- Many models built upon BERT to address these limitations and improve performance:
- **RoBERTa (Facebook):** Improved training procedures, larger datasets, and dynamic masking. Generally outperformed BERT.
- **ALBERT (Google):** Parameter reduction techniques to make models smaller and faster.
- **ELECTRA (Google):** Used a more efficient pre-training objective (replaced token detection instead of masked token prediction).
- **DeBERTa (Microsoft):** Introduced disentangled attention mechanisms for better contextual understanding. Often considered one of the best BERT-style models.
- **Longformer and other Long-Context Models:** Addressed BERT's limitation with handling long sequences of text.
- **T5 (Text-to-Text Transfer Transformer):** Reframed all NLP tasks as text-to-text problems.
- **GPT-3 and later LLMs (OpenAI):** While different in architecture (decoder-only), these models, and especially their successors, represent a major shift in scale and capabilities. They're often used for zero-shot or few-shot learning, reducing the need for extensive fine-tuning.
-
- **The Rise of Large Language Models (LLMs):** The current focus is heavily on extremely large language models (LLMs) with billions or even trillions of parameters. These models, like GPT-4, PaLM, and others, demonstrate emergent abilities and can perform a wide range of tasks with minimal task-specific training. They often use variations of the Transformer architecture but are trained on massive datasets.

★ BERT was a foundational model and a critical step in the development of modern NLP. However, it's been surpassed by more recent advancements, particularly the rise of large language models. While BERT is still valuable to understand as a stepping stone,

★ Comparison ELMO VS BERT

- **ELMo: Shallow Concatenation** - ELMo has two separate LSTMs, one reading the text left-to-right and the other right-to-left. It then combines the outputs of these two LSTMs, but this combination happens *after* each LSTM has already processed the entire sequence independently. Think of it like two separate experts giving their opinions and then someone briefly summarizing both.
- **BERT: Deeply Bidirectional** - BERT processes the entire input sequence at once, and *every layer* of the Transformer has access to information from both the left and the right context. It's not just combining separate left-to-right and right-to-left analyses; instead, the model is constantly considering the entire context as it builds its representation. Think of it like a group of experts constantly discussing and debating the meaning of the text together.

BERT's bidirectionality is integrated throughout the entire model, while ELMo's is more of a post-processing step. This allows BERT to learn more complex and nuanced relationships between words and their context.

★ two-stage framework: pre-training and fine-tuning. During pre-training

pre-training

the model learns from unlabeled data using tasks like masked language modeling and next sentence prediction.

Fine-Tuning

the pre-trained parameters are used to initialize the model, and then all parameters are adjusted using labeled data from specific downstream tasks.

Model Architecture

Two model sizes are used: BERTBASE (12 layers, 110M parameters) and BERTLARGE (24 layers, 340M parameters). BERTBASE was designed to be the same size as OpenAI GPT for comparison ,, , BERT uses **bidirectional self-attention,**

CLS

Definition : The [CLS] token starts as a special token with a random (or pre-trained) embedding. As it passes through the BERT network, its vector representation is updated based on the entire input sequence, becoming a powerful representation of the sequence as a whole.

Input/output Representations

Input:
- BERT can handle single sentences or pairs of sentences as input.
- Input is tokenized using WordPiece embeddings (30,000 token vocabulary).
- Every input sequence starts with a special classification token [CLS].
- Sentences in a pair are separated by a special token [SEP].
- To distinguish between the two sentences in a pair, each token is given a "segment embedding" indicating whether it belongs to sentence A or sentence B.
- Each token also receives a "position embedding" to indicate its position in the sequence.
- The final input representation for a token is the sum of its WordPiece embedding, **segment embedding, and position embedding.**

Output:
- The final hidden state vector corresponding to the [CLS] token (denoted as C) is used as the aggregate representation of the *entire sequence* for classification tasks.
- The final hidden state vectors for the individual input tokens (denoted as Ti) can be used for token-level tasks.

Segment Embeddings

To help BERT distinguish between two different sentences when they are fed into the model as a *single* input sequence.

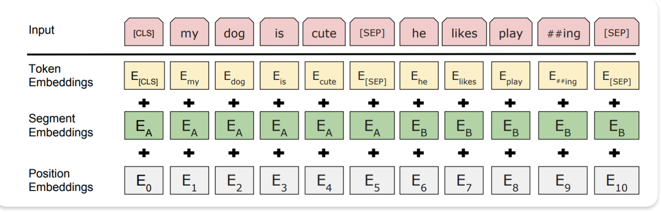
What it is: The segment embedding is just a learned vector of numbers. All tokens belonging to sentence A get the same vector added to their representation, and all tokens belonging to sentence B get a *different* vector added. The model learns what these vectors should be during training.

- **Segment Embeddings:** Like wearing different colored shirts to show which team you're on (sentence A or sentence B).
- **Position Embeddings:** Like having a number pinned to your shirt to show your place in line (the order of the tokens).

Position Embeddings

To tell BERT the *order* of the tokens in the sequence.

What it is: The position embedding is a learned vector of numbers, just like the segment embedding. Each position in the sequence has its own unique position embedding vector. The model learns these vectors during training.



Pre-training BERT

Task #1 :Masked LM

- **Goal:** To train a deep bidirectional language model (assumed to be more powerful than left-to-right or shallow concatenated models).
- **Method:** Mask a certain percentage (15%) of input tokens at random and train the model to predict those masked tokens.
- **Advantages:**
- Enables training with deep bidirectional context.
- Effective for pre-training Transformer encoders (bidirectional).

- **Implementation:**
- The final hidden vectors corresponding to the mask tokens are fed into an output softmax over the vocabulary for prediction.
- Instead of always replacing masked tokens with the [MASK] token, the i-th token is replaced with: 80% [MASK] token, 10% a random token, and 10% the original i-th token.
-

MLM Limitations

- **Mismatch between Pre-training and Fine-tuning:**
- The [MASK] token is used during pre-training, but it does not appear during fine-tuning.
- This means the model's learning during pre-training may not be fully applicable during fine-tuning, potentially leading to performance degradation.

- **Mitigation:**
- Instead of always replacing masked words with the [MASK] token, they are sometimes replaced with random tokens or the original token to prevent the model from over-relying on the specific token.

Task #2 :Next Sentence Prediction

- **Goal:** To train a model that understands sentence relationships.
- **Method:** Pre-train the model on a binarized next sentence prediction task.
- **Implementation:**
- For each pre-training example, select two sentences, A and B.
- 50% of the time, B is the actual sentence that follows A (labeled "IsNext").
- 50% of the time, B is a random sentence from the corpus (labeled "NotNext").
- The model then predicts whether B is the next sentence after A.
-
- **Benefit:** Pre-training on this task is shown to be very beneficial for both QA and NLI tasks.

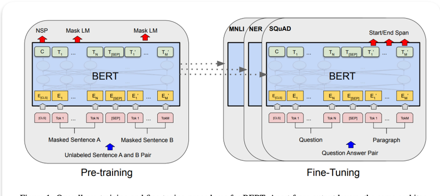


Figure 1: Overall pre-training and fine-tuning procedures for BERT. Apart from output layers, the same architectures are used in both pre-training and fine-tuning. The same pre-trained model parameters are used to initialize models for different down-stream tasks. During fine-tuning, all parameters are fine-tuned. [CLS] is a special symbol added in front of every input example, and [SEP] is a special separator token (e.g. separating questions/answers).

Fine-training BERT

- **Unified Approach:** BERT uses **self-attention** to unify the independent encoding of text pairs and bidirectional cross-attention, unlike previous methods. Encoding a concatenated text pair with self-attention effectively includes bidirectional cross-attention between the two sentences.
- **End-to-End Fine-tuning:** Task-specific inputs and outputs are plugged into BERT, and all parameters are fine-tuned end-to-end.

Experiments

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**

GLUE

- **Dominant Performance:** BERT significantly outperformed previous state-of-the-art models on the GLUE benchmark, demonstrating its superior capabilities in various natural language understanding tasks. Notably, it showed impressive performance gains even in data-scarce scenarios.

- **Importance of Model Size:** A larger BERT model (BERTLARGE) leads to improved performance. This indicates that model size plays a crucial role in natural language understanding capabilities.

- **Structural Similarity:** BERTBASE shares a nearly identical model structure with OpenAI GPT, except for the attention masking technique. This suggests that attention masking is a key factor contributing to BERT's performance.

- **Conclusion:** BERT is a powerful model that can be effectively applied to a wide range of natural language understanding tasks. The fine-tuning approach leveraging the [CLS] token, along with model size, significantly impacts BERT's performance.

SQuAD v1.1+SQuAD v2.0

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**

Table 2: SQuAD 1.1 results. The BERT ensemble is 7x systems which use different pre-training checkpoints and fine-tuning seeds.

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

Table 3: SQuAD 2.0 results. We exclude entries that use BERT as one of their components.

- **Excellent Performance of BERT:** BERT models (BERTBASE, BERTLARGE) demonstrated outstanding performance, surpassing previous state-of-the-art models on both SQuAD v1.1 and v2.0.
- **Importance of Fine-tuning:** By fine-tuning on the SQuAD datasets, BERT was able to maximize its question answering abilities.
- **Benefits of TriviaQA:** Pre-training (additional fine-tuning before the main fine-tuning) using the TriviaQA dataset had a positive impact on SQuAD performance.

SQuAD v1.1 Results:
- **Outperforming the Leaderboard:** BERT achieved higher F1 scores than the top systems on the SQuAD v1.1 leaderboard, both in ensemble and single-model configurations.
- **Power of a Single Model:** Notably, the single BERT model achieved a higher F1 score than the top ensemble system.

SQuAD v2.0 Results:
- **Handling Unanswerable Questions:** SQuAD v2.0 includes questions with no answer, requiring models to determine whether a question is answerable. BERT effectively handled these questions by utilizing the [CLS] token.
- **Surpassing Previous Best Models:** BERT also achieved higher F1 scores than previous state-of-the-art models on SQuAD v2.0, demonstrating its superior performance.

SWAG

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

Table 4: SWAG Dev and Test accuracies. †Human performance is measured with 100 samples, as reported in the SWAG paper.

- BERT answered questions much more accurately than other models!
- In particular, it was over 27% more accurate than the existing ESIM+ELMo model!
- It even performed over 8% better than OpenAI GPT!

Model size

#L	#H	#A	LM (ppl)	MNLI-m	MRPC	SST-2
Hyperparams				Dev Set Accuracy		
3	768	12	5.84	77.9	79.8	88.4
6	768	3	5.24	80.6	82.2	90.7
6	768	12	4.68	81.9	84.8	91.3
12	768	12	3.99	84.4	86.7	92.9
12	1024	16	3.54	85.7	86.9	93.3
24	1024	16	3.23	86.6	87.8	93.7

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.

compared the performance of several BERT models, trained using the **same dataset**, but with **different sizes**. In other words

★ Learnings :

- **Bigger is Better!** As the size of the BERT model (number of layers, hidden units, attention heads) increases, performance on GLUE tasks steadily improves.
- **Effective Even with Limited Data!** Increasing the model size helps improve performance even on tasks with very little training data (MRPC).
- **Much Larger than Existing Models!** BERT has a much larger scale than models used in previous research.
- **Pre-training is Crucial!** In order for increasing the model size to be effective, the model must be sufficiently pre-trained.

Feature-Based Approach

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	-

Table 7: CoNLL-2003 Named Entity Recognition results. Hyperparameters were selected using the Dev set. The reported Dev and Test scores are averaged over 5 random restarts using those hyperparameters.

BERT's Performance:
- BERTLARGE achieved very high performance when using fine-tuning, with a Dev F1 score of 96.6 and a Test F1 score of 92.8. This is a superior result compared to existing models such as ELMo, CVT, and CSE.

Conclusion:
BERT is effective for NER tasks in both fine-tuning and feature-based approaches. Fine-tuning provides the highest performance, but the feature-based approach, when using a combination of features from multiple layers, can achieve performance comparable to fine-tuning. This demonstrates that BERT can be flexibly applied to a variety of NLP tasks.