BERT

Bidirectional Encoder Representations from Transformers

- ❖ ***BERT***'s key technical innovation is applying the ***bidirectional training of Transformer***.
- ❖ BERT is a popular ***attention model, for language modeling***.
- ❖ The paper's results show that a language model which is bi-directionally trained can have a deeper sense of language context and flow than single-direction language models.

- ❖ In the paper, the researchers detail a novel technique named Masked LM (MLM) which allows bidirectional training in models in which it was previously impossible.

❖ Background

In the field of computer vision, researchers have repeatedly shown the value of transfer learning — pre-training a neural network model on a known task, for instance, ImageNet, and then performing fine-tuning — using the trained neural network as the basis of a new purpose-specific model. In recent years, researchers have been showing that a similar technique can be useful in many natural language tasks.

❖ How BERT works

BERT makes use of **Transformer, an attention mechanism that learns contextual relations between words (or sub-words) in a text**. In its vanilla form, **Transformer** includes two separate mechanisms — an

- ❖ **Encoder** that reads the text input.
- ❖ **Decoder** that produces a prediction for the task.

Since BERT's goal is to **generate a language model**, only the **encoder** mechanism is necessary.

As opposed to directional models, which read the text input sequentially (left-to-right or right-to-left), the **Transformer encoder reads the entire sequence of words at once**.

Therefore it is considered **bidirectional**, though it would be more accurate to say that it's **non-directional**. This characteristic allows the model to learn the context of a word based on all of **its surroundings** (left and right of the word).

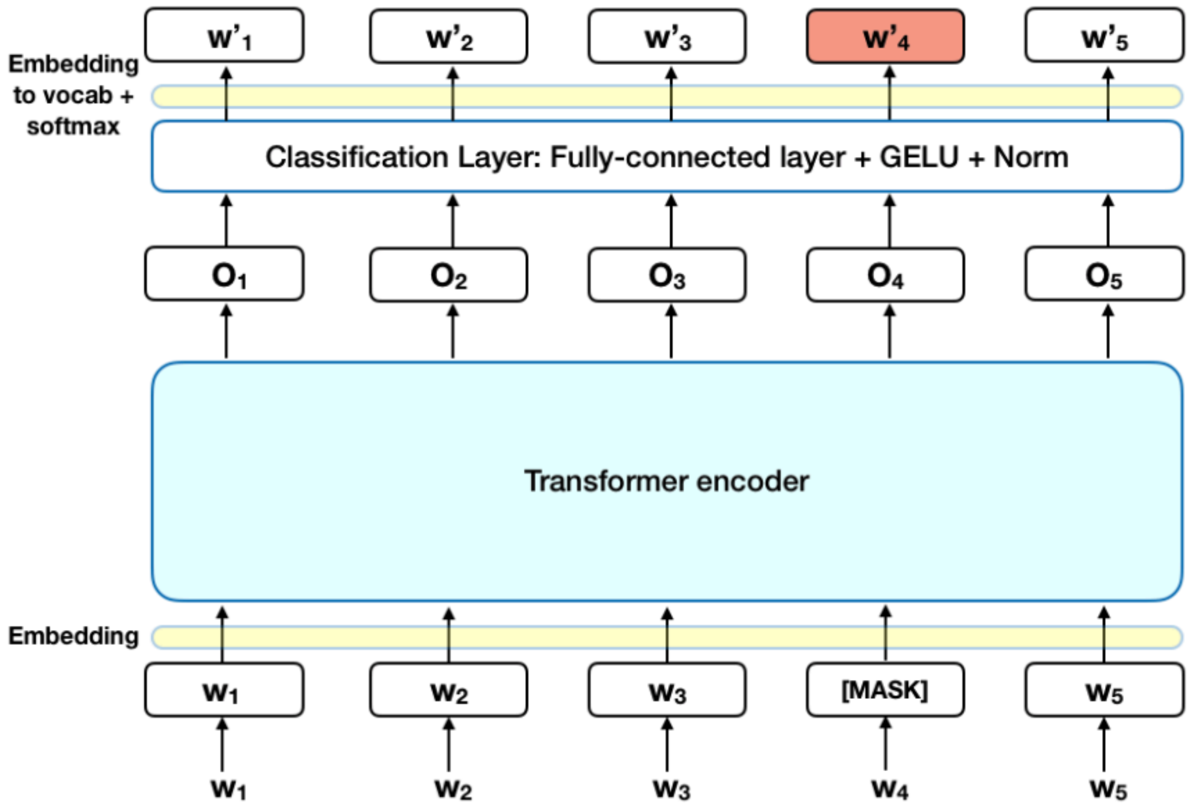
a directional approach that inherently **limits context learning**. To overcome this challenge, BERT uses two training strategies:

1. Masked LM (MLM):

Before feeding word sequences into BERT, ***15% of the words in each sequence are replaced with a [MASK] token***. The **model then attempts to predict the original value of the masked**

words, based on the context provided by the other, non-masked, words in the sequence. In technical terms, the prediction of the output words requires:

1. Adding a **classification layer** on top of the **encoder output**.
2. **Multiplying** the output vectors by the **embedding matrix**, transforming them into the **vocabulary dimension**.
3. Calculating the **probability** of each word in the **vocabulary** with **softmax**.


W'1
W'2
W'3
W'4
W'5
Embedding to vocab + softmax
Classification Layer: Fully-connected layer + GELU + Norm
O1
O2
O3
O4
O5
Transformer encoder
Embedding
W1
W2
W3
[MASK]
W5
W1
W2
W3
W4
W5

The BERT ***loss function*** takes into consideration only the ***prediction of the masked values and ignores the prediction of the non-masked words***.

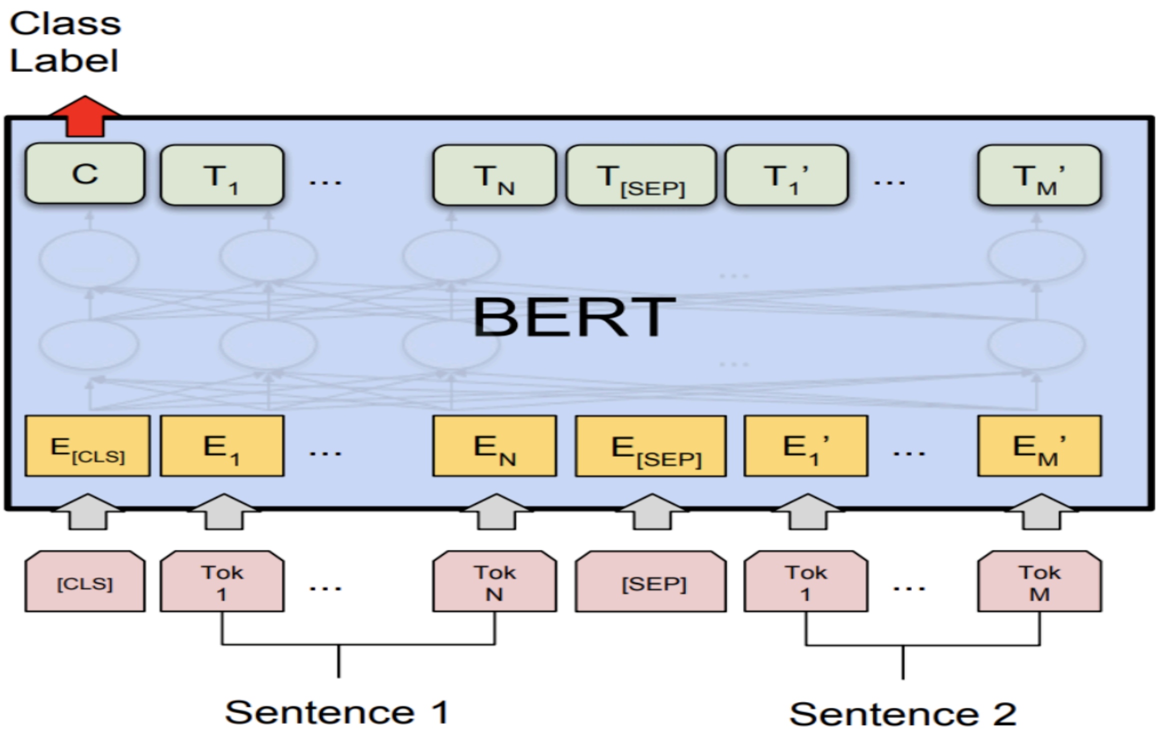
As a consequence, the model ***converges slower*** than *directional models*, a characteristic that is offset by its *increased context-awareness*.

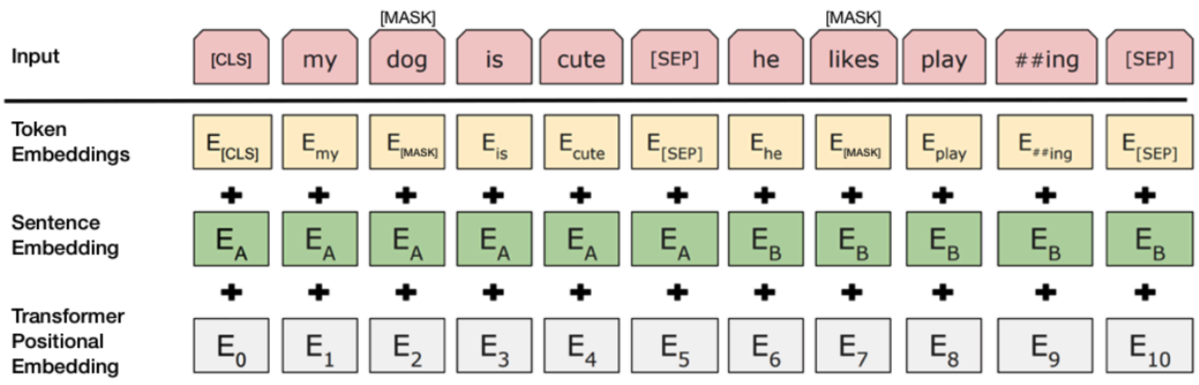
2. Next Sentence Prediction (NSP)

In the BERT training process, ***the model receives pairs of sentences as input and learns to predict if the second sentence in the pair is the subsequent sentence in the original document***. During training, 50% of the inputs are a pair in which the second sentence is the subsequent sentence in the original document, while in the other 50% a random sentence from the corpus is chosen as the second sentence. The assumption is that the random sentence will be disconnected from the first sentence.

To help the model distinguish between the two sentences in training, the input is processed in the following way before entering the model:

1. A [CLS] token is inserted at the beginning of the first sentence and a [SEP] token is inserted at the end of each sentence.
2. A sentence embedding indicating Sentence A or Sentence B is added to each token. Sentence embeddings are similar in concept to token embeddings with a vocabulary of 2.
3. A positional embedding is added to each token to indicate its position in the sequence.





To predict if the second sentence is indeed connected to the first, the following steps are performed:

1. The entire input sequence goes through the Transformer model.

2. The output of the [CLS] token is transformed into a 2×1 shaped vector, using a simple classification layer (learned matrices of weights and biases).
3. Calculating the probability of IsNextSequence with softmax.

When training the BERT model, **Masked LM** and **Next Sentence** Prediction are *trained together*, with the goal of ***minimizing the combined loss function*** of the two strategies.

❖ How to use BERT (Fine-tuning)

Using BERT for a specific task is relatively straightforward:

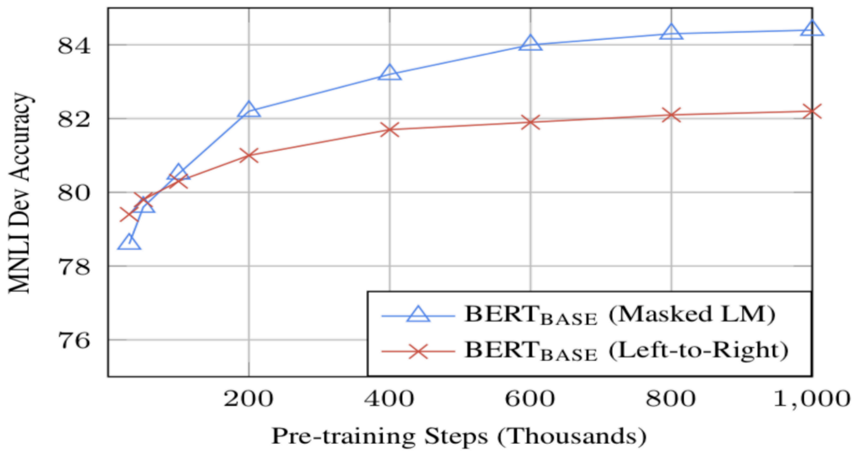
BERT can be used for a wide variety of language tasks, while only adding a small layer to the core model:

1. **Classification** tasks such as sentiment analysis are done *similarly to Next Sentence classification*, by *adding a classification layer on top of the* ***Transformer*** *output for the [CLS] token*.
2. In **Question Answering** tasks, the software receives a question regarding a text sequence and is *required to mark the answer in the sequence. Using BERT, a Q&A model can be trained by learning two extra vectors that mark the beginning and the end of the answer*.
3. In **Named Entity Recognition (NER)**, the software receives a text sequence and is required to mark the various types of entities (Person, Organization, Date, etc) that appear in the text. *Using BERT, a NER model can be trained by*

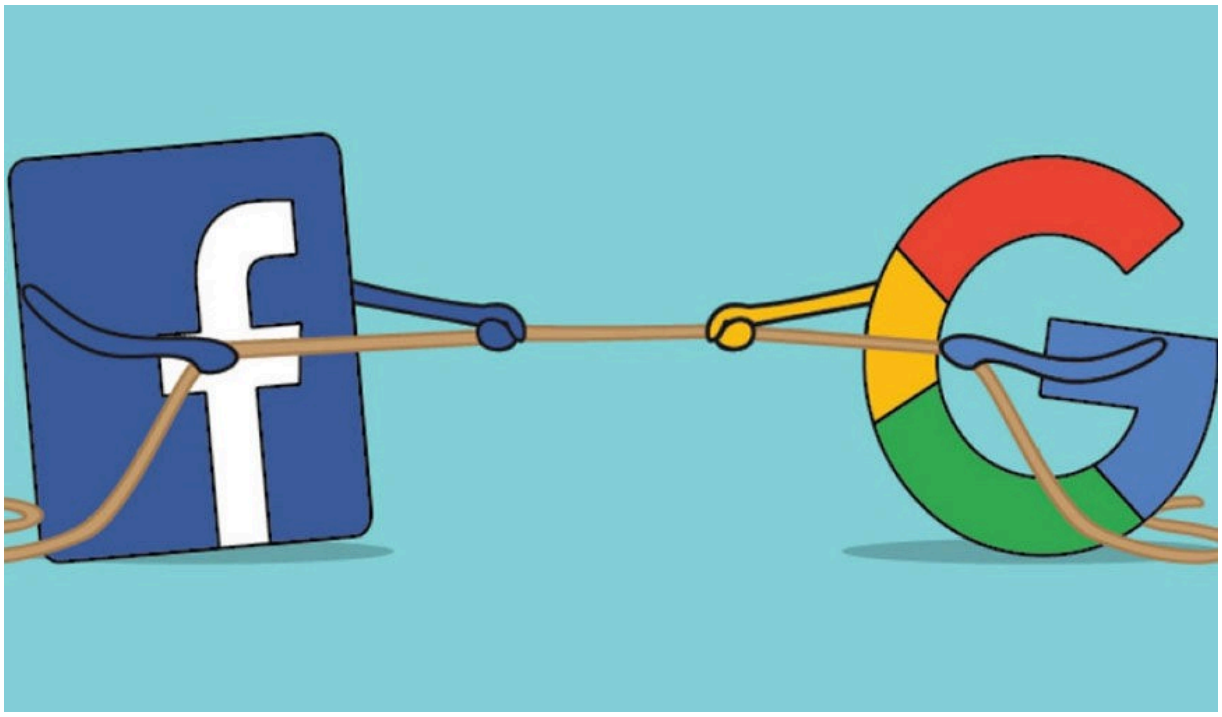
feeding the output vector of each token into a classification layer that predicts the NER label.

❖ **Takeaways:**

1. *Model size matters, even at a huge scale*. **BERT_large**, with **345 million parameters**, is the largest model of its kind. It is demonstrably superior on small-scale tasks to **BERT_base**, which uses the same architecture with "only" **110 million parameters**.
2. *With enough training data, more training steps == higher accuracy*. For instance, on the MNLI task, the BERT_base accuracy improves by 1.0% when trained on 1M steps (128,000 words batch size) compared to 500K steps with the same batch size.
3. ***BERT's bidirectional approach (MLM) converges slower than left-to-right approaches*** (*because only 15% of words are predicted in each batch*) but bidirectional training still outperforms left-to-right training after a small number of pre-training steps.



RoBERTa



- A ***robustly optimized*** method for ***pretraining natural language processing (NLP)*** systems that ***improve*** on

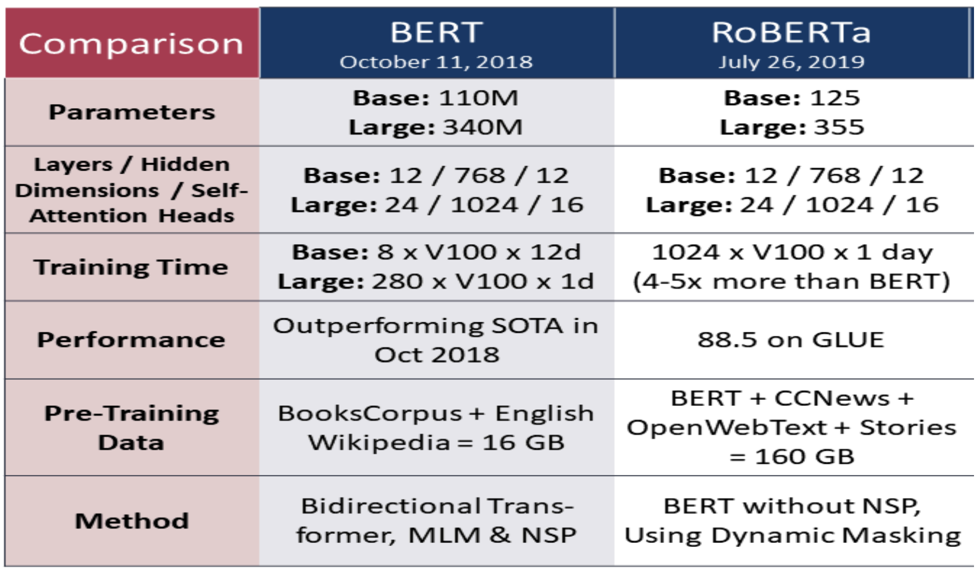
Bidirectional Encoder Representations from Transformers, or **BERT**, the *self-supervised method* released by Google in 2018.

Self-supervised learning (SSL) is **a method of machine learning**. It learns from *unlabeled sample data*. It can be regarded as an *intermediate* form between *supervised and unsupervised learning*.

- ❖ **RoBERTa**, produces state-of-the-art results on the widely used NLP benchmark, *General Language Understanding Evaluation* (GLUE).
- ❖ **GLUE:**
  - ➢ ***The General Language Understanding Evaluation (GLUE)*** benchmark is a collection of resources for training, evaluating, and analyzing natural language understanding systems. GLUE consists of:
  - ➢ A diagnostic dataset designed to evaluate and analyze model performance with respect to a wide range of linguistic phenomena found in natural language.
  - ➢ A public leaderboard for tracking performance on the benchmark and a dashboard for visualizing the performance of models on the diagnostic set.

- ❖ **RoBERTa** is was implemented in ***PyTorch***.
- ❖ It modifies key ***hyperparameters*** in BERT:
  1. removing BERT's next-sentence pretraining(NSP) objective and
  2. training with much larger mini-batches and learning rates.

- ❖ This allows RoBERTa to improve on the masked language modeling objective compared with BERT and leads to better downstream task performance.



Comparison	BERT October 11, 2018	RoBERTa July 26, 2019
Parameters	**Base:** 110M **Large:** 340M	**Base:** 125 **Large:** 355
Layers / Hidden Dimensions / Self-Attention Heads	**Base:** 12 / 768 / 12 **Large:** 24 / 1024 / 16	**Base:** 12 / 768 / 12 **Large:** 24 / 1024 / 16
Training Time	**Base:** 8 x V100 x 12d **Large:** 280 x V100 x 1d	1024 x V100 x 1 day (4-5x more than BERT)
Performance	Outperforming SOTA in Oct 2018	88.5 on GLUE
Pre-Training Data	BooksCorpus + English Wikipedia = 16 GB	BERT + CCNews + OpenWebText + Stories = 160 GB
Method	Bidirectional Trans-former, MLM & NSP	BERT without NSP, Using Dynamic Masking

❖ **RoBERTa** uses **160 GB** of text for pre-training, including
   ➢ 16GB of ***Books Corpus and English Wikipedia*** used in BERT.
   ➢ The additional data included ***CommonCrawl News dataset*** (63 million articles, 76 GB),
   ➢ ***Web text corpus*** (38 GB), and
   ➢ ***Stories from Common Crawl*** (31 GB).

This coupled with a whopping ***1024 V100 Tesla GPUs*** running for a day, led to pre-training of RoBERTa.

❖ **How RoBERTa is different from BERT:**
- More **training data** (16G vs 160G).
- Uses a **dynamic masking pattern** instead of a **static masking pattern.**
- Replacing the **next sentence prediction** objective with **full sentences without NSP.**
- Training on **Longer Sequences.**