BERT from Scratch: 6-Labor Project Plan

Labor 1: Foundations and Data Acquisition

- Understand the two core pre-training tasks of BERT: **Masked Language Model (MLM)** and **Next Sentence Prediction (NSP)**.
- Set up the project environment and download the **IMDB movie review dataset**.

Labor 2: Data Preparation - Vocabulary and Tokenization

- Implement the PyTorch `Dataset` class.
- Split the raw text reviews into individual sentences.
- Build a word **vocabulary** from the sentences, including special tokens like `[CLS]`, `[SEP]`, `[MASK]`, and `[PAD]`.

Labor 3: Data Preparation - Creating MLM and NSP Training Items

- Implement the logic to create true and false **Next Sentence Prediction (NSP)** pairs.
- Implement the **Masked Language Model (MLM)** task by randomly masking 15% of the tokens in the input sequences.
- Combine these elements, **pad sequences** to a uniform length, and prepare the final training items for the model.

Labor 4: Model Architecture - Input Embeddings

- Build the `JointEmbedding` module.

- Implement the three types of embeddings required by BERT:
    - **Token Embeddings:** To represent the words in the vocabulary.
    - **Segment Embeddings:** To distinguish between the first and second sentences.
    - **Positional Embeddings:** To give the model information about the position of each word.

Labor 5: Model Architecture - The Transformer Encoder

- Build the core **Transformer encoder block**.
- This includes implementing the **multi-head self-attention** mechanism and the **position-wise feed-forward** network.
- Stack these encoder blocks to build the full BERT model.

Labor 6: Trainer, Loss Functions, and Execution

- Build the `BertTrainer` class to handle the training loop.
- Set up the `DataLoader` to feed data to the model in batches.
- Define two separate **loss functions**: one for the MLM task (e.g., `NLLLoss`) and one for the NSP task (e.g., `BCEWithLogitsLoss`).
- Implement the training step, including forward pass, **combined loss calculation**, backward pass, and optimizer step.
- Add helper functions to calculate the accuracy for both MLM and NSP tasks.