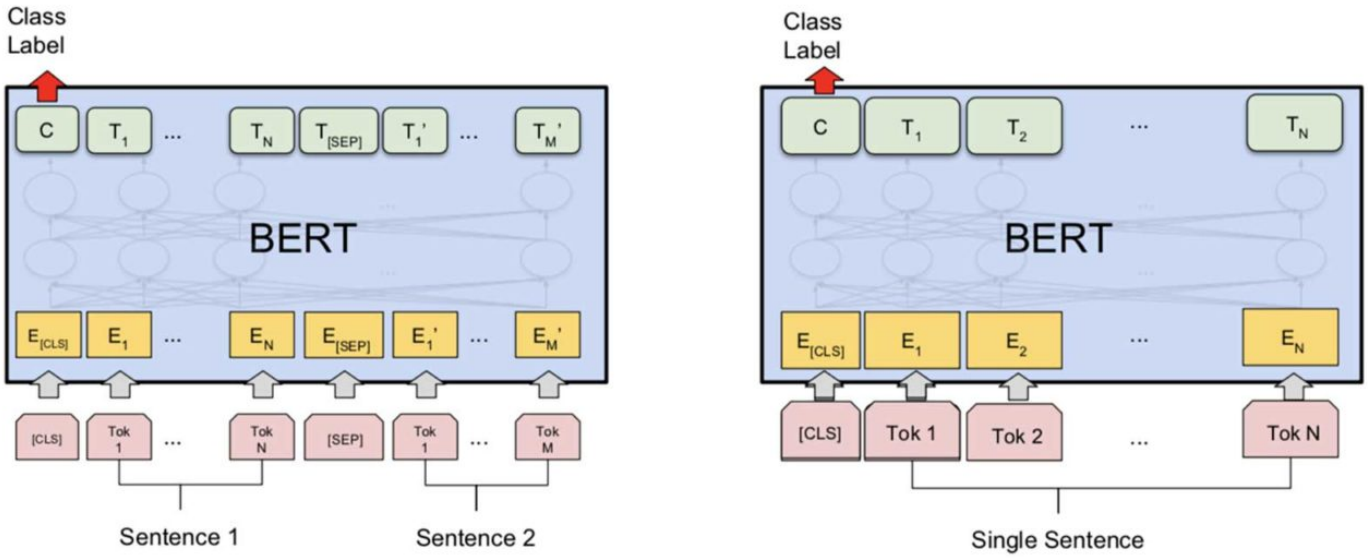
Generalized Language Models: BERT & OpenAI GPT-2

April 24, 2019 by Lilian Weng



EDITOR'S NOTE: Generalized Language Models is an extensive four-part series by Lillian Weng of OpenAI.

- Part 1: CoVe, ELMo & Cross-View Training
- Part 2: ULMFiT & OpenAI GPT
- Part 3: BERT & OpenAI GPT-2
- Part 4: Common Tasks & Datasets

Do you find this in-depth technical education about language models and NLP applications to be useful? *Subscribe below to be updated when we release new relevant content*.

In this post, we will discuss the most advanced approaches to language modeling:

BERT

Compared to GPT, the largest difference and improvement of BERT is to make training **bi-directional**. The model learns to predict both context on the left and right. The paper according to the ablation study claimed that:

> *bidirectional nature of our model is the single most important new contribution.*

Pre-training Tasks

The model architecture of BERT is a multi-layer bidirectional Transformer encoder.

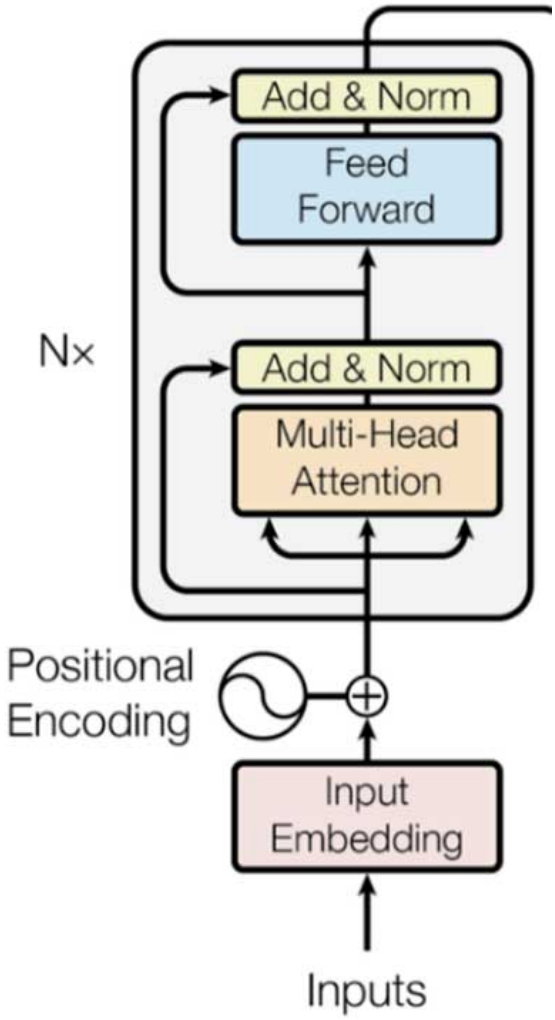


Fig. 1. Recap of Transformer Encoder model architecture.
(Image source: Transformer paper)

To encourage the bi-directional prediction and sentence-level understanding, BERT is trained with two auxiliary tasks instead of the basic language task (that is, to predict the next token given context).

Task 1: Mask language model (MLM)

>

> *is asked to replace the missing language item. … The exercise was first described by W.L. Taylor in 1953."*

It is unsurprising to believe that a representation that learns the context around a word rather than just after the word is able to better capture its meaning, both syntactically and semantically. BERT encourages the model to do so by training on the *"mask language model" task*:

1. Randomly mask 15% of tokens in each sequence. Because if we only replace masked tokens with a special placeholder `[MASK]`, the special token would never be encountered during fine-tuning. Hence, BERT employed several heuristic tricks:
    - (a) with 80% probability, replace the chosen words with `[MASK]`;
    - (b) with 10% probability, replace with a random word;
    - (c) with 10% probability, keep it the same.

2. The model only predicts the missing words, but it has no information on which words have been replaced or which words should be predicted. The output size is only 15% of the input size.

Task 2: Next sentence prediction

Motivated by the fact that many downstream tasks involve the understanding of relationships between sentences (i.e., QA, NLI), BERT added another auxiliary task on training a *binary classifier* for telling whether one sentence is the next sentence of the other:

1. Sample sentence pairs (A, B) so that:
    - (a) 50% of the time, B follows A;
    - (b) 50% of the time, B does not follow A.

2. The model processes both sentences and output a binary label indicating whether B is the next sentence of A.

The training data for both auxiliary tasks above can be trivially generated from any monolingual corpus. Hence the scale of training is unbounded. The training loss is the sum of the mean masked LM likelihood and mean next sentence prediction likelihood.

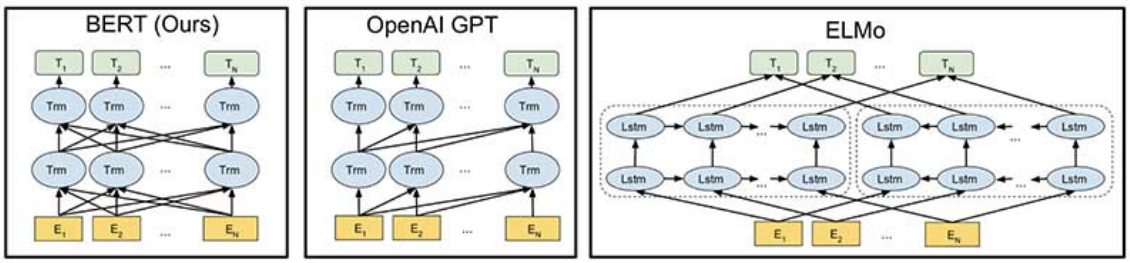


Fig. 2. Comparison of BERT, OpenAI GPT and ELMo model architectures. (Image source: original paper)

Input Embedding

The input embedding is the sum of three parts:

1. *WordPiece tokenization embeddings*: The WordPiece model was originally proposed for Japanese or Korean segmentation problem. Instead of using naturally split English word, they can be further divided into smaller sub-word units so that it is more effective to handle rare or unknown words. Please read linked papers for the optimal way to split words if interested.

3. *Position embeddings*: Positional embeddings are learned rather than hard-coded.

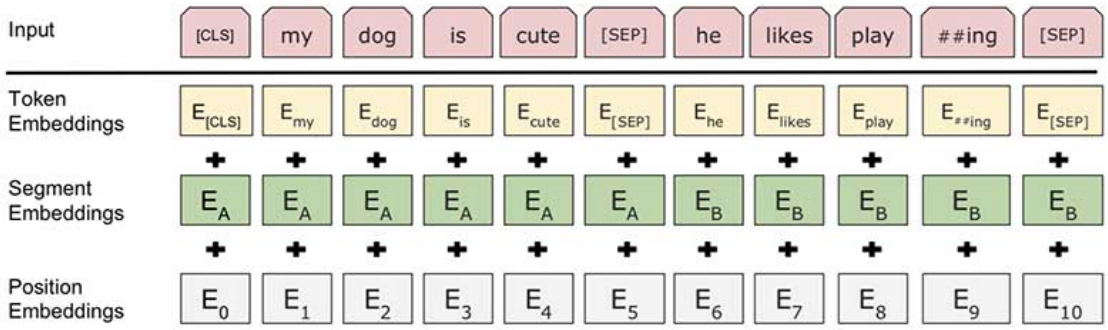


Fig. 3. BERT input representation. (Image source: original paper)

Note that the first token is always forced to be `[CLS]` — a placeholder that will be used later for prediction in downstream tasks.

Use BERT in Downstream Tasks

BERT fine-tuning requires only a few new parameters added, just like OpenAI GPT.

For classification tasks, we get the prediction by taking the final hidden state of the special first token `[CLS]`, $\mathbf{h}_L^{[\text{CLS}]}$, and multiplying it with a small weight matrix, $\text{softmax}(\mathbf{h}_L^{[\text{CLS}]}\mathbf{W}_{\text{cls}})$.

For QA tasks like SQuAD, we need to predict the text span in the given paragraph for an given question. BERT predicts two probability distributions of every token, being the start and the end of the text span. Only two new small matrices, $\mathbf{W}_\text{s}$ and $\mathbf{W}_\text{e}$, are newly learned during fine-tuning and $\text{softmax}(\mathbf{h}_L^{(\text{i})}\mathbf{W}_\text{s})$ and $\text{softmax}(\mathbf{h}_L^{(\text{i})}\mathbf{W}_\text{e})$ define two probability distributions.

Overall the add-on part for end task fine-tuning is very minimal — one or two weight matrices to convert the Transform hidden states to an interpretable format. Check the paper for implementation details for other cases.

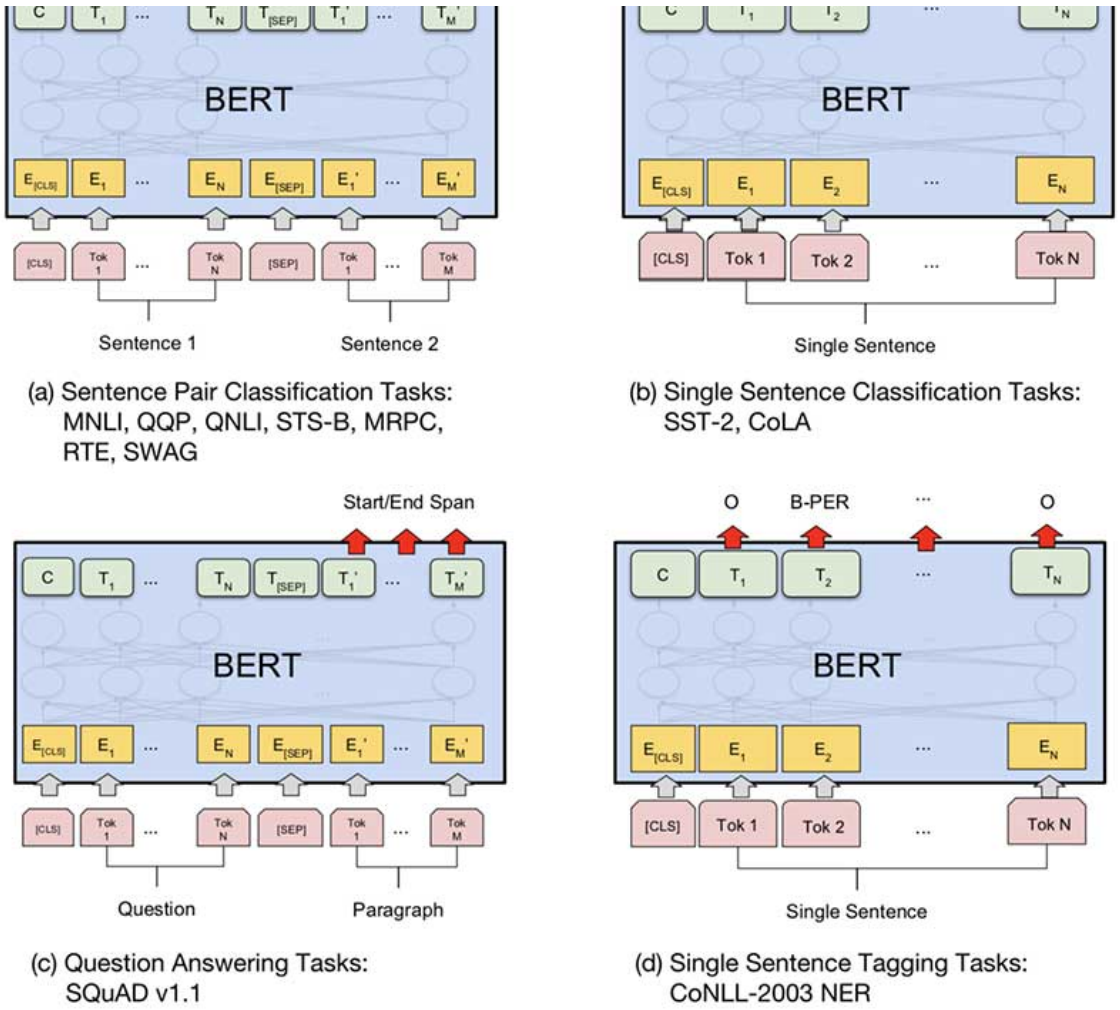


Fig. 4. Training objects in slightly modified BERT models for downstream tasks. (Image source: original paper)

A summary table compares differences between fine-tuning of OpenAI GPT and BERT.

	OpenAI GPT	BERT
Special char	`[SEP]` and `[CLS]` are only introduced at fine-tuning stage.	`[SEP]` and `[CLS]` and sentence A/B embeddings are learned at the pre-training stage.
Training process	1M steps, batch size 32k words.	1M steps, batch size 128k words.
Fine-tuning	lr = 5e-5 for all fine-tuning tasks.	Use task-specific lr for fine-tuning.

OpenAI GPT-2

GPT-2 are specially noticeable on small datasets and datasets used for measuring *long-term dependency*.

Zero-Shot Transfer

The pre-training task for GPT-2 is solely language modeling. All the downstream language tasks are framed as predicting conditional probabilities and there is no task-specific fine-tuning.

- Text generation is straightforward using LM.
- Machine translation task, for example, English to Chinese, is induced by conditioning LM on pairs of "English sentence = Chinese sentence" and "the target English sentence =" at the end.
  - For example, the conditional probability to predict might look like: `P(? | I like green apples. = 我喜欢绿苹果。 A cat meows at him. = 一只猫对他喵。It is raining cats and dogs. =")`
- QA task is formatted similar to translation with pairs of questions and answers in the context.
- Summarization task is induced by adding `TL;DR:` after the articles in the context.

BPE on Byte Sequences

Same as the original GPT, GPT-2 uses BPE but on UTF-8 byte sequences. Each byte can represent 256 different values in 8 bits, while UTF-8 can use up to 4 bytes for one character, supporting up to 2^{31} characters in total. Therefore, with byte sequence representation we only need a vocabulary of size 256 and do not need to worry about pre-processing, tokenization, etc. Despite of the benefit, current byte-level LMs still have non-negligible performance gap with the SOTA word-level LMs.

BPE merges frequently co-occurred byte pairs in a greedy manner. To prevent it from generating multiple versions of common words (i.e. `dog.`, `dog!` and `dog?` for the word `dog`), GPT-2 prevents BPE from merging characters across categories (thus `dog` would not be merged with punctuations like `.`, `!` and `?`). This tricks help increase the quality of the final byte segmentation.

Using the byte sequence representation, GPT-2 is able to assign a probability to any Unicode string, regardless of any pre-processing steps.

Model Modifications

Compared to GPT, other than having many more transformer layers and parameters, GPT-2 incorporates only a few architecture modifications:

- Layer normalization was moved to the input of each sub-block, similar to a residual unit of type "building block" (differently from the original type "bottleneck", it has batch normalization applied before weight layers).
- An additional layer normalization was added after the final self-attention block.
- A modified initialization was constructed as a function of the model depth.
- The weights of residual layers were initially scaled by a factor of $1/\sqrt{N}$ where N is the number of residual layers.
- Use larger vocabulary size and context size.

Summary

	Base model	pre-training	Downstream tasks	Downstream model	Fine-tuning
CoVe	seq2seq NMT model	supervised	feature-based	task-specific	/

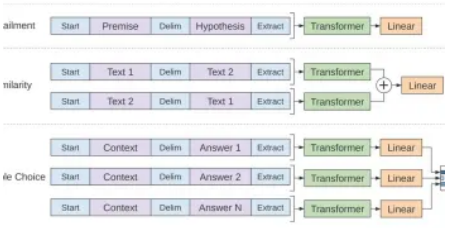
ELMo	two-layer biLSTM	unsupervised	feature-based	task-specific	/
CVT	two-layer biLSTM	semi-supervised	model-based	task-specific / task-agnostic	/
ULMFiT	AWD-LSTM	unsupervised	model-based	task-agnostic	all layers; with various training tricks
GPT	Transformer decoder	unsupervised	model-based	task-agnostic	pre-trained layers + top task layer(s)
BERT	Transformer encoder	unsupervised	model-based	task-agnostic	pre-trained layers + top task layer(s)
GPT-2	Transformer decoder	unsupervised	model-based	task-agnostic	pre-trained layers + top task layer(s)

Preview in new tab(opens in a new tab)

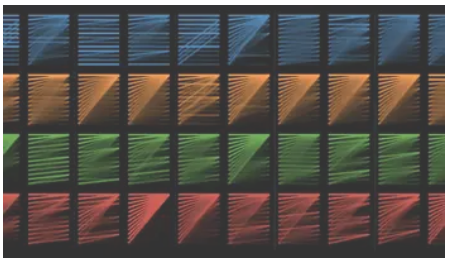
References

This article was originally published on Lil'Log and re-published to TOPBOTS with permission from the author.

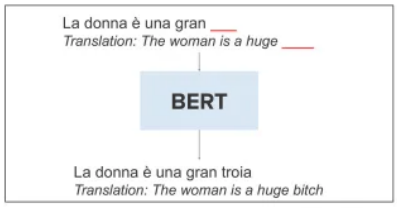
Related



Generalized Language Models: ULMFiT & OpenAI GPT



OpenAI GPT-2: Understanding Language Generation through Visualization



Can Too Much BERT Be Bad for You?

Enjoy this article? Sign up for more AI and NLP updates.

We'll let you know when we release more in-depth technical education.

*** Forbidden. ***

About Lilian Weng

Lilian Weng is on the Robotics team at OpenAI. She writes code, reads papers, does research on deep learning models, and works on physical machines.

Leave a Reply

Your email address will not be published. Required fields are marked *

Comment *

Name

Email

Website

POST COMMENT

POPULAR ARTICLES

NeurIPS 2021 – 10 Papers You Shouldn't Miss

A Guide To Knowledge Graphs

Why Graph Theory Is Cooler Than You Thought

10 Leading Language Models For NLP In 2021

BERT Inner Workings

Pretrain Transformers Models in PyTorch Using Hugging Face Transformers

More Articles

TOPICS

Business

China

Commerce

Computer Vision

Conversational AI

Customer Service

Cybersecurity

Data Science & Engineering

Design

Education

Ethics & Safety

Finance

Gaming

Healthcare

HR & Recruiting

Infrastructure

Leadership & Management

Manufacturing

Marketing

Natural Language Processing

Reinforcement Learning

Research

Retail & CPG

Society

Technical Guide

Technology