BERT Model – NLP

Last Updated : 10 Dec, 2024

BERT (**Bidirectional Encoder Representations from Transformers)** stands as an open-source **machine learning framework** designed for the **natural language processing (NLP)**. Originating in 2018, this framework was crafted by researchers from Google AI Language.

*The article aims to explore the **architecture, working and applications of BERT**.*

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 original **Transformer architecture**, there are both encoder and decoder modules. The decision to use an encoder-only architecture in BERT suggests a primary emphasis on understanding input sequences rather than generating output sequences.

Bidirectional Approach of BERT

Traditional language models process text sequentially, either from left to right or right to left. This method limits the model's awareness to the immediate context preceding the target word. BERT uses a bi-directional approach considering both the left and right context of words in a sentence, instead of analyzing the text sequentially, BERT looks at all the words in a sentence simultaneously.

Example: "The bank is situated on the _______ of the river."

In a unidirectional model, the understanding of the blank would heavily depend on the preceding words, and the model might struggle to discern whether "bank" refers to a financial institution or the side of the river.

BERT, being bidirectional, simultaneously considers both the left ("The bank is situated on the") and right context ("of the river"), enabling a more nuanced understanding. It comprehends that the missing word is likely related to the geographical location of the bank, demonstrating the contextual richness that the bidirectional approach brings.

Pre-training and Fine-tuning BERT Model

The BERT model undergoes a two-step process:

1. Pre-training on Large amounts of unlabeled text to learn contextual embeddings.
2. Fine-tuning on labeled data for specific NLP tasks.

Pre-Training on Large Data

- BERT is pre-trained on large amount of unlabeled text data. The model learns contextual embeddings, which are the 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.

Fine-Tuning on Labeled Data

- After the pre-training phase, the BERT model, armed with its contextual embeddings, is then fine-tuned for specific natural language processing (NLP) tasks. This step tailors the model to more

targeted applications by adapting its general 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 could include sentiment analysis, question-answering, named entity recognition, or any other NLP application. The model's parameters are 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.

How BERT work?

BERT is designed to generate a language model so, only the encoder mechanism is used. Sequence of tokens are fed to 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 trained to predict the original values of these masked words based on the context provided by the surrounding words.

In simple terms,

1. **Masking words:** Before BERT learns from sentences, it hides some words (about 15%) and replaces them with a special symbol, like

[MASK].

2. **Guessing Hidden Words:** BERT's job is to figure out what these hidden words are by looking at the words around them. It's like a game of guessing where some words are missing, and BERT tries to fill in the blanks.
3. **How BERT learns:**
   - BERT adds a special layer on top of its learning system to make these guesses. It then checks how close its guesses are to the actual hidden words.
   - It does this by converting its guesses into probabilities, saying, "I think this word is X, and I'm this much sure about it."
4. **Special Attention to Hidden Words**
   - BERT's main focus during training is on getting these hidden words right. It cares less about predicting the words that are not hidden.
   - This is because the real challenge is figuring out the missing parts, and this strategy helps BERT become really good at understanding the meaning and context of words.

In technical terms,

1. BERT adds a classification layer on top of the output from the encoder. This layer is crucial for predicting the masked words.
2. The output vectors from the classification layer are multiplied by the embedding matrix, transforming them into the vocabulary dimension. This step helps align the predicted representations with the vocabulary space.
3. The probability of each word in the vocabulary is calculated using the SoftMax activation function. This step generates 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 model is penalized for the deviation between its predictions and the actual values of the masked words.
5. The model converges slower than directional models. This is because, during training, BERT is only concerned with predicting the masked values, ignoring the prediction of the non-masked words.

The increased context awareness achieved 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 of the [CLS] token into a 2×1 shaped vector using a classification layer, and then calculating the probability of whether the second sentence follows the first using SoftMax.

1. In the training process, BERT learns to understand the relationship between pairs of sentences, predicting if the second 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 document, and the other 50% have a randomly chosen sentence.
3. To help the model distinguish between connected and disconnected sentence pairs. The input is processed before entering the model:
    - A [CLS] token is inserted at the beginning of the first sentence, and a [SEP] token is added at the end of each sentence.
    - A sentence embedding indicating Sentence A or Sentence B is added to each token.
    - A positional embedding indicates the position of each token in the sequence.

4. BERT predicts if the second sentence is connected to the first. This is done by transforming the output of the [CLS] token into a 2×1 shaped vector using a classification layer, and then calculating the probability of whether the second sentence follows the first using SoftMax.

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

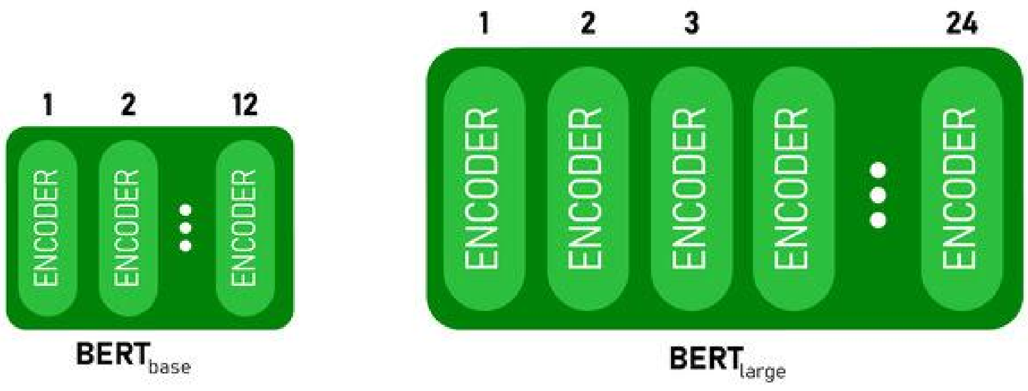
Why to train Masked LM and Next Sentence Prediction together?

Masked LM helps BERT to understand the context within a sentence and Next Sentence Prediction helps BERT grasp the connection or relationship between pairs of sentences. Hence, training both the strategies together ensures that BERT learns a broad and comprehensive understanding of language, capturing both details within sentences and the flow between sentences.

BERT Architecture

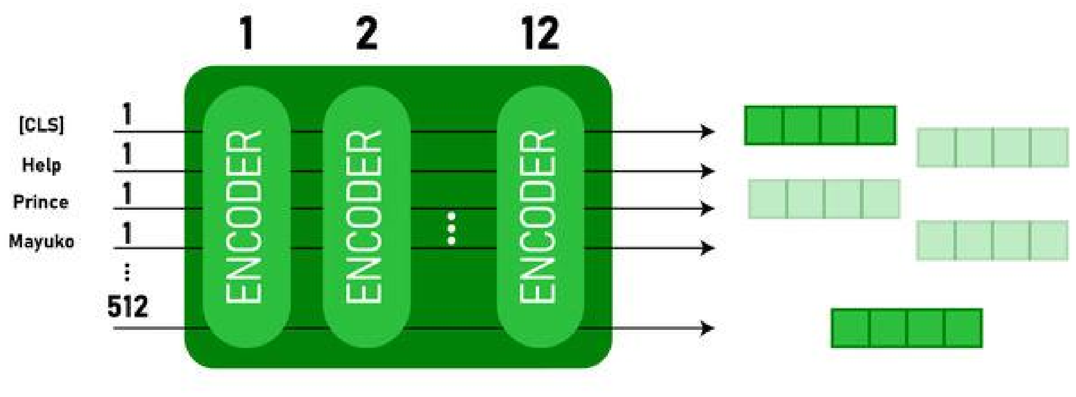
The architecture of BERT is a multilayer bidirectional transformer encoder which is quite similar to the transformer model. A transformer architecture is an encoder-decoder network that uses self-attention on the encoder side and attention on the decoder side.

1. $\text{BERT}_{\text{BASE}}$ has 1*2 layers in the Encoder stack* while $\text{BERT}_{\text{LARGE}}$ has *24 layers in the Encoder stack*. These are more than 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 suggested in the original paper. It contains *512 hidden units and 8 attention heads*.
3. $\text{BERT}_{\text{BASE}}$ contains 110M parameters while $\text{BERT}_{\text{LARGE}}$ 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. Here CLS is a classification token. It then passes the input to the above layers. Each layer applies self-attention and passes the result through a feedforward network after then it hands off to the next encoder. The model outputs a vector of hidden size (*768* for BERT BASE). If we want to output a classifier from this model we can take the output corresponding to the CLS token.



BERT output as Embeddings

Now, this trained vector can be used to perform a number of tasks such as classification, translation, etc. For Example, 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 different categories (positive/ negative/ neutral), BERT can be employed by adding a classification layer on the top of the Transformer output for the [CLS] token.
- The [CLS] token represents the aggregated information from the entire input sequence. This pooled representation 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 text sequence, 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, Date) in a text sequence.
- A BERT-based NER model is trained by taking the output vector of each token form the Transformer and feeding it into 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 tokenize 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

# Load pre-trained BERT tokenizer
tokenizer = BertTokenizer.from_pretrained("bert-base-cased")

# Input text
text = 'ChatGPT is a language model developed by OpenAI, based on the GPT
(Generative Pre-trained Transformer) architecture. '

# Tokenize and encode the text
encoding = tokenizer.encode(text)

# Print the token IDs
print("Token IDs:", encoding)

# Convert token IDs back to tokens
tokens = tokenizer.convert_ids_to_tokens(encoding)

# Print the corresponding tokens
print("Tokens:", tokens)
```

Output:

```
Token IDs: [101, 24705, 1204, 17095, 1942, 1110, 170, 1846,
2235, 1872, 1118, 3353, 1592, 2240, 117, 1359, 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', '##I', ',', 'based', '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:

1. **Text Representation:** BERT is used to generate word embeddings or representation for words in a sentence.
2. **Named Entity Recognition (NER)**: BERT can be fine-tuned for named entity recognition tasks, where the goal is to identify 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, spam detection, and topic categorization. It has demonstrated excellent performance in understanding and classifying the context of textual data.
4. **Question-Answering Systems:** BERT has been applied to question-answering systems, where the model is trained to understand the context of a question and provide relevant answers. This is particularly useful for tasks like reading comprehension.
5. **Machine Translation:** BERT's contextual embeddings can be leveraged for improving machine translation systems. 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 concise and 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 assistants, and dialogue systems. Its ability to grasp context makes it effective for understanding and generating natural language responses.
8. **Semantic Similarity:** BERT embeddings can be used to measure semantic similarity between sentences or documents. This is valuable in tasks like duplicate detection, paraphrase identification, and information retrieval.

BERT vs GPT

The difference between BERTand GPT are as follows:

	BERT	GPT
Architecture	BERT is designed for bidirectional representation learning. It uses a masked language model objective, where it predicts missing words in a sentence based on both left and right context.	GPT, on the other hand, is designed for generative language modeling. It predicts the next word in a sentence given the preceding context, utilizing a unidirectional autoregressive approach.
Pre-training Objectives	BERT is pre-trained using a masked language model objective and next sentence prediction. It focuses on capturing bidirectional context and understanding relationships between words in a sentence.	GPT is pre-trained to predict the next word in a sentence, which encourages the model to learn a coherent representation of language and generate contextually relevant sequences.
Context Understanding	BERT is effective for tasks that require a deep understanding of context and relationships within a sentence, such as text classification, named entity recognition, and question-answering.	GPT is strong in generating coherent and contextually relevant text. It is often used in creative tasks, dialogue systems, and tasks requiring the generation of natural language sequences.
Task types and Use Cases	Commonly used in tasks like text classification, named	Applied to tasks such as text generation, dialogue systems,

	BERT	GPT
	entity recognition, sentiment analysis, and question-answering.	summarization, and creative writing.
Fine-tuning vs Few-Shot Learning	BERT is often fine-tuned on specific downstream tasks with labeled data to adapt its pre-trained representations to the task at hand.	GPT is designed to perform few-shot learning, where it can generalize to new tasks with minimal task-specific training data.

Also Check:

- Sentiment Classification Using BERT
- How to Generate Word Embedding using BERT?
- BART Model for Text Auto Completion in NLP
- Toxic Comment Classification using BERT
- Next Sentence Prediction using BERT

Frequently Asked Questions (FAQs) on BERT

What is BERT used for?

BERT is used for performing NLP tasks like text representation, named entity recognition, text classification, Q&A systems, machine translation, text summarization, and more.

What are the advantages of the BERT model?

The BERT language model stands out due to its extensive pre-training in multiple languages, offering a broad linguistic coverage compared to other models. This makes BERT particularly advantageous for non-English-based projects, as it provides *robust contextual representations and semantic understanding across a diverse range of languages, enhancing its versatility in multilingual applications.*

How does BERT work for sentiment analysis?

BERT excels in sentiment analysis by leveraging its bidirectional representation learning to capture contextual nuances, semantic meanings, and syntactic structures within a given text. This enables BERT to understand the sentiment expressed in a sentence by considering the relationships between words, resulting in highly effective sentiment analysis outcomes.

Is Google based on BERT?

***BERT** and **RankBrain** are components of Google's search algorithm to process queries and web pages content to gain better understanding to improve the search results.*

Next Article

HuBERT Model

Similar Reads

Understanding BERT - NLP

BERT stands for Bidirectional Representation for Transformers. It was proposed by researchers at Google Research in 2018. Although the mai...

5 min read

Fine-tuning BERT model for Sentiment Analysis

Google created a transformer-based machine learning approach for natural language processing pre-training called Bidirectional Encoder...

6 min read

Next Sentence Prediction using BERT

Pre-requisite: BERT-GFG BERT stands for Bidirectional Representation for Transformers. It was proposed by researchers at Google Research in...

7 min read

ALBERT - A Light BERT for Supervised Learning

The BERT was proposed by researchers at Google AI in 2018. BERT has created something like a transformation in NLP similar to that caused by...

4 min read

What are some key strengths of BERT over ELMO/ULMFiT?

Answer: BERT excels over ELMO and ULMFiT due to its bidirectional context understanding, capturing complex relationships in language by...

2 min read

What is the use of [SEP] in paper BERT?

Answer: [SEP] token is used in BERT to separate input segments or sentences in the input sequence.In BERT (Bidirectional Encoder...

2 min read

What GPU Size Do I Need to Fine Tune BERT Base Cased?

Answer : At least 16GB of VRAM is recommended for fine-tuning BERT Base Cased.Fine-tuning BERT Base Cased requires careful consideration...

1 min read

Can BERT do the next-word-predict task?

Answer: Yes, BERT can be adapted for next-word prediction tasks through fine-tuning, despite being primarily designed for understanding the...

2 min read

Sentence Similarity using BERT Transformer

Conventional techniques for assessing sentence similarity frequently struggle to grasp the intricate nuances and semantic connections found...

5 min read

Toxic Comment Classification using BERT

Social media users frequently encounter abuse, harassment, and insults from other users on a majority of online communication platforms like...

15+ min read

Biology

Social Science

English Grammar

Management

HR Management

Finance

Income Tax

Databases

SQL

MYSQL

PostgreSQL

PL/SQL

MongoDB

Preparation Corner

Company-Wise Recruitment Process

Resume Templates

Aptitude Preparation

Puzzles

Company-Wise Preparation

Companies

Colleges

Competitive Exams

JEE Advanced

UGC NET

UPSC

SSC CGL

SBI PO

SBI Clerk

IBPS PO

IBPS Clerk

More Tutorials

Software Development

Software Testing

Product Management

Project Management

Linux

Excel

All Cheat Sheets

Recent Articles

Free Online Tools

Typing Test

Image Editor

Code Formatters

Code Converters

Currency Converter

Random Number Generator

Random Password Generator

Write & Earn

Write an Article

Improve an Article

Pick Topics to Write

Share your Experiences

Internships

DSA/Placements

DSA - Self Paced Course

DSA in JavaScript - Self Paced Course

DSA in Python - Self Paced

C Programming Course Online - Learn C with Data Structures

Complete Interview Preparation

Master Competitive Programming

Core CS Subject for Interview Preparation

Mastering System Design: LLD to HLD

Tech Interview 101 - From DSA to System Design [LIVE]

DSA to Development [HYBRID]

Placement Preparation Crash Course [LIVE]

Development/Testing

JavaScript Full Course

React JS Course

React Native Course

Django Web Development Course

Complete Bootstrap Course

Full Stack Development - [LIVE]

JAVA Backend Development - [LIVE]

Complete Software Testing Course [LIVE]

Android Mastery with Kotlin [LIVE]

Machine Learning/Data Science

Complete Machine Learning & Data Science Program - [LIVE]

Programming Languages

C Programming with Data Structures

C++ Programming Course