LVC 2: BERT and its Applications

Natural Language Processing with Large Language Models

Agenda

- Transformers - Recap
- Types of Transformer Models
- BERT
- Training BERT
- Extensions of BERT

Transformer - Recap

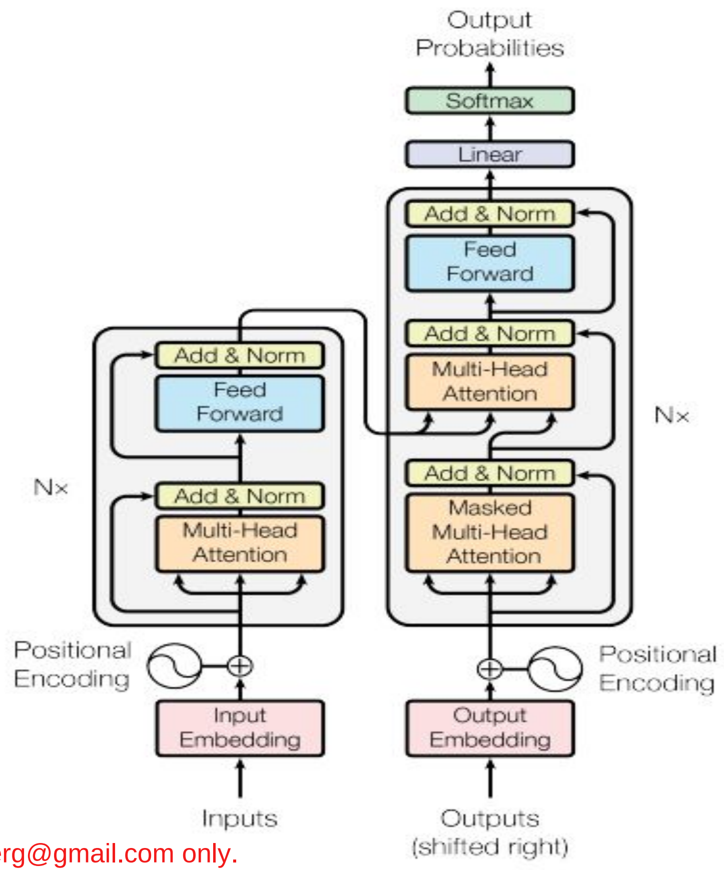
The Transformer Model - Overview

Transformers were introduced in a paper by **Vaswani et al. in 2017**

Transformers are a type of **neural network** architecture

Transformers are based on the idea of **self-attention**

Transformers consist of an **encoder** and a **decoder**

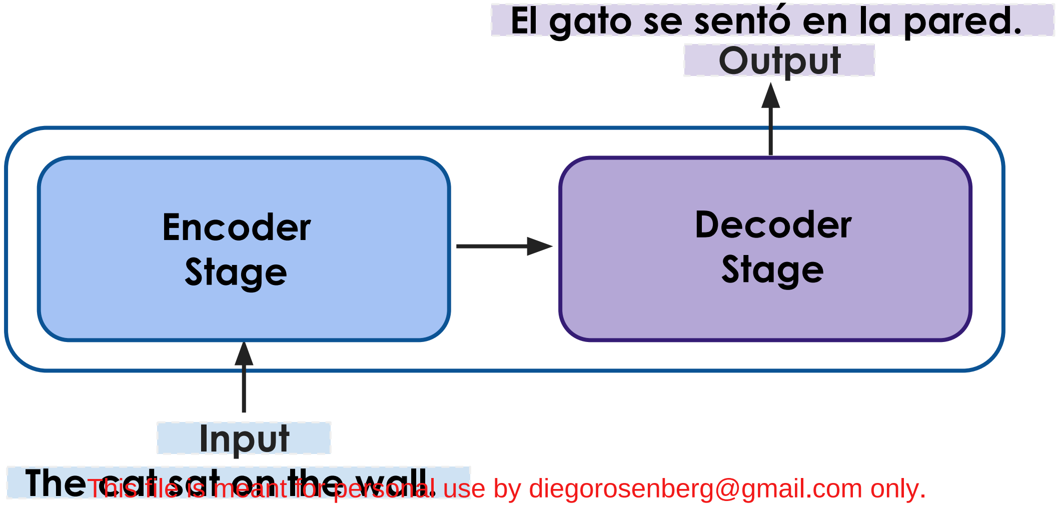


Source: Image from the original research paper Attention Is All You Need

The Transformer Model - High-level Flow

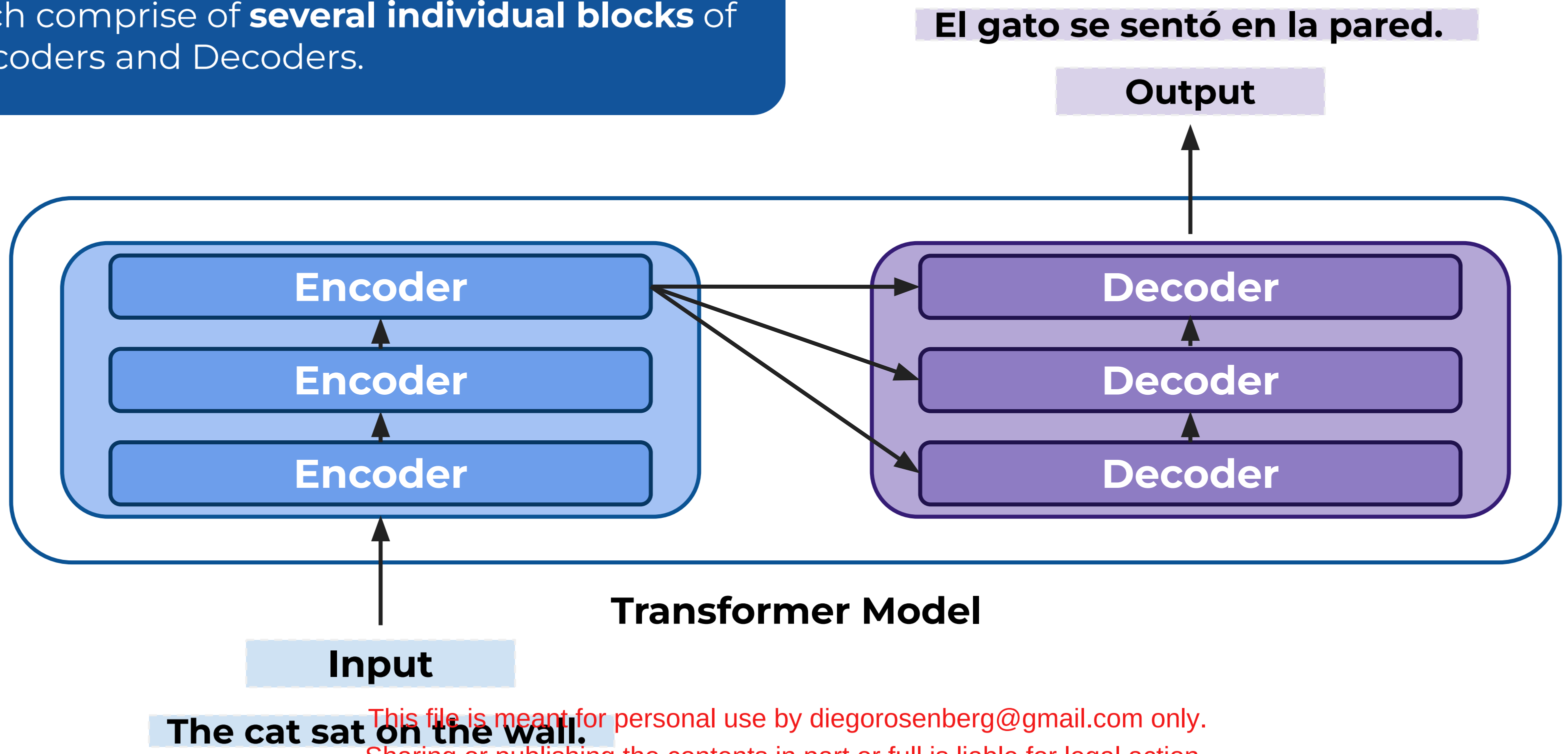
The **encoder** takes in a sequence of tokens (e.g. words or characters) and outputs a **latent representation**

The **decoder** then takes this latent representation as input and outputs a **sequence of tokens**



The Transformer Model - High-level Flow

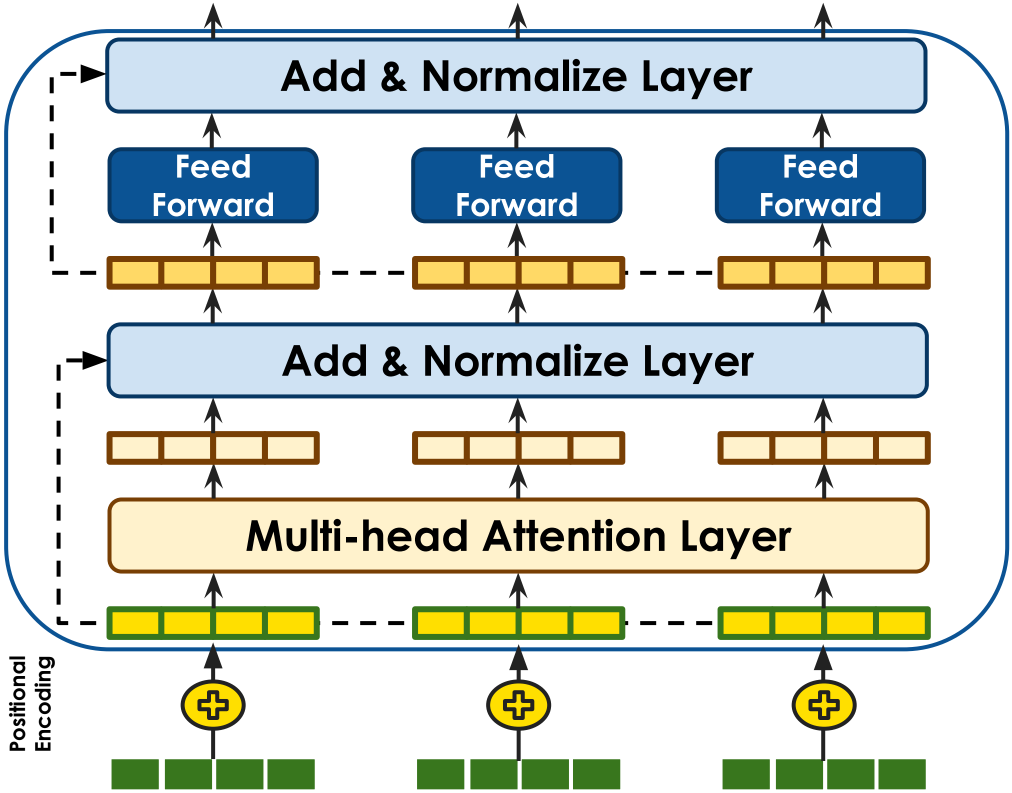
In reality, the **Encoder** and **Decoder** stage each comprise of **several individual blocks** of Encoders and Decoders.



The Transformer Model - Encoder

The Encoder block of a Transformer architecture consists of the following components:

1. **Multi-head Attention**: A stack of self-attention layers that allows the Encoder to attend to different parts of the input sequence simultaneously.
2. **Feedforward Neural Network**: Processes the outputs of the Multi-head Attention layer using a standard fully connected neural network with activations like ReLU.
3. **Residual Connections and Layer Normalization**: Improves the flow of information through the Encoder and avoids the vanishing gradient problem. These are added after each sub-layer.
4. **Positional Encoding**: Typically added to the input embeddings of the Encoder to provide positional information for words, using a set of learned sinusoidal functions.



The Transformer Model - Decoder

Most of these operations in the Decoder are identical to the Encoder.

Self-Attention

Add & Normalize Layer

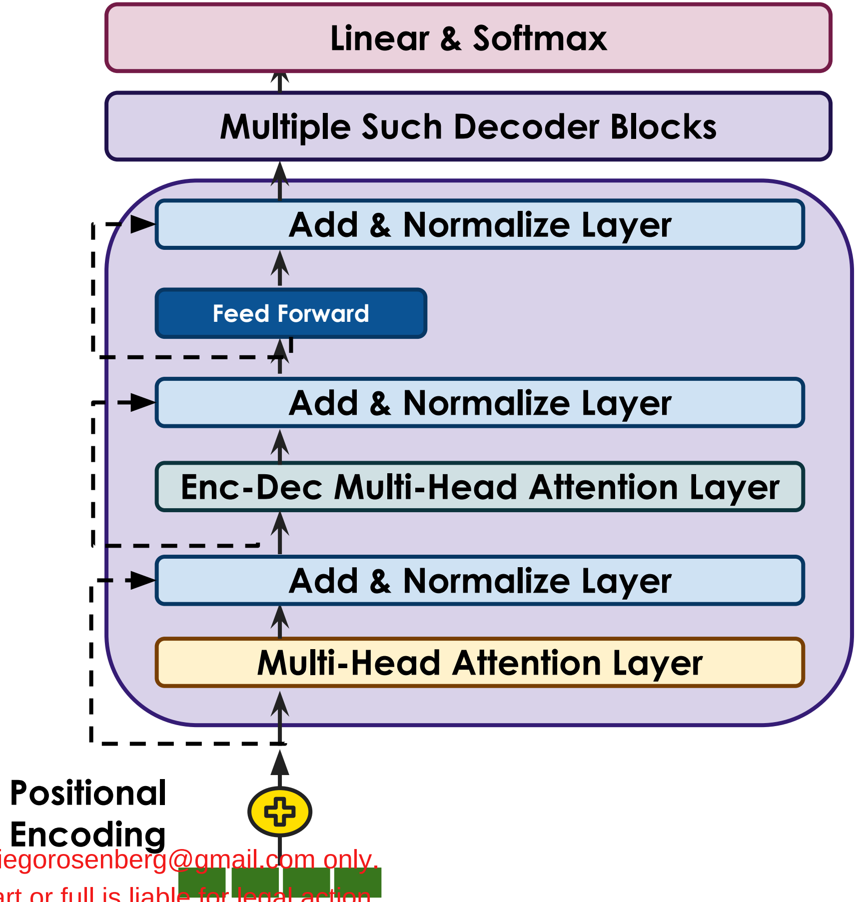
Feed Forward

But there are a few other operations **unique to the Decoder.**

Masking

Encoder-Decoder Attention Layer

Linear & Softmax



Types of Transformer Models

Types of Transformer Models

There are broadly three-types of transformer models today, based on their usage of Encoder and Decoder blocks:

- **Encoder-Decoder**
- **Encoder-only**
- **Decoder-only**

Types of Transformer Models

There are broadly three-types of transformer models today, based on their usage of Encoder and Decoder blocks:

Encoder-Decoder

Encoder-only

Decoder-only

Utilize the Encoder and Decoder blocks in tandem, similar to the original transformer architecture

Typically used in tasks where the output heavily relies on the input, like **Machine Translation** and **Text Summarization**

Examples: **T5** and **FLAN-T5**

Types of Transformer Models

There are broadly three-types of transformer models today, based on their usage of Encoder and Decoder blocks:

Encoder-Decoder

Encoder-only

Decoder-only

Utilize only Encoder blocks to generate continuous embeddings from the input

Typically used in discriminative tasks that require embeddings, like for **Text Classification** and **Semantic Search**

Examples: **BERT** and **DistilBERT**

Types of Transformer Models

There are broadly three-types of transformer models today, based on their usage of Encoder and Decoder blocks:

Encoder-Decoder

Encoder-only

Decoder-only

Utilize only Decoder blocks to auto-regressively predict* the next token based on the input

Typically used in generative tasks like **Sentence Completion** and **Question-Answering**

Examples: **GPT** and **Llama**

* Autoregressive prediction involves predicting future values based on past values

BERT

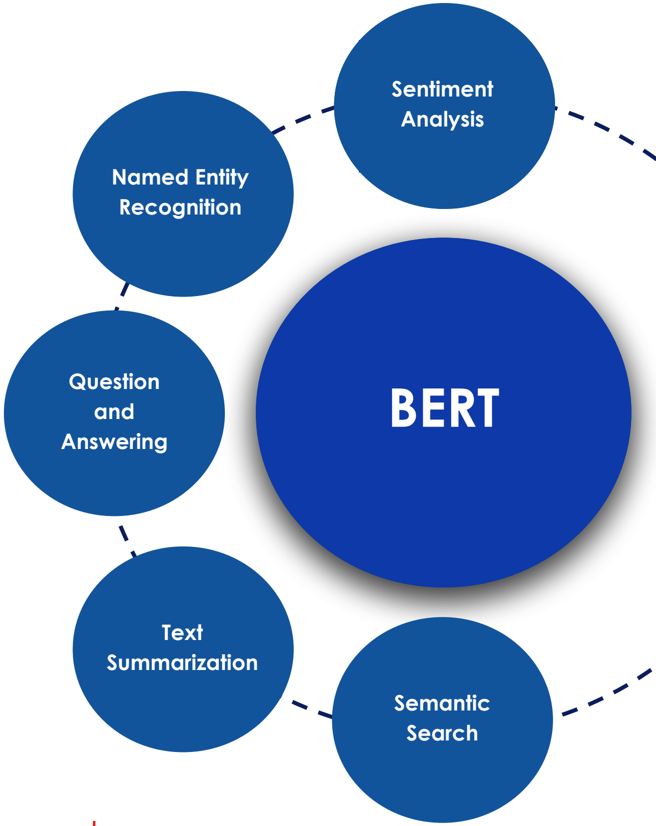
Introduction to BERT

BERT stands for **B**idirectional **E**ncoder **R**epresentations **T**ransformer

It was introduced in a **research paper** by **Jacob Devlin** and his **colleagues** at Google Research in **2018**.

BERT is an **encoder only transformer** model that leverages a **stack** of **encoders** to get an deeper **understanding** of **language context**

BERT **revolutionized** Natural Language Processing (NLP) by exhibiting **state-of-the-art performance** in a variety of NLP tasks



BERT - Architecture

There are two variants of BERT

	$\text{BERT}_{\text{BASE}}$
# parameters	110M
# encoders	12
# attention heads	12
word embeddings dimension	768

BERT - Architecture

Great Learning
POWER AHEAD

There are two variants of BERT

	$\text{BERT}_{\text{BASE}}$	$\text{BERT}_{\text{LARGE}}$
# parameters	110M	340M
# encoders	12	24
# attention heads	12	16
word embeddings dimension	768	1024

BERT - Architecture

There are two variants of BERT

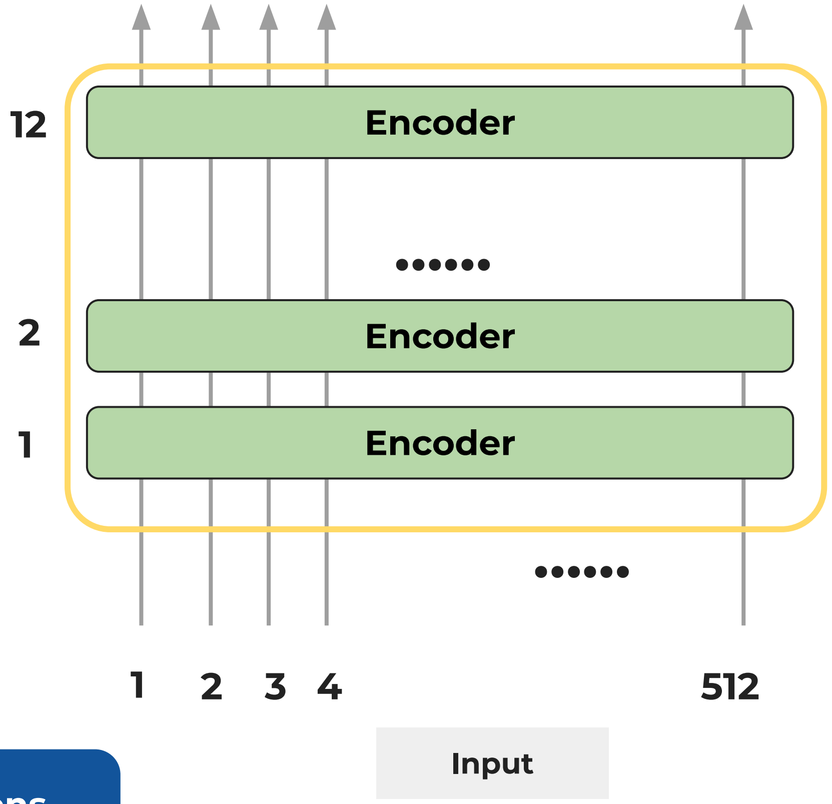
	$\text{BERT}_{\text{BASE}}$	$\text{BERT}_{\text{LARGE}}$
# parameters	110M	340M
# encoders	12	24
# attention heads	12	16
word embeddings dimension	768	1024

The **GPT 3.5** model used in ChatGPT (free tier) has **7B parameters**

The **VGG16** model used in Computer Vision has **138M parameters**

BERT - Architecture

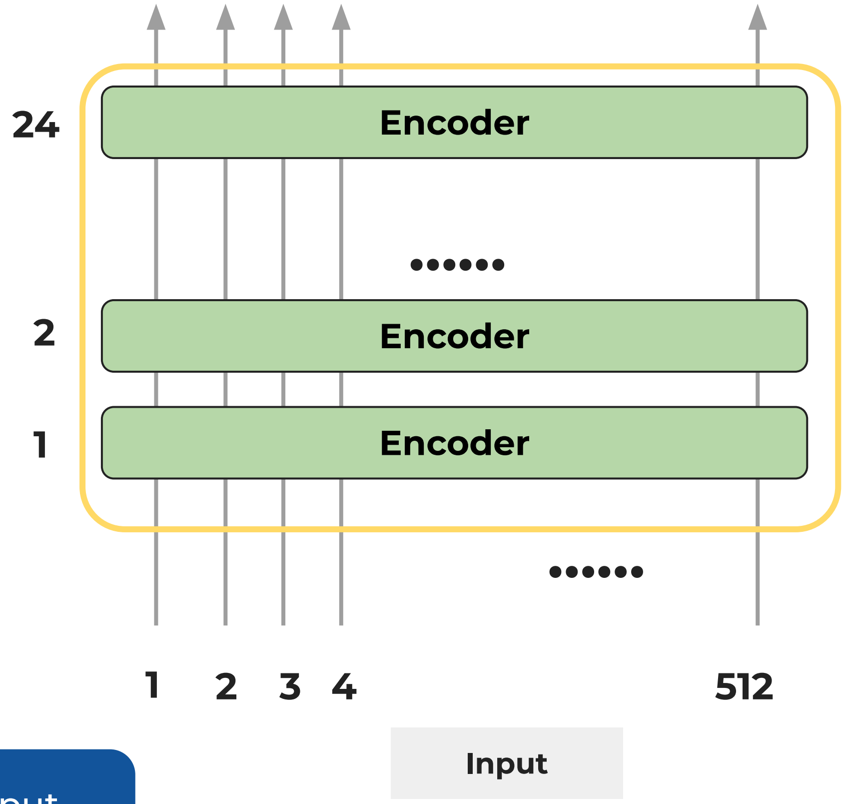
	$BERT_{BASE}$
# parameters	110M
# encoders	12
# attention heads	12
word embeddings dimension	768



$BERT_{BASE}$ has a **limitation** on the **number of input tokens** - it can take a maximum of **512** input tokens

BERT - Architecture

	$\text{BERT}_{\text{LARGE}}$
# parameters	340M
# encoders	24
# attention heads	16
word embeddings dimension	1024



$\text{BERT}_{\text{LARGE}}$ has a similar limitation on the number of input tokens as $\text{BERT}_{\text{BASE}}$

Training BERT

Training BERT

BERT's training essentially consists of two stages - a **Pre-training** Stage and a **Fine-tuning** Stage

Pre-training Stage

The model builds a foundational understanding of language

Involves exposing the model to a vast amount of text data*, where it learns about word relationships and gains contextual knowledge

Consider a novel of 500 pages, each page containing 150 words on average. The Wikipedia data used to train BERT will contain data from ~33000 such novels

*** Wikipedia Data (2,500M words), BooksCorpus Data (800M words)**

Training BERT

BERT's training essentially consists of two stages - a **Pre-training** Stage and a **Fine-tuning** Stage

Pre-training Stage + **Fine-tuning Stage**

Pre-training Stage	Fine-tuning Stage
The model builds a foundational understanding of language	The model adapts its foundational understanding of language to perform better on downstream tasks**
Involves exposing the model to a vast amount of text data*, where it learns about word relationships and gains contextual knowledge	Involves adjustment of model parameters (weights) to specialize in a particular task.

*** Wikipedia Data (2,500M words), BooksCorpus Data (800M words)**

**** Downstream tasks: Downstream tasks are those supervised-learning tasks that utilize a pre-trained model or component**

BERT - Pre-training Stage

BERT's **pre-training** stage is done in **two parts**

Pre-training Stage

Masked Language Modeling (MLM)

- Learns to predict masked (missing) words within sentences
- It sees sentences with some words intentionally masked and learns to predict these masked words based on the surrounding context.
- This task helps the model understand word relationships and meanings.

Next Sentence Prediction (NSP)

- NSP focuses on understanding relationships between pairs of sentences.
- It learns to predict whether a second sentence follows the first sentence in the text.
- This task helps the model understand the connection and coherence between sentences in longer text sequences.

BERT - Pre-training Stage

Masked Language Modeling

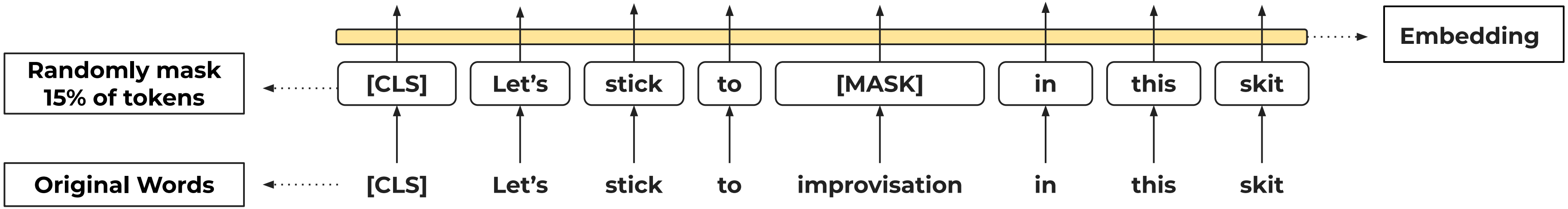
We'll elaborate on the [CLS] token shortly

Original Words	**[CLS]**	**Let's**	**stick**	**to**	**improvisation**	**in**	**this**	**skit**

The input sequence, along with a [CLS] token at the first position, is passed to BERT

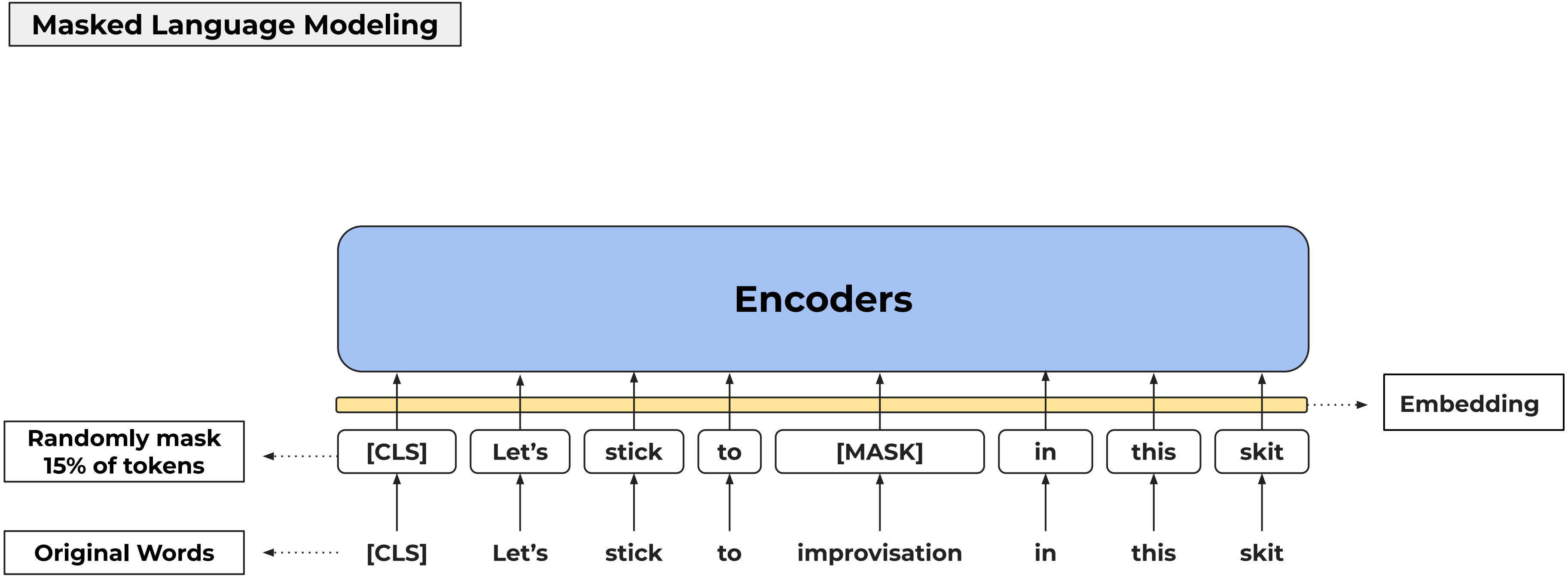
BERT - Pre-training Stage

Masked Language Modeling



Before feeding the input sequence to BERT, 15% of the words are randomly replaced with a [MASK] token.

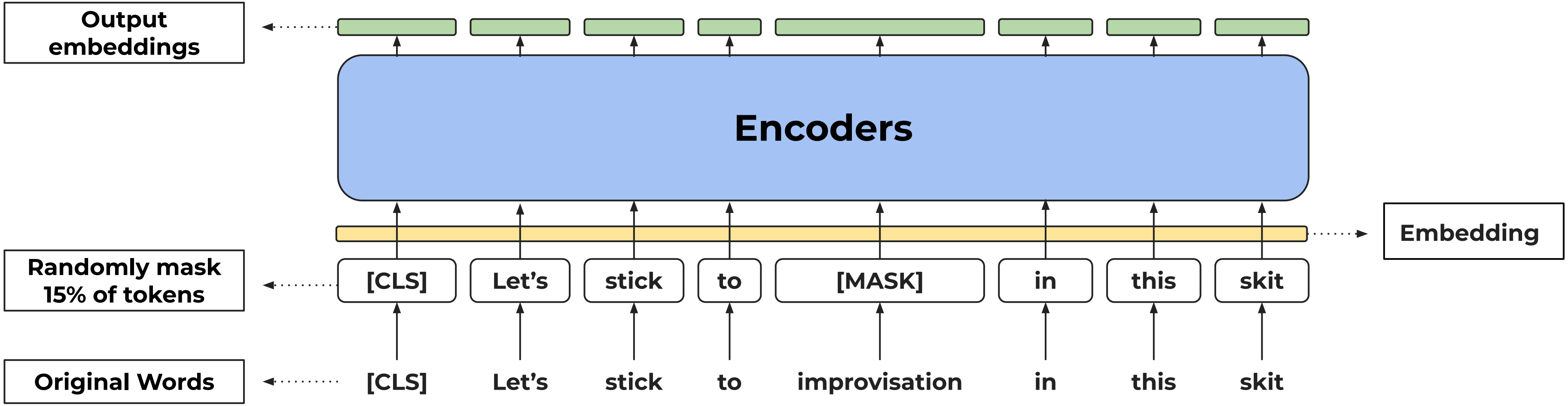
BERT - Pre-training Stage



The masked word embeddings are passed to the BERT encoders.

BERT - Pre-training Stage

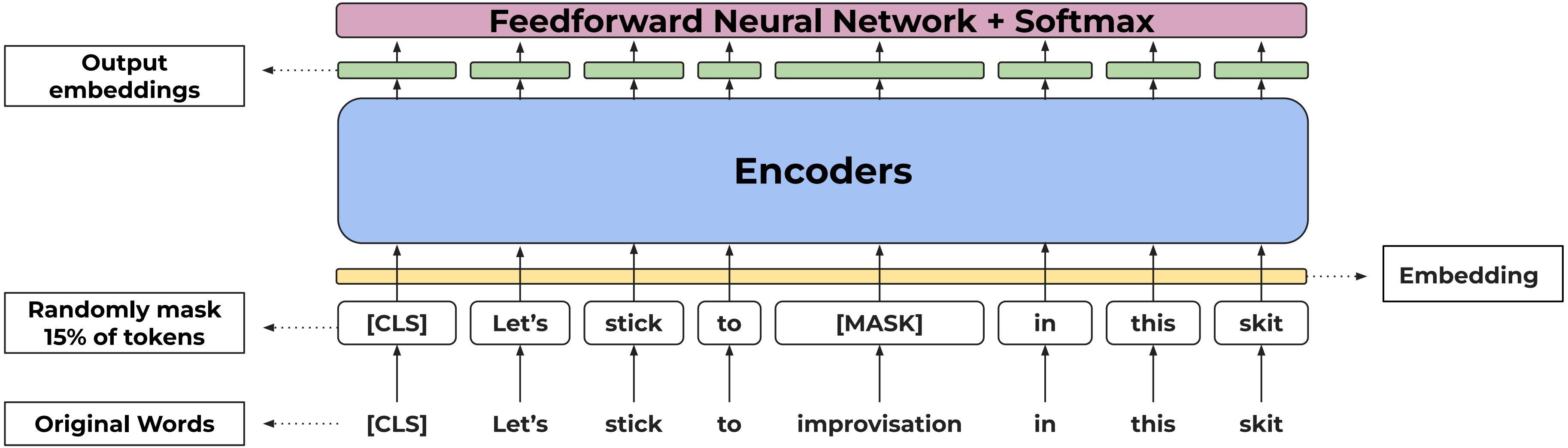
Masked Language Modeling



BERT encoders processes the input sequence, where the information flows through multiple self-attention layers within each encoder. A context-aware representation is generated (denoted here by output embeddings)

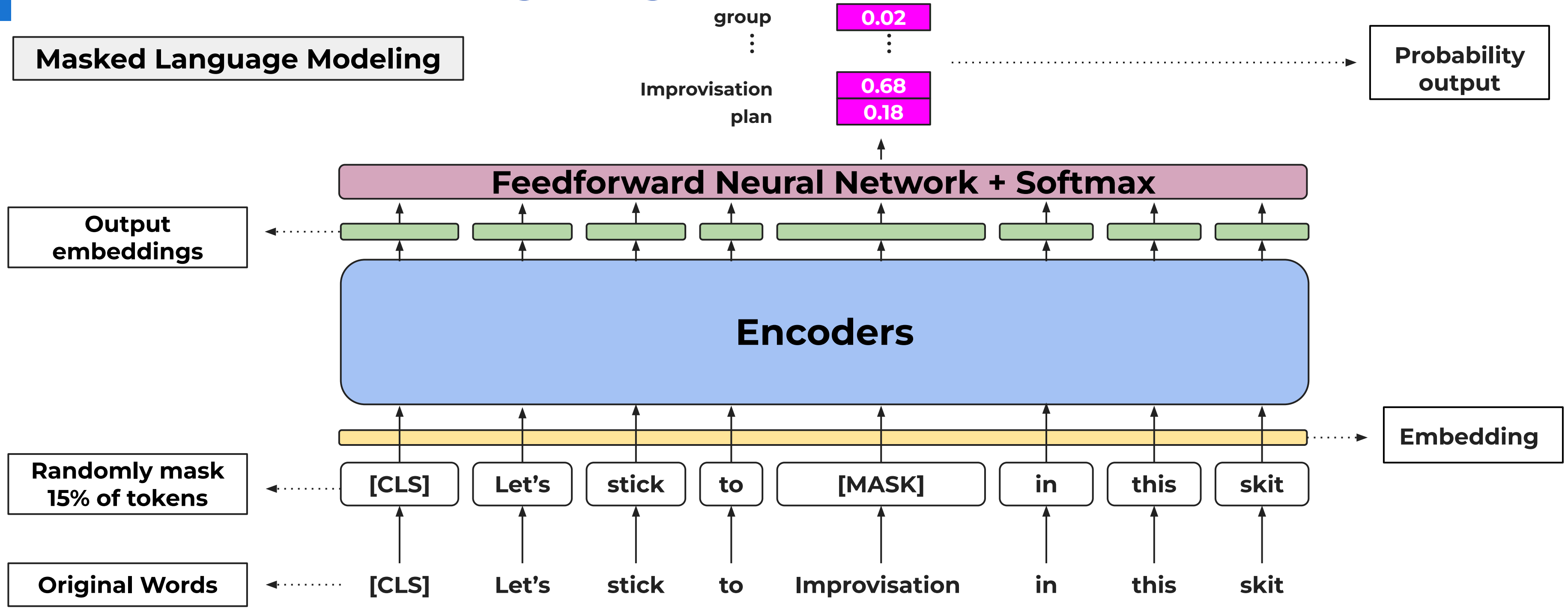
BERT - Pre-training Stage

Masked Language Modeling



The output from the stack of encoders is passed to the feed-forward neural network with softmax activation.

BERT - Pre-training Stage



The end output is a probability distribution, from which we get the model's prediction of the masked (hidden) word

BERT - Pre-training Stage

Next Sentence Prediction

In this training process, the model receives a pair of sentences as input and learns to predict whether the second sentence in the pair follows directly after the first sentence in the document.

During the training process, half of the inputs sequence consists of pairs where the second sentence follows directly after the first sentence.

The other half includes pairs where a sentence selected randomly from the corpus is used as the second sentence.

BERT - Pre-training Stage

Next Sentence Prediction

[CLS] is a special token used for classification

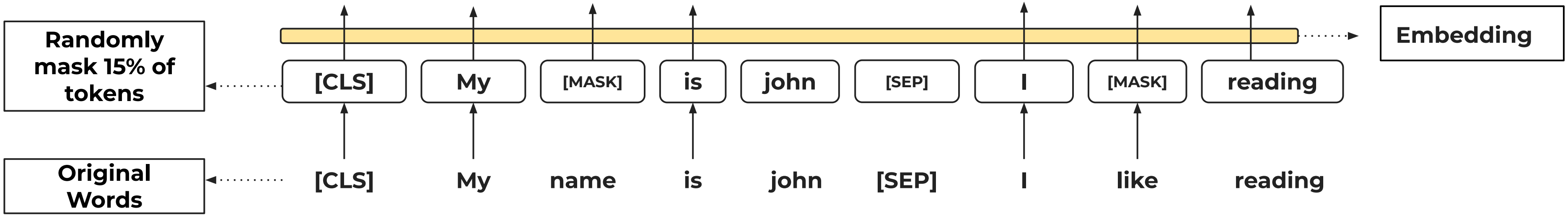
It appears at the very beginning of each sentence, and has a fixed embedding and positional embedding

Original Words	[CLS]	My	name	is	john	[SEP]	I	like	reading	[SEP]

The input sequence consists of two sentences, along with a [CLS] token at the first position, and a [SEP] token to separate the two sentences

BERT - Pre-training Stage

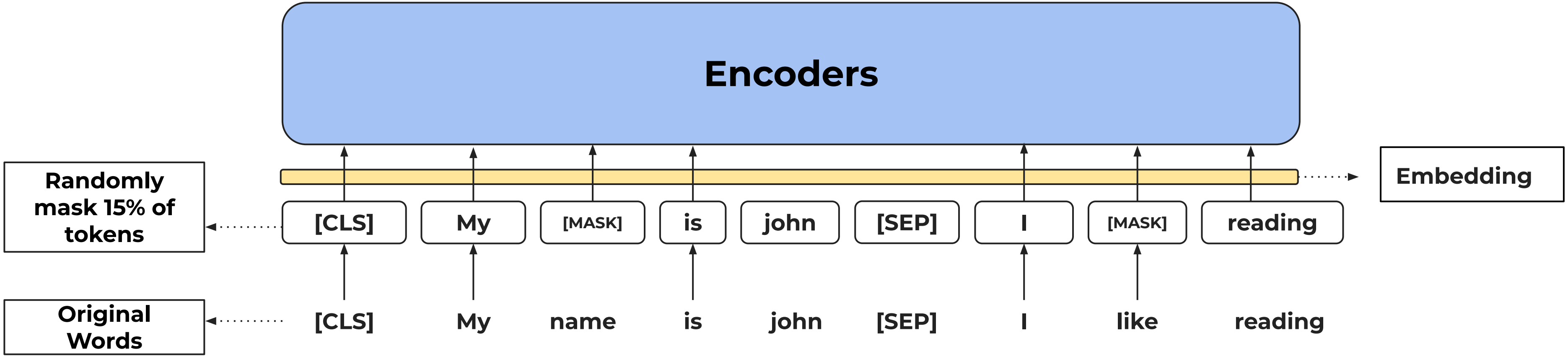
Next Sentence Prediction



Before feeding the input sequence to BERT, 15% of the words are randomly replaced with a [MASK] token.

BERT - Pre-training Stage

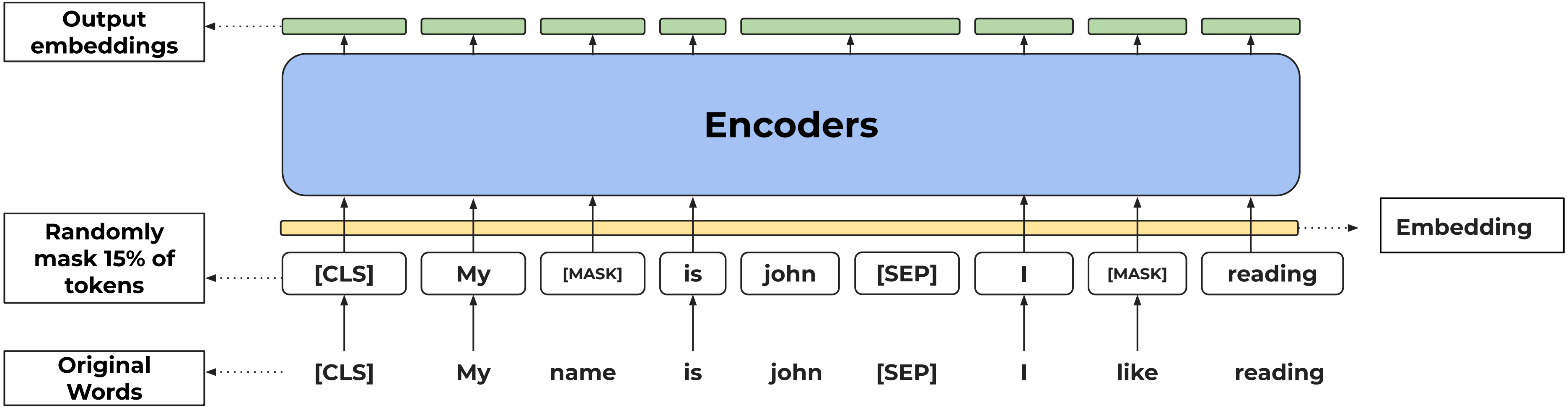
Next Sentence Prediction



The masked word embeddings are passed to the BERT encoders.

BERT - Pre-training Stage

Next Sentence Prediction

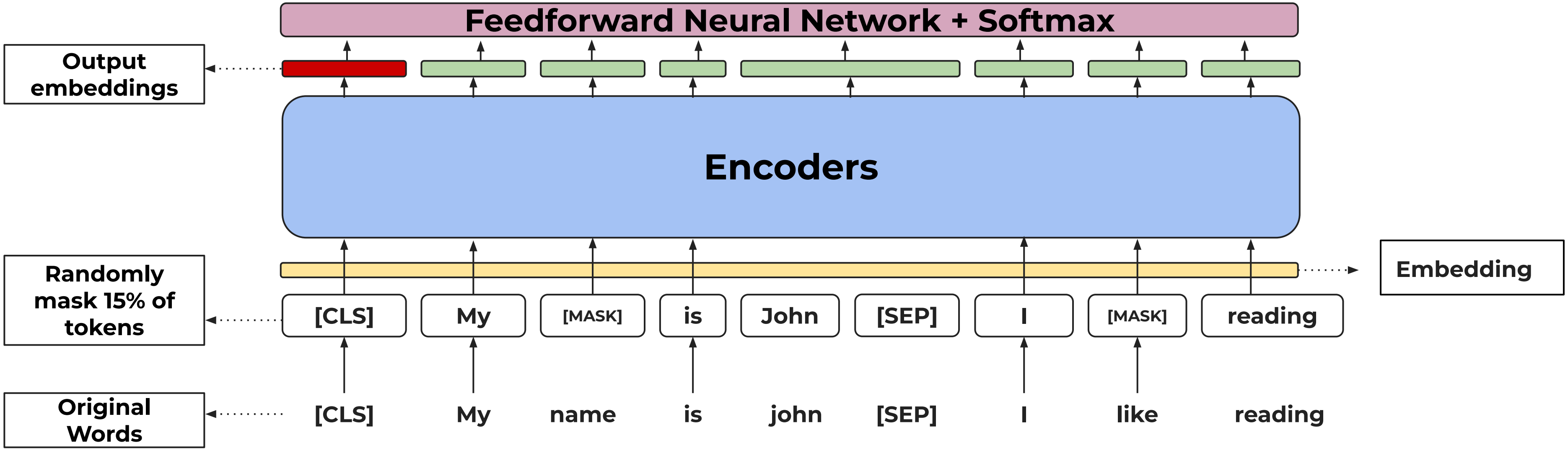


BERT encoders processes the input sequence, where the information flows through multiple self-attention layers within each encoder. A context-aware representation is generated (denoted here by output embeddings)

BERT - Pre-training Stage

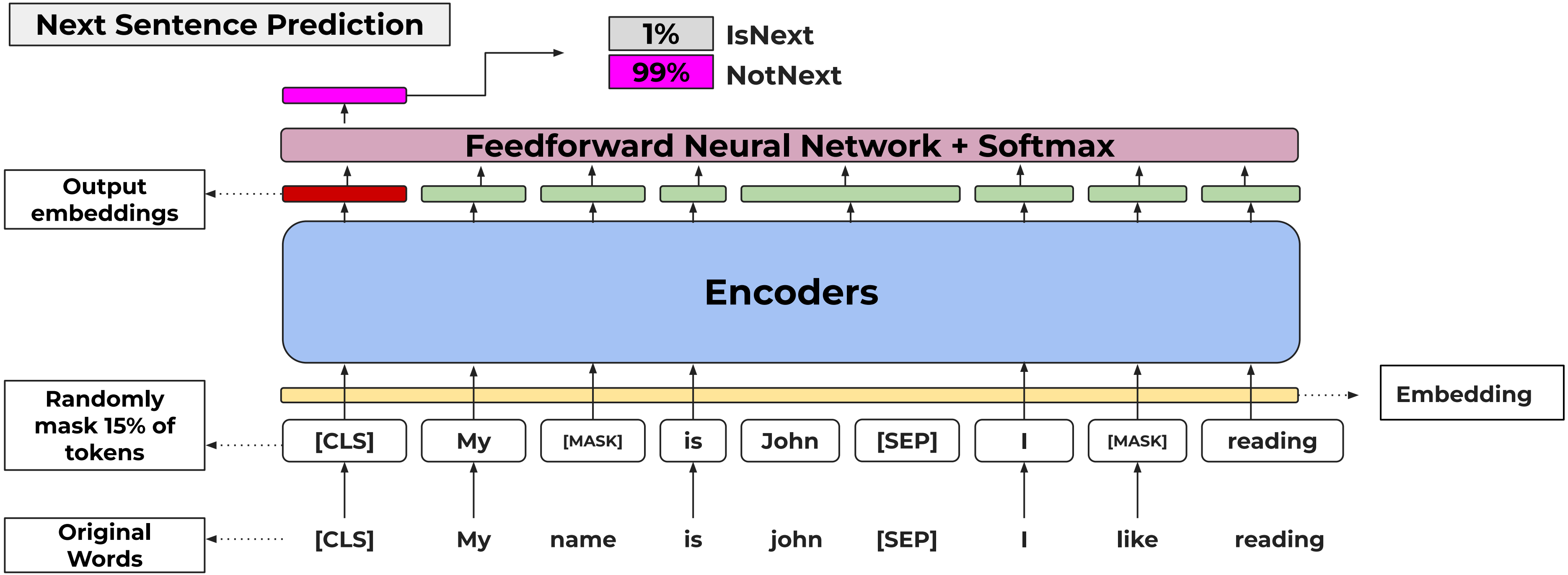
Next Sentence Prediction

The output of [CLS] is helpful because it contains BERT's understanding at the sentence-level



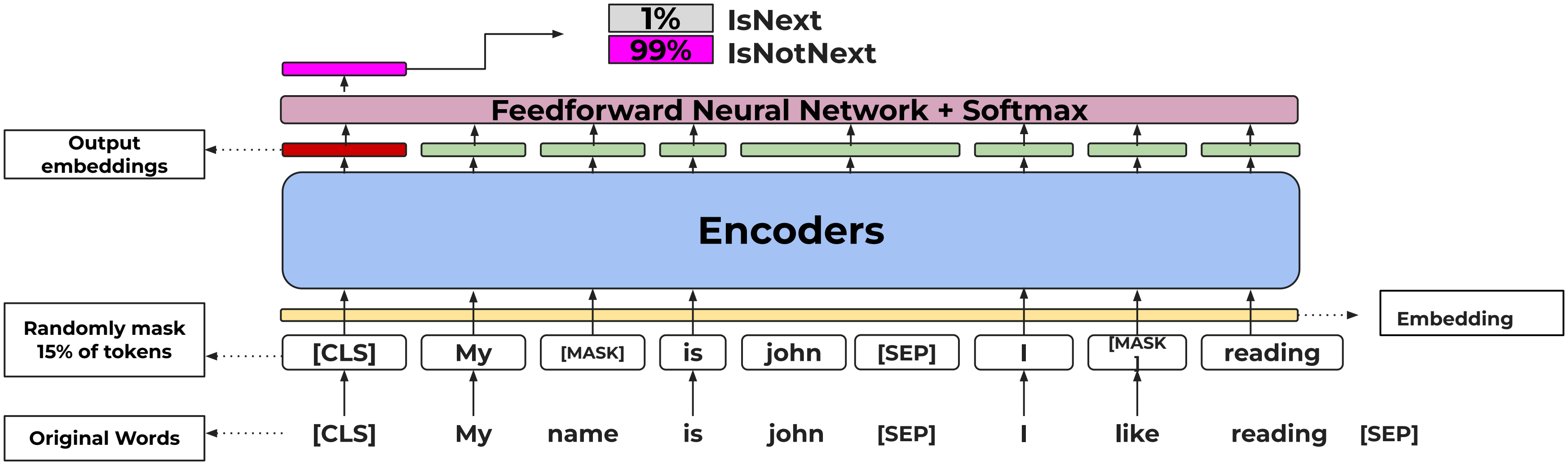
The output from the stack of encoders is passed to the feed-forward neural network with softmax activation.

BERT - Pre-training Stage



The end output is a probability distribution from which we obtain the model's prediction about whether the pair of sentences is 'IsNext' or 'NotNext'

BERT - Pre-training Stage



Training of BERT via **MLM** and **NSP** is done **simultaneously -** hence the need for [CLS] token

Input sequences contain **masked words** that **BERT** aims to **predict** [MLM], and also learns to understand **relationships between sequences** through a **separate task** [NSP].

BERT - Fine-tuning

Pre-training Stage

The model builds a foundational understanding of language

Learned from Wikipedia data and a collection of 11038 free novel books (BooksCorpus data)

Might not have learned the necessary language nuances for understanding business-specific data

For example, customer review are very differently framed compared to novels and Wikipedia articles

BERT - Fine-tuning

Pre-training Stage

Fine-tuning Stage

The model adapts its foundational understanding of language

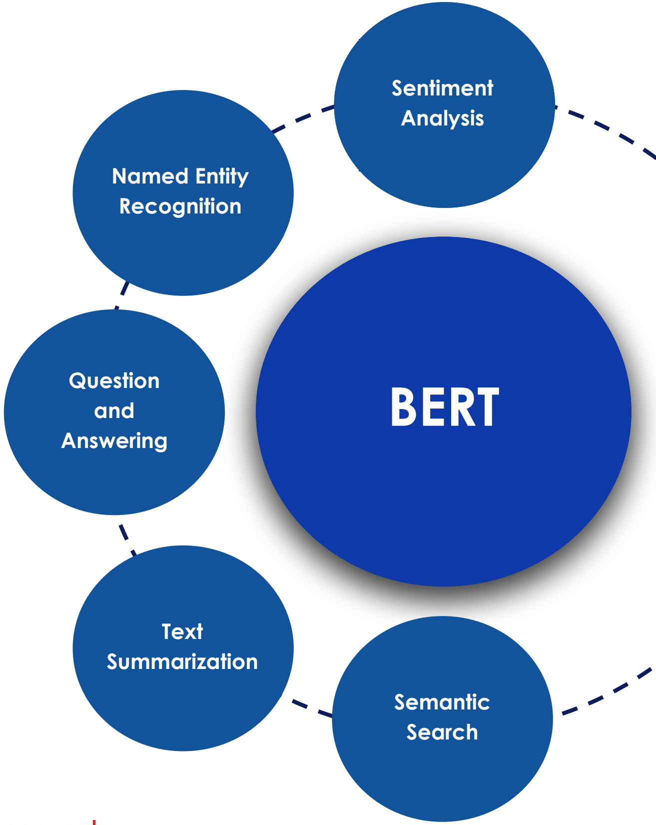
Exposed to use case specific data to understand the nuances and adjusts its parameter as per the requirement of the task at hand

For example, it tries to capture the nuances between customer reviews which has different structure but same meaning

"The phone is great! The battery is good, but the camera is not good enough" vs *"good phone, bad camera"*

BERT - Fine-tuning

Fine-tuning allows BERT to perform better across a wide variety of language tasks, while only minor modifications to the model weights

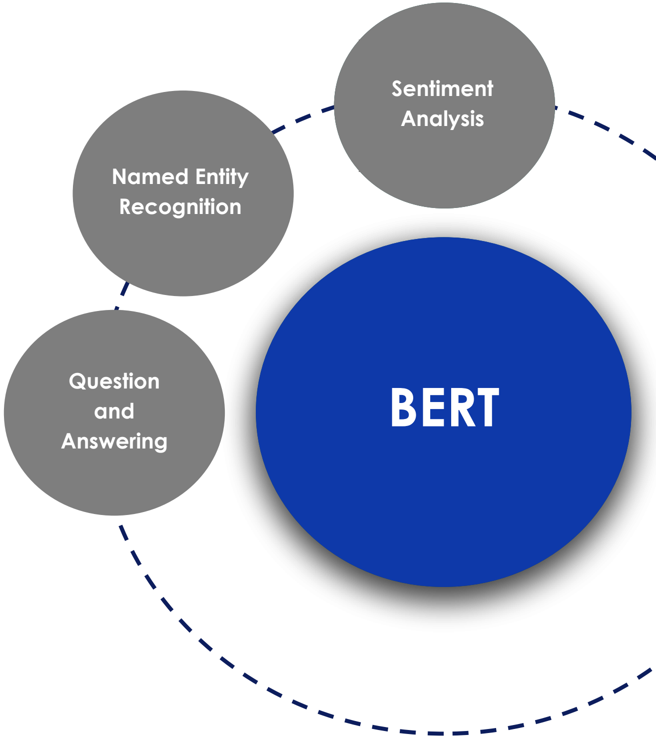


BERT - Fine-tuning

Fine-tuning allows BERT to perform better across a wide variety of language tasks, while only minor modifications to the model weights

Let's explore how fine-tuning of BERT is done in three different use cases

We'll understand how the problem Let's explore how fine-tuning of BERT is done in three different use cases



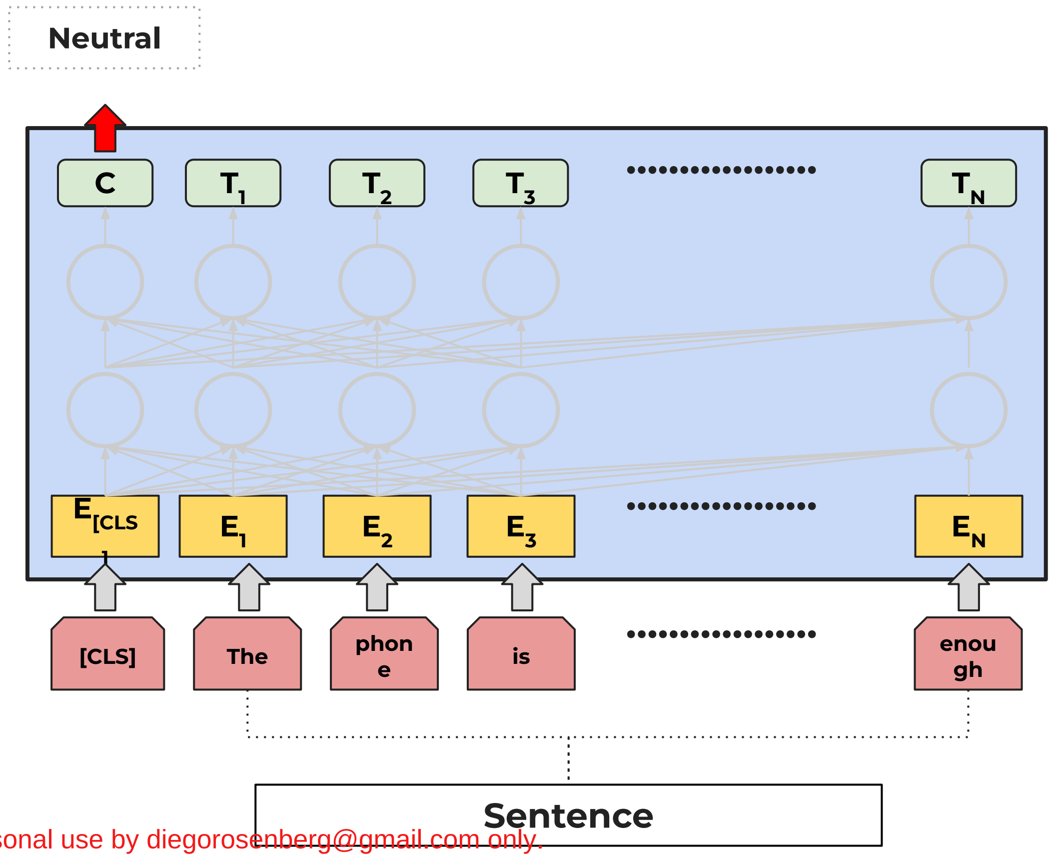
BERT - Fine-tuning

Sentiment Classification

For sentiment analysis, the functioning of BERT is same as that for NSP

Encoder process the input sequence and outputs the content aware representation for each of the token.

This representation is passed to a FFNN with softmax, which outputs a probability distribution, and the sentiment with the highest probability is taken as the predicted sentiment



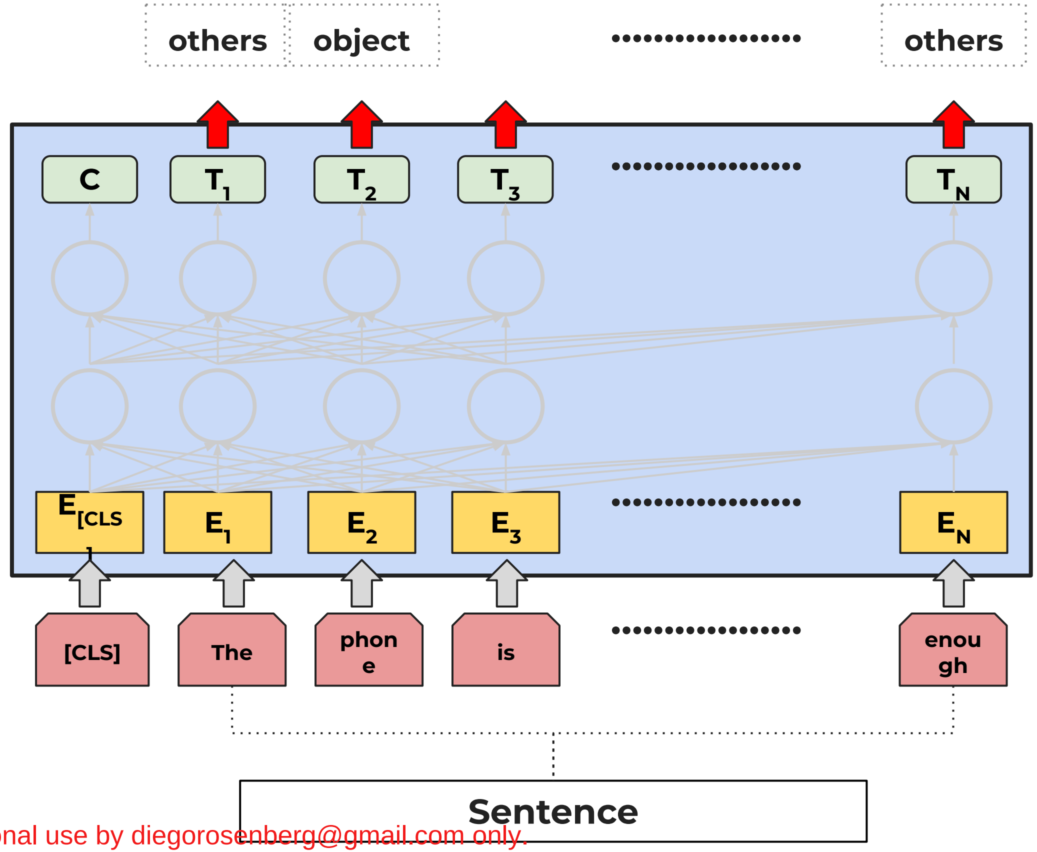
BERT - Fine-tuning

Named Entity Recognition

Named entity recognition (NER) involves extracting and categorizing detected entities in a text into predetermined categorie

For named entity recognition, the functioning of BERT is slightly different compared to that for NSP

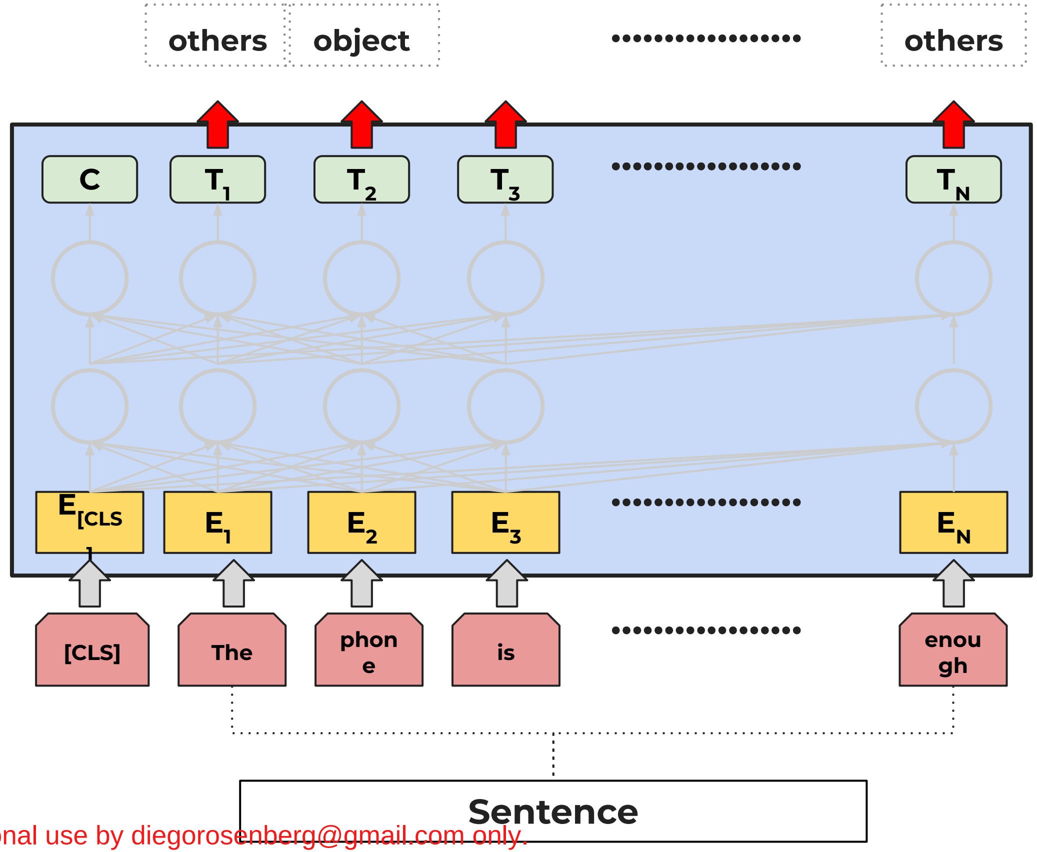
It ignores the [CLS] output and forwards all other outputs from the encoder to the Feed-Forward Neural Network (FFNN) with softmax



BERT - Fine-tuning

FFNN outputs a probability distribution for each token's entity label

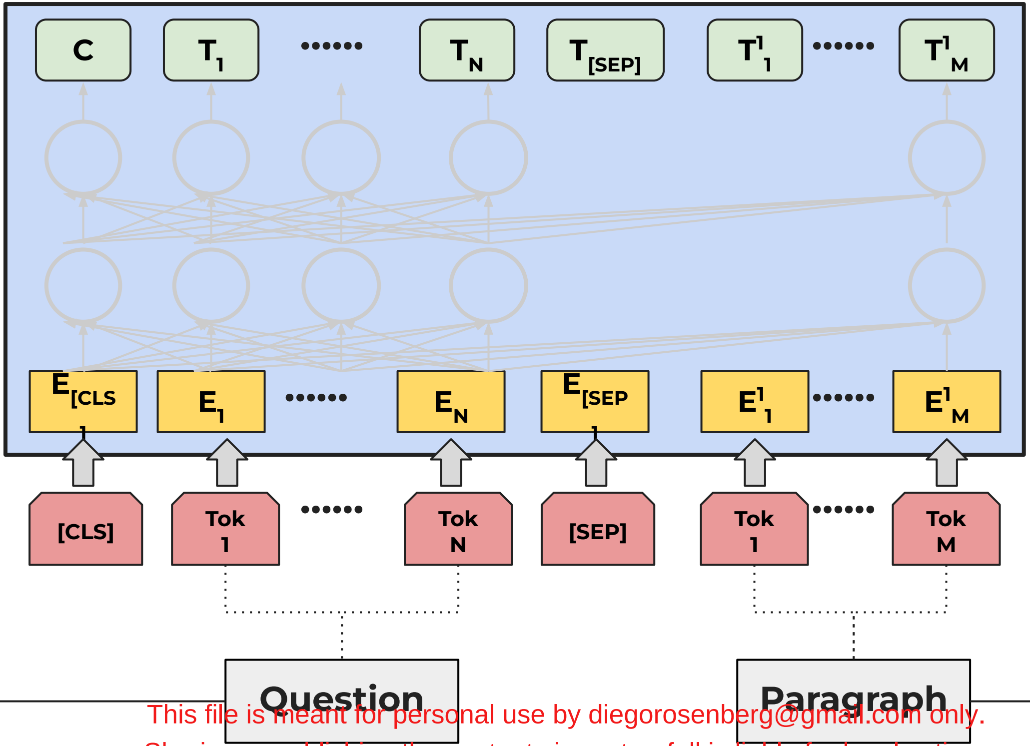
The token with the highest for a specific entity label is considered as part of that named entity.



BERT - Fine-tuning

Question Answering

How many attention heads does each transformer encoder in BERT have?

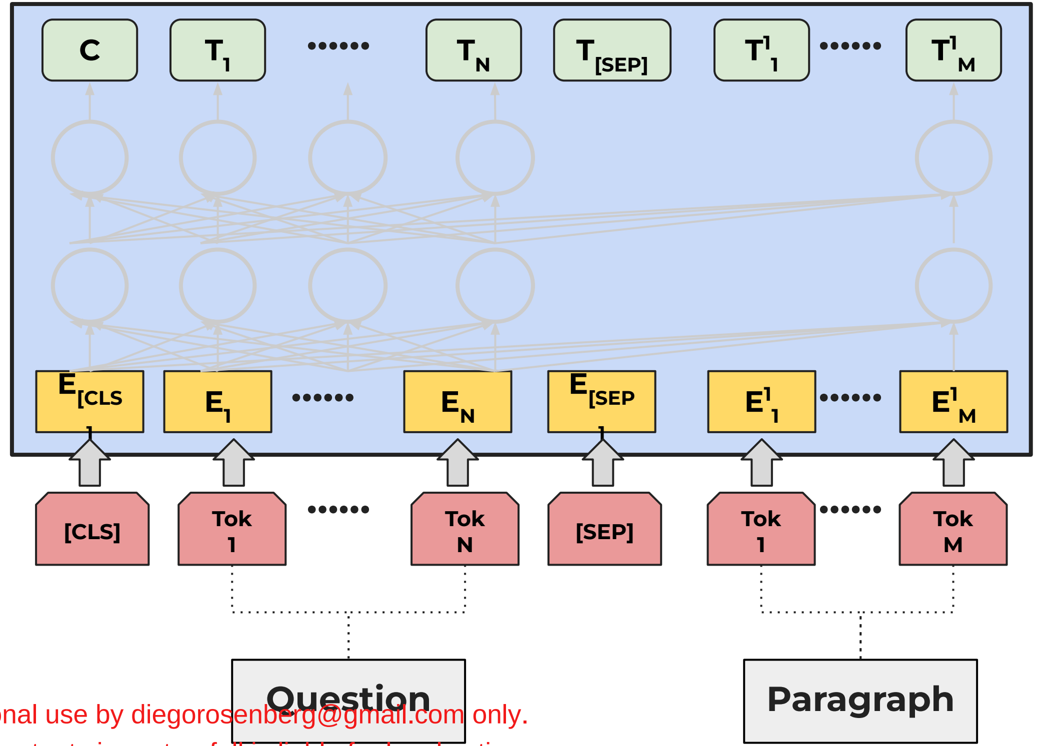


BERT consists of 12 transformer encoders with 12 attention heads in each, totaling approximately 110 million parameters. This design allows BERT to deeply understand language nuances and relationships, making it adept at various language tasks.

BERT - Fine-tuning

For Question Answering task, question and paragraph (context) separated by [SEP] token

Encoders generates contextual representations for each token in question and paragraph capturing their relationship

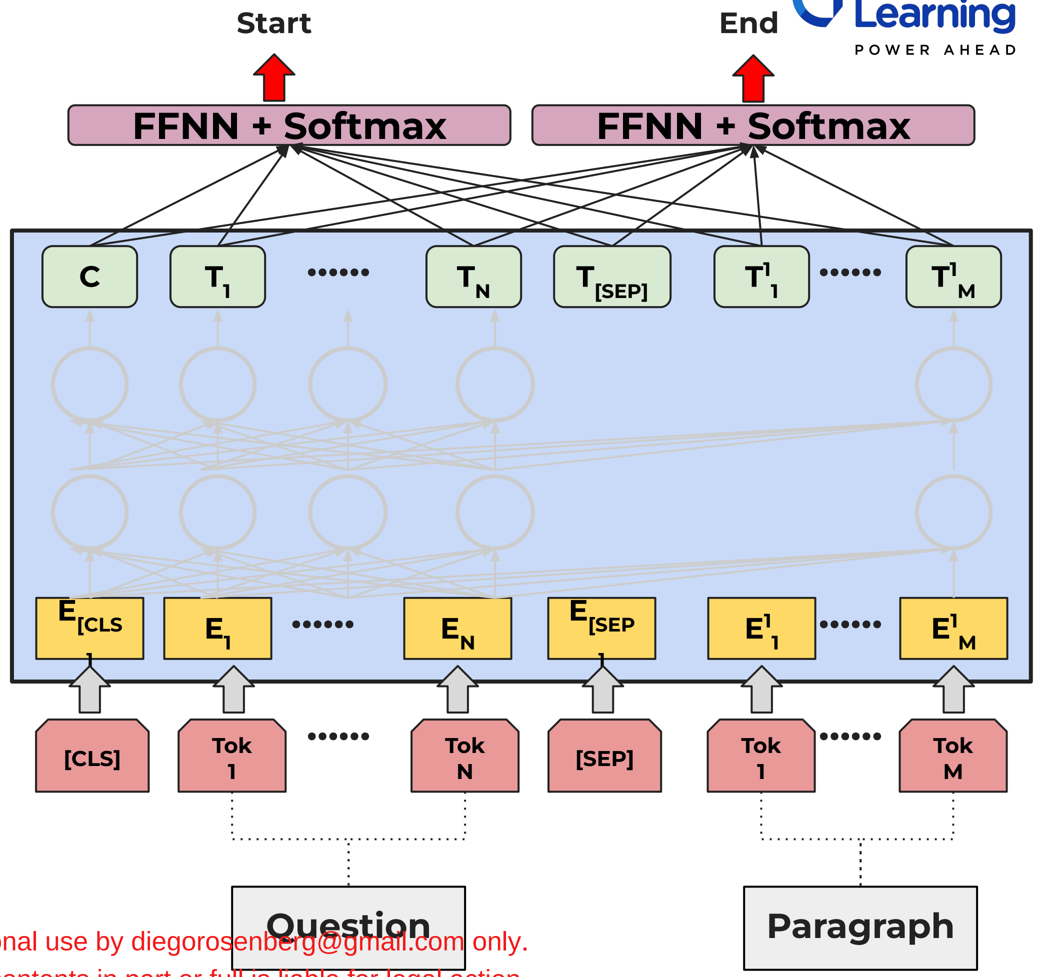


BERT - Fine-tuning

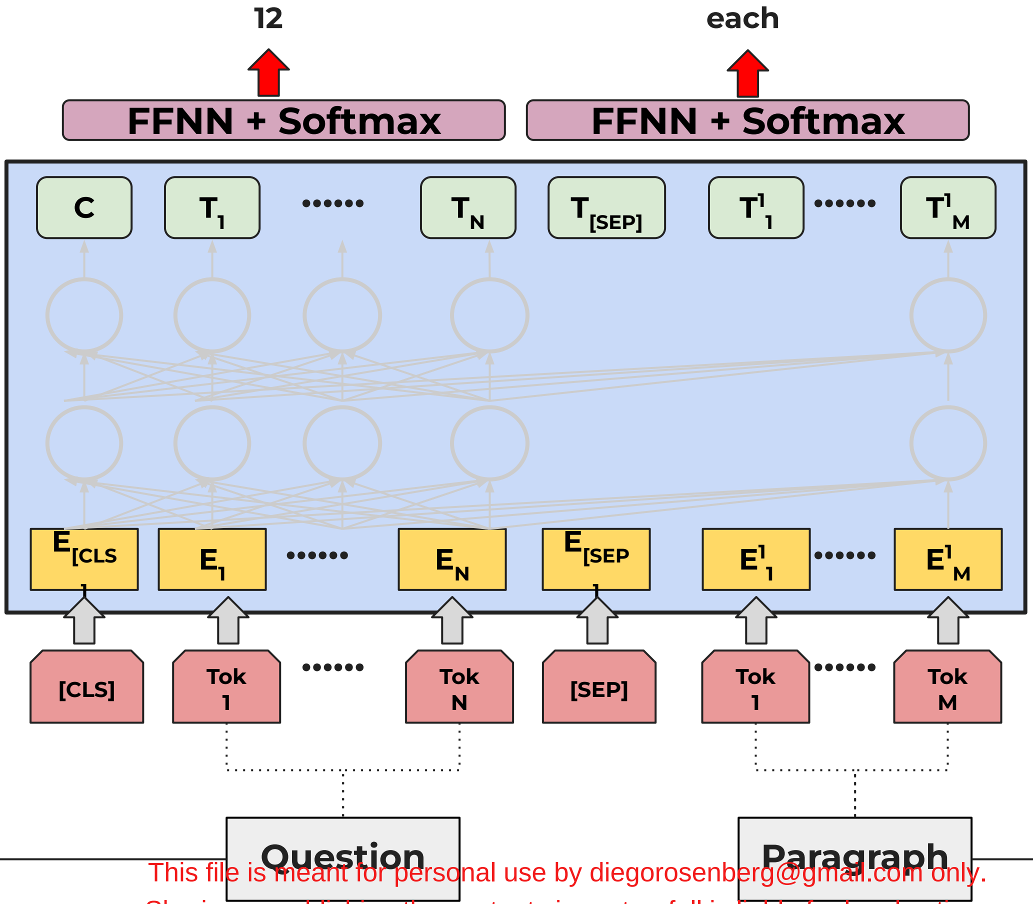
The output of the encoders is passed to two classifiers - one to predict the starting token of the answer and one for the ending token

Both classifiers output probability distributions over all the tokens

The tokens having the highest probability are taken as the start and end tokens of the answer.



BERT - Fine-tuning



How many attention heads does each transformer encoder in BERT have?

BERT consists of 12 transformer encoders with **12 attention heads in each**, totaling approximately 110 million parameters. This design allows BERT to deeply understand language nuances and relationships, making it adept at various language tasks.

Extensions of BERT

Extensions of BERT

Since the release of the original BERT model, researchers have come up with different extensions of BERT

Each extension builds on the original BERT model to add computation and/or performance gains

We'll discuss three common extensions of BERT

RoBERTa

XLNET

DistilBERT

RoBERTa

RoBERTa stands for A **R**obustly **o**ptimized **BERT** Pretraining **a**pproach

It was developed by **Facebook AI Research (FAIR)** in **2019**

RoBERTa has the exact **same architecture as BERT, but** the **Pre-training** Approach was **changed**

Masked Language Modeling is **replaced** with **Randomly Masked Language Modeling**

Next Sentence Prediction is **discarded**

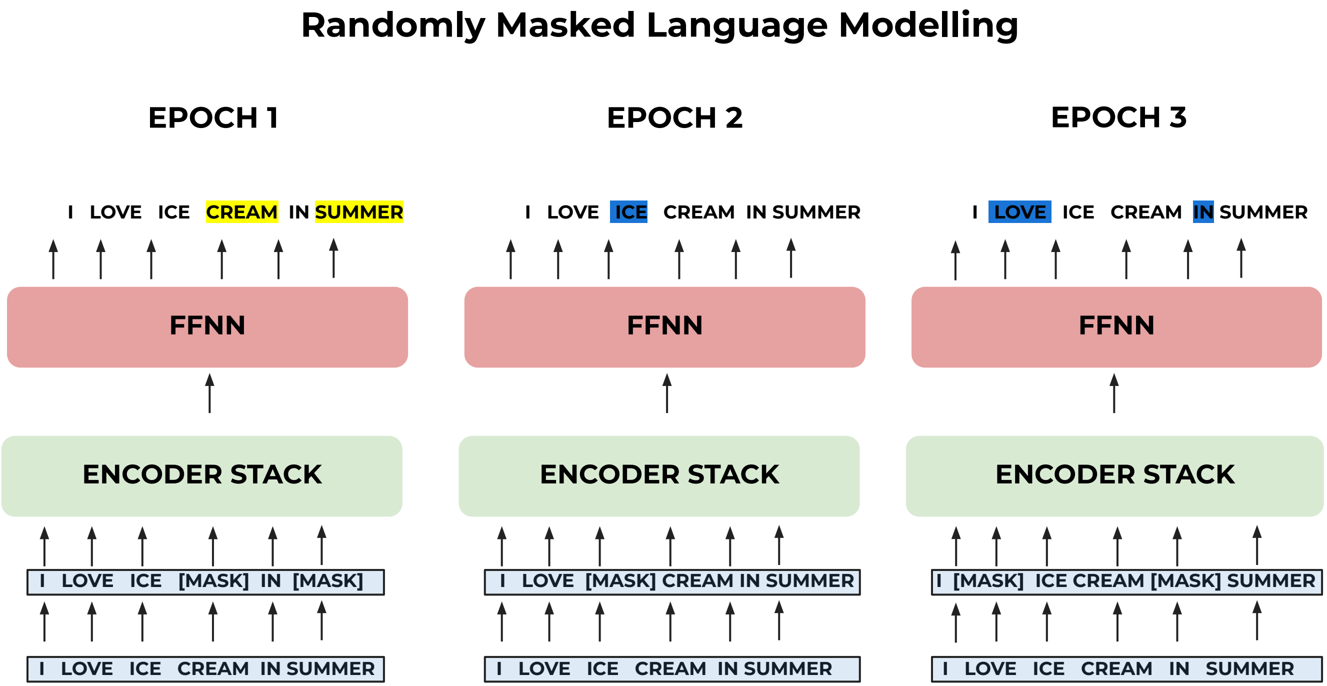
Trained on a larger dataset*

*** Wikipedia Data (2,500M words), BooksCorpus Data (800M words), CC-News (63M news articles), OpenWebText (38 GB), Stories(31 GB)**

RoBERTa - Training

Randomly Masked Language Modeling is a technique where **words** in a sentence are **randomly masked**.

The choice of words being **masked** changes **randomly** in **each epoch** of training.



XLNet

XLNet stands for e**X**treme **L**earning Machine **Net**work

It was developed by **Google AI Brain team** which was released around the same time as RoBERTa in 2019

XLNet has the exact **same architecture as BERT, but** the **Pre-training** Approach was **changed**

- **Masked Language Modeling** is **replaced** with **Permutation Language Modeling**
- **Next Sentence Prediction** is **discarded**
- **Trained on a larger dataset***

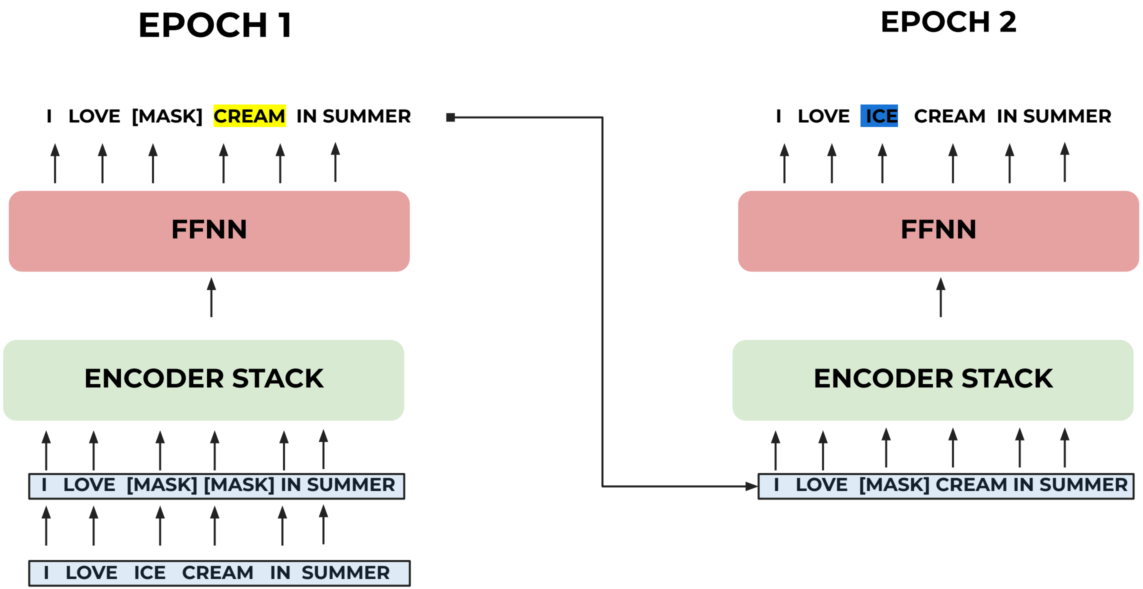
*** Wikipedia Data (2,500M words), BooksCorpus Data (800M words), Giga5 (16 GB text), ClueWeb 2012-B, Common Crawl**

XLNet - Training Process

In **Permutation Language Modeling**, instead of predicting all the masked tokens in one run, the model predicts one masked token, and then used this prediction (and future ones) to predict all subsequent masked tokens

This approach is known as autoregressive prediction, where the past predictions are used for the next set of predictions

Permutation Language Modelling



DistilBERT

DistilBERT uses the process of **knowledge distillation**

It was developed by **HuggingFace team** and was released a couple of months after RoBERTa in 2019

DistilBERT has a **smaller architecture than BERT**, but was trained on the same data as BERT

Masked Language Modeling and **Next Sentence Prediction**

Smaller architecture (6 encoder layers vs 12 of $\text{BERT}_{\text{BASE}}$)

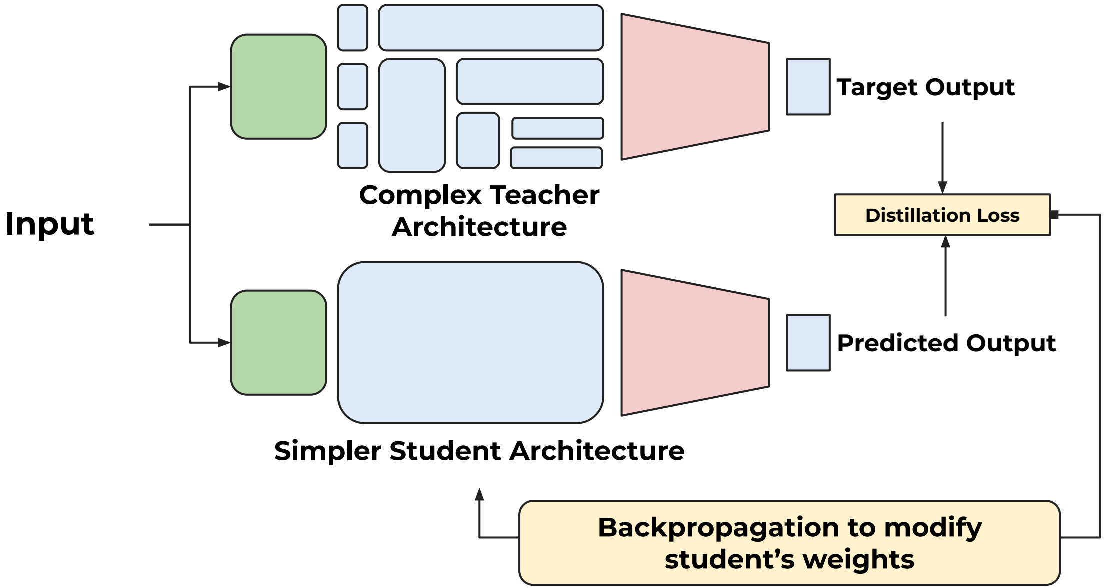
Trained on the same data

DistilBERT - The Idea of Distillation

The idea of **knowledge distillation**, also called **Teacher-Student Training**, is one where a simpler ‘Student’ model is trained to replicate the BERT (‘Teacher’) performance

Training a Student Model on the output of a Teacher Model in this manner forces it to become more efficient and perform nearly as well as a more complex model.

The idea of distillation can be applied to any architecture as the Teacher



Useful in **resource-constrained settings** such as individual laptops or mobile phones that need to deploy heavy Deep Learning applications.

Comparison - BERT and its Extensions

	BERT	RoBERTa	DistilBERT	XLNet
Parameters (millions)	Base: 110 Large: 340	Base: 110 Large: 340	Base: 110	Base: ~110 Large: ~340
Training Time	Base: X * Large: Y **	Large: 4-5Y	Base: 0.25X	Large: 4Y
Performance	SOTA in Oct 2018	2 - 20% > BERT	3% < BERT	2 - 15% > BERT

SOTA - State of the Art *** X - 8 x Nvidia V100 GPU x 12 days** **** Y - 280 x Nvidia V100 GPU x 1 day**

Nvidia V100 is the most advanced data center GPU ever built to accelerate AI, high performance computing, data science and graphics

Happy Learning !