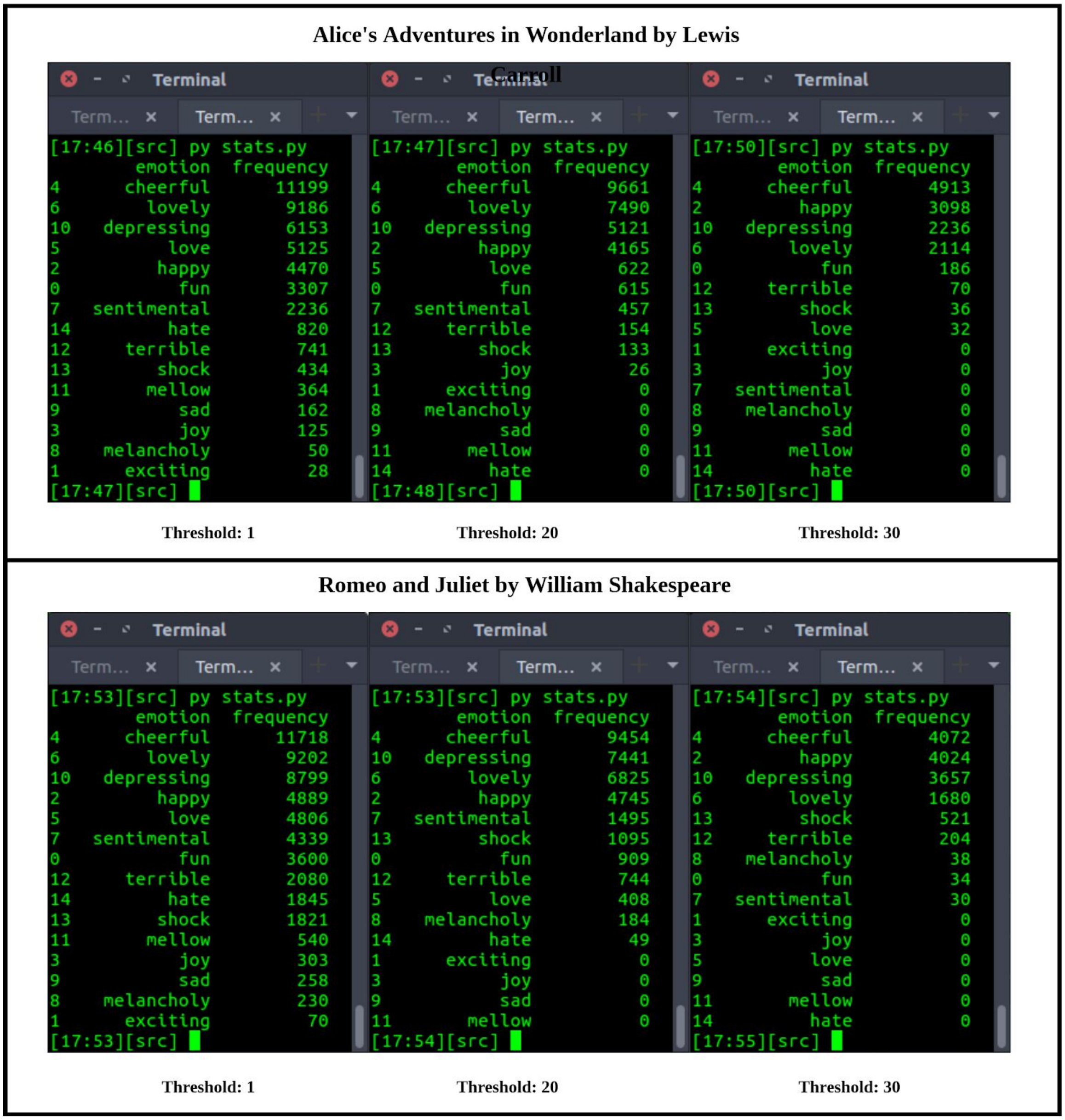
Fig 12: Mean and standard deviations (SD) of different emotions on VAD dimensions as observed by Verma, Gyanendra and Tiwary, Uma Shanker **[10]**.

```
Terminal
Terminal                                   Terminal
>>> df = pd.read_csv("emotions_Warriner.csv")
>>> df
        emotion  V_mean  V_std  A_mean  A_std  D_mean  D_std
0           fun    8.37   0.96    6.32   2.62    7.04   1.85
1      exciting    7.95   1.62    6.95   2.67    7.21   1.87
2         happy    8.47   1.28    6.05   2.13    7.21   1.96
3           joy    8.21   1.18    5.55   2.85    7.00   2.05
4      cheerful    8.00   1.41    5.76   2.41    7.32   1.76
5          love    8.00   1.39    5.36   3.23    5.92   2.92
6        lovely    7.55   1.65    3.95   2.75    6.38   2.42
7   sentimental    6.27   2.10    2.95   2.11    6.73   2.21
8    melancholy    3.74   1.69    4.13   2.56    3.96   2.16
9           sad    2.10   0.91    3.49   2.21    3.84   2.21
10   depressing    2.48   1.69    3.00   2.11    3.57   2.31
11     terrible    2.10   1.12    4.39   2.66    3.40   2.17
12        shock    3.90   2.05    5.95   2.86    5.79   2.94
13         hate    1.96   1.33    6.26   2.31    4.47   2.54
>>>
```

Fig 13: Mean and standard deviations (SD) of different emotions on VAD dimensions from Warriner, et al. **[2]**.

We used these datasets to synthetically generate our data by finding truncated normal continuous random variables using mean and standard deviations of each emotion. As mentioned before, the lower and upper bounds were retained at (mean - standard deviation) and (mean + standard deviation). The values generated are from highly-dense 68% space (Gaussian graph analogy).

Like with Emobank, we trained this synthetically generated training set N different times on N K-nearest-neighbor models. Here, we kept `N=100` with the length of the training corpus at 10^6. For the KNN model, K=5 was used with all points in each neighborhood weighted by the inverse of their distance.



Alice's Adventures in Wonderland by Lewis Carroll

```
[17:46][src] py stats.py
        emotion  frequency
4      cheerful      11199
6        lovely       9186
10   depressing       6153
5          love       5125
2         happy       4470
0           fun       3307
7   sentimental       2236
14         hate        820
12     terrible        741
13        shock        434
11       mellow        364
9           sad        162
3           joy        125
8    melancholy         50
1      exciting         28
[17:47][src]
```

Threshold: 1

```
[17:47][src] py stats.py
        emotion  frequency
4      cheerful       9661
6        lovely       7490
10   depressing       5121
2         happy       4165
5          love        622
0           fun        615
7   sentimental        457
12     terrible        154
13        shock        133
3           joy         26
1      exciting          0
8    melancholy          0
9           sad          0
11       mellow          0
14         hate          0
[17:48][src]
```

Threshold: 20

```
[17:50][src] py stats.py
        emotion  frequency
4      cheerful       4913
2         happy       3098
10   depressing       2236
6        lovely       2114
0           fun        186
12     terrible         70
13        shock         36
5          love         32
1      exciting          0
3           joy          0
7   sentimental          0
8    melancholy          0
9           sad          0
11       mellow          0
14         hate          0
[17:50][src]
```

Threshold: 30

Romeo and Juliet by William Shakespeare

```
[17:53][src] py stats.py
        emotion  frequency
4      cheerful      11718
6        lovely       9202
10   depressing       8799
2         happy       4889
5          love       4806
7   sentimental       4339
0           fun       3600
12     terrible       2080
14         hate       1845
13        shock       1821
11       mellow        540
3           joy        303
9           sad        258
8    melancholy        230
1      exciting         70
[17:53][src]
```

Threshold: 1

```
[17:53][src] py stats.py
        emotion  frequency
4      cheerful       9454
10   depressing       7441
6        lovely       6825
2         happy       4745
7   sentimental       1495
13        shock       1095
0           fun        909
12     terrible        744
5          love        408
8    melancholy        184
14         hate         49
1      exciting          0
3           joy          0
9           sad          0
11       mellow          0
[17:54][src]
```

Threshold: 20

```
[17:54][src] py stats.py
        emotion  frequency
4      cheerful       4072
2         happy       4024
10   depressing       3657
6        lovely       1680
13        shock        521
12     terrible        204
8    melancholy         38
0           fun         34
7   sentimental         30
1      exciting          0
3           joy          0
5          love          0
9           sad          0
11       mellow          0
14         hate          0
[17:55][src]
```

Threshold: 30

Fig 14: The above figure shows the emotions obtained in the books *Alice's Adventures in Wonderland* (book `11`) and *Romeo and Juliet* (book `1777`). The paragraph tokenization was done with `M=15` seconds.

As can be seen in Fig 14, with a threshold of `1`, there is a minute difference seen in the two books. Book 11 has more *fun* emotions than *sentimental*, which is opposite for book `1777`. However with threshold of `30`, key emotions of both the books gets a boost and for book `11`, which in reality is a children's fantasy, emotions like cheerful, lovely, happy, love, and fun are predicted. With book `1777`, which in reality is a tragedy and contains themes like war, love, and tragedy, primary emotions predicted are: cheerful, depressing, lovely, happy, sentimental, and shock. The first `7` emotions and their weights are more stark when the threshold is `40`.

Another important observation from Fig `14` is that it appears that the model is overfitting on the *cheerful* emotion. The values from figure `12` were used to create the models. We assumed that the overfitting was caused because the model was predicting VAD values in the neutral space to emotions such as *cheerful and lovely.* To test this assumption, we removed the VAD values in space `4.5-5.5`. Also, to improve performance, we used a Radius Neighbor Classifier, which classifies using a vote among neighbors within a given radius. This improves performance by removing VAD values which do not fall under any range or with a weak correlation to some particular *emotion*. The results for the same are as follows:

Fig 15: The above figure shows the emotions obtained in the books *Alice's Adventures in Wonderland* (book `11`) and *Romeo and Juliet* (book `1777`) after using a Radius Neighbor Classifier and removing values of range `4.5-5.5`. The paragraph tokenization was done with `M=15` seconds. Also, the radius was kept at `0.2`.

As can be observed in Fig 15, there is slight performance improvement in book `1777`, however the improvement seen in book `11` is negligible.

Apart from these tunings, we have designed an entire pipeline which performs the following: *Paragraph tokenize* the book → find valence, arousal, and dominance of each paragraph → find the emotions of all the paragraphs.

Although we have shown 2 books throughout this report, the performance remains comparable throughout the Gutenberg corpus. Some of the issues we are presently tackling are:

1. Neutral Space: VAD values in neutral or close to neutral space are hard to tackle. We have tried removing values between the `4.5-5.5` range, giving more weight to emotions within a particular radius, and boosting the valence emotion. While these techniques offer minor improvements, overfitting is still observed in the emotion prediction. We have also tried using mean and standard deviations (SD) as observed in Fig 13. The results are shown in Fig 16.
2. Training Model: We believe this is a major bottleneck which is halting the performance of our entire model. The Gensim and BERT models are being trained on Emobank corpus. While this corpus contains sentences, many of these sentences are tweets or twitter/facebook like sentences. The issue with these kinds of training data is that each training example is *filled with emotions*. When we scale the model trained on such data to Gutenberg corpus, our model finds test examples which contain subtle emotions and are not richly filled with emotions. This training data - test data mismatch is a major issue. To overcome this situation, we need labeled training data as seen in the Gutenberg corpus. Unfortunately, such data has not been created by any individual or entity. The alternatives like:
    a. 4000 stories with sentiment analysis dataset [5]: We have found this dataset to be highly skewed. As can be observed from Figure 17, the number of examples with valence or dominance less than `0.5` is negligible. A similar observation can be made with arousal values wherein all the examples have arousal `< 0.5`. Therefore the dataset contains examples which have low arousal and medium to high dominance and valence. Fig 18 shows the statistical analyses of this dataset.
    b. Affective Norms for English Text (ANET) [7]: The primary issue with ANET is that it contains `120` training examples. While this works for rule-based approaches, it becomes incredibly difficult to create ML models without severely overfitting the dataset.

3. Semantic Understanding: The Gutenberg corpus contains books which are published across many centuries. The Gensim model and BERT model primarily have semantic understanding of modern text. Therefore gaining semantic knowledge of text written in the Victorian era or other medieval eras is incredibly difficult. It can be argued that an average human can face a similar issue while finding emotions from medivial books.

Figure 16: Emotions predicted using the mean and standard deviations (SD) from Warriner (Fig 13). As can be observed, these mean and SD further overfits on certain emotions. The following results were calculated by using K Nearest Neighbour implementation without removing the `4.5-5.5` range.

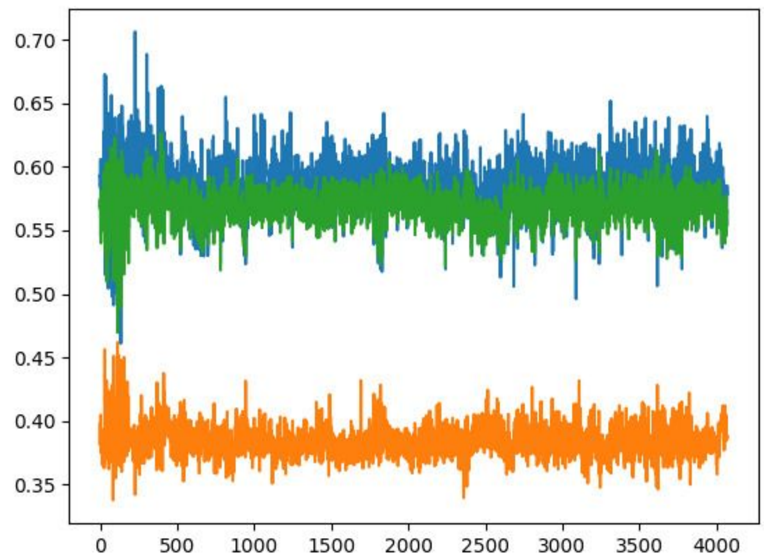


Fig 17: Shows the distribution of dataset from **[5]**. As observed the data appears to be highly skewed in the valence, arousal, and dominance space. The color codes are: Valence-Blue, Arousal-Orange, Dominance-Green. The dataset is in the range `0-1`.

```
[21:17][corpus] py
Python 3.6.10 (default, Dec 19 2019, 23:04:32)
[GCC 5.4.0 20160609] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import pandas as pd
>>> df = pd.read_csv("4000-Stories-with-sentiment-analysis.csv")
>>> df[['length', 'valence', 'arousal', 'dominance', 'word_count', 'sentence_length', 'string_length', 'unique_words', 'lexical_diversity'
]].describe()
             length      valence      arousal    dominance    word_count  sentence_length  string_length  unique_words  lexical_diversity
count   4071.000000  4071.000000  4071.000000  4071.000000   4071.000000      4071.000000    4071.000000   4071.000000        4071.000000
mean    4377.427413     0.583720     0.384278     0.569728   2168.737165        19.891637       5.484378    868.253255           0.480057
std     3820.637473     0.020509     0.011598     0.012639   1880.779769         6.811405       0.214577    544.569275           0.134247
min       21.000000     0.460964     0.337950     0.469683      9.000000         3.053947       3.134698      9.000000           0.139151
25%     1928.000000     0.570921     0.376795     0.561931    941.500000        15.295606       5.361048    508.000000           0.380694
50%     3497.000000     0.584304     0.383348     0.570493   1745.000000        18.566667       5.499499    775.000000           0.465655
75%     5664.500000     0.596421     0.390402     0.577928   2835.500000        23.182683       5.623999   1145.000000           0.561816
max    51725.000000     0.706250     0.461964     0.626766  25412.000000       122.000000       6.544162   5373.000000           1.000000
>>>
```

Fig 18: Statistically analyzes the distribution of the dataset from **[5]**. As can be observed, standard deviation is low for valence, arousal, and dominance values.

As seen in Fig 15, removing values of range `4.5-5.5` improved the accuracy by preventing overfitting of the neutral emotions. To further improve the accuracy, it was observed that by calculating the mean and standard deviation of valence, arousal, and dominance for a particular book and removing the range (`VAD_mean-VAD_std`)-(`VAD_mean+VAD_std`), we can obtain a much better accuracy. For the rest of this report, we call this technique as *mean-std*.

Hyperparameter tuning for the models described in Figure 14 and 15 was done. For further exploration, Random Forest Classifier and Adaboost models were trained. The *B* values in the *status* column are ones which perform the best for a given ML model while the *W* are the ones which had a significantly worse performance. The remaining models have a performance near to the *B* status models. The results of the three ML models are as follows:

Models - KNN

KNN is done with 100 models and training length of 10^5 (for each model).

Status	K	Neutral technique	Weights	Comments
W	10	4-6	distance	Severely Overfits
	10	4.5-5.5	distance	
	10	mean-std	distance	
	7	mean-std	distance	*Sentimental* not there, with k=8 and 9, *sentimental* and *fun* both come in.
	12	mean-std	distance	
	12	4.5-5.5	distance	More preference to *cheerful*
	12	mean-std	distance	Using algorithm ball-tree
B	12	mean-std	uniform	Using algorithm ball-tree. Better negative and positive ranking than *distance*, but negative is ranked a bit lower.

Models - Random Forest

Random Forest is done with 10 models and training length of 10000 (for each model). *Mean-Std* was used as a neutral technique for all the models below.

Status	Max depth	Size of model	Comments
W	15	1.2 GB	Using `max_depth=15`, `min_samples_leaf=4`, `min_samples_split=10`, `max_features=3`.

			`Max_features` causes overfitting.
	2	10-20 MB	
	10	204 MB	
	15	1.8 GB	No major effect was observed with `bootstrap=True` or `n_estimators=200`.
	15	1.4 GB	Using `min_samples_leaf=4`. Reduces size without drop in performance
B	15	1.2 GB	Using `min_samples_leaf=4`, `min_samples_split=1`. Reduces size without drop in performance. Does not work well with `max_depth=20`. Performs slightly worse with `oob_score=True`.

The *B* model mentioned above was used in the final pipeline to calculate emotions of all the books from Gutenberg corpus.

Models - Adaboost

Adaboost is done with 10 models and training length of 10000 (for each model). *Mean-Std* was used as a neutral technique for all the models below.

Status	N estimators	Size of model	Comments
W	100	1.6 MB	Tested without *mean-std.* Model overfitted.
W	300	4.9 MB	Overfits on negative examples.
W	200	3.3 MB	With `algorithm=SAMME`, overfits on *cheerful*, *lovely*, *sentimental.*
	100	1.6 MB	
	200	3.3 MB	Better than with `n_estimators=100`.
	50	817.4 KB	The top 3 emotions are optimal.
B	200	32.6 MB	Trained with 100 models each of 10000 training length.

Apart from the above mentioned model, it was observed that for RadiusNeighbour Classifier, `r=0.1` gives an improvement in performance.

Figures 19 and 20 show the predictions made by Adaboost and Random Forest models using the model with status *B*.

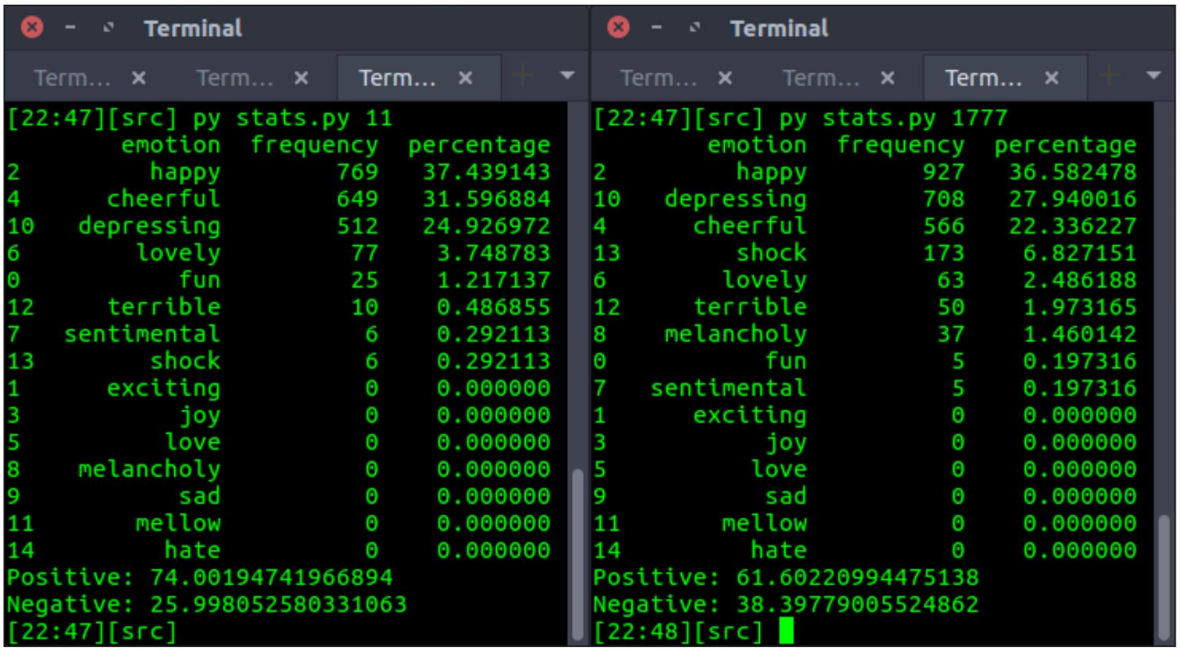


Figure 19: The above figure shows the emotions obtained in the books *Alice's Adventures in Wonderland* (on the left - book `11`) and *Romeo and Juliet* (on the right - book `1777`) after using the Random Forest model and removing values using *mean-std* technique. The maximum depth was kept at `15` and `10` models each of `10000` training examples were used for training, primarily due to memory constraints.

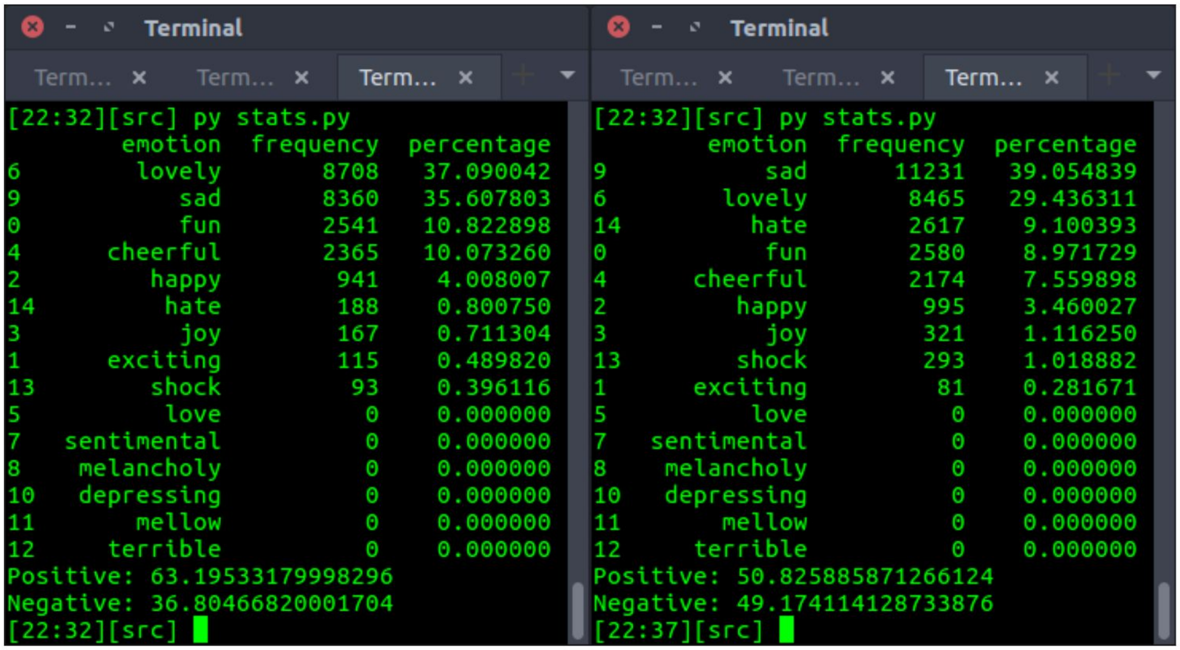


Figure 20: The above figure shows the emotions obtained in the books *Alice's Adventures in Wonderland* (on the left - book `11`) and *Romeo and Juliet* (on the right - book `1777`) after using the Adaboost model and removing values using *mean-std* technique. `200` N estimators were used for training the model.

As seen in figures 19 and 20, both the models worked well in finding the sentiments from Gutenberg corpus. Adaboost overfits slightly less than the Random Forest model. The training time greatly varied for the models. To find emotions present in 100 books, the compute time required was around 8 hours for the Random Forest model, which is being used in the production. Adaboost is observed to perform significantly slower than Random Forest models.

Questionnaire section was primarily designed using two question-asking techniques: *objective* and *subjective* questions. Objective questions consist of fixed-value questions, whereas subjective questions are open-ended questions.

Primary goal of the questionnaire was to understand the reader's mood in the past 24 hours. The answers were then used to recommend books such that reading them would *improve* the reader's mood. There are a few questionnaires which can be used to understand the reader's mood.

Mood and Feelings Questionnaire (MFQ) [19] is a questionnaire used to indicate the presence of depression in a respondent. The adult-long version of the questionnaire contains 33 questions with options: *not true*, *sometimes*, and *true*. The rating for the same are:

Not True	0
Sometimes	1
True	2

As mentioned in the paper, scoring 27 or higher on the long version may indicate the presence of depression in the respondent.

The issue with MFQ is that it becomes difficult to convert the point-scale into VAD space for book recommendations. To overcome such difficulties, a variant of the The Discrete Emotions Questionnaire (DEQ) [20] can be used. The DEQ asks subjective questions such as:
Please remember a SPECIFIC time when someone else was to blame for something bad that happened to you. The person or thing who was at fault harmed you in some way, or prevented you from getting something you wanted. Please think of a negative situation, caused by someone else, in which you experienced an extremely intense emotional response. (for *angry*)

It gives the respondent 180 seconds to answer these questions, following which the questionnaire asks the respondent to explain the same mood using 4 or less words. Count of words on the answers is then performed to understand the intensity of that emotion.

For our objective questionnaire, we asked the respondent to rate different emotions from a scale of 1 to 9 based on their past day's mood (1 is low intensity and 9 is high intensity). We then captured the high intensity emotions, converted them to an optimal VAD space and recommended books with paragraphs near that VAD space. To convert emotions to an optimal VAD space, their VAD_mean and VAD_std were first obtained. Truncated normal continuous random variables were then generated using these mean and standard deviation values.

If the respondent had a negative mood, we would specifically recommend books with a positive VAD space to improve their mood. For a respondent in neutral or positive mood, a wide range of books matching their questionnaire's VAD space would be recommended. Figure 21 shows the objective questionnaire. Figure 22 shows the recommendations made by the model using the respondent's answer from Figure 21.

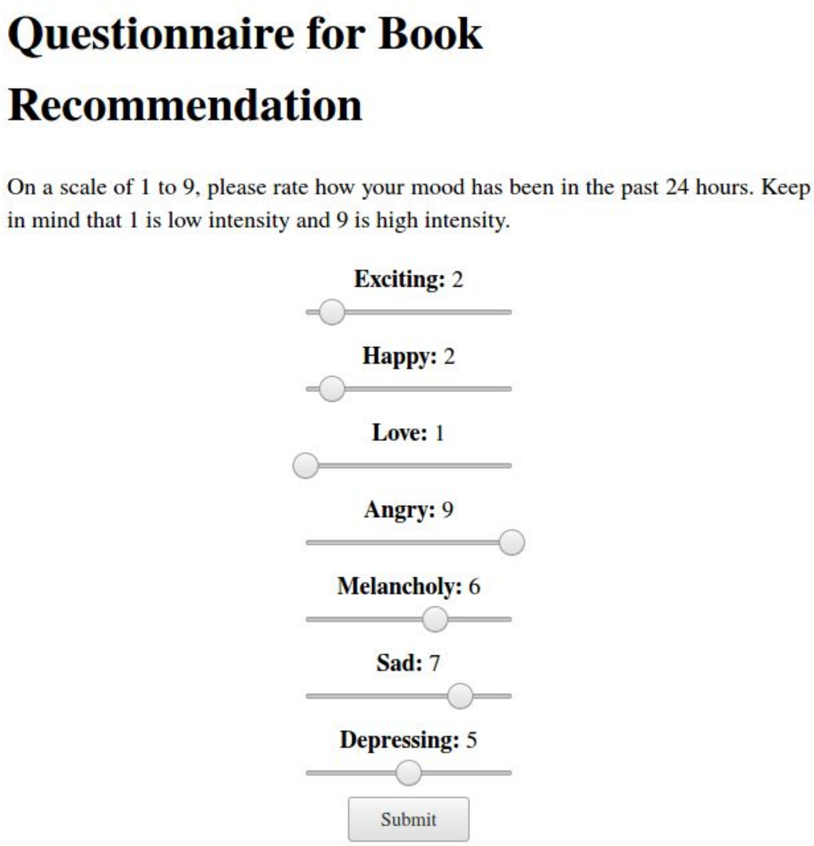


Figure 21: Objective questionnaire with basic emotions as seen in Figure 12. From the above, we can observe that the respondent is in a negative mood space.

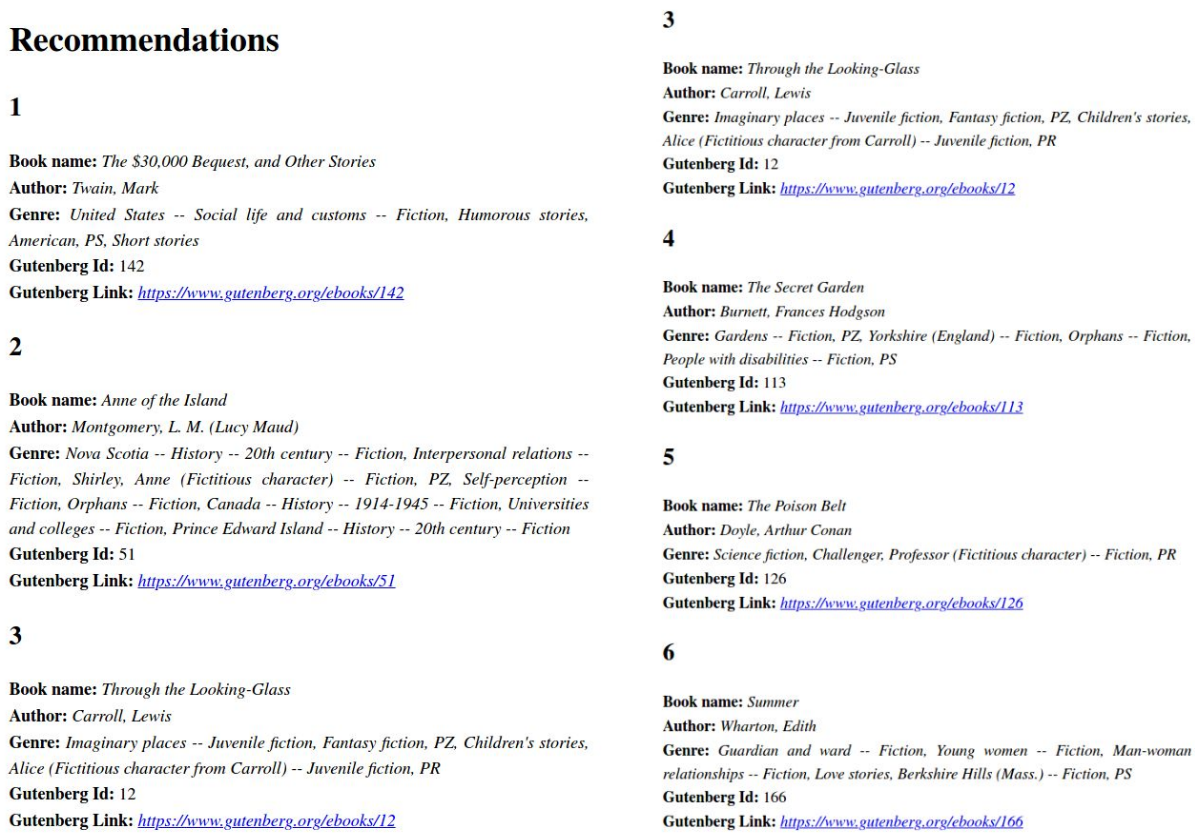


Figure 22: The above figure shows the recommendations generated after submitting the objective questionnaire present in Figure 21. Each book's title, author name, genres, Gutenberg Id and Gutenberg link are sent as metadata. As the respondent's mood in Figure 21 was negative, books with themes such as humour, fantasy fiction, love, and science fiction are recommended.

For the subjective questionnaire, the respondent was first shown the following prompt: *Describe your current mood in the next 180 seconds.* A text box was provided for the respondent to write their current mood along with a submit button. After 180 seconds, the respondent was shown another page with the following question: *In the spaces below, write 4 more words that describe your current mood. If you can't think of 4 words, write as many words as you can, and write 'none' in the other spaces.* The prompts can be observed on Figure 23 and 24.

The VAD space can be calculated from text using numerous techniques mentioned earlier in the *Methodology* section. Once the VAD space is obtained, truncated normal continuous random variables (as mentioned in the Literature *Survey* section) are created using this VAD space with a standard deviation of S, with S typically less than 3. To provide more weightage to the second part of the questionnaire, the frequency of words entered are increased by a constant. We chose the constant as 4.

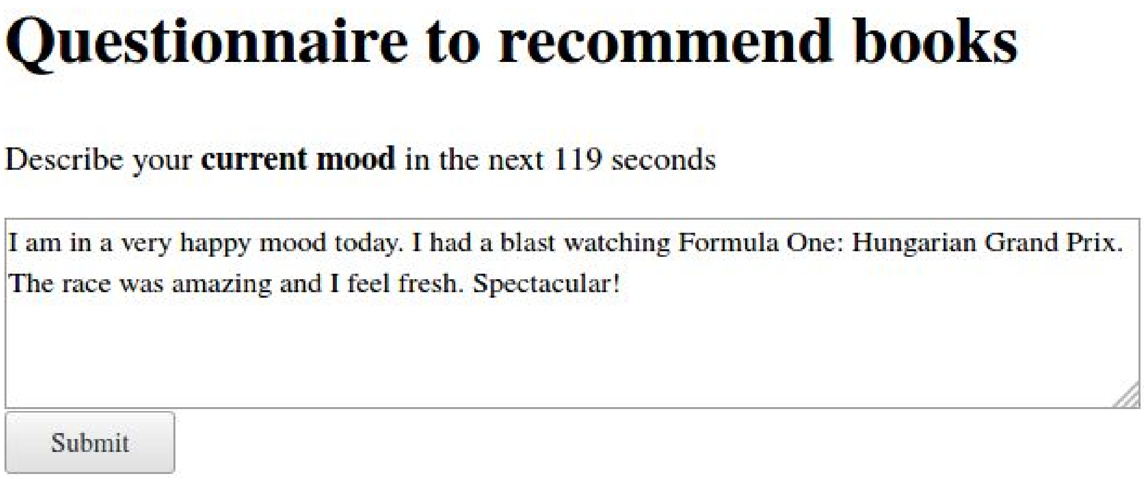


Figure 23: The above figure shows the first part of the subjective questionnaire. A timer starts from 180, and after 3 minutes sends the respondents' answer to the server side. It then renders the next part of the questionnaire as seen in Figure 24.

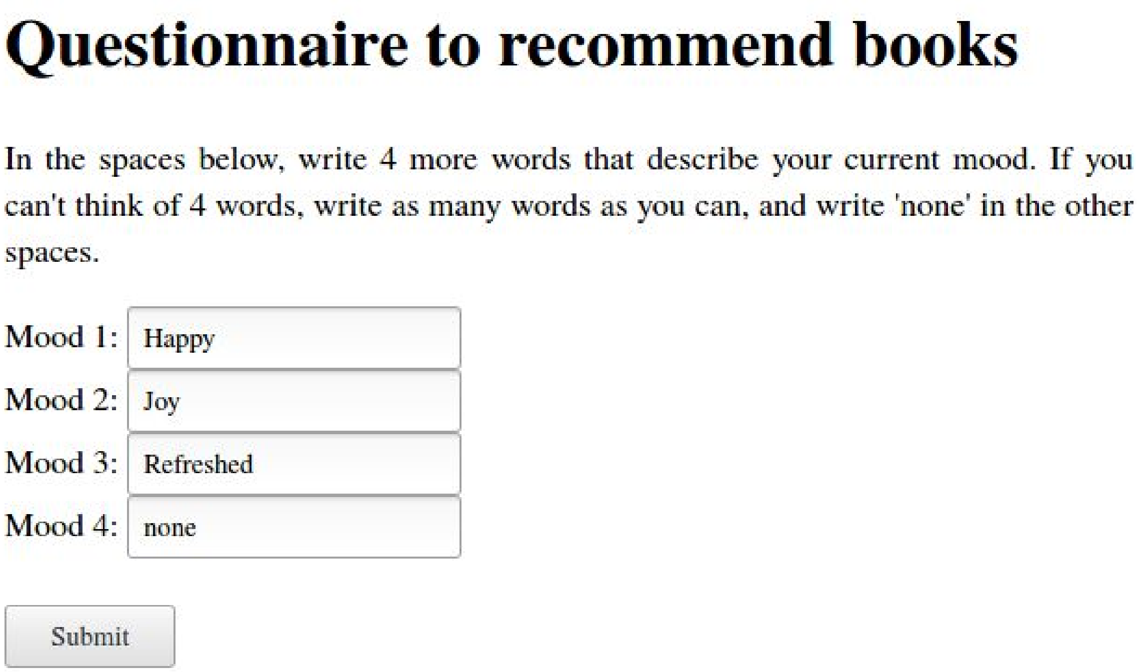


Figure 24: The above figure shows the second part of the subjective questionnaire. One or more boxes can be left empty in case the respondent is unable to think of 4 one-word moods.

Recommendations

1

Book name: *The Return of Sherlock Holmes*
Author: *Doyle, Arthur Conan*
Genre: *Holmes, Sherlock (Fictitious character) -- Fiction, Detective and mystery stories, English, PR*
Gutenberg Id: 108
Gutenberg Link: https://www.gutenberg.org/ebooks/108

2

Book name: *The Awakening, and Selected Short Stories*
Author: *Chopin, Kate*
Genre: *Women -- Louisiana -- New Orleans -- Social conditions -- Fiction, New Orleans (La.) -- Fiction, Louisiana -- Social life and customs -- Fiction, Adultery -- Fiction, Self-actualization (Psychology) -- Fiction, PS*
Gutenberg Id: 160
Gutenberg Link: https://www.gutenberg.org/ebooks/160

5

Book name: *The Prisoner of Zenda*
Author: *Hope, Anthony*
Genre: *Impostors and imposture -- Fiction, British -- Foreign countries -- Fiction, Adventure stories, PR*
Gutenberg Id: 95
Gutenberg Link: https://www.gutenberg.org/ebooks/95

6

Book name: *Alice's Adventures in Wonderland*
Author: *Carroll, Lewis*
Genre: *Imaginary places -- Juvenile fiction, Fantasy fiction, PZ, Alice (Fictitious character from Carroll) -- Juvenile fiction, Children's stories, PR*
Gutenberg Id: 11
Gutenberg Link: https://www.gutenberg.org/ebooks/11

3

Book name: *The Secret Garden*
Author: *Burnett, Frances Hodgson*
Genre: *Gardens -- Fiction, PZ, Yorkshire (England) -- Fiction, Orphans -- Fiction, People with disabilities -- Fiction, PS*
Gutenberg Id: 113
Gutenberg Link: https://www.gutenberg.org/ebooks/113

4

Book name: *Anne of Green Gables*
Author: *Montgomery, L. M. (Lucy Maud)*
Genre: *Girls -- Fiction, Bildungsromans, PZ, Shirley, Anne (Fictitious character) -- Fiction, Orphans -- Fiction, Islands -- Fiction, Friendship -- Fiction, Country life -- Prince Edward Island -- Fiction, Prince Edward Island -- History -- 20th century -- Fiction, Canada -- History -- 1867-1914 -- Fiction*
Gutenberg Id: 45
Gutenberg Link: https://www.gutenberg.org/ebooks/45

7

Book name: *Anne of the Island*
Author: *Montgomery, L. M. (Lucy Maud)*
Genre: *Nova Scotia -- History -- 20th century -- Fiction, Interpersonal relations -- Fiction, Shirley, Anne (Fictitious character) -- Fiction, PZ, Self-perception -- Fiction, Orphans -- Fiction, Canada -- History -- 1914-1945 -- Fiction, Universities and colleges -- Fiction, Prince Edward Island -- History -- 20th century -- Fiction*
Gutenberg Id: 51
Gutenberg Link: https://www.gutenberg.org/ebooks/51

8

Book name: *Tess of the d'Urbervilles: A Pure Woman*
Author: *Hardy, Thomas*
Genre: *Didactic fiction, Children of the rich -- Fiction, Wessex (England) -- Fiction, Women household employees -- Fiction, Man-woman relationships -- Fiction, Rape victims -- Fiction, Triangles (Interpersonal relations) -- Fiction, Children of clergy -- Fiction, Women murderers -- Fiction, PR, Poor families -- Fiction, Pastoral fiction*
Gutenberg Id: 110
Gutenberg Link: https://www.gutenberg.org/ebooks/110

Figure 25: The above figure shows the recommendations generated after submitting the subjective questionnaire present in Figure 23 and 24. As the respondent's mood was positive, books with a VAD space having a larger standard deviation are recommended.

For recommending books using the VAD space, a K-Nearest Neighbour Model is used. The recommendations generated in Figure 22 and 25 were using 5 neighbours with *distance* weights and *ball tree* algorithm. The data generated after training the books on models such as Random Forest and Adaboost can be used as training set in the K-Nearest Neighbours approach. The moods predicted by the models can also be used as a threshold to filter out books with high negative emotions for a depressed respondent.

Software Used

Our code base is primarily in Python. We have extensively used Jupyter Notebooks (both internally and via Google Colab). The following softwares are currently being used in this project:

1. Scipy: To get truncated normal continuous random variables and find the pearson correlation.
2. Scikit-Learn: To perform TF-IDF, K-Nearest Neighbour Classification, Radius Neighbor Classification, Random Forest Classification, AdaBoost Classification, Linear Regression, Support Vector Regression, Nu Support Vector Regression, Multilayer Preceptron, pipelines, Principal Component Analysis, Mean Normalization, calculating metrics such as mean squared error, and to divide the dataset into Train-Dev-Test split.
3. NLTK: To perform lemmatization, sentence tokenization, and word tokenization.
4. Gensim: To create doc2vec models.
5. Matplotlib: To plot the various charts both for internal usage and for the purpose of this report.
6. Pytorch: To create the BERT model using DistilBERT.
7. Pandas: The datasets are primarily in CSV format which are then imported and modified entirely in Pandas.
8. Gutenberg, dammit created by Allison Parrish [16] is used to clean the books downloaded from Project Gutenberg.
9. Gutenberg: A python repository used to download books from Project Gutenberg. [17]
10. JEmAS [3]: Although we are not using the original code which is written in Java, we referred to their codebase to create a Python port of JEmAS.
11. Numpy: For Numpy array conversion and concatenation (while calculating doc2vec using Gensim).
12. Flask: For hosting the web framework used in book recommendation.
13. Apart from these, standard libraries such as math, collections, ast, re, json, os, glob, sys, gc, and pickle are used.

References

[1] Mohammad, S. M. (2018a). Obtaining reliable human ratings of valence, arousal, and dominance for 20,000 English words. InACL 2018

[2] Warriner, A.B., Kuperman, V. & Brysbaert, M. Norms of valence, arousal, and dominance for 13,915 English lemmas. Behav Res 45, 1191–1207 (2013). https://doi.org/10.3758/s13428-012-0314-x

[3] Sven Buechel and Udo Hahn: Emotion Analysis as a Regression Problem - Dimensional Models and Their Implications on Emotion Representation and Metrical Evaluation. In: ECAI 2016. 22nd European Conference on Artificial Intelligence. August 29 - September 2, 2016, The Hague, Netherlands, pp. 1114-1122.

[4] Daniel Jurafsky and James H. Martin. 2019. Speech and Language Processing. Draft of October 2, 2019.

[5] Carney, James; Robertson, Cole (2019): 4000 stories with sentiment analysis dataset. Brunel University London. Dataset. https://doi.org/10.17633/rd.brunel.7712540.v1

[6] Sven Buechel and Udo Hahn. 2017. EmoBank: Studying the Impact of Annotation Perspective and Representation Format on Dimensional Emotion Analysis. In EACL 2017 - Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics. Valencia, Spain, April 3-7, 2017. Volume 2, Short Papers, pages 578-585. Available: http://aclweb.org/anthology/E17-2092

[7] Margaret M. Bradley and Peter J. Lang, 'Affective norms for English text (ANET): Affective ratings of text and instruction manual', Technical Report D-1, University of Florida, Gainesville, FL, (2007).

[8] Reagan, A.J., Mitchell, L., Kiley, D. et al. The emotional arcs of stories are dominated by six basic shapes. EPJ Data Sci. 5, 31 (2016). https://doi.org/10.1140/epjds/s13688-016-0093-1

[9] Hugo van Kemenade, Gutenberg-metadata, (2015-2020), GitHub repository, https://github.com/hugovk/gutenberg-metadata

[10] Verma, Gyanendra & Tiwary, Uma Shanker. (2015). Affect Representation and Recognition in 3D Continuous Valence-Arousal-Dominance Space. Multimedia Tools and Applications. 76. 10.1007/s11042-015-3119-y.

[11] Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32 (ICML'14). JMLR.org, II–1188–II–1196.

[12] McNair, James. What Is the Average Reading Speed and the Best Rate of Reading? 3 May 2009, https://ezinearticles.com/?What-is-the-Average-Reading-Speed-and-the-Best-Rate-of-Reading?&id=2298503.

[13] Mayer, J. D., & Gaschke, Y. N. (1988). The experience and meta-experience of mood. Journal of Personality and Social Psychology, 55, 102-111.

[14] von Zerssen D. (1986) Clinical Self-Rating Scales (CSRS) of the Munich Psychiatric Information System (PSYCHIS München). In: Sartorius N., Ban T.A. (eds) Assessment of Depression. Springer, Berlin, Heidelberg
DOI: https://doi.org/10.1007/978-3-642-70486-4_25

[15] 2009. *Models.Doc2vec – Doc2vec Paragraph Embeddings*. [online] Available at: <https://radimrehurek.com/gensim/models/doc2vec.html> [Accessed 5 July 2020].

[16] GitHub. 2020. *Aparrish/Gutenberg-Dammit*. [online] Available at: <https://github.com/aparrish/gutenberg-dammit> [Accessed 5 July 2020].

[17] GitHub. 2020. *c-w/gutenberg*. [online] Available at: <https://github.com/c-w/Gutenberg> [Accessed 5 July 2020].

[18] Park, Sungjoon, et al. "Toward Dimensional Emotion Detection from Categorical Emotion Annotations." *ArXiv:1911.02499 [Cs, Eess]*, Nov. 2019. *arXiv.org*, http://arxiv.org/abs/1911.02499.

[19] Angold, A., Costello, E. J., Messer, S. C., Pickles, A., Winder, F., & Silver, D. (1995) The development of a short questionnaire for use in epidemiological studies of depression in children and adolescents. *International Journal of Methods in Psychiatric Research*, 5, 237 – 249.

[20] Harmon-Jones C, Bastian B, Harmon-Jones E. The Discrete Emotions Questionnaire: A New Tool for Measuring State Self-Reported Emotions. *PLoS One*. 2016;11(8):e0159915. Published 2016 Aug 8. doi:10.1371/journal.pone.0159915