Movie Reviews Sentiment Analysis

Team ID: 24

|  |  |  |
| --- | --- | --- |
| Level | ID | Name |
| SC 4 | **2021170488** | **محمد مصطفي محمد السيد** |
| SC 4 | **2021170345** | **عمار إبراهيم سمير قطورة** |
| SC 4 | **2021170520** | **مصطفي خالد محمود محمد** |
| SC 4 | **2021170493** | **محمود أحمد عبدالرحيم السيد** |
| SC 4 | **2021170232** | **سلمي سعيد محمد طه** |
| SC 4 | **2021170262** | **شهد عبدالمقصود علي أحمد** |

# Overview

This project implements sentiment analysis on movie reviews using both traditional Machine Learning and BERT-based approaches. The goal is to classify movie reviews as either positive or negative.

# Data

* **Source**: Movie reviews dataset from rt\_polarity stored in `all\_reviews.csv`
* **Structure**:
  + Size: **10,662** reviews
  + Features:
  + review: The movie review text
  + label: Sentiment label (1 for positive, 0 for negative)
* **Data Overview**:
  + No missing (null) values in the dataset
  + Some duplicate entries present (removed during preprocessing)
  + Balanced classes: 5,331 positive and 5,331 negative reviews
* **Text Characteristics**:
  + Review lengths are well within BERT's 128-token limit
  + Maximum review length: 59 tokens
  + Average review length: 21 tokens

## Review Length Distribution

# Data Preprocessing

## Text Preprocessing Pipeline

1. **Clean Text**
   * Remove URLs
   * Remove special characters and numbers
   * Remove extra whitespace
2. **Character Normalization**

* Normalize repeated characters (e.g., "goooood" → "good")

1. **N-gram Deduplication**
   * Remove duplicate n-grams for n=1,2,3
   * Process contiguous sequences
2. **Stop Words Removal**
   * Using NLTK stopwords
   * Preserving sentiment words:
     + not, no, never, very, really, too, so, much, many, few, little, hardly, barely, scarcely
3. **Word Normalization**
   * Primary approach: Lemmatization using NLTK's WordNetLemmatizer
   * Alternative tested: Stemming was also evaluated using NLTK's PorterStemmer
   * Comparison results showed no significant performance difference between stemming and lemmatization
   * Chose lemmatization for final implementation as it produces more readable words

# Models

## Machine Learning Approach

* **Vectorization**: TF-IDF with unigrams and bigrams
* **Models Used**:
  + - Logistic Regression
    - Linear SVM
    - Random Forest (n\_estimators=100)
* **Data Split**: 80% training, 20% testing

### Classification Reports

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer screen

AI-generated content may be incorrect.

A screenshot of a computer screen

AI-generated content may be incorrect.

## BERT Model

* **Base Model**: textattack/bert-base-uncased-SST-2
* **Device**: CUDA/CPU auto-detection

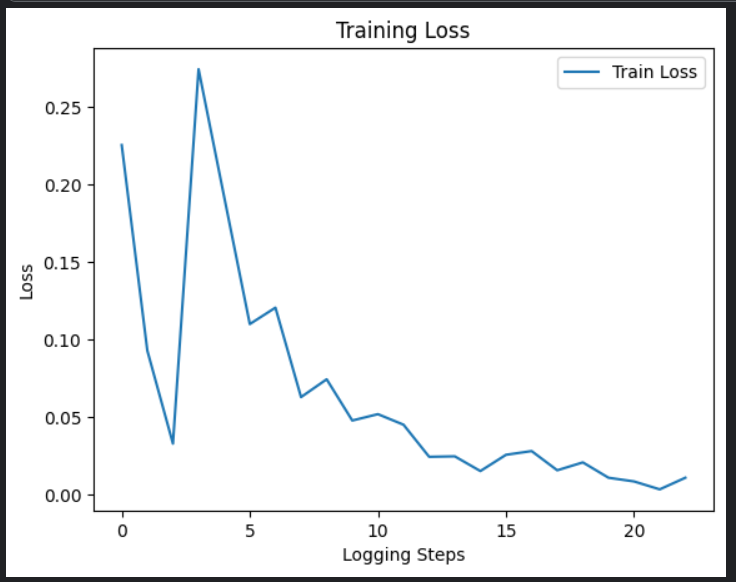
### Parameters

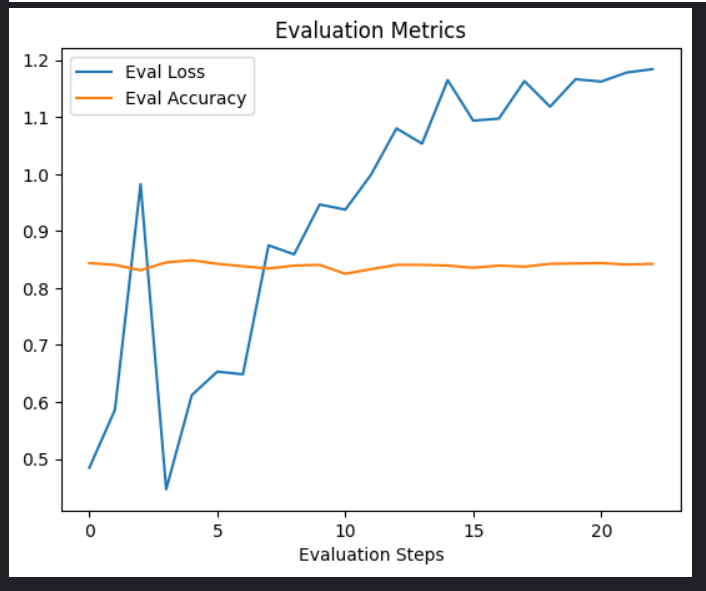
* BATCH\_SIZE = 16
* MAX\_LENGTH = 128
* LEARNING\_RATE = 2e-5
* NUM\_EPOCHS = 10
* PATIENCE = 2
* LOGGING\_STEPS = 50

### Training Configuration

* AdamW optimizer
* CrossEntropyLoss
* Data split:
* 70% training
* 15% validation
* 15% test
* Training arguments:
  + warmup\_steps = 200
  + weight\_decay = 0.01
  + gradient\_accumulation\_steps = 2

### Training Progress



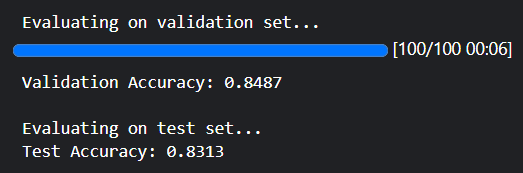


# Results and Evaluation

## Machine Learning Models Performance

* Evaluated using:
  + Accuracy
  + Precision
  + Recall
  + F1 Score

## BERT Model Performance



* Validation Accuracy: 84%
* Test Accuracy: 83%