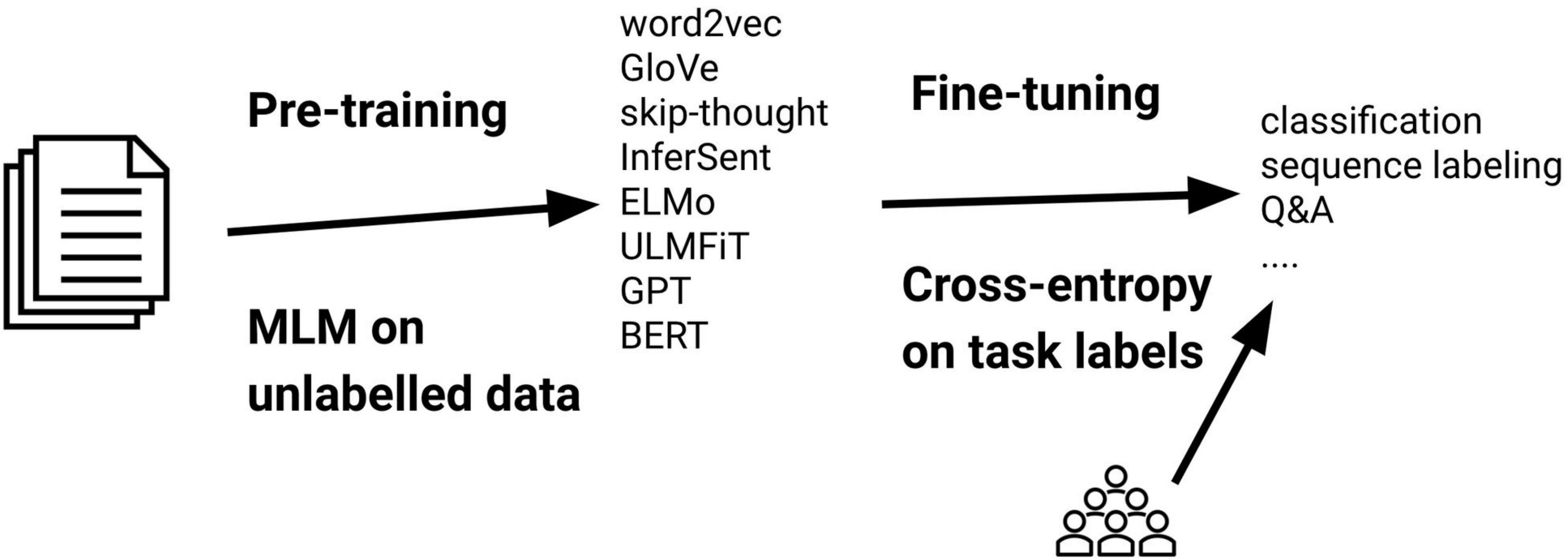
Pre-Training, Fine Tuning & In-Context Learning


Pre-training
MLM on
unlabelled data
word2vec
GloVe
skip-thought
InferSent
ELMo
ULMFiT
GPT
BERT
Fine-tuning
Cross-entropy
on task labels
classification
sequence labeling
Q&A
....

Pre-training

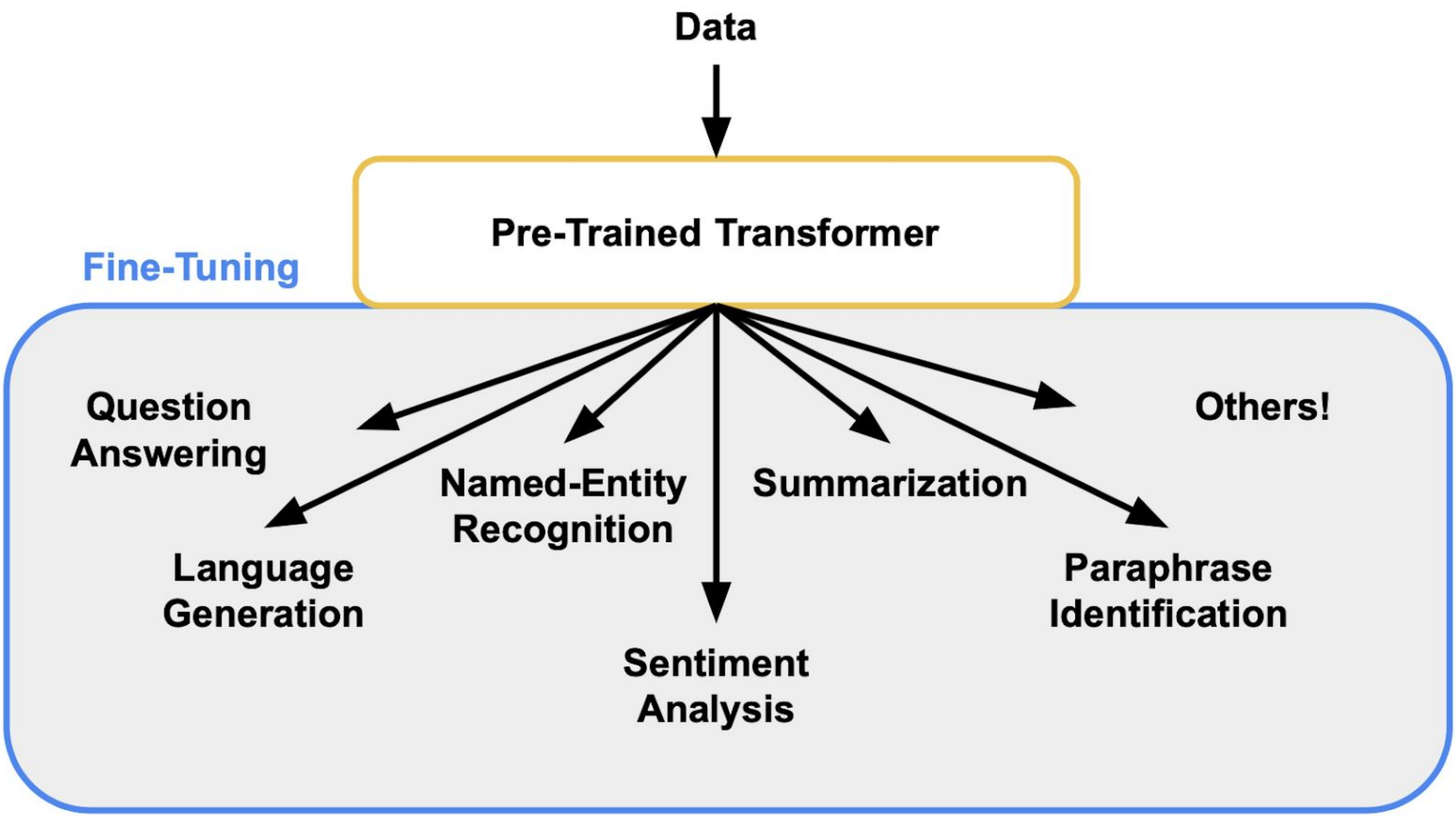
is like a child learning to
read and write his/her mother tongue.

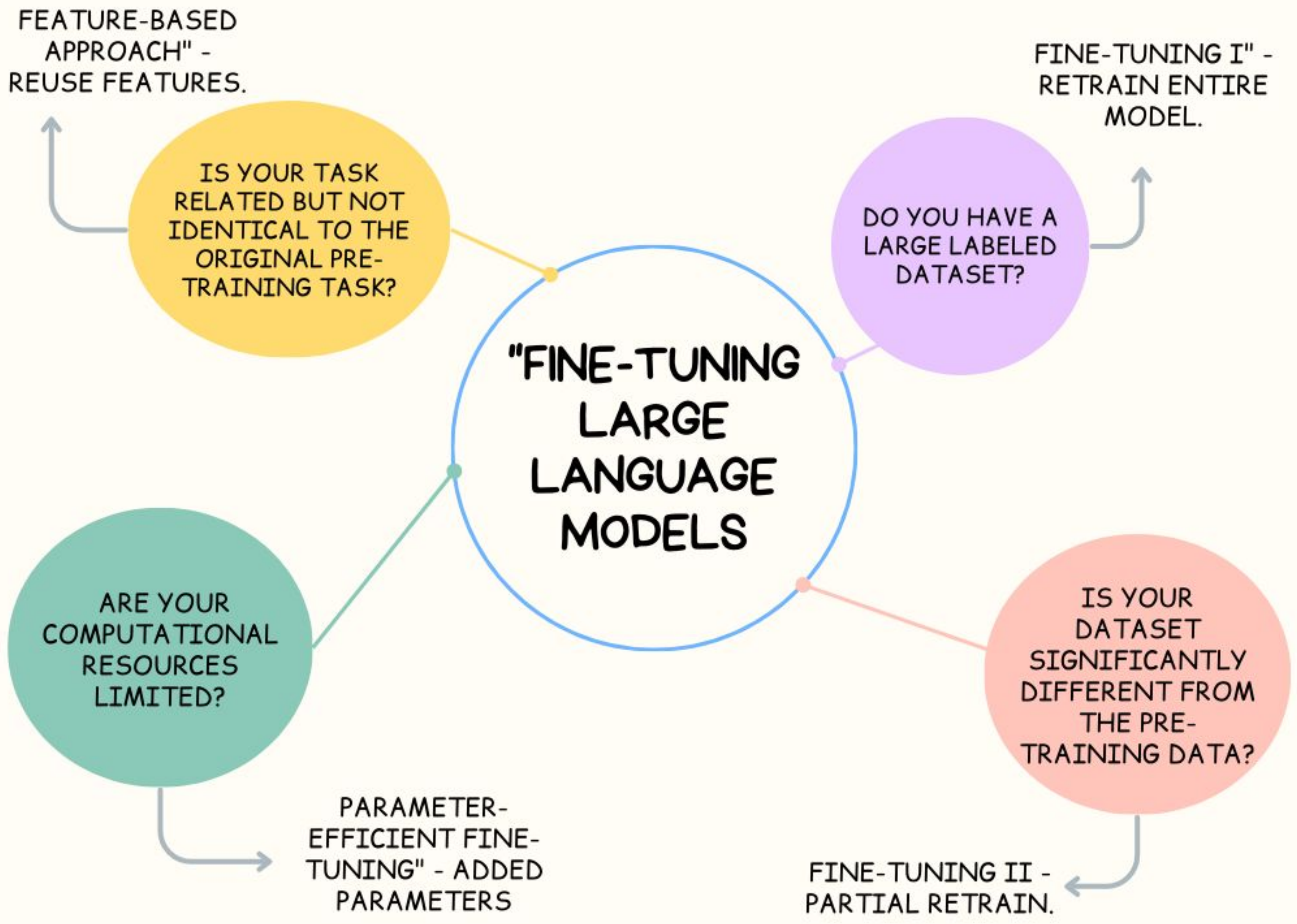
Fine Tuning

is like a student learning to use language to
perform complex tasks in high school and college.

In-Context Learning

is like a working professional trying to
figure out his/her manager's instructions
Zero Shot vs Few Shot


Data
Pre-Trained Transformer
Fine-Tuning
Question Answering
Others!
Named-Entity Recognition
Summarization
Language Generation
Paraphrase Identification
Sentiment Analysis


"FINE-TUNING LARGE LANGUAGE MODELS
IS YOUR TASK RELATED BUT NOT IDENTICAL TO THE ORIGINAL PRE-TRAINING TASK?
FEATURE-BASED APPROACH" - REUSE FEATURES.
DO YOU HAVE A LARGE LABELED DATASET?
FINE-TUNING I" - RETRAIN ENTIRE MODEL.
ARE YOUR COMPUTATIONAL RESOURCES LIMITED?
PARAMETER-EFFICIENT FINE-TUNING" - ADDED PARAMETERS
IS YOUR DATASET SIGNIFICANTLY DIFFERENT FROM THE PRE-TRAINING DATA?
FINE-TUNING II - PARTIAL RETRAIN.

In-Context Learning (few shot learning)

In addition to the task description, the model sees a few examples of the task. No gradient updates are performed.

`Translate English to French:`	*task description*
`sea otter => loutre de mer`	*examples*
`peppermint => menthe poivrée`	
`plush girafe => girafe peluche`	
`cheese => ...`	*prompt*

The LLM Landscape

	BERT	GPT3	Llama 2
Year	2018	2020	2023
Developer	Google	OpenAI	Meta
Parameters	110 M, 340 M	175 B	7 B, 13 B, 70 B
Architecture	Encoder only	Decoder only	Decoder only
Embedding Size	768	12888	3204
Context Length	512	2048	4000
Tokenization	WordPiece	BPE	SentencePiece
Use Case	Classification, NER, Q&A	Text Generation	Text Generation

The GPT Models

	GPT-1	GPT-2	GPT-3
Parameters	117 Million	1.5 Billion	175 Billion
Decoder Layers	12	48	96
Context Token Size	512	1024	2048
Hidden Layer	768	1600	12288
Batch Size	64	512	3.2M

LLM Benchmarks

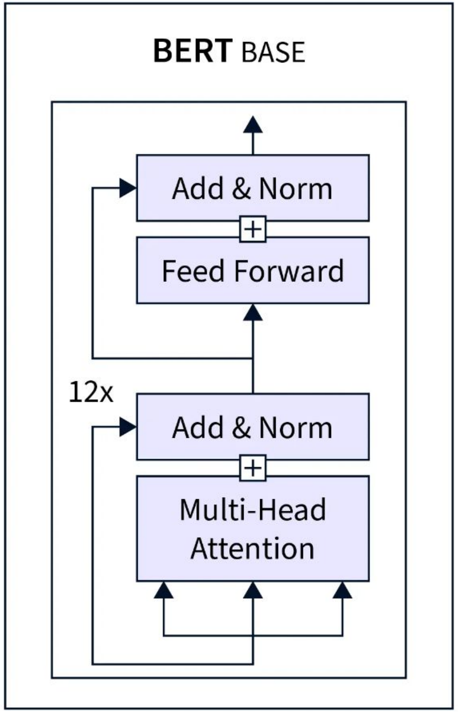
Benchmark	What does it measure?
GLUE	Natural Language Understanding
SQuAD	Reading Comprehension
HellaSwag	Common Sense Inference
ROGUE	Text Summarization
RACE	Reading Comprehension
BLEU	Machine Translation
Perplexity	Probability Distribution
METEOR	Machine Translation

BERT Pre-Training

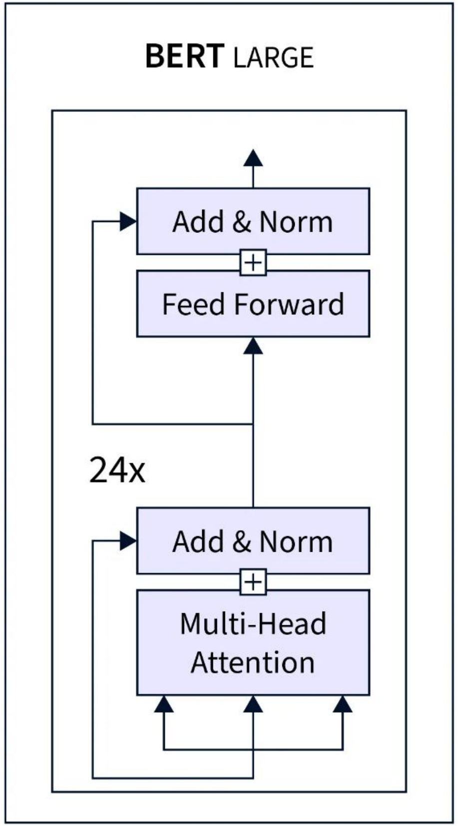
BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

Jacob Devlin **Ming-Wei Chang** **Kenton Lee** **Kristina Toutanova**
Google AI Language
{jacobdevlin,mingweichang,kentonl,kristout}@google.com

BERT Size & Architecture

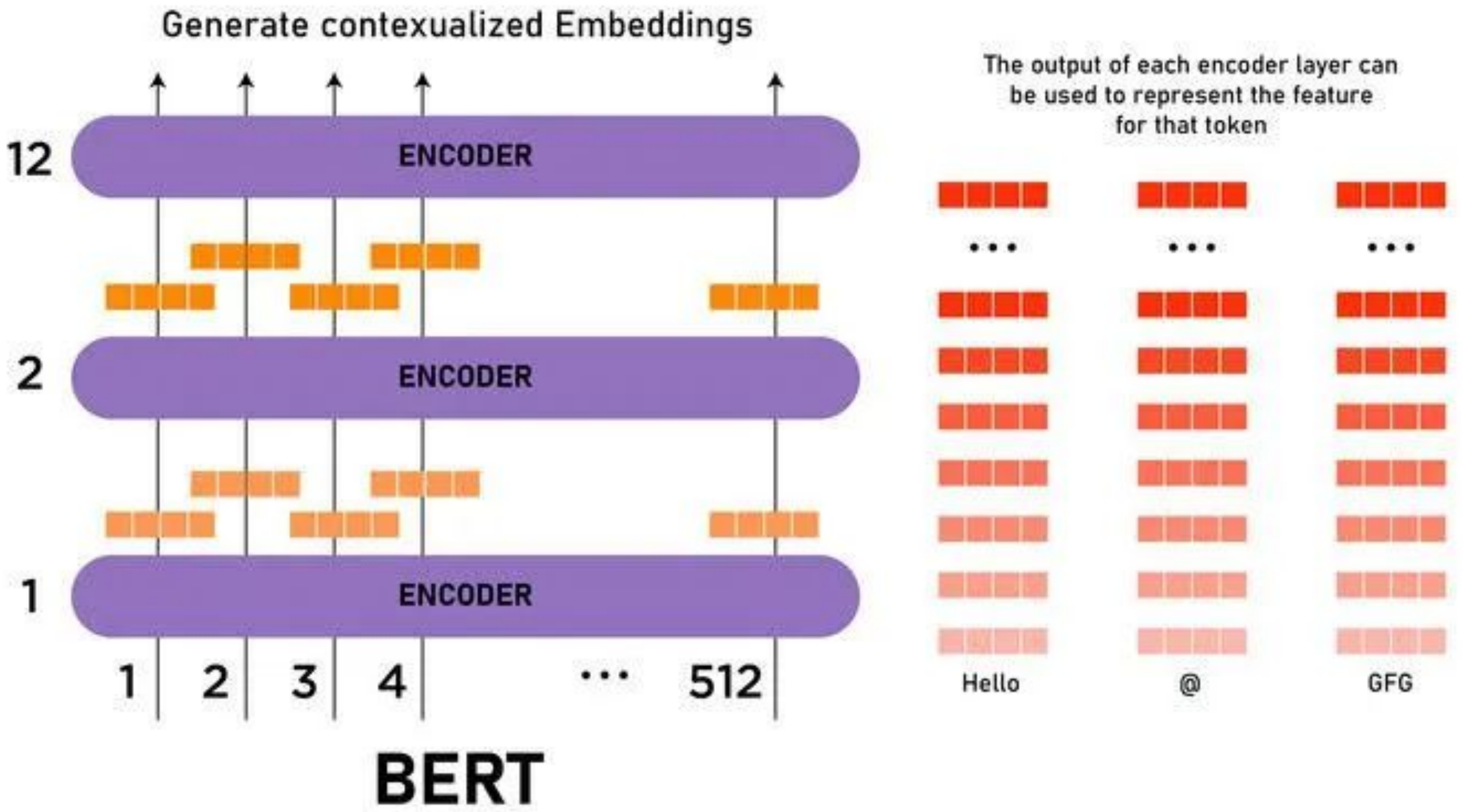


110M Parameters

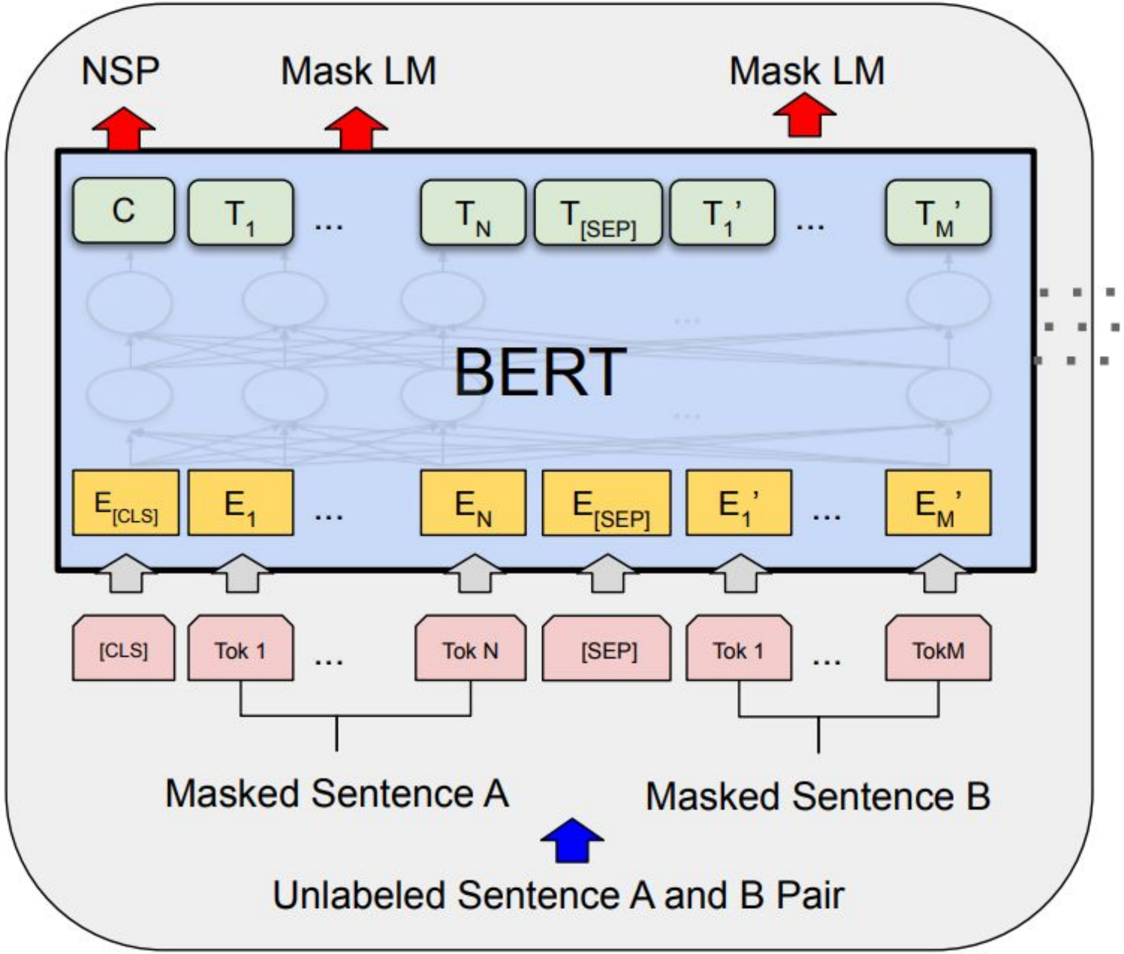


340M Parameters

ONLY ENCODER



Generate contexualized Embeddings
12
ENCODER
2
ENCODER
1
ENCODER
1
2
3
4
...
512
BERT
The output of each encoder layer can be used to represent the feature for that token
Hello
@
GFG



Pre-training

In this work, we denote the number of layers (i.e., Transformer blocks) as L, the hidden size as H, and the number of self-attention heads as A.[3] We primarily report results on two model sizes: $\mathbf{BERT_{BASE}}$ (L=12, H=768, A=12, Total Parameters=110M) and $\mathbf{BERT_{LARGE}}$ (L=24, H=1024, A=16, Total Parameters=340M).

$\text{BERT}_{\text{BASE}}$ was chosen to have the same model size as OpenAI GPT for comparison purposes. Critically, however, the 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.[4]

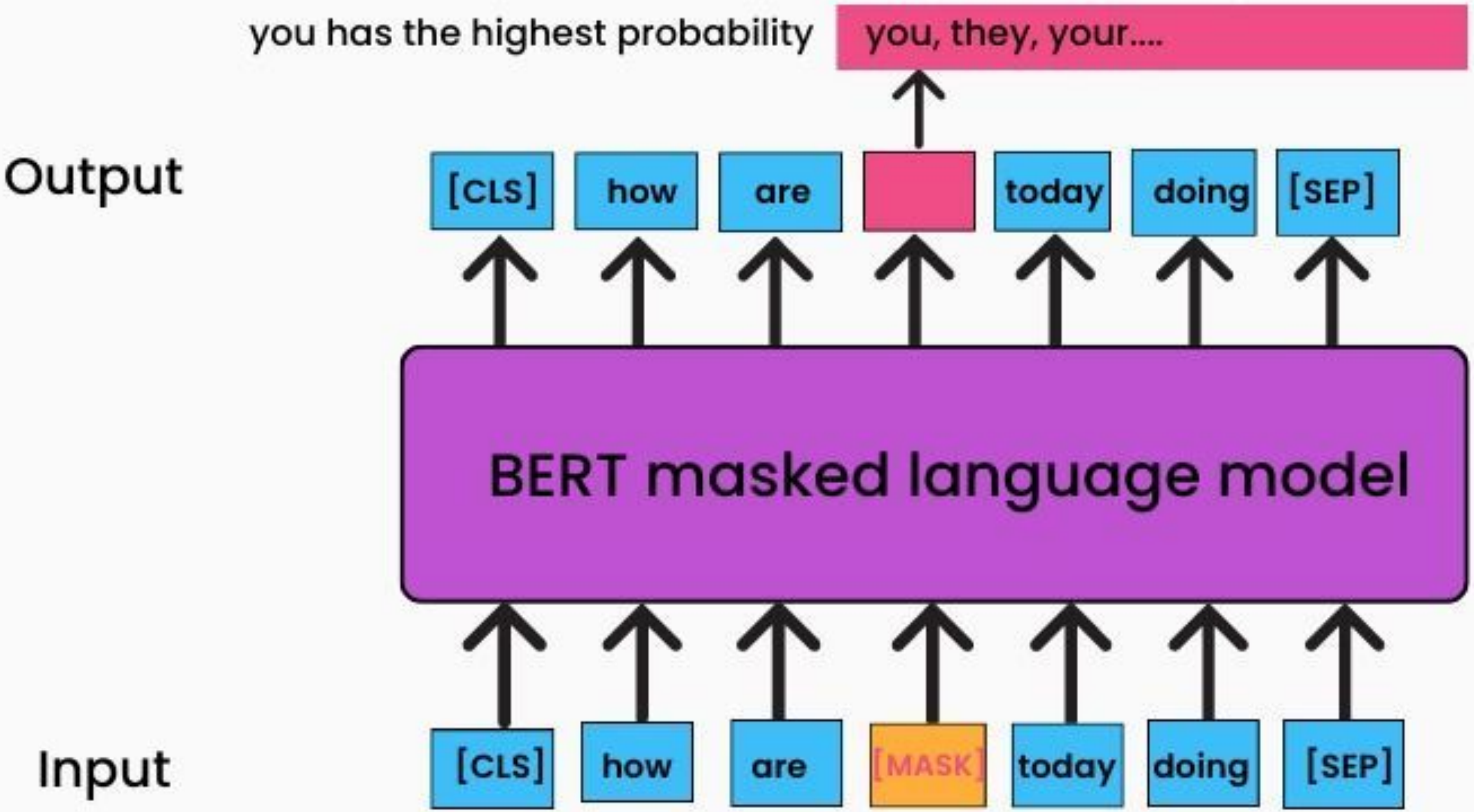
tokens. We refer to this procedure as a "masked LM" (MLM), although it is often referred to as a *Cloze* task in the literature (Taylor, 1953). In this case, the final hidden vectors corresponding to the mask tokens are fed into an output softmax over the vocabulary, as in a standard LM. In all of our experiments, we mask 15% of all WordPiece tokens in each sequence at random. In contrast to denoising auto-encoders (Vincent et al., 2008), we only predict the masked words rather than reconstructing the entire input.

Task #1: Masked LM Intuitively, it is reasonable to believe that a deep bidirectional model is strictly more powerful than either a left-to-right model or the shallow concatenation of a left-to-right and a right-to-left model. 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.

- 80% of the time: Replace the word with the `[MASK]` token, e.g., `my dog is hairy` → `my dog is [MASK]`

- 10% of the time: Replace the word with a random word, e.g., `my dog is hairy` → `my dog is apple`

- 10% of the time: Keep the word unchanged, e.g., `my dog is hairy` → `my dog is hairy`. The purpose of this is to bias the representation towards the actual observed word.

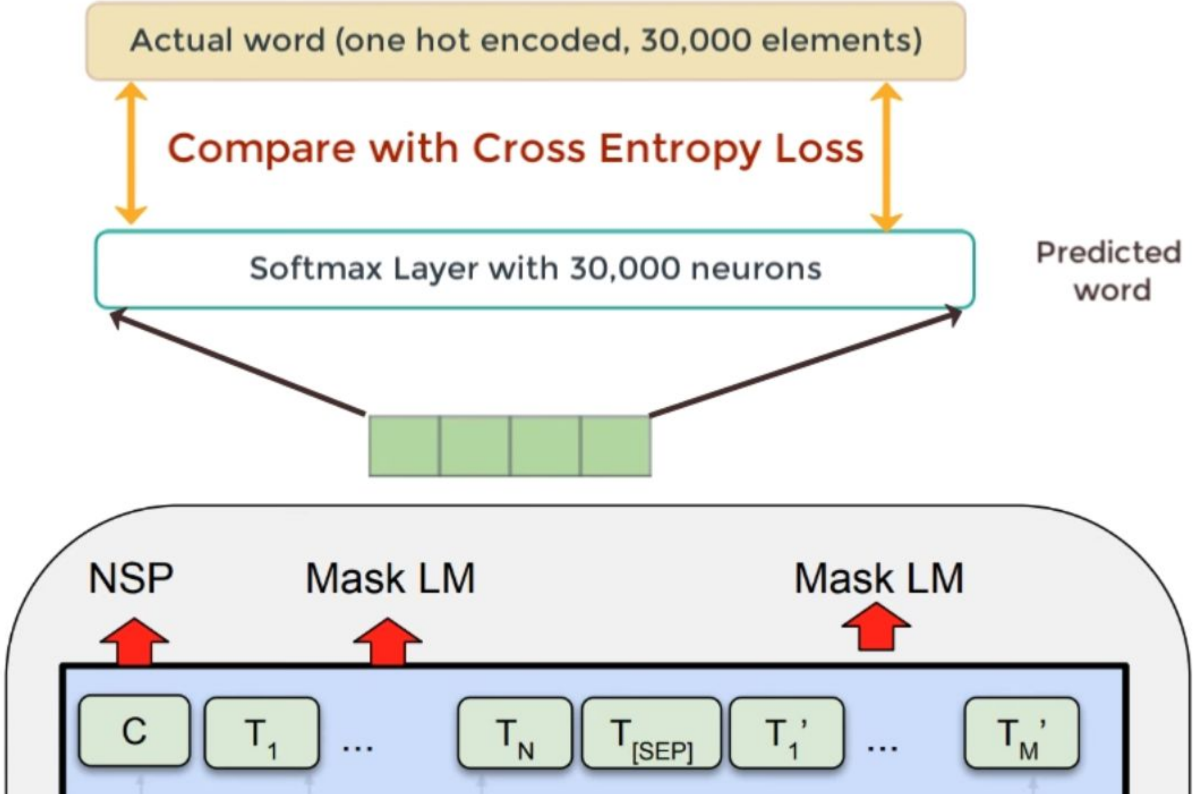

you has the highest probability
you, they, your....
Output
[CLS]
how
are
today
doing
[SEP]
BERT masked language model
Input
[CLS]
how
are
[MASK]
today
doing
[SEP]

Bidirectional Encoder Representation from Transformers

Pretraining (Pass 3)

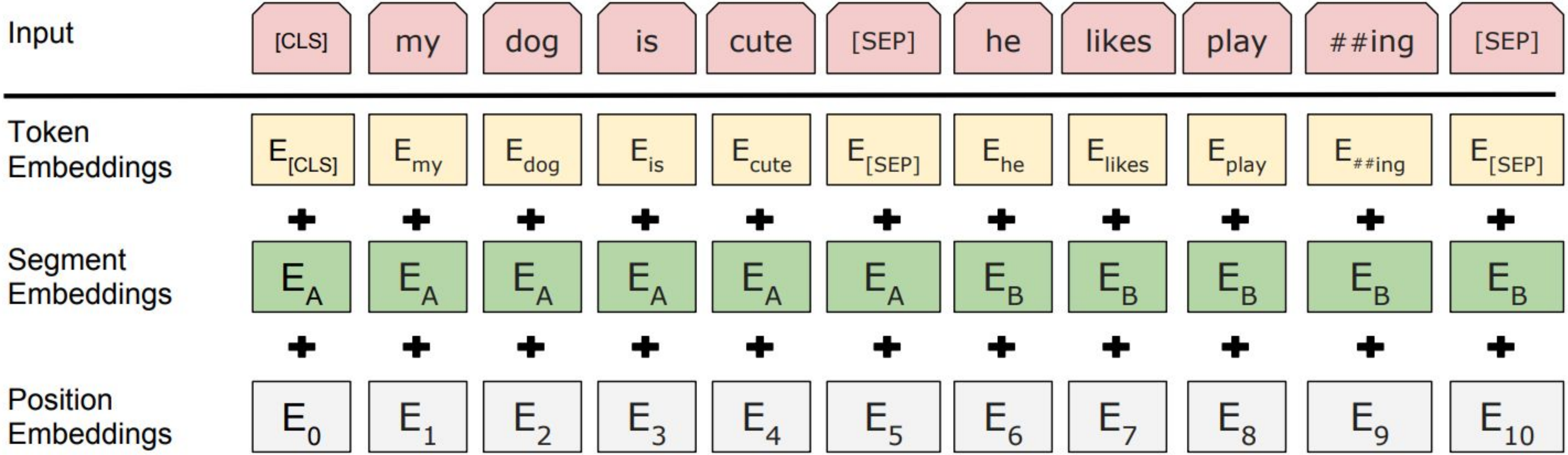
Word vectors T_i have the same size.

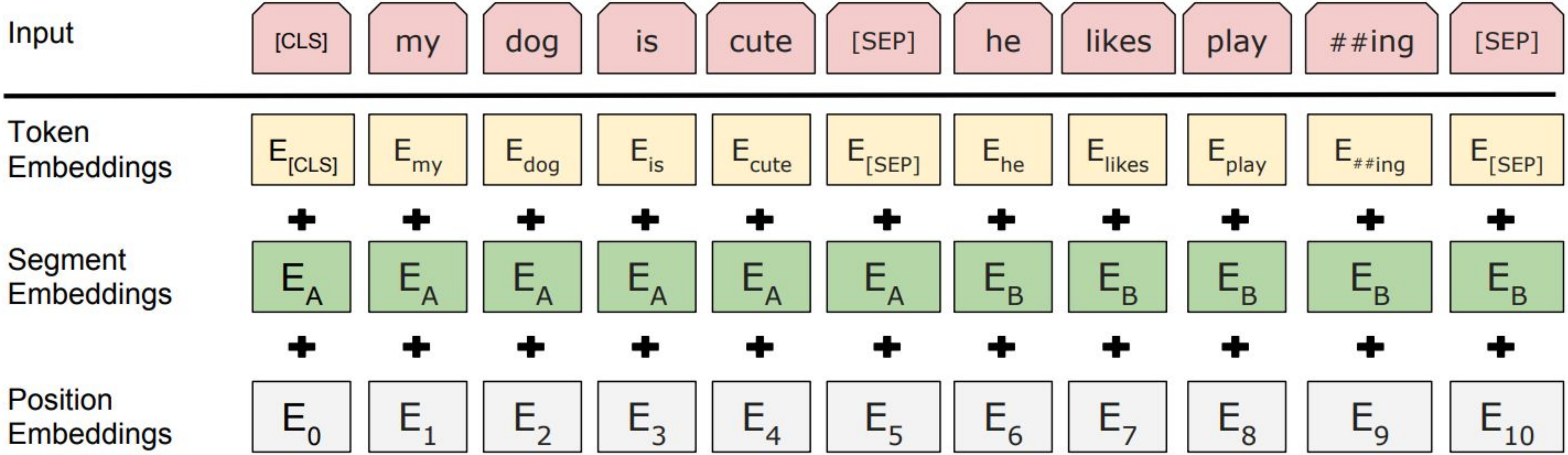
Word vectors T_i are generated simultaneously



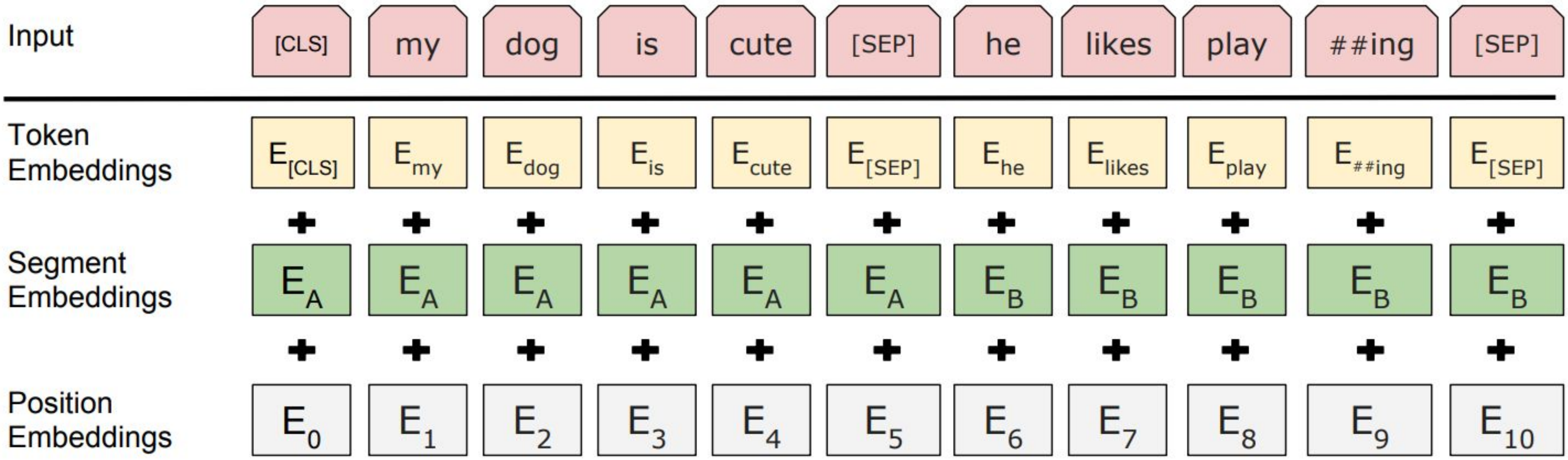
Task #2: Next Sentence Prediction (NSP) Many important downstream tasks such as Question Answering (QA) and Natural Language Inference (NLI) are based on understanding the *relationship* between two sentences, which is not directly captured by language modeling. In order to train a model that understands sentence relationships, we pre-train for a binarized *next sentence prediction* task that can be trivially generated from any monolingual corpus. Specifically, ated from any monolingual corpus. Specifically, when choosing the sentences `A` and `B` for each pre-training example, 50% of the time `B` is the actual next sentence that follows `A` (labeled as `IsNext`), and 50% of the time it is a random sentence from the corpus (labeled as `NotNext`). As we show

Pre-training data The pre-training procedure largely follows the existing literature on language model pre-training. For the pre-training corpus we use the BooksCorpus (800M words) (Zhu et al., 2015) and 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 (Chelba et al., 2013) in order to extract long contiguous sequences.


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}



- Sinusoidal functions
- Learnt from Data
- Rotary Positional Embeddings [RoPE]



- Sinusoidal functions
- Learnt from Data — BERT
- Rotary Positional Embeddings [RoPE] — LLaMA

Training of $\text{BERT}_{\text{BASE}}$ was performed on 4 Cloud TPUs in Pod configuration (16 TPU chips total).[13] Training of $\text{BERT}_{\text{LARGE}}$ was performed on 16 Cloud TPUs (64 TPU chips total). Each pre-training took 4 days to complete.

Longer sequences are disproportionately expensive because attention is quadratic to the sequence length. To speed up pretraing in our experiments, we pre-train the model with sequence length of 128 for 90% of the steps. Then, we train the rest 10% of the steps of sequence of 512 to learn the positional embeddings.

Training of $\text{BERT}_{\text{BASE}}$ was performed on 4 Cloud TPUs in Pod configuration (16 TPU chips total).[13] Training of $\text{BERT}_{\text{LARGE}}$ was performed on 16 Cloud TPUs (64 TPU chips total). Each pre-training took 4 days to complete.

Longer sequences are disproportionately expensive because attention is quadratic to the sequence length. To speed up pretraing in our experiments, we pre-train the model with sequence length of 128 for 90% of the steps. Then, we train the rest 10% of the steps of sequence of 512 to learn the positional embeddings.

LLaMA Model

Architecture [edit]

Like GPT-3, the Llama series of models are decoder-only Transformers, but there are some minor differences:

- SwiGLU[37] activation function instead of GeLU;
- rotary positional embeddings (RoPE)[38] instead of absolute positional embedding;
- RMSNorm[39] instead of layer normalization;[40]

Name	Release date	Parameters	Training cost (petaFLOP-day)	Context length	Corpus size	Commercial viability?
LLaMA	February 24, 2023	• 6.7B • 13B • 32.5B • 65.2B	6,300[32]	2048	1–1.4T	No
Llama 2	July 18, 2023	• 6.7B • 13B • 69B	21,000[33]	4096	2T	Yes
Code Llama	August 24, 2023	• 6.7B • 13B • 33.7B • 69B				
Llama 3	April 18, 2024	• 8B • 70.6B	100,000[34][35]	8192	15T	
Llama 3.1	July 23, 2024	• 8B • 70.6B • 405B	440,000[31][36]	128,000		

Name	Release date	Parameters	Training cost (petaFLOP-day)	Context length	Corpus size	Commercial viability?
LLaMA	February 24, 2023	• 6.7B • 13B • 32.5B • 65.2B	6,300[32]	2048	1–1.4T	No
Llama 2	July 18, 2023	• 6.7B • 13B • 69B	21,000[33]	4096	2T	Yes
Code Llama	August 24, 2023	• 6.7B • 13B • 33.7B • 69B				
Llama 3	April 18, 2024	• 8B • 70.6B	100,000[34][35]	8192	15T	
Llama 3.1	July 23, 2024	• 8B • 70.6B • 405B	440,000[31][36]	128,000		

Dollars??

key hyperparameters of Llama 3.1

	8B	70B	405B
Layers	32	80	126
Model Dimension	4,096	8192	16,384
FFN Dimension	14,336	28,672	53,248
Attention Heads	32	64	128
Key/Value Heads	8	8	8
Peak Learning Rate	3×10^{-4}	1.5×10^{-4}	0.8×10^{-4}
Activation Function	SwiGLU		
Vocabulary Size	128,000		
Positional Embeddings	$\mathrm{RoPE}(\theta = 500,000)$		

Instruct Mode

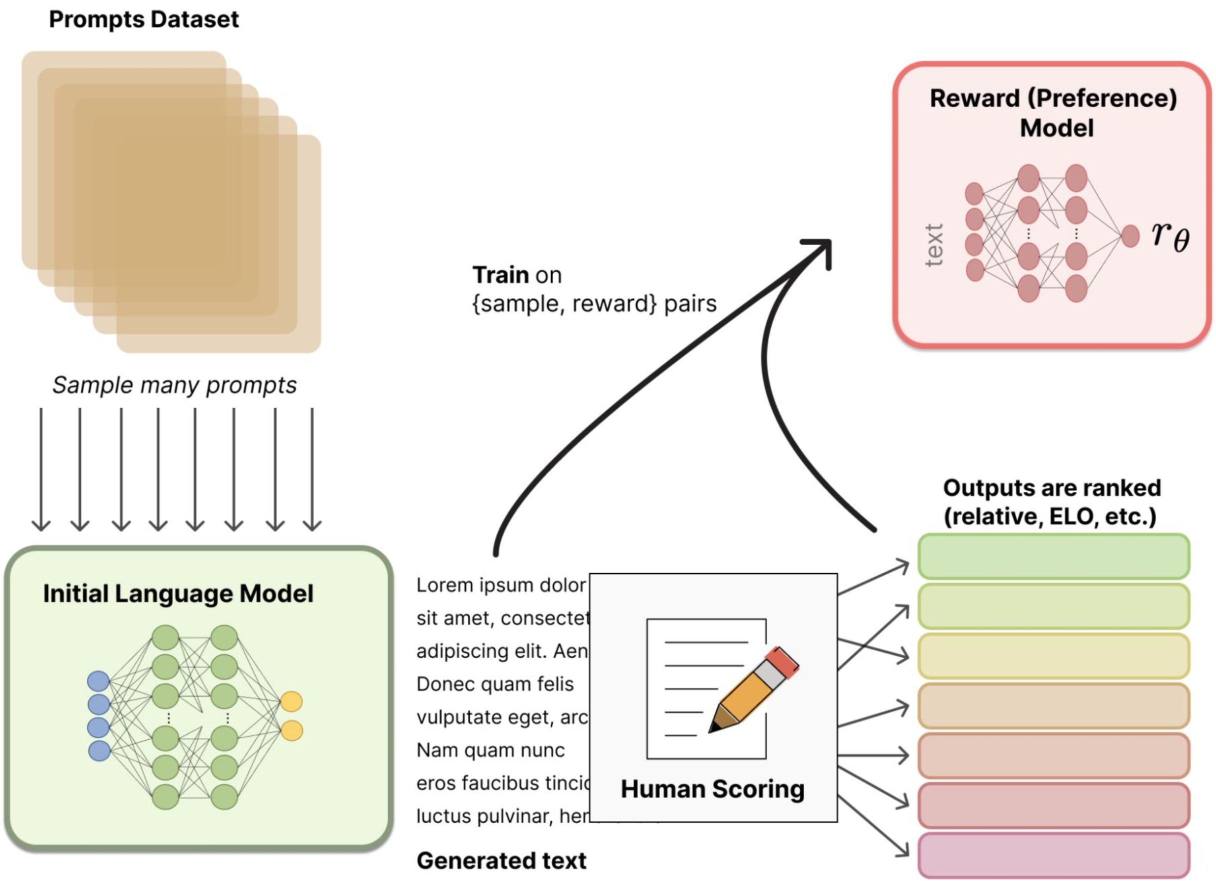
Instruct mode is typically used for direct, task-oriented commands. Users provide explicit instructions, and the model generates responses based on those commands. This mode is highly effective for tasks such as data summarization, translation, and specific question answering. For example, a user might instruct the model to "Summarize the latest research on quantum computing," and the model will generate a concise summary based on the input data.

Chat Mode

Chat mode, on the other hand, is optimized for interactive, conversational exchanges. It is designed to handle multi-turn dialogues, maintain context over several interactions, and provide responses that are coherent and contextually relevant. This mode is particularly useful for customer service bots, virtual assistants, and any application requiring a natural conversational flow.

Reinforcement Learning with Human Feedback

[RLHF]



NON-GRADED TASKS

Write a 400-word blog on the BERT model that a non-CS person can understand

Teach the concept of word embeddings
and sentence similarity to at least 3 first year students

(without getting into details of the transformer model)

Compute the BERT embedding vectors for the SU chatbot data and:

- Find their PCA components (n=2) and see if they form any clusters.
- Do K-Means clustering of the full embedding vectors
- Compare the results from [CLS] and pooler_output
- Instead of the final layer, use embeddings from intermediate layers
- Make random changes in the model parameters and see its effect

Repeat the above with SBERT (try different models)