**Data Preprocessing and Training for Toxic Comment Detection**

**1. Data Preparation**

**Loading and Preprocessing Data**

* The dataset is loaded from a CSV file (final\_labels.csv).
* If toxicity\_level is missing but level\_1 exists, it is renamed accordingly.
* Only the relevant columns (body, toxicity\_level, split) are retained, and missing values are dropped.
* Labels are encoded using LabelEncoder to convert toxicity\_level categories into numerical values.
* Available labels are verified, ensuring that Misogynistic and Nonmisogynistic labels are correctly identified.
* The dataset is split into training (train\_df) and testing (test\_df) based on the split column.

**2. Custom Dataset Class**

* A PyTorch Dataset class (ToxicityDataset) is defined for handling text data.
* The dataset is tokenized using AutoTokenizer from the transformers library with a maximum length of 128 tokens.
* Each sample includes:
  + input\_ids: Tokenized input text.
  + attention\_mask: Mask to handle padding.
  + labels: Encoded target labels.

**3. Model and Tokenizer Setup**

* The roberta-base model is selected for fine-tuning.
* The tokenizer corresponding to the model is loaded.
* The classification model is initialized with two output labels (num\_labels=2).
* The training and testing datasets are converted into ToxicityDataset objects.

**4. Metrics Calculation**

* Evaluation metrics include:
  + **Accuracy**: Overall model performance.
  + **Precision, Recall, and F1-score**: Classification performance.
  + **AUC-ROC & AUC-PR**: Measures for assessing model confidence in distinguishing classes.
  + **Confusion Matrix** components: Used to calculate False Positive Rate (FPR) and False Negative Rate (FNR).
* Predictions are converted to class labels using argmax.
* Softmax is applied to logits for probability-based metrics.

**5. Training Arguments and Model Training**

* Training arguments are defined using TrainingArguments, including:
  + Evaluation and saving after each epoch.
  + A batch size of 8 for training and evaluation.
  + 3 epochs of training.
  + Weight decay regularization.
  + Logging every 10 steps.
  + Loading the best model based on F1-score.
* A Trainer object is instantiated for training and evaluation.
* The model is fine-tuned using trainer.train().

**6. Model Evaluation**

* The model is evaluated on the test dataset using trainer.predict().
* Predictions are stored in test\_df.
* Toxic comments are identified based on the predicted label.

**7. Detecting Toxic Comments**

* The target label (Misogynistic) is identified in the label encoder.
* Comments classified as toxic are extracted and displayed.

**8. Generating Alternative Phrasings**

* A text-to-text generation model (t5-base) is used to rephrase toxic comments into respectful alternatives.
* The function generate\_alternative() constructs a polite version of each toxic comment.
* Rewritten comments are printed alongside the original ones.

**Conclusion**

This workflow enables:

* Preprocessing of text-based toxicity classification datasets.
* Fine-tuning a RoBERTa model for classification.
* Performance evaluation using multiple metrics.
* Extraction of toxic comments.
* Rewriting toxic comments into constructive alternatives.