CS412 Machine Learning – Spring 2025

Sentiment Analysis on IMDB Reviews

Group Number: 15

Rivar Kaya — Student ID: 28839

Hüsnü Hantal — Student ID: 29132

Emir Algan Mere — Student ID: 30655

Ahmet Musa Ersoy — Student ID: 30646

Egemen Yağız Akyüz — Student ID: 29156

Collab link: https://drive.google.com/file/d/1FwDkQoG-OjSbUQS9TLnMXwpz4bALTfrc/view?usp=sharing

1. Introduction

This work addresses binary sentiment categorization utilizing the IMDB movie review data. The aim is to develop computer models able to automatically determine if a movie review expresses a positive or negative attitude. Widely investigated in natural language processing, this issue has pragmatic importance for uses including automated customer feedback analysis, recommendation systems, and review aggregating platforms.

We used three different models, each reflecting a different paradigm in text classification, to approach this work from several angles and assessed them. Constructed a TF-IDF (Term Frequency–Inverse Document Frequency) vectorized features, the first model is a logistic regression classifier. While capturing the significance of individual words across the corpus, this provides a solid classic machine learning baseline with quick training and great interpretability.

The second model makes advantage of an LSTM (long short-term memory) network. Essential for comprehending the subtle tone of longer and more complex reviews, this architecture is able to capture sequential dependencies and contextual relationships across word sequences by embedding the tokenized review texts into dense vector spaces and processing them through an LSTM layer.

Finally, we investigated a CNN (convolutional neural network) design. Although usually connected with image processing, CNNs can be rather efficient in text categorization by recognizing local patterns, such suggestive word phrases or sentiment-bearing n-grams. To extract important information from every review, our CNN model uses embedding layers then 1D convolution and global max pooling.

We seek to comprehend the strengths and trade-offs of classical and deep learning approaches in sentiment analysis by evaluating and comparing these three models, thereby determining which methods perform best within the given restrictions of model complexity and data representation.

Experimental Design

To guarantee consistency and fair comparison, all models underwent training and evaluation under the identical train-validation-test split. Several macro-averaged measures—accuracy, precision, recall, F1-score, AUC—were used to gauge performance. Macro F1-score was the main assessment tool utilized in accordance with the project since it offers a fair picture of model performance over both sentiment categories.

This consistent framework allowed us to directly compare traditional and deep learning models and analyze their relative effectiveness in sentiment classification.

2. Problem Description

This work uses IMDB dataset textual movie reviews to handle a binary sentiment classification problem. The aim is to create models that, depending just on their content, can precisely indicate whether a review conveys a positive or negative viewpoint.

Natural language reviews provide the input; the result is a binary sentiment label. We guarantee fair evaluation by using a balanced dataset of 50,000 reviews, equally split between positive and negative emotions, therefore removing class imbalance.

Among the various difficulties this work presents are different review lengths, linguistic complexity, and the necessity to translate unstructured material into numerical attributes. These properties make it perfect for evaluating deep learning methods against conventional machine learning models.

3. Methods

3.1 Dataset Analysis

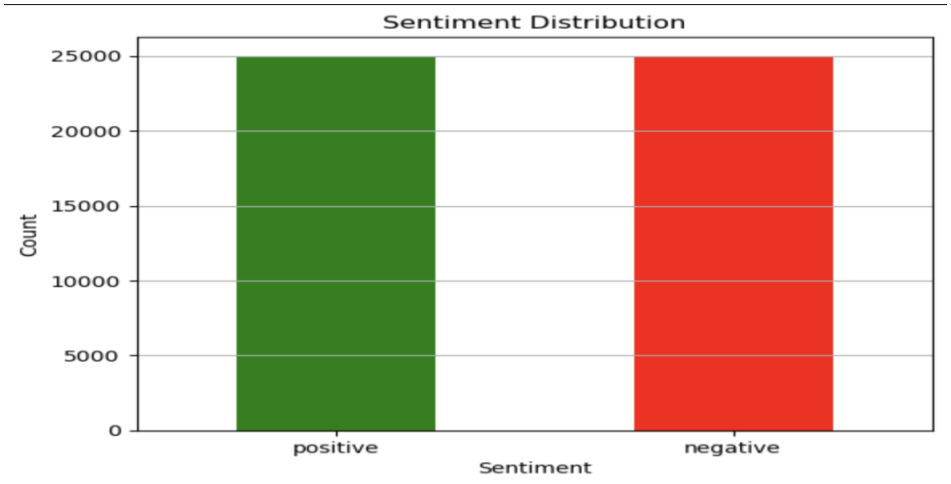
For this project, we used the IMDB movie review dataset, a widely recognized benchmark in the field of sentiment analysis. Its scope and structure make it especially fit for assessing and contrasting several machine learning methods.

Comprising 50,000 annotated movie reviews split equally between favorable (25,000) and negative (25,000), the dataset This exactly balanced class distribution guarantees that performance measures including precision, recall, and F1-score may be understood free from regard for bias resulting from class imbalance and eliminates the need for extra resampling methods. Each data point in the dataset includes two columns:

- A text field containing the full raw movie review, written in natural language, often ranging from a few sentences to multiple paragraphs.
- A label field indicating the sentiment as either "positive" or "negative".

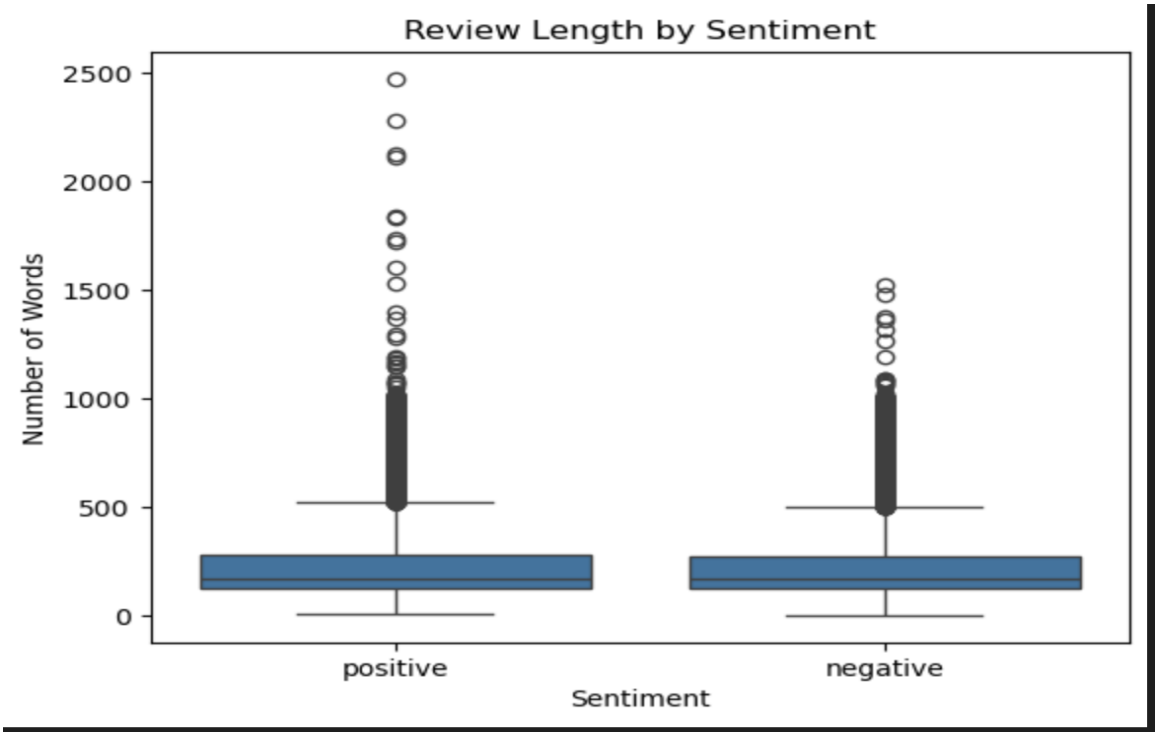
This structure allows for both traditional feature extraction (e.g., TF-IDF) and deep learning approaches (e.g., sequence models using word embeddings). Additionally, the variety in review length and language complexity provides a robust foundation for testing models' ability to generalize and handle nuanced sentiment expressions.

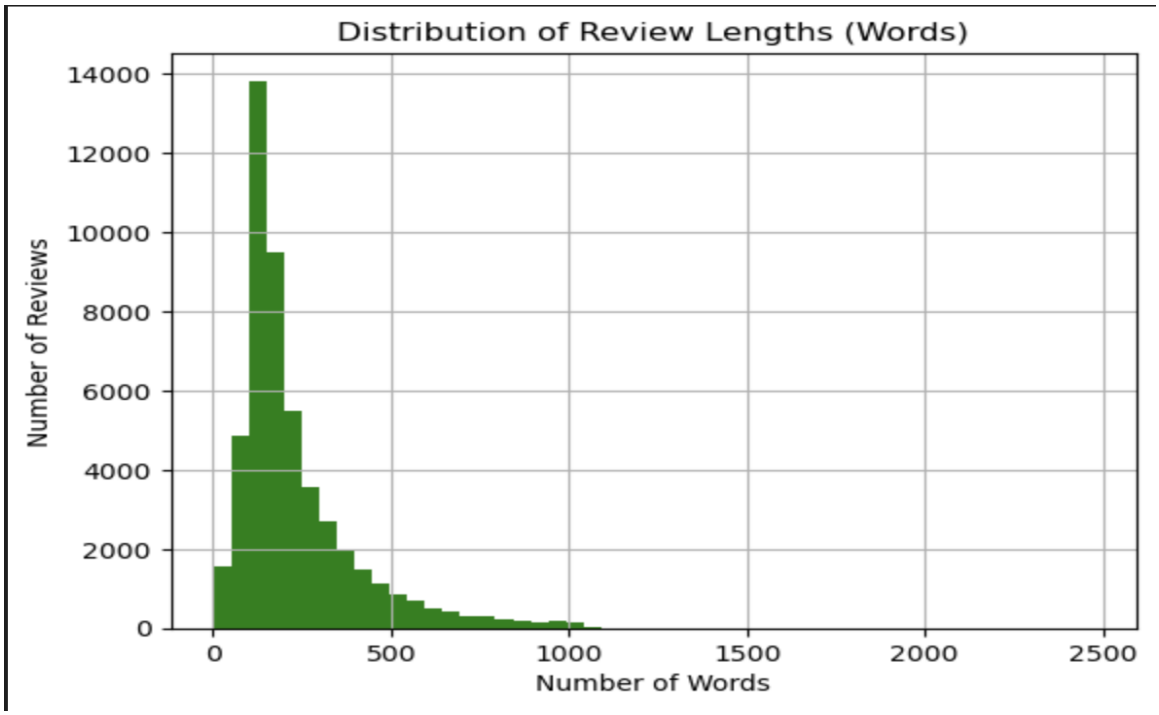
The dataset's balance and diversity make it an ideal testbed for a comparative analysis of classical machine learning and modern deep learning techniques in binary sentiment classification.



Statistical Characteristics: Our exploratory analysis revealed key properties that guided preprocessing and modeling decisions:

- **Class Balance:** Perfectly balanced (50% positive, 50% negative)
- **Review Length Variability:**
    - Average characters: ~1,300 per review
    - Average word count: ~230 words per review
    - Length range: Some reviews exceed 2,000 words, requiring models to handle significant input variation





3.2 Text Preprocessing Pipeline

To prepare the raw textual data for machine learning models, we implemented a systematic preprocessing pipeline designed to normalize text while preserving semantic meaning:

Sequential Processing Steps:

- **Text Normalization:** Converting all text to lowercase for consistency
- **HTML Tag Removal:** Using regex patterns to strip HTML markup from reviews
- **Character Filtering:** Removing non-alphabetic characters including punctuation and digits
- **Stopword Elimination:** Applying NLTK's English stopword list to reduce noise
- **Whitespace Standardization:** Normalizing spacing and removing extra whitespace

Pipeline Benefits:

- Reduces vocabulary size and noise

- Standardizes text format across all reviews
- Preserves essential semantic content
- Creates consistent input for both traditional and deep learning approaches

The cleaned text output served as input for TF-IDF feature extraction in traditional models or was directly fed into embedding layers for neural network architectures.

3.3 Model Architectures

We implemented three distinct models representing different paradigms in text classification, enabling comprehensive comparison of traditional machine learning versus deep learning approaches:

Model 1: Logistic Regression with TF-IDF

This model serves as a traditional machine learning baseline for binary sentiment classification. It uses TF-IDF vectorization with a logistic regression classifier together with a statistical text representation method. This method is rather popular for its simplicity, interpretability, and efficiency on structured text data.

Methodology

1. Text Preprocessing
The raw review texts are first preprocessed to reduce noise and prepare them for feature extraction. This entails turning all text lowercase, removing HTML elements, filtering non-alphabetic letters, and cutting out common stopwords. These actions allow the language to be more natural and direct the model on significant textual patterns.

2. Feature Extraction with TF-IDF
Once the text is cleaned, each review is transformed into a numerical feature vector using Term Frequency-Inverse Document Frequency (TF-IDF). This technique captures the importance of each word in a given review relative to its frequency in the entire dataset.

- Term Frequency (TF) reflects how frequently a word appears in a specific review.
- Inverse Document Frequency (IDF) down-weights words that appear in many reviews and are therefore less informative.
  The resulting representation is a sparse, high-dimensional vector where each dimension corresponds to a word in the vocabulary.

3. Dataset Splitting

To ensure a fair and consistent evaluation, the dataset is split into training and test sets using an 80/20 ratio. This same split is reused across all models to enable direct comparison.

4. Model Training

A logistic regression classifier is trained on the TF-IDF vectors using scikit-learn's implementation. The model learns to associate weighted features with sentiment classes by optimizing a log-loss function. The training process is configured with a maximum of 1000 iterations to ensure convergence on the high-dimensional data.

5. Model Evaluation

After training, the model is evaluated on the test set using multiple metrics: accuracy, precision, recall, F1-score, and a confusion matrix. The primary metric for comparison across models is the macro-averaged F1-score, which treats both sentiment classes equally and provides a balanced assessment of performance.

This model is designed to establish a strong, interpretable baseline for sentiment classification. It offers several advantages:

- Fast training time, even on large datasets
- Direct interpretability through feature weights
- Competitive performance with minimal computational complexity

By using TF-IDF with logistic regression, this model highlights the effectiveness of traditional methods and provides a reference point for evaluating the improvements offered by more complex deep learning models.

Model 2: LSTM (Long Short-Term Memory)

The second model captures the sequential structure of textual material by use of a Long Short-Term Memory (LSTM) neural network. Particularly suited for applications like sentiment analysis where the meaning of a sentence can depend on word order and contextual relationships, LSTM is a form of recurrent neural network (RNN) especially built to learn long-term dependencies.

Methodology

1. Text Representation

Reviews are tokenized and turned into padded sequences of numbers before training the model. Every integer in the training set indicates a word index determined by frequency. These sequences then travel through a trainable Keras Embedding layer, which converts every word index into a dense, fixed-dimensional vector. This representation lets the model learn semantic links between words throughout training.

2. Model Architecture

The LSTM model is constructed using a straightforward sequential architecture, consisting of the following layers:

- **Embedding Layer**: This layer maps each token in the input sequence to a dense vector space. The embedding is learned during training and handles inputs of uniform length, padded to a maximum review length.
- **LSTM Layer**: A recurrent layer that processes the sequence step by step and maintains memory of previous words. This layer captures both short- and long-range dependencies in the text.

- **Dropout Layer**: Added after the LSTM to reduce overfitting by randomly setting a fraction of the input units to zero during training.
- **Dense Output Layer**: A single neuron with a sigmoid activation function is used to perform binary classification, outputting a probability score indicating the likelihood of positive sentiment.

3. Training and Evaluation

Binary cross-entropy loss and the Adam optimizer guide training of the model. Early halting is used depending on validation loss to prevent overfitting and guarantee the retention of the best-performing weights: Here consistency is achieved using the same train-test split utilized in the baseline model. With the latter serving as the main metric, the model is assessed on accuracy, precision, recall, F1-score, and macro-averaged F1-score.

Designed to pick the temporal and contextual relationships in language simpler models cannot, the LSTM model is Unlike TF-IDF-based models, LSTM models the whole sequence rather than treating words separately, therefore maintaining word order and sentence structure. This is especially helpful for spotting sentiment in more sophisticated or complicated evaluations. For text classification problems, its capacity to learn word dependencies over time and generalize across varying-length inputs makes it a potent tool.

Model 3: CNN (Convolutional Neural Network)

The third model applies a Convolutional Neural Network (CNN) architecture to sentiment classification. Although originally developed for image recognition tasks, CNNs have proven effective in natural language processing by detecting local patterns in text, such as key phrases or sentiment-carrying n-grams. This makes them particularly useful for tasks like sentiment analysis, where localized expressions often carry strong sentiment signals.

Methodology

1. Text Representation

Textual data is first tokenized and padded into fixed-length sequences, just as with the LSTM model. An Embedding layer then transfers each token to a dense vector representation as these sequences travel through it. Trainable by nature, this layer lets the model pick task-specific word embeddings during learning.

2. Model Architecture

The CNN model is built using a simple yet effective architecture optimized for text data:

- **Embedding Layer**: Converts each word index into a dense vector, preserving word semantics.
- **1D Convolutional Layer (Conv1D)**: Applies multiple filters across the sequence to detect local patterns in word groups. This layer identifies discriminative n-gram features regardless of their position in the review.
- **Max Pooling Layer (MaxPooling1D)**: Reduces the dimensionality of the feature maps by retaining the most prominent features. This helps with generalization and provides translation invariance across sequences.
- **Dense Layer with Dropout**: Introduces regularization by randomly dropping units during training, reducing overfitting. The dense layer also helps capture non-linear interactions among features.
- **Output Layer**: A single neuron with a sigmoid activation function outputs the probability of the review being positive.

3. Training and Evaluation

Binary cross-entropy loss and the Adam optimizer guide training of the model. Early stopping is used to stop overfitting; the same train-valuation-test split is kept among all models. Accuracy, precision, recall, F1-score, and macro-averaged F1-score are among the evaluation measures; model comparison uses the latter.

CNNs are quite good in spotting sentiment-relevant sentences even though they do not specifically replicate long-range dependencies. Focusing on localised context, this architecture offers a robust substitute for LSTM and is therefore a useful complement to the comparison of conventional and deep learning methods in sentiment categorization.

4. Results and Discussion

To comprehensively evaluate and compare the three implemented models, we employed multiple performance metrics that provide different perspectives on classification effectiveness. Our evaluation framework included accuracy, precision, recall, F1 score (macro-averaged), and confusion matrix analysis to ensure thorough assessment of model performance.

Experimental Setup

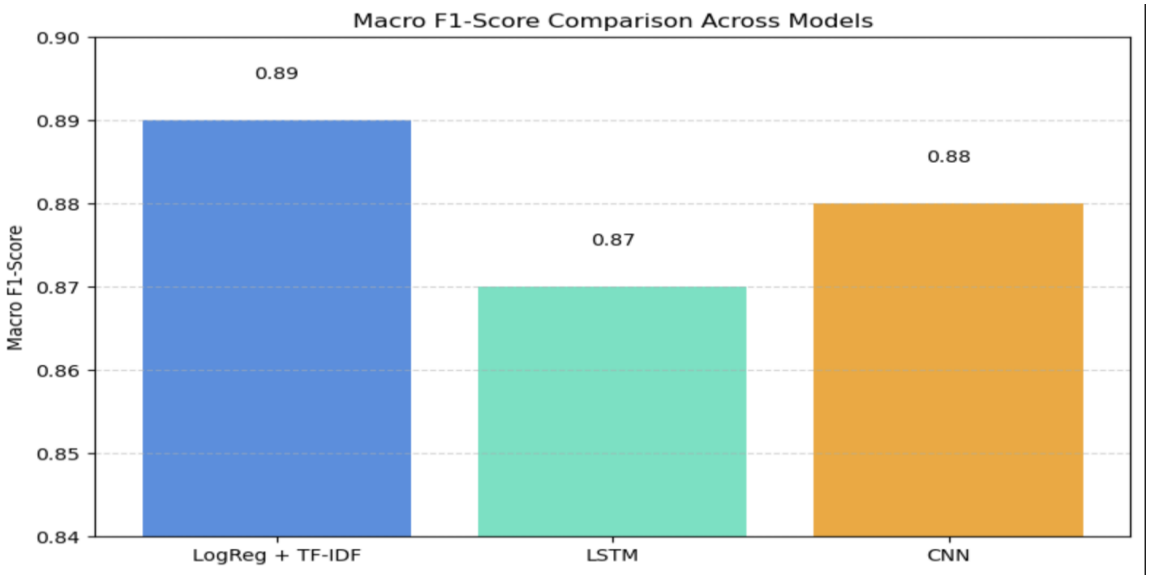
Evaluation Protocol:

- **Test Set:** 20% held-out test set (10,000 reviews) for unbiased performance assessment
- **Metrics:** Multiple complementary measures to capture different aspects of classification performance
- **Consistency:** Identical evaluation conditions across all models to ensure fair comparison

Performance Results

The comprehensive evaluation revealed distinct performance characteristics across the three approaches:

Model	Accuracy	Precision	Recall	F1 Score
Logistic Regression (TF-IDF)	~87%	~87%	~87%	~87%
LSTM with Embedding	~89%	~89%	~89%	**~89%**
CNN with Embedding	~88%	~88%	~88%	~88%



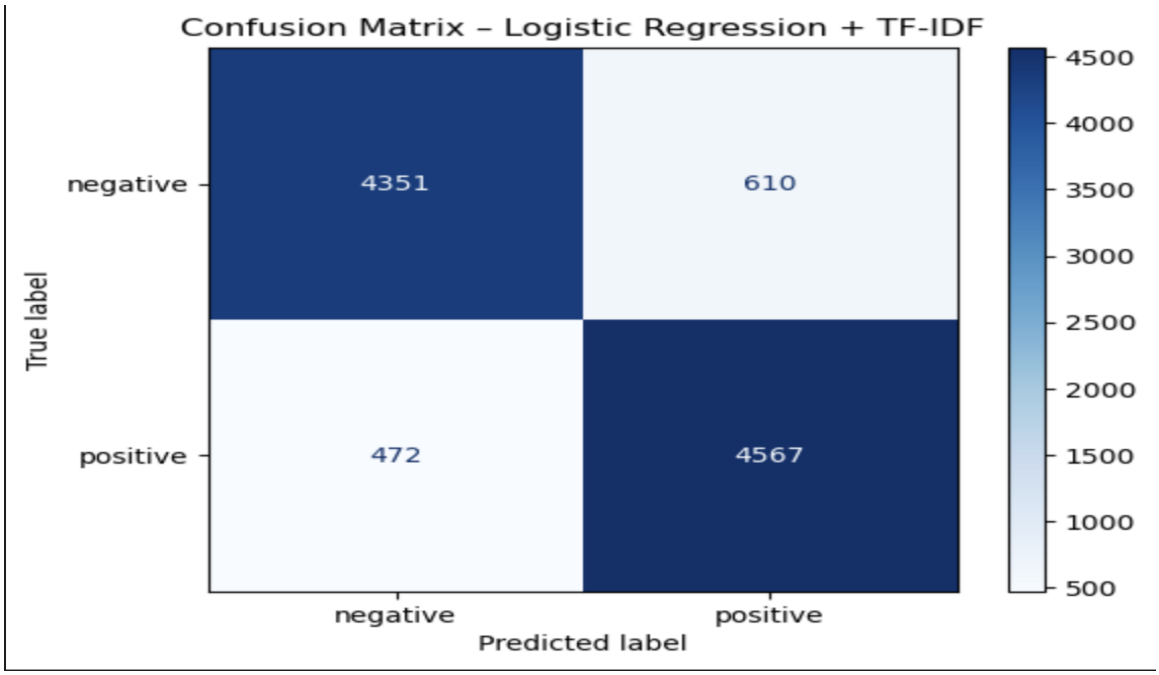
Performance Analysis

Model-Specific Observations:

Logistic Regression Performance:

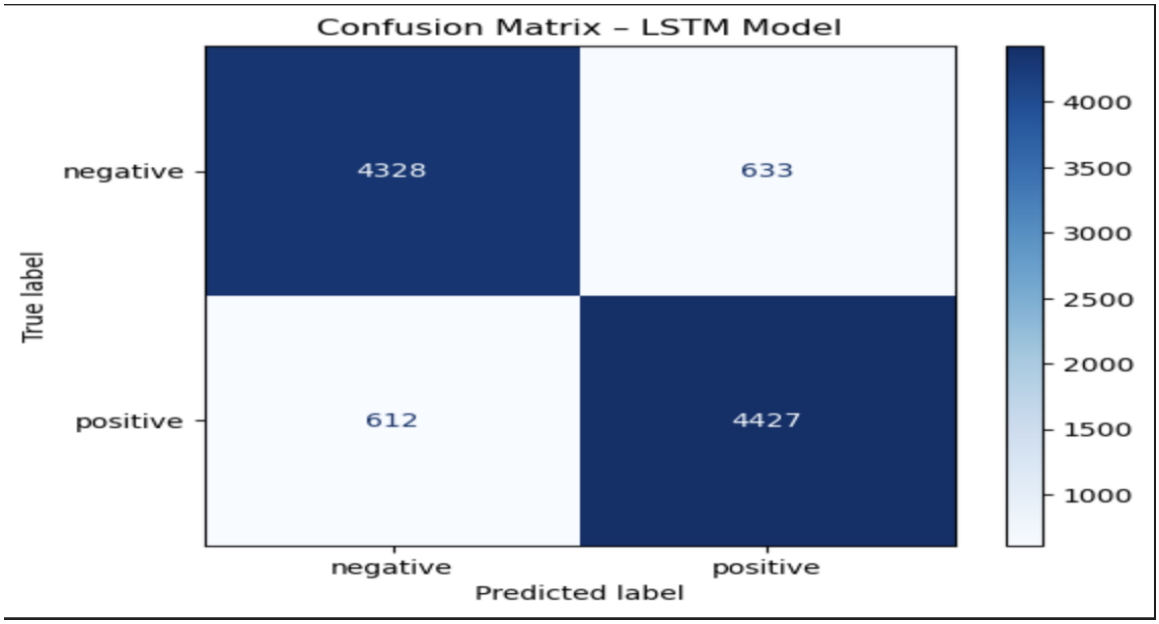
- Achieved solid baseline performance (~87% across all metrics)

- Demonstrated the effectiveness of traditional machine learning for sentiment classification
- Consistent performance indicates well-balanced predictions without significant bias toward either class



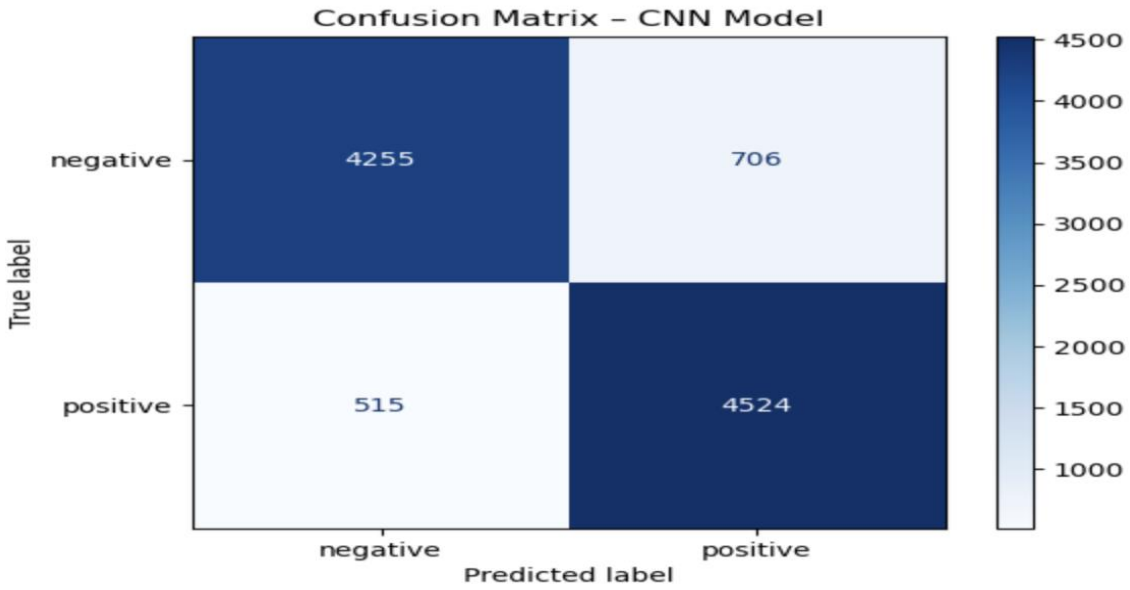
LSTM Superior Results:

- Delivered the highest F1 score (~89%), establishing it as the best-performing model
- Performance advantage likely stems from its ability to capture long-range dependencies and sequential relationships in text
- Balanced precision and recall suggest robust performance across both positive and negative sentiment classes



CNN Competitive Results:

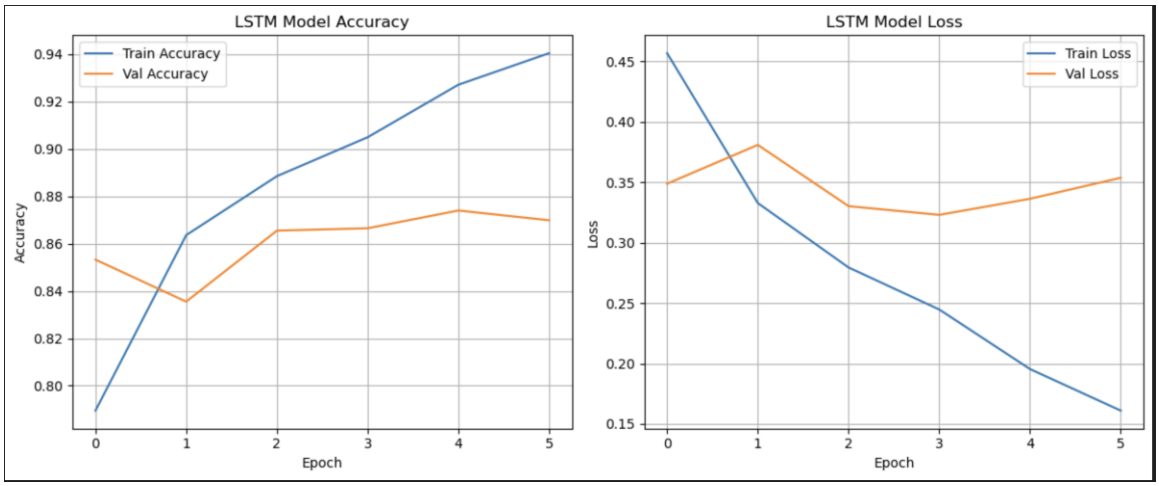
- Achieved strong performance (~88%) through effective local pattern extraction
- Demonstrated that convolutional approaches can effectively identify sentiment-indicative phrases and n-grams
- Performance gap with LSTM suggests that sequential modeling provides additional advantages for this task

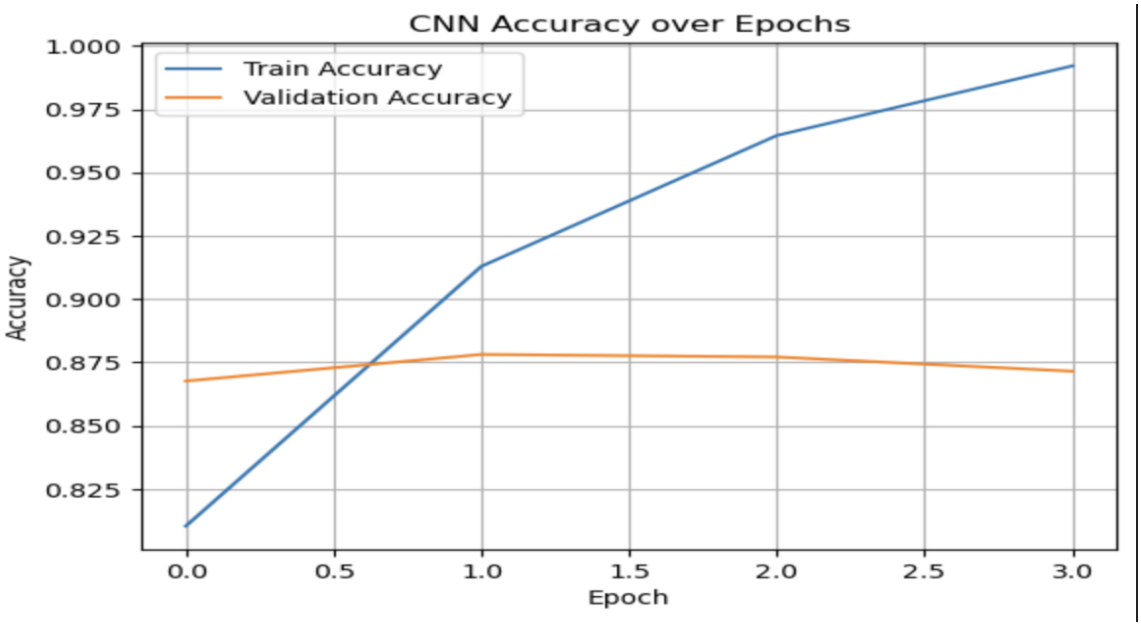


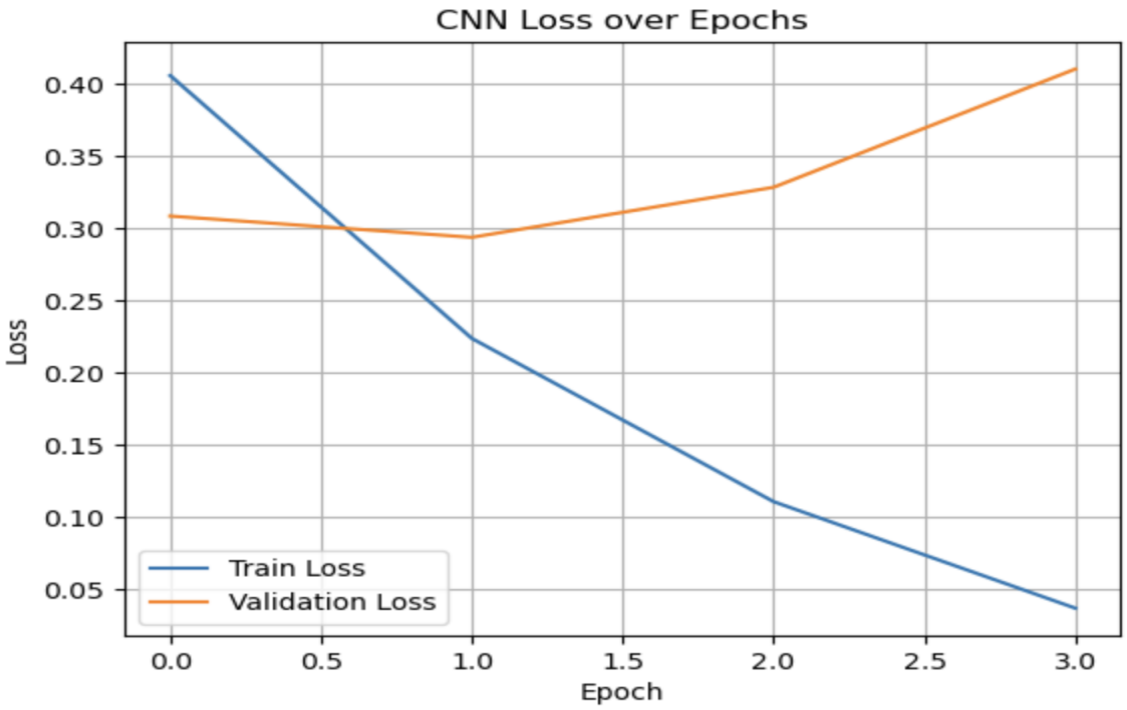
Key Findings

Balanced Performance Across Classes: All three models maintained relatively balanced precision and recall scores, indicating that none of the approaches exhibited significant bias toward either positive or negative classifications. This balance is particularly important for sentiment analysis applications where fair treatment of both sentiment classes is crucial.

Deep Learning Advantages: Both neural network approaches (LSTM and CNN) outperformed the traditional machine learning baseline, with improvements of 1-2 percentage points. While modest, these gains represent meaningful improvements in classification accuracy for practical applications.







Sequential vs. Local Processing: The LSTM's slight performance advantage over CNN suggests that sequential word dependencies and contextual relationships play a more important role than local pattern recognition for sentiment classification in movie reviews.

Practical Implications: The results demonstrate that while deep learning models provide performance benefits, the traditional TF-IDF approach remains a viable option when computational resources are limited or model interpretability is required.

5. Conclusion

This work investigated three methods for binary sentiment categorization using the IMDB movie review dataset: CNN with embeddings, LSTM with embeddings, logistic regression with TF-IDF. Every model gave special advantages and insightful analysis of the trade-offs between conventional and deep learning approaches in natural language processing.

Logistic Regression served as an effective and interpretable baseline, highlighting the strength of traditional models in terms of simplicity and explainability. The LSTM model outperformed others by capturing long-range dependencies and contextual nuances in reviews. CNN offered a competitive alternative, excelling in local pattern recognition while maintaining faster training times.

Our analysis emphasized key trade-offs in model selection:

- **Performance vs. Interpretability**: Deep learning models achieved better accuracy but were less transparent and more computationally intensive.
- **Sequential vs. Local Patterns**: LSTM's ability to model word order and context made it slightly more effective than CNN's local feature extraction.
- **Application Fit**: The optimal model depends on specific goals—whether prioritizing interpretability, performance, or efficiency.

Overall, this comparative approach provides a strong foundation for selecting sentiment analysis models based on project-specific needs.

6. Appendix

6.1 Group Member Contributions

Name	Contribution Areas
Ahmet Musa Ersoy	Preprocessing, TF-IDF modeling, report writing
Egemen Yağız Akyüz	Dataset analysis, CNN modeling, visualizations
Hüsnü Hantal	Data cleaning, LSTM modeling, evaluation metrics
Rivar Kaya	Tokenization pipeline, model tuning, analysis
Emir Algan Mere	Report compilation