Natural Language Processing with Deep Learning

Lecture 10 – Text classification 6: BERT part two

Prof. Dr. Ivan Habernal

December 15, 2023

Natural Language Processing Group
Paderborn University
We focus on Trustworthy Human Language Technologies

www.trusthlt.org

Motivation

Last time we started with the transformer and BERT, but we

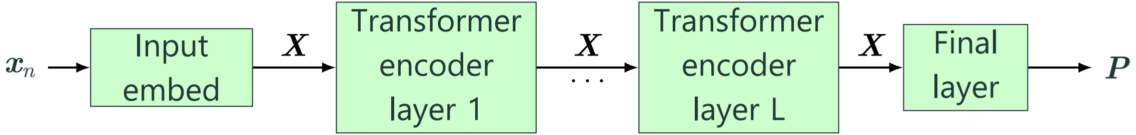
- skipped some important architectural details
- were unprecise with some graphical representation
- did not talk about pre-training and fine-tuning

Let's fix that today!

After this lecture you should be able to build BERT

BERT — Encoder architecture in detail

Transformer encoder (BERT)



As usual, green boxes are functions with trainable parameters

BERT (encoding-only transformer, forward pass)

function ETRANSFORMER($\boldsymbol{x}; \boldsymbol{\mathcal{W}}$)
    $\ldots$

Input:

$\boldsymbol{x}$ — $\boldsymbol{x} \in V^*$, a sequence of token IDs

$\boldsymbol{\mathcal{W}}$ — all trainable parameters

Output:

Typically an embedding vector for each input token

Or: $\boldsymbol{P} \in (0,1)^{\ell_{\mathsf{x}} \times N_{\mathsf{V}}}$, where each row of $\boldsymbol{P}$ is a distribution over the vocabulary

Input embeddings

The cat sat $\boldsymbol{x}_n = \begin{pmatrix} 21 & 11987 & 5438 \end{pmatrix}$

$$\begin{bmatrix} 21 & 11987 & 5438 \end{bmatrix} \xrightarrow{\boldsymbol{x}_n} \boxed{\text{Input embeddings}} \xrightarrow{\boldsymbol{X}} \begin{bmatrix} \cdots & \boldsymbol{e}_1 & \cdots \\ \cdots & \boldsymbol{e}_2 & \cdots \\ \cdots & \boldsymbol{e}_3 & \cdots \end{bmatrix}$$

BERT (encoding-only transformer, forward pass)

```
function ETRANSFORMER(x; W)
    ℓ ← length(x)
    for t ∈ [ℓ] : e_t ← W_e[x[t], :] + W_p[t, :]        ▷ Token emb. + positional emb.
    X ← Stack row-wise[e_1, e_2, . . . e_ℓ]
    . . .
```

function $\textsc{ETransformer}(\boldsymbol{x}; \boldsymbol{\mathcal{W}})$

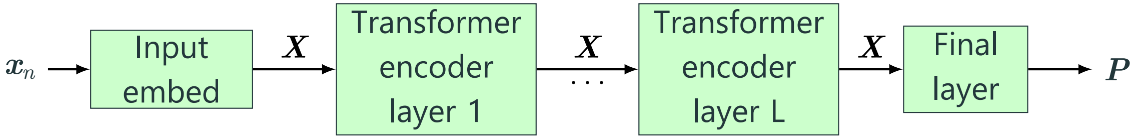
$\ell \leftarrow \text{length}(\boldsymbol{x})$

for $t \in [\ell] : \boldsymbol{e}_t \leftarrow \boldsymbol{W_e}[x[t], :] + \boldsymbol{W_p}[t, :]$ ▷ Token emb. + positional emb.

$\boldsymbol{X} \leftarrow$ Stack row-wise$[\boldsymbol{e}_1, \boldsymbol{e}_2, \ldots \boldsymbol{e}_\ell]$

$\ldots$

Transformer encoder (BERT)



The transformer encoder layer is repeated L-times (each with different parameters)

BERT (encoding-only transformer, forward pass)

function ETRANSFORMER($\boldsymbol{x}; \boldsymbol{\mathcal{W}}$)

  $\ell \leftarrow \text{length}(\boldsymbol{x})$

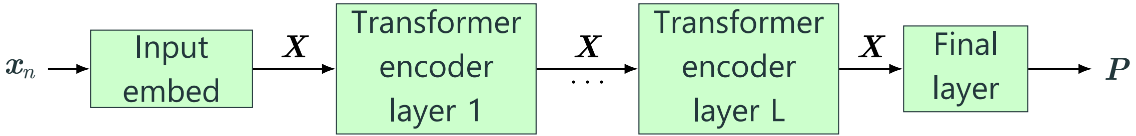
  for $t \in [\ell] : \boldsymbol{e}_t \leftarrow \boldsymbol{W_e}[x[t], :] + \boldsymbol{W_p}[t, :]$ ▷ Token emb. + positional emb.

  $\boldsymbol{X} \leftarrow \text{Stack row-wise}[\boldsymbol{e}_1, \boldsymbol{e}_2, \ldots \boldsymbol{e}_\ell]$

  for $l = 1, 2, \ldots, L$ **do**

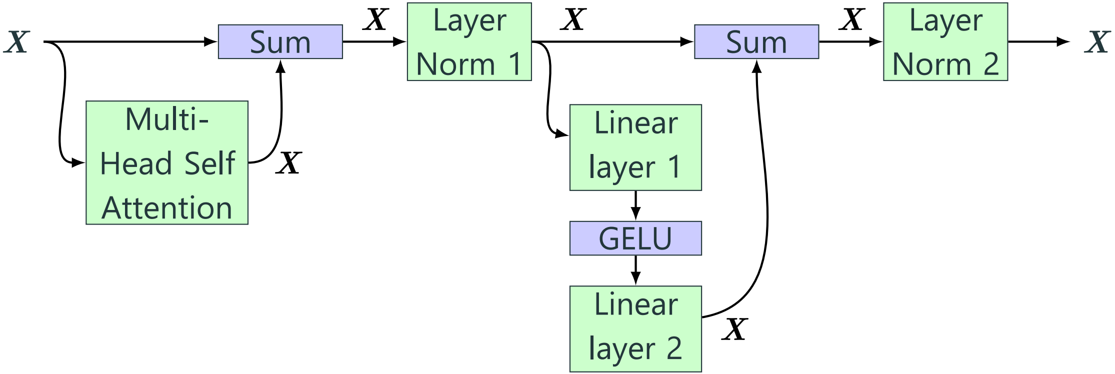
    $\ldots$

Transformer encoder (BERT)



Let's look at a single transformer encoder layer

Transformer encoder layer (BERT)



Let's focus on Multi-Head Self Attention

BERT (encoding-only transformer, forward pass)

function ETRANSFORMER($\boldsymbol{x}; \mathcal{W}$)

  $\ell \leftarrow \text{length}(\boldsymbol{x})$

  for $t \in [\ell]$: $\boldsymbol{e}_t \leftarrow \boldsymbol{W}_{\boldsymbol{e}}[x[t], :] + \boldsymbol{W}_{\boldsymbol{p}}[t, :]$ ▷ Token emb. + positional emb.

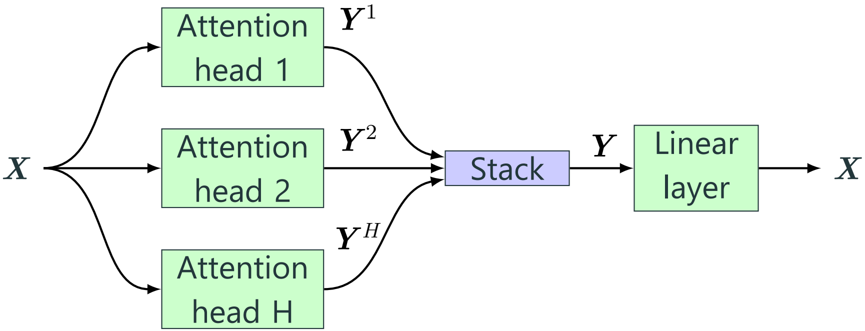
  $\boldsymbol{X} \leftarrow \text{Stack row-wise}[\boldsymbol{e}_1, \boldsymbol{e}_2, \ldots \boldsymbol{e}_\ell]$

  for $l = 1, 2, \ldots, L$ **do**

    $\boldsymbol{X} \leftarrow \boldsymbol{X} + \text{MHATTENTION}(\boldsymbol{X} | \mathcal{W}_l)$ ▷ Multi-head att., residual conn

    $\ldots$

Multi-head unmasked self-attention (BERT)



Multi-head bidirectional / unmasked self-attention

Input: $\boldsymbol{X} \in \mathbb{R}^{\ell_\mathsf{x} \times d_\mathsf{x}}$, vector representations of the sequence of length ℓ_x

Output: $\tilde{\boldsymbol{V}} \in \mathbb{R}^{\ell_\mathsf{x} \times d_\mathsf{out}}$, updated vector representations of tokens in $\boldsymbol{X}$

Hyper-param: H, number of attention heads

Params for each $h \in [H]$: $\boldsymbol{\mathcal{W}}^h_{\boldsymbol{qkv}}$:

- $\boldsymbol{W}^h_{\boldsymbol{q}}, \boldsymbol{W}^h_{\boldsymbol{k}} \in \mathbb{R}^{d_\mathsf{x} \times d_\mathsf{attn}}$, $\boldsymbol{b}^h_{\boldsymbol{q}}, \boldsymbol{b}^h_{\boldsymbol{k}} \in \mathbb{R}^{d_\mathsf{attn}}$, $\boldsymbol{W}_{\boldsymbol{v}} \in \mathbb{R}^{d_\mathsf{x} \times d_\mathsf{mid}}$, $\boldsymbol{b}_{\boldsymbol{v}} \in \mathbb{R}^{d_\mathsf{mid}}$
- $\boldsymbol{W}_{\boldsymbol{o}} \in \mathbb{R}^{H \cdot d_\mathsf{mid} \times d_\mathsf{out}}$, $\boldsymbol{b}_{\boldsymbol{o}} \in \mathbb{R}^{d_\mathsf{out}}$

```
function MHAttention(X; W)
    for h ∈ [H] do
        Y^h ← Attention(X; W^h_qkv)                ▷ Y^h ∈ R^{ℓx × dmid}
    Y ← [Y^1; Y^2; ...; Y^H]                       ▷ Y ∈ R^{ℓx × H·dmid}
    return Ṽ = Y W_o + b_o
```

function $\text{MHAttention}(\boldsymbol{X}; \boldsymbol{\mathcal{W}})$

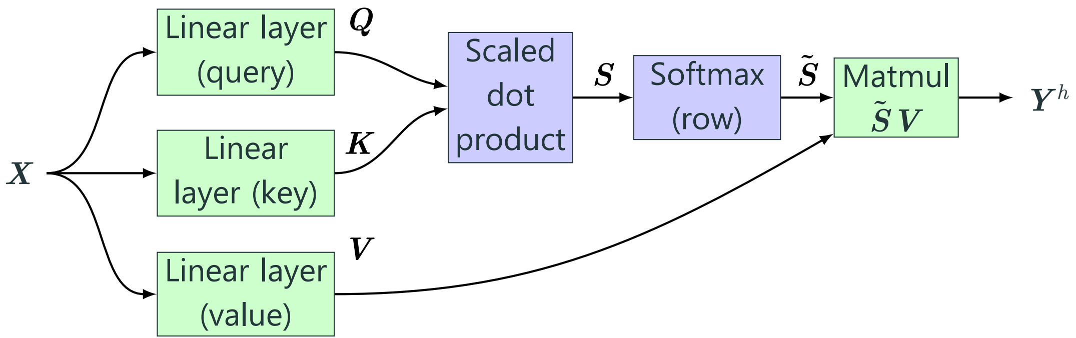
  for $h \in [H]$ **do**

    $\boldsymbol{Y}^h \leftarrow \text{Attention}(\boldsymbol{X}; \boldsymbol{\mathcal{W}}^h_{\boldsymbol{qkv}})$ $\triangleright\ \boldsymbol{Y}^h \in \mathbb{R}^{\ell_\mathsf{x} \times d_\mathsf{mid}}$

  $\boldsymbol{Y} \leftarrow [\boldsymbol{Y}^1; \boldsymbol{Y}^2; \ldots; \boldsymbol{Y}^H]$ $\triangleright\ \boldsymbol{Y} \in \mathbb{R}^{\ell_\mathsf{x} \times H \cdot d_\mathsf{mid}}$

  return $\tilde{\boldsymbol{V}} = \boldsymbol{Y}\boldsymbol{W}_{\boldsymbol{o}} + \boldsymbol{b}_{\boldsymbol{o}}$

Single unmasked self-attention head (BERT)



Bidirectional / unmasked self-attention (recap from last lecture)

Input: $\boldsymbol{X} \in \mathbb{R}^{\ell_\mathsf{x} \times d_\mathsf{x}}$, vector representations of the sequence of length ℓ_x

Output: $\tilde{\boldsymbol{V}} \in \mathbb{R}^{\ell_\mathsf{x} \times d_\mathsf{out}}$, updated vector representations of tokens in $\boldsymbol{X}$

Params $\boldsymbol{\mathcal{W}}_{\boldsymbol{qkv}}$: $\boldsymbol{W_q}, \boldsymbol{W_k} \in \mathbb{R}^{d_\mathsf{x} \times d_\mathsf{attn}}$, $\boldsymbol{b_q}, \boldsymbol{b_k} \in \mathbb{R}^{d_\mathsf{attn}}$, $\boldsymbol{W_v} \in \mathbb{R}^{d_\mathsf{x} \times d_\mathsf{out}}$, $\boldsymbol{b_v} \in \mathbb{R}^{d_\mathsf{out}}$

function ATTENTION($\boldsymbol{X}; \boldsymbol{\mathcal{W}}_{\boldsymbol{qkv}}$)

$\boldsymbol{Q} \leftarrow \boldsymbol{X}\boldsymbol{W_q} +_{\text{(rows)}} \boldsymbol{b_q}$ ▷ Query $\in \mathbb{R}^{\ell_\mathsf{x} \times d_\mathsf{attn}}$

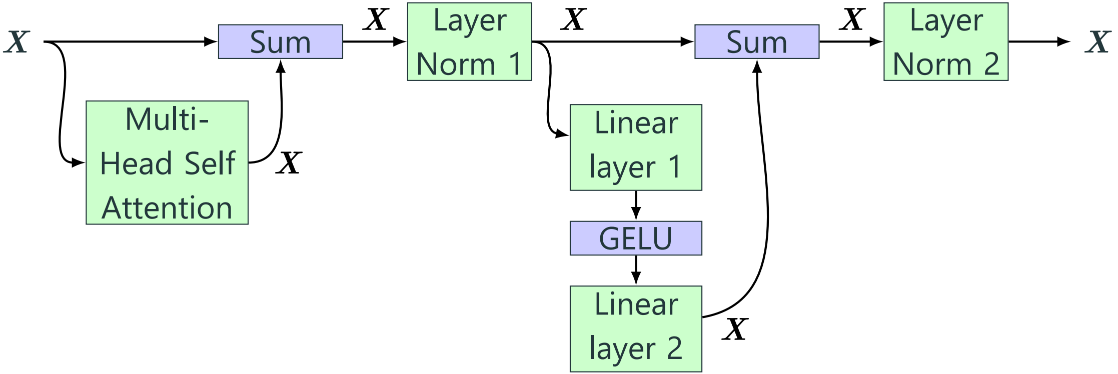
$\boldsymbol{K} \leftarrow \boldsymbol{X}\boldsymbol{W_k} +_{\text{(rows)}} \boldsymbol{b_k}$ ▷ Key $\in \mathbb{R}^{\ell_\mathsf{x} \times d_\mathsf{attn}}$

$\boldsymbol{V} \leftarrow \boldsymbol{X}\boldsymbol{W_v} +_{\text{(rows)}} \boldsymbol{b_v}$ ▷ Value $\in \mathbb{R}^{\ell_\mathsf{x} \times d_\mathsf{out}}$

$\boldsymbol{S} \leftarrow \frac{1}{\sqrt{d_\mathsf{attn}}} (\boldsymbol{Q}\boldsymbol{K}^\top)$ ▷ Scaled score $\in \mathbb{R}^{\ell_\mathsf{x} \times \ell_\mathsf{x}}$

return $\tilde{\boldsymbol{V}} = \mathrm{softmax}_{\text{row}}(\boldsymbol{S})\,\boldsymbol{V}$

Transformer encoder layer (BERT)



Let's add Layer Normalization and GELU

Simplifying notation: Perform LayerNorm on each row

Recall: LayerNorm

Input: $\boldsymbol{e} \in \mathbb{R}^d$ (output of a layer), Output: $\hat{e} \in \mathbb{R}^d$

Params: $\boldsymbol{\gamma}, \boldsymbol{\beta} \in \mathbb{R}^d$, trainable element-wise scale and offset

function LayerNorm($\boldsymbol{e} | \boldsymbol{\gamma}, \boldsymbol{\beta}$)
    $m \leftarrow \frac{1}{d} \sum_{i=1}^{d} \boldsymbol{e}[i]$ ▷ 'Sample mean' of $\boldsymbol{e}$
    $v \leftarrow \frac{1}{d} \sum_{i=1}^{d} (\boldsymbol{e}[i] - m)^2$ ▷ 'Sample variance' of $\boldsymbol{e}$
    return $\hat{e} = \frac{e-m}{\sqrt{v}} \odot \boldsymbol{\gamma} + \boldsymbol{\beta}$ ▷ Offset and scale

function LayerNormEachRow($\boldsymbol{X} \in \mathbb{R}^{m \times n} | \boldsymbol{\gamma}, \boldsymbol{\beta}$)
    for $t \in [m]$ **do**
        $\boldsymbol{X}[t, :] \leftarrow$ LayerNorm$(\boldsymbol{X}[t, :] | \boldsymbol{\gamma}, \boldsymbol{\beta})$
    return $\boldsymbol{X}$

GELU — Gaussian Error Linear Units

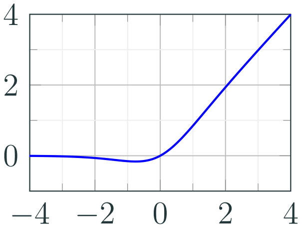
Recall: CDF $\Phi(x)$ of standard normal $X \sim \mathcal{N}(0;1)$

$$\Phi(x) = \Pr(X \leq x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} \exp\left(\frac{-t^2}{2}\right) \mathrm{d}t$$

D. Hendrycks and K. Gimpel (2016). ***Gaussian Error Linear Units (GELUs)***. arXiv: `1606.08415`

For vectors $\boldsymbol{x} \in \mathbb{R}^n$, GELU$(\boldsymbol{x})$ is applied element-wise

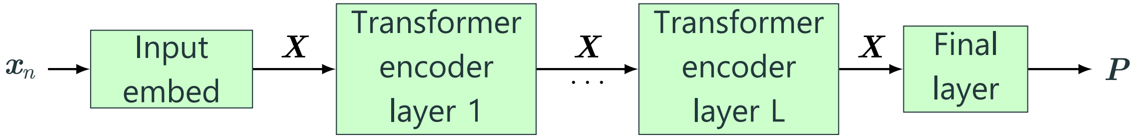
$$\begin{aligned} \text{GELU}(x) &= x \cdot \Phi(x) \\ &\approx x \cdot \sigma(1.702x) \qquad \text{(if speed > exactness)} \end{aligned}$$



BERT (encoding-only transformer, forward pass)

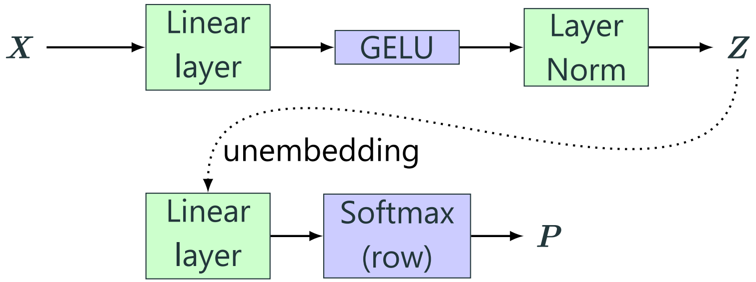
function $\text{ETRANSFORMER}(\boldsymbol{x}; \mathcal{W})$
  $\ell \leftarrow \text{length}(\boldsymbol{x})$
  for $t \in [\ell]$: $\boldsymbol{e}_t \leftarrow \boldsymbol{W_e}[x[t], :] + \boldsymbol{W_p}[t, :]$ ▷ Token emb. + positional emb.
  $\boldsymbol{X} \leftarrow \text{Stack row-wise}[\boldsymbol{e}_1, \boldsymbol{e}_2, \dots \boldsymbol{e}_\ell]$
  for $l = 1, 2, \dots, L$ **do**
    $\boldsymbol{X} \leftarrow \boldsymbol{X} + \text{MHATTENTION}(\boldsymbol{X} | \mathcal{W}_l)$ ▷ Multi-head att., residual conn
    $\boldsymbol{X} \leftarrow \text{LAYERNORMPERROW}(\boldsymbol{X} | \boldsymbol{\gamma}_l^1, \boldsymbol{\beta}_l^1)$
    $\boldsymbol{X} \leftarrow \boldsymbol{X} + \left(\text{GELU}(\boldsymbol{X}\boldsymbol{W}_l^{\text{mlp1}} +_{\text{(row)}} \boldsymbol{b}_l^{\text{mlp1}})\,\boldsymbol{W}_l^{\text{mlp2}} +_{\text{(row)}} \boldsymbol{b}_l^{\text{mlp2}}\right)$ ▷ MLP
    $\boldsymbol{X} \leftarrow \text{LAYERNORMPERROW}(\boldsymbol{X} | \boldsymbol{\gamma}_l^2, \boldsymbol{\beta}_l^2)$
  $\dots$

Transformer encoder (BERT)



Let’s look at the final layers

Final layer (BERT)



BERT (encoding-only transformer, forward pass)

function $\text{ETRANSFORMER}(\boldsymbol{x}; \boldsymbol{\mathcal{W}})$

$\ell \leftarrow \text{length}(\boldsymbol{x})$

for $t \in [\ell] : \boldsymbol{e}_t \leftarrow \boldsymbol{W_e}[x[t], :] + \boldsymbol{W_p}[t, :]$ ▷ Token emb. + positional emb.

$\boldsymbol{X} \leftarrow \text{Stack row-wise}[\boldsymbol{e}_1, \boldsymbol{e}_2, \ldots \boldsymbol{e}_\ell]$

for $l = 1, 2, \ldots, L$ **do**

$\boldsymbol{X} \leftarrow \boldsymbol{X} + \text{MHATTENTION}(\boldsymbol{X} | \boldsymbol{\mathcal{W}}_l)$ ▷ Multi-head att., residual conn

$\boldsymbol{X} \leftarrow \text{LAYERNORMPERROW}(\boldsymbol{X} | \boldsymbol{\gamma}_l^1, \boldsymbol{\beta}_l^1)$

$\boldsymbol{X} \leftarrow \boldsymbol{X} + \left(\text{GELU}(\boldsymbol{X} \boldsymbol{W}_l^{\text{mlp1}} +_{\text{(row)}} \boldsymbol{b}_l^{\text{mlp1}}) \boldsymbol{W}_l^{\text{mlp2}} +_{\text{(row)}} \boldsymbol{b}_l^{\text{mlp2}} \right)$ ▷ MLP

$\boldsymbol{X} \leftarrow \text{LAYERNORMPERROW}(\boldsymbol{X} | \boldsymbol{\gamma}_l^2, \boldsymbol{\beta}_l^2)$

$\boldsymbol{X} \leftarrow \text{GELU}(\boldsymbol{X} \boldsymbol{W_f} +_{\text{(row)}} \boldsymbol{b_f})$

$\boldsymbol{X} \leftarrow \text{LAYERNORMPERROW}(\boldsymbol{X} | \boldsymbol{\gamma}_l, \boldsymbol{\beta}_l)$

return $\boldsymbol{P} = \text{softmax}(\boldsymbol{X} \boldsymbol{W_u})$ ▷ Project to vocab., probabilities

BERT parameters and hyperparameters

Hyperparameters: $\ell_{\mathsf{max}}, L, H, d_{\mathsf{e}}, d_{\mathsf{mlp}}, d_{\mathsf{f}} \in \mathbb{N}$

Parameters:

$\boldsymbol{W_e} \in \mathbb{R}^{N_{\mathsf{V}} \times d_{\mathsf{e}}}$, $\boldsymbol{W_p} \in \mathbb{R}^{\ell_{\mathsf{max}} \times d_{\mathsf{e}}}$, the token and positional embedding matrices

For $l \in [L]$: $\boldsymbol{\mathcal{W}}_l$, multi-head attention parameters for layer l:

- $\boldsymbol{\gamma}_l^1, \boldsymbol{\beta}_l^1, \boldsymbol{\gamma}_l^2, \boldsymbol{\beta}_l^2$, two sets of layer-norm parameters
- $\boldsymbol{W}_l^{\mathsf{mlp1}} \in \mathbb{R}^{d_{\mathsf{e}} \times d_{\mathsf{mlp}}}$, $\boldsymbol{b}_l^{\mathsf{mlp1}} \in \mathbb{R}^{d_{\mathsf{mlp}}}$
- $\boldsymbol{W}_l^{\mathsf{mlp2}} \in \mathbb{R}^{d_{\mathsf{mlp}} \times d_{\mathsf{e}}}$, $\boldsymbol{b}_l^{\mathsf{mlp2}} \in \mathbb{R}^{d_{\mathsf{e}}}$

$\boldsymbol{W_f} \in \mathbb{R}^{d_{\mathsf{e}} \times d_{\mathsf{f}}}$, $\boldsymbol{b_f} \in \mathbb{R}^{d_{\mathsf{f}}}$, $\boldsymbol{\gamma}, \boldsymbol{\beta} \in \mathbb{R}^{d_{\mathsf{f}}}$, the final linear projection and layer-norm parameters.

$\boldsymbol{W_u} \in \mathbb{R}^{d_{\mathsf{e}} \times N_{\mathsf{V}}}$, the unembedding matrix

Input and pre-training

BERT — Encoder architecture in detail

Input and pre-training

Pre-training

Downstream tasks and fine-tuning

BERT: Tokenization

Tokenizing into a multilingual WordPiece inventory

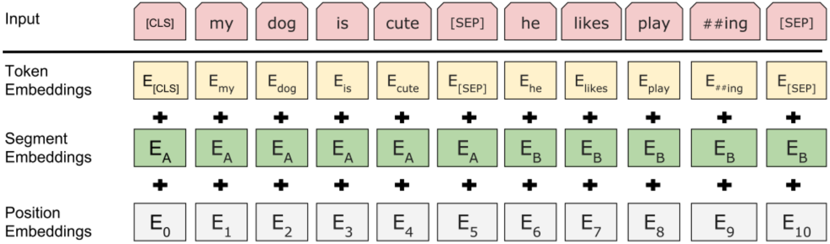
- Recall that WordPiece units are sub-word units
- 30,000 WordPiece units (newer models 110k units, 100 languages)

Implications: BERT can ”consume” any language

BERT: Input representation

- Each WordPiece token from the input is represented by a **WordPiece embedding** (randomly initialized)
- Each position from the input is associated with a **positional embedding** (also randomly initialized)
- Input length limited to **512** WordPiece tokens, using <PAD>ding
- Special tokens
    - The fist token is always a special token **[CLS]**
    - If the task involves two sentences (e.g., NLI), these two sentences are separated by a special token **[SEP]**; also special two **segment position embeddings**

BERT: Input representation summary



Pre-training

BERT — Encoder architecture in detail
Input and pre-training
Pre-training
Downstream tasks and fine-tuning

BERT: Self-supervised multi-task pre-training

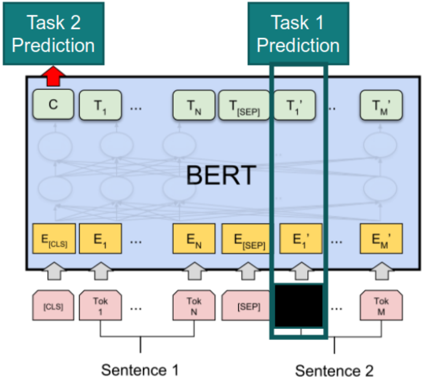
Prepare two auxiliary tasks that need no labeled data

Task 1: Cloze-test task

- Predict the masked WordPiece unit (multi-class, 30k classes)

Task 2: Consecutive segment prediction

- Did the second text segment appeared after the first segment? (binary)



BERT: Pre-training data generation

Take the entire Wikipedia (in 100 languages; 2,5 billion words)

To generate a single training instance, sample two segments (max combined length 512 WordPiece tokens)

- For Task 2, replace the second segment randomly in 50% (negative samples)
- For Task 1, choose random 15% of the tokens, and in 80% replace with a [MASK]

BERT: Pre-training data – Simplified example

Input = [CLS] the man went to [MASK] store [SEP]

he bought a gallon [MASK] milk [SEP]

Label = IsNext

Input = [CLS] the man [MASK] to the store [SEP]

penguin [MASK] are flight ##less birds [SEP]

Label = NotNext

- <PAD>ding is missing
- The actual segments are longer and not necessarily sentences (just spans)
- The WordPiece tokens match full words here

BERT: pre-training by masked language modeling

function ETRAINING($\{\boldsymbol{x}_n\}_{n=1}^{N_{\text{data}}}$ seqs, $\boldsymbol{\theta}$ init. params; $p_{\text{mask}} \in (0, 1)$, N_{epochs}, η)
    for $i \in [N_{\text{epochs}}]$ **do**
        for $n \in [N_{\text{data}}]$ **do**
            $\ell \leftarrow \text{length}(\boldsymbol{x}_n)$
            for $t \in [\ell]$ **do**
                $\tilde{\boldsymbol{x}}_n[t] \leftarrow$ `<mask_token>` with prob. p_{mask}, otherwise $\boldsymbol{x}_n[t]$
            $\tilde{T} \leftarrow \{t \in [\ell] : \tilde{\boldsymbol{x}}_n[t] =$ `<mask_token>`$\}$ ▷ Indices of masked tokens
            $\boldsymbol{P}_{\boldsymbol{\theta}} \leftarrow$ ETRANSFORMER$(\tilde{\boldsymbol{x}}_n | \boldsymbol{\theta})$
            $\text{loss}_{\boldsymbol{\theta}} \leftarrow -\sum_{t \in \tilde{T}} \log \boldsymbol{P}_{\boldsymbol{\theta}}[t, \boldsymbol{x}_n[t]]$
            $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \cdot \nabla \text{loss}_{\boldsymbol{\theta}}$
    return $\boldsymbol{\theta}$

Simple example explaining lines 6–7 (masking)

$\begin{pmatrix}\text{The} & \text{cat} & \text{sat}\end{pmatrix} \to \boldsymbol{x}_n = \begin{pmatrix}21 & 11987 & 5438\end{pmatrix}$ (Indices in V)

Random masking (index of `<mask_token>` = 50001):

1. For $t = 1$, the random outcome is "mask"
2. For $t = 2$, the random outcome is "keep"
3. For $t = 3$, the random outcome is "mask"

$$\tilde{\boldsymbol{x}}_n = \begin{pmatrix}50001 & 11987 & 50001\end{pmatrix}, \tilde{T} = \{1, 3\}$$

Explaining line 9 (negative log likelihood)

$\begin{pmatrix}\text{The} & \text{cat} & \text{sat}\end{pmatrix} \to \boldsymbol{x}_n = \begin{pmatrix}21 & 11987 & 5438\end{pmatrix}, \tilde{\boldsymbol{x}}_n = \begin{pmatrix}50001 & 11987 & 50001\end{pmatrix}, \tilde{T} = \{1, 3\}$

$\boldsymbol{P_\theta} \leftarrow \text{ETRANSFORMER}(\tilde{\boldsymbol{x}}_n | \boldsymbol{\theta})$

$$\boldsymbol{P_\theta} = \begin{pmatrix} 0.001 & 0.0007 & \dots & 0.0003 \\ 0.0013 & 0.0065 & \dots & 0.0001 \\ 0.079 & 0.015 & \dots & 0.0001 \end{pmatrix}$$

$\boldsymbol{P_\theta} \in (0, 1)^{\ell_\mathsf{x} \times N_\mathsf{V}}$, where each row of $\boldsymbol{P}$ is a distribution over the vocabulary

Explaining line 9 (negative log likelihood), $t = 1$

$\boldsymbol{x}_n = (21, 11987, 5438)$, $\tilde{\boldsymbol{x}}_n = (50001, 11987, 50001)$, $\tilde{T} = \{1, 3\}$

$$\boldsymbol{P_\theta} = \begin{pmatrix} 0.001 & \dots & 0.0041_{21} & \dots 0.0003 \\ \vdots & & & \end{pmatrix}$$

For $t = 1$, the model should learn to predict "The" (index 21)

Gold: $\boldsymbol{y} = (0, 0, \dots, 1_{21}, \dots, 0) \in \mathbb{R}^{N_\text{V}}$

Pred: $\hat{\boldsymbol{y}} = \boldsymbol{P_\theta}[1, :] = (0.001, \dots, 0.0041_{21}, \dots 0.0003) \in \mathbb{R}^{N_\text{V}}$

Categorical cross entropy loss (Lec. 4)

$L(\hat{\boldsymbol{y}}, \boldsymbol{y}) := - \sum_{k=1}^{K} \boldsymbol{y}_{[k]} \log \left(\hat{\boldsymbol{y}}_{[k]} \right)$
$= -1 \cdot \log(\hat{\boldsymbol{y}}[21]) = -\log(\boldsymbol{P_\theta}[1, 21])$
$= -\log(\boldsymbol{P_\theta}[1, \boldsymbol{x}_n[1]]) = -\log(\boldsymbol{P_\theta}[t, \boldsymbol{x}_n[t]])$

Explaining line 9 (negative log likelihood), $t = 3$

$\boldsymbol{x}_n = (21, 11987, 5438)$, $\tilde{\boldsymbol{x}}_n = (50001, 11987, 50001)$, $\tilde{T} = \{1, 3\}$

$$\boldsymbol{P_\theta} = \begin{pmatrix} \vdots & \ldots & \ldots & \end{pmatrix}$$

For $t = 3$, the model should learn to predict "sat" (id 5438)

Categorical cross entropy loss

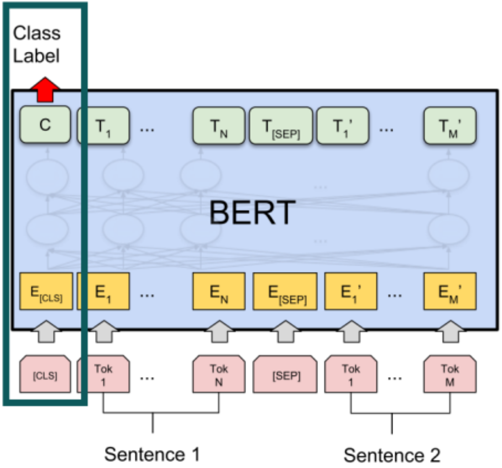
$L(\hat{\boldsymbol{y}}, \boldsymbol{y}) := -\sum_{k=1}^{K} \boldsymbol{y}_{[k]} \log \left(\hat{\boldsymbol{y}}_{[k]} \right)$
$= -1 \cdot \log(\hat{\boldsymbol{y}}[5438]) = -\log(\boldsymbol{P_\theta}[3, 5438]) = -\log(\boldsymbol{P_\theta}[t, \boldsymbol{x}_n[t]])$

Sum over all masked token positions in $\tilde{T}$ gives us line 9:

$$\text{loss}_{\boldsymbol{\theta}} \leftarrow -\sum_{t \in \tilde{T}} \log \boldsymbol{P_\theta}[t, \boldsymbol{x}_n[t]]$$

Downstream tasks and fine-tuning

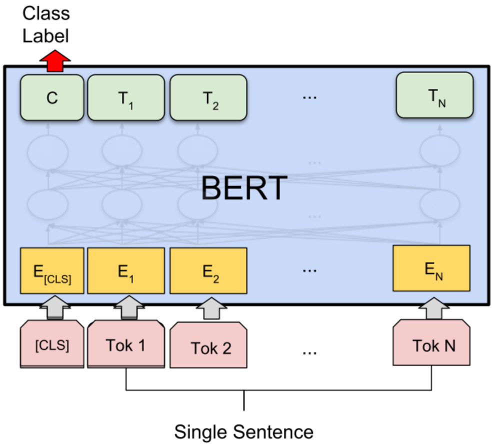
BERT: Representing various NLP tasks



(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG

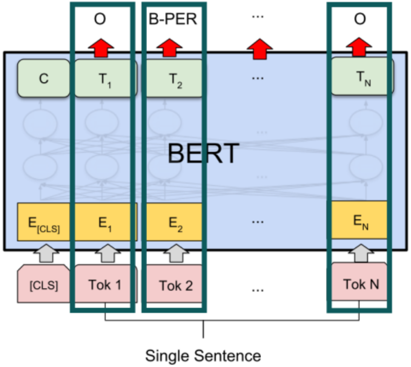
That explains the special [CLS] token at sequence start

BERT: Representing various NLP tasks



(b) Single Sentence Classification Tasks:
SST-2, CoLA

BERT: Representing various NLP tasks



(d) Single Sentence Tagging Tasks: CoNLL-2003 NER

Not conditioned on surrounding predictions

BERT pre-training time

Pretraining BERT took originally 4 days on 64 TPUs[1]

Once pre-trained, transfer and "fine-tune" on your small-data task and get competitive results

P. Izsak, M. Berchansky, and O. Levy (2021). **"How to Train BERT with an Academic Budget".** In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, pp. 10644–10652

[1]Can be done more efficiently, see, e.g., Izsak, Berchansky, and Levy (2021)

Recap

A. Rogers, O. Kovaleva, and A. Rumshisky (2020). **"A Primer in BERTology: What We Know About How BERT Works".** In: *Transactions of the Association for Computational Linguistics* 8, pp. 842–866

BERT stays on the shoulders of many clever concepts and techniques, mastered into a single model

What do we know about how BERT works?

"BERTology has clearly come a long way, but it is fair to say we still have more questions than answers about how BERT works." — Rogers, Kovaleva, and Rumshisky (2020)[2]

[2]Highly recommended reading!

License and credits

Credits

Ivan Habernal