IMDB Movie Reviews Sentiment Analysis

Project Summary:

We created a shiny app to do sentiment classification (Positive / Negative) on IMDB movie review dataset. Classification is done through a neural network model (trained with 8 special features engineered from movie reviews) that performs with an AUC of 91% (10 pts improvement vs. base model). Apart from the classification task the shiny app also provides visualizations that generate insights into the test reviews data.

This report is divided into below sections:

1. Data Source & Description
2. Methodology
    a. Feature Identification / Qualitative Analysis
    b. Text Processing (including Feature Extraction)
    c. Final Feature Selection / Model Inputs
    d. Modelling - Neural Network Model vs. Base Model
3. Results of Neural Network Model vs. Base Model
4. Shiny App Insights
5. Challenges and Avenues for further work.
6. Appendix: Summary of Main Functions

Data Source & Description:

The data is from IMDb website, using web scraping. Dataset has 17474 reviews from IMDB website (https://www.imdb.com). The R script used is 01_IMDb_WebScraping_Script.

There are two steps performed in this script:

1. Scrape IMDb movie id from main IMDb website.

- In this script, date range is from 2017-12-31 to 2019-01-01 with 100 movies per page.

2. Scrape reviews for each movie.

- Use movie ids from 'all_movies.csv', loop over the list and scrape top 25 reviews for each movie.

https://github.com/iesegmbd/SMAGroup3/blob/master/FinalScripts/01_IMDb_WebScraping_Script.R

Methodology:

Qualitative Analysis of Movie Reviews / Identifying Features:

Each group member analyzed 20 randomly chosen reviews to identify qualitative data aspects of our classification problem. Manually classifying this small sample set of review allowed us to customize our approach to better fit the context of text. We shortlisted below 8 features that were frequently observed in our sample set and had a strong connection with overall sentiment of the review.

1. *Rating inside reviews text:* For e.g. '5/10 on acting', 'three out of five etc'.
2. *Elongated Words:* Words that were stretched to emphasize sentiment. For e.g. looooong movie
3. *Exclamation Marks:* Exclamation marks acted as intensifiers of words in the sentence.
4. *All CAPS:* Words in all capital letters were used to intensify emotion, Frequency.
5. Sentences with 'While': Long sentences that went against the overall sentiment of review.
6. *Sentences before 'However', 'But':* Sentiment expressed before these words went against the overall review sentiment.
7. *Sentences after 'despite of, inspite of):* Sentences following these words did not express the true sentiment.
8. *Emoticons:* Reviews contained emoticons that were indicative of overall sentiment.

Text Preprocessing / Feature Extraction:

At the stage of pre-processing data we performed a number of actions through functions specifically created for understanding features identified for reviews dataset. Along with more standard functions required for building a ML model. The below list explains important actions performed along with the functions used.

Features	Actions Performed	Functions Used
Effect of special words	Split reviews into sentences based on punctuation	chunk_into_sentences
	Delete sentences that come before (however,but)	cleanup_function_before
	Delete sentences that come after (despite,inspite of)	cleanup_function_after
	Delete sentences that contain (while)	cleanup_function_this_sentence
Emoticons	Replace emoticons with their text descriptions	func_replace_emoji
Elongated Words	Identify if review contains elongated words	elongated_words2
	Replace elongated words with correct spellings	clean_elongated_words2, correct
	Increase valence of elongated words by multiplying with 2	clean_reviews
Words in all caps	Check if any word is in all caps, return list of capital words.	IsWordCap
	Frequency of capital words.	CountWordsCap
	Return list of words in all capital letters	ListWordsCap
Rating in Review Text	Extract rating given in review text in three forms (1/10, five out of ten, 3/5) converts these to a numeric rating between 1- 10	clean_words, clean_reviews

Exclamations Marks	Check if a review has exclamation marks.	
	Pick the sentence with exclamation marks. Apply POS, lemmatize adjectives and verbs. Lookup valence in dictionary and multiply by 2. Take mean of review valence,	clean_reviews
Other Pre-processing Tasks	convert to lower letters, convert to ascii, remove numbers, punctuation	standard functions
	Vectorize words i.e. create vectors for words to create dtm	text2vec package
	Create Dtm for training and test data sets	tm package
	Create tf.idf for data sets	tm package

Final Feature Selection:

Below are the final set of variables shortlisted from pre-processing stage. These 11 variables were fed into our neural network model.

Model Features / Input Variables	
Variable Name	**Description**
Text	Contains text of movie review
Ratings Scale	Numeric rating between 1 - 10
sentiment	Binary variable. Rating > 5 is considered positive review (1) Rating < 5 is negative (0)
elongated words frequency	Count of elongated words in a review
ratings_word_value.	Numeric. User rating given within review text. Three formats (5/10, five out of ten, 3/5) reduced to a number out of 10
Is_elongated	Binary variable. Value =1 if review contains elongated words
Elongated_sentiment	Valence for elongated words.
is_exclaimed	Binary variable. Value = 1 if review contains exclamation marks
Exclaimed_sentiment.	Valence for adjective/adverb in sentence with exclamation marks
Is_capital	Binary variable. Value = 1 indicates review contains words in capital letters
Capital_freq	Frequency of words in capital letters

Model Building:

Below table summarizes the main steps between our Base Model that we use as a benchmark model and our selected Neural Network Model.

Model Building Synopsis	
Base Model (Benchmark)	Neural Network Model
Standard Pre-processing	Special Pre-processing for feature extraction
Trained on 8736 observations	Trained on 8736 observations
Create dtm, tf.idf	Vectorize words and then create dtm
Load concept loadings from training data to test data	Load concept loadings from training data to test data
No Additional Features	8 additional features extracted from data
Use SVD and Train Random Forest Model	Train Neural Network Model
Test Model Accuracy	Test Model Accuracy

Please see below links for our Base Model and Neural Network model. We have commented the code to explain all the steps taken to build and test models.

Base Model:
https://github.com/iesegmbd/SMAGroup3/blob/master/FinalScripts/02_Baseline_Model.R

Neural Network Model: paste link here.

https://github.com/iesegmbd/SMAGroup3/blob/master/FinalScripts/04_GLMNet_Model.R

Results of Neural Network Model vs. Base Model

Below table presents key metrics for evaluating the two models. Looking at accuracy alone can be misleading therefore we compare F1 score and AUC. There's improvement on all metrics vs. our base benchmark model.

Model Performance Evaluation			
Metrics	Base Model	Neural Network Model	Improvement
Accuracy	0.73	0.84	0.11
Precision	0.74	0.84	0.11
Recall	0.85	0.90	0.05
F1 Score	0.79	0.87	0.08
AUC	0.80	0.91	0.11

Note on Lexicon Based Approach:

We performed lexicon - based sentiment analysis. The results of different packages / lexicons were more or less similar. Positive reviews were accurately predicted ~70% of the time however negative reviews were accurately predicted around ~50%. As a result of these low scores we did not add this feature to our shiny app.

Please see below link for evaluation of nrc (13,901 words), bing (6,788 words) and afinn (2,476 words) lexicons on our movies review dataset.

https://github.com/iesegmbd/SMAGroup3/blob/master/FinalScripts/Lexicon_based.r

Shiny App Insights:

Shiny app provides classification of sentiment into positive / negative. In addition to that it also visualizes review data along different dimensions to provide deeper understanding and possible insights.

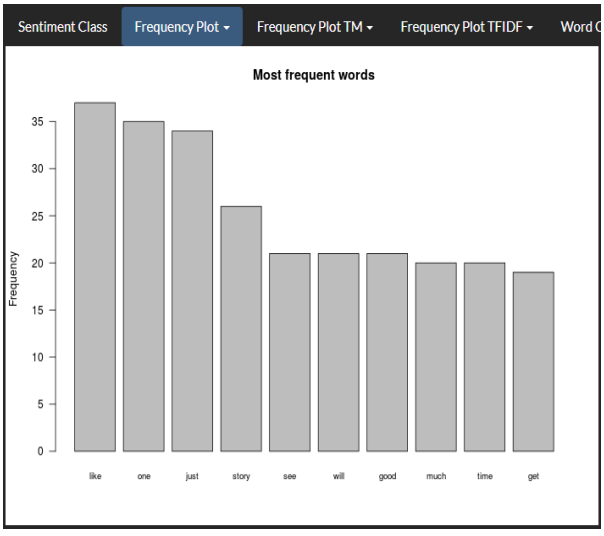
Most plots are based on two documents:

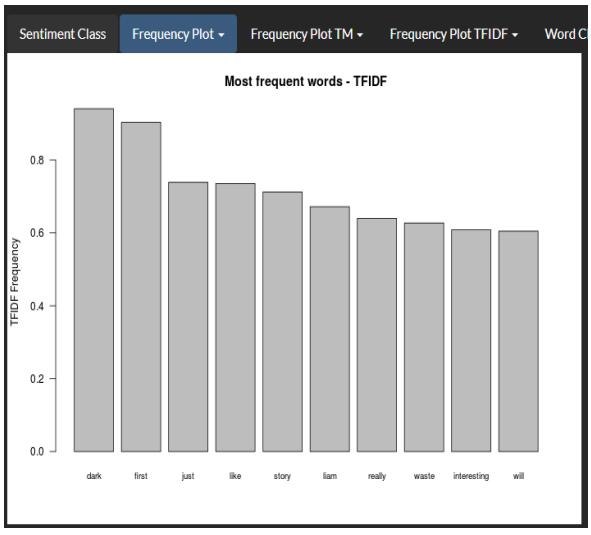
TM – This is our document term matrix. It has frequency of all words used in our document. We can see the top most frequent words in our review. However commonly used terms may not have good explanatory value. To fix this issue we also look at TF*IDF for comparison purposes.

TF*IDF – This measure decreases weight for commonly used words and increases weight for terms used less frequently in the reviews. The less frequent words may have more explanatory value.

Frequency Plots:

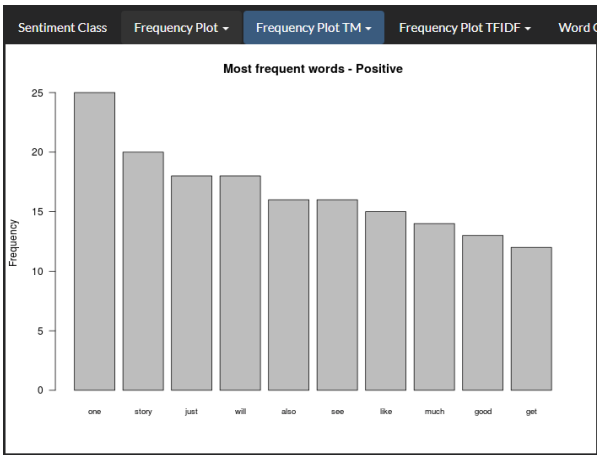
We visualize bar plots of most frequently used words for both TM and TF*IDF. TM frequency plot helps us understand what the most commonly used words are in the documents. TD*IDF plot shows us frequency of top terms adjusted for how rarely it is used.

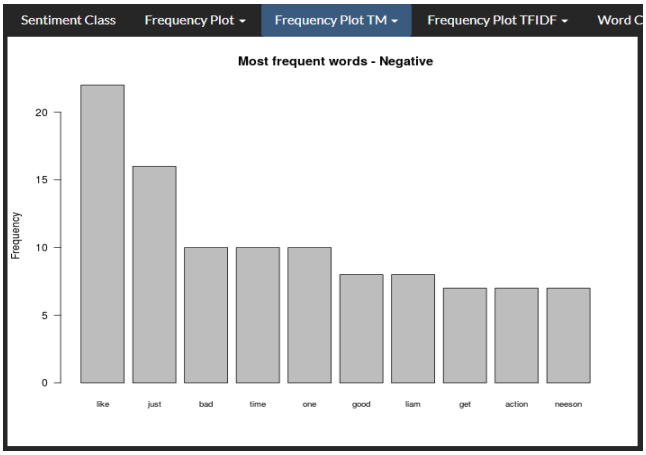




Frequency Plot TM (positive and negative reviews)

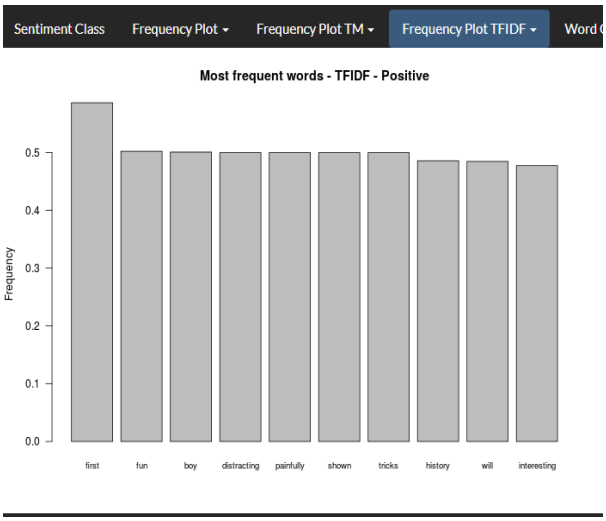
Here we analyze high frequency words separately for positive reviews as well as negative reviews.

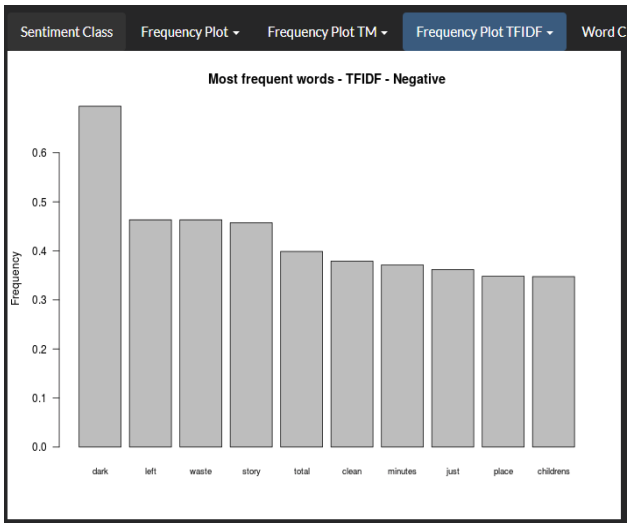




Frequency Plot TM*IDF:

Here we analyze high frequency words for positive reviews as well as negative reviews.





Word Cloud TM:

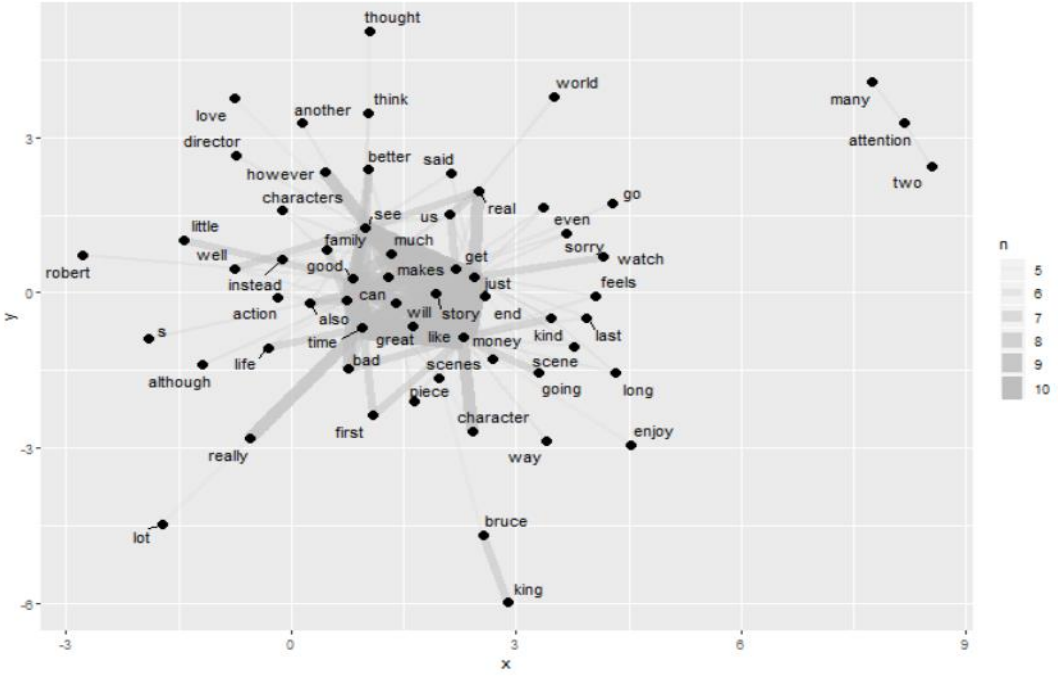
This visualization shows us categories of positive / negative words. The main benefit of this visualization is that it shows a larger number of words in reviews for positive and negative ones. Gives better understanding of what words people use most

Word Cloud TF*IDF:

This visualization shows us categories of positive / negative words. This time we use TF*IDF to get a frequency adjusted terms in our positive and negative reviews. It shows us different set of words that helps us understand what terms reviewers are using.

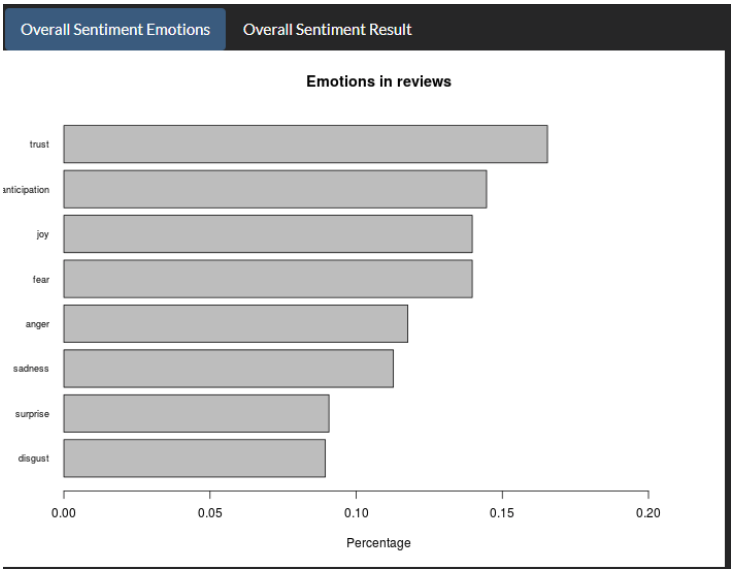
Word Cooccurances Network:

This graph shows us what terms are commonly used together within a review. A thicker line denotes a stronger relationship. This can reveal for example aspect words and sentiment towards a target.



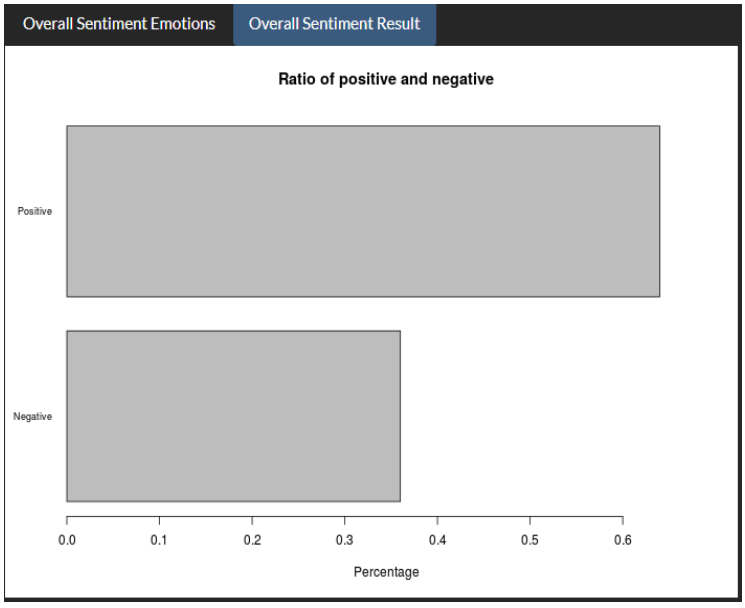
Overall Sentiment Emotions:

This graphs provides wider range of emotions to classify sentiments on. Instead of just a positive / negative assessment, here we can understand what emotions reviewers express in the dataset.



Overall Sentiment Result:

This is a helpful summary graph to show what percentage of total reviews is positive and negative.



Challenges:

Key challenge was understanding if our rules manually observed in sample set were applicable to all reviews. For e.g. always deleting sentence before ‘however, but’ may not apply to all instances. If we had more time, we would have reviewed a larger sample set before defining rule-based approach.

Avenue for Further Work:

At this stage we have only addressed the polarity of the sentiment. If we had more time, we would also try to extract the target of the sentiment and the aspect of the target that the sentiment is expressed about. For e.g. a sentiment could be expressed about a target (Film 1) and aspect of the target (visual effects). This would allow for a richer understanding of the sentiment and its context.

Appendix: Summary of Main Functions

Here we describe the purpose of a function by listing inputs and outputs of a function. This is done to keep the report short and readable. Details on these functions are commented as part of the code submission.

1. chunk_into_sentences
   Input: Takes reviews as input
   Output: divides reviews into sentences. Based on word connected punctuation (?,!,<>, ())

2. cleanup_function_before
   Input: takes sentences
   Output: deletes sentences before 'however, but' and puts sentences back into reviews.

   In our sample review we noticed that sentences that came after 'however' and 'but' described the overall sentiment of the review.

3. cleanup_function_after
   Input: takes sentences
   Output: deletes sentences that are after words like (despite of, in spite of)
   In our sample review we noticed that sentences that came before 'despite of' and 'in spite of' described the overall sentiment of the review.

4. cleanup_function_this_sentence
   Input: review text
   Output: deletes the sentences that contains 'while'
   In our sample review sentences that contained the word 'while' were often long and did not reflect the overall sentiment of the review.

5. func_replace_emoji

   Input: takes reviews text
   Output: replaces emoji's (list of 3000 emoticons) with word descriptions of the emoticon that are then used for sentiment analysis.

6. elongated_words2
   Input: review text
   Output: identifies if review contains elongated words (more than 2 letters occurring together) For e.g. loooong.

7. clean_elongated_words2
   Input: takes elongated words in the review
   Output: fixes elongated words to their original form for e.g, goooooood changes to good.

8. Correct
   Input: takes review text
   Output: compares words with frequently used wordlist and corrects any spelling mistakes it has.

9. IsWordCap
   Input: takes review text
   Output: checks if any word is in capital letters

10. CountWordsCap
    Input: takes reviews text
    Output: Frequency of words in all caps in a review

11. ListWordsCap
    Input: takes review text
    Output: returns list of words in capital letters in the review.

12. clean_words
    Input: may be it is used to
    Output: Feature: Picks up rating given in the review. Changes (5/10, five out of 10, 5 out of 10) to 5 in a column (line 355-415)

13. clean_reviews
    Input: Review data with text column for reviews

    Output: Multiple features added to the data by calling functions defined earlier.
    - elongated_words_freq - Elongated word frequency
    - rating_words_value - Rating of movie written in the review comment
    - elongated_sentiment - Average intensified valence for the elongated words
    - exclamed_sentiment - Average intensified valence for the exclaimed words
    - is_elongated - Boolean (If the review has elongated words)
    - is_exclaimed - Boolean (If the review has exclaimed words)
    - capital_freq - Capital words frequency
    - is_capital - Boolean (If the review has capital words)

14. create_features_from_model
    Input: data- Data frame with reviews, model- TFIDF model used while training, vectorizer- Training vocabulary word vector, lsa- LSA model used while training

    Output: TFIDF, LSA and all the features combined in a matrix which is used for modelling

15. Model_creation
    Input: training data
    Output: List of variables which will be used to transfer information to the testing data tfidf_model- TFIDF model used while training, vectorizer- Training vocabulary word vector, m_lsa- LSA model used while training, train_tfidf- Training TFIDF

16. sentiment_prediction

    Inputs: test- Data frame with reviews under prediction, tr_model- output for model_creation for training data, glmnet_class- Logistic net Model created from training data.
    Output: Classifies sentiment as positive or negative.