Natural Language Processing with Deep Learning

Lecture 8 — BERT part 2

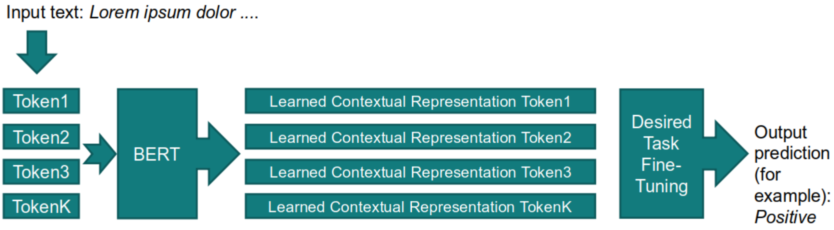
Prof. Dr. Ivan Habernal

December 18, 2024

`www.trusthlt.org`
Trustworthy Human Language Technologies Group (TrustHLT)
Ruhr University Bochum & Research Center Trustworthy Data Science and Security

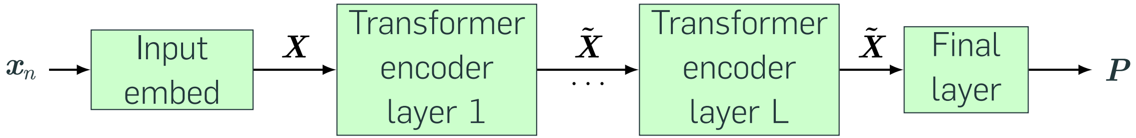
Where we finished last time

BERT: Very abstract view



- BERT produces contextualized token embeddings
- BERT can learn them in a 'clever' way
- BERT can be applied to many downstream tasks

Transformer encoder (BERT)



As usual, green boxes are functions with trainable parameters

$\tilde{\boldsymbol{X}}$ is just a placeholder for **updated** token embeddings matrix $\boldsymbol{X}$

BERT (encoding-only transformer, forward pass)

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

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

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

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

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

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

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

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

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

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

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

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

Input and pre-training

BERT: Tokenization

Tokenizing into a multilingual WordPiece inventory

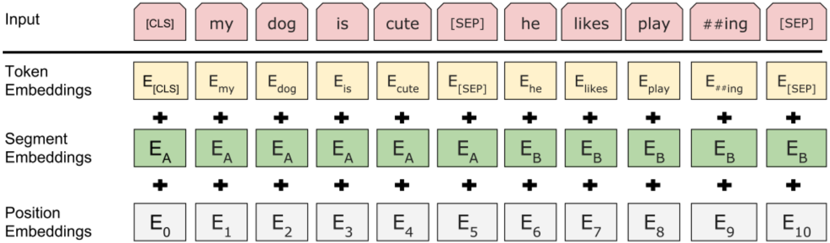
- Recall that WordPiece units are sub-word units
- 30,000 WordPiece units (newer models 110k units, 100 languages)

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



Pre-training

BERT: Self-supervised multi-task pre-training

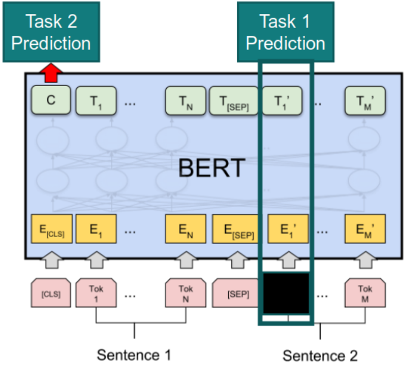
Prepare two auxiliary tasks that need no labeled data

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 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**

      $\ell \leftarrow$ 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

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

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

      $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \cdot \nabla \text{loss}_{\boldsymbol{\theta}}$

  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)

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

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

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

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

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

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

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

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

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

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

Recall: Categorical cross entropy loss

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

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

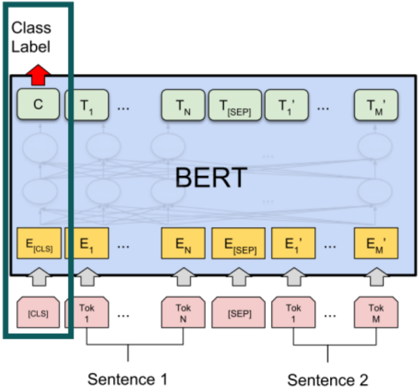
$L(\hat{\boldsymbol{y}}, \boldsymbol{y}) := -\sum_{k=1}^{K} \boldsymbol{y}_{[k]} \log \left(\hat{\boldsymbol{y}}_{[k]} \right)$
$= -1 \cdot \log(\hat{\boldsymbol{y}}[5438]) = -\log(\boldsymbol{P_\theta}[3, 5438]) = -\log(\boldsymbol{P_\theta}[t, \boldsymbol{x}_n[t]])$

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

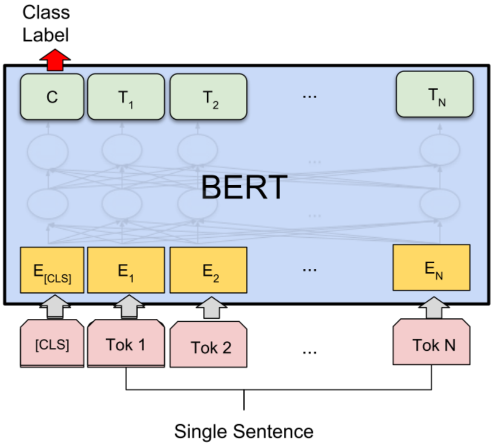
BERT: Representing various NLP tasks



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

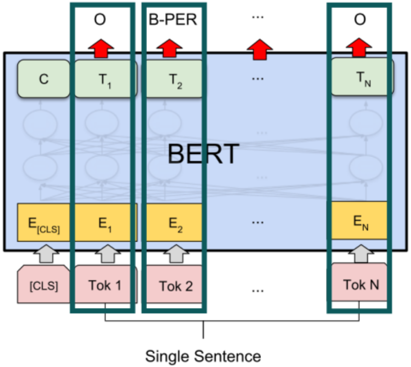
That explains the special [CLS] token at sequence start

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

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]

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

[2] Highly recommended reading!

Sentence BERT

Motivation for Sentence BERT

BERT — state-of-the-art performance on sentence-pair tasks (e.g, semantic textual similarity), but

- both sentences fed into the network → massive computational overhead
- "finding the most similar pair in a collection of 10,000 sentences requires about 50 million inference computations (65 hours) with BERT"

BERT unsuitable for semantic similarity search or unsupervised tasks like clustering

N. Reimers and I. Gurevych (2019). "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, pp. 3980–3990

Sometimes we need just ‘suitable’ sentence embeddings

For **semantic clustering** or **semantic search** — we map each sentence to a vector space

- semantically similar sentences are close to each other

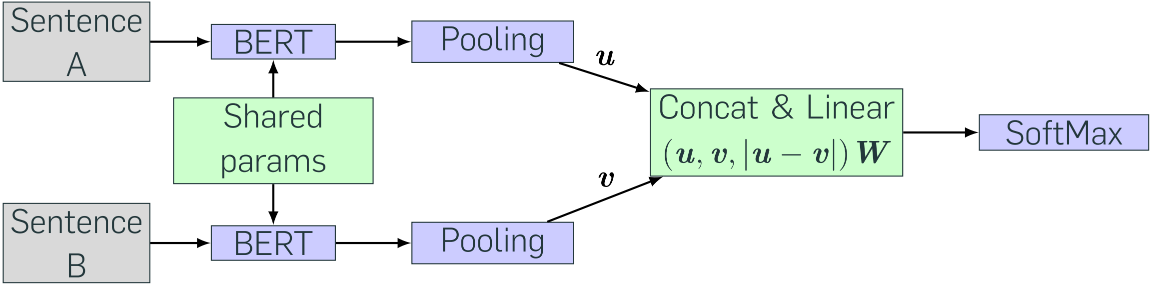
How to use BERT for that?

- average the BERT output layer (known as BERT embeddings)
- use the first token (the [CLS] token) output

→ this practice yields rather bad sentence embeddings, often worse than averaging static word embeddings

N. Reimers and I. Gurevych (2019). “Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks”. In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, pp. 3980–3990

Fine-tune S-BERT on sentence pair classification



Training data:

- SNLI (570,000 sentence pairs)
- MultiNLI (430,000 sentence pairs)

S-BERT provided superior sentence embeddings

Model	MR	CR	SUBJ	MPQA	SST	TREC	MRPC	Avg.
Avg. GloVe embeddings	77.25	78.30	91.17	87.85	80.18	83.0	72.87	81.52
Avg. fast-text embeddings	77.96	79.23	91.68	87.81	82.15	83.6	74.49	82.42
Avg. BERT embeddings	78.66	86.25	94.37	88.66	84.40	92.8	69.45	84.94
BERT CLS-vector	78.68	84.85	94.21	88.23	84.13	91.4	71.13	84.66
InferSent - GloVe	81.57	86.54	92.50	**90.38**	84.18	88.2	75.77	85.59
Universal Sentence Encoder	80.09	85.19	93.98	86.70	86.38	**93.2**	70.14	85.10
SBERT-NLI-base	83.64	89.43	94.39	89.86	88.96	89.6	**76.00**	87.41
SBERT-NLI-large	**84.88**	**90.07**	**94.52**	90.33	**90.66**	87.4	75.94	**87.69**

Figure 1: Evaluation of SBERT sentence embeddings using the SentEval toolkit. SentEval evaluates sentence embeddings on different sentence classification tasks by training a logistic regression classifier using the sentence embeddings as features.

N. Reimers and I. Gurevych (2019). "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, pp. 3980–3990

S-BERT efficiency

"For example, clustering of 10,000 sentences with hierarchical clustering requires with BERT about 65 hours (50 Million sentence combinations). With SBERT, we were able to reduce the effort to about 5 seconds."

Nils' guest lecture (3 parts):

- `https://www.youtube.com/watch?v=qmN1fJ7Fdmo`
- `https://www.youtube.com/watch?v=0RV-q0--NLs`
- `https://www.youtube.com/watch?v=t4Gf4LruVZ4`

N. Reimers and I. Gurevych (2019). "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, pp. 3980–3990

License and credits

Credits

Ivan Habernal