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

Abstract

BERT, which stands for **B**idirectional **E**ncoder **R**epresentations from **T**ransformers.

- BERT is designed to
  - pretrain deep bidirectional representations from unlabeled text by jointly **conditioning** on both **left** and **right** context in all layers
  - As a result, the pre-trained BERT model can be finetuned with just one additional output layer
    - question answering
    - language inference
  - without substantial task-specific architecture modifications.

1 Introduction

- Language model pre-training has been shown to be effective for improving many natural language processing tasks
- There are **two** existing strategies for **applying pre-trained** language representations to downstream tasks
  - feature-based
    - such as ELMo

    - fine-tuning
        - the Generative Pre-trained Transformer (OpenAI GPT)
        - introduces minimal task-specific parameters
        - and is trained on the downstream tasks by simply fine-tuning all pretrained parameters

> 🌟 The two approaches share the same objective function during pre-training, where they use unidirectional language models to learn general language representations

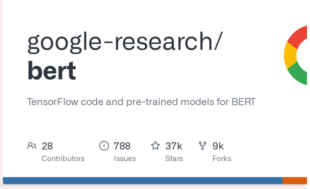
- We argue that current techniques restrict the power of the pre-trained representations
- especially for the fine-tuning approaches.
- The **major limitation** is that standard language models are **unidirectional**
    - and this **limits the choice of architectures** that can be used during pre-training
    - in OpenAI GPT
        - the authors use a **left-to-right** architecture
        - where **every token can only attend to previous tokens** in the self-attention layers of the Transformer
        - Such restrictions are **sub-optimal for sentence-level tasks**
        - could be very harmful when applying finetuning based approaches to token-level tasks
            - question answering
                - where it is crucial to incorporate context from both directions.
- In this paper, we improve the fine-tuning based approaches by proposing **BERT**
- BERT alleviates the previously mentioned unidirectionality constraint by using
    - “***masked language model***” (**MLM**) pre-training objective
    - The masked language model randomly masks some of the tokens from the input
    - and the objective is to predict the original vocabulary id of the masked word based only on its context
    - Unlike left-toright language model pre-training
    - the MLM objective enables the representation to fuse the left and the right context
    - which allows us to **pretrain a deep bidirectional Transformer**
- we also use a “next sentence prediction” task
    - that jointly pretrains text-pair representations.
- The contributions of our paper are as follows:
    - the importance of bidirectional pre-training for language representations.
        - Unlike
            - The use of unidirectional language models for pre-training

        - In n contrast to methods that
            - use a shallow concatenation of independently trained left-to-right and right-to-left LMs.
    - We show that pre-trained representations reduce the need for many heavily-engineered task-specific architectures
    - BERT is the first finetuning based representation model that achieves state-of-the-art performance on a large suite
      of sentence-level and token-level tasks outperforming many task-specific architectures.
    - BERT advances the state of the art for eleven NLP tasks.

      GitHub - google-research/bert: TensorFlow code and pre-trained models for BERT

      TensorFlow code and pre-trained models for BERT. Contribute to google-research/bert development by creating an account on GitHub.

      https://github.com/google-research/bert/tree/master



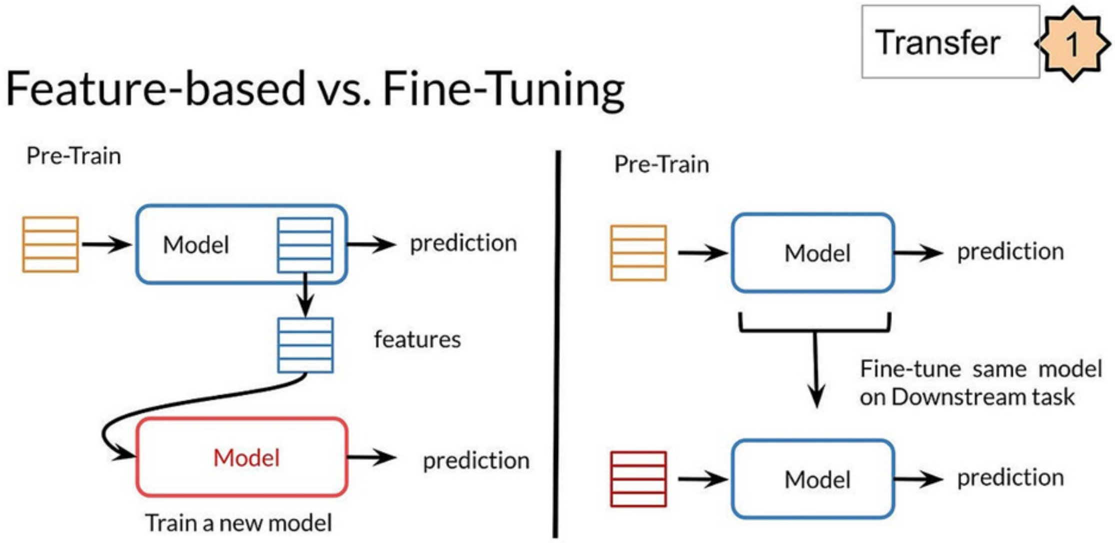
2. Related Work

2.1 Unsupervised Feature-based Approaches

> ***Learning widely applicable representations of words has been an active area of research for decades***
>
> - including non-neural methods
> - and neural methods

- Pre-trained word embeddings are an integral part of modern NLP systems
- offering significant improvements over embeddings learned from scratch
- To pretrain word embedding vectors, left-to-right language modeling objectives have been used
- as well as objectives to discriminate correct from incorrect words in left and right context
- ELMo and its predecessor generalize traditional word embedding research along a different dimension.
- They extract context-sensitive features from a left-to-right and a right-to-left language model
- The contextual representation of each token is the concatenation of the left-to-right and right-to-left representations.

🌟 integrating contextual word embeddings with existing task-specific architectures, ELMo advances the state of the art for several major NLP



2.2 Unsupervised Fine-tuning Approaches

> 🌟 More recently, sentence or document encoders which produce contextual token representations have been pre-trained from unlabeled text and fine-tuned for a supervised downstream task

- The advantage of these approaches is that few parameters need to be learned from scratch

2.3 Transfer Learning from Supervised Data

- There has also been work showing effective transfer from supervised tasks with large datasets, such as
    - natural language inference
    - machine translation

BERT

- There are two steps in our framework:
    - pre-training
        - the model is trained on ***unlabeled*** data over different pre-training tasks.
    - fine-tuning
        - the BERT model is first **initialized** with the pre-trained parameters
        - and **all of the parameters** are **fine-tuned using labeled data** from the downstream tasks.
- Each downstream task has separate fine-tuned models
    - even though they are initialized with the same pre-trained parameters.
- A distinctive feature of BERT is its **unified architecture across different tasks**

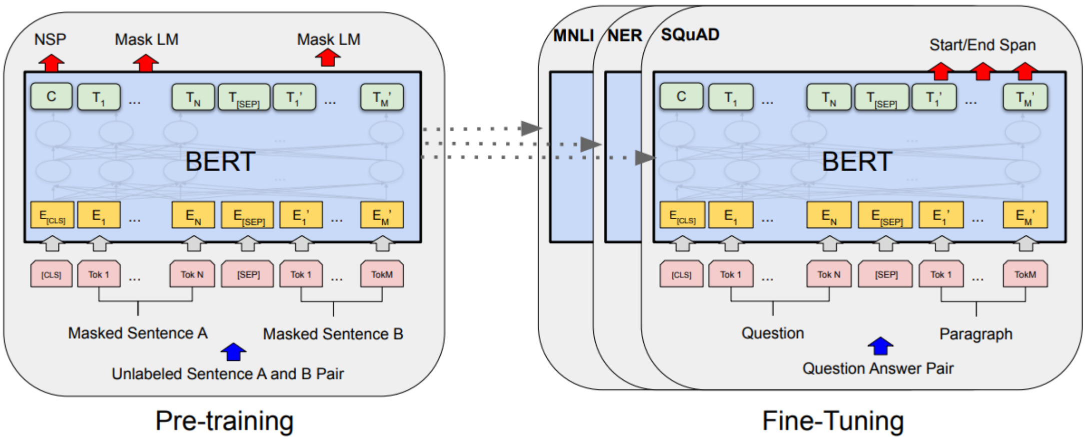


Figure 1: Overall pre-training and fine-tuning procedures for BERT. Apart from output layers, the same architectures are used in both pre-training and fine-tuning. The same pre-trained model parameters are used to initialize models for different down-stream tasks. During fine-tuning, all parameters are 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).

- Model Architecture
  - multi-layer bidirectional Transformer encoder
- the number of layers (i.e., Transformer blocks) as L
- the hidden size as H
- the number of self-attention heads as A
- We primarily report results on two model sizes
  - $BERT_{BASE}$
    - $L = 12$, $H = 768$, $A = 12$
    - Total parameters $110M$
  - $BERT_{LARGE}$
    - $L = 24$, $H = 1024$, $A = 16$
    - Total parameters $340M$
- $BERT_{BASE}$
  - was chosen to have the same model size as OpenAI GPT for comparison purposes.
- BERT Transformer uses **bidirectional self-attention**
- while the GPT Transformer uses **constrained self-attention** where every token can only attend to context to its left
- Input/Output Representations
  - To make BERT handle a variety of down-stream tasks
  - our input representation is able to unambiguously represent both a
    - single sentence

        - and a pair of sentences
            - $< Question, Answer >$ *pair in one token sequence*
    - Throughout this work

    > 🌟 a “***sentence***” can be an arbitrary span of contiguous text, rather than an actual linguistic sentence.

    > 🌟 A “***sequence***” refers to the input token sequence to BERT

- We use $WordPiece$ embeddings with a $30,000$ **token vocabulary**
- The first token of every sequence is always a special classification token ($[CLS]$).
    - The final hidden state corresponding to this token is used as the aggregate sequence representation for classification
      tasks.
- Sentence pairs are packed together into a single sequence.
- We differentiate the sentences in two ways.
    - First, we separate them with a special token ($[SEP]$)
    - Second, we add a learned embedding to every token indicating whether it belongs to **sentence** A or **sentence** B
    - we denote input embedding as E
    - the final hidden vector of the special $[CLS]$ token as $C \in \mathbb{R}^H$ and and the final hidden vector for the i^{th} input token as $T_i \in \mathbb{R}^H$

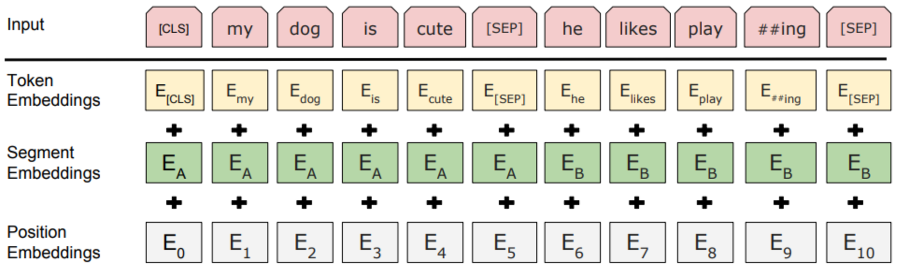

Input
[CLS]
my
dog
is
cute
[SEP]
he
likes
play
##ing
[SEP]
Token Embeddings
$E_{[CLS]}$ E_{my} E_{dog} E_{is} E_{cute} $E_{[SEP]}$ E_{he} E_{likes} E_{play} $E_{\#\#ing}$ $E_{[SEP]}$
Segment Embeddings
E_A E_A E_A E_A E_A E_A E_B E_B E_B E_B E_B
Position Embeddings
E_0 E_1 E_2 E_3 E_4 E_5 E_6 E_7 E_8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

Figure 2: BERT input representation. The input embeddings are the sum of the token embeddings, the segmentation embeddings and the position embeddings.

> 🌟 For a given token, its input representation is constructed by summing the corresponding **token**, **segment**, and **position embeddings**

3.1 Pre-training BERT

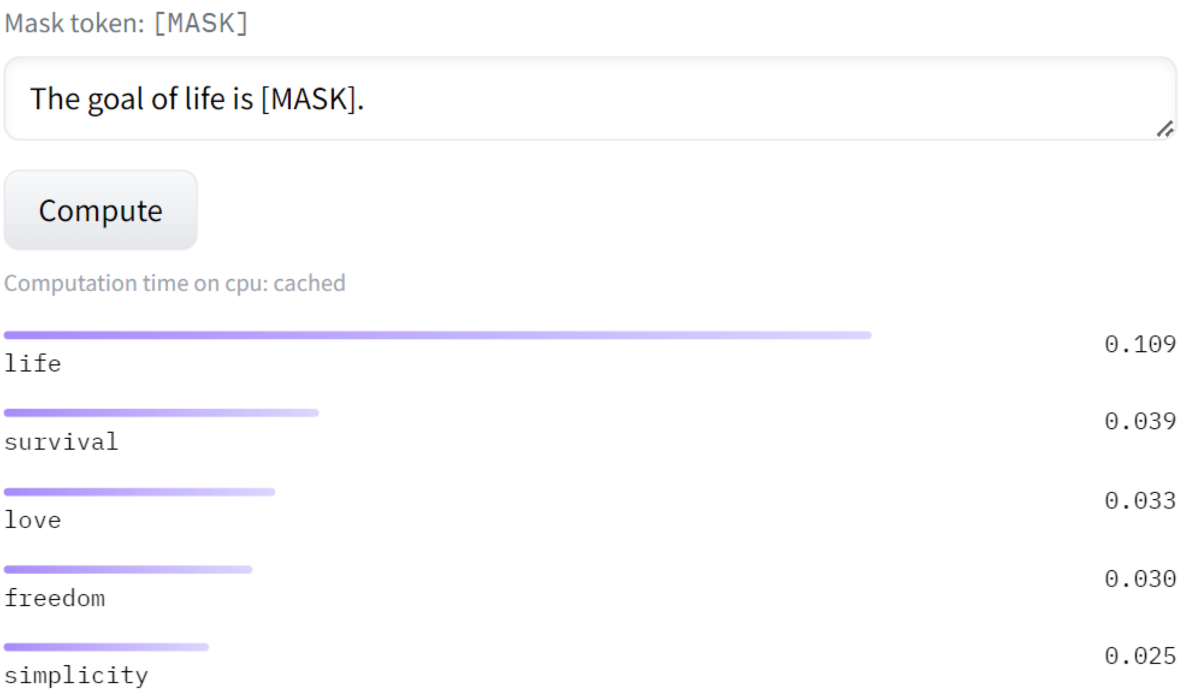
- we pre-train BERT using two unsupervised tasks

Task #1: Masked LM

- **Unfortunately**, standard conditional language models can only be trained left-to-right or right-to-left
- since **bidirectional conditioning** would allow each word **to indirectly "see itself"**
  - and the model could trivially predict the target word in a multi-layered context
- In order to train a deep bidirectional representation
- we simply **mask some percentage of the input tokens** at random
- then predict those masked tokens.
- In this case, the final hidden vectors corresponding to the mask tokens are fed into an output softmax over the vocabulary
- a downside is that we are creating a mismatch between pre-training and fine-tuning
  - since the $[MASK]$ token does not appear during fine-tuning.

> 🌟 To mitigate this, we do not always replace "**masked**" words with the actual $[MASK]$ token

- The training data generator chooses 15% of the token positions at random for prediction
- If the $i-th$ token is chosen, we replace the $i-th$ token with
  1. the $[MASK]$token 80% of the time
  2. a random token 10% of the time
  3. the unchanged $i-th$ token 10% of the time.
- Then, T_i will be used to predict the original token with cross entropy loss



Mask token: [MASK]

The goal of life is [MASK].

Computation time on cpu: cached

life	0.109
survival	0.039
love	0.033
freedom	0.030
simplicity	0.025

> **Masked language modeling (MLM)**: taking a sentence, the model randomly masks 15% of the words in the input then run the entire masked sentence through the model and has to predict the masked words.

Task #2: Next Sentence Prediction (NSP)

> **Next sentence prediction (NSP)**: the models concatenates two masked sentences as inputs during pretraining. Sometimes they correspond to sentences that were next to each other in the original text, sometimes not. The model then has to predict if the two sentences were following each other or not.

- Tasks like
  - Question Answering (QA)
  - Natural Language Inference (NLI)

  are based on understanding the relationship between two sentences
- In order to train a model that **understands sentence relationships**
  - we pre-train for a binarized next sentence prediction task
  - that can be trivially generated from any monolingual corpus.
- when choosing the sentences A and B for each pretraining example
  - 50% of the time B is the actual next sentence that follows A (labeled as `IsNext`)
  - the rest of the time it is a random sentence from the corpus (labeled as `NotNext`)

- Pre-training data
    - use the BooksCorpus (800M words)
    - English Wikipedia (2,500M words)
        - For Wikipedia we extract only the text passages and ignore lists, tables, and headers.
    - It is critical to use a document-level corpus rather than a shuffled sentence-level corpus such as the Billion
      Word Benchmark

3.2 Fine-tuning BERT

- For each task, we simply plug in the tasks-pecific inputs and outputs into BERT and finetune all the parameters end-to-end
- At the input, **sentence** $A$and **sentence** B from pre-training are analogous to
    1. sentence pairs in paraphrasing
    2. **hypothesis-premise pairs in entailment**

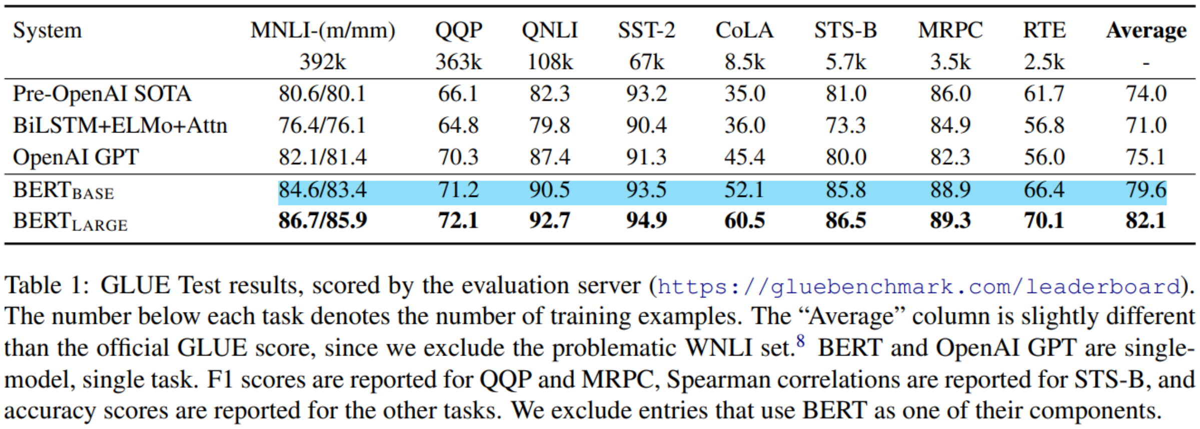
       NLI

    3. question-passage pairs in question answering
    4. a degenerate text-∅ pair in text classification or sequence tagging.
- Compared to pre-training, fine-tuning is relatively inexpensive

4 Experiments

4.1 GLUE

- The General Language Understanding Evaluation (GLUE) benchmark
    - is a collection of diverse natural language understanding tasks

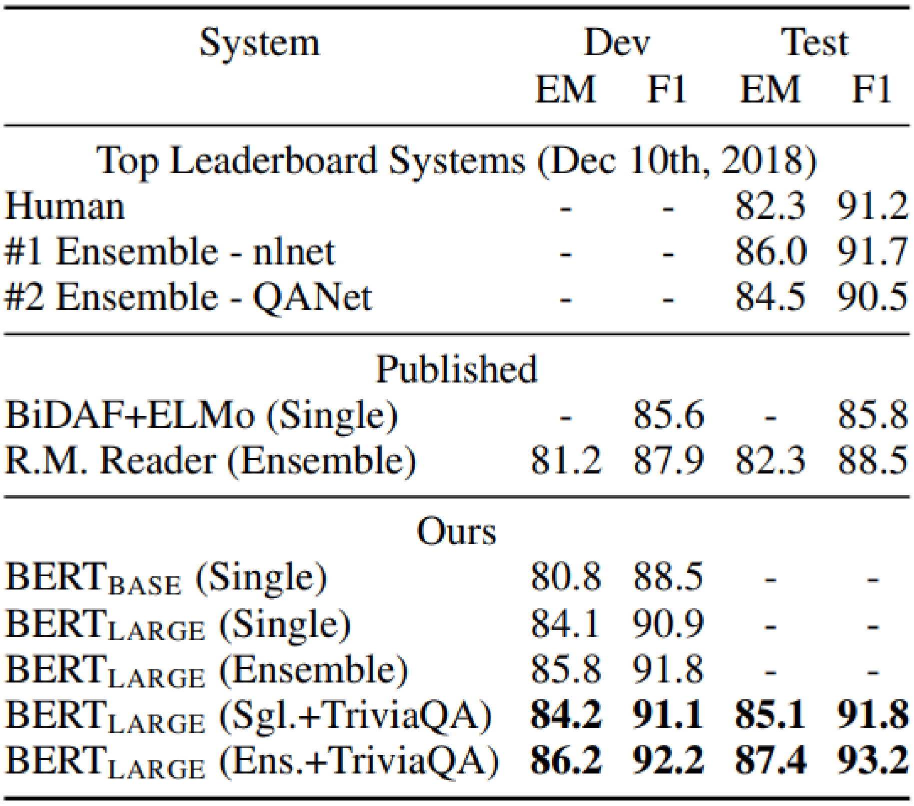


System	MNLI-(m/mm)	QQP	QNLI	SST-2	CoLA	STS-B	MRPC	RTE	**Average**
	392k	363k	108k	67k	8.5k	5.7k	3.5k	2.5k	-
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.8	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	87.4	91.3	45.4	80.0	82.3	56.0	75.1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

Table 1: GLUE Test results, scored by the evaluation server (https://gluebenchmark.com/leaderboard). 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.[8] BERT and OpenAI GPT are single-model, single task. F1 scores are reported for QQP and MRPC, Spearman correlations are reported for STS-B, and accuracy scores are reported for the other tasks. We exclude entries that use BERT as one of their components.

4.2 SQuAD v1.1

- The Stanford Question Answering Dataset (SQuAD v1.1)
  - is a collection of 100k crowdsourced question/answer pairs
  - Given a question and a passage from Wikipedia containing the answer
    - the task is to predict the answer text span in the passage



System	Dev		Test	
	EM	F1	EM	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$_{\text{BASE}}$ (Single)	80.8	88.5	-	-
BERT$_{\text{LARGE}}$ (Single)	84.1	90.9	-	-
BERT$_{\text{LARGE}}$ (Ensemble)	85.8	91.8	-	-
BERT$_{\text{LARGE}}$ (Sgl.+TriviaQA)	**84.2**	**91.1**	**85.1**	**91.8**
BERT$_{\text{LARGE}}$ (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.

5 Ablation Studies

5.3 Feature-based Approach with BERT

- feature-based approach, where fixed features are extracted from the pretrained model has certain advantages:
  - Not all tasks can be easily represented by a Transformer encoder architecture
    - and therefor task-specific model architecture to be added
  - there are major **computational benefits** to pre-compute an expensive representation of the training data once and then run many experiments

  - with **cheaper models** on top of this representation

- we apply the feature-based approach by extracting the activations from one or more layers without fine-tuning any parameters of BERT
- These contextual embeddings are used as input to a randomly initialized two-layer 768-dimensional BiLSTM before the classification layer

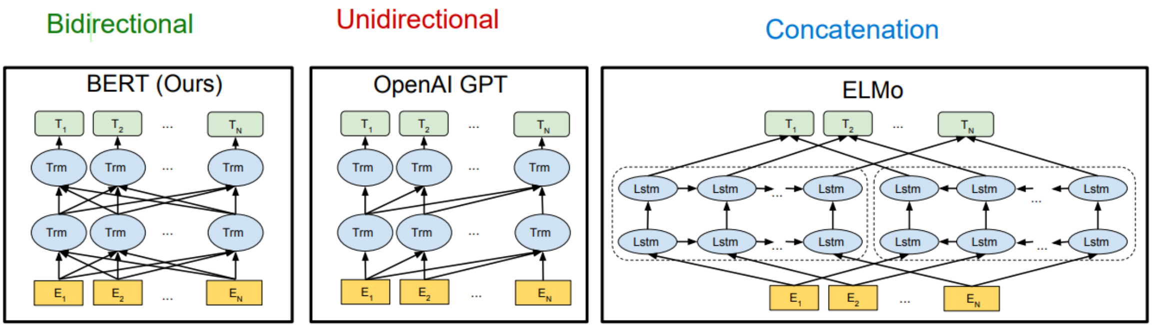


Figure 3: Differences in pre-training model architectures. BERT uses a bidirectional Transformer. OpenAI GPT uses a left-to-right Transformer. ELMo uses the concatenation of independently trained left-to-right and right-to-left LSTMs to generate features for downstream tasks. Among the three, only BERT representations are jointly conditioned on both left and right context in all layers. In addition to the architecture differences, BERT and OpenAI GPT are fine-tuning approaches, while ELMo is a feature-based approach.