Natural Language Processing

Class 8 Supplement: BERT-based Finetuning

Adam Faulkner

October 28, 2025

BERT

- Transformer-based models like ChatGPT are *decoder-only*; but what about *encoder-only* models?
- BERT (Bidirectional Encoder Representations from Transformers)
- BERT-style models are trained via the **masked language modeling** (MLM) task: a word in the middle of context window is masked and the model must predict the masked word given the words on *both* sides(hence the term *bidirectional*

Masked-language modeling

- MLMs are encoder-only since because they produce an encoding for each input token but don’t generate text by decoding
- MLMs like BERT are not used for generation; instead they’re generally used for semantic similarity tasks and classification (via finetuning, which we’ll discuss in a bit)

Bidirectional training of MLMs

- So far we've only discussed next-word prediction, also called **causal**, transformer language models
- Instead of trying to predict the next word, an MLM learns to perform a fill-in-blank task, technically called the **cloze** task
- Instead of trying the predicting the next word given a sequence of previous words, we want predict a work given the surrounding context, e.g,

The ___ of Walden Pond is so beautifully...

Bidirectional training of MLMs

- Each token in the training corpus is used in one of three ways:
  1. It is replaced with the special vocabulary token named `[MASK]`.
  2. It is replaced with another token from the vocabulary, randomly sampled based on token unigram probabilities.
  3. It is left unchanged.
- In the original BERT paper 80% are replaced with `[MASK]`, 10% are replaced with randomly selected tokens, and the remaining 10% are left unchanged.

Bidirectional training of MLMs

- The MLM tries to predict the original inputs for each of the masked tokens using a bidirectional encoder (see below)
- As with the models we've discussed, the cross-entropy loss from these predictions drives the training process for all the parameters in the model

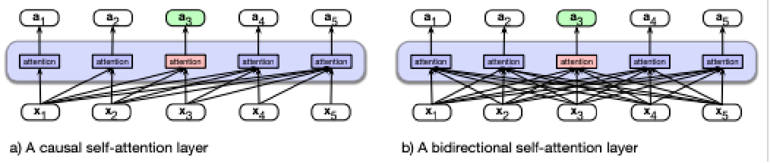
a1
a2
a3
a4
a5
attention
x1
x2
x3
x4
x5
a) A causal self-attention layer
b) A bidirectional self-attention layer

Figure 1: In the causal model (a) the attention computation for token 3 only used only information seen earlier in the context; in the bidirectional model (b) the model attends to all inputs on both left and right.

Bidirectional training of MLMs

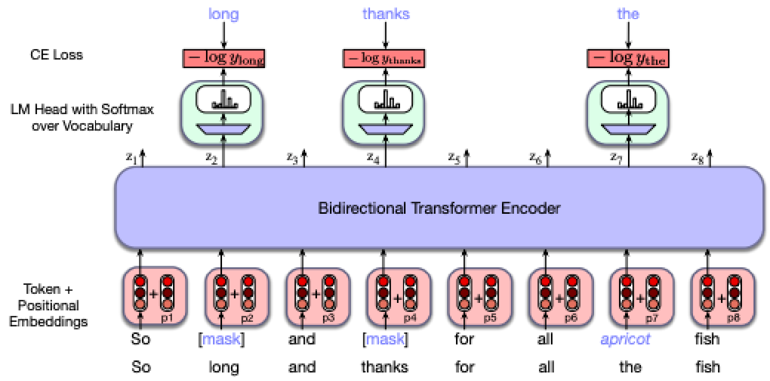
long
thanks
the
CE Loss
$-\log y_{\text{long}}$
$-\log y_{\text{thanks}}$
$-\log y_{\text{the}}$
LM Head with Softmax over Vocabulary
z_1 z_2 z_3 z_4 z_5 z_6 z_7 z_8
Bidirectional Transformer Encoder
Token + Positional Embeddings
p1 p2 p3 p4 p5 p6 p7 p8
So [mask] and [mask] for all apricot fish
So long and thanks for all the fish

Figure 2: Example of token replacement and bidirectional encoding: *long* and *thanks* are masked and *the* is replaced with randomly selected token *apricot*

Next sentence prediction

- MLM has the goal of producing effective word-level representations
- But many tasks in NLP involve the relationship between pairs of sentences.
- To learn the kind of knowledge required for these BERT includes a *second* learning objective: Next Sentence Prediction (NSP)

Next sentence prediction

- Several classic NLP tasks can be solved by applying the the sentence-level representations output by the NSP task
- **Paraphrase Detection**:

  Does *The company's stock dropped* = *The organization's stock slumped*?

- **Logical entailment**:

  Does *Fido is a mammal* logically entail *Fido is a dog*?

- **Discourse coherence**:

  Is the ordering of these two sentences coherent?

  Once upon a time there was a girl named Goldilocks

  2 + 2 is 4

Next sentence prediction

- NSP: Given a pair of sentences, predict whether each pair consists of an actual pair of adjacent sentences from the training corpus or a pair of unrelated sentences.
- The NSP loss is based on how well the model can distinguish true pairs from random pairs. In BERT, 50% of the training pairs consisted of positive pairs and in the other 50% the second sentence of a pair was randomly selected from elsewhere in the corpus.
- BERT introduces two special tokens to the input representation
  - The token `[CLS]` is prepended to the input sentence pair
  - The token `[SEP]` is placed between the sentences and after the final token of the second sentence
- During training, the output vector h^L_{CLS} from the final layer associated with the `[CLS]` token represents the next sentence prediction: *isNext* or *notNext*

Bidirectional training of MLMs

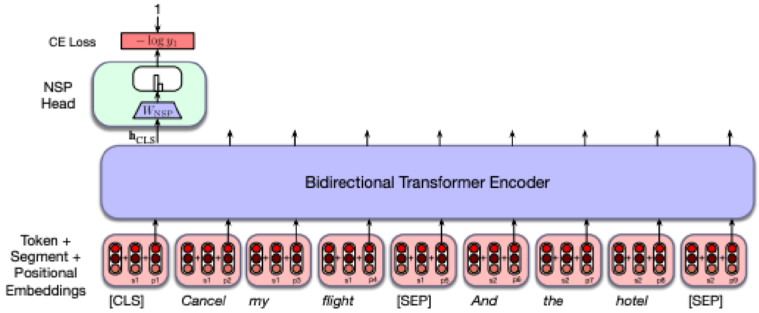

Figure 3: During training, the output vector h^L_{CLS} from the final layer associated with the `[CLS]` token represents the next sentence prediction: *isNext* or *notNext*

Finetuning (for BERT)

- The original excitement generated by BERT lay in its enablement of **transfer learning** via a technique called **finetuning**
- Rather then training task-specific models, a single model (BERT) can be used to solve multiple downstream tasks sequence classification, sentence-pair classification, and sequence labeling

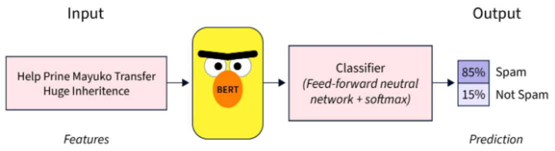
Input
Help Prine Mayuko Transfer Huge Inheritence
Features
BERT
Classifier (Feed-forward neutral network + softmax)
Output
85% Spam
15% Not Spam
Prediction

Figure 4: BERT can be finetuned for many different tasks, including spam detection.

Finetuning (for BERT)

- BERT-based classification is made possible with the addition of a unique token to the vocabulary called `[CLS]` which is prepended to the start of all input sequences, both during pretraining and encoding.
- The output vector in the final layer of the model for the `[CLS]` input represents the entire input sequence and serves as the input to a **classifier head**, a logistic regression or neural network classifier that makes the relevant decision.

Finetuning (for BERT)

Suppose we have a three-class (*positive, negative, neutral*) classification task and labeled data for this task, e.g. a dataset of tagged product reviews. Finetuning BERT on this task involves the following steps

1. Represent the first 512 tokens (BERT's token limit) of each product review as single vector by prepending `[CLS]` (the same token used in BERT's pretraining) to the start of the input
2. The output vector in the final layer of the model for the `[CLS]` input represents the entire input sequence and serves as the input to a classifier head, a logistic regression or neural network classifier that makes the relevant decision
3. Finetuning this classification head involves learning a set of weights W_c to map the output vector for the `[CLS]` token to a set of scores over the possible sentiment classes
4. To classify a review, we pass the input text through BERT to generate `[CLS]` multiply it by W_c and pass the resulting vector through a softmax.

Sequence Modeling Tasks

- Sequence modeling involves assigning a label or prediction to each element (token) in an input sequence.
- Unlike Sequence Classification (which assigns one label per sequence), this is a Token Classification task.

Sequence Modeling Tasks

- Sequence modeling involves assigning a label or prediction to each element (token) in an input sequence.
- Unlike Sequence Classification (which assigns one label per sequence), this is a Token Classification task.
- **Goal:** Identify and classify named entities (Person, Location, Organization, etc.) in text.
- **Input:** A sentence.
- **Output:** A label for *every word/token*.

Sequence Modeling Tasks

- Sequence modeling involves assigning a label or prediction to each element (token) in an input sequence.
- Unlike Sequence Classification (which assigns one label per sequence), this is a Token Classification task.
- **Goal:** Identify and classify named entities (Person, Location, Organization, etc.) in text.
- **Input:** A sentence.
- **Output:** A label for *every word/token*.

Token	John	lives	in	New	York	City	.
Label	B-PER	O	O	B-LOC	I-LOC	I-LOC	O

tableIOB Tagging Scheme for NER

- **B-**: Beginning of an entity.
- **I-**: Inside/Continuation of an entity.
- **O**: Outside of an entity.

BERT Fine-Tuning for Token Classification (NER)

- The pre-trained BERT model (the encoder stack) is used as a fixed or semi-fixed feature extractor.
- A small, task-specific layer is added on top.

BERT Fine-Tuning for Token Classification (NER)

- The pre-trained BERT model (the encoder stack) is used as a fixed or semi-fixed feature extractor.
- A small, task-specific layer is added on top.

1. Input: The tokenized input sentence is passed through the BERT encoder.
2. BERT Output: The hidden state $\mathbf{h}_i$ for *every input token* t_i is used (not just the `[CLS]` token).
3. Classification Head: A linear layer (dense layer) is added on top of the hidden states.
4. Prediction: The linear layer maps the hidden state $\mathbf{h}_i$ to a probability distribution over the **N** possible NER tags (e.g., B-PER, I-LOC, O).

BERT Fine-Tuning for Token Classification (NER)

- The pre-trained BERT model (the encoder stack) is used as a fixed or semi-fixed feature extractor.
- A small, task-specific layer is added on top.

1. Input: The tokenized input sentence is passed through the BERT encoder.
2. BERT Output: The hidden state $\mathbf{h}_i$ for *every input token* t_i is used (not just the `[CLS]` token).
3. Classification Head: A linear layer (dense layer) is added on top of the hidden states.
4. Prediction: The linear layer maps the hidden state $\mathbf{h}_i$ to a probability distribution over the **N** possible NER tags (e.g., B-PER, I-LOC, O).

$$\text{Prediction}_i = \text{Softmax}(\mathbf{W} \cdot \mathbf{h}_i + \mathbf{b})$$

The Fine-Tuning Procedure

The goal is to update the weights of **both** BERT's encoder and the new classification head using a labeled dataset.

1. **Data Preparation:**
    - Map text and IOB labels to numerical IDs.
    - Tokenization Challenges: BERT uses Subword Tokenization (e.g., 'Washington' → 'Washing', '##ton'). This requires careful alignment of original word labels to new subword tokens.

The Fine-Tuning Procedure

The goal is to update the weights of **both** BERT's encoder and the new classification head using a labeled dataset.

1. **Data Preparation:**
    - Map text and IOB labels to numerical IDs.
    - Tokenization Challenges: BERT uses Subword Tokenization (e.g., 'Washington' → 'Washing', '##ton'). This requires careful alignment of original word labels to new subword tokens.
2. **Training Objective:**
    - Use a standard Cross-Entropy Loss for token-level classification.

The Fine-Tuning Procedure

The goal is to update the weights of **both** BERT's encoder and the new classification head using a labeled dataset.

1. **Data Preparation:**
   - Map text and IOB labels to numerical IDs.
   - Tokenization Challenges: BERT uses Subword Tokenization (e.g., 'Washington' $\rightarrow$ 'Washing', '##ton'). This requires careful alignment of original word labels to new subword tokens.
2. **Training Objective:**
   - Use a standard Cross-Entropy Loss for token-level classification.
3. **Hyperparameters:**
   - Small Learning Rate ($\sim 10^{-5}$ to 5×10^{-5}): Prevents catastrophic forgetting of pre-trained knowledge.
   - Few Epochs ($\sim 2-4$): BERT is already highly optimized; too many epochs lead to overfitting.

The Fine-Tuning Procedure

The goal is to update the weights of **both** BERT's encoder and the new classification head using a labeled dataset.

1. **Data Preparation:**
    - Map text and IOB labels to numerical IDs.
    - Tokenization Challenges: BERT uses Subword Tokenization (e.g., 'Washington' → 'Washing', '##ton'). This requires careful alignment of original word labels to new subword tokens.
2. **Training Objective:**
    - Use a standard Cross-Entropy Loss for token-level classification.
3. **Hyperparameters:**
    - Small Learning Rate ($\sim 10^{-5}$ to 5×10^{-5}): Prevents catastrophic forgetting of pre-trained knowledge.
    - Few Epochs ($\sim 2-4$): BERT is already highly optimized; too many epochs lead to overfitting.
4. **Evaluation:**
    - Use sequence-level metrics, typically the **F1 Score** (Precision and Recall on entire entities, not just tokens).

Advantages and Summary

- Contextual Embeddings: Bidirectional attention provides a much richer understanding of context than older models (e.g., LSTMs or traditional word embeddings).
- Transfer Learning: Significant reduction in training time and required labeled data compared to training a model from scratch.
- State-of-the-Art Performance: BERT and its variants (RoBERTa, ELECTRA) consistently achieve superior results on various sequence labeling benchmarks.

Summary: BERT Fine-Tuning for Sequence Modeling

- A Pre-trained BERT Encoder is adapted by adding a simple Token Classification Head.
- The entire model is trained end-to-end with a low learning rate on the specific sequence task (like NER).
- This leverages BERT's general language knowledge for highly accurate, task-specific predictions.

Finetuning (for BERT): Adapting BERT to particular domains using MLM fine-tuning

- In addition to task-based finetuning using `[CLS]` we can perform another kind of finetuning that adjusts BERT's weights to reflect a particular domain: **MLM fine-tuning**
- BERT is trained on a vast corpus incorporating hundreds of different subjects. During MLM pre-training it has domain-general weights. But what if we have a specific domain (movies, sports, medicine, finance) that we want to adapt the model to? BERT's weights can be adjusted to orient toward this domain by finetuning on the domain's corpus– the learning task is the same: predict '[MASK]' using the surrounding words.