BERT

State of the art language model for NLP

BERT

Bidirectional Encoder Representations from Transformers

Introduction

- BERT's key technical innovation is applying the **bidirectional training of Transformer**, a popular attention model, to language modelling.
- This is in contrast to previous efforts which looked at **a text sequence either from left to right** or combined left-to-right and right-to-left training.
- A language model which is bidirectionally trained can have a **deeper sense of language context and flow than single-direction language models**.
- In the BERT's 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.

Background

A different approach, which is also popular in NLP tasks and is **feature-based training**.

In this approach, a pre-trained **neural network produces word embeddings which are then used as features in NLP models**.

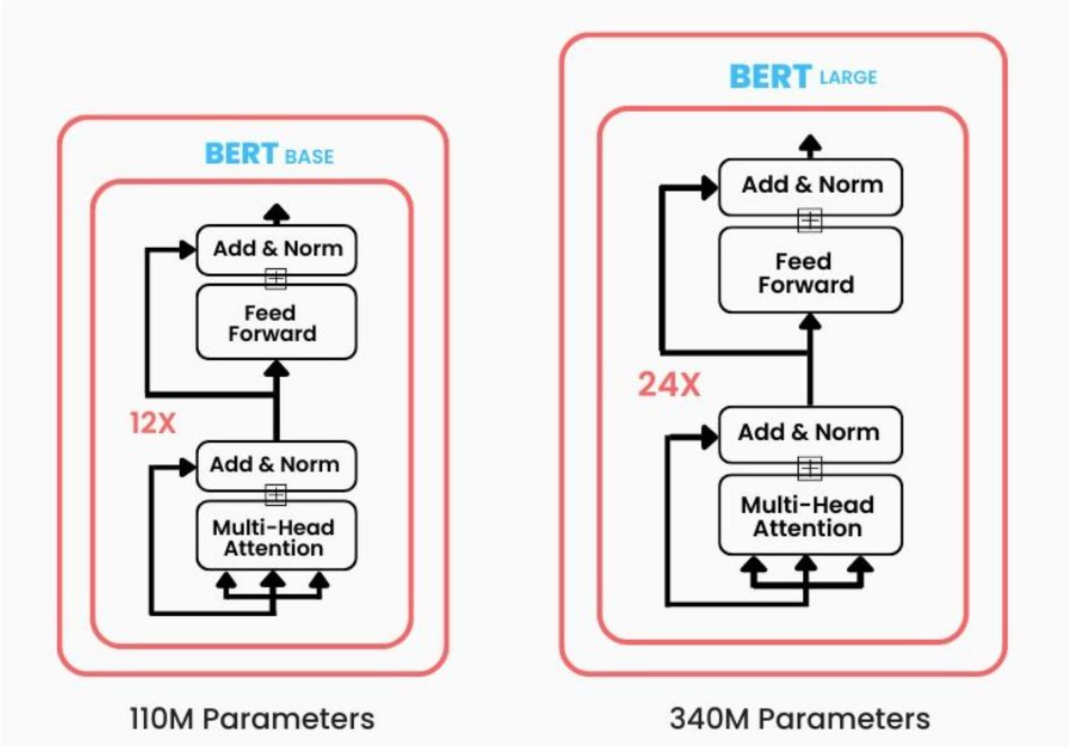
How BERT works

- 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 and
  - a **decoder** that produces a prediction for the task.
- Since BERT's goal is to generate a language model, only the **encoder** mechanism is necessary.
- The detailed workings of Transformer are described in a <u>paper</u> by Google.

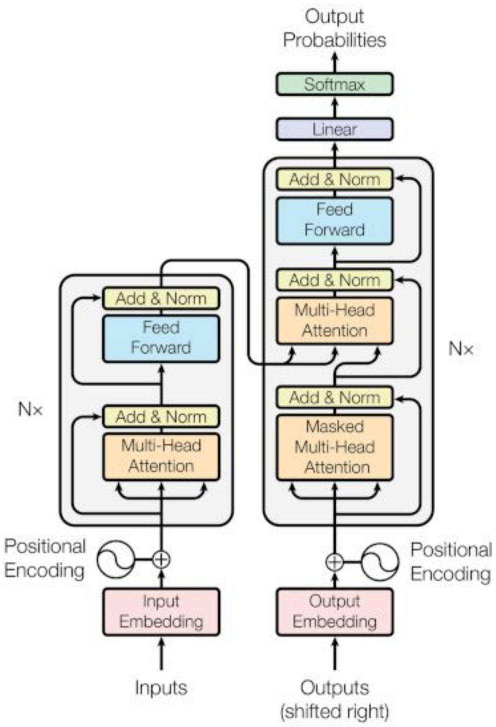
How BERT works

- 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).

BERT Architecture



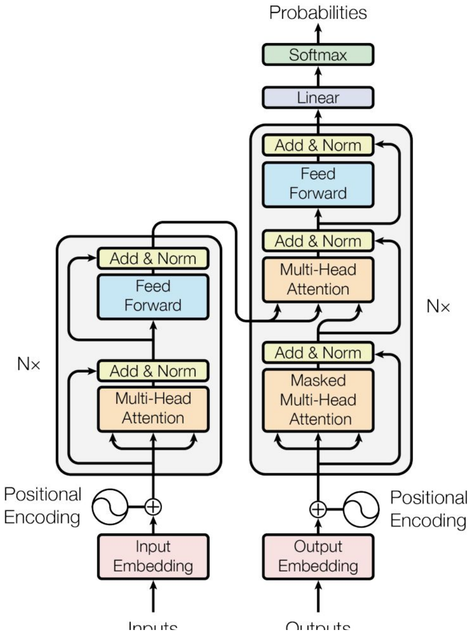
BERT Architecture



BERT Architecture

BERT

Encoder



GPT

Decoder

How BERT works

- The **input** is a sequence of tokens, which are first embedded into vectors and then processed in the neural network.
- The **output** is a sequence of vectors of size H, in which each vector corresponds to an input token with the same index.

How BERT works

When training language models, there is a challenge of defining a prediction goal.

Many models predict the next word in a sequence (e.g. “The child came home from ___”), a directional approach which inherently limits context learning.

To overcome this challenge, BERT uses two training strategies:

1. **Masked LM (MLM)**
2. **Next Sentence Prediction (NSP)**

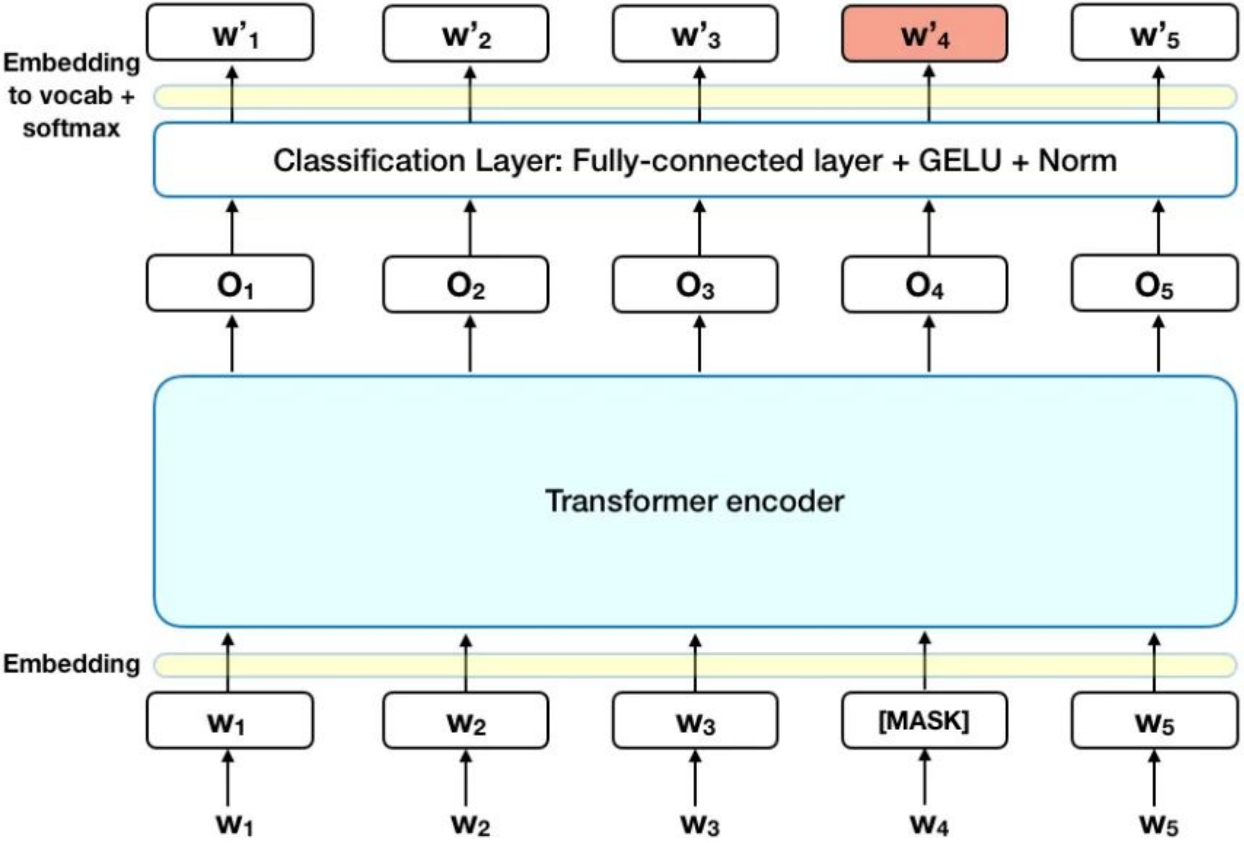
Masked LM (MLM)

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

Masked LM (MLM)

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 which is offset by its increased context awareness.

Note: In practice, the BERT implementation is slightly more elaborate and doesn't replace all of the 15% masked words.

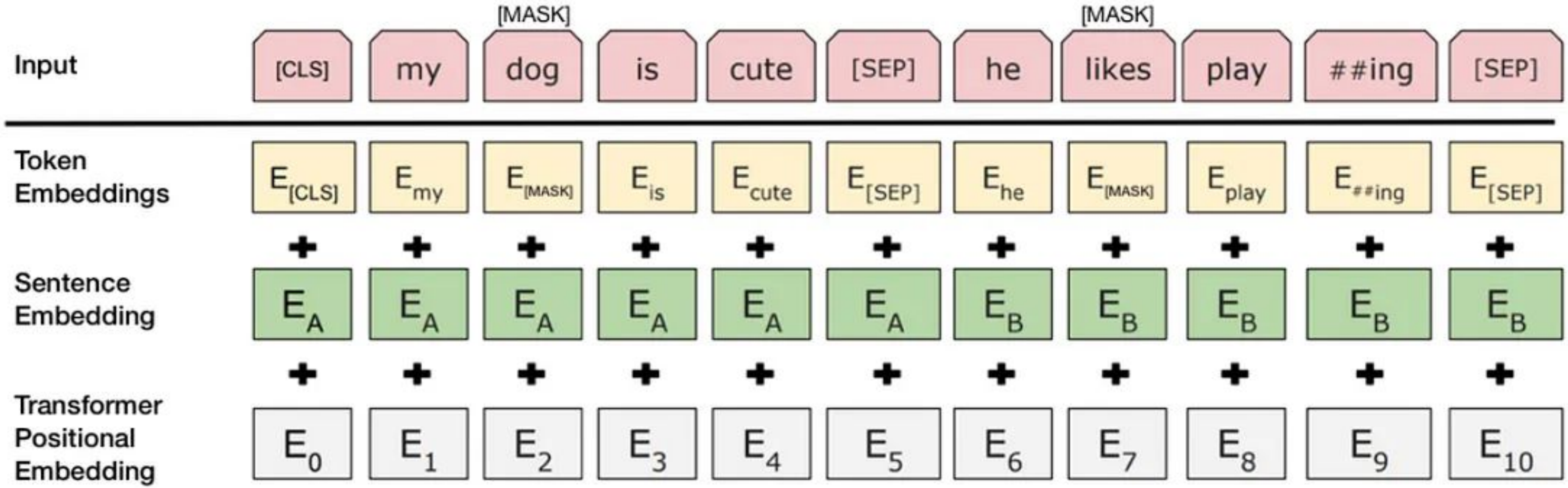
Next Sentence Prediction (NSP)

- 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.

Next Sentence Prediction (NSP)

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. The concept and implementation of positional embedding are presented in the Transformer paper.



[MASK]
[MASK]
Input
[CLS]
my
dog
is
cute
[SEP]
he
likes
play
##ing
[SEP]
Token Embeddings
E[CLS]
Emy
E[MASK]
Eis
Ecute
E[SEP]
Ehe
E[MASK]
Eplay
E##ing
E[SEP]
Sentence Embedding
EA
EB
Transformer Positional Embedding
E0
E1
E2
E3
E4
E5
E6
E7
E8
E9
E10

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)

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

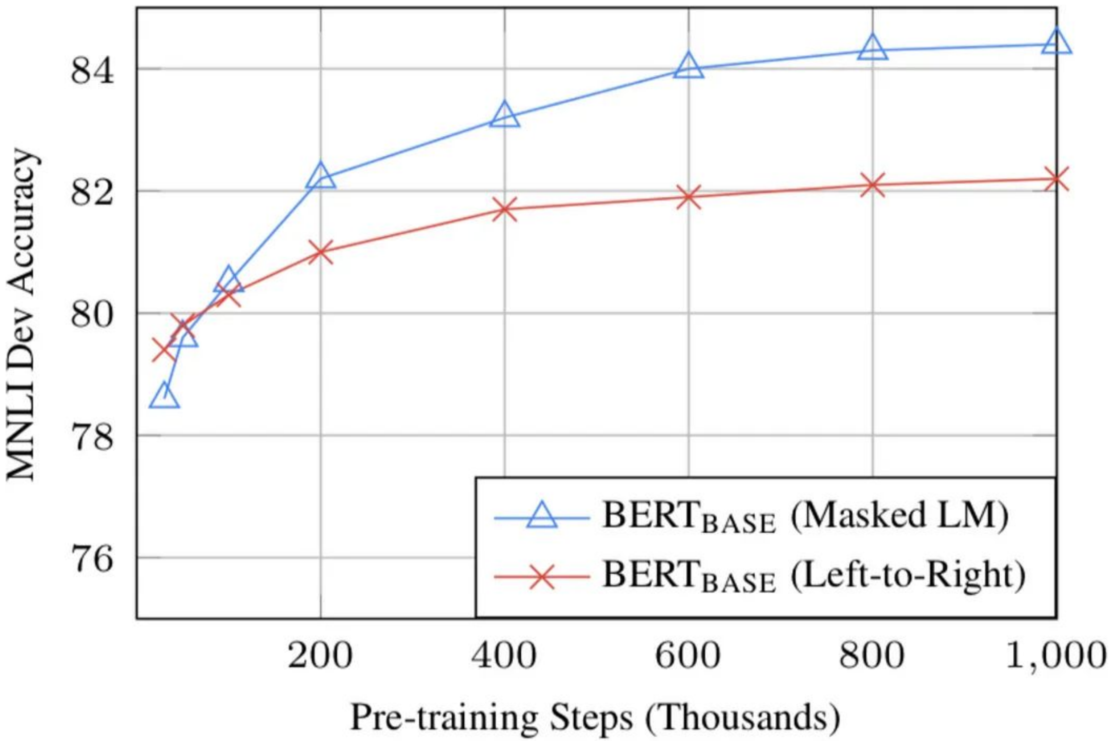
1. **Classification tasks** such as sentiment analysis are done similarly to Next Sentence classification, by adding a classification layer on top of the Transformer output for the [CLS] token.
2. In **Question Answering tasks** (e.g. SQuAD v1.1), the software receives a question regarding a text sequence and is required to mark the answer in the sequence.
   a. 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.
   a. 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.

In the fine-tuning training, most hyper-parameters stay the same as in BERT training, and the authors gives specific guidance on the hyper-parameters that require tuning.

The BERT team has used this technique to achieve state-of-the-art results on a wide variety of challenging natural language tasks.

Takeaways

1. ***Model size matters**, even at 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***.
   a. 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.


84
82
80
78
76
MNLI Dev Accuracy
200
400
600
800
1,000
Pre-training Steps (Thousands)
BERTBASE (Masked LM)
BERTBASE (Left-to-Right)

Compute considerations (training and applying)

	Training Compute + Time	Usage Compute
$BERT_{BASE}$	4 Cloud TPUs, 4 days	1 GPU
$BERT_{LARGE}$	16 Cloud TPUs, 4 days	1 TPU

Conclusion

- BERT is undoubtedly a breakthrough in the use of Machine Learning for Natural Language Processing.
- The fact that it's approachable and allows fast fine-tuning will likely allow a wide range of practical applications in the future.

Appendix A — Word Masking

Training the language model in BERT is done by predicting 15% of the tokens in the input, that were randomly picked. These tokens are pre-processed as follows — 80% are replaced with a “[MASK]” token, 10% with a random word, and 10% use the original word. The intuition that led the authors to pick this approach is as follows :

- If we used [MASK] 100% of the time the model wouldn’t necessarily produce good token representations for non-masked words. The non-masked tokens were still used for context, but the model was optimized for predicting masked words.
- If we used [MASK] 90% of the time and random words 10% of the time, this would teach the model that the observed word is *never* correct.
- If we used [MASK] 90% of the time and kept the same word 10% of the time, then the model could just trivially copy the non-contextual embedding.

Credits

- https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270