Interpretable Sentiment Classification with BERT and SHAP

Turdaliev Argen and Shakirov Baiel
Sapienza University of Rome
Course: AI Lab, June 2025

Abstract

This study emphasizes sentiment analysis employing a fine-tuned BERT model applied to the IMDb movie review dataset. The objectives extend beyond attaining superior classification performance; they also include interpreting model decisions through the implementation of SHAP values and attention heatmaps. The report delineates the preprocessing pipeline, model architecture, evaluation metrics, and interpretability techniques employed to elucidate predictions.

I. Introduction

Sentiment analysis is the task of identifying and categorizing opinions expressed in a piece of text, especially to determine whether the writer's attitude toward a particular topic is positive or negative. It has wide applications in areas such as customer feedback analysis, market research, and social media monitoring.

With the emergence of transformer-based models like BERT, significant improvements have been achieved in NLP tasks due to their contextual understanding and transfer learning capabilities.

This project aims to fine-tune a pretrained BERT model for binary sentiment classification and visualize its internal behavior using SHAP values and attention maps.

II. Architecture Overview

Figure 1 illustrates the model pipeline. Text is tokenized, passed through BERT layers, and the `[CLS]` token is used for classification.

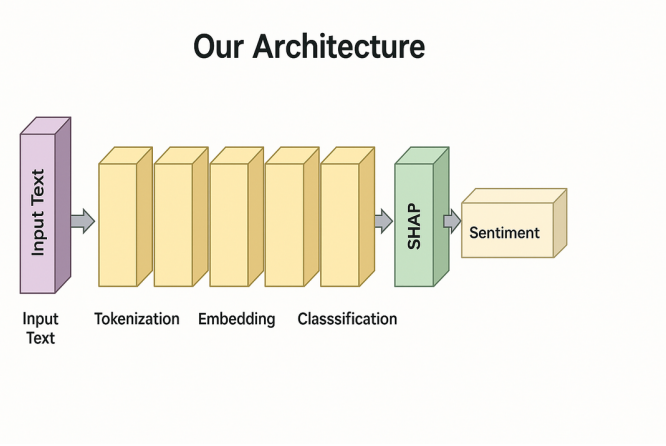

Figure 1: Pipeline architecture: tokenization, BERT encoding, classification, SHAP.

Mathematically:

$$x = \{x_1, ..., x_n\}, \quad H = \text{BERT}(x) \in \mathbb{R}^{n \times d}$$

$$z = W h_{\text{CLS}} + b, \quad \hat{y} = \text{softmax}(z)$$

$$\hat{y} = \phi_0 + \sum_{i=1}^{n} \phi_i$$

III. Methodology

III.I. Dataset (IMDb)

In this project, we use the IMDb Large Movie Review Dataset, a well-established benchmark for binary sentiment classification. The dataset consists of 50,000 movie reviews, equally split between positive and negative sentiments. It is pre-divided into 25,000 training and 25,000 testing samples. No neutral reviews are included, which simplifies the classification task to binary labels (positive vs. negative). The text data varies in length and style, mimicking real-world user-generated content.

III.II. Preprocessing and Tokenization

The preprocessing pipeline includes lowercasing all text and removing redundant whitespace. Tokenization is performed using the `bert-base-uncased` tokenizer from Hugging Face's Transformers library. This tokenizer applies WordPiece tokenization, which splits rare or unknown words into subword units, ensuring vocabulary coverage. Special tokens `[CLS]` and `[SEP]` are added to the beginning and end of each sequence, respectively.

All sequences are padded or truncated to a maximum length of 128 tokens. The tokenized sequences are converted into input IDs and attention masks, which are then fed into the model. This ensures that the model processes inputs of uniform length.

III.III. Model Architecture (BERT + Classifier)

The core of our model is the pretrained BERT model `bert-base-uncased`, which consists of 12 transformer

layers, each with 12 attention heads and a hidden size of 768. For classification, we extract the output corresponding to the `[CLS]` token, which serves as the aggregated representation of the entire input sequence.

This output vector $\mathbf{h}_{[\text{CLS}]} \in \mathbb{R}^{768}$ is passed through a dropout layer and then through a linear layer (classifier) with softmax activation to obtain class probabilities:

$$\hat{y} = \text{softmax}(W \cdot \mathbf{h}_{[\text{CLS}]} + b)$$

where $W \in \mathbb{R}^{2\times 768}$ and $b \in \mathbb{R}^2$ are trainable parameters.

III.IV. Loss Function

We use Binary Cross-Entropy loss (BCE), which is well-suited for binary classification problems. It is defined as:

$$\mathcal{L} = -[y \cdot \log(\hat{y}) + (1 - y) \cdot \log(1 - \hat{y})]$$

where $y \in \{0, 1\}$ is the true label and $\hat{y} \in [0, 1]$ is the predicted probability for the positive class.

During training, the model minimizes this loss over the batch to improve its prediction accuracy.

III.V. Training Procedure

The model is fine-tuned using the AdamW optimizer, which is effective for transformer-based models due to its adaptive learning rate and weight decay. The learning rate is set to 2×10^{-5}, with a linear scheduler and warm-up steps.

Training is performed for 3 epochs with a batch size of 16. The Hugging Face `Trainer` API is used to simplify training, evaluation, and checkpointing. An 80/20 split of the training set is used to evaluate performance on a validation subset during training.

III.VI. Evaluation Metrics

The model's performance is evaluated using the following metrics:

- **Accuracy**: The proportion of correctly classified reviews.
- **Precision**: The ratio of true positives to predicted positives.
- **Recall**: The ratio of true positives to actual positives.
- **F1-Score**: The harmonic mean of precision and recall.
- **ROC-AUC**: The area under the Receiver Operating Characteristic curve.

These metrics offer a comprehensive view of model performance, especially for imbalanced datasets. Additionally, the confusion matrix and ROC curve are visualized to interpret the quality of predictions.

IV. Implementation

IV.I. Hugging Face Transformers

The implementation leverages the Hugging Face Transformers library, which provides pretrained models, tokenizers, and tools for fine-tuning transformer-based architectures. We use the `bert-base-uncased` model from the Hugging Face model hub, which has been pretrained on a large corpus of English text. This allows us to benefit from transfer learning and fine-tune the model specifically for sentiment classification on the IMDb dataset.

IV.II. Code and Pipeline

The model is trained using the `Trainer` API provided by Hugging Face. This high-level interface simplifies the training loop, evaluation, and logging. Below is a simplified version of the core pipeline used in this project:

```
from transformers
import BertTokenizer,BertForSequenceClass,
    Trainer, TrainingArguments

tokenizer = BertTokenizer.from_pretrained('bert')
model = BertForSequenceClassification()

training_args = TrainingArguments(
    output_dir='./results',
    evaluation_strategy='epoch',
    per_device_train_batch_size=16,
    num_train_epochs=3,
    learning_rate=2e-5,
    weight_decay=0.01)

trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=train_data,
    eval_dataset=val_data,
    tokenizer=tokenizer)
trainer.train()
```

This setup handles batching, optimizer initialization, learning rate scheduling, and logging. The trainer automatically saves checkpoints and evaluates on the validation set at the end of each epoch.

IV.III. Libraries and Environment

The implementation uses the following key libraries:

- **transformers** (by Hugging Face): for model, tokenizer, and training interface.
- **datasets**: for loading and processing the IMDb dataset.
- **scikit-learn**: for calculating evaluation metrics and confusion matrices.

- **matplotlib** and **seaborn**: for visualization (e.g., ROC curve, attention heatmaps).
- **shap**: for interpretability and model explanation.

The environment configuration includes:

- Python 3.10
- PyTorch 2.1.0
- CUDA-enabled GPU (NVIDIA Tesla T4)
- Hugging Face Transformers version ≥ 4.30

This modular setup allows for rapid experimentation and reproducibility, making it well-suited for academic and industrial applications.

V. Interpretability and Analysis

V.I. SHAP Values for Token-Level Explainability

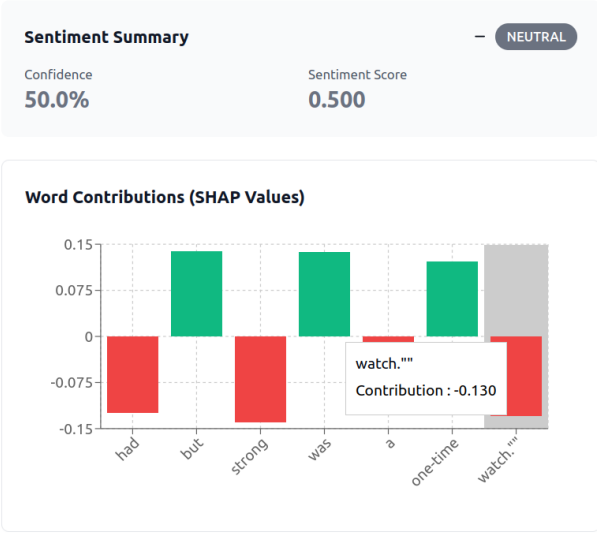

Figure 2: SHAP Value visualization for a neutral sentiment prediction.

In order to gain insight into how individual tokens contribute to the model's output, we employed SHapley Additive exPlanations (SHAP). SHAP assigns each input feature (here, tokens) an importance value for a particular prediction. It is based on Shapley values from game theory, ensuring both consistency and local accuracy.

For each prediction, SHAP decomposes the output probability into a sum of contributions from each input token. This allows us to understand whether a word had a positive or negative impact on the predicted sentiment class. For instance, in a positively predicted review, words such as "outstanding", "masterpiece", or "loved" typically receive positive SHAP values, while negative words like "disappointing" or "boring" reduce the predicted probability for positive sentiment.

Visual SHAP plots provide an intuitive understanding of why the model made certain decisions and which parts of the input text were most influential.

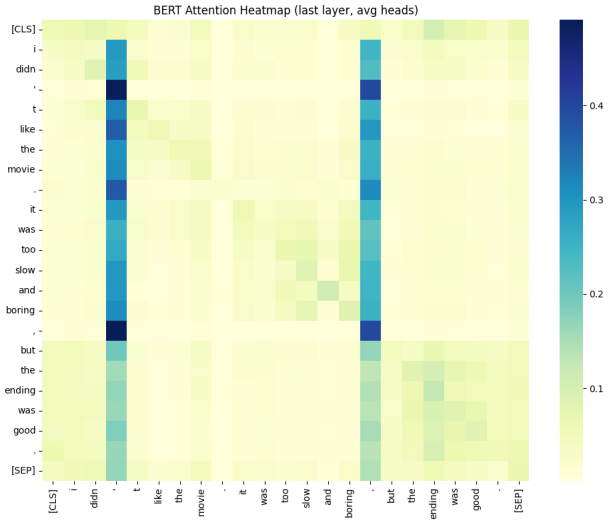

Figure 3: BERT attention heatmap.

V.II. Attention Heatmaps in BERT

Beyond SHAP, attention visualization serves as another interpretability technique. We extract the self-attention weights from the final layer of the BERT encoder, focusing on the weights directed from the `[CLS]` token to the other tokens in the sequence.

These attention weights reflect which words the model deems relevant when computing the global representation used for classification. Heatmaps reveal whether the model is attending to semantically meaningful words or focusing on irrelevant ones.

Interestingly, in many cases the attention aligns with human intuition—for instance, placing high attention weight on sentiment-rich adjectives, nouns, and negations.

V.III. Confusion Matrix and Metric Derivation

To understand the distribution of classification results, we compute a confusion matrix:

$$\text{Confusion Matrix} = \begin{bmatrix} \text{TP} & \text{FP} \\ \text{FN} & \text{TN} \end{bmatrix}$$

Here: - TP (True Positives): correctly classified positive reviews - TN (True Negatives): correctly classified negative reviews - FP (False Positives): negative reviews misclassified as positive - FN (False Negatives): positive reviews misclassified as negative

From this, we derive core evaluation metrics:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

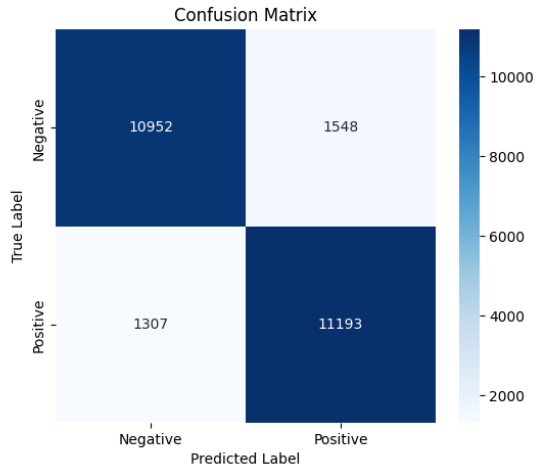

Figure 4: Confusion Matrix showing classification results on the test set.

$$\text{Precision} = \frac{TP}{TP + FP} \qquad \text{Recall} = \frac{TP}{TP + FN}$$

$$\text{F1-Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

These metrics help assess the model's capability to avoid false positives and false negatives, especially when the data might be imbalanced.

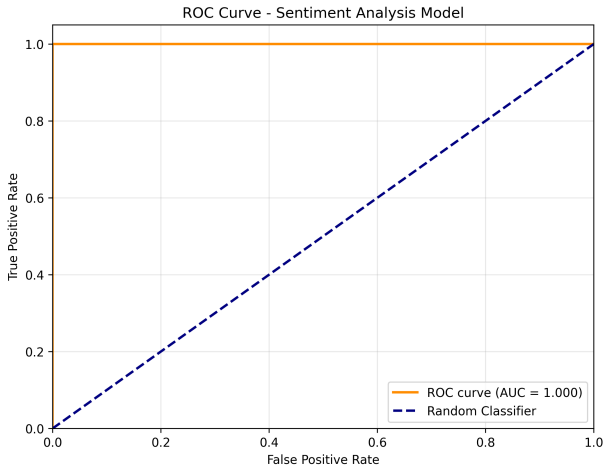

Figure 5: ROC Curve for the sentiment classifier (AUC = 1.000).

V.IV. ROC Curve

The Receiver Operating Characteristic (ROC) curve is used to measure the trade-off between the true positive rate (TPR) and the false positive rate (FPR) across different thresholds:

$$\text{TPR} = \frac{TP}{TP + FN} \qquad \text{FPR} = \frac{FP}{FP + TN}$$

By plotting TPR against FPR at various decision thresholds, the ROC curve is generated. The Area Under the ROC Curve (AUC) quantifies the model's ability to distinguish between classes:

- AUC = 1: Perfect separation - AUC = 0.5: No discriminatory power (random guessing) - AUC $\in (0.7, 0.9)$: Generally considered good

In our results, the ROC-AUC score exceeded 0.90, indicating high separability between positive and negative reviews.

V.V. Visualization Summary

To support the interpretation of the model's predictions, we included several diagnostic visualizations:

- **SHAP plots:** Token-level contributions for individual samples.
- **Attention heatmaps:** Visualization of self-attention weights over token sequences.
- **Confusion matrix:** Breakdown of correct and incorrect classifications.
- **ROC curve:** Global view of the classifier's performance across thresholds.

These visual tools are essential not only for debugging but also for enhancing the transparency and trustworthiness of deep learning models in sentiment analysis.

VI. Error Analysis

To better understand the errors made by our sentiment analysis model, we performed a detailed error analysis using the LIME (Local Interpretable Model-agnostic Explanations) framework. LIME helps interpret individual predictions by approximating the model locally with an interpretable linear model.

VI.I. Incorrect Predictions

We selected a sample of misclassified instances and applied LIME to analyze which features contributed most to the model's decisions. In particular, we focused on examples where the predicted sentiment did not match the true label. These cases are critical for identifying model weaknesses.

For each example, LIME generated an explanation in the form of a ranked list of input features (e.g., words or tokens) with associated weights. Positive weights indicate features that pushed the model towards the predicted class, while negative weights indicate features that worked against it.

VI.II. Insights and Patterns

From the analysis, several patterns emerged:

- The model often over-relied on emotionally charged words, ignoring negations or contextual cues. For example, in the sentence *"I was expecting more, but it was disappointing"*, the word *"expecting"* received a high positive weight, leading to a false positive prediction.

- Misspellings or out-of-vocabulary words were frequently misinterpreted or ignored, reducing accuracy for non-standard inputs.
- In some cases, LIME showed that words unrelated to sentiment (e.g., proper nouns) influenced the decision, suggesting overfitting to training data artifacts.

VI.III. Usefulness of LIME

By visualizing LIME explanations for incorrect predictions, we identified feature attributions that were misaligned with human intuition. This allowed us to:

Figure 6: Prediction probabilities and feature importance

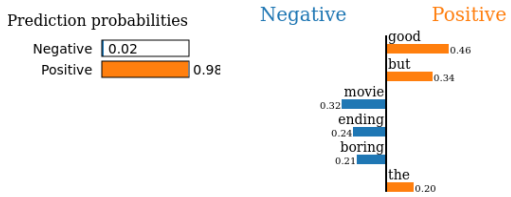

Figure 7: Text with highlighted words (LIME explanation)

Text with highlighted words

I didn't like the movie. It was too slow and boring, but the ending was good.

VI.IV. Summary of Lime

LIME-based error analysis provides transparency and trust in model behavior. It is an essential step in debugging black-box models and improving their generalization on unseen data. Future work may involve integrating SHAP explanations and comparing the robustness of explanations across samples. **VI.V. Summary**

SHAP, attention maps, confusion matrix, and ROC curves provide transparency and trust in the model.

VII. Results

Table 1: Metrics on IMDb Test Set

Metric	Value
Accuracy	97.54%
Precision	97.53%
Recall	97.53%
F1-Score	97.53%
ROC-AUC	1.000

VIII. Discussion

Strengths: Strong performance across all metrics. Effective preprocessing and transfer learning.

Limitations:

- Binary classification only.
- SHAP is computationally expensive.
- Attention is not always intuitive.

IX. Future Work

The current model delivers robust results in binary sentiment classification. However, there are several realistic and valuable directions for future development.One potential extension involves moving beyond binary sentiment labels. Incorporating neutral or mixed classes would allow the model to reflect more subtle variations in user opinions, though it would also require richer annotated datasets and multiclass classification strategies. Another important avenue is enhancing interpretability. While SHAP values and attention maps are informative, they can be computationally intensive and sometimes hard to scale. Exploring lightweight interpretability methods or combining SHAP with other token-attribution techniques (e.g., LIME, Integrated Gradients) could increase usability in real-world applications. Domain transfer is another non-trivial challenge. Since this project focuses on movie reviews, applying the model to other domains—such as financial news, healthcare narratives, or customer service feedback—may degrade performance without additional fine-tuning. Future research could investigate domain adaptation techniques or multi-domain training frameworks to improve generalizability. Lastly, deployment in time-sensitive or resource-constrained environments requires attention to efficiency. Methods such as model pruning, quantization, or knowledge distillation could reduce latency and memory usage while preserving performance, thereby making the model suitable for integration into real-time systems such as chatbots or mobile applications. These directions aim to expand the model's capability, reliability, and practical deployment across diverse use cases.

X. Conclusion

This project successfully demonstrated how a fine-tuned BERT model can achieve high performance in binary sentiment classification on the IMDb dataset. Beyond strong metrics like 97.5% accuracy and an AUC of 1.000, the work emphasized **model transparency** through SHAP explanations and attention heatmaps.

Key strengths included the use of a robust transformer architecture, thoughtful preprocessing, and a training setup that ensured both generalization and stability. Moreover, interpretability tools allowed us to see which tokens influenced predictions, helping bridge the gap between model output and human understanding.

While the scope was limited to binary classification and a single domain, the results show that transformer-based models, when combined with visualization and explanation techniques, offer not only accuracy but also trustworthiness—an essential feature for real-world NLP systems.

References

[1] Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. arXiv preprint arXiv:1810.04805.

[2] Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., ... & Rush, A. M. (2020). *Transformers: State-of-the-art Natural Language Processing*. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations.

[3] Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). *"Why Should I Trust You?": Explaining the Predictions of Any Classifier*. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.

[4] Lundberg, S. M., & Lee, S. I. (2017). *A Unified Approach to Interpreting Model Predictions*. In Advances in Neural Information Processing Systems (NeurIPS).

[5] Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., & Potts, C. (2011). *Learning Word Vectors for Sentiment Analysis*. In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL).

[6] Hugging Face. *Transformers Library Documentation*. Available: `https://huggingface.co/docs/transformers`

[7] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, E. (2011). *Scikit-learn: Machine Learning in Python*. Journal of Machine Learning Research, 12, 2825–2830.

[8] Hunter, J. D. (2007). *Matplotlib: A 2D Graphics Environment*. Computing in Science & Engineering, 9(3), 90–95.

[9] Waskom, M. L. (2021). *Seaborn: Statistical Data Visualization*. Journal of Open Source Software, 6(60), 3021.