Introduction to BERT and Transformer:
pre-trained self-attention models to leverage unlabeled corpus data


Acknowledgement to all used slides, figures, tables, equations, texts from the papers, blogs and codes!
Pre-training general language representations

• Feature-based approaches
  • Non-neural word representations
  • Neural embedding
    • Word embedding: Word2Vec, Glove, …
    • Sentence embedding, paragraph embedding, …

• Deep contextualised word representation (ELMo, Embeddings from Language Models) (Peters et al., 2018)

• Fine-tuning approaches
  • OpenAI GPT (Generative Pre-trained Transformer) (Radford et al., 2018a)
  • BERT (Bi-directional Encoder Representations from Transformers) (Devlin et al., 2018)
Content

• ELMo (Peters et al., 2018)
• OpenAI GPT (Radford et al., 2018a)
• Transformer (especially self-attention) (Vaswani et al., 2017)
• BERT (Devlin et al., 2018)
• Analyses & Future Studies
ELMo: deep contextualised word representation
(Peters et al., 2018)

- “Instead of using a fixed embedding for each word, ELMo looks at the entire sentence before assigning each word in it an embedding.”

Acknowledgement to Figure from http://jalammar.github.io/illustrated-bert/
ELMo represents a word $t_k$ as a linear combination of corresponding hidden layers (inc. its embedding).

Unlike usual word embeddings, ELMo is assigned to every *token* instead of a *type*.

**biLMs**

- **Forward**: $p(t_1, t_2, \ldots, t_N) = \prod_{k=1}^{N} p(t_k | t_1, t_2, \ldots, t_{k-1})$
- **Backward**: $p(t_1, t_2, \ldots, t_N) = \prod_{k=1}^{N} p(t_k | t_{k+1}, t_{k+2}, \ldots, t_N)$

Acknowledgement to slides from https://www.slideshare.net/shuntaroy/a-review-of-deep-contextualized-word-representations-peters-2018
ELMo can be integrated to almost all neural NLP tasks with simple concatenation to the embedding layer.

Acknowledgement to slides from https://www.slideshare.net/shunta_roy/a-review-of-deep-contextualized-word-representations-peters-2018
ELMo

Many linguistic tasks are improved by using ELMo

<table>
<thead>
<tr>
<th>TASK</th>
<th>PREVIOUS SOTA</th>
<th>OUR</th>
<th>ELMo + baseline baseline</th>
<th>INCREASE (ABSOLUTE/RELATIVE)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q&amp;A</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SQuAD</td>
<td>Liu et al. (2017)</td>
<td>84.4</td>
<td>81.1</td>
<td>4.7 / 24.9%</td>
</tr>
<tr>
<td>SNLI</td>
<td>Chen et al. (2017)</td>
<td>88.6</td>
<td>88.0</td>
<td>0.7 / 5.8%</td>
</tr>
<tr>
<td>SNLI</td>
<td>He et al. (2017)</td>
<td>81.7</td>
<td>81.4</td>
<td>3.2 / 17.2%</td>
</tr>
<tr>
<td>Coref</td>
<td>Lee et al. (2017)</td>
<td>67.2</td>
<td>67.2</td>
<td>3.2 / 9.8%</td>
</tr>
<tr>
<td>NER (NER)</td>
<td>Peters et al. (2017)</td>
<td>91.93 ± 0.19</td>
<td>90.15</td>
<td>92.22 ± 0.10</td>
</tr>
<tr>
<td>SST-5</td>
<td>McCann et al. (2017)</td>
<td>53.7</td>
<td>51.4</td>
<td>3.3 / 6.8%</td>
</tr>
</tbody>
</table>

Table 1: Test set comparison of ELMo enhanced neural models with state-of-the-art single model baselines across six benchmark NLP tasks. The performance metric varies across tasks – accuracy for SNLI and SST-5; F1 for SQuAD, SRL and NER; average F1 for Coref. Due to the small test sizes for NER and SST-5, we report the mean and standard deviation across five runs with different random seeds. The “increase” column lists both the absolute and relative improvements over our baseline.
OpenAI GPT (Generative Pre-trained Transformer) – (1) pre-training

• Unsupervised pre-training, maximising the log-likelihood,

\[ L_1(\mathcal{U}) = \sum_i \log P(u_i|u_{i-k}, \ldots, u_{i-1}; \Theta) \]

• where \( \mathcal{U} = \{u_1, \ldots, u_n\} \) is an unsupervised corpus of tokens, \( k \) is the size of context window, \( P \) is modelled as a neural network with parameters \( \Theta \).

\[ h_0 = UW_e + W_p \]
\[ h_l = \text{transformer\_block}(h_{l-1}) \forall i \in [1, n] \]
\[ P(u) = \text{softmax}(h_n W_e^T) \]

• where \( U \) is one-hot representation of tokens in the window, \( n \) is the total number of transformer layers, \text{transformer\_block()} denotes the decoder of the Transformer model (multi-headed self-attention and position-wise feedforward layers).

Equations in (Radford et al., 2018)
GPT: (2) Fine-tuning

Given labelled data $C$, including each input as a sequence of tokens $x^1, x^2, \ldots, x^m$, each label as $y$.

$$P(y|x^1, \ldots, x^m) = \text{softmax}(h^m_i W_y)$$

$$L_2(C) = \sum_{(x,y)} \log P(y|x^1, \ldots, x^m)$$

Then maximise the final objective function:

$$L_3(C) = L_2(C) + \lambda \times L_1(C)$$

$\lambda$ is set as 0.5 in the experiment.

Acknowledgement to Figure from [http://jalammar.github.io/illustrated-bert/](http://jalammar.github.io/illustrated-bert/)
Transformer: a seq2seq model

\[ N = 6 \]
\[ d_{\text{model}} = 512 \]

Residual connection & Layer normalisation

Acknowledgement to Figure from http://jalammar.github.io/illustrated-bert/

Figure 1: The Transformer - model architecture.
Figure in (Vaswani et al., 2017)
Self-attention (1)

"The animal didn’t cross the street because it was too tired"

"The animal didn’t cross the street because it was too wide"

\[ \text{Attention}(Q, K, V) = \text{softmax} \left( \frac{QK^T}{\sqrt{d_k}} \right) V \]

Acknowledgement to Figure from http://jalammar.github.io/illustrated-bert/

Equation and Figure in (Vaswani et al., 2017)
Self-attention (2)

Acknowledgement to Figure from http://jalammar.github.io/illustrated-bert/
### Self-attention (3)

<table>
<thead>
<tr>
<th>Input</th>
<th>Embedding</th>
<th>x₁</th>
<th>x₂</th>
</tr>
</thead>
<tbody>
<tr>
<td>Queries</td>
<td>q₁</td>
<td></td>
<td>q₂</td>
</tr>
<tr>
<td>Keys</td>
<td>k₁</td>
<td></td>
<td>k₂</td>
</tr>
<tr>
<td>Values</td>
<td>v₁</td>
<td></td>
<td>v₂</td>
</tr>
</tbody>
</table>

**Score**

\[ q₁ \cdot k₁ = 112 \]
\[ q₁ \cdot k₂ = 96 \]

**Divide by 8** \( (\sqrt{d_k}) \)

14 12

**Softmax**

0.88 0.12

**Softmax X Value**

v₁ v₂

**Sum**

z₁ z₂

Acknowledgement to Figure from [http://jalammar.github.io/illustrated-bert/](http://jalammar.github.io/illustrated-bert/)
Acknowledgement to Figure from \url{http://jalammar.github.io/illustrated-bert/}
Multi-head attention

Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V

\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W^O

\text{where } \text{head}_i = \text{Attention}(QW^Q_i, KW^K_i, VW^V_i)

W^Q_i \in \mathbb{R}^{d_{\text{model}} \times d_k}, W^K_i \in \mathbb{R}^{d_{\text{model}} \times d_k}, W^V_i \in \mathbb{R}^{d_{\text{model}} \times d_v}

W^O \in \mathbb{R}^{hd_v \times d_{\text{model}}}

h = 8, d_k = d_v = \frac{d_{\text{model}}}{h} = 64

(Vaswani et al., 2017)
Multi-head attention

ATTENTION HEAD #0

Q₀

K₀

V₀

W₀^Q

W₀^K

W₀^V

W₁^Q

W₁^K

W₁^V

ATTENTION HEAD #1

Q₁

K₁

V₁

W₁^Q

W₁^K

W₁^V

Acknowledgement to Figure from http://jalammar.github.io/illustrated-bert/
Modelling the dependencies between (keitakurita, 2019)
(1) the input and output tokens
(2) the input tokens themselves
(3) the output tokens themselves.

Acknowledgement to Figure from 

http://mlexplained.com/2017/12/29/attention-is-all-you-need-explained/
Three Multi-Head attention blocks

• Encoder Multi-Head Attention (left)
  • Keys, values and queries are the output of the previous layer in the encoder.
  • Multiple word-word alignments.

• Decoder Masked Multi-Head Attention (lower right)
  • Set the word-word attention weights for the connections to illegal “future” words to $-\infty$.

• Encoder-Decoder Multi-Head Attention (upper right)
  • Keys and values from the output of the encoder, queries from the previous decoder layer.

Figure in (Vaswani et al., 2017)
Decoder Self-Attention

Acknowledgement to Figure from http://web.stanford.edu/class/cs224n/slides/cs224n-2019-lecture14-transformers.pdf
Why self-attention? - Efficiency and Path

Table 1: Maximum path lengths, per-layer complexity and minimum number of sequential operations for different layer types. \( n \) is the sequence length, \( d \) is the representation dimension, \( k \) is the kernel size of convolutions and \( r \) the size of the neighborhood in restricted self-attention.

<table>
<thead>
<tr>
<th>Layer Type</th>
<th>Complexity per Layer</th>
<th>Sequential Operations</th>
<th>Maximum Path Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Self-Attention</td>
<td>( O(n^2 \cdot d) )</td>
<td>( O(1) )</td>
<td>( O(1) )</td>
</tr>
<tr>
<td>Recurrent</td>
<td>( O(n \cdot d^2) )</td>
<td>( O(n) )</td>
<td>( O(n) )</td>
</tr>
<tr>
<td>Convolutional</td>
<td>( O(k \cdot n \cdot d^2) )</td>
<td>( O(1) )</td>
<td>( O(\log_k(n)) )</td>
</tr>
<tr>
<td>Self-Attention (restricted)</td>
<td>( O(r \cdot n \cdot d) )</td>
<td>( O(1) )</td>
<td>( O(n/r) )</td>
</tr>
</tbody>
</table>

Table in (Vaswani et al., 2017)
Maximum Path Length in RNN and Self-attention

Acknowledged to Figure from http://mlexplained.com/2017/12/29/attention-is-all-you-need-explained/
[Acknowledgement to Figure from http://jalammar.github.io/illustrated-bert/]

Decoding time step: 1 2 3 4 5 6

OUTPUT

ENCODER

ENCODER

LINEAR + SOFTMAX

DECODER

DECODER

EMBEDDING WITH TIME SIGNAL

EMBEDDINGS

INPUT: Je suis étudiant
Acknowledgement to Figure from http://jalammar.github.io/illustrated-bert/
Positional Embedding

• In order to add position information (order of the sequence)

\[
PE_{(pos,2i)} = \sin(pos/10000^{2i/d_{model}})
\]

\[
PE_{(pos,2i+1)} = \cos(pos/10000^{2i/d_{model}})
\]

• Each dimension of the positional encoding corresponds to a sinusoid.

• For any fixed offset \( k \), \( PE_{pos+k} \) can be represented as a linear transformation of \( PE_{pos} \). This would allow the model to easily learn to attend by relative positions.

Equations in (Vaswani et al., 2017)
The Transformer

Adopted by GPT

Table 2: The Transformer achieves better BLEU scores than previous state-of-the-art models on the English-to-German and English-to-French newstest2014 tests at a fraction of the training cost.

<table>
<thead>
<tr>
<th>Model</th>
<th>BLEU</th>
<th>Training Cost (FLOPs)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>EN-DE</td>
<td>EN-FR</td>
</tr>
<tr>
<td>ByteNet [18]</td>
<td>23.75</td>
<td></td>
</tr>
<tr>
<td>GNMT + RL [38]</td>
<td>24.6</td>
<td>39.92</td>
</tr>
<tr>
<td>ConvS2S [9]</td>
<td>25.16</td>
<td>40.46</td>
</tr>
<tr>
<td>MoE [32]</td>
<td>26.03</td>
<td>40.56</td>
</tr>
<tr>
<td>Deep-Att + PosUnk Ensemble [39]</td>
<td></td>
<td>40.4</td>
</tr>
<tr>
<td>GNMT + RL Ensemble [38]</td>
<td>26.30</td>
<td>41.16</td>
</tr>
<tr>
<td>ConvS2S Ensemble [9]</td>
<td>26.36</td>
<td><strong>41.29</strong></td>
</tr>
<tr>
<td>Transformer (base model)</td>
<td>27.3</td>
<td>38.1</td>
</tr>
<tr>
<td>Transformer (big)</td>
<td><strong>28.4</strong></td>
<td><strong>41.8</strong></td>
</tr>
</tbody>
</table>

Table in (Vaswani et al., 2017)
Table 3: Variations on the Transformer architecture. Unlisted values are identical to those of the base model. All metrics are on the English-to-German translation development set, newstest2013. Listed perplexities are per-wordpiece, according to our byte-pair encoding, and should not be compared to per-word perplexities.

<table>
<thead>
<tr>
<th></th>
<th>N</th>
<th>d_{model}</th>
<th>d_{ff}</th>
<th>h</th>
<th>d_k</th>
<th>d_v</th>
<th>P_{drop}</th>
<th>\epsilon_s</th>
<th>train steps</th>
<th>PPL (dev)</th>
<th>BLEU (dev)</th>
<th>params \times 10^6</th>
</tr>
</thead>
<tbody>
<tr>
<td>base</td>
<td>6</td>
<td>512</td>
<td>2048</td>
<td>8</td>
<td>64</td>
<td>64</td>
<td>0.1</td>
<td>0.1</td>
<td>100K</td>
<td>4.92</td>
<td>25.8</td>
<td>65</td>
</tr>
<tr>
<td>(A)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>512</td>
<td>512</td>
<td>4</td>
<td>128</td>
<td>128</td>
<td>0.1</td>
<td>0.1</td>
<td>100K</td>
<td>5.29</td>
<td>24.9</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4</td>
<td></td>
<td></td>
<td>16</td>
<td>32</td>
<td>32</td>
<td></td>
<td></td>
<td></td>
<td>5.00</td>
<td>25.5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>16</td>
<td></td>
<td></td>
<td></td>
<td>16</td>
<td>16</td>
<td></td>
<td></td>
<td></td>
<td>4.91</td>
<td>25.8</td>
<td></td>
</tr>
<tr>
<td></td>
<td>32</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>5.01</td>
<td>25.4</td>
<td></td>
</tr>
<tr>
<td>(B)</td>
<td></td>
<td>16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>5.16</td>
<td>25.1</td>
<td>58</td>
</tr>
<tr>
<td></td>
<td></td>
<td>32</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>5.01</td>
<td>25.4</td>
<td>60</td>
</tr>
<tr>
<td>(C)</td>
<td>2</td>
<td></td>
<td></td>
<td>256</td>
<td>32</td>
<td>32</td>
<td></td>
<td></td>
<td></td>
<td>6.11</td>
<td>23.7</td>
<td>36</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td></td>
<td></td>
<td>1024</td>
<td>128</td>
<td>128</td>
<td></td>
<td></td>
<td></td>
<td>5.19</td>
<td>25.3</td>
<td>50</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td></td>
<td></td>
<td>1024</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4.88</td>
<td>25.5</td>
<td>80</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4096</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>5.75</td>
<td>24.5</td>
<td>28</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4.66</td>
<td>26.0</td>
<td>168</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1024</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>5.12</td>
<td>25.4</td>
<td>53</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4.75</td>
<td>26.2</td>
<td>90</td>
</tr>
<tr>
<td>(D)</td>
<td></td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>5.77</td>
<td>24.6</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4.95</td>
<td>25.5</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4.67</td>
<td>25.3</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>5.47</td>
<td>25.7</td>
<td></td>
</tr>
<tr>
<td>(E)</td>
<td></td>
<td>positional embedding instead of sinusoids</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4.92</td>
<td>25.7</td>
<td></td>
</tr>
<tr>
<td>big</td>
<td>6</td>
<td>1024</td>
<td>4096</td>
<td>16</td>
<td></td>
<td></td>
<td>0.3</td>
<td>300K</td>
<td></td>
<td>4.33</td>
<td>26.4</td>
<td>213</td>
</tr>
</tbody>
</table>
What is BERT (Bidirectional Encoder Representations from Transformers)?

Figure 1: Differences in pre-training model architectures. BERT uses a bidirectional Transformer. OpenAI GPT uses a left-to-right Transformer. ELMo uses the concatenation of independently trained left-to-right and right-to-left LSTM to generate features for downstream tasks. Among three, only BERT representations are jointly conditioned on both left and right context in all layers.

Figure in (Devlin et al., 2018)
Input Representation

Hidden state corresponding to [CLS] will be used as the sentence representation

- Token Embeddings: WordPiece embedding (Wu et al., 2016)
- Segment Embeddings: randomly initialized and learned; single sentence input only adds $E_A$
- Position embeddings: randomly initialized and learned

Figure in (Devlin et al., 2018)
Training tasks (1) - Masked Language Model

- **Masked Language Model:** Cloze Task

- **Masking**(input_seq):
  - For every input_seq:
    - Randomly select 15% of tokens (not more than 20 per seq)
      - For 80% of the time:
        - Replace the word with the [MASK] token.
      - For 10% of the time:
        - Replace the word with a random word
      - For 10% of the time:
        - Keep the word unchanged.

- For related code see `def create_masked_lm_predictions(···)` in
  [https://github.com/google-research/bert/blob/master/create_pretraining_data.py](https://github.com/google-research/bert/blob/master/create_pretraining_data.py)

Acknowledgement to the Figure from [http://jalammar.github.io/illustrated-bert/](http://jalammar.github.io/illustrated-bert/)
Training tasks (2) – Next Sentence Prediction

- Next sentence prediction – Binary classification
- For every input document as a sentence-token 2D list:
  - Randomly select a split over sentences:
    - Store the segment A
    - For 50% of the time:
      - Sample random sentence split from another document as segment B.
    - For 50% of the time:
      - Use the actual sentences as segment B.
  - Masking (Truncate([segment A, segment B]))
- For related code see `def create_instances_from_document(...)` in https://github.com/google-research/bert/blob/master/create_pretraining_data.py

Acknowledgement to the Figure adapted from http://jalammar.github.io/illustrated-bert/
Pre-Training datasets and details

- Training loss $L$ is the sum of the mean masked LM likelihood and mean next sentence prediction likelihood.

- Dataset: Long contiguous word sequences.
  - BooksCorpus (800M words), about 7,000 unique unpublished books from a variety of genres including Adventure, Fantasy, and Romance.
  - English Wikipedia (2,500M words), excluding lists, tables, headers.

- Sequence length 512; Batch size 256; trained for 1M steps (approximately 40 epochs); learning rate $1e^{-4}$; Adam optimiser, $\beta_1$ as 0.9, $\beta_2$ as 0.999; dropout as 0.1 on all layers; GELU activation; L2 weight decay of 0.01; learning rate warmup over the first 10,000 steps, linear decay of learning rate …
• BERT_{BASE}: N = 6, d_{model} = 512, h = 12, Total Parameters=110M
• 4 cloud TPUs in Pod configuration (16 TPU chips total)

• BERT_{LARGE}: N = 24, d_{model} = 1024, h = 16, Total Parameters=340M
• 16 Cloud TPUs (64 TPU chips total)

• Each pretraining took 4 days to complete.
Fine-tuning with BERT

• Context vector $C$: Take the final hidden state corresponding to the first token in the input: [CLS].

• Transform to a probability distribution of the class labels:

$$P = \text{softmax}(CW^T)$$

• **Batch size**: 16, 32
• **Learning rate (Adam)**: 5e-5, 3e-5, 2e-5
• **Number of epochs**: 3, 4

Figure in (Devlin et al., 2018)
Evaluation for BERT: GLUE

• General Language Understanding Evaluation (GLUE) benchmark: Standard split of data to train, validation, test, where labels for the test set is only held in the server.

• Sentence pair tasks
  • MNLI, Multi-Genre Natural Language Inference
  • QQP, Quora Question Pairs
  • QNLI, Question Natural Language Inference
  • STS-B The Semantic Textual Similarity Benchmark
  • MRPC Microsoft Research Paraphrase Corpus
  • RTE Recognizing Textual Entailment
  • WNLI Winograd NLI is a small natural language inference dataset

• Single sentence classification
  • SST-2 The Stanford Sentiment Treebank
  • CoLA The Corpus of Linguistic Acceptability
## Evaluation for BERT: GLUE

<table>
<thead>
<tr>
<th>System</th>
<th>MNLI-(m/mm)</th>
<th>QQP</th>
<th>QNLI</th>
<th>SST-2</th>
<th>CoLA</th>
<th>STS-B</th>
<th>MRPC</th>
<th>RTE</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>392k</td>
<td>363k</td>
<td>108k</td>
<td>67k</td>
<td>8.5k</td>
<td>5.7k</td>
<td>3.5k</td>
<td>2.5k</td>
<td></td>
</tr>
<tr>
<td>Pre-OpenAI SOTA</td>
<td>80.6/80.1</td>
<td>66.1</td>
<td>82.3</td>
<td>93.2</td>
<td>35.0</td>
<td>81.0</td>
<td>86.0</td>
<td>61.7</td>
<td>74.0</td>
</tr>
<tr>
<td>BiLSTM+ELMo+Attn</td>
<td>76.4/76.1</td>
<td>64.8</td>
<td>79.9</td>
<td>90.4</td>
<td>36.0</td>
<td>73.3</td>
<td>84.9</td>
<td>56.8</td>
<td>71.0</td>
</tr>
<tr>
<td>OpenAI GPT</td>
<td>82.1/81.4</td>
<td>70.3</td>
<td>88.1</td>
<td>91.3</td>
<td>45.4</td>
<td>80.0</td>
<td>82.3</td>
<td>56.0</td>
<td>75.2</td>
</tr>
<tr>
<td>BERT_{BASE}</td>
<td>84.6/83.4</td>
<td>71.2</td>
<td>90.1</td>
<td>93.5</td>
<td>52.1</td>
<td>85.8</td>
<td>88.9</td>
<td>66.4</td>
<td>79.6</td>
</tr>
<tr>
<td>BERT_{LARGE}</td>
<td><strong>86.7/85.9</strong></td>
<td><strong>72.1</strong></td>
<td><strong>91.1</strong></td>
<td><strong>94.9</strong></td>
<td><strong>60.5</strong></td>
<td><strong>86.5</strong></td>
<td><strong>89.3</strong></td>
<td><strong>70.1</strong></td>
<td><strong>81.9</strong></td>
</tr>
</tbody>
</table>

Table 1: GLUE Test results, scored by the GLUE evaluation server. The number below each task denotes the number of training examples. The “Average” column is slightly different than the official GLUE score, since we exclude the problematic WNLI set. OpenAI GPT = (L=12, H=768, A=12); BERT\textsubscript{BASE} = (L=12, H=768, A=12); BERT\textsubscript{LARGE} = (L=24, H=1024, A=16). BERT and OpenAI GPT are single-model, single task. All results obtained from [https://gluebenchmark.com/leaderboard](https://gluebenchmark.com/leaderboard) and [https://blog.openai.com/language-unsupervised/](https://blog.openai.com/language-unsupervised/).
Evaluation on SQUAD

- The Stanford Question Answering Dataset (SQuAD) is a collection of 100k crowdsourced question/answer pairs.

- **Input Question:**
  Where do water droplets collide with ice crystals to form precipitation?

- **Input Paragraph:**
  ... Precipitation forms as smaller droplets coalesce via collision with other rain drops or ice crystals within a cloud. ...

- **Output Answer:**
  within a cloud

<table>
<thead>
<tr>
<th>System</th>
<th>Dev EM</th>
<th>Dev F1</th>
<th>Test EM</th>
<th>Test F1</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Leaderboard (Oct 8th, 2018)</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Human</td>
<td>-</td>
<td>-</td>
<td>82.3</td>
<td>91.2</td>
</tr>
<tr>
<td>#1 Ensemble - nlnet</td>
<td>-</td>
<td>-</td>
<td>86.0</td>
<td>91.7</td>
</tr>
<tr>
<td>#2 Ensemble - QANet</td>
<td>-</td>
<td>-</td>
<td>84.5</td>
<td>90.5</td>
</tr>
<tr>
<td>#1 Single - nlnet</td>
<td>-</td>
<td>-</td>
<td>83.5</td>
<td>90.1</td>
</tr>
<tr>
<td>#2 Single - QANet</td>
<td>-</td>
<td>-</td>
<td>82.5</td>
<td>89.3</td>
</tr>
<tr>
<td><strong>Published</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BiDAF+ELMo (Single)</td>
<td>-</td>
<td>85.8</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>R.M. Reader (Single)</td>
<td>78.9</td>
<td>86.3</td>
<td>79.5</td>
<td>86.6</td>
</tr>
<tr>
<td>R.M. Reader (Ensemble)</td>
<td>81.2</td>
<td>87.9</td>
<td>82.3</td>
<td>88.5</td>
</tr>
<tr>
<td><strong>Ours</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BERT&lt;sub&gt;BASE&lt;/sub&gt; (Single)</td>
<td>80.8</td>
<td>88.5</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>BERT&lt;sub&gt;LARGE&lt;/sub&gt; (Single)</td>
<td>84.1</td>
<td>90.9</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>BERT&lt;sub&gt;LARGE&lt;/sub&gt; (Ensemble)</td>
<td>85.8</td>
<td>91.8</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>BERT&lt;sub&gt;LARGE&lt;/sub&gt; (Sgl.+TriviaQA)</td>
<td>84.2</td>
<td>91.1</td>
<td>85.1</td>
<td>91.8</td>
</tr>
<tr>
<td>BERT&lt;sub&gt;LARGE&lt;/sub&gt; (Ens.+TriviaQA)</td>
<td>86.2</td>
<td>92.2</td>
<td>87.4</td>
<td>93.2</td>
</tr>
</tbody>
</table>

Table 2: SQuAD results. The BERT ensemble is 7x systems which use different pre-training checkpoints and fine-tuning seeds.
Evaluation on Named Entity Recognition

- The CoNLL 2003 Named Entity Recognition (NER) dataset. This dataset consists of 200k training words which have been annotated as **Person**, **Organization**, **Location**, **Miscellaneous**, or **Other** (non-named entity).

Table in (Devlin et al., 2018)

<table>
<thead>
<tr>
<th>System</th>
<th>Dev F1</th>
<th>Test F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>ELMo+BiLSTM+CRF</td>
<td>95.7</td>
<td>92.2</td>
</tr>
<tr>
<td>CVT+Multi (Clark et al., 2018)</td>
<td>-</td>
<td>92.6</td>
</tr>
<tr>
<td>BERT&lt;sub&gt;BASE&lt;/sub&gt;</td>
<td>96.4</td>
<td>92.4</td>
</tr>
<tr>
<td>BERT&lt;sub&gt;LARGE&lt;/sub&gt;</td>
<td>96.6</td>
<td>92.8</td>
</tr>
</tbody>
</table>

Table 3: CoNLL-2003 Named Entity Recognition results. The hyperparameters were selected using the Dev set, and the reported Dev and Test scores are averaged over 5 random restarts using those hyperparameters.
Ablation Study (1) – on pre-train tasks

<table>
<thead>
<tr>
<th>Tasks</th>
<th>MNLI-m (Acc)</th>
<th>QNLI (Acc)</th>
<th>MRPC (Acc)</th>
<th>SST-2 (Acc)</th>
<th>SQuAD (F1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>BERT\textsubscript{BASE}</td>
<td>84.4</td>
<td>88.4</td>
<td>86.7</td>
<td>92.7</td>
<td>88.5</td>
</tr>
<tr>
<td>No NSP</td>
<td>83.9</td>
<td>84.9</td>
<td>86.5</td>
<td>92.6</td>
<td>87.9</td>
</tr>
<tr>
<td>LTR &amp; No NSP</td>
<td>82.1</td>
<td>84.3</td>
<td>77.5</td>
<td>92.1</td>
<td>77.8</td>
</tr>
<tr>
<td>+ BiLSTM</td>
<td>82.1</td>
<td>84.1</td>
<td>75.7</td>
<td>91.6</td>
<td>84.9</td>
</tr>
</tbody>
</table>

Table 5: Ablation over the pre-training tasks using the BERT\textsubscript{BASE} architecture. “No NSP” is trained without the next sentence prediction task. “LTR & No NSP” is trained as a left-to-right LM without the next sentence prediction, like OpenAI GPT. “+ BiLSTM” adds a randomly initialized BiLSTM on top of the “LTR + No NSP” model during fine-tuning.

Table in (Devlin \textit{et al.}, 2018)
Ablation Study (2) – on model sizes

<table>
<thead>
<tr>
<th>Hyperparams</th>
<th>Dev Set Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>#L   #H #A</td>
</tr>
<tr>
<td>3</td>
<td>768  12</td>
</tr>
<tr>
<td>6</td>
<td>768  3</td>
</tr>
<tr>
<td>6</td>
<td>768  12</td>
</tr>
<tr>
<td>12</td>
<td>768  12</td>
</tr>
<tr>
<td>12</td>
<td>1024 16</td>
</tr>
<tr>
<td>24</td>
<td>1024 16</td>
</tr>
</tbody>
</table>

Table 6: Ablation over BERT model size. #L = the number of layers; #H = hidden size; #A = number of attention heads. “LM (ppl)” is the masked LM perplexity of held-out training data.

Table in (Devlin et al., 2018)
Ablation Study (3) – on pre-training steps

Figure in (Devlin et al., 2018)
Ablation Study (4) – using BERT as feature extractor (*without* fine-tuning)

<table>
<thead>
<tr>
<th>Layers</th>
<th>Dev F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Finetune All</td>
<td>96.4</td>
</tr>
<tr>
<td>First Layer (Embeddings)</td>
<td>91.0</td>
</tr>
<tr>
<td>Second-to-Last Hidden</td>
<td>95.6</td>
</tr>
<tr>
<td>Last Hidden</td>
<td>94.9</td>
</tr>
<tr>
<td>Sum Last Four Hidden</td>
<td>95.9</td>
</tr>
<tr>
<td>Concat Last Four Hidden</td>
<td>96.1</td>
</tr>
<tr>
<td>Sum All 12 Layers</td>
<td>95.5</td>
</tr>
</tbody>
</table>

Table 7: Ablation using BERT with a feature-based approach on CoNLL-2003 NER. The activations from the specified layers are combined and fed into a two-layer BiLSTM, without backpropagation to BERT.
Why BERT works?

• Leveraging huge unlabeled and high quality data: 7000 books + Wikipedia (together 3300M words)

• Multi-head self-attention blocks in Transformer:
  • modelling the intra- and extra- word-word relations
  • parallelable within instance and thus efficient

• Task similarity: masked language modelling + next sentence prediction
How to improve BERT?

• **Pre-training**
  • Better tasks for pre-training for more complex usage
  • Better (larger, high-quality) data
  • Cross-lingual BERT for unsupervised learning (Lample & Conneau, 2019)
  • Even larger model, GPT-2: zero shot to outperform the SOTA (Radford et al., 2018b)

• **Fine-tuning**
  • Better loss in fine-tuning
  • Introduce new tasks in fine-tuning
An architecture for multi-label classification
(Dong, 2019)

Is it possible? Any further thought?
Recommended Learning Resources

References


