Training a BERT Model from Scratch

Importing the Data and Preprocessing

Import Required Libraries

- pandas
- re
- torch
- transformers (BertTokenizer, BertForSequenceClassification, AdamW, get_linear_schedule_with_warmup)
- sklearn (accuracy_score, precision_score, recall_score, f1_score)
- torch.utils.data (DataLoader, Dataset)
- torch.nn.utils.rnn (pad_sequence)
- sklearn.utils (resample)

Load and Parse the Log File

1. **Load the Log File**:
    - Path: `'cisco_log.txt'`
2. **Parse Log Entries**:
    - Use regex to extract date and message from each log line.
    - If a line does not match the pattern, store the whole line as the message with `None` as the date.
3. **Convert Parsed Entries to DataFrame**:
    - Create a DataFrame `df_logs` from the parsed log entries.

Labeling the Data

1. **Define Error Keywords**:
    - Keywords: `'deny', 'fail', 'error', 'denied'`
2. **Apply Labels**:
    - Label as `1` if any error keyword is found in the message (case insensitive), else `0`.

Handling Imbalanced Data

1. **Separate Majority and Minority Classes**:
    - Majority: `df_majority`
    - Minority: `df_minority`
2. **Upsample Minority Class** (if not empty):
    - Resample minority class to match the size of the majority class.
    - Combine resampled minority class with majority class to form `df_upsampled`.
3. **Display Class Counts**:
    - Print the count of each class.

Tokenizing

1. **Initialize Tokenizer**:
    - Use `BertTokenizer` with `'bert-base-uncased'`.
2. **Tokenize Log Messages**:
    - Apply tokenizer to each log message, storing the result in `df_upsampled['tokens']`.

Dataset and DataLoader

Define Custom Dataset

- **Class `LogDataset`**:
    - `__init__(self, df)`
    - `__len__(self)`
    - `__getitem__(self, idx)`

Define Collate Function

- **Function `collate_fn(batch)`**:
    - Pads sequences in the batch and stacks labels.

Create Dataset and DataLoader

1. **Create Dataset**:
    - Instantiate `LogDataset` with `df_upsampled`.
2. **Create DataLoader**:
    - Use DataLoader with batch size of 16, shuffle enabled, and custom collate function.

Model Initialization and Training

Initialize the Model

- **Model**:
    - `BertForSequenceClassification` with `'bert-base-uncased'`.

Define Optimizer and Scheduler

- **Optimizer**:
    - `AdamW` with learning rate `2e-5`.
- **Scheduler**:
    - `get_linear_schedule_with_warmup` with 0 warmup steps and training steps equal to `len(dataloader) * 3`.

Training Loop

1. **Training Settings**:
    - Number of epochs: 3
2. **Training Process**:
    - For each epoch, iterate over batches in the DataLoader.
    - Zero gradients, perform forward pass, compute loss, backward pass, and step the optimizer and scheduler.

    - Print completion message after each epoch.

Model Evaluation

Assuming a Separate Validation Set

- **Validation Dataset and DataLoader**:
    - Same dataset and DataLoader settings as for training, but without shuffling.

Evaluation Process

1. **Set Model to Evaluation Mode**:
    - `model.eval()`
2. **Collect Predictions and True Labels**:
    - For each batch in the validation DataLoader, perform forward pass without gradient computation, collect predictions, and true labels.
3. **Compute Metrics**:
    - Accuracy, precision, recall, and F1 score.

Display Evaluation Metrics

- Print accuracy, precision, recall, and F1 score.

Save the Trained Model and Tokenizer

Save Paths

- **Model Save Path**: `'trained_model_from_SCRATCH'`
- Save model and tokenizer configurations and vocab files to the specified path.