P15: Sentiment Classification of IMDb Movie Reviews

CS485 – Project Report

Zervos Spiridon Chrisovalantis (csd4878)
Drakakis Rafail (csd5310)

June 29, 2025

Abstract

We present an end-to-end pipeline for sentiment classification of the IMDb Movie Reviews dataset using both classical and deep learning techniques. We preprocess the data, extract features, train multiple models, and evaluate their performance. Classical ML models include Logistic Regression, Naive Bayes, and Linear SVM trained on TF–IDF vectors. Deep learning models include LSTM and 1D-CNN built using PyTorch with learned embeddings. We analyze accuracy, confusion matrices, and inference time. Our results show strong performance from classical methods, competitive CNN results, and poor LSTM learning without pretrained embeddings or tuning.

Contents

1 Introduction

Sentiment analysis seeks to classify text into categories such as positive or negative sentiment. The IMDb dataset offers a widely used benchmark of 50,000 labeled movie reviews. We compare two families of sentiment classification models:

- **Classical ML**: Feature engineering with TF–IDF followed by Logistic Regression, Naive Bayes, and Linear SVM.
- **Deep Learning**: LSTM and CNN architectures with learned word embeddings and end-to-end training in PyTorch.

2 Related Work

Early approaches to movie-review sentiment analysis relied on surface-level representations such as bag-of-words and handcrafted lexicons. Pang and Lee demonstrated that simple unigram and bigram features fed into classical classifiers (e.g., Naïve Bayes, SVM) already achieved strong baselines on the IMDb dataset [1]. Lexicon-based methods such as VADER and SentiWordNet leverage word-level sentiment scores, offering interpretability but often struggling with context and negation.

The advent of distributed word embeddings (e.g., Word2Vec, GloVe) enabled neural models to capture semantic similarity. Kim popularized one-layer CNNs over word embeddings for sentence classification, showing that shallow convolutional filters can extract n-gram features effectively [3]. Subsequent work introduced recurrent architectures—particularly LSTMs and GRUs—to model sequential dependencies and long-range context [4].

More recent studies incorporate hierarchical and attention-based mechanisms. Yang et al. proposed a Hierarchical Attention Network that applies attention at both word and sentence levels, improving document-level sentiment classification [5]. The Transformer architecture introduced self-attention, laying the groundwork for large pretrained language models [6], and fine-tuning BERT and RoBERTa has since set new state-of-the-art results on IMDb and related benchmarks [7, 8].

Finally, ensemble and hybrid methods—combining neural models with lexicon features or classical classifiers—have been shown to further boost robustness, particularly on noisy and out-of-domain data, motivating our exploration of contextual embeddings, attention layers, and ensemble strategies.

3 Methodology

3.1 Dataset

- **Name:** IMDb Large Movie Review Dataset
- **Size:** 25,000 training and 25,000 test samples
- **Labels:** Binary sentiment – Positive (1), Negative (0)
- **Source:** `https://ai.stanford.edu/~amaas/data/sentiment/`

3.2 Preprocessing

- Dataset is downloaded and extracted from Google Drive.
- Text normalization: lowercasing, tokenization (using NLTK `word_tokenize`), removal of punctuation, numbers, and stopwords.
- For deep models: vocabulary built with a minimum frequency of 2 and capped at 20,000 tokens. Input sequences are padded or truncated to a maximum length of 200.

3.3 Model Architectures & Training Details

3.3.1 Classical Models

Logistic Regression: ℓ_2 regularization, $C = 1.0$, max_iter = 1000

Multinomial Naive Bayes: $\alpha = 1.0$

Linear SVM: $C = 1.0$

3.3.2 Deep Learning Models

- **Embedding Layer:** Embedding dimension = 100
- **LSTM:** Single-layer LSTM with 128 hidden units; classifier uses final hidden state
- **CNN:** 1D convolutions with filter sizes [3, 4, 5], 100 filters each; followed by max pooling and a fully connected layer
- **Training:** Optimized with Adam, learning rate = $1e^{-3}$, 5 epochs, batch size = 64

4 Implementation

4.1 Preprocessing & Vocabulary

- NLTK downloads handled programmatically for required packages: `punkt`, `punkt_tab`, `stopwords`.
- Texts are lowercased, tokenized (using `word_tokenize`), filtered for alphabetic words, and cleaned of English stopwords.
- For deep models: a vocabulary is built from training tokens with `min_freq=2`, `max_size=20000`. Sequences are padded or truncated to a fixed length of 200.

4.2 Feature Extraction

- **Classical ML:** TF–IDF vectorization (unigrams only), limited to the top 5000 features.
- **Deep Learning:** Each word is mapped to a trainable embedding of dimension 100.

4.3 Prediction Script

A separate standalone script, `predict.py`, is provided for inference on new review texts using the trained CNN model.

- **Architecture:** Re-implements the exact CNN Classifier architecture used during training.
- **Workflow:**
  1. Loads the vocabulary from `vocab.pkl`.
  2. Loads the trained model weights from `model.pt`.
  3. Preprocesses the input review with the same tokenization and stopword removal.
  4. Converts the token sequence to indices and pads to length 200.
  5. Performs inference using PyTorch (CPU or CUDA).
  6. Outputs the predicted label (positive or negative) and the confidence score.

5 Results & Evaluation

5.1 Classical Machine Learning

Table 1: Accuracy and Inference Time (ms/sample) on 25,000 test samples

Model	**Accuracy**	**Time (ms/sample)**
Logistic Regression	0.880	0.47
Multinomial Naïve Bayes	0.840	0.13
Linear SVM	0.863	0.65

Table 2: Classification Report for Logistic Regression

Class	**Precision**	**Recall**	**F1-score**	**Support**
Negative	0.88	0.88	0.88	12500
Positive	0.88	0.88	0.88	12500
Overall accuracy: 0.88				

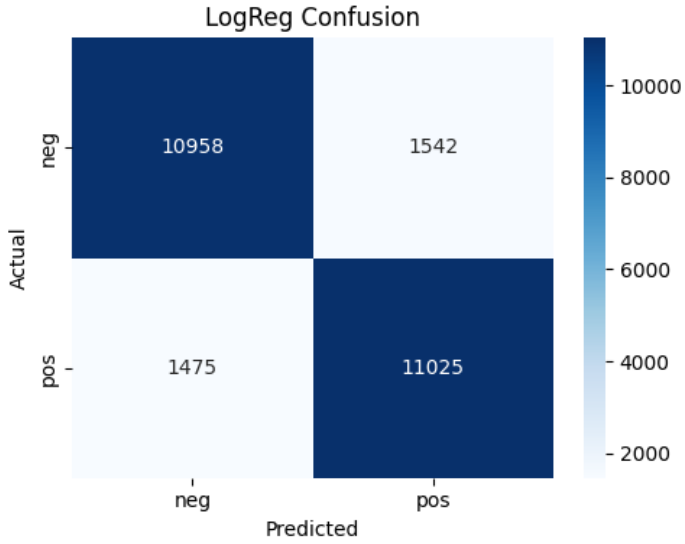

Figure 1: Confusion matrix for Logistic Regression

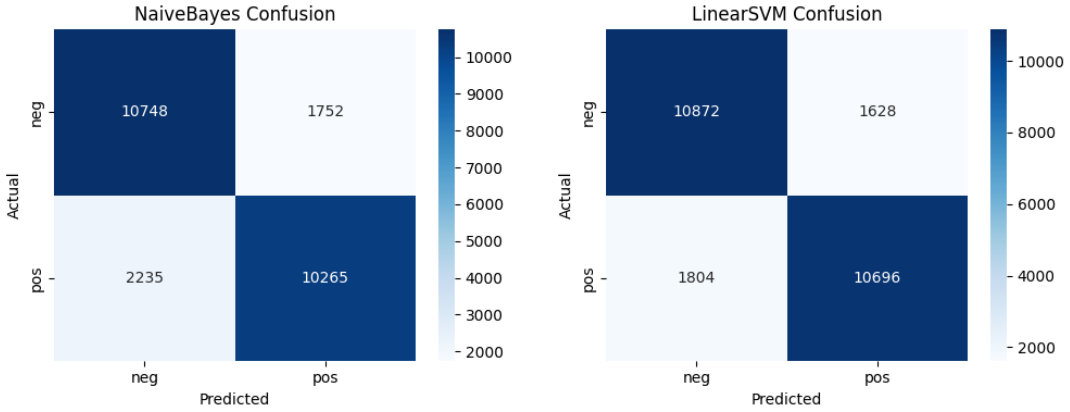

Figure 2: Confusion matrices: Naïve Bayes (left), Linear SVM (right)

5.2 Deep Learning

Table 3: Test Accuracy for Deep Models

Model	Accuracy	Notes
LSTM	0.511	Underfitting, poor learning across epochs
CNN	0.856	Competitive with classical models

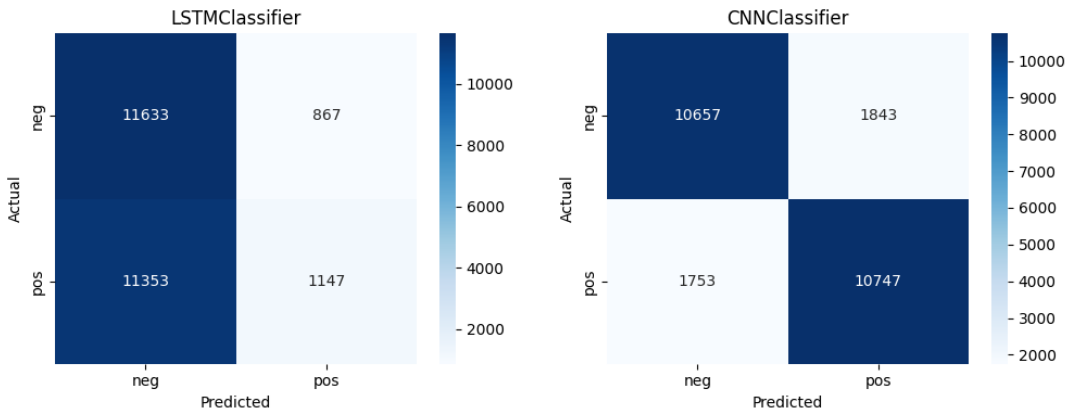

Figure 3: Confusion matrices: LSTM (left), CNN (right)

6 Discussion

- **Inference speed:** Naïve Bayes is fastest (0.13ms/sample), followed by Logistic Regression (0.47ms), and Linear SVM (0.65ms).
- **Accuracy:** Logistic Regression is best overall (0.88), CNN comes close (0.856), and LSTM performs poorly (0.511).
- **Deep Learning Issues:**
  - LSTM struggles due to lack of pretraining, insufficient data augmentation, and shallow architecture.
  - CNN benefits from local n-gram pattern detection via convolution and max pooling.
- **Generalization:** CNN is more robust than LSTM but needs GPU for fast training.
- **Short reviews:** Posed challenges for all models, especially with sarcasm or implicit sentiment.

7 Conclusion & Future Work

- **Conclusion:** We implemented a full classification pipeline using classical and deep learning models for sentiment analysis on the IMDb dataset. Classical methods with TF–IDF and Logistic Regression remain strong baselines for accuracy and speed. While the CNN shows promising results, the LSTM architecture underperformed due to training instability and lack of optimization. Further improvements may include hyperparameter tuning, pretrained embeddings, and attention mechanisms.
- **Future Work:** Future work could leverage pretrained transformers (e.g., BERT or RoBERTa) fine-tuned on your IMDb data and employ advanced augmentation (like back-translation) for greater robustness, alongside systematic hyperparameter search and regularization to stabilize training. Incorporating attention layers and ensembling neural and classical models can sharpen focus on key sentiment cues and exploit complementary strengths. Detailed error analysis with interpretability

tools (e.g., LIME, SHAP) will uncover failure modes, while model compression (quantization, pruning) readies the pipeline for CPU-constrained deployment.

References

[1] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up? Sentiment Classification using Machine Learning Techniques. In Proceedings of EMNLP 2002, pages 79–86. Association for Computational Linguistics. https://aclanthology.org/W02-1011/

[2] C. J. Hutto and E. Gilbert. 2014. VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text. In Proceedings of the Eighth International Conference on Weblogs and Social Media (ICWSM-14). https://ojs.aaai.org/index.php/ICWSM/article/view/14550

[3] Yoon Kim. 2014. Convolutional Neural Networks for Sentence Classification. In EMNLP 2014. Association for Computational Linguistics. https://aclanthology.org/D14-1181/

[4] Duyu Tang, Bing Qin, and Ting Liu. 2015. Document Modeling with Gated Recurrent Neural Network for Sentiment Classification. Association for Computational Linguistics. https://aclanthology.org/D15-1167/

[5] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical Attention Networks for Document Classification. Association for Computational Linguistics. https://aclanthology.org/N16-1174/

[6] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. (NeurIPS 2017). https://papers.neurips.cc/paper/7181-attention-is-all-you-need.pdf

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. Association for Computational Linguistics. https://aclanthology.org/N19-1423/

[8] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. https://arxiv.org/abs/1907.11692