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

Research Report for the NLP Cryptonite Taskphase

By Pratham Shah, Research AI JTP 240905614

- Authors: Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova
- Published: 2018
- PDF: https://arxiv.org/pdf/1810.04805.pdf

This report reviews Bidirectional Encoder Representations from Transformers (BERT) as a new state-of-the-art model for a range of natural language processing (NLP) tasks.

BERT introduced the idea of deeply pre-training an encoder transformer model in both directions on massive corpora, then fine-tuning it on specific tasks. This approach resulted in significant gains across tasks like question answering, sentiment analysis, and named entity recognition.

The report highlights BERT's architecture, pre-training strategy, practical impact, limitations, and enduring influence on modern language model research.

Introduction

Natural language understanding tasks, such as question answering, text classification, and language inference, require models to draw on rich language context. Traditionally, most models processed text left-to-right (or right-to-left), like RNNs and earlier Transformers, limiting their ability to grasp complex bidirectional dependencies.

BERT fundamentally changed this paradigm by:

- Pre-training a deep Transformer encoder on large unlabeled text corpora, capturing both left and right context simultaneously (bidirectionality).

- Demonstrating that a single, large pre-trained model can be simply fine-tuned (with minimal architectural changes) to achieve state-of-the-art results on a diverse array of downstream NLP tasks.

Problem Statement

Before BERT, language models and sentence encoders were typically trained using uni-directional context or shallow tasks (eg. predicting the next word), which restricted their expressiveness and limited downstream performance for tasks that need understanding of a word's full context.
Key limitations were:

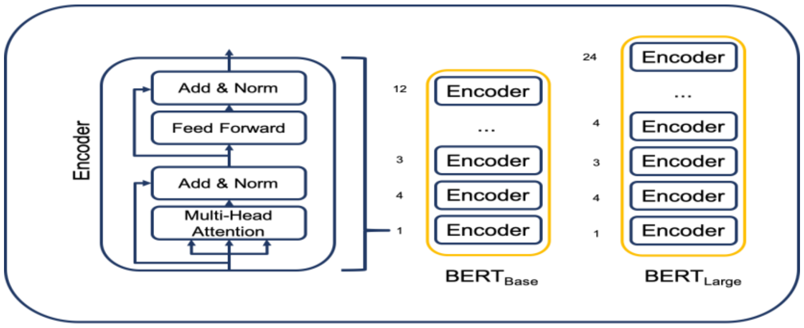
- Unidirectionality: Existing models could only incorporate information from one direction at a time, constraining their language representations.
- Task-Specific Design: For each target NLP task, it was common to train a new model from scratch or use heavy task-specific architectural changes.
- Limited Transferability: Universal pre-trained features that could easily be applied to many tasks were lacking.

BERT directly addresses these shortfalls with a robust, generic and bidirectional pre-training strategy.

BERT Architecture and Pre-training

Architecture

- Encoder-Only Transformer: BERT is based solely on the Transformer encoder stack (not the decoder).
    - BERT-Base: 12 identical encoder layers (transformer blocks), each with 12 self-attention heads, 768 hidden size, and 110M parameters.
    - BERT-Large: 24 encoder layers, 16 attention heads per layer, 1,024 hidden size, 340M parameters.

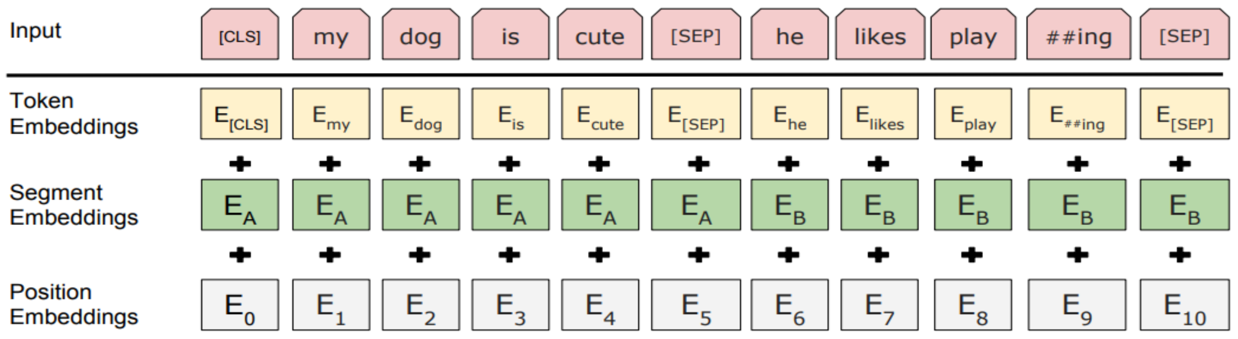


- Each Layer Contains:
    - Multi-Head Self-Attention: Lets every token consider information from every other token in the input simultaneously. All attention heads are computed in parallel and their outputs are concatenated and linearly projected.
    - Feedforward Network: After self-attention, each position/node (token embedding) passes through a two-layer fully connected feedforward neural network with a non-linearity (typically GELU activation).
    - Residual Connections & LayerNorm: The outputs of both the self-attention and feedforward sublayers are each added to their respective inputs (residual/skip connection) and normalized (LayerNorm), stabilizing training and improving optimization.

Traditional models (like left-to-right language models) only use context from earlier (left) positions. BERT's bidirectionality allows each token to attend to all other tokens on both its left and right at every layer, thanks to self-attention and the way the masked language modeling task is formulated. This enables BERT to capture both past and future context for every word simultaneously, resulting in richer, more context-sensitive embeddings.

BERT encodes each input sequence using three types of embeddings, which are summed:

1. Token Embeddings:
    - Each input word is split into subword units with the WordPiece tokenizer (e.g., "playing" -> "play", "##ing"). Each token/subword is mapped to a learned embedding vector.
2. Segment Embeddings:
    - To handle sentence pairs (for tasks like next sentence prediction or QA), BERT adds a learned embedding indicating Sentence A/B… to each token, letting the model differentiate their roles in multi-sentence tasks.
3. Position Embeddings:
    - Since Transformers don't have any inherent notion of sequence order, a unique positional embedding is added to each token's representation depending on its position in the sequence, allowing the model to capture order information.


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}

Example input pipeline:

- The structure looks like:
  [CLS] sentence_A_tokens [SEP] sentence_B_tokens [SEP]
  Where [CLS] is a special classification token at the start, and [SEP] separates sentences or marks the end.

Embedding Type	**Purpose**	**Example Value**
Token Embeddings	Semantic of each subword/token	"play", "##ing"
Segment Embeddings	Distinguish sentence A/B for paired tasks	0 for A, 1 for B
Position Embeddings	Encode order in sequence	0, 1, 2, …, 511

Pre-training Tasks

BERT is pre-trained on two unsupervised tasks:

1. Masked Language Modeling (MLM)

- Random Token Masking: For each input sentence (or sentence pair), 15% of the tokens are selected at random. Of these,

  - 80% are replaced with a special [MASK] token,

  - 10% are replaced with a random word,

  - 10% are left unchanged.

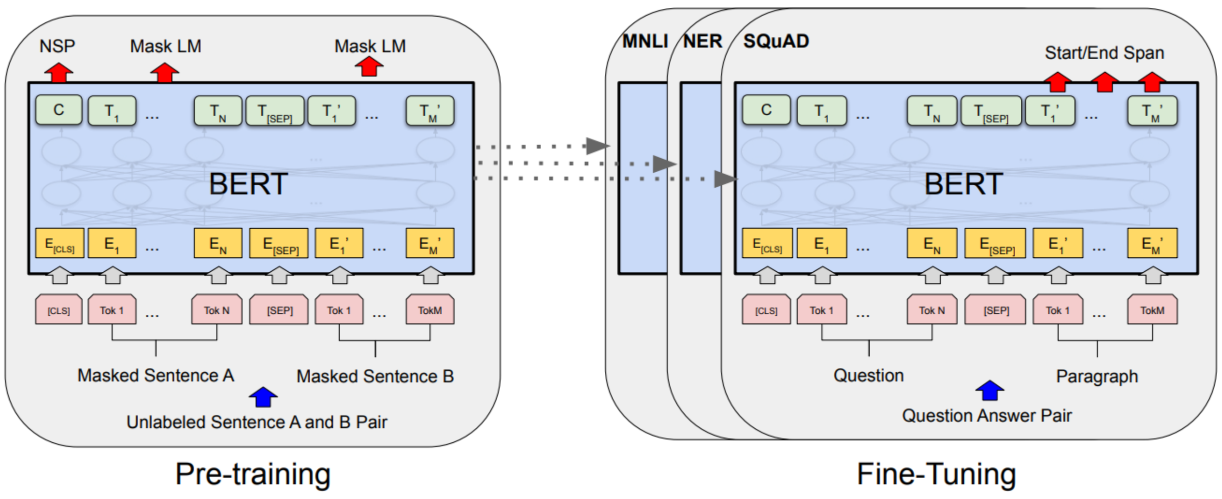
This makes the model robust to mismatches between pre-training and fine-tuning.

- Prediction Objective: The model receives the modified sequence and is trained to predict the original words at the masked positions using information from all the other words in the sequence - both to the left and to the right of every masked token.
- By predicting missing words using the full context, BERT learns truly bidirectional, deeply contextualized representations for each token, capturing subtle dependencies within language rather than just what comes before or after.

2. Next Sentence Prediction (NSP)

- Pair Construction: Input is constructed as pairs of sentences (A, B):
    - For 50% of pairs, B is the actual next sentence that follows A in the source text.
    - For the other 50%, B is a randomly selected sentence from the corpus, unrelated to A.
- Prediction Objective: The model is given the pair [CLS] A [SEP] B [SEP] and learns to classify whether sentence B is the true next sentence after A ("IsNext") or a random one ("NotNext").
- This encourages BERT to learn the relationship between sentences, enabling it to understand discourse, context flow, and perform well on tasks requiring comprehension of multiple sentences, such as question answering, natural language inference, and passage ranking.

Fine-tuning



- For downstream tasks, the pre-trained BERT model is extended with a minimal output layer (e.g., classification head), and the whole network is fine-tuned jointly using task-specific labeled data.

- Fine-tuning is data-efficient, yielding excellent results even when only small supervised datasets are available.

Sample Pretraining and Fine-Tuning for the NLP Task

Pretraining and Fine-Tuning BERT for Named Entity Recognition (CoNLL-2003, English):

1. Pre-training BERT, and its 2 methods, has been discussed above.

Eg. Loss calculation:

Assume 2 sentences are drawn from a corpus:

- Sentence A: the cat sat on the mat
- Sentence B: it was very comfortable

After token formatting, this will be:
[CLS] the cat sat on the mat [SEP] it was very comfortable [SEP]

After random masking, the tokenized list of words will represent:

[CLS] the cat [MASK] on the mat [SEP] it was [MASK] comfortable [SEP]

This set of tokens and their list of embeddings is passed through repetitive attention, feedforward and residual networks until the feed forward is complete and outputs are generated.

Assume:

- For [MASK] at "sat" position, model assigns: "sat" p = 0.7 (the correct answer)
- For [MASK] at "very" position, model assigns: "very" p = 0.15 (the correct answer)
- The embedding for the [CLS] token (post-attention positional + semantic embedding) is passed through a classification layer that predicts whether sentence B follows sentence A. Suppose the model assigns “isNext” probability to 0.92 (incorrect).

$$\text{“}sat\text{” } loss \;=\; -1. log(prob. correct\ word) \;+ 0 \;=\; -log(0.7) \;=\; 0.357$$

$$\text{“}very\text{” } loss \;=\; -1. log(prob. correct\ word) + 0 = -log(0.15) \;=\; 1.897$$

Therefore total MLM loss = 0.357 + 1.897 = 2.254.

The NSP loss is: (y is 1 for IsNext and 0 for NotNext):

$$-y\,(\log p(\,IsNext\ \mid\ [CLS]\,) + (1\ -\ y)\log p(\,NotNext\ \mid\ [CLS]\,))$$
$$=\ -log(0.92)\ =\ 0.083.$$

The Total loss is thus MLM + NSP loss = 2.254 (MLM) + 0.083 (NSP)=2.337 for this example.

The example loss is used to backpropagate and refine the model.

2. Fine-tuning BERT on CoNLL-2003 for NER

a. Prepare Data

- The CoNLL-2003 English data is formatted as token lines with label lines (eg. John B-PER).
- Convert data into the format required by BERT:
    - Tokenize each sentence using BERT's WordPiece tokenizer.
    - Align NER tags with the tokenized output.

b. Model Architecture

- Initialize pre-trained BERT (base or large).
- Add a token-level classification head: a linear layer on top of the Transformer outputs. The number of output units = number of unique NER tags.

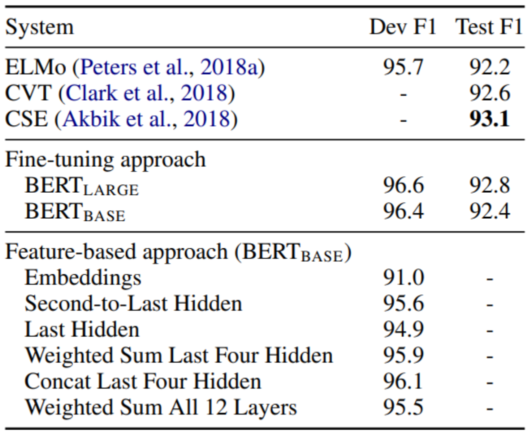
c. Training Setup

- Input: Tokenized sentences, attention masks, and label (tag) sequences.
- Loss Function: Cross-entropy loss calculated only for actual (non-padding) tokens.
- Optimizer: AdamW, with a low learning rate (in the order of e-5).
- Training: Standard mini-batch training for 5 epochs, using the training and validation splits for monitoring.

d. Evaluation & Metrics

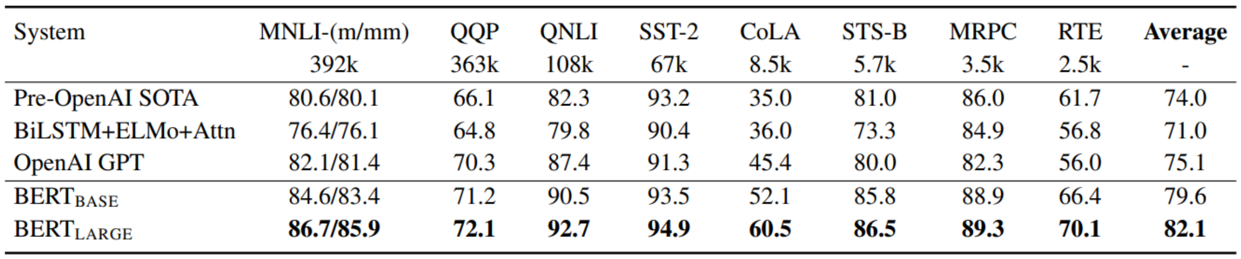
- During training, evaluate on the F1-score (the main metric for NER).
- Use early stopping or learning rate scheduling if needed (not necessary).

After fine-tuning, predict token labels on the test set, align subword predictions to original tokens, and compute precision, recall, and F1 scores. F1 scores for the BERT models are:



System	Dev F1	Test F1
ELMo (Peters et al., 2018a)	95.7	92.2
CVT (Clark et al., 2018)	-	92.6
CSE (Akbik et al., 2018)	-	**93.1**
Fine-tuning approach		
BERT$_{LARGE}$	96.6	92.8
BERT$_{BASE}$	96.4	92.4
Feature-based approach (BERT$_{BASE}$)		
Embeddings	91.0	-
Second-to-Last Hidden	95.6	-
Last Hidden	94.9	-
Weighted Sum Last Four Hidden	95.9	-
Concat Last Four Hidden	96.1	-
Weighted Sum All 12 Layers	95.5	-

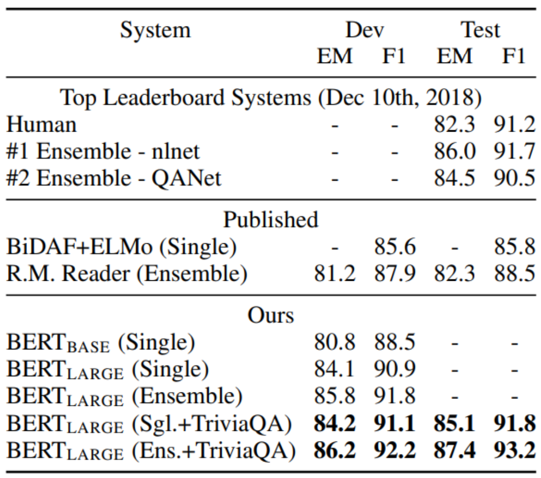
More Experimental Findings



System	MNLI-(m/mm)	QQP	QNLI	SST-2	CoLA	STS-B	MRPC	RTE	**Average**
	392k	363k	108k	67k	8.5k	5.7k	3.5k	2.5k	-
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.8	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	87.4	91.3	45.4	80.0	82.3	56.0	75.1
BERT$_{BASE}$	84.6/83.4	71.2	90.5	93.5	52.1	85.8	88.9	66.4	79.6
BERT$_{LARGE}$	**86.7/85.9**	**72.1**	**92.7**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**82.1**

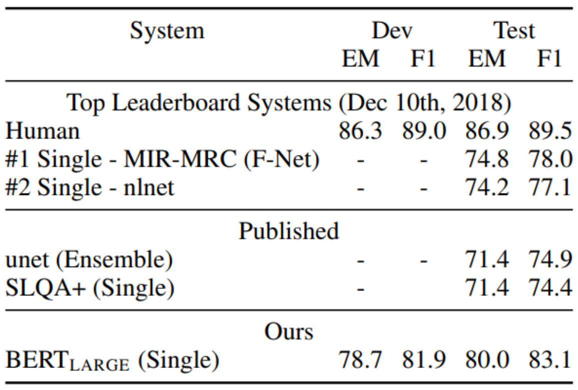
This table benchmarks various NLP systems (including BERT variants, OpenAI GPT, and earlier baselines) across language understanding tasks. Each column represents a different NLP task/dataset, such as:

- Natural language inference (MNLI, QNLI, RTE)
- Paraphrase detection (QQP, MRPC)
- Sentence classification (SST-2, CoLA)
- Semantic similarity (STS-B)



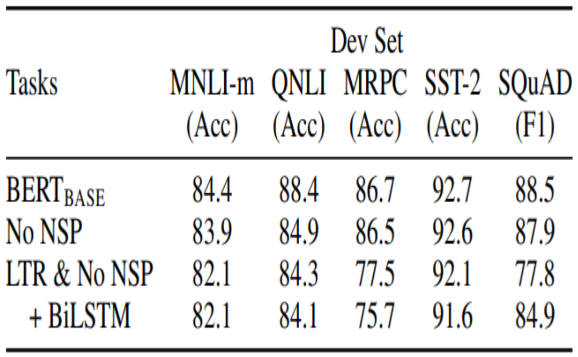
System	Dev EM	Dev F1	Test EM	Test F1
Top Leaderboard Systems (Dec 10th, 2018)				
Human	-	-	82.3	91.2
#1 Ensemble - nlnet	-	-	86.0	91.7
#2 Ensemble - QANet	-	-	84.5	90.5
Published				
BiDAF+ELMo (Single)	-	85.6	-	85.8
R.M. Reader (Ensemble)	81.2	87.9	82.3	88.5
Ours				
BERT$_{BASE}$ (Single)	80.8	88.5	-	-
BERT$_{LARGE}$ (Single)	84.1	90.9	-	-
BERT$_{LARGE}$ (Ensemble)	85.8	91.8	-	-
BERT$_{LARGE}$ (Sgl.+TriviaQA)	**84.2**	**91.1**	**85.1**	**91.8**
BERT$_{LARGE}$ (Ens.+TriviaQA)	**86.2**	**92.2**	**87.4**	**93.2**

SQuAD 1.1 results



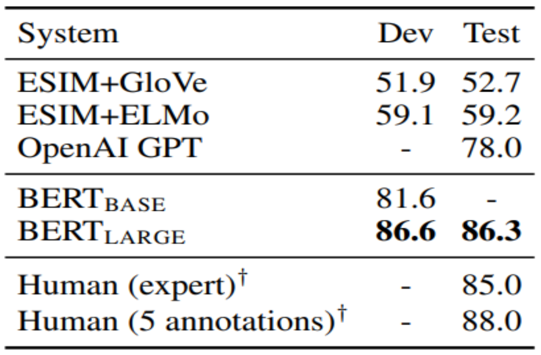
System	Dev EM	Dev F1	Test EM	Test F1
Top Leaderboard Systems (Dec 10th, 2018)				
Human	86.3	89.0	86.9	89.5
#1 Single - MIR-MRC (F-Net)	-	-	74.8	78.0
#2 Single - nlnet	-	-	74.2	77.1
Published				
unet (Ensemble)	-	-	71.4	74.9
SLQA+ (Single)	-		71.4	74.4
Ours				
BERT$_{LARGE}$ (Single)	78.7	81.9	80.0	83.1

SQuAD 2 results



Tasks	Dev Set MNLI-m (Acc)	QNLI (Acc)	MRPC (Acc)	SST-2 (Acc)	SQuAD (F1)
BERT$_{BASE}$	84.4	88.4	86.7	92.7	88.5
No NSP	83.9	84.9	86.5	92.6	87.9
LTR & No NSP	82.1	84.3	77.5	92.1	77.8
+ BiLSTM	82.1	84.1	75.7	91.6	84.9

Ablation report (LTR = left to right)



System	Dev	Test
ESIM+GloVe	51.9	52.7
ESIM+ELMo	59.1	59.2
OpenAI GPT	-	78.0
BERT$_{BASE}$	81.6	-
BERT$_{LARGE}$	**86.6**	**86.3**
Human (expert)†	-	85.0
Human (5 annotations)†	-	88.0

SWAG dev/test accuracies

BERT achieved new best results with large margins in numerous NLP tasks, including:

Task	**Dataset/Benchmark**	**Result**
Question Answering	SQuAD v1.1, v2.0	State-of-the-art F1
Natural Language Inference	MNLI, QNLI, RTE	Highest accuracy

Named Entity Recognition	CoNLL-2003, etc	SOTA F1
Sentence Classification	MRPC, SST-2, etc	SOTA accuracy/F1

The model's pre-training + fine-tuning setup surpassed previous results with a single unified architecture.

Impact

- Paradigm Shift: BERT made it standard to pre-train massive language models on generic tasks, then fine-tune for specific applications. This approach was derived into models like RoBERTa, ALBERT, and DistilBERT.
- Large Scale NLP: BERT's success accelerated research into even larger and more sophisticated models, including GPT-2/3, T5, and others.
- Easy Transferability: With minor changes, BERT could be adapted for classification, sequence tagging, QA, and other tasks across languages and domains.
- Open Source and Public Availability: Pre-trained BERT models were released, making powerful NLP readily accessible.
- Input Length Constraint: BERT's architecture limits the maximum input length (typically 512 tokens).
- Not Generative: The model does not generate text, limiting its direct use for text generation (unlike autoregressive models like GPT).

Conclusion

The introduction of BERT marked a major milestone in NLP research.

By thoroughly pre-training deep, bidirectional models and enabling simple, efficient transfer to any language task, BERT set a new standard for language understanding.

Its core ideas have become foundational in nearly all modern NLP systems and research as of 2025.

****