SentimentScope Project Report

Project Overview

As a Machine Learning Engineer at CineScope, the objective was to enhance the recommendation system by understanding user sentiment. This was achieved by building and training a transformer model from scratch to perform binary classification on the IMDB dataset, distinguishing between positive and negative movie reviews.

Methodology and Implementation

1. Data Preparation & Tokenisation

- **Data Loading:** The IMDB dataset was processed by reading text files from the 'pos' (positive) and 'neg' (negative) directories, creating a labelled Pandas DataFrame where positive reviews are labelled as 1 and negative reviews are labelled as 0.
- **Splitting:** The training data was shuffled and split into a 90% training set and a 10% validation set to ensure the model could be evaluated on unseen data during training.
- **Tokenisation:** Utilised the bert-base-uncased AutoTokenizer from Hugging Face. This approach employs subword tokenisation, splitting words into smaller components and converting them into numerical token IDs, which enhances vocabulary efficiency compared to character-level tokenisation.
- **Dataset Class:** A custom IMDBDataset class was implemented to handle text preprocessing, truncation, and padding to a maximum sequence length of 128.

2. Transformer Architecture Customisation

To adapt the transformer for binary classification, specific architectural modifications were made to the DemoGPT class:

- **Embeddings:** The model utilises a token embedding layer and a positional embedding layer with a dimensionality of 128.
- **Core Blocks:** The architecture consists of four stacked transformer layers (layers_num), each containing Multi-Head Attention and Feed-Forward modules.
- **Classification Head:** Unlike generation models, this architecture includes a linear layer (classification_head) that maps the final embeddings to 2 output classes.
- **Pooling:** A mean pooling mechanism was implemented (torch.mean) to aggregate token-level embeddings across the time dimension into a single vector before classification.

Project Results

The project successfully established a complete End-to-End Machine Learning pipeline:

- **Data Pipeline:** A robust PyTorch DataLoader was implemented to manage batching (batch size 32) and shuffling, ensuring efficient data feeding during training.
- **Training Loop:** A training loop was executed over 5 epochs using CrossEntropyLoss and

the AdamW optimiser with a learning rate of 3e^-4.
- **Validation Monitoring:** An accuracy calculation function was integrated to monitor the model's performance on the validation set after every epoch, ensuring the model generalises well rather than just memorising training data.
- **Evaluation Goal:** The final pipeline includes a testing phase to evaluate the model against the test dataset, with the explicit goal of achieving greater than 75% accuracy to ensure utility for CineScope's recommendation system.

Key Takeaways

- **Adapting Transformers for Classification:** While often used for text generation, transformers are highly effective for classification when modified with a pooling strategy and a specific classification head.
- **Importance of Subword Tokenisation:** Using a pre-trained tokeniser like bert-base-uncased handles out-of-vocabulary words effectively by breaking them into known subwords, which is crucial for processing diverse movie reviews.
- **Generalisation via Validation:** Splitting training data into training and validation sets is critical. Randomly shuffling data before splitting ensures that the validation set is representative, preventing the model from biasing toward specific data patterns.
- **PyTorch Abstractions:** Leveraging torch.utils.data.Dataset and DataLoader simplify complex data handling tasks, such as batching and padding, allowing the focus to remain on model architecture and optimisation.