Sentiment Analysis Using Fine-Tuned BERT Architecture with Semantic Search

Abdulrrahim Ahmadov, Tomáš Timko, Maroš Matej
Faculty of Cybernetics and Artificial Intelligence
Technical University of Košice
Emails: abdulrrahim.ahmadov@student.tuke.sk, tomas.timko@student.tuke.sk, maros.matej@student.tuke.sk

***Abstract*—This paper presents the development and implementation of a sentiment classification model leveraging a fine-tuned BERT architecture. The model categorizes text data into three classes: Positive, Negative, and Neutral. The preprocessing pipeline, efficient memory management, and model training strategies are outlined, along with the results achieved in terms of accuracy, precision, recall, and F1 score. Additionally, a semantic search pipeline incorporating the fine-tuned model is discussed. Challenges encountered during the project and insights gained are also detailed.**

I. Introduction

Sentiment classification is a crucial task in natural language processing (NLP), with applications spanning business, healthcare, and entertainment. This work explores the use of a pre-trained BERT model fine-tuned on the Rotten Tomatoes Movie Reviews dataset to classify sentiments effectively. The goal is to transform reviews into labeled data representing Positive, Negative, and Neutral sentiments, while ensuring efficient processing of a dataset containing 1.44 million reviews.

II. Preprocessing

The preprocessing of the Rotten Tomatoes dataset involved several steps to handle its large scale effectively. Key steps included:

- **Data Cleaning:** Rows with missing values in critical columns (e.g., reviewText and scoreSentiment) were removed. Sentiment labels (*fresh* and *rotten*) were mapped to numerical values (0 and 1).
- **Chunk-Based Processing:** The dataset was processed in 10,000-row chunks using pandas to manage memory usage effectively. Each chunk was processed, tokenized, and saved incrementally.
- **Tokenization and Padding:** A BERT tokenizer (*bert-base-uncased*) was employed to tokenize text. Tokenized reviews were padded or truncated to a maximum sequence length of 64 tokens, chosen based on the 95th percentile of token lengths to ensure minimal truncation.
- **Label Encoding:** Sentiment labels were encoded as integers for compatibility with machine learning frameworks.
- **Data Conversion:** The processed data was converted to PyTorch tensors, including `input_ids`, `attention_masks`, and sentiment labels.
- **Data Splitting:** Stratified splitting ensured an even distribution of sentiment classes across the training (80%), validation (10%), and testing (10%) subsets. Data chunks were stored as PyTorch (`.pt`) files to avoid redundant processing.
- **DataLoader Preparation:** Processed chunks were combined using PyTorch's `ConcatDataset`, and DataLoader objects were prepared with a batch size of 16 for training and evaluation.

III. Methodology

The project initially aimed to classify reviews as fake or genuine but shifted to sentiment classification due to challenges in dataset availability and subjective labeling.

- **Dataset:** The Rotten Tomatoes dataset included 1.44 million reviews with attributes like *reviewText*, *scoreSentiment*, and *originalScore*.
- **Model Architecture:** The Hugging Face `BertForSequenceClassification` model (*bert-base-uncased*) was fine-tuned for sentiment analysis, with three output labels representing Positive, Negative, and Neutral sentiments.
- **Training Configuration:**
  - Optimizer: Adam with a learning rate of 2e-5.
  - Batch Size: 16.
  - Epochs: 3.
  - Weight Decay: 0.01.
- **Evaluation Metrics:** The performance was assessed using accuracy, precision, recall, and F1 score (weighted), computed via scikit-learn's metrics.

IV. Results

Training the model on an NVIDIA GTX 1070 GPU took approximately seven hours for three epochs. The performance metrics achieved were:

TABLE I
Performance Metrics per Epoch

Epoch	Loss	Accuracy	Precision	Recall	F1-Score
1	0.23	90%	90%	90%	90%
2	0.22	91%	91%	91%	91%
3	0.25	91%	91%	91%	91%

V. Semantic Search and Sentiment Analysis Pipeline

The project also implemented a semantic search pipeline, enabling efficient review retrieval based on semantic similarity and sentiment classification:

- **Semantic Search Workflow:** Reviews were converted into dense vector representations using the fine-tuned BERT model. These vectors were stored in a SQLite database alongside review text. A FAISS index enabled efficient nearest-neighbor search.
- **Search Query:** User queries were vectorized using the same BERT model, and FAISS retrieved the most similar reviews based on cosine similarity scores.
- **Combined Output:** The final results included review text, similarity scores, sentiment labels, and confidence levels for sentiment classification.

VI. Challenges and Insights

- **Problem Redefinition:** The original aim of classifying reviews as fake or genuine was revised due to the subjective nature of *fake* reviews and lack of clear datasets.
- **Resource Constraints:** Careful tuning of batch size, sequence length, and hyperparameters was essential to maximize model performance without exceeding hardware limits.
- **Scalability:** Chunk-based preprocessing, batch tokenization, and intermediate saves ensured the pipeline could handle large datasets efficiently.
- **Dynamic Sequence Length Selection:** Analyzing token length distributions enabled the selection of a 64-token limit, balancing computational efficiency and truncation avoidance.

VII. Conclusion

This work demonstrates the successful implementation of a fine-tuned BERT model for sentiment classification and semantic search. By addressing preprocessing challenges, optimizing model configurations, and integrating semantic similarity retrieval, the project provides a robust framework for large-scale text analysis. Future work may involve extending the model to additional sentiment categories or exploring multilingual datasets.