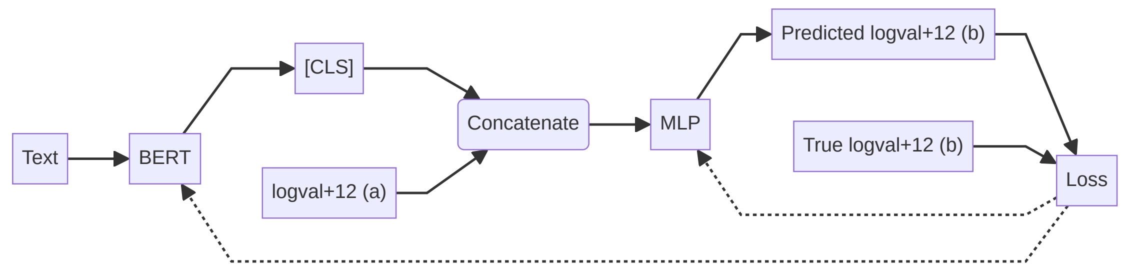
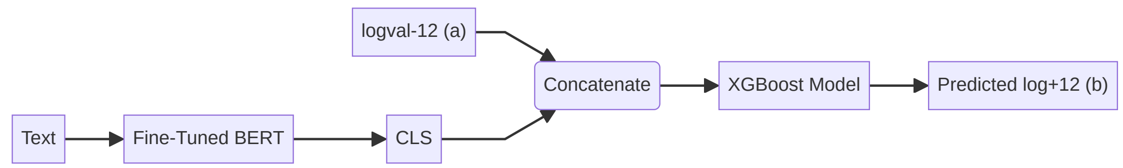
Figure 1: Fine-Tuning Stage



Remark: `solid line` stands for forward process, `dashed line` stands for backward process.

Figure 2: Prediction Stage with XGBoost



Architecture Description

We propose a two-stage architecture for the regression task of predicting a one-dimensional vector (`logval+12`) from text and a one-dimensional vector (`logval-12`) as input features.

Stage 1: Fine-Tuning BERT with MLP

In the first stage, we focus on fine-tuning a BERT model to learn task-specific representations. The process is outlined as follows:

1. **Input:**
    - **Text:** A sequence of tokens.
    - **Vector (`logval-12` , written as a below):** A one-dimensional vector.
2. **output:**
    - **Vector (`logval+12` , written as b below):** A one-dimensional vector.
3. **Processing:**

- The text is processed by a pre-trained BERT model to extract the hidden state corresponding to the `[CLS]` token, denoted as ($\mathbf{h}_{[\text{CLS}]}$).
- The one-dimensional vector (a) is concatenated with ($\mathbf{h}_{[\text{CLS}]}$) to form a combined feature vector ($[\mathbf{h}_{[\text{CLS}]}; a]$).
- This feature vector ($[\mathbf{h}_{[\text{CLS}]}; a]$) is fed into a multi-layer perceptron (MLP) to predict the target value ($\hat{b}$).

4. **Training:**
   - The BERT model and the MLP are fine-tuned end-to-end using a regression loss function, such as mean squared error (MSE), to minimize the difference between the predicted ($\hat{b}$) and the ground-truth (b).

This stage ensures that the BERT model adapts its representations to the specific regression task by leveraging both the text and the additional vector (`logval-12`).

Stage 2: Feature Extraction and XGBoost Regression

In the second stage, we utilize the fine-tuned BERT model as a feature extractor and employ XGBoost for the final regression task. The steps are outlined below:

1. **Feature Extraction:**
   - The fine-tuned BERT model from Stage 1 processes the input text to extract the `[CLS]` token's hidden state, ($\mathbf{h}_{[\text{CLS}]}$).
   - This ($\mathbf{h}_{[\text{CLS}]}$) is concatenated with the input vector (a) to produce the feature vector ($[\mathbf{h}_{[\text{CLS}]}; a]$).
2. **XGBoost Training:**
   - The feature vector ($[\mathbf{h}_{[\text{CLS}]}; a]$) serves as input to train an XGBoost regression model, which predicts the target ($\hat{b}$).
   - The XGBoost model is trained on the same dataset used for fine-tuning or a specific subset, depending on the application's requirements.

3. **Prediction:**
    - For a new input pair (text and (`logval-12`)), the fine-tuned BERT model computes the feature vector ($[\mathbf{h}_{[\mathrm{CLS}]}; a]$).
    - The trained XGBoost model then uses ($[\mathbf{h}_{[\mathrm{CLS}]}; a]$) to predict ($\hat{b}$).

This two-stage approach combines BERT's powerful representation learning, fine-tuned for the specific task, with XGBoost's efficiency and potential interpretability, offering a hybrid solution that leverages the strengths of both deep learning and gradient boosting techniques.