Sentiment Analysis of Movie Reviews with Machine Learning and Deep Learning Methods

Elias Bier
Department of Computer Science
University of North Carolina
elias.m.bier@gmail.com

Mason William Boyles
Department of Computer Science
University of North Carolina
masonwb@cs.unc.edu

Chua Chong Sun
University of North Carolina
chongsun@unc.edu

Jerry Yang Shao Ern
University of North Carolina
jeryjerr@unc.edu

Abstract

This project investigates the application of Machine Learning (ML) and Deep Learning (DL) techniques for sentiment analysis (SA) on movie reviews. SA aims to classify the sentiment (positive or negative) expressed towards a movie, providing valuable insights for filmmakers into audience opinions and preferences. We experiment with traditional ML models (Random Forest Classifier, Multinomial Naïve Bayes, and Logistic Regression) utilizing CountVectorizer and TF-IDF vectorization, and a modern DL approach involving the fine-tuning of a DistillBERT transformer model. A comprehensive data preprocessing pipeline was implemented, and the class imbalance in the dataset was addressed using random oversampling. Our results, particularly for binary classification, demonstrate that the DL approach using DistillBERT achieves the highest performance (93.0% accuracy), outperforming the ML models (up to 90.6% accuracy) and offering a computationally efficient model suitable for applications with resource constraints.

1 Introduction

Sentiment analysis (SA) of text data is a critical task in natural language processing (NLP), which involves determining the emotional tone or opinion expressed within a piece of text. In the context of movie reviews, SA helps to categorize the expressed sentiment as either positive or negative. This is essential for providing valuable insights into audience opinions and preferences, helping filmmakers and studios understand current trends and interests of moviegoers, and thus make informed decisions about what movies to create.

In this study, we experiment with both conventional Machine Learning (ML) algorithms and advanced Deep Learning (DL) techniques, specifically transformer models, to conduct sentiment analysis on a dataset of movie reviews. Our work compares these two paradigms, evaluates their performance under different classification objectives (3, and 2 classes), and explores the impact of various feature extraction methods.

2 Dataset and Preprocessing

Data preprocessing was performed to reduce noise and variability, allowing models to focus on meaningful information. The preprocessing steps varied between ML and DL approaches.

2.1 Preprocessing for Machine Learning

ML methods require extensive preprocessing: (1) cleaning (removing HTML tags, URLs, emojis, punctuation), (2) text normalization (lowercasing), (3) contraction expansion, (4) tokenization for embedding algorithms (TF-IDF, BoW), (5) stopword removal, and (6) lemmatization to reduce words to base forms.

2.2 Preprocessing for Deep Learning

The DL approach requires minimal preprocessing since DistillBERT leverages stopwords and punctuation for context. We applied: (1) basic cleaning (HTML tags, URLs), and (2) DistillBERT's Wordpiece tokenizer to convert text into token IDs with special tokens, padding, and truncation for uniform sequence lengths.

2.3 Exploratory Data Analysis (EDA)

Various EDAs were conducted to analyze the positive and negative sentiment datasets. A boxplot was utilized to identify and remove outliers in terms of word length, which helps reduce noise and enhance the signal-to-noise ratio.

N-gram Analysis Analysis of the top 10 positive and negative trigrams (Figure 1) revealed that both datasets share common trigrams, such as "have ever seen" and "movie have ever". This may be because reviewers compare movies, which may be problematic since these trigrams do not differentiate sentiment. However, other trigrams included sentiment-indicative words like "worst" or "bad" in negative reviews, and "great" or "best" in good reviews. The trigram "New York City" was among the top positive trigrams but does not contain sentiment information, representing noise that could cause the model to learn spurious patterns.

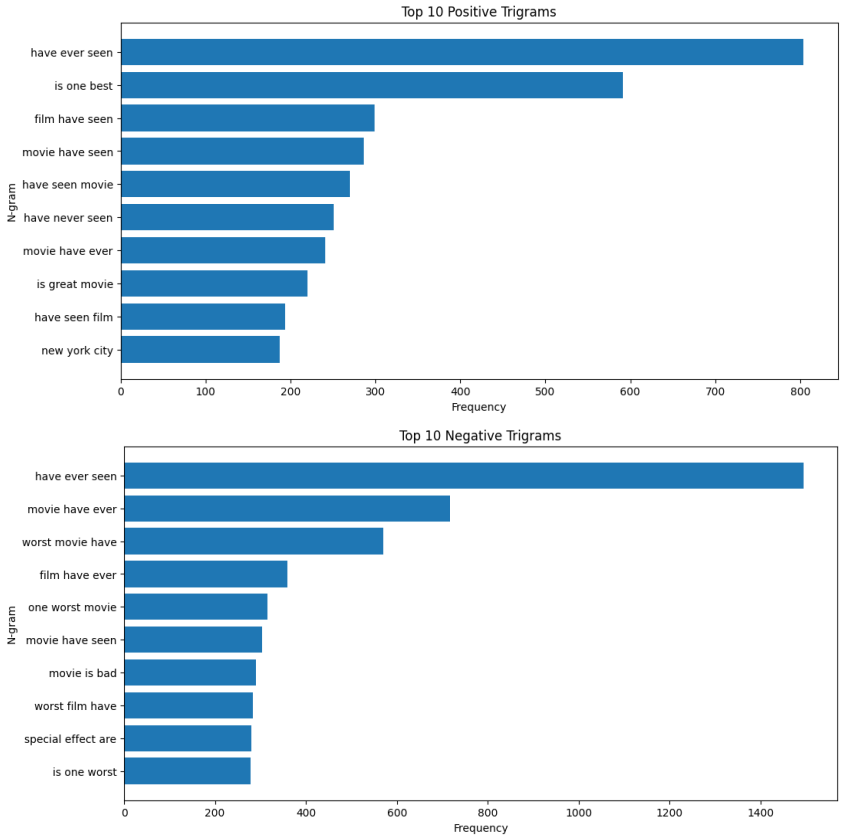

Figure 1: Top 10 Positive (top) and Negative (bottom) Trigrams.

Class Imbalance Countplots of the ratings (Figure 2) showed that viewers are more likely to give extreme ratings (1 or 10). This causes the rating classes to be imbalanced in the dataset, which can lead to the training of biased models.To improve on the limitation of class imbalance random oversampling was conducted. This works by replicating randomly selected instances of the minority classes, giving the model a more balanced dataset for training.

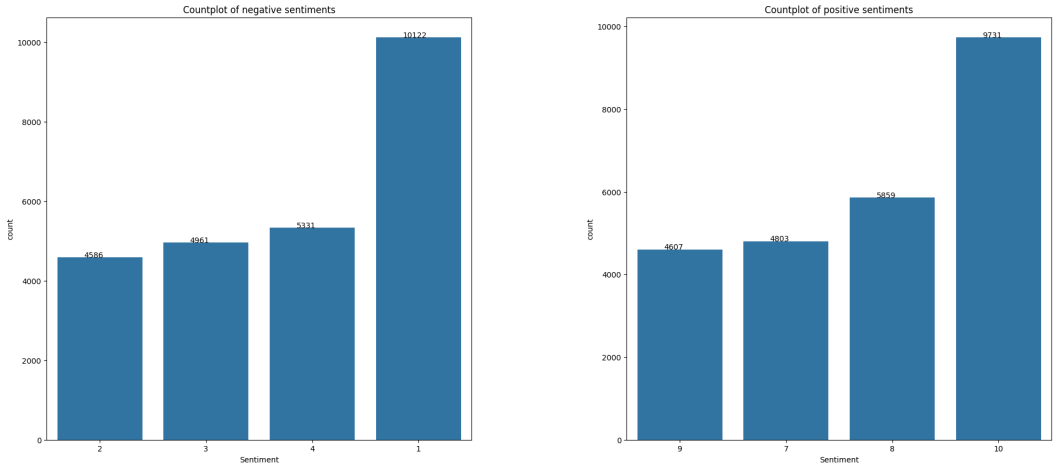

Figure 2: Countplots of ratings: Negative (left) and Positive (right).

3 Methods

3.1 Feature Vectorization for ML Models

Before utilizing the ML algorithms, Count Vectorizer (to create the BoW representation) and TF-IDF vectorizer were used to transform the data into embeddings.

1. Count Vectorizer: Transforms the text data into matrices representing the frequency of each word.
2. TF-IDF Vectorizer: Transforms a collection of text documents into numerical feature vectors, where each feature represents the importance of a term within a document relative to a collection of documents.

$$\text{TF-IDF} = \text{TF} \times \text{IDF}$$

Term Frequency (TF) is the number of times a word appears in the document divided by the total number of words in that document. Inverse Document Frequency (IDF) calculates the word's importance across all documents:

$$\text{IDF} = \log\left(\frac{\text{no. of documents}}{\text{no. of documents containing the word}}\right)$$

The TF-IDF vectorizer used $ngram\_range = (1, 3)$, $min\_df = 2$, and $max\_df = 0.85$. These settings ensure the inclusion of unigrams to trigrams (capturing context), filtering of rare n-grams (reducing noise), and removal of overly common terms.

3.2 Machine Learning Models

We tested the following models:

1. Random Forest Classifier: An ensemble learning method that constructs multiple decision trees during training and outputs the mode of the individual trees to reduce overfitting and increase accuracy. Bagging is used to train each tree on a bootstrapped subset of the dataset.
2. Multinomial Naïve Bayes (MNB): A classification algorithm based on Bayes' theorem that assumes features are independent. It is ideal for discrete frequency count features, like word counts.
3. Logistic Regression: A common baseline model for binary classification tasks.

3.3 Deep Learning Model (DistillBERT)

We fine-tuned DistillBERT, a distilled version of BERT that retains 97% of BERT's performance while being 60% faster and 40% smaller. BERT's contextual embeddings leverage bidirectional attention, masked language modeling, and next-sentence prediction to capture semantic nuances better than frequency-based methods. DistillBERT's efficiency enabled training for more epochs compared to BERT-base.

The architecture consists of DistillBERT's encoder (768-dimensional embeddings), a fully connected layer with ReLU activation, Dropout ($p = 0.1$), and a 2-node output layer. We used AdamW optimization with Cross Entropy loss. Bayesian Optimization yielded optimal hyperparameters: learning rate 1e-5 and dropout $p = 0.1$.

4 Results and Discussion

Model performance was benchmarked on the test dataset using Accuracy, Precision, Recall, and F1-score.

4.1 Classification Performance Summary

Table 1: Best performance achieved for 3 Classes (Good/Average/Poor Sentiments).

Method	Accuracy	Precision	Recall	F1-score
Multinomial Naive Bayes (with BoW)	73.3%	82.0%	61.0%	54.0%

Table 2: Classifying into 2 Classes (Good/Bad Sentiments).

Method	Accuracy	Precision	Recall	F1-score
Random Forest Classifier (with TF-IDF)	86.3%	87.0%	86.0%	86.0%
Multinomial Naive Bayes (with TF-IDF)	89.2%	89.0%	89.0%	89.0%
Logistic Regression (with TF-IDF)	90.6%	91.0%	91.0%	91.0%
DistillBERT with finetuning	**93.0%**	**94.0%**	**93.0%**	**93.0%**

4.2 Key Observations

- Vectorization Method: The models using the TF-IDF vectorizer worked better than those using the Count vectorizer. TF-IDF emphasizes words that are important to a document but not overly common in the corpus, providing better insights.
- DL vs. ML Performance: Text embeddings from DistillBERT outperformed frequency-based embedding approaches (TF-IDF and Count Vectorizer) since these embeddings also consider the nuances and semantic meaning of words, extracting more informative features.
- Proposed Model: Our DL approach outperformed the ML benchmarks, achieving performance comparable to related studies for ML methods (85-91%). Our proposed best model is the fine-tuned DistillBERT, as it is less computationally expensive and provides good performance (93.0% accuracy), making it suitable for applications with limited computation resources for training and inference.

References

[1] Rahman A, Hossen MdS (2019) Sentiment analysis on movie review data using Machine Learning Approach. *2019 International Conference on Bangla Speech and Language Processing (ICBSLP)*. doi: 10.1109/icbslp47725.2019.201470.

[2] Sahu, T. P., & Ahuja, S. (2016). Sentiment analysis of movie reviews: A study on feature selection & classification algorithms. *2016 International Conference on Microelectronics, Computing and Communications (MicroCom)*. doi:10.1109/microcom.2016.7522583.

[3] Yoo, J.-Y., & Yang, D. (2015). Classification scheme of unstructured text document using TF-IDF and naive bayes classifier. *Advanced Science and Technology Letters*. https://doi.org/10.14257/astl.2015.111.50.

[4] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 6000–6010.

[5] Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018). BERT: Pre-Training of Deep Bidirectional Transformers for Language Understanding. https://doi.org/10.48550/arXiv.1810.04805.

[6] Sun, C., Qiu, X., Xu, Y., & Huang, X. (2019). How to Fine-Tune BERT for Text Classification? *Computation and Language (Cs.CL)*. https://doi.org/10.48550/arXiv.1905.05583.

[7] Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

[8] Loshchilov, I., & Hutter, F. (2019). Decoupled Weight Decay Regularization. *Machine Learning (Cs.LG)*. https://doi.org/https://doi.org/10.48550/arXiv.1711.05101.

[9] Maneja, D. (2021). How and Why TF-IDF Works. *Medium*. https://towardsdatascience.com/how-tf-idf-works-3dbf35e568f0.