Name: Suhaila Ahmed Hassan
Track: AI
Branch: Alexandria

BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

BERT

BERT's framework consists of pre-training and fine-tuning.

A. During pre-training, the model is trained on unlabeled data through specific tasks.
B. In fine-tuning, the model is initialized with pre-trained parameters and fine-tuned on labeled data for each downstream task. Each task has its own fine-tuned model.

A key feature of BERT is its unified architecture across tasks, with minimal differences between the pre-trained and fine-tuned models.

Model Architecture

BERT uses a multi-layer bidirectional Transformer encoder.
The two models architecture are:

- BERTBASE: L=12, H=768, A=12, 110M parameters.
- BERTLARGE: L=24, H=1024, A=16, 340M parameters.

L: Layers, H: Hidden Size, A: Attention Heads.

Input/Output Representations

To support various tasks, BERT represents both single and paired sentences in one token sequence. Sentences are separated by a [SEP] token and identified with segment embeddings. The first token is always [CLS], whose final hidden state is used for classification tasks. The embeddings are a sum of token, segment, and position embeddings.

1. Pre-training BERT

BERT is pre-trained using two unsupervised tasks:

Task #1: Masked Language Modeling (MLM)

BERT randomly masks 15% of WordPiece tokens and predicts them.
To avoid a mismatch during fine-tuning, masked tokens are:

- Replaced with [MASK] 80% of the time
- Replaced with a random token 10% of the time
- Left unchanged 10% of the time

Task #2: Next Sentence Prediction (NSP)

To model sentence relationships, 50% of the time sentence B follows sentence A (label: IsNext), and 50% of the time it is a random sentence (label: NotNext).

Pre-training data:

BooksCorpus (800M words) and English Wikipedia (2,500M words), excluding lists, tables, and headers. Document-level corpora are essential for long sequences.

2. Fine-tuning BERT

For tasks involving text pairs, BERT uses self-attention over concatenated sequences, effectively modeling cross attention.

Each task uses task-specific inputs and outputs:

- Token representations for output layers.
- [CLS] representation for classification layers.