NANYANG
THE INNOVATIVE POLYTECHNIC

NLP Models (Part 6)

Learning Outcomes

1. Describe the fundamental structure of Bidirectional Encoder Representations from Transformers (BERT).

2. Explain the two methods in training BERT.

3. Compare and contrast between BERT and the original transformer.

4. Apply BERT to classify sentiment.

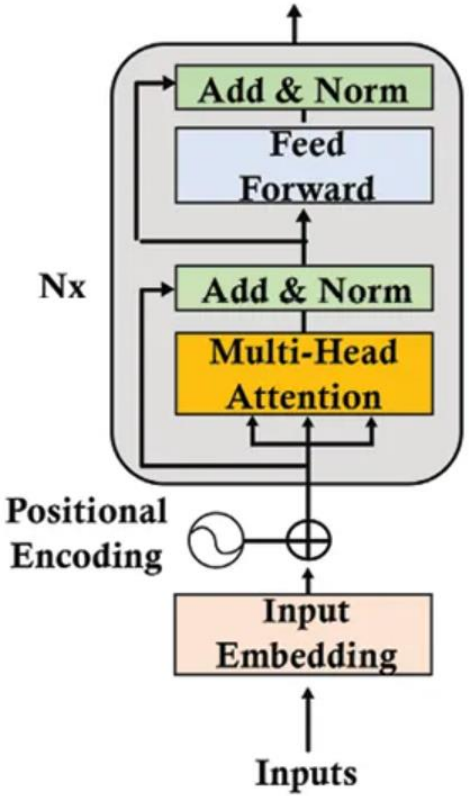
BERT Transformer

- BERT model has an encoder but no decoder stack. The masked tokens are in the attention layers.

- The original Transformer contains:
  - N = 6 layers
  - model dimension, d_{model} = 512
  - number of attention heads, A = 8
  - head dimension, d_k = 64

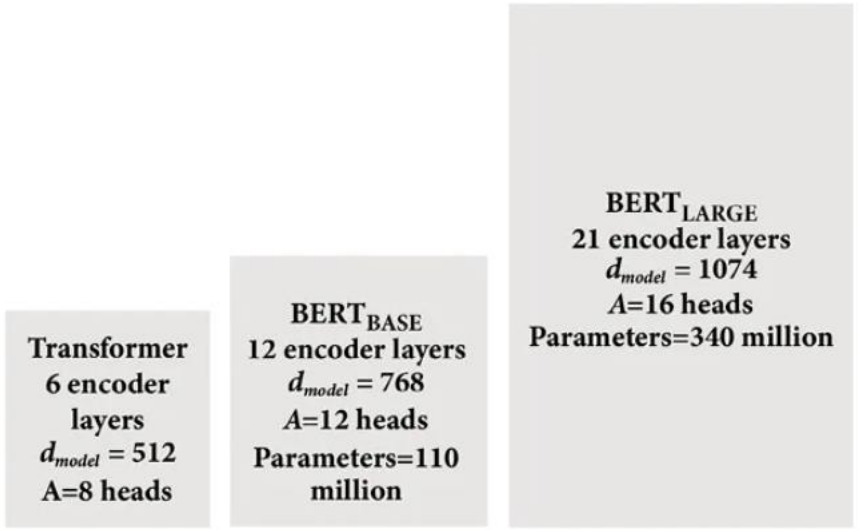
- BERT encoder layers are larger than the original Transformer model.

BERT Transformer



BERT Transformer

- Two BERT models:
  - $BERT_{BASE}$ contains N = 12 encoder layers, d_{model} = 768 and A = 12 heads.
  - $BERT_{LARGE}$ contains N = 24 encoder layers, d_{model} = 1024 and A = 16 heads.



BERT Transformer

- BERT has no decoder stack and therefore no masked multi-head attention sublayer.
- The BERT developers state that a masked multi-head attention hinders the attention process.
- In BERT, the attention heads attend to both left and right context for every token.
- A masked multi-head attention masks all tokens beyond the current position. This iis to inhibit the model from attending to the target output. However, the learning potential of the model is limited. To know what “it” is referring to in the example, the model has to attend to the whole sentence.
    - A bear eats honey because it <MASK> <MASK> <MASK>

BERT Transformer

- The model was trained with two tasks:
  - Masked Language Modeling (MLM)
  - Next Sentence Prediction (NSP)

BERT Transformer

- MLM does not train a model using a sequence of words followed by a masked sequence. This approach involves randomly masking certain words in the input sequence and training the model to predict the masked words based on the surrounding context.

    - A bear eats honey <MASK> it is delicious.

- BERT uses WordPiece, a subword tokenization method, and learn positional information, instead of using the sine-cosine approach.

BERT Transformer

- The input tokens were masked in various ways to force the model to train longer but achieve better results using three methods:
    - Not masking a token on 10% of the dataset - A bear eats honey because it is delicious.
    - Replacing a token with a random token on 10% of the dataset - A bear eats honey often it is delicious.
    - Replacing a token with a masked token on 80% of the dataset - A bear eats honey <MASK> it is delicious.

BERT Transformer

- The second method for training BERT is Next Sentence Prediction (NSP). The input consists of two sentences. In 50% of the cases, the second sentence is the actual second sentence of a document. In 50% of the cases, the second sentence was selected at random.

- Two new tokens:

  - [CLS] is a binary classification token added at the beginning of the first sequence to predict whether the second sequence follows the first sequence. A positive sample is a pair of consecutive sequences from a document. A negative sample is created from sequences from different documents.
  - [SEP] is a separation token that represents the end of a sequence.

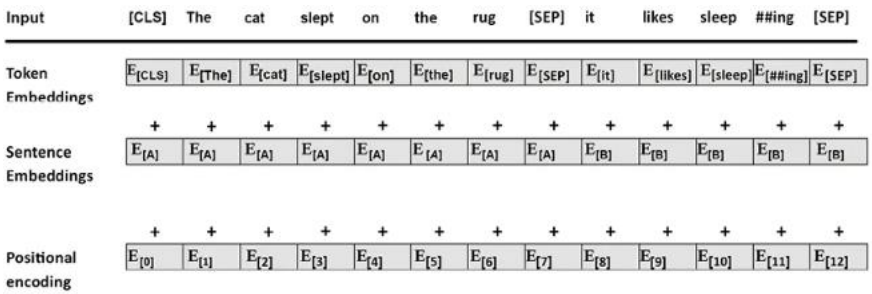
BERT Transformer

```
The cat slept on the rug. It likes sleeping
all day.
```

```
[CLS] the cat slept on the rug [SEP] it likes
sleep ##ing all day[SEP]
```

BERT Transformer

- This approach requires additional encoding information to distinguish sequence A from sequence B.



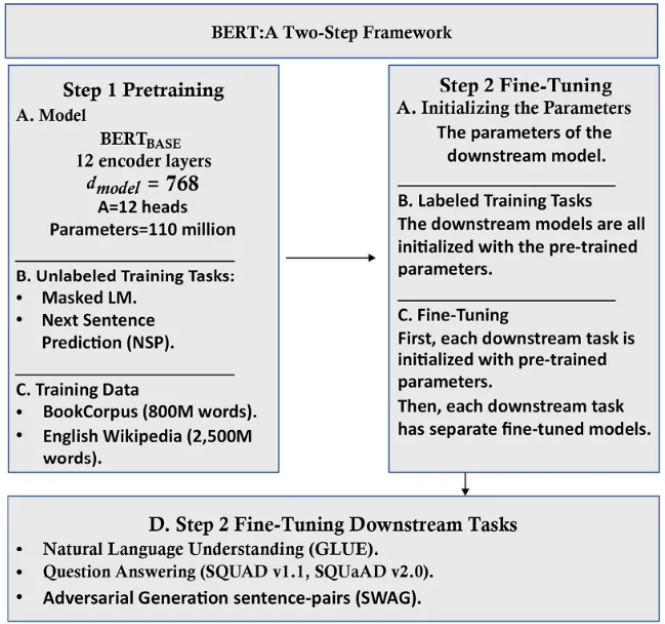
- The input embedding is the sum of the token embedding, the segment embedding and the positional encoding embedding.

BERT Transformer

- A sequence of words is broken down into WordPiece tokens.
- A [MASK] token randomly replaces a word token for training.
- A [CLS] token is inserted at the beginning of a sequence for classification purposes.
- A [SEP] token separates two sequences for training.
- Sentence embedding is added to token embedding.
- Positional encoding is learned.

BERT Transformer

- BERT is a two-step framework. The first step is pretraining, and the second is fine-tuning.



BERT Transformer

- Pretraining consists of two sub-steps:
  - Defining the model's architecture: number of layers, number of heads, dimensions
  - Training the model on MLM and NSP tasks

- Fine-tuning consists of two sub-steps:
  - Initializing the downstream model with the trained parameters of the pretrained BERT model
  - Fine-tuning the parameters for specific downstream tasks such as question answering and text classification