



## Next Article:

[Natural Language Processing \(NLP\) - Overview](#)

## BERT Model - NLP

Last Updated : 17 Jul, 2025

BERT (Bidirectional Encoder Representations from Transformers) stands as an open-source machine learning model designed for the natural language processing (NLP). The article aims to explore the architecture, work of BERT.

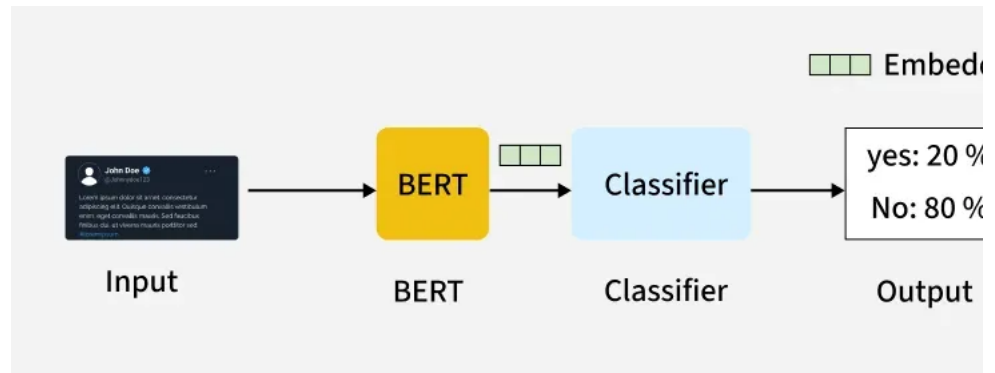


Illustration of BERT Model Use Case

### What is BERT?

**BERT (Bidirectional Encoder Representations from Transformers)** leverages a transformer-based neural network to understand and generate human-like language. BERT employs an encoder-only architecture. In the traditional **encoder-decoder architecture**, there are both encoder and decoder modules. The decision to use an encoder-only architecture suggests a primary emphasis on understanding input sequences rather than generating output sequences.

Traditional language models process text sequentially, either from left to right or right to left. This means the model has no awareness to the immediate context preceding the target word. BERT uses a bi-directional approach, processing both the left and right context of words in a sentence, instead of analyzing the text sequentially. BERT processes all words in a sentence simultaneously.

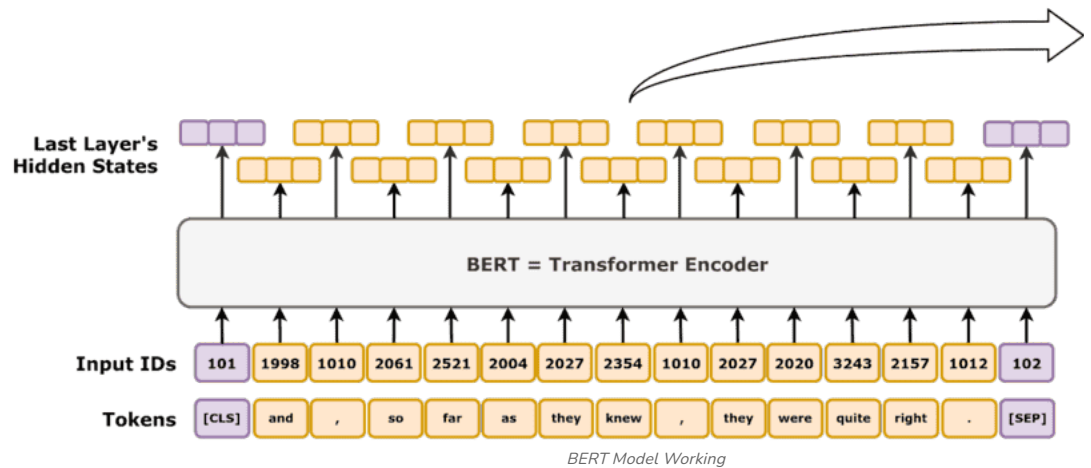
### Pre-training BERT Model

The BERT model undergoes Pre-training on Large amounts of unlabeled text to learn contextual embeddings.

- BERT is pre-trained on large amount of unlabeled text data. The model learns contextual embeddings or representations of words that take into account their surrounding context in a sentence.
- BERT engages in various unsupervised pre-training tasks. For instance, it might learn to predict missing words in a sentence (Masked Language Model or MLM task), understand the relationship between two sentences, or predict the next sentence in a pair.

### Workflow of BERT

BERT is designed to generate a language model so, only the encoder mechanism is used. The input sequence is processed by the Transformer encoder. These tokens are first embedded into vectors and then processed in the neural network. The output is a sequence of vectors, each corresponding to an input token, providing contextualized representations. When training language models, defining a prediction goal is a challenge. Many models predict the next word in a sequence, which is a directional approach and may limit context learning.



BERT addresses this challenge with two innovative training strategies:

1. [Masked Language Model \(MLM\)](#).
2. [Next Sentence Prediction \(NSP\)](#).

### 1. Masked Language Model (MLM)

In BERT's pre-training process, a portion of words in each input sequence is masked and the model is original values of these masked words based on the context provided by the surrounding words.

1. BERT adds a classification layer on top of the output from the encoder. This layer is important for p words.
2. The output vectors from the classification layer are multiplied by the embedding matrix, transformi vocabulary dimension. This step helps align the predicted representations with the vocabulary spa
3. The probability of each word in the vocabulary is calculated using the [SoftMax activation function](#). a probability distribution over the entire vocabulary for each masked position.
4. The loss function used during training considers only the prediction of the masked values. The moc the deviation between its predictions and the actual values of the masked words.
5. The model converges slower than directional models because during training, BERT is only concern the masked values, ignoring the prediction of the non-masked words. The increased context aware through this strategy compensates for the slower convergence.

### 2. Next Sentence Prediction (NSP)

BERT predicts if the second sentence is connected to the first. This is done by transforming the output into a  $2 \times 1$  shaped vector using a classification layer, and then calculating the probability of whether tl follows the first using SoftMax.

1. In the training process, BERT learns to understand the relationship between pairs of sentences, pre sentence follows the first in the original document.
2. 50% of the input pairs have the second sentence as the subsequent sentence in the original docum 50% have a randomly chosen sentence.
3. To help the model distinguish between connected and disconnected sentence pairs. The input is pr entering the model.
4. BERT predicts if the second sentence is connected to the first. This is done by transforming the out token into a  $2 \times 1$  shaped vector using a classification layer, and then calculating the probability of v sentence follows the first using SoftMax.

*During the training of BERT model, the Masked LM and Next Sentence Prediction are trained together to minimize the combined loss function of the Masked LM and Next Sentence Prediction, leading to a language model with enhanced capabilities in understanding context within sentences and relationships between sentences.*

## Why to train Masked LM and Next Sentence Prediction together?

Masked LM helps BERT to understand the context within a sentence and [Next Sentence Prediction](#) helps in understanding the connection or relationship between pairs of sentences. Hence, training both the strategies together enables the model to learn a broad and comprehensive understanding of language, capturing both details within sentences and relationships between sentences.

## Fine-Tuning on Labeled Data

We perform Fine-tuning on labeled data for specific [NLP](#) tasks.

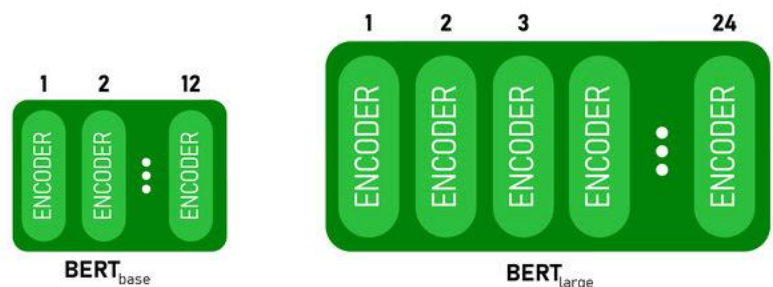
- After the pre-training phase, the BERT model, armed with its contextual embeddings, is fine-tuned for various language processing (NLP) tasks. This step tailors the model to more targeted applications by adapting its language understanding to the nuances of the particular task.
- BERT is fine-tuned using labeled data specific to the downstream tasks of interest. These tasks can include sentiment analysis, question-answering, [named entity recognition](#), or any other NLP application. The model's architecture is adjusted to optimize its performance for the particular requirements of the task at hand.

BERT's unified architecture allows it to adapt to various downstream tasks with minimal modifications, making it a versatile and highly effective tool in [natural language understanding](#) and processing.

## BERT Architecture

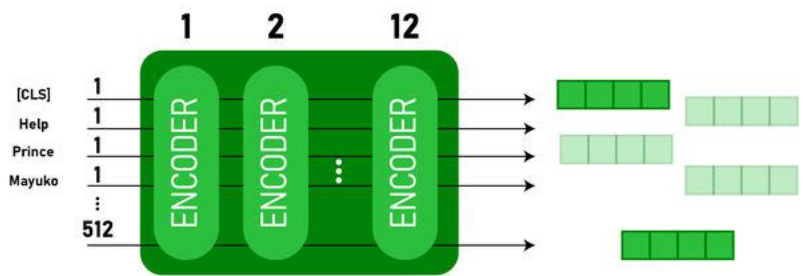
The architecture of BERT is a multilayer bidirectional transformer encoder which is quite similar to the standard Transformer architecture. A transformer architecture is an encoder-decoder network that uses [self-attention](#) on the encoder side and [self-attention](#) on the decoder side.

1. BERT<sub>BASE</sub> has 12 layers in the Encoder stack while BERT<sub>LARGE</sub> has 24 layers in the Encoder stack. Both follow the Transformer architecture described in the original paper (6 encoder layers).
2. BERT architectures (BASE and LARGE) also have larger feedforward networks (768 and 1024 hidden units respectively), and more attention heads (12 and 16 respectively) than the Transformer architecture described in the original paper. It contains 512 hidden units and 8 attention heads.
3. BERT<sub>BASE</sub> contains 110M parameters while BERT<sub>LARGE</sub> has 340M parameters.



BERT BASE and BERT LARGE architecture

This model takes the CLS token as input first, then it is followed by a sequence of words as input. The model also takes a classification token. It then passes the input to the above layers. Each layer applies [self-attention](#) and passes the output through a feedforward network after which it hands off to the next encoder. The model outputs a vector (or BERT BASE). If we want to output a classifier from this model we can take the output corresponding to the classification token.



BERT output as Embeddings

Now, this trained vector can be used to perform a number of tasks such as classification, translation, etc. The paper achieves great results just by using a single layer [Neural Network](#) on the BERT model in the classification task.

## How to use BERT model in NLP?

BERT can be used for various natural language processing (NLP) tasks such as:

### 1. Classification Task

- BERT can be used for classification task like [sentiment analysis](#), the goal is to classify the text into (positive/ negative/ neutral), BERT can be employed by adding a classification layer on the top of the output for the [CLS] token.
- The [CLS] token represents the aggregated information from the entire input sequence. This pooled output can then be used as input for a classification layer to make predictions for the specific task.

### 2. Question Answering

- In question answering tasks, where the model is required to locate and mark the answer within a given passage, BERT can be trained for this purpose.
- BERT is trained for question answering by learning two additional vectors that mark the beginning and end of the answer. During training, the model is provided with questions and corresponding passages, and it learns to predict the start and end positions of the answer within the passage.

### 3. Named Entity Recognition (NER)

- BERT can be utilized for NER, where the goal is to identify and classify entities (e.g., Person, Organization, Location) within a text sequence.
- A BERT-based NER model is trained by taking the output vector of each token from the Transformer encoder and passing it through a classification layer. The layer predicts the named entity label for each token, indicating the type of entity it represents.

## How to Tokenize and Encode Text using BERT?

To tokenize and encode text using BERT, we will be using the 'transformer' library in Python.

**Command to install transformers:**

```
pip install transformers
```

- We will load the pretrained BERT tokenizer with a cased vocabulary using `BertTokenizer.from_pretrained('bert-base-cased')`.
- `tokenizer.encode(text)` tokenizes the input text and converts it into a sequence of token IDs.
- `print("Token IDs:", encoding)` prints the token IDs obtained after encoding.
- `tokenizer.convert_ids_to_tokens(encoding)` converts the token IDs back to their corresponding tokens.
- `print("Tokens:", tokens)` prints the tokens obtained after converting the token IDs.

```
from transformers import BertTokenizer

tokenizer = BertTokenizer.from_pretrained("bert-base-cased")
text = 'ChatGPT is a language model developed by OpenAI, based on the GPT (Generative Transformer) architecture. '

# Tokenize and encode the text
encoding = tokenizer.encode(text)
print("Token IDs:", encoding)

# Convert token IDs back to tokens
tokens = tokenizer.convert_ids_to_tokens(encoding)
print("Tokens:", tokens)
```

#### Output

```
Token IDs: [101, 24705, 1204, 17095, 1942, 1110, 170, 1846, 2235, 1872, 1118, 3353, 1592, 22, 1113, 1103, 15175, 1942, 113, 9066, 15306, 11689, 118, 3972, 13809, 23763, 114, 4220, 119, 102]

Tokens: ['[CLS]', 'Cha', '##t', '##GP', '##T', 'is', 'a', 'language', 'model', 'developed', 'by', 'Open', '##A', 'on', 'the', 'GP', '##T', '(', 'Gene', '##rative', 'Pre', '-', 'trained', 'Trans', '##former', ')', 'architecture', '.', '[SEP]']
```

The `tokenizer.encode` method adds the special [CLS] - classification and [SEP] - separator tokens at the beginning and end of the encoded sequence. In the token IDs section, token id: 101 refers to the start of the sentence and token id: 102 represents the end of the sentence.

## Application of BERT

BERT is used for various applications. Some of these are:

1. **Text Representation:** BERT is used to generate word embeddings or representation for words in a given text.
2. **Named Entity Recognition (NER):** BERT can be fine-tuned for named entity recognition tasks, where the model identifies entities such as names of people, organizations, locations, etc., in a given text.
3. **Text Classification:** BERT is widely used for text classification tasks, including sentiment analysis, topic categorization. It has demonstrated excellent performance in understanding and classifying text data.
4. **Question-Answering Systems:** BERT has been applied to question-answering systems, where the model understands the context of a question and provides relevant answers. This is particularly useful for text comprehension.
5. **Machine Translation:** BERT's contextual embeddings can be leveraged for improving machine translation models. The model captures the nuances of language that are crucial for accurate translation.
6. **Text Summarization:** BERT can be used for abstractive text summarization, where the model generates meaningful summaries of longer texts by understanding the context and semantics.
7. **Conversational AI:** BERT is employed in building conversational AI systems, such as chatbots, virtual dialogue systems. Its ability to grasp context makes it effective for understanding and generating human-like responses.
8. **Semantic Similarity:** BERT embeddings can be used to measure semantic similarity between sentences. This is valuable in tasks like duplicate detection, paraphrase identification, and information retrieval.

## BERT vs GPT


The difference between BERT and GPT are as follows:

	BERT	GPT
Architecture	Bidirectional; predicts masked words based on left, right context.	Unidirectional; predicts preceding co
Pre-training Objectives	BERT is pre-trained using a masked language model objective and next sentence prediction.	GPT is pre-trained using N only.
Context Understanding	Strong at understanding and analyzing text.	Strong in generating contextually rele
Tasks and Use Cases	Commonly used in tasks like text classification, NER, sentiment analysis, and QA	Applied to tasks like text summarizatio
Fine-tuning vs Few-Shot Learning	Fine-tuning with labeled data to adapt its pre-trained representations to the task at hand.	GPT is designed to perform shot learning, where it ca minimal task-spe


**Example to differentiate between output by BERT and GPT:** *"The bank is situated on the \_\_\_\_\_"*  
In the above example, we can observe that BERT considers bidirectional approach enabling a more understanding compared to unidirectional models.

### Related Articles

- [How to Generate Word Embedding using BERT?](#)
- [Sentiment Classification Using BERT](#)
- [Toxic Comment Classification using BERT](#)
- [Fine-tuning BERT for Sentiment Analysis](#)
- [Sentence Similarity using BERT](#)
- [BART Model for Text Auto Completion in NLP](#)

 Comment

More info



Advertise with us

Natural Language Proces

### Similar Reads

#### Natural Language Processing (NLP) Tutorial

Natural Language Processing (NLP) is a branch of Artificial Intelligence (AI) that helps machines to understand and process human la audio form. It is used across a variety of applications from speech recognition to language translation and text summarization.Natural

 5 min read

Introduction to NLP
Libraries for NLP
Text Normalization in NLP
Text Representation and Embedding Techniques
NLP Deep Learning Techniques
NLP Projects and Practice

Corporate & Communications Address:

A-143, 7th Floor, Sovereign Corporate Tower, Sector-136, Noida, Uttar Pradesh (201305)

Registered Address:

K 061, Tower K, Gulshan Vivante Apartment, Sector 137, Noida, Gautam Buddh Nagar, Uttar Pradesh, 201305

f

ig

in

X

yt

Advertise with us

Company

About Us

Legal

Privacy Policy

In Media

Contact Us

Advertise with us

GFG Corporate

Solution

Placement Training

Program

Languages

Python

Java

C++

PHP

GoLang

SQL

R Language

Android Tutorial

Tutorials Archive

DSA

DSA Tutorial

Basic DSA Problems

DSA Roadmap

Top 100 DSA

Interview Problems

DSA Roadmap by Sandeep Jain

All Cheat Sheets

Data Science & ML

Data Science With Python

Data Science For Beginner

Machine Learning

ML Maths

Data Visualisation

Pandas

NumPy

NLP

Deep Learning

Web Technologies

HTML

CSS

JavaScript

TypeScript

ReactJS

NextJS

Bootstrap

Web Design

Python Tutorial

Python

Programming

Examples

Python Projects

Python Tkinter

Python Web

Scraping

OpenCV Tutorial

Python Interview

Question

Django

Computer Science

Operating Systems

Computer Network

Database

Management System

Software

Engineering

Digital Logic Design

Engineering Maths

Software

Development

Software Testing

DevOps

Git

Linux

AWS

Docker

Kubernetes

Azure

GCP

DevOps Roadmap

System Design

High Level Design

Low Level Design

UML Diagrams

Interview Guide

Design Patterns

OOAD

System Design

Bootcamp

Interview Questions

Interview Preparation

Competitive

Programming

Top DS or Algo for CP

Company-Wise

Recruitment Process

Company-Wise

Preparation

Aptitude Preparation

Puzzles

School Subjects

Mathematics

Physics

Chemistry

Biology

Social Science

English Grammar

Commerce

GeeksforGeeks Videos

DSA

Python

Java

C++

Web Development

Data Science

CS Subjects

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#)

Got It !

https://www.geeksforgeeks.org/nlp/explanation-of-bert-model-nlp/

Page 7 of 7