Final Report:
Movie Reviews' Sentiment Analysis: Recognizing the Polarity of Reviews Using Different Machine Learning Models and Evaluating Their Quality

Sattam Alammar (218110372), Ibrahim Bazzoun (218110498), Ali Basoodan (218110378), Saleh Al-Mohaimeed (219110413), Serry Sibaee (218110246)

I. Problems

The process of analyzing movie reviews is multilayered and filled with choices, and each route taken will lead to different outcomes. For that, there are paramount questions that precede the experiments of training and testing concerning planning the way and choosing the right options.

Options and Choices

First, one has to choose their training dataset, which is evaluated by of multiple criteria such as the existence of meta-data, its balance between the classes, how clean it is (e.g. null-values are handled, outliers are removed, etc.), the fitness of the source (is it a reflection of how normal movie reviews are?) etc. Our chosen dataset is *IMDB Dataset of 50K Movie Reviews*, which we will be detailed and evaluated later in its own section.

Second, how the dataset is treated in pre-processing. After tokenization, there are many options to normalization, stemming, lemmatization, etc. such as to remove special characters (including emojis or not?), turning the characters into lowercase (eliminating information found within capitalization such as emphasis), etc.

After tokenizing with NLTK library, for our normalization, we chose to remove HTML tags, since the line breaks in the reviews, for example, are represented with their HTML tag
, and as such ought to be removed. We also removed all URLs, dates, and special characters since they are meaningless and/or unworthy to model. We then removed all stopwords as they are very frequently used and with no intrinsic meaning to them. Lastly, we turned all the characters to lowercase to treat all the capitalization variations of a word the same. We also stemmed the words.

Third, languages could be modeled varyingly. One could use e.g. Bag of Words (BoW) or Term frequency–inverse document frequency (TFIDF), and get different results with each. We decided to use both, and with different N-grams. In addition to that, we will use word embedding as Word2Vec.

Forth, there are many Natural Language Processing (NLP) Machine Learning (ML) algorithms, such as Logistic Regression, k-nearest neighbors (KNN), Decision trees, Recurrent neural network (RNN), Naïve Bayes, etc. We chose seven different algorithms: Multinomial Naïve Bayes, Logistic Regression, KNN, RNN, Light Gradient Boosting Machine (LightGBM) Support-Vector Machine (SVM), and our own average document-length model (to predict the sentiment of a review based on its length in comparison to the average length of the reviews in each sentiment/class).

II. Dataset

As previously stated, the dataset of choice is *IMDB Dataset of 50K Movie Reviews*, published by Stanford University [1], which contains highly polarized, binary, balanced, 50,000 sentiment reviews of movies posted on IMDB. The reviews are equally divided into training and testing sets (25k each) and presented as raw text or processed into Bag of Words' (BoW) format.

In judging the dataset, the dataset's size is considered decent, and a good outcome is expected to be reached by such a size. Moreover, its balance between the sentiments (and the sets of training and testing) provides a foundation for unbiased learning. In addition, its preexisting processing of words into BoW format is of help as this model is already planned to be used. Lastly, the dataset contains unlabeled data which, although not planned to be used, gives us greater room of freedom if we so desire to manually intervene in labeling.

Limitations

The dataset could be considered limited in some aspects, most noticeably its lack of meta-data such as of the reviewed movies' names and their genre. If such information was present, our models could lean towards expecting a plausible norm of having, say, family movies being more positive and horror movies more negative, as per nature. However, this limit is easily overcome, and great accuracy can still be achieved without such informative representation and predictive information.

Another limit to this dataset is its binary sentiments, where the reviews are classified as either positive or negative. If the labels were more informative, such as about the

emotion conveyed in the comment (e.g. anger, disgust, disappointment, etc.), and possibly such information was correlated with the movie's name and genre, the language models could have benefited tremendously such that a certain word will no longer be bound to its frequency, for example, to the corpus of positive and negative reviews, but also to more nuanced and clearer sentiments that could more accurately classify the review (especially when a review is quite ambiguous). This limitation, too, is although unfortunate, is indeed possibly overcome.

The limitations in this dataset are all possible to overcome. As such, this dataset will suffice on its own, and no more datasets will be added.

III. Exploratory Data Analysis (EDA)

With our first exploration of the data, we looked at the words in each sentiment. Counting and sorting for the most common words in each sentiment using unigrams yielded similar results for each sentiment with mostly general words such as 'film', 'movie', 'one', 'like', etc. sometimes with different ordering (see: image 1). Positive reviews, however, had the word *love* in the top ten most commonly used words, whilst negative reviews had the word *bad* instead.

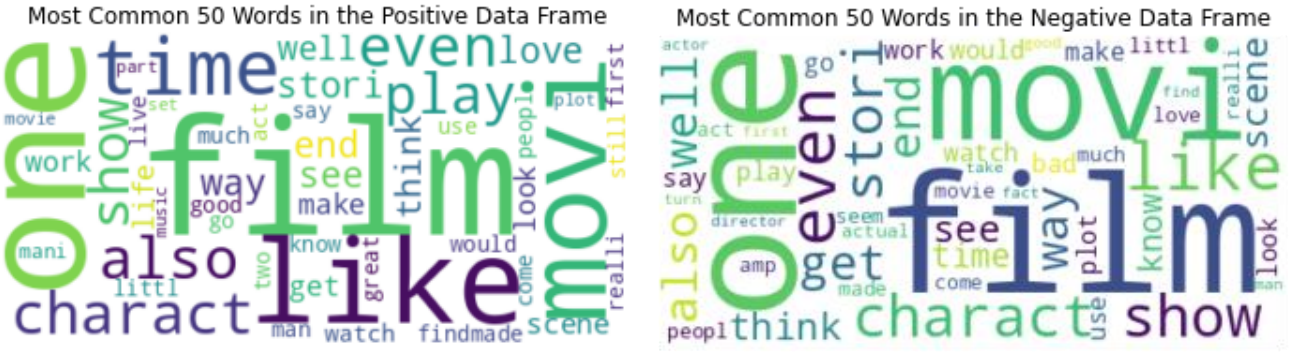

Image 1: most common words for each sentiment using Unigrams

Using Trigrams, however, showed the uniqueness of each sentiment (see: image 2), with very polarized sayings such as ‘worst movie ever’ and ‘best movie ever’. Such sentences presented here might look off, such as ‘one worst movi’ due to the normalization done onto them. This example is originally ‘one of the worst movies’ with the stopwords ‘of’ and ‘the’ being removed, and the pluralism stemmed. The morphology of some words is wrong, such as the word ‘city’ being changed to ‘citi’ as if a form of normalization between it and the plural of ‘cities’. This, however, although working, is not optimal, and another library ought to be used.

Looking into unique words, we found ~65K vs. 62.5K unique words in positive and

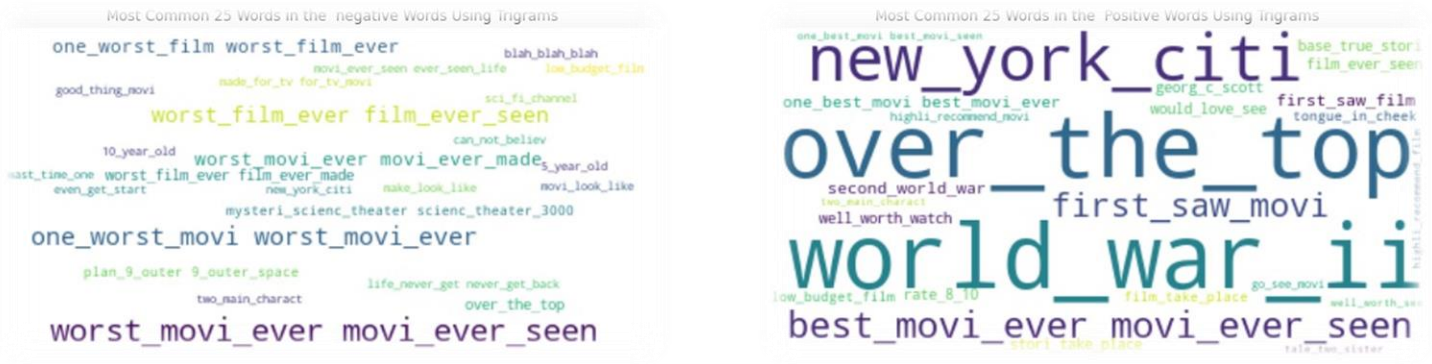

Image ٢: most common words for each sentiment using Trigrams

negative reviews respectively. These include names such as Toyoda (+), possibly misspelled such as Capracorn (-); foreign languages such as *ferrailleur* (+); or simply normal words such as *handhold* (-). Although not very informative, the notion of an existing difference in the number of unique words between the sentiments indicates that positive sentiments are more novel (uncommon details) and variable which could be both linked to personality and emotions, as well as, therefore, used to improve the accuracy of our predictions; if the review is highly novel, it is more probably positive than negative.

Following the previous correlation between novelty and positivity, another hint for a greater hypothesis about a correlation between detailing and expressiveness, and positivity (possibly correlated with Openness and Extroversion personality, for example, which indeed might make people more open and accepting for the unexpected e.g. disappointing movies are okay) is found with the number of letters and words in each sentiment’s reviews. With positive reviews, the number of characters can reach more than 8000 characters, and the number of words more than 1400 (figure 1). With negative reviews, however, the numbers are 5000 and 800

respectively (figure 2). This is a 60% and 75% difference respectively in letters' and words' count.

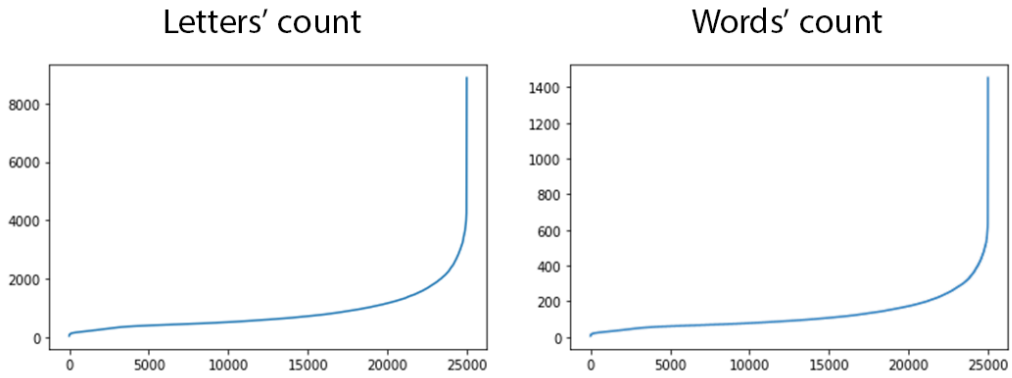

Figure 1: Positive; Y-Axis: count. X-Axis: word index (sorted)

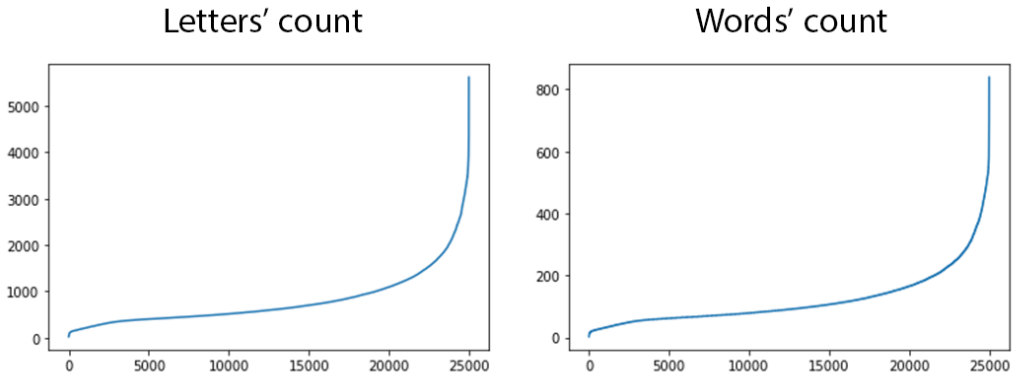

Figure 2: Negative; Y-Axis: count. X-Axis: word index (sorted)

Measuring the density of the words (figure 3), however, shows that those high word and letter count found in positive reviews are considered outliers, with the majority of reviews having similar number of words across the different classes. As such, the correlative hypothesis about novelty and expressiveness, and its relation to positivity is put to rest or at best limited to outliers.

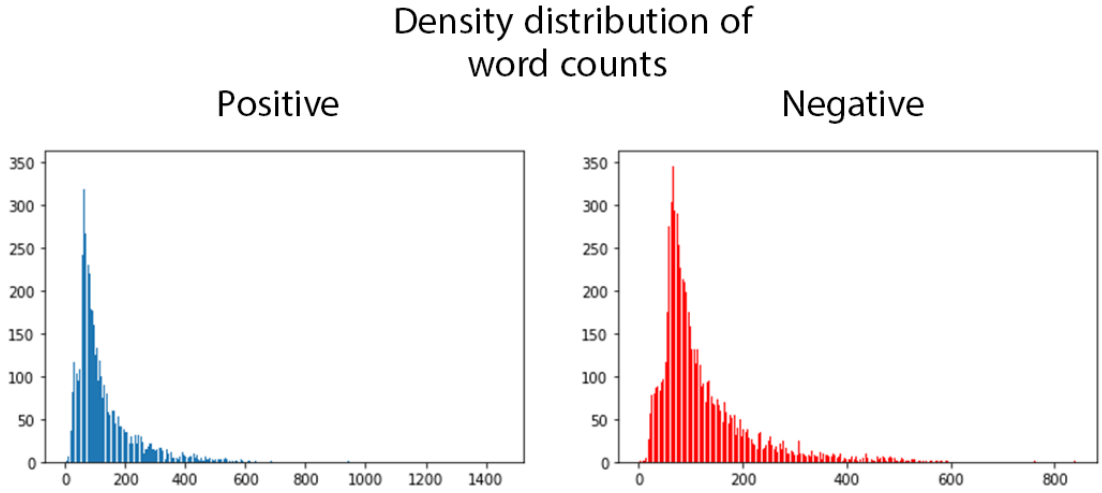

Figure 3: Y-Axis: frequency. X-Axis: word's count

IV. Literature Review

As mentioned above, the number of ways this experiment could be implemented are many, and multiple choices need to be taken. Our plan has already been outlined, but reiterated and explained:

We justify using our chosen dataset because it is indeed a very common and famous dataset that is easy to access and use. Multiple datasets that were found were either restricted, limited, or cluttered and unsuited for our task. Furthermore, the popularity of this dataset means that many similar projects could be found, and as such we can compare our results and algorithms with them. Lastly, it is indeed a well-balanced and rich dataset that, although has its limitations, is good to work with.

Going through the literature, we found many implementations using our dataset. Starting off with the language and libraries used, we used Python as it is the prominent programming language for NLP and ML, and used the libraries of Pandas for data frames, Natural Language Toolkit (NLTK) for general language processing utilities, RE for regular expressions, Scikit-learn for ML algorithms, Seaborn for Confusion Matrix, Matplotib for graphs, and Wordcloud for word presentation. For pre-processing, normalizing by removing stopwords, special characters, and html tags were common, as well as stemming the words (removing affixes to reduce the word to its root) as used in here [2]. A similar approach was taken by us with the addition of removing URLs, and indeed such pre-processing seems sufficient.

In language modeling, many methods such as Bow and TFIDF with different N-Gram were used by the average researchers (see [2]). However, better word embedding algorithms such as Global Vectors (GloVe) were used in the top experiments, such as with this 97% accuracy model [3]. With word embeddings, there are multiple options to implement them such as whether to train them on your own dataset or use a pre-trained model (preferably from a similar corpus. For example, Twitter might be similar in sentiment to IMDB). Also, questions as to how to use them, such as taking the average word embedding to represent the document.

As mentioned, we decided to start off with similar foundations (Bow and TFIDF with multiple N-Grams), whilst also attempting to use a more complex language model of word embedding (Word2Vec). We are going to attempt to train our own model on our own dataset (or corpus), since this will probably provide a more accurate language-representation for us.

For ML models, the choices were endless, from the usual Logistic Regression [2] with accuracy of 75%, to RNN [5] with accuracy of 87%, to Support Vector Machine (SVM) [4] with accuracy of 90%, to BERT [6] with accuracy of 93% (not solely due to the ML algorithm, but also for their specific feature selection methods). As previously stated, we have chosen 7 models in total: Multinomial Naïve Bayes, Logistic Regression, KNN, RNN, SVM, LightGBM, and our own average review-length model. We have chosen Multinomial Naïve Bayes as it is a very simple ML commonly used in NLP that will give us preliminary and presumably limited results. Then, Logistic Regression which is the essence of ML, alongside KNN as a data-driven model are to be used. Similar to KNN, we use SVM as a more powerful algorithm that holds great promise. These three models are prominent in the field of ML, and as such are commonly used in most ML projects. Then, we will use RNN, which is a neural network model, one that is supposed to be within a higher tier of complexity. Although more expensive to run, it might be very efficient. Similarly, we'll use LightGBM as a strong recurrent, loss-function optimizing model too.

As an addition, we decided to make our own model using simple probabilistic thinking. This *average document-length* model simply compares the length of the review with the average reviews' length of each sentiment (77 and 82) with a range (± 12), and classifies the review accordingly. If the review is outside of the specified ranges of both classes (x=a(x)), then a random class is assigned to it. The algorithm as used in our experiment with its averages is provided below:

$$f(x)\begin{cases} x = T\ ,\ x \in [67,79] \\ x = N\ ,\ x \in [80,92] \\ x = a(x) \end{cases}$$

T means Positive review

N means Negative review

$$a(x) = \begin{cases} P(x \mid T) = 0.5 \\ P(x \mid N) = 0.5 \end{cases}$$

This to choose randomly the x in 'a' function

$$\lim_{x \to \infty} f(x) \approx 50$$

Figure 4; Average Document-Length Model

V. Experiments

We first attempted to train our word embedding language model (Word2Vec) as mentioned. This, however, failed, mostly due to processing-resources limitations. As such, only BoW and TFIDF were used. This is unfortunate and is predicted to bring

down the accuracy on many models, since word embedding is a very efficient feature-selection method.

After pre-processing (tokenization, normalization, stemming) the accuracy using BoW and TFIDF models, respectively, per ML algorithm, per N-Gram, using 75% training and 25% testing –are as follows [8-9]:

Unigram:

Language Model / Algorithm	Multinomial Naïve Bayes	Logistic Regression	KNN	RNN	SVM	LightGBM
BoW	86%	88%	64%	81%	86%	85%
TFIDF	86%	89%	78%	NA	89%	85%

Bigram:

Language Model / Algorithm	Multinomial Naïve Bayes	Logistic Regression	KNN	RNN	SVM	LightGBM
BoW	88%	87%	NA	NA	88%	79%
TFIDF	89%	86%	NA	NA	89%	79%

Trigram:

Language Model / Algorithm	Multinomial Naïve Bayes	Logistic Regression	KNN	RNN	SVM	LightGBM
BoW	81%	75%	NA	NA	NA	65%
TFIDF	80%	78%	NA	NA	NA	65%

Figure 5: Table
Accuracy results using all models and algorithms with N-Grams

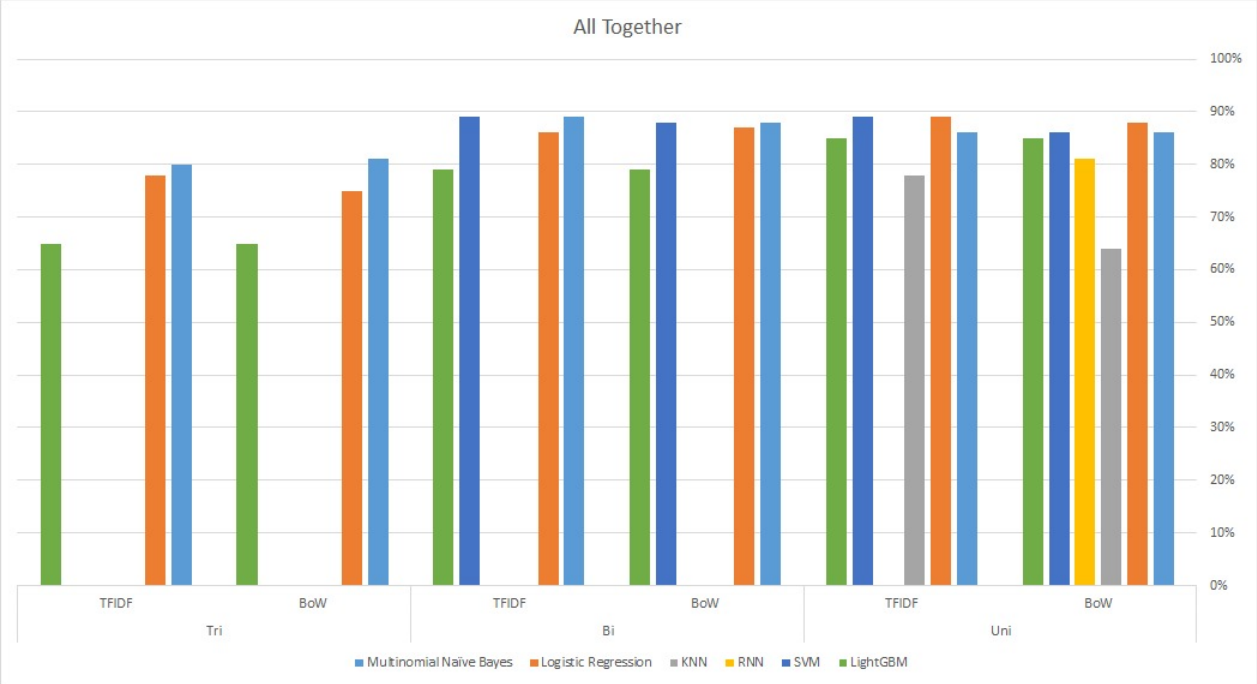

Figure 6: Bar Graph
Accuracy results using all models and algorithms with N-Grams

Our own algorithm we made (average document-length) gave us 50% accuracy [8-9], which means its prediction is a mere chance.

As apparent, some values are missing from the table. This, again, is mostly due to processing-resources limitation. See the full-code on our Deepnote [8].

As seen in the tables, we have achieved up to 89% accuracy, mostly with TFIDF. If we had used mixed N-grams instead of Bigrams and Trigrams alone, the accuracy might have risen more. Also, of course, if word embedding was used, the accuracy might have also been higher.

VI. References

[1] Sentiment Analysis (stanford.edu)

[2] Sentiment Analysis of IMDB Movie Reviews | Kaggle

[3] IMDB sentiment Classifier ~ 97% accuracy model | Kaggle

[4] text classification SVM imdb | Kaggle

[5] 😊 Sentiment Analysis 😠 | Kaggle

[6] Sentiment classification using BERT | Kaggle

[7] Sentiment Analysis for Classifying Sentiment of Movie Reviews (enjoyalgorithms.com)

[8] Project Scratch | Deepnote

[9] NLP project - Colaboratory (google.com)