IT1244 Project Report: Sentiment Analysis of Movie Reviews

Team 18: Chia Kai En Erika, Tan Qing Zhe, Tan Shyan Q'yan Kate

Introduction

With the exponential growth in media consumption driven by the digital age, there has been a significant increase in user-generated movie reviews. Analyzing these reviews' sentiments can offer valuable insights into public opinion, viewer engagement, marketing effectiveness, and content reception, thus guiding firms strategically (Duan et al., 2008). This project aims to perform sentiment analysis (SA) on a dataset of 50,000 movie reviews, classifying them as either positive or negative using various Machine Learning (ML) and Deep Learning (DL) methods.

Recent studies show that classical machine learning models can effectively classify sentiment. Ghosh (2022) used TF-IDF with SVM, Logistic Regression, and Naïve Bayes, where SVM best performed. Similarly, Steinke et al. (2022) applied SVM, Decision Trees, and Random Forests to IMDb reviews, with SVM again performing best. Reddy et al. (2021) achieved the best results using TF-IDF with Logistic Regression and GridSearchCV tuning. Wongkar and Angdresey (2019) explored Naïve Bayes in SA and did well. However, these methods lacked visual interpretability during Exploratory Data Analysis (EDA). To address this gap, we supplement n-gram features with word clouds to offer richer, sentiment-driven insights (Kabir, Ahmed, and Karim, 2020). Additionally, to improve scalability on larger datasets, we explore LinearSVC with Stochastic Gradient Descent as efficient alternatives to traditional SVM.

DL Approaches are exceptionally suitable in analyzing complex language patterns for SA due to their automated feature extraction, word embeddings, and scalability to large datasets (Zhang, Wang and Liu, 2018). A study by Solanki et al. (2024) employed traditional DL Models such as CNN, LSTM and proposed a hybrid of the two models. This improved upon classical ML by effectively combining local and sequential feature extraction. However, this hybrid model struggled to capture deeper contextual relationships and lacked transfer learning capabilities. To overcome these constraints, we were inspired by the paper 'Attention is All You Need' and explored Bidirectional Encoder Representations from Transformers (BERT), a transformer-based model that uses a self-attention mechanism. BERT's transfer learning capabilities allow it to capture deeper contextual relationships, making it a powerful and scalable solution for sentiment analysis (Vaswani et al., 2017).

We also explored VADER sentiment scoring to classify movie reviews and adjusting the threshold values to improve accuracy. Despite efforts to optimize these thresholds, the overall accuracy remained suboptimal, likely because VADER is tailored for short, informal language of social media rather than longer, more complex sentences typical of movie reviews (Youvan, 2024). and is known to be more computationally efficient as compared to ML models like SVM (Hutto and Gilbert, 2014).

Dataset

Upon inspection, we found that many reviews contained HTML tags, punctuation, and digits. As they do not contribute meaningfully to sentiment analysis, they were removed to reduce noise and prevent overfitting. Additionally, to ensure that the same words were analyzed similarly, all text was converted to lowercase. Then, the cleaned reviews were saved into a CSV file to reduce the number of Input/Output (I/O) operations during processing.

Next, we performed tokenization, which involved splitting each review into individual tokens. These tokens were then lemmatized (converted to their base forms) so that different inflections of the same word would be analyzed as a single feature.

Additionally, to gain a visual understanding of the textual data, we created separate word clouds for positive and negative reviews to guide our analysis (see Appendix A).

During preprocessing, we removed standardized stop words using the Natural Language Toolkit (NLTK) while first retaining key negation words (e.g., "not" and "no") to preserve important sentiment cues. We then extracted unigrams, bigrams, and trigrams from our positive and negative datasets and compared their top ten frequencies. N-grams with relative frequency differences within a 0.1 threshold were considered sentiment-neutral and removed. Although negation words are typically crucial, our frequency analysis revealed that the unigram "not" and the trigrams ("ive", "ever", "seen") occurred with similar prevalence across classes and were subsequently excluded (see Appendix B).

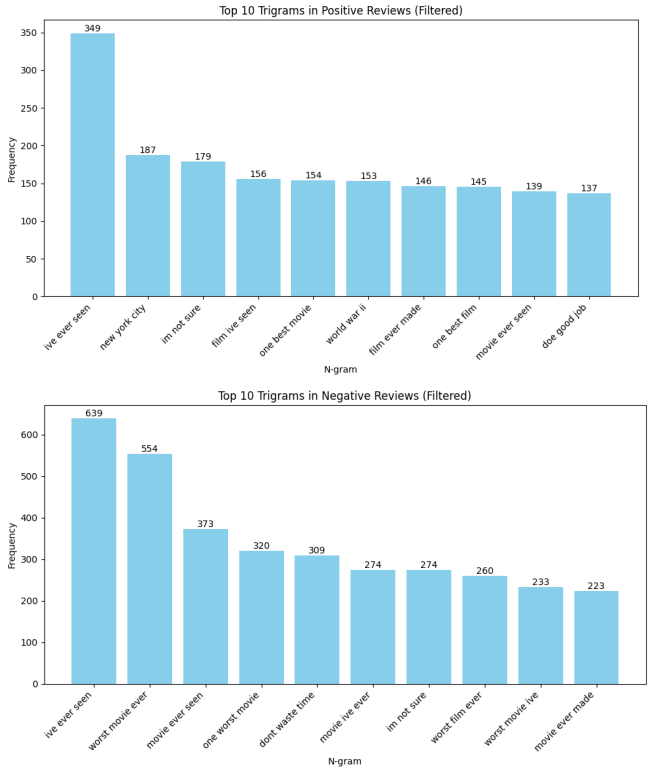

Figure 1: Trigrams after Common Stop Words Removal.

We then saved both the original (regular stop words removed) and the modified (extra stop words removed) versions of the dataset and experimented with various ML models on each to determine which preprocessing approach yielded superior performance.

Methods

This task was framed as a binary classification problem, predicting sentiment (positive or negative) from text input. A range of methods were tested based on the reviewed literature, from ML to DL models and, and lexicon-based approaches. Each model followed a consistent pipeline of preprocessing, feature extraction, training, and evaluation for fair comparison.

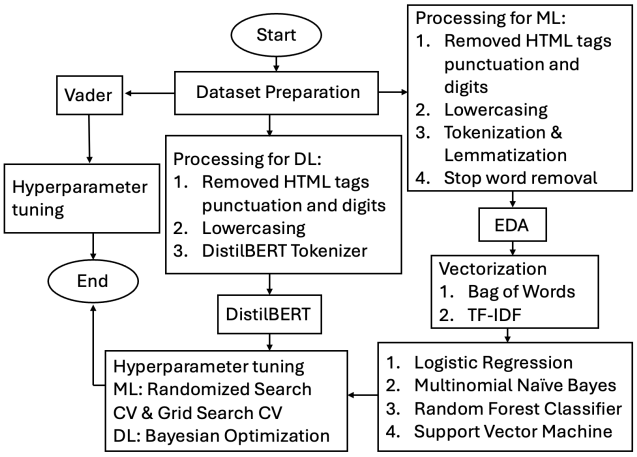

Figure 2: Model Pipeline

To assess the performance of each model, cross-validation along with metrics such as accuracy, precision, recall, and F1-score was utilized. This ensured robust evaluation across data splits and highlighted trade-offs between false positives and false negatives. Ultimately, we chose to focus on accuracy because our dataset exhibited a balanced class distribution, making overall correct predictions a reliable indicator of model performance.

Text Vectorization

Raw, cleaned text data is still not suitable for machine learning algorithms, so we converted the text into numerical representations (vectors) using:

- Bag of Words (BoW): Represents reviews by word frequency, resulting in sparse vectors that ignore word order.
- Term Frequency–Inverse Document Frequency (TF-IDF): Enhances BoW by down-weighting common terms, emphasizing informative words but still sparse in nature.

These high-dimensional sparse feature vectors were then passed to classifiers. The sparsity can impact both computational efficiency and model performance, making

algorithms that handle sparse data well (like Logistic Regression or LinearSVC) more suitable for this task.

Machine Learning Models

We trained and evaluated four main different machine learning models:

1. Naive Bayes: Probabilistic model assuming feature independence. It is effective and fast for text due to reliance on word distributions.
2. Logistic Regression: Linear classifier with logistic function. It performs well on sparse and large data.
3. Random Forest Classifier: An ensemble model of decision trees which captures nonlinear relationships and is robust to overfitting, though less efficient on sparse data.
4. Support Vector Machines (SVMs): Supervised Learning Models that utilise critical data points ("support vectors") to determine the position of the decision boundary of classes. We experimented with three variations, namely:
    a. Linear Kernel SVM: A margin-based classifier that finds the optimal hyperplane to separate classes. The linear kernel is particularly effective for sparse, high-dimensional text features but slower on larger datasets.
    b. LinearSVC: A scalable, optimized implementation of a linear SVM. It uses hinge loss and is faster and more memory-efficient than traditional SVMs, making it suitable for large text datasets. It was the best performing variation in terms of evaluation metrics out of the three variations experimented.
    c. Stochastic Gradient Descent (SGD) Classifier: It trains a linear model in mini-batches using SGD. However, the stochastic nature could affect model accuracy. This variation performed best in terms of time efficiency.

To improve model performance, we employed a two-step hyperparameter tuning strategy:

1. RandomizedSearchCV: Broadly explore the hyperparameter space and identify promising regions for further analysis.
2. GridSearchCV: Systematically evaluate specific parameter combinations within these regions to fine-tune hyperparameters based on a relevant hyperparameter grid, using 5-fold cross-validation to evaluate performance. (see Appendix C).

Naive Bayes, Logistic Regression, Random Forest, LinearSVC, and SGDClassifier were fine-tuned to generalize better on unseen data. Due to computational limits, the linear-kernel SVM variant was excluded from fine-tuning.

Deep Learning Model (distilBERT)

For DL, we drew inspiration from the usage of Bidirectional Encoder Representations from Transformers (BERT) in modern-day applications.

BERT is a deep learning model pre-trained on large-scale text using a bidirectional self-attention mechanism. Unlike traditional sequential models like LSTMs and RNNs that process text in-order and struggle with long-range dependencies, BERT captures context from both preceding and following words simultaneously, resulting in nuanced sentiment understanding. Furthermore, its transformer-based architecture allows for parallel processing and faster fine-tuning, making it more efficient and accurate than LSTMs for SA.

Given the resource constraints of our task, we opted for DistilBERT, a smaller, distilled version of BERT. It harnesses knowledge distillation, where a smaller model mimics the output of the larger BERT model. While it retains much of BERT's contextual understanding, DistilBERT is more computationally efficient, though it may result in slight accuracy loss compared to BERT on complex tasks. Still, it maintains much of BERT's performance while being more lightweight, making DistilBERT ideal for real-time SA in resource-constrained environments (Sanh et al., 2019).

During experimentation, we considered two approaches for handling text files longer than 512 tokens: truncation and chunking (see Appendix E). In our tests, truncation performed better, likely because the initial segments of the reviews contained the most coherent and relevant

sentiment cues. Next, we experimented with freezing the pre-trained DistilBERT layers and training only a two-layer classification head to reduce overfitting, inspired by Srivastava et al. (2014). However, full fine-tuning of the entire model ultimately yielded superior performance.

To further optimize the model, we applied Bayesian Optimization (see appendix F) to determine the best dropout and learning rates. After selecting the optimal dropout and learning rate, we fine-tuned the batch size and extended training from 2 to 4 epochs. The best model performance was achieved with 2 epochs, a dropout rate of 0.1, a learning rate of 0.00005, and a batch size of 4.

Valence Aware Dictionary and sEntiment Reasoner (VADER)

VADER is a lexicon and rule-based SA tool that assigns sentiment scores to words and phrases using a predefined lexicon combined with heuristic rules that capture punctuation, capitalization, emojis, and slang. It computes a compound score ranging from -1 to 1 to represent the overall sentiment of a text as positive, negative, or neutral. VADER is specifically designed to handle the nuances of informal text, differentiating itself from ML or DL models.

We applied VADER to the movie review dataset by obtaining the compound score for each review and assigning a predicted sentiment label based on VADER's default thresholds: a score above 0.05 is considered positive, below -0.05 is negative, and scores in between are neutral. We then evaluated the accuracy of these predicted classes against the ground truth labels in the dataset before experimenting with fine-tuning the threshold values to improve the accuracy of the model.

Results and Discussion

We evaluated the performance of our models using accuracy, precision, recall and F1 score.

$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$	$Precision = \frac{TP}{TP + FP}$
$Recall = \frac{TP}{TP + FN}$	$F1 = 2 \times \frac{Recall \times Precision}{Recall + Precision}$

Figure 3: Trigrams after Common Stop Words Removal.

Based on our testing, TF-IDF consistently outperformed the BoW approach. The table below presents the performance of all our models using the TF-IDF vectorizer. For completeness, the BoW results are included in the appendix (see Appendix D).

Method	Accuracy	Precision	Recall	F1
Naive Bayes	86.4%	87.0%	85.6%	86.3%
Random Forest Classifier	86.1%	85.1%	87.4%	86.3%
Logistic Regression	89.8%	88.7%	91.1%	89.9%
Linear SVC	83.3%	83.0%	83.8%	83.4%
SGD Classifier	82.3%	82.0%	82.9%	82.4%
DistilBERT	93.4%	93.1%	93.7%	93.4%
VADER	71.2%	67.7%	81.3%	73.8%

Figure 4: Performance of all models (with TF-IDF vectorization for ML models) after hyper-parameter tuning.

The best-performing ML and DL models were Logistic Regression (89.8% accuracy) and DistilBERT (93.4%) respectively. We believe Logistic Regression performed well with TF-IDF features due to its ability to handle high-dimensional, sparse data and exploit strong word-sentiment associations.

Additionally, we found that TF-IDF outperformed BoW by not only counting word occurrences but also weighting them by their importance across the corpus, allowing the model to focus on more informative terms. In contrast, BoW treats all words equally, which can dilute discriminative features and introduce noise.

Overall, DistilBERT outperformed all models by generating deep, context-aware embeddings that capture semantic meaning and long-range dependencies, crucial for understanding the subtleties of movie reviews. Unlike frequency-based methods, it draws on knowledge from large-scale pretraining to grasp context and word variations. VADER underperformed, as it is designed for short, social media-style texts and struggles with the complexity of full-length reviews which was consistent with findings by Hutto and Gilbert (2014).

References

[1] Duan, W., Gu, B., and Whinston, A. B. 2008. Impact of User-Generated and Professional Critics Reviews on Bollywood Movie Success. ResearchGate. https://www.researchgate.net/publication/274406708_Impact_of_User-Generated_and_Professional_Critics_Reviews_on_Bollywood_Movie_Success

[2] Ghosh, A. 2022. Sentiment Analysis of IMDb Movie Reviews: A Comparative Study on Performance of Hyperparameter-Tuned Classification Algorithms. In Proceedings of the 2022 8th International Conference on Advanced Computing and Communication Systems (ICACCS), 289–294. IEEE. https://doi.org/10.1109/ICACCS54159.2022.9784961

[3] Kabir, A. I., Ahmed, K., and Karim, R. 2020. Word Cloud and Sentiment Analysis of Amazon Earphones Reviews with R Programming Language. Informatica Economică 24(4): 55–71. https://doi.org/10.24818/issn14531305/24.4.2020.05

[4] Hutto, C. J., and Gilbert, E. 2014. VADER: A Parsimonious Rule-Based Model for Sentiment Analysis of Social Media Text. In Proceedings of the Eighth International AAAI Conference on Weblogs and Social Media (ICWSM-14), 216–225. http://eegilbert.org/papers/icwsm14.vader.hutto.pdf

[5] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., Polosukhin, I. 2017. Attention is All You Need. In Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS 2017), 6000–6010. https://arxiv.org/abs/1706.03762

[6] Solanki, S., and Trivedi, K. 2024. Sentiment Analysis of Movie Review Using Deep Learning Techniques. Preprint, May 2024. https://doi.org/10.21203/rs.3.rs-4388226/v1

[7] Sanh, V., Debut, L., Chaumond, J., and Wolf, T. 2019. DistilBERT, a Distilled Version of BERT: Smaller, Faster, Cheaper and Lighter. In Proceedings of the 5th Workshop on Energy Efficient Machine Learning and Cognitive Computing (EMC²) at NeurIPS 2019.

[8] Srivastava, N., Hinton, G., 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. https://dl.acm.org/doi/pdf/10.5555/2627435.2670313

[9] Steinke, I., Wier, J., Simon, L., and Seetan, R. 2022. Sentiment Analysis of Online Movie Reviews Using Machine Learning. International Journal of Advanced Computer Science and Applications (IJACSA) 13(9): 618–624. https://www.ijacsa.thesai.org

[10] Zhang, L., Wang, S., and Liu, B. 2018. Deep Learning for Sentiment Analysis: A Survey. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 8(4): e1253. https://doi.org/10.1002/widm.1253

[11] Reddy, P. S., Sri, D. R., Cheerka, S. R., and Shaik, S. 2021. Sentimental Analysis Using Logistic Regression. International Journal of Engineering Research and Applications 11(7): 36–40. https://doi.org/10.9790/9622-1107023640

[12] Wongkar, M., and Angdresey, A. 2019. Sentiment Analysis Using Naive Bayes Algorithm of the Data Crawler: Twitter. In Proceedings of the 2019 Fourth International Conference on Informatics and Computing (ICIC), 1–5. https://doi.org/10.1109/ICIC47613.2019.8985884

[13] Youvan, D. C. 2024. Understanding Sentiment Analysis with VADER: A Comprehensive Overview and Application. Preprint, June 2024. https://doi.org/10.13140/RG.2.2.33567.98726

Appendix

A. Word clouds for positive & negative sentiments

Figure 5: Word clouds for positive & negative sentiments.

B. Frequency Analysis for Identifying Neutral N-grams

In our analysis, we extracted unigrams, bigrams, and trigrams from both the positive and negative datasets and computed the relative frequency of each n-gram by dividing its count by the total number of n-grams in the respective dataset. We then compared the top ten most frequent n-grams between the two classes by calculating the ratio of their relative frequencies. N-grams whose ratios between positive and negative reviews were within the range of 0.9 and 1.1 were considered sentiment-neutral and removed from further analysis. This ensured that only those n-grams which provided meaningful sentiments were retained. While negation words are typically important for

sentiment analysis, our findings indicated that the unigram "not" and the trigram ("ive", "ever", "seen") exhibited similar frequencies in both classes, and were thus excluded to reduce noise and enhance model focus.

C. Hyperparameter Tuning

Due to the computational cost associated with tuning an extensive number of hyperparameters, we chose to focus on those with the most significant impact on model performance. For Naive Bayes, we tuned the smoothing parameter alpha, which plays a crucial role in avoiding zero probabilities in text classification by managing the trade-off between overfitting and underfitting, especially for rare words. In Logistic Regression, we explored parameters such as C (the inverse regularization strength that controls the trade-off between fitting the training data and maintaining a simpler model), penalty (choosing between l1, which encourages sparsity by driving some coefficients to zero, and l2, which uniformly penalizes large coefficients to prevent overfitting), solver (using the liblinear algorithm for efficiency and compatibility with l1 penalty), and max_iter (to ensure sufficient iterations for convergence). For the Random Forest Classifier, we concentrated on n_estimators (the number of trees, where more trees generally reduce variance but require more computation), max_depth (to limit tree depth and prevent overfitting), min_samples_split and min_samples_leaf (to control how the tree splits and the size of terminal nodes), and max_features (to balance the number of features considered for each split). With LinearSVC, we tuned the regularization parameter C (referred to in the pipeline as linearsvc__C), which balances margin maximization with error minimization for better generalization. Lastly, for the SGD classifier with a linear SVM, we focused on alpha (the regularization term multiplier) and penalty (which determines the type of regularization—l1, l2, or elastic net).

By strategically narrowing the search space to these key hyperparameters, we efficiently optimized model complexity and generalization while maintaining computational feasibility, ultimately improving overall performance.

D. Performance of all models using the BoW vectorizer

Method	Accuracy	Precision	Recall	F1
Naive Bayes	85.6%	89.9%	80.3%	84.8%
Random Forest Classifier	86.7%	85.9%	87.8%	86.8%
Logistic Regression	88.7%	87.7%	90.2%	88.9%
Linear SVC	83.9%	83.9%	83.9%	83.9%
SGD Classifier	83.8%	83.1%	85.0%	84.0%

Figure 6: Performance of all models (with BoW vectorization for ML models) after hyper-parameter tuning.

E. Truncation and Chunking for handling long text.

To handle texts exceeding 512 tokens, we investigated two strategies: truncation and chunking. Truncation involves simply cutting the text at the 512-token mark, thereby preserving only the initial segment, which is often the most sentimentally relevant portion. However, this approach may discard later information that could be valuable for sentiment analysis. In contrast, chunking divides long texts into multiple segments, each containing up to 512 tokens. To maintain context between these segments, an overlap of 50 tokens is introduced between consecutive chunks. This overlapping ensures that important contextual cues that span segment boundaries are not lost, allowing the model to capture a more comprehensive representation of the text.

F. Bayesian Optimization for Hyperparameter Tuning.

We applied Bayesian optimization to fine-tune DistilBERT by targeting two key hyperparameters: learning rate and dropout factor. This method uses a surrogate model—typically a Gaussian process—to efficiently approximate the objective function, which is ideal given the black-box nature of BERT's internal computations. The learning rate controls convergence behavior, while dropout acts as a regularizer to prevent overfitting. Our search space spanned learning rate exponents from 3 to 7 and dropout values between 0.1 and 0.6. After 10 initial random evaluations, we conducted 5 Bayesian optimization iterations, which led to improved model performance on sentiment classification.