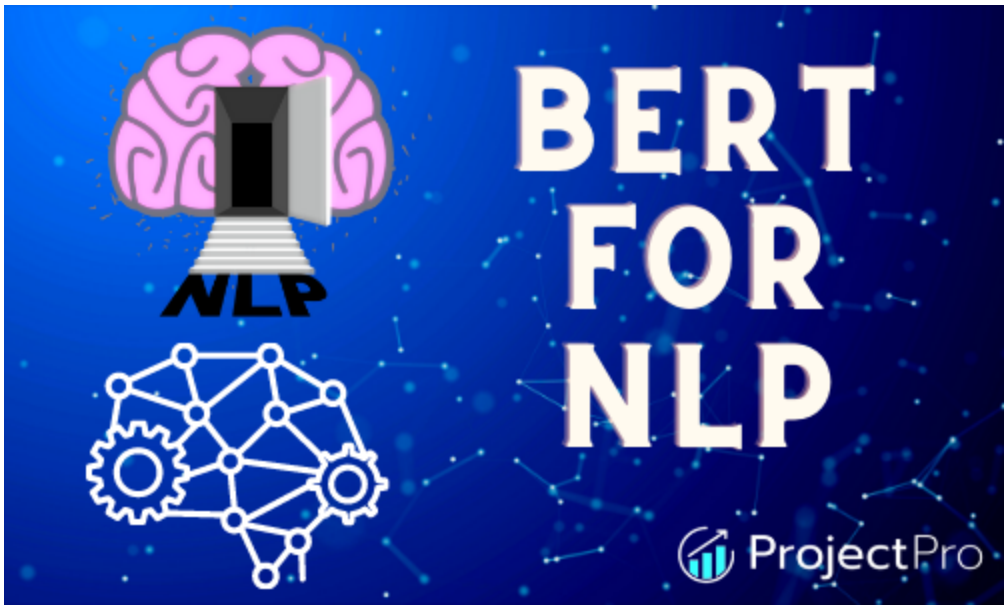
BERT NLP Model Explained for Complete Beginners

projectpro.io/article/bert-nlp-model-explained/558



BERT NLP Tutorial - A-Z Guide on using state-of-the-art BERT NLP Model for Complete Beginners.
Last Updated: 24 Apr 2023

Written by:

ProjectPro

ProjectPro is the only online platform designed to help professionals gain practical, hands-on experience in big data, data engineering, data science, and machine learning related technologies. Meet The Author

From sending letters in physical mailboxes to direct messages through your favorite social media application, the explosion of text has been astronomical. The innovation and development of mobile devices and computers helped push this increase, and this geometric

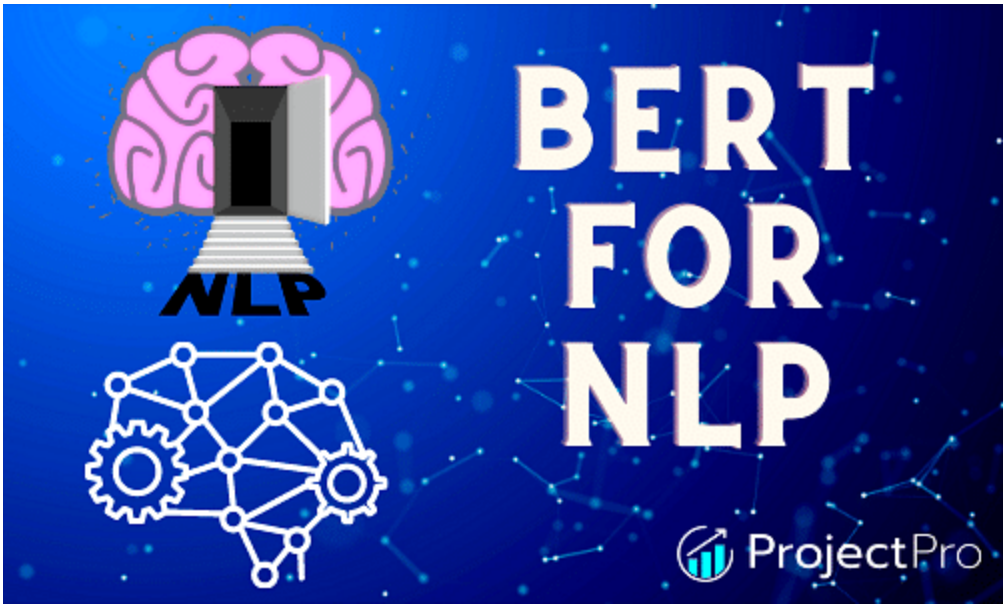
growth has called for innovative ways to understand and process text. With machine learning taking some significant leaps in the early 2010s, model creation and prediction have been refined to mirror human understanding of linguistic expressions.

Multi-Class Text Classification with Deep Learning using BERT

Downloadable solution code | Explanatory videos | Tech Support

There has been a significant leap in Natural Language Processing (NLP) in recent years. From BERT ELMO NLP (Embedding from Language Models) to GPT (Generative Pre-trained Transformer), researchers have tried and succeeded – to an extent– in making models understand the contextual meaning of words. The creation of BERT significantly helped NLP scientists to take significant bounds in getting a solution. It is now straightforward to search for new words and translate between languages to a more suitable one. Bidirectional Encoder Representations from Transformers(BERT) is one of the most used models in natural language processing in the last decade, all thanks to BERT NLP, the state of the art language model.

Introduction to BERT NLP Model



BERT NLP model is a group of Transformers encoders stacked on each other. – BERT is a precise, huge transformer-masked language model in more technical terms.

Let's break that statement down:

Models are the output of an algorithm run on data, including the procedures used to make predictions on data.

A Language model is a numerical model of the probability of words, sentences, or phrases. An example language model can return is as follows -

The probability of the sentence *"Project Pro blog is informative"* as output is greater than (>) the Probability of returning *"Informative is Project Pro"* as output based on a group of text it has learned from.

Probability simply means which is most likely to happen. For example, in this case, the sentences that have a better chance of being outputted by the machine output after learning from a group of text.

The quicker the language model learns from the training set to give more accurate predictions, the better the model is said to be.

Start your journey as a Data Scientist today with solved end-to-end Data Science Projects

Masked Language Model

One of the significant contributions of the BERT NLP contribution paper is masking. Masking involves removing parts of sentences and having the model fill in the gaps. Masked Language models are becoming very useful because they are one of the ways contextual embedding is done. One of the drawbacks of traditional word embedding used in earlier language models is the lack of contextual meaning.

For example, let's take the following words:

The *speed boat was packed close to the river bank.*

Eddy went to the bank to deposit some money and store some jewels.

The word bank means very different things in the above sentences. For the two sentences, the numerical representation (also known as an embedding) in the traditional language model of the word bank would be the same. The masked language model helps solve this problem (amongst other things). Later in this BERT NLP tutorial, a section will highlight how such a model solves this problem.

Transformers

BERT was created on the Transformer architecture, a family of Neural Network architectures. The general idea of Transformer architecture is based on self-attention, and the paper in which it was proposed is Attention is All You Need. Self-attention is learning to weigh the relationship between each item or word in an input sequence to other words in the input sequence.

Consider the sentence:

The animal took too long to cross the road because it was too slow.

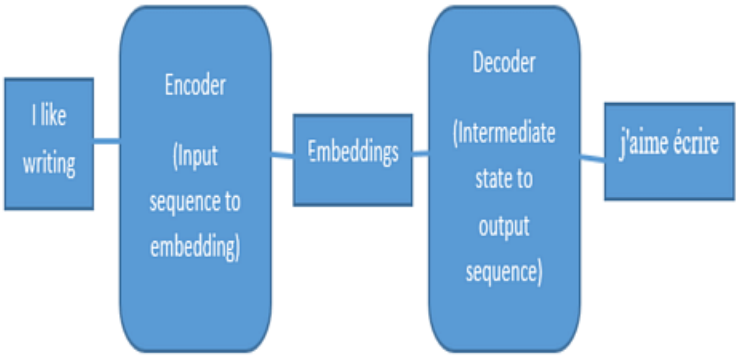
If this sentence is passed through an algorithm, what word does it associate “it” with? Is “it” the animal or the road? While this might be easy for a person to understand, it is difficult for the algorithm to decipher at a glance. Self-attention ensures that as the model goes through each of the words sequentially in the training data, it looks at the input text for hints that can assist in encoding the word better.

BERT NLP Model Architecture Explained in Detail

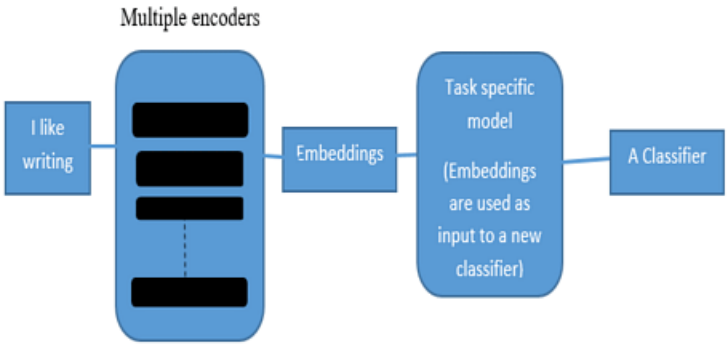
The original Transformer architecture uses the sequence-to-sequence model with an encoder and a decoder. The encoder turns the input into embedding, and the decoder turns the embedding into string output.

Example of the Original Transformer Architecture

Below is an architecture of a language interpreting transformer architecture.



The BERT architecture has a different structure. Depending on the use case, it stacks encoders on each other (12 base or 24 large encoders).



The output embeddings will look like this:

[CLS] Her dog is cute. He can be annoying sometimes [SEP]

[CLS] is a token used to represent the classification of specific inputs. You can use this to train models in supervised learning because you can know the value. An example is using 1 to represent a Home win and 0 to represent an away win in sports analytics. In the tokens, CLS is represented by 101.

[SEP] is also a unique token used at the end of input. This is used for separating sentences. In the tokens, SEP is represented by 102. A common way the BERT architecture is also used is to continue the picture above. Here, the feedback from the classifier goes back into the transformer with updated weights and embeddings.

How to use BERT NLP model?

- The BERT Model can be used to change the input to numerical representation (changing text into word embeddings.)
- BERT can be used as an all-purpose pre-trained model fine-tuned for specific tasks.

All You Need to Know About How BERT Works

BERT NLP Model, at the core, was trained on 2500M words in Wikipedia and 800M from books. BERT was trained on two modeling methods:

1. MASKED LANGUAGE MODEL (MLM)
2. NEXT SENTENCE PREDICTION (NSP)

These models are also used in practice to fine-tune text when doing natural language processing with BERT.

With these Data Science Projects in Python, your career is bound to reach new heights. Start working on them today!

Masked Language Model (MLM)

There was a small introduction to the masked language model in the earlier section. In MLM, a certain percentage of the corpus is hidden or *masked* and trained. BERT is given a group of words or sentences, and the contextual weights are maximized to output the sentence on the other side. An incomplete sentence is inputted into BERT, and an output is received in the easiest terms.

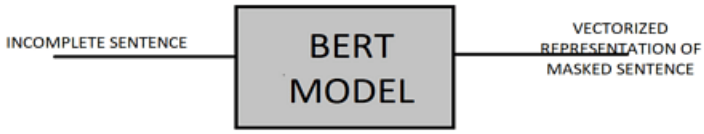
Let us consider the sample sentence below:

In a year, there are [MASK] months in which [MASK] is the first.

The [MASKS] keywords in the sentence above represent masked words. It is a form of *Fill in the gaps* (as used in schools for setting questions used in tests/questionnaires.)

You could guess what goes into the masks gaps in the sentence above to make it a complete one. The first part of the sentence *In a year* helps us unlock the sentence, and it becomes very simple to understand from there. The word month acts like the subject for the last part of the sentence and in which is used to combine the different parts of the sentence. This was easy to break down because we understand the contextual weights of these words, and most importantly, we are familiar with the Linguistic expressions of the English Language.

For BERT to complete the sentence, it must read a lot and learn the linguistic patterns of the language very well. BERT might not know what month, year are --- but it is inevitable that given the large corpus of words to learn the patterns and context of the words, the answers are likely to be twelve and January.



In practice, 15% of words in a sentence or text corpus are masked to optimize the output result. If there are 20 words in a sentence, 2/3 can be masked. A masking function at the end of the input (from the picture above) can assist in creating the masked sentence. Recall that the basis of this model is to help improve the contextual understanding between words in a sentence.

Next Sentence Prediction (NSP)

While MLM is training the relationship between words, next sentence prediction (NSP) tries to create a long-term relationship between sentences. The original BERT NLP paper by Google mentioned that the model performed poorly across each measured metric when NSP was not used.

What is NSP?

NSP involves giving BERT two sentences, sentence 1 and sentence 2. Then, BERT is asked the question: "HEY BERT, DOES SENTENCE 1 COME AFTER SENTENCE 2?" --- and BERT replies with isNextSentence or NotNextSentence.

Consider the following three sentences below:

1. Tony drove home after playing football in front of his friend's house for three hours.
2. In the milky way galaxy, there are eight planets, and Earth is neither the smallest nor the largest.
3. Once home, Tony ate the remaining food he left in the Fridge and fell asleep on the floor.

Which of the sentences would you say followed the other logically? 2 after 1? Probably not. These are the questions that BERT is supposed to answer.

Sentence 3 comes after 1 because of the contextual follow-up in both sentences. Secondly, an easy takeaway is that both sentences contain "Tony". Training BERT with NSP after MLM makes it understandable to a reasonable extent the linguistic connotations and properties of

the language to a good extent.

Applications of BERT NLP

- Matching and retrieving documents based on a text or group of words given.
- Text Summarization: Summarization of large corpus/documents into smaller chunks to be consumed.
- Highlight paragraphs with crucial entry points when a question is asked.
- Google Search: Previously, word matching was used when searching words through the internet. If a person searched “Lagos to Kenya flights,” there was a high chance of showing sites that included “Kenya to Lagos” flights in the top results. BERT instead uses contextualized matching instead of only word matching.
- Sentiment Analysis
- Question Answering
- Language Translation

Advantages Of Using BERT NLP Model Over Other Models

- BERT works well for task-specific models. The state of the art model, BERT, has been trained on a large corpus, making it easier for smaller, more defined nlp tasks.
- Metrics can be fine-tuned and be used immediately.
- The accuracy of the model is outstanding because it is frequently updated. You can achieve this with successful fine-tuning training.
- The BERT model is available and pre-trained in more than 100 languages. This can be useful for projects that are not English-based.

Disadvantages of BERT

Most of the drawbacks of BERT can be linked to its size. While training the data on a large corpus significantly helps how the computer predicts and learns, there is also another side to it. They include:

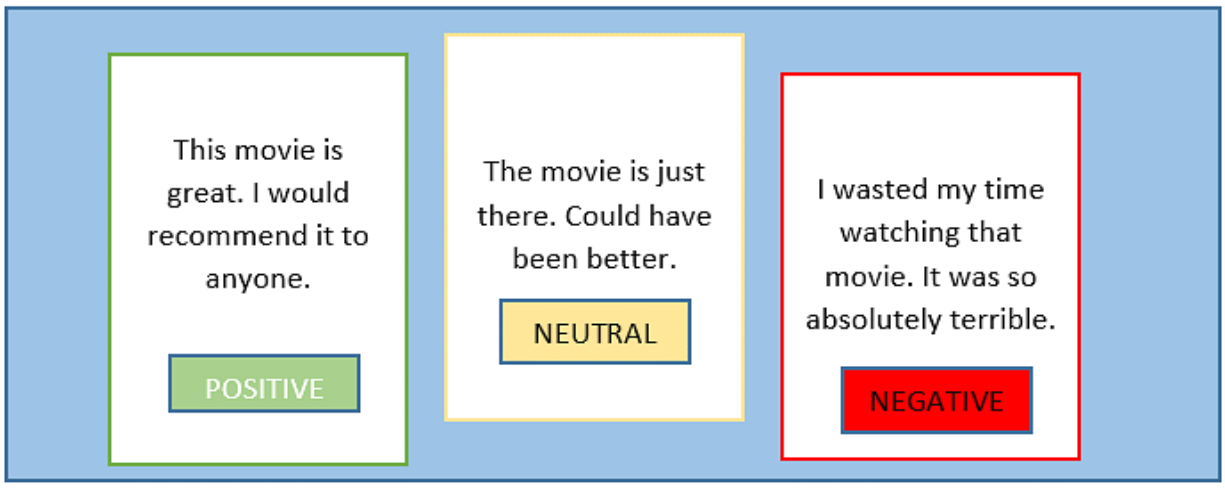
- The model is large because of the training structure and corpus.
- It is slow to train because it is big and there are a lot of weights to update.
- It is expensive. It requires more computation because of its size, which comes at a cost.
- It is designed to be input into other systems (not a standalone program), and because of that, it has to be fine-tuned for downstream tasks, which can be fussy.

Common Mistakes to Avoid When Working with BERT NLP Model

These are some common mistakes NLP engineers or data scientists make when using BERT -

- Type of Tokenizer Used: The WordPiece tokenizer must be used when using BERT. You have to use the same kind of tokenizer originally used to train BERT to train your model.
- Training BERT Model from Scratch. Use pre-trained models instead of training new models when using BERT. This is very expensive and it is not advisable to do so.
- Task Specific Problems. For some tasks, when fine-tuning, the results from the runs will not converge (also known as degeneration). This usually depends on the tasks, and it is advised to be aggressive with early stopping when fine-tuning it yourself.

BERT NLP Tutorial: Use of BERT for Sentiment Analysis PyTorch



Sentiment Analysis is a technique in natural language processing used to determine the sentiment of a corpus of words. It is used to determine a group of words' degree of positivity or negativity. Sentiment Analysis finds huge applications in analyzing reviews, ratings, or feedback.

Businesses and organizations use sentiment analysis to analyze their products or services on the go. This feedback helps the organization know areas where to improve and enlightens future audiences and users of products of what to expect. This also is a good indication if an interested individual buys the product or service. The innovation and improvement of language models in recent years have ensured that the management of companies usually asks for this kind of analysis. Almost all profitable businesses have a feedback platform for their customers.

We will explore a short example of BERT for sentiment analysis to understand how BERT is used in practice.

Setting up the Environment

A good practice is setting up a new Virtual environment for projects, especially when installing new dependencies on your computer. This is to ensure that your core libraries are not affected if an installation issue needs a revert. Jupyter notebook will be used for this project. For anaconda, you can create a virtual environment with the following command -

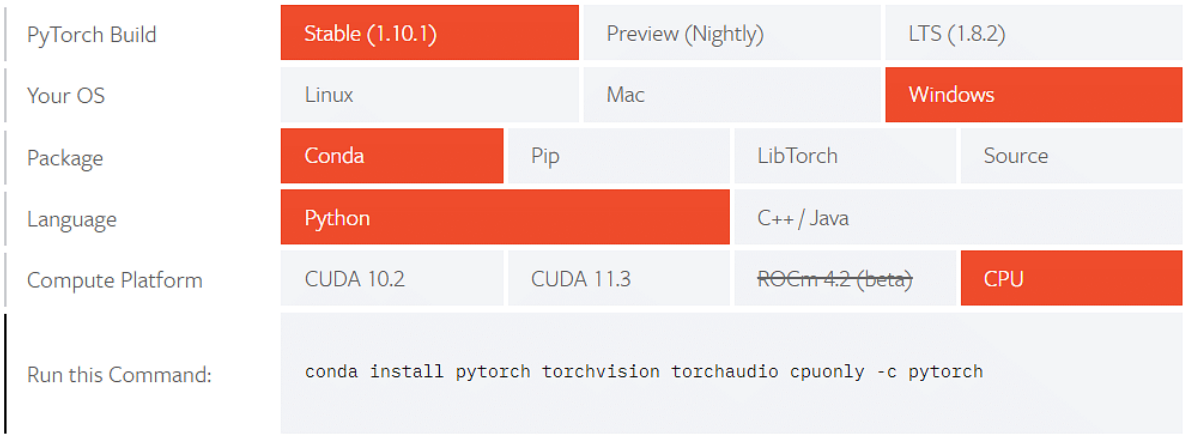
conda create --name nlp_bert

Here, for easy recall, nlp_bert is used as the name. You can replace it with any name you are more comfortable with.

Installing Required Dependencies

PyTorch

On the PyTorch website, there are some selections to pick depending on the operating system and installation method that will be used:



The CUDA option is for computers with a graphics card (for example, NVidia graphics card), so CPU remains the safest option. Most data scientists use anaconda so the conda option will be selected.

Transformers

For this NLP project example, we will use the Huggingface pre-trained BERT model will be used. The code for installing the dependency is:

conda install -c huggingface transformers

Recall that one of the points above (under the standard errors section) is creating a BERT model from scratch. Huggingface has developed an open-source BERT model that we can use for this example.

Importing Required Dependencies

import torch

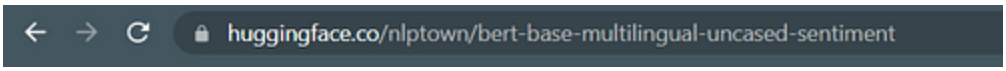
from transformers import AutoTokenizer, AutoModelForSequenceClassification

Instantiating the Model

You will create the Tokenzier in this code section and the model loaded from Huggingface. The first time the model is loaded, it will be downloaded into your computer and might take some time.

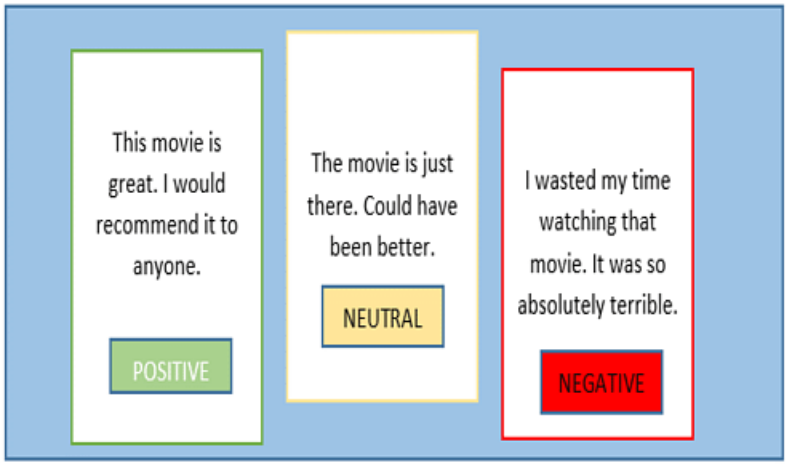
```
In [3]: tokenizer = AutoTokenizer.from_pretrained("nlptown/bert-base-multilingual-uncased-sentiment")

Downloading: 100% 39.0/39.0 [00:00<00:00, 1.41kB/s]

Downloading: 100% 953/953 [00:00<00:00, 18.0kB/s]

Downloading: 100% 851k/851k [00:04<00:00, 286kB/s]

Downloading: 100% 112/112 [00:00<00:00, 2.87kB/s]

In [4]: model = AutoModelForSequenceClassification.from_pretrained("nlptown/bert-base-multilingual-uncased-sentiment")

Downloading: 100% 638M/638M [29:59<00:00, 324kB/s]
```

The links in the brackets are the ending section to the website URL, effectively acting as an API (Application Programming Interface).



Calculating the Sentiment

We will calculate the Sentiment for three different sentences to explain how it works. It ranges from 1 to 5 where 1 is negative and 5 is positive.



Let's take a look at the sentiments of the sentences in the picture that is shown above:

1. The movie is great. I would recommend it to anyone.
2. The movie is just there. Could have been better.
3. I wasted my time watching that movie. It was so absolutely terrible

For the first sentence, the token is created by tokenizing the input text. Return sensors is set to "pt" which means PyTorch. Our tokens are in a list of lists, so indexing returns the list with the sentence tokens.

```
tokens = tokenizer.encode("The movie is great. I would recommend it to anyone.", return_tensors="pt")
```

```
tokens
```

```
]: tensor([[  101, 10103, 13113, 10127, 11838,   119,   151, 11008, 44909, 55667,
         10163, 10197, 10114, 42946,   119,   102]])
```

```
tokenizer.decode(tokens[0])
```

```
]: '[CLS] the movie is great. i would recommend it to anyone. [SEP]'
```

The tokenizer.decode() function above is not necessary for sentiment analysis, and it is to highlight the special tokens that were explained earlier in the first section. [CLS] and [SEP] are special characters representing classification and sentence separator, respectively.

```
result = model(tokens)
```

```
result
```

```
]: SequenceClassifierOutput(loss=None, logits=tensor([[-2.3169, -2.2531, -0.4282,  1.6122,  2.6564]],
       grad_fn=<AddmmBackward0>), hidden_states=None, attentions=None)
```

```
result.logits
```

```
]: tensor([[-2.3169, -2.2531, -0.4282,  1.6122,  2.6564]],
       grad_fn=<AddmmBackward0>)
```

The logits with the highest value are where the sentiment analytical value will be from. Logits is the final layer in a neural network that returns the raw values for the prediction. Logit is the tensor which the argmax function is used to return the predicted class.

```
torch.argmax(result.logits)
```

```
: tensor(4)
```

```
int(torch.argmax(result.logits))+1
```

```
: 5
```

The rating for the sentence “The movie is great. I would recommend it to anyone.” is a semantic rating of 5, meaning very positive.

At a glance, the semantic score can be easily calculated because the model being used has been pre-trained using the methods explained in earlier sections. Depending on the project and use case, you can fine-tune the parameters to match requirements.

To calculate the sentiments for the three sample reviews. You will create a function as shown below-

```
In [53]: reviews = ["The movie is great. I would recommend it to anyone.", \
                    "The movie is just there. Could have been better.", \
                     "I wasted my time watching that movie. It was absolutely terrible."]

In [54]: def sentiment_calculator(review):
             tokens = tokenizer.encode(review, return_tensors="pt")
             result = model(tokens)
             print(int(torch.argmax(result.logits))+1)

In [55]: for review in reviews:
             print(review)
             print("The sentiment score is:")
             sentiment_calculator(review)
             print("---"*15)

The movie is great. I would recommend it to anyone.
The sentiment score is:
5
---------------------------------------------
The movie is just there. Could have been better.
The sentiment score is:
3
---------------------------------------------
I wasted my time watching that movie. It was absolutely terrible.
The sentiment score is:
1
---------------------------------------------
```

For each of the sentences, the semantic score matches our sentences logically. “The movie is great. I would recommend it” is a very positive review and as such has the highest positive score, which is 5.

You can take the next step by web scraping numerous reviews or feedbacks from a product platform and generating a semantic score of the overall feedback of the product or service.

BERT NLP Project Ideas

1. Fake News Detection

In this digital era, the thirst of curiosity to know about almost anything is quickly fulfilled by a few touches and swipes on a screen. But with this boon, there is a bane of spreading of incorrect information. Use the web scraping method to source data and train the BERT

algorithm to deduce whether or not the input text represents factually correct information.

2. Multi-Class Text Classification

In this project, you will learn how to formulate a business problem into a data science problem. You will learn about the architecture of the BERT model and preparing data for parsing it through the model. This project will guide you through the solution of a multi-class text classification problem by training the pre-trained BERT model with the help of CUDA or CPU.

Source Code: NLP Project for Multi-Class Text Classification using BERT Model

3. Human Emotion Identification

In this project, you will learn about transformer-based models, RoBERTa and XLNet. The goal is to analyze the textual data and label the data with one of the six emotions- anger, fear, joy, love, sadness, and surprise. You will learn how to implement NLP methods like the Bag of Words model. TF-IDF, Word2vec embeddings, etc. The project solution will give you an in-depth understanding of the BERT architecture.

Source Code: Text Classification with Transformers-RoBERTa and XLNet Model

4. Sentiment Analysis

Analyzing customers' feedback is key to the growth of any business. In this BERT NLP Github project, you will learn how to analyze the sentiments of customers by passing their feedback to the BERT model. This project will guide you about NLP methods such as Tokenization and encoding textual data. You will also learn how to analyze model performance using metrics like F1-Score, accuracy, etc.

GitHub Repository: Sentiment Analysis with Deep Learning using BERT by Dhruv Aggarwal

BART Vs. BERT NLP

When reading about the BERT algorithm, you are likely to come across the BART model, and you must understand the difference between the two models so that you do not get confused between them.

BERT	BART

BERT: Bidirectional Encoder Representations from Transformers	BART: Bidirectional Auto-Regressive Transformers
BERT architecture consists of a bidirectional encoder.	BART architecture consists of a combination of a bidirectional encoder and an autoregressive decoder.
It is suited for text classification tasks.	It is suited for text-generation tasks.

Get FREE Access to Data Analytics Example Codes for Data Cleaning, Data Munging, and Data Visualization

ELMo Vs. BERT NLP

Another algorithm that is widely compared to BERT is ELMO. Let us try to understand how these two are different from each other.

ELMo	BERT
ELMo: Embeddings from language model	BERT: Bidirectional Encoder Representations from Transformers
It uses right-to-left and left-to-right LSTMs.	It is deeply bidirectional because of the underlying masked language modeling technique on which it is built.
The input to this model is characters and not words.	It tokenizes words into sub-words, which serve as input to the model.
The core of this model is based on Long Short Term Memory Networks (LSTMs).	The core of the BERT model is based on the Transformers model.

BERT NLP -Learning Takeaways

Contextual understanding of sentences has created significant bounds in natural language processing. The continuous innovation around this subject will get even more precise in the future. These improvements can all be traced back to attention – Self-attention.

This article simplifies BERT for easy understanding. We started with the sentence–BERT is a precise, huge, masked language model. Breaking down the technical terms used in the sentence has helped give a brief overview of what the model is about and what it tries to achieve. How it is created gives an insight into what happens behind the scenes, and sentiment analysis is an example of how it is used in practice.

FAQs on BERT Algorithm

What is BERT used for in NLP?

BERT or Bidirectional Encoder Representations from Transformers are used for completing various NLP tasks such as Sentiment Analysis, language translation, etc.

Is BERT an NLP model?

BERT stands for Bidirectional Encoder Representations from Transformers. It is a commonly used machine learning model for applications in NLP.

Is BERT a supervised or unsupervised learning algorithm?

Neither BERT is a supervised learning nor an unsupervised learning algorithm. Rather, BERT is a self-supervised learning algorithm.

PREVIOUS NEXT