Report: Sentiment Analysis using Indic BERT and RNN

Team Binokligons

1. Adarsh Malviya – 200101007
2. Gandrathi Srija – 200101030
3. Sidhatrth Choudhary - 200101095

Problem Statement:

The problem addressed in this project is sentiment analysis of Hindi text. Sentiment analysis involves determining the sentiment or emotion expressed in a piece of text, whether it is positive, negative, or neutral. In this case, the focus is on analyzing the sentiment of Hindi text using a combination of Indic BERT and an RNN model.

Dataset:

The details about the dataset used for training and evaluation is a labeled dataset of Hindi text for sentiment analysis. The dataset is loaded from a CSV file named "sentiment_data.csv." In sentiment analysis, labeled datasets typically consist of text samples along with corresponding sentiment labels indicating the sentiment expressed in the text (e.g., positive, negative, neutral). The datasets which were used for training and evaluating are also labeled in same format.

Model:

The implemented approach combines the power of Indic BERT and an RNN model for sentiment analysis. Here is an overview of the implementation:

1. **Tokenization:** The input text from the dataset is tokenized using the Indic BERT tokenizer. Each text is encoded, and input IDs are created as tensors.
2. **Feature Extraction:** The encoded input IDs are passed through the Indic BERT model to extract the hidden states from the last four layers. These hidden states are concatenated and stored in the **`features`** tensor.
3. **RNN Model:** A custom RNN model for sentiment analysis, called **`SentimentRNN`**, is defined. It consists of an RNN layer followed by a linear layer. The RNN model takes the **`features`** tensor as input, performs the forward pass, and predicts the sentiment.
4. **Training:** The RNN model is trained using the features extracted from the Indic BERT model. The training is performed using batches of features and labels. The loss is calculated using the **`nn.BCEWithLogitsLoss()`** function, and optimization is done using the Adam optimizer.
5. **Saving and Loading Model:** The trained weights of the RNN model are saved to a file named "sentiment_rnn.pth" using **`torch.save()`**. Later, the model is loaded by creating a new instance of **`SentimentRNN`** and loading the saved weights using **`torch.load()`** and **`model.load_state_dict()`**.

Bert Model and RNN model :

The AlbertModel is based on the ALBERT (A Lite BERT) architecture, which is a variant of the BERT (Bidirectional Encoder Representations from Transformers) model specifically designed for efficient parameter sharing and model compression.

The model consists of several components:

- **Embeddings**: The AlbertEmbeddings module handles the input embeddings for the model. It includes word embeddings, position embeddings, and token type embeddings. The word embeddings are initialized with a vocabulary size of 200,000 and embedding dimension of 128.
- **Encoder**: The AlbertTransformer module represents the core transformer encoder of the model. It consists of multiple layers of AlbertLayer. Each AlbertLayer contains an attention mechanism, feed-forward network, and layer normalization.
- **Pooler**: The Linear layer used for pooling the hidden states of the last layer in the transformer encoder. It maps the hidden state representation to a fixed-sized output representation.
- **Pooler Activation**: The Tanh activation function applied to the pooled output representation.

The specific configuration and hyperparameters of the AlbertModel used in the code are not provided. It's worth noting that the AlbertModel is initialized with the pre-trained weights from the "ai4bharat/indic-bert" model, which is a variant of ALBERT pre-trained on Indian languages.

The model is then trained using the SentimentRNN model, which is an RNN-based architecture for sentiment analysis. The SentimentRNN takes the hidden states from the AlbertModel as input and produces sentiment predictions.

Overall, the model architecture combines the power of the ALBERT transformer encoder with the sentiment analysis capabilities of the SentimentRNN to perform sentiment analysis on Hindi text.

Results:

The provided code includes a training loop that prints the loss for every 10 batches during the training process. Here are the reported loss values for each epoch:

- Epoch [1/10], Batch [10/16], Loss: 0.6799
- Epoch [2/10], Batch [10/16], Loss: 0.4821
- Epoch [3/10], Batch [10/16], Loss: 0.2505
- Epoch [4/10], Batch [10/16], Loss: 0.2459
- Epoch [5/10], Batch [10/16], Loss: 0.2081
- Epoch [6/10], Batch [10/16], Loss: 0.1861
- Epoch [7/10], Batch [10/16], Loss: 0.5587
- Epoch [8/10], Batch [10/16], Loss: 0.6298
- Epoch [9/10], Batch [10/16], Loss: 0.0468
- Epoch [10/10], Batch [10/16], Loss: 0.2510

After the training process, the model is evaluated on the validation set, and the validation loss is reported as 0.2401.

Observation:

Based on the training loss values, it has been observed that the model's performance improves over epochs as the loss decreases.

The validation loss of 0.2401 indicates that the model generalizes well on the unseen validation set. It is important to note that the performance of the model can be further evaluated by considering other evaluation metrics such as accuracy, precision, recall, and F1 score.

Conclusion:

During the training process, the RNN model was trained over multiple epochs, and the loss decreased gradually with each epoch. This indicates that the model was learning and improving its ability to predict the sentiment of Hindi movie reviews.

To evaluate the model's performance, a validation set was used, and the validation loss was calculated. The obtained validation loss suggests that the model is capable of generalizing well on unseen data, as the loss value was relatively low.

It is worth mentioning that the current implementation could benefit from certain improvements. Firstly, a larger and more diverse dataset would help to enhance the model's ability to generalize across different types of movie reviews and sentiments. Additionally, fine-tuning the Indic BERT model on domain-specific data or exploring transfer learning techniques might further enhance the model's performance.

Overall, the combination of Indic BERT and an RNN model shows potential for sentiment analysis of Hindi text. However, further experimentation, evaluation metrics, and comprehensive understanding of the dataset are required to draw more definitive conclusions and optimize the sentiment analysis system for Hindi movie reviews.