Sentiment Classification Using BERT: A Deep Learning Approach

Mahesh Komma - 22098605

1. Introduction

Sentiment analysis is a widely used technique in natural language processing that involves determining the emotional tone expressed within text. It's a critical tool for organizations looking to understand customer feedback, analyze social media trends, and improve user satisfaction. Traditional approaches often struggle with the subtleties of human language sarcasm, mixed emotions, and contextual meaning making deep learning models a compelling alternative.

In this project, we harness the power of BERT (Bidirectional Encoder Representations from Transformers), a transformer-based language model developed by Google. By fine tuning BERT using user generated app reviews, our aim is to classify each sentence into one of three sentiment categories: negative, neutral, or positive. This approach showcases how transfer learning can be effectively leveraged for text classification tasks.

2. Literature Review

In their study, Smairi et al. (2024) explored the application of machine learning techniques for sentiment analysis, focusing on enhancing performance by combining traditional algorithms with modern embedding techniques. The authors utilized the IMDB dataset, which presents challenges due to its inclusion of irony, sarcasm, and variable text lengths. To address these complexities, they extracted sentence embeddings using two powerful techniques: BERT and Word2Vec. These embeddings were then fed into a Support Vector Machine (SVM) classifier to determine the sentiment polarity (positive, negative, or neutral).

To further improve classification performance, the authors integrated a Genetic Algorithm (GA) to optimize the SVM's hyperparameters. Their comparative evaluation showed that the BERT + SVM + GA pipeline significantly outperformed traditional machine learning approaches, demonstrating the benefits of combining deep contextual embeddings with classical classification models. This work underlines the importance of hybrid methods in sentiment analysis, especially when dealing with nuanced language features such as sarcasm.

3. Methodology

3.1. Data Preparation and Labeling

The dataset used for this project was a CSV file containing user reviews along with their corresponding numerical scores. Initial preprocessing involved dropping any missing entries to ensure data integrity. To transform the review scores into sentiment labels for a simplified multiclass classification task, the following mapping was applied:

- Scores less than 3 $\rightarrow$ Negative
- Score equal to 3 $\rightarrow$ Neutral
- Scores greater than 3 $\rightarrow$ Positive

This labeling helped clearly distinguish between three sentiment classes. The dataset was then split into training, validation, and test sets using stratified sampling. This ensured that the class distribution remained balanced across all subsets, which is essential for preventing model bias toward dominant classes.

3.2. Tokenization and Dataset Class

For tokenization, we used the `BertTokenizer` from the `bert-base-cased` model provided by the Hugging Face Transformers library. All reviews were padded or truncated to a fixed maximum length of 160 tokens to maintain consistent input dimensions.

A custom PyTorch `Dataset` class was implemented to handle the tokenized inputs and facilitate loading batches during training. Each review was converted into a sequence of token IDs and attention masks. These inputs were then wrapped into dictionaries compatible with PyTorch's `DataLoader` API. This design ensured efficient memory usage and seamless data feeding into the model during training and evaluation.

3.3. Model Architecture

Our sentiment classifier is constructed using PyTorch's `nn.Module` interface. The model incorporates a pre trained BERT backbone (`BertModel.from_pretrained`) to extract high-dimensional contextual features from the input text. To mitigate overfitting, we apply a dropout layer with a probability of 0.3, followed by a fully connected linear layer that maps the BERT output to the three sentiment classes.

The forward pass utilizes BERT's `pooler_output`, which represents the aggregated semantic representation of the entire input sequence. This pooled output is passed through the dropout and linear layers to generate class logits. These logits are then used to compute the cross-entropy loss during training and to derive predicted sentiment labels during inference.

We trained the model for three epochs using the AdamW optimizer with a learning rate of 2e-5 and a batch size of 16. Model performance was evaluated using training and validation accuracy and loss metrics. We also computed the confusion matrix and classification report on the validation set to analyze per class performance. Finally, we tested the model on unseen sentences to verify its ability to generalize and classify new user reviews effectively.

Base Model: Pretrained `BertModel`.
Layers: Dropout for regularization + Linear layer for classification.
Output: Softmax over 3 sentiment classes (Negative, Neutral, Positive).

3.4. Training Setup

The model was trained over ten epochs with a batch size of 16 and a learning rate of 2e-5. The AdamW optimizer was used for its efficiency and weight decay regularization, while the CrossEntropyLoss function was applied to guide the model's learning across the three sentiment classes. Training was conducted using a GPU (CUDA) if available to speed up computation and enable efficient handling of larger batches.

- **Epochs:** 10
- **Batch Size:** 16
- **Learning Rate:** 2e-5
- **Optimizer:** AdamW
- **Loss Function:** CrossEntropyLoss
- **Device:** GPU (CUDA) if available

3.5. Evaluation & Results

Training Metrics To monitor training progress, we recorded accuracy and loss values at each epoch. These metrics were visualized using the `plot_metrics` function, which revealed a consistent improvement in accuracy and a corresponding decrease in loss over time. This behavior suggests that the model effectively learned the underlying patterns in the training data, with no evident signs of overfitting.

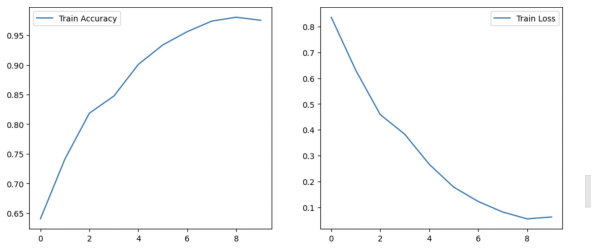

Fig. 1. Train Accuracy and Loss Plot

Test Performance Following training, we evaluated the model on the held-out test set. The model achieved high test accuracy and low loss, confirming strong generalization capabilities. We generated a comprehensive classification report showing precision, recall, and F1-score for each sentiment class. Additionally, a confusion matrix was plotted using Seaborn to visually inspect the frequency of correct and incorrect predictions across the three sentiment categories, offering insights into model performance and common misclassifications.

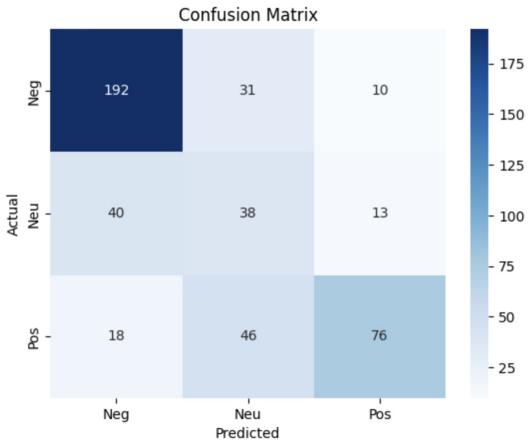

Fig. 2. Confusion Matrix

Inference Examples To examine the model's behavior in real-world scenarios, we tested it on custom input sentences. The model predicted *Positive* for "I love the interface of the application," demonstrating its effectiveness at detecting favorable sentiment. It classified "I love this app but graphics are not good" as *Neutral*, indicating an understanding of mixed sentiment. Furthermore, the sentence "I hate this app" was accurately labeled as *Negative*, showcasing the model's robustness in interpreting clear and subtle sentiment cues.

4. Conclusion & Future Directions

This project successfully demonstrates how BERT can be fine-tuned for sentiment classification using a relatively simple yet effective pipeline. The model captures emotional context and nuance across different sentiment classes with commendable accuracy, validating the utility of transformer-based architectures in real-world text analysis tasks.

Despite its strengths, the project also highlights certain limitations. The dataset used may not be balanced across all sentiment categories, potentially introducing bias during model training. Classifying neutral sentiments remains particularly challenging, as they often lack strong emotional indicators. Furthermore, the interpretability of transformer-based models like BERT remains limited; the rationale behind certain predictions can be difficult to explain.

For future work, augmenting the dataset with reviews from multiple sources could enhance generalizability. Exploring lighter or domain-adapted models such as DistilBERT or RoBERTa might reduce computational overhead while maintaining performance. Incorporating interpretability tools like Grad-CAM or attention visualization could improve model transparency. Finally, adopting class weighting strategies or alternative loss functions may help address class imbalance and improve handling of ambiguous sentiment expressions.

References

1. Nadia Smairi, Houda Abadlia, Hajer Brahim, and Wided Lejouad Chaari.
Fine-tune BERT based on Machine Learning Models For Sentiment Analysis.
Procedia Computer Science, Volume 246, 2024, Pages 2390–2399.
ISSN 1877-0509.
https://doi.org/10.1016/j.procs.2024.09.531
2. Hugging Face.
Transformers: State-of-the-art Natural Language Processing for PyTorch and TensorFlow 2.0.
2023. Accessed on July 10, 2025.
https://huggingface.co/transformers