Natural Language Processing with Deep Learning

Lecture 9 – Text classification 5: Introduction to transformers with BERT

Prof. Dr. Ivan Habernal

December 8, 2023

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

PADERBORN UNIVERSITY

www.trusthlt.org

Motivation

Problems: Token (word) embeddings – do not model contextual information

We want contextualized token embeddings

RNN processes left-to-right (or right-to-left)

BERT — The "NLP gamechanger"

J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova (2019). **"BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding".** In: *Proceedings of NAACL*. Minneapolis, Minnesota: Association for Computational Linguistics, pp. 4171–4186

Best paper award at NAACL 2019

State-of-the-art results on various NLP tasks

Directly applicable to other domains and languages

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

Jacob Devlin Ming-Wei Chang Kenton Lee Kristina Toutanova
Google AI Language
{jacobdevlin,mingweichang,kentonl,kristout}@google.com

Abstract

We introduce a new language representation model called **BERT**, which stands for **B**idirectional **E**ncoder **R**epresentations from **T**ransformers. Unlike recent language repre-

There are two existing strategies for applying pre-trained language representations to downstream tasks: *feature-based* and *fine-tuning*. The feature-based approach, such as ELMo (Peters et al., 2018a), uses task-specific architectures that

Neural Machine Translation

Neural Machine Translation

Attention "is all you need"

Multi-task learning

BERT

- Input and pre-training
- Pre-training
- Downstream tasks and fine-tuning

Neural machine translation (NMT)

Why machine translation here?

BERT builds upon techniques from MT

What is machine translation?

- Another popular NLP task
- Many large-scale parallel corpora available

Figure 1: MT is a challenging task!

Image source: `https://languagelog.ldc.upenn.edu/nll/?p=3978`

Recurrent networks for neural MT

Traditionally **encoder-decoder** architectures

- One recurrent neural network processes the entire input and generate its dense representation (**encoder**)
- Other recurrent network produces one token at the time conditioned on the previous states and generated tokens (**decoder**)

Neural MT: Typical architectures (up to 2016-2017)

Long short-term memory (LSTM) / GRU networks

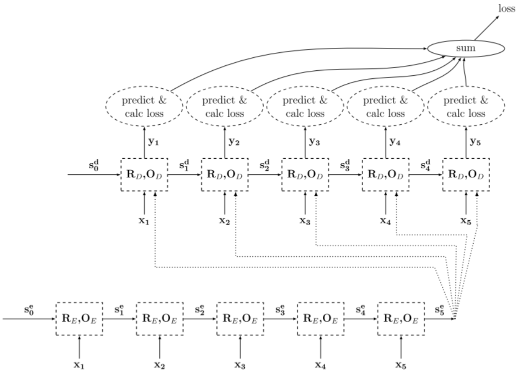


Figure 2: Encoder-decoder RNN

Figure from Y. Goldberg (2016). **"A Primer on Neural Network Models for Natural Language Processing".** In: *Journal of Artificial Intelligence Research* 57, pp. 345–420

Bottlenecks of RNN for machine translation?

Inherently **sequential** nature

- No parallelization
- Big memory footprint (you must "remember" the entire sequence)
- Long-range dependencies modeling: Distance plays a role!

...but when the goal is to learn a good representation of the input sequence, why not use...

- Convolutional neural networks?

Convolutional neural nets (CNN)

One particular property of CNNs

- Modeling dependencies for a **local context**, but by **stacking layers**, one exactly controls the context size

Figure from I. Goodfellow, Y. Bengio, and A. Courville (2016). ***Deep Learning.*** MIT Press

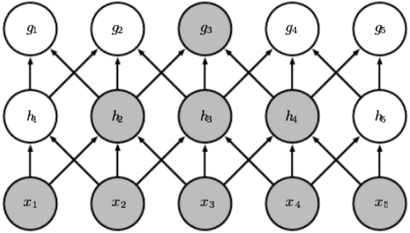


Figure 3: Receptive field of units in deeper layers is larger

Convolutional neural nets for MT

CNNs competitive with RNNs for MT[1]

- Input tokens as word embeddings (not new) or sub-words (will be explained later)
- Fixed-length input? Set-up a maximum length and use <PAD>ding
- But positional information of tokens is lost...

J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin (2017). **"Convolutional Sequence to Sequence Learning".** In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by D. Precup and Y. W. Teh. Sydney, Australia: PMLR, pp. 1243–1252

---

[1] Gehring, Auli, Grangier, Yarats, and Dauphin (2017), Facebook AI Research

Convolutional neural nets for MT by Gehring, Auli, Grangier, Yarats, and Dauphin (2017)

Solution: Positional embeddings

- For each input position n, train another embedding vector P_n: $P_1 = (1.12, -78.6, \dots), P_2, \dots, P_N$
- Word embeddings and position embeddings are simply summed up for each input token
- Why? The model knows with which part of the input/output is dealing with
  - Notice: Removing positional embeddings $\rightarrow$ only slightly worse performance

State-of-the-art results and **9.3–21.3$\times$ faster** than LSTMs on GPU

Attention “is all you need”

- Neural Machine Translation
- Attention “is all you need”
- Multi-task learning
- BERT
    - Input and pre-training
    - Pre-training
    - Downstream tasks and fine-tuning

Attention: Modeling dependencies

Recap: How to model long-range dependencies in input?

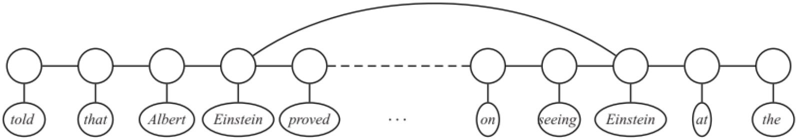


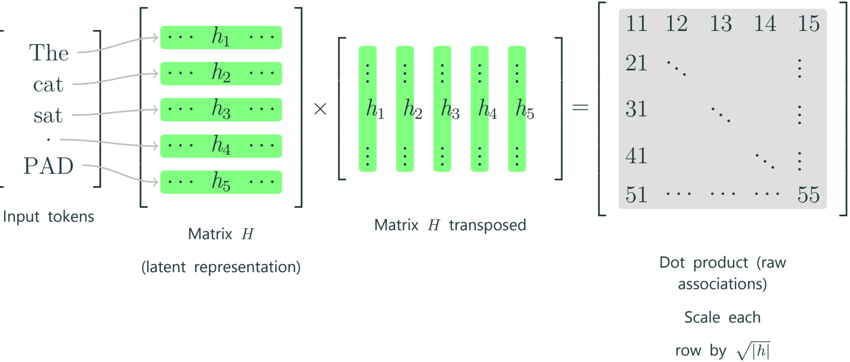
Figure 1: An example of the label consistency problem. Here we would like our model to encourage entities *Albert Einstein* and *Einstein* to get the same label, so as to improve the chance that both are labeled *PERSON*.

– RNNs or stacking CNNs

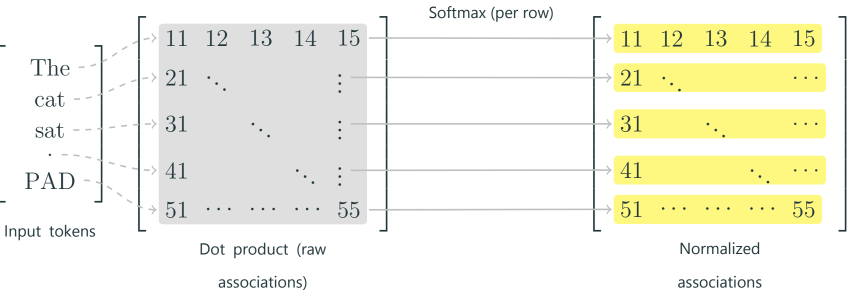
– **Self-Attention**: Utilize associations between all input word pairs

Figure source: V. Krishnan and C. D. Manning (2006). **“An Effective Two-Stage Model for Exploiting Non-Local Dependencies in Named Entity Recognition”.** In: *Proceedings of ACL*. Sydney, Australia: Association for Computational Linguistics, pp. 1121–1128

Self-Attention in detail (1)

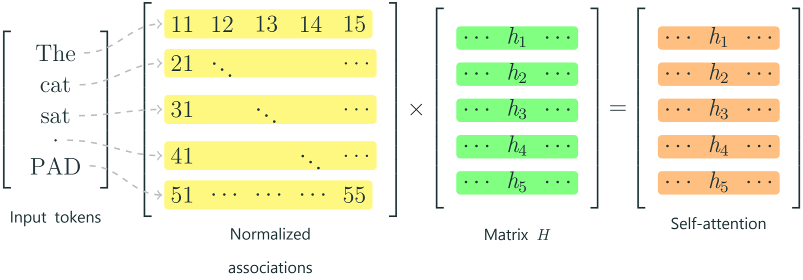


Self-Attention in detail (2)



- Each row corresponds to an input token
- Each row sums up to 1
- Each cell shows the "association strength" with all other tokens

Self-Attention in detail (3)

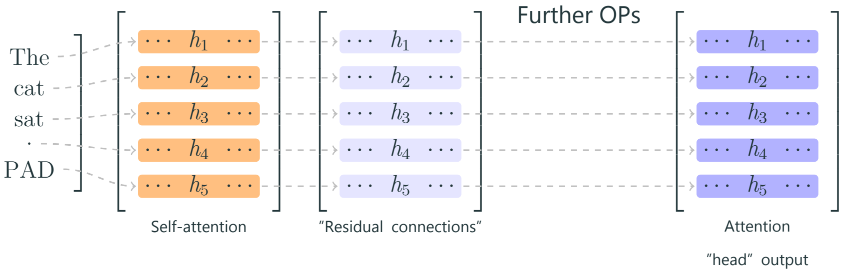


Each position in the latent representation of a token is weighted by the association strength with other tokens

Self-Attention in detail (4)

$$\underset{\text{Self-attention}}{\begin{bmatrix} \cdots & h_1 & \cdots \\ \cdots & h_2 & \cdots \\ \cdots & h_3 & \cdots \\ \cdots & h_4 & \cdots \\ \cdots & h_5 & \cdots \end{bmatrix}} + \underset{\text{Matrix } H}{\begin{bmatrix} \cdots & h_1 & \cdots \\ \cdots & h_2 & \cdots \\ \cdots & h_3 & \cdots \\ \cdots & h_4 & \cdots \\ \cdots & h_5 & \cdots \end{bmatrix}} = \underset{\text{"Residual connections"}}{\begin{bmatrix} \cdots & h_1 & \cdots \\ \cdots & h_2 & \cdots \\ \cdots & h_3 & \cdots \\ \cdots & h_4 & \cdots \\ \cdots & h_5 & \cdots \end{bmatrix}}$$

Self-Attention in detail: Head



Further operations

- Layer normalization
- Feed-forward layer with ReLU
- Another residual connection and layer normalization

Self-attention: More subtleties

- Run N attention "heads" in parallel and concatenate
- Stack on top of each other M-times

Why self-attention?

- Self-attention layer connects all positions with a constant number of sequentially executed operations
- Recurrent layer requires O(n) sequential operations
- Self-attention layers are **fast**

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin (2017). **"Attention Is All You Need".** In: *Advances in Neural Information Processing Systems 30.* Long Beach, CA, USA: Curran Associates, Inc., pp. 5998–6008

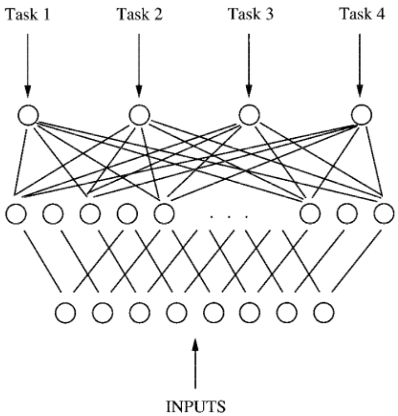
Multi-task learning

Neural Machine Translation
Attention “is all you need”
Multi-task learning
BERT
  Input and pre-training
  Pre-training
  Downstream tasks and fine-tuning

Multi-task Learning

Approach to inductive transfer that improves generalization

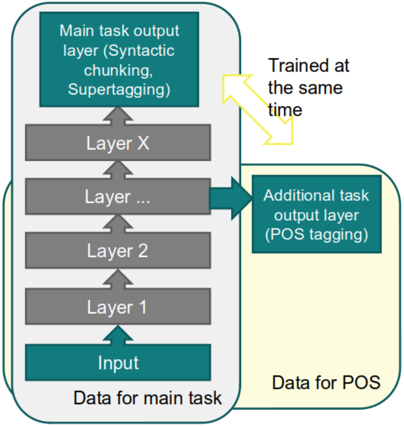
By learning tasks in parallel while using a shared representation



R. Caruana (1997). **"Multi-task Learning".** In: *Machine Learning* 28.1, pp. 41–75

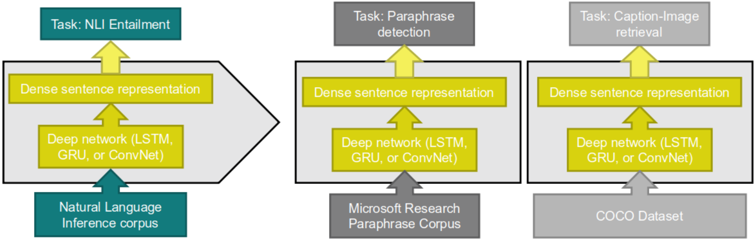
Multi-task learning in NLP

"In case we suspect the existence of a hierarchy between the different tasks, we show that it is worth-while to incorporate this knowledge in the MTL architecture's design, by making lower level tasks affect the lower levels of the representation."



A. Søgaard and Y. Goldberg (2016). **"Deep multi-task learning with low level tasks supervised at lower layers".**

Learn a sentence representation on a different task



"Models learned on NLI can perform better than models trained in ~~unsupervised conditions or~~ on other supervised tasks."[2]

[2] A. Conneau, D. Kiela, H. Schwenk, L. Barrault, and A. Bordes (2017). **"Supervised Learning of Universal Sentence Representations from Natural Language Inference Data".** In: *Proceedings of EMNLP*. Copenhagen, Denmark, pp. 670–680

BERT

Neural Machine Translation

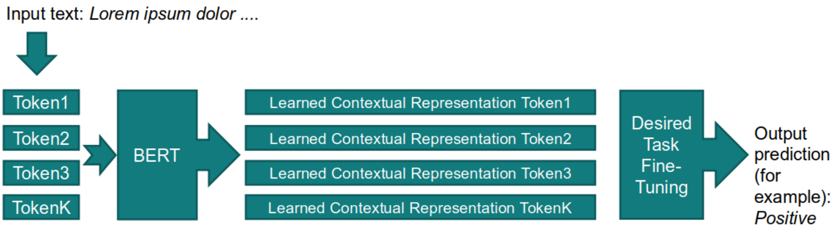
Attention "is all you need"

Multi-task learning

BERT

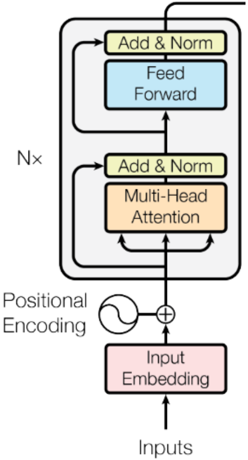
- Input and pre-training
- Pre-training
- Downstream tasks and fine-tuning

BERT: Very abstract view



BERT: The transformer encoder

- Multiple parallel attention "heads" (16 heads)
- With residual connections
- With layer normalization
- Stacked on top of each other (24-times)
- 310,000,000 trainable parameters



Some details (Notation)

Notation and formal description of algorithms adopted from M. Phuong and M. Hutter (2022). ***Formal Algorithms for Transformers.*** arXiv: `2207.09238` Note that they use column-vector notation while here (and in all lectures) we use row-vector notation.

Simplify the set notation

$\{1, 2, \ldots, N\}$ is a set of integers $1, 2, \ldots, N-1, N$

simplify to $[N]$

For example $t \in [N] \equiv t \in \{1, 2, \ldots, N\}$

Basic single-query attention

Input: $\boldsymbol{e} \in \mathbb{R}^{d_{\text{in}}}$, vector representation of the current token

Input: $\boldsymbol{e}_t \in \mathbb{R}^{d_{\text{in}}}$, vector representations of the context tokens $t \in [T]$

Output: $\tilde{\boldsymbol{v}} \in \mathbb{R}^{d_{\text{out}}}$, vector representation of the token and context combined

Params: $\boldsymbol{W_q}, \boldsymbol{W_k} \in \mathbb{R}^{d_{\text{in}} \times d_{\text{attn}}}$, $\boldsymbol{b_q}, \boldsymbol{b_k} \in \mathbb{R}^{d_{\text{attn}}}$, the query and key linear projections

$\boldsymbol{W_v} \in \mathbb{R}^{d_{\text{in}} \times d_{\text{out}}}$, $\boldsymbol{b_v} \in \mathbb{R}^{d_{\text{out}}}$, the value linear projection

function Basic single-query attention

  $\boldsymbol{q} \leftarrow \boldsymbol{e}\boldsymbol{W_q} + \boldsymbol{b_q}$ ▷ Query linear projection

  for $t \in [T]$ **do**

    $\boldsymbol{k_t} \leftarrow \boldsymbol{e_t}\boldsymbol{W_k} + \boldsymbol{b_k}$ ▷ Key linear projection

    $\alpha_t = \frac{\exp(\boldsymbol{q} \cdot \boldsymbol{k_t} / \sqrt{d_{\text{attn}}})}{\sum_{u=1}^{T} \exp(\boldsymbol{q} \cdot \boldsymbol{k_u} / \sqrt{d_{\text{attn}}})}$ ▷ Softmax over scaled dot products ($\alpha_t \in \mathbb{R}$)

    $\boldsymbol{v_t} \leftarrow \boldsymbol{e_t}\boldsymbol{W_v} + \boldsymbol{b_v}$ ▷ Value linear projection

  return $\tilde{\boldsymbol{v}} = \sum_{t=1}^{T} \alpha_t \boldsymbol{v_t}$

Some details

Concatenate matrices of the same dimensions along rows

$$\boldsymbol{Y} = [\boldsymbol{X}^1; \boldsymbol{X}^2; \ldots; \boldsymbol{X}^H] \qquad \boldsymbol{X}^i \in \mathbb{R}^{m \times n} \qquad \boldsymbol{Y} \in \mathbb{R}^{m \times H \cdot n}$$

Example

$$\boldsymbol{A} = \begin{pmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{pmatrix}, \boldsymbol{B} = \begin{pmatrix} 11 & 12 \\ 13 & 14 \\ 15 & 16 \end{pmatrix} \qquad \boldsymbol{Y} = [\boldsymbol{A}; \boldsymbol{B}] = \begin{pmatrix} 1 & 2 & 11 & 12 \\ 3 & 4 & 13 & 14 \\ 5 & 6 & 15 & 16 \end{pmatrix}$$

Some details

How to add a single vector $\boldsymbol{b}$ to each row in a matrix $\boldsymbol{W}$ ($\boldsymbol{W} \in \mathbb{R}^{m \times n}$, $\boldsymbol{b} \in \mathbb{R}^n$)

We want $\boldsymbol{Z} = \boldsymbol{X} +_{\text{(rows)}} \boldsymbol{b}$

Let $\mathbf{1}^m = (1, 1, \ldots, 1_m)$, then $\boldsymbol{Z} = \boldsymbol{X} +_{\text{(rows)}} \boldsymbol{b} = \boldsymbol{X} + (\boldsymbol{b}^\top \mathbf{1}^m)^\top$

Example

$$\boldsymbol{X} = \begin{pmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{pmatrix}, \boldsymbol{b} = \begin{pmatrix} 10 & 20 \end{pmatrix}$$

$$\boldsymbol{b}^\top \mathbf{1}^m = \begin{pmatrix} 10 \\ 20 \end{pmatrix} \begin{pmatrix} 1 & 1 & 1 \end{pmatrix} = \begin{pmatrix} 10 & 10 & 10 \\ 20 & 20 & 20 \end{pmatrix} \qquad (\boldsymbol{b}^\top \mathbf{1}^m)^\top = \begin{pmatrix} 10 & 20 \\ 10 & 20 \\ 10 & 20 \end{pmatrix}$$

Some details

Soft-max for matrices row-wise, $\boldsymbol{A} \in \mathbb{R}^{m \times n}$

$$\underset{\text{row}}{\text{softmax}} : \mathbb{R}^{m \times n} \mapsto \mathbb{R}^{m \times n}$$

$$\underset{\text{row}}{\text{softmax}}(\boldsymbol{A})[i,j] = \frac{\exp(\boldsymbol{A}[i,j])}{\sum_{k=1}^{n} \exp(\boldsymbol{A}[i,k])}$$

Bidirectional / unmasked self-attention

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

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

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

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

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

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

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

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

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

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 $\text{MHA}\textsc{ttention}(\boldsymbol{X}; \boldsymbol{\mathcal{W}})$
  for $h \in [H]$ **do**
    $\boldsymbol{Y}^h \leftarrow \textsc{Attention}(\boldsymbol{X}; \boldsymbol{\mathcal{W}}^h_{\boldsymbol{qkv}})$ $\triangleright\ \boldsymbol{Y}^h \in \mathbb{R}^{\ell_\mathsf{x} \times d_\mathsf{mid}}$
  $\boldsymbol{Y} \leftarrow [\boldsymbol{Y}^1; \boldsymbol{Y}^2; \ldots; \boldsymbol{Y}^H]$ $\triangleright\ \boldsymbol{Y} \in \mathbb{R}^{\ell_\mathsf{x} \times H \cdot d_\mathsf{mid}}$
  return $\tilde{\boldsymbol{V}} = \boldsymbol{Y}\boldsymbol{W}_{\boldsymbol{o}} + \boldsymbol{b}_{\boldsymbol{o}}$

Layer normalization

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

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

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

function $\textsc{LayerNorm}(\boldsymbol{e}; \boldsymbol{\gamma}, \boldsymbol{\beta})$

$m \leftarrow \frac{1}{d} \sum_{i=1}^{d} \boldsymbol{e}[i]$ ▷ '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 $\hat{\boldsymbol{e}} = \frac{e-m}{\sqrt{v}} \odot \boldsymbol{\gamma} + \boldsymbol{\beta}$ ▷ $\odot$ element-wise product

BERT

Input and pre-training

BERT: Tokenization

Tokenizing into a multilingual WordPiece inventory

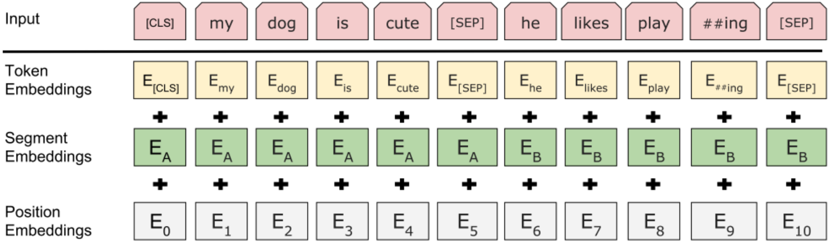
- Recall that WordPiece units are sub-word units
- 30,000 WordPiece units (newer models 110k units, 100 languages)

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



BERT

Pre-training

BERT: Self-supervised multi-task pre-training

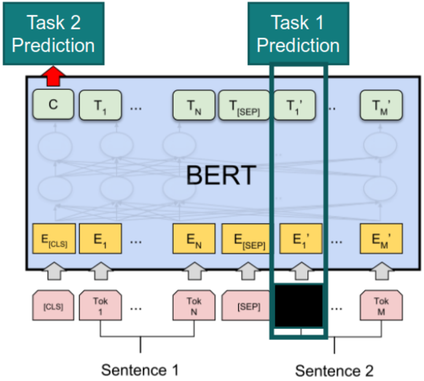
Prepare two auxiliary tasks that need no labeled data

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

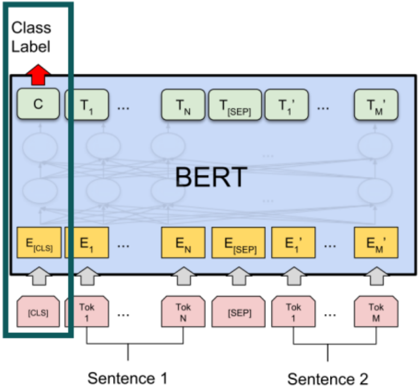
Input = [CLS] the man [MASK] to the store [SEP]
penguin [MASK] are flight ##less birds [SEP]
Label = NotNext

- <PAD>ding is missing
- The actual segments are longer and not necessarily actual sentences (just spans)
- The WordPiece tokens match full words / morphology well in this English text, but recall the ones we have seen before

BERT

Downstream tasks and fine-tuning

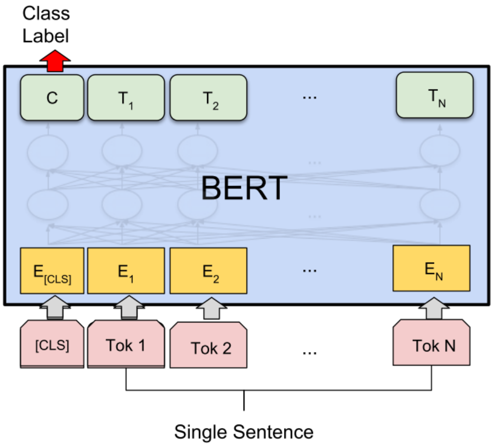
BERT: Representing various NLP tasks



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

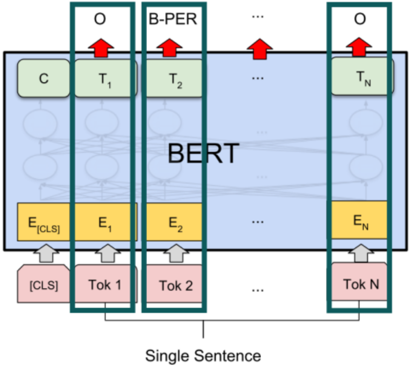
That explains the special [CLS] token at sequence start

BERT: Representing various NLP tasks



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

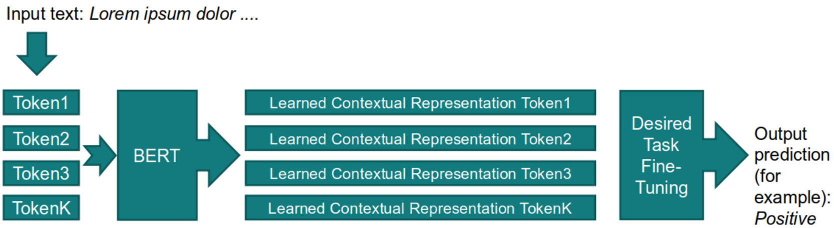
BERT: Representing various NLP tasks



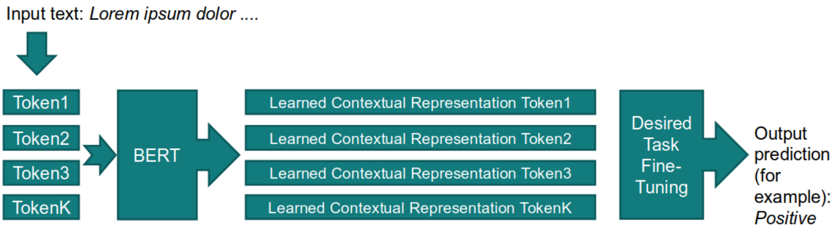
(d) Single Sentence Tagging Tasks: CoNLL-2003 NER

Not conditioned on surrounding predictions

BERT: Very abstract view



BERT: Very abstract view



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

Pretraining BERT took originally 4 days on 64 TPUs[3]

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

[3] 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)[4]

[4] Highly recommended reading!

License and credits

Credits

Ivan Habernal