

Natural Language Processing with Deep Learning

RUHR
UNIVERSITÄT
BOCHUM

RUB

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



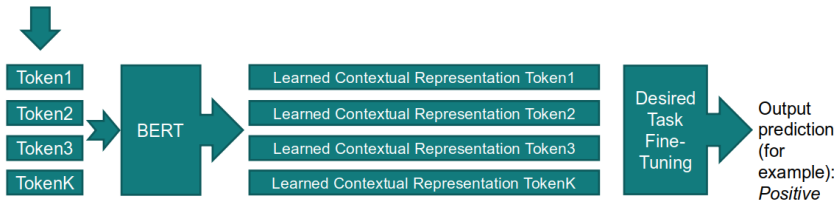
CENTER FOR TRUSTWORTHY
DATA SCIENCE AND SECURITY

Where we finished last time

- 1 Where we finished last time
- 2 Input and pre-training
- 3 Pre-training
- 4 Downstream tasks and fine-tuning
- 5 Sentence BERT

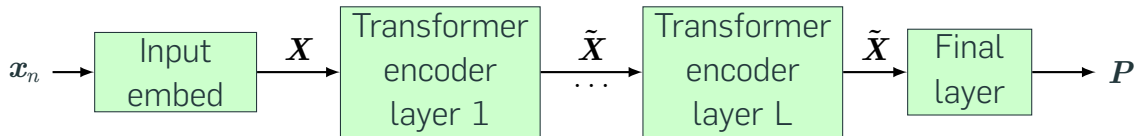
BERT: Very abstract view

Input text: *Lorem ipsum dolor*



- 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{X} is just a placeholder for **updated** token embeddings matrix X

BERT (encoding-only transformer, forward pass)

```
1: function ETransformer( $\mathbf{x}; \mathcal{W}$ )
2:    $\ell \leftarrow \text{length}(\mathbf{x})$ 
3:   for  $t \in [\ell] : \mathbf{e}_t \leftarrow \mathbf{W}_e[\mathbf{x}[t], :] + \mathbf{W}_p[t, :]$            ▷ Token emb. + positional emb.
4:    $\mathbf{X} \leftarrow \text{Stack row-wise}[\mathbf{e}_1, \mathbf{e}_2, \dots, \mathbf{e}_\ell]$ 
5:   for  $l = 1, 2, \dots, L$  do
6:      $\mathbf{X} \leftarrow \mathbf{X} + \text{MHAttention}(\mathbf{X} | \mathcal{W}_l)$            ▷ Multi-head att., residual conn
7:      $\mathbf{X} \leftarrow \text{LayerNormPerRow}(\mathbf{X} | \gamma_l^1, \beta_l^1)$ 
8:      $\mathbf{X} \leftarrow \mathbf{X} + \left( \text{GELU}(\mathbf{X} \mathbf{W}_l^{\text{mlp1}} +_{(\text{row})} \mathbf{b}_l^{\text{mlp1}}) \mathbf{W}_l^{\text{mlp2}} +_{(\text{row})} \mathbf{b}_l^{\text{mlp2}} \right)$    ▷ MLP
9:      $\mathbf{X} \leftarrow \text{LayerNormPerRow}(\mathbf{X} | \gamma_l^2, \beta_l^2)$ 
10:     $\mathbf{X} \leftarrow \text{GELU}(\mathbf{X} \mathbf{W}_f +_{(\text{row})} \mathbf{b}_f)$ 
11:     $\mathbf{X} \leftarrow \text{LayerNormPerRow}(\mathbf{X} | \gamma_l, \beta_l)$ 
12:    return  $\mathbf{P} = \text{softmax}(\mathbf{X} \mathbf{W}_u)$            ▷ Project to vocab., probabilities
```

Input and pre-training

- 1 Where we finished last time
- 2 Input and pre-training**
- 3 Pre-training
- 4 Downstream tasks and fine-tuning
- 5 Sentence BERT

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 first 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

Input	[CLS]	my	dog	is	cute	[SEP]	he	likes	play	##ing	[SEP]
Token Embeddings	$E_{[CLS]}$	E_{my}	E_{dog}	E_{is}	E_{cute}	$E_{[SEP]}$	E_{he}	E_{likes}	E_{play}	$E_{##ing}$	$E_{[SEP]}$
	+	+	+	+	+	+	+	+	+	+	+
Segment Embeddings	E_A	E_A	E_A	E_A	E_A	E_A	E_B	E_B	E_B	E_B	E_B
	+	+	+	+	+	+	+	+	+	+	+
Position Embeddings	E_0	E_1	E_2	E_3	E_4	E_5	E_6	E_7	E_8	E_9	E_{10}

Pre-training

- 1 Where we finished last time
- 2 Input and pre-training
- 3 Pre-training**
- 4 Downstream tasks and fine-tuning
- 5 Sentence BERT

BERT: Self-supervised multi-task pre-training

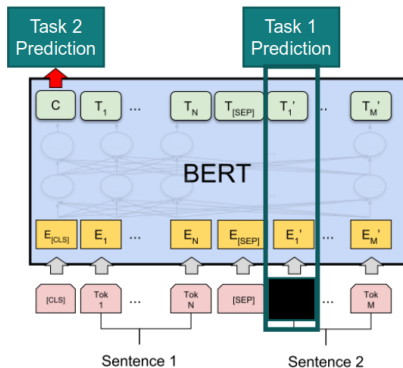
Prepare two auxiliary tasks that need no labeled data

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

```
1: function ETraining( $\{\mathbf{x}_n\}_{n=1}^{N_{\text{data}}}$  seqs,  $\boldsymbol{\theta}$  init. params;  $p_{\text{mask}} \in (0, 1)$ ,  $N_{\text{epochs}}$ ,  $\eta$ )
2:   for  $i \in [N_{\text{epochs}}]$  do
3:     for  $n \in [N_{\text{data}}]$  do
4:        $\ell \leftarrow \text{length}(\mathbf{x}_n)$ 
5:       for  $t \in [\ell]$  do
6:          $\tilde{\mathbf{x}}_n[t] \leftarrow \text{<mask\_token> with prob. } p_{\text{mask}}, \text{ otherwise } \mathbf{x}_n[t]$ 
7:          $\tilde{T} \leftarrow \{t \in [\ell] : \tilde{\mathbf{x}}_n[t] = \text{<mask\_token>}\}$  ▷ Indices of masked
tokens
8:          $\mathbf{P}_{\boldsymbol{\theta}} \leftarrow \text{ETransformer}(\tilde{\mathbf{x}}_n | \boldsymbol{\theta})$ 
9:          $\text{loss}_{\boldsymbol{\theta}} \leftarrow - \sum_{t \in \tilde{T}} \log \mathbf{P}_{\boldsymbol{\theta}}[t, \mathbf{x}_n[t]]$ 
10:         $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \cdot \nabla \text{loss}_{\boldsymbol{\theta}}$ 
11:   return  $\boldsymbol{\theta}$ 
```

Simple example explaining lines 6–7 (masking)

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

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"

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

Explaining line 9 (negative log likelihood)

$$\left(\text{The cat sat}\right) \rightarrow \mathbf{x}_n = \begin{pmatrix} 21 & 11987 & 5438 \end{pmatrix}, \tilde{\mathbf{x}}_n = \begin{pmatrix} 50001 & 11987 & 50001 \end{pmatrix}, \tilde{T} = \{1, 3\}$$

$$\mathbf{P}_\theta \leftarrow \text{ETransformer}(\tilde{\mathbf{x}}_n | \theta)$$

$$\mathbf{P}_\theta = \begin{pmatrix} 0.001 & 0.0007 & \dots & 0.0003 \\ 0.0013 & 0.0065 & \dots & 0.0001 \\ 0.079 & 0.015 & \dots & 0.0001 \end{pmatrix}$$

$\mathbf{P}_\theta \in (0, 1)^{\ell_x \times N_V}$, where each row of \mathbf{P} is a distribution over the vocabulary

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

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

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

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

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

$$\text{Pred: } \hat{\mathbf{y}} = \mathbf{P}_\theta[1, :] = (0.001, \dots, 0.0041_{21}, \dots, 0.0003) \in \mathbb{R}^{N_v}$$

Recall: Categorical cross entropy loss

$$\begin{aligned} L(\hat{\mathbf{y}}, \mathbf{y}) &:= - \sum_{k=1}^K \mathbf{y}_{[k]} \log(\hat{\mathbf{y}}_{[k]}) \\ &= -1 \cdot \log(\hat{\mathbf{y}}[21]) = -\log(\mathbf{P}_\theta[1, 21]) \\ &= -\log(\mathbf{P}_\theta[1, \mathbf{x}_n[1]]) = -\log(\mathbf{P}_\theta[t, \mathbf{x}_n[t]]) \end{aligned}$$

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

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

$$\mathbf{P}_\theta = \begin{pmatrix} \vdots & \dots & \dots \end{pmatrix}$$

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

Categorical cross entropy loss

$$\begin{aligned} L(\hat{\mathbf{y}}, \mathbf{y}) &:= - \sum_{k=1}^K \mathbf{y}_{[k]} \log(\hat{\mathbf{y}}_{[k]}) \\ &= -1 \cdot \log(\hat{\mathbf{y}}[5438]) = -\log(\mathbf{P}_\theta[3, 5438]) = -\log(\mathbf{P}_\theta[t, \mathbf{x}_n[t]]) \end{aligned}$$

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

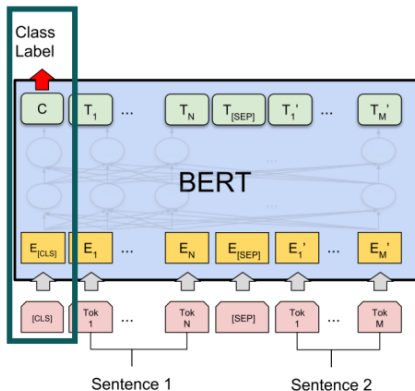
$$\text{loss}_\theta \leftarrow - \sum_{t \in \tilde{T}} \log \mathbf{P}_\theta[t, \mathbf{x}_n[t]]$$

Downstream tasks and fine-tuning

- 1 Where we finished last time
- 2 Input and pre-training
- 3 Pre-training
- 4 Downstream tasks and fine-tuning**
- 5 Sentence BERT

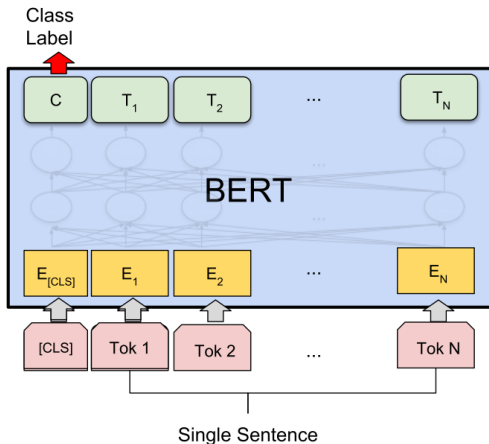
BERT: Representing various NLP tasks

That explains the special [CLS] token at sequence start



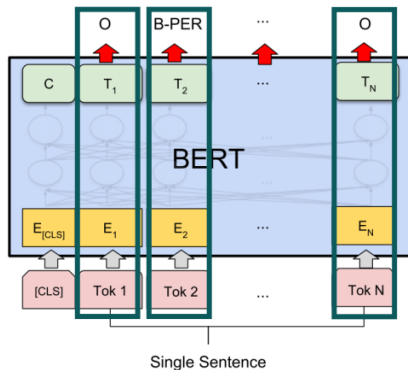
(a) Sentence Pair Classification Tasks:
MNLI, QQP, QNLI, STS-B, MRPC,
RTE, SWAG

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 TPUs¹

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

¹Can 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

²Highly recommended reading!

Sentence BERT

- 1 Where we finished last time
- 2 Input and pre-training
- 3 Pre-training
- 4 Downstream tasks and fine-tuning
- 5 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

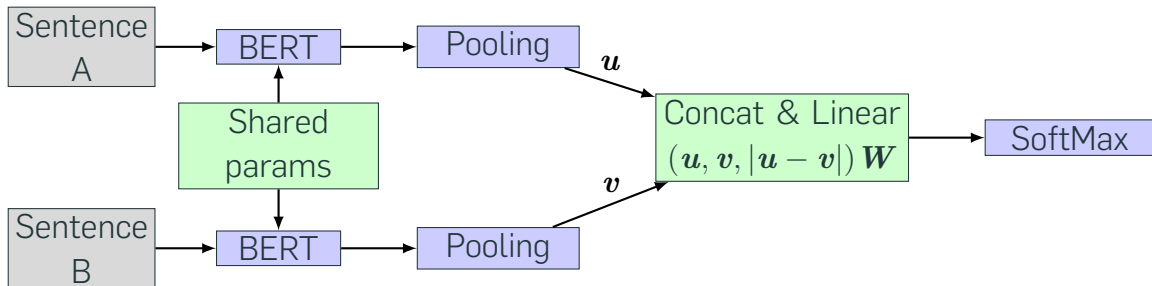
How to use BERT for that?

- 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

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

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.

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=qmN1fJ7Fdm0>
- <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

Licensed under Creative Commons
Attribution-ShareAlike 4.0 International
(CC BY-SA 4.0)



Credits

Ivan Habernal

Content from ACL Anthology papers licensed under CC-BY
<https://www.aclweb.org/anthology>