Fine-tuning BERT for text classification

Using Hugging Face and Comet to fine-tune BERT models

Derrick Mwiti · Following

Published in Heartbeat · 5 min read · Nov 1, 2022

Photo by Pawel Czerwinski on Unsplash

BERT — Bidirectional Encoder Representations from Transformers — is a pre-trained language model for natural language processing tasks such as text classification and question and answering. This article will look at fine-tuning the BERT for text classification. In the end, the BERT model will learn to label if a review from the `imdb` dataset is positive or negative.

To understand how the model is learning, we need to visualize histograms of the weights and biases, the activations and gradients. To achieve that, we use Comet to track the project. Comet automatically tracks these and other items such as:

- Optimizer Parameters
- Code
- Optimizer Parameters
- Metrics
- Weight histograms

Getting started

When using Comet, these items are logged by default, but you can manually configure what will be logged.

```python
import comet_ml

experiment = comet_ml.Experiment(
    api_key="YOUR_API_KEY",
     project_name="HF", log_code=True,
    auto_metric_logging=True,
    auto_param_logging=True,
    auto_histogram_weight_logging=True,
    auto_histogram_gradient_logging=True,
    auto_histogram_activation_logging=True,
)
```

comet_ml.py hosted with ❤ by GitHub view raw

Log parameters

Logging various parameters makes it easy to update them and compare how they affect the model's performance. You can easily change a parameter when all parameters are saved in one dictionary. The `log_parameters` function is used for logging a dictionary of parameters in Comet.

```python
# these will all get logged
params = {
    "bert": "bert-base-uncased",
    "num_labels": 2,
    "return_tensors": "tf",
    "batch_size": 8,
    "epochs": 3,
    "padding":"max_length",
    "truncation": True,
    "dataset": "imdb",
}

experiment.log_parameters(params)
```

params.py hosted with ❤ by GitHub view raw

Tokenize text data

We'll use the `imdb` dataset to fine-tune BERT. Create a numerical representation of the data because it's in text form. Use the `BertTokenizer` since you are fine-tuning a BERT model. This ensures that the data is in the form that the BERT requires. Next, we define a function that will tokenize the data and apply a maximum length and truncation to ensure that all sentences are the same length. Tokenizing the data converts it to a numerical representation that's acceptable by the machine learning model. You can't pass the raw sentences to the model. N

```
def tokenize_function(examples):
    from transformers import BertTokenizer
    tokenizer = BertTokenizer.from_pretrained(params['bert'])
    return tokenizer(examples["text"], padding=params["padding"], truncation=params["truncation"]
```

tokenize_function.py hosted with ❤ by GitHub view raw

Next, apply the function to the dataset. The map function applies the tokenization function to all the sentences. Next, shuffle the data and select the number of data points you would like to use.

```
from datasets import load_dataset

dataset = load_dataset(params['dataset'])

tokenizer = AutoTokenizer.from_pretrained(params['bert'])
tokenized_datasets = dataset.map(tokenize_function, batched=True)

small_train_dataset = tokenized_datasets["train"].shuffle(seed=42).select(range(1000))
small_eval_dataset = tokenized_datasets["test"].shuffle(seed=42).select(range(1000))
```

load_dataset.py hosted with ❤ by GitHub view raw

Create TensorFlow dataset

We'll fine-tune the BERT model in TensorFlow. Let's convert the dataset to a TensorFlow dataset format. Hugging Face provides the `DefaultDataCollator` function to batch the dataset and perform data augmentation. After that, use the `to_tf_dataset` function to convert the dataset to TensorFlow format.

The `to_tf_dataset` method allows you to define the columns and labels included in the dataset. Converting the data to TensorFlow makes it possible to train the model using the `fit` method and later evaluate it using the `evaluate` method.

```
from transformers import DefaultDataCollator
data_collator = DefaultDataCollator(return_tensors=params['return_tensors'])

tf_train_dataset = small_train_dataset.to_tf_dataset(
    columns=["attention_mask", "input_ids", "token_type_ids"],
    label_cols=["labels"],
    shuffle=True,
    collate_fn=data_collator,
    batch_size=params['batch_size'],)

tf_validation_dataset = small_eval_dataset.to_tf_dataset(
    columns=["attention_mask", "input_ids", "token_type_ids"],
    label_cols=["labels"],
    shuffle=False,
    collate_fn=data_collator,
    batch_size=params['batch_size'],)
```

tf_dataset.py hosted with ❤ by GitHub view raw

Train BERT model

The `TFAutoModelForSequenceClassification` is a model class with a sequence classification head. We can use it to initialize a pre-trained BERT classification model. Next, compile the model under a low learning rate and fit it to the data. Using a low learning rate is important in transfer learning to ensure that we don't overfit the model.

```
import tensorflow as tf
from transformers import TFAutoModelForSequenceClassification

bert = TFAutoModelForSequenceClassification.from_pretrained(params['bert'], num_labels=params['nu
bert.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=5e-5),loss=tf.keras.losses.SparseCa
bert.fit(tf_train_dataset, validation_data=tf_validation_dataset, epochs=params['epochs'])
```

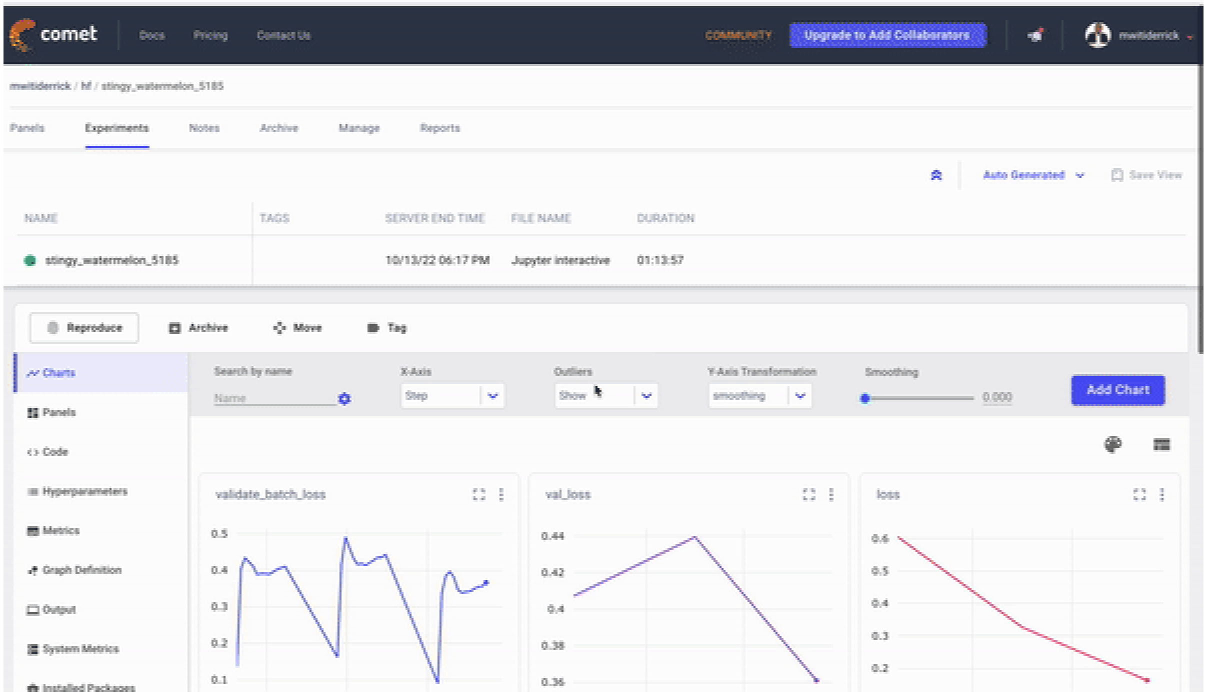
train.py hosted with ❤ by GitHub view raw

> Innovation and academia go hand-in-hand. Listen to our own CEO Gideon Mendels chat with the Stanford MLSys Seminar Series team about the future of MLOps and give the Comet platform a try for free!

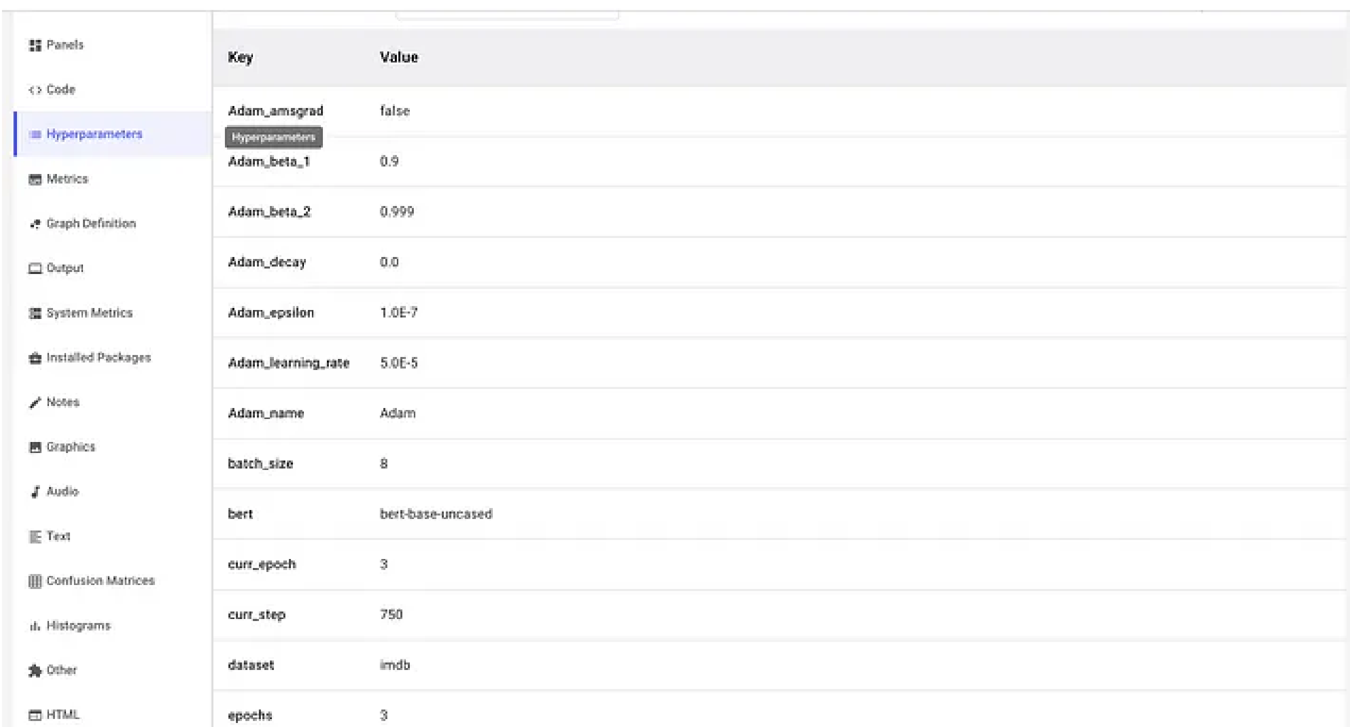
Evaluate model performance

Since auto-logging is active, you will see live results of the model training on Comet. On the charts panel, you will see graphs for the:

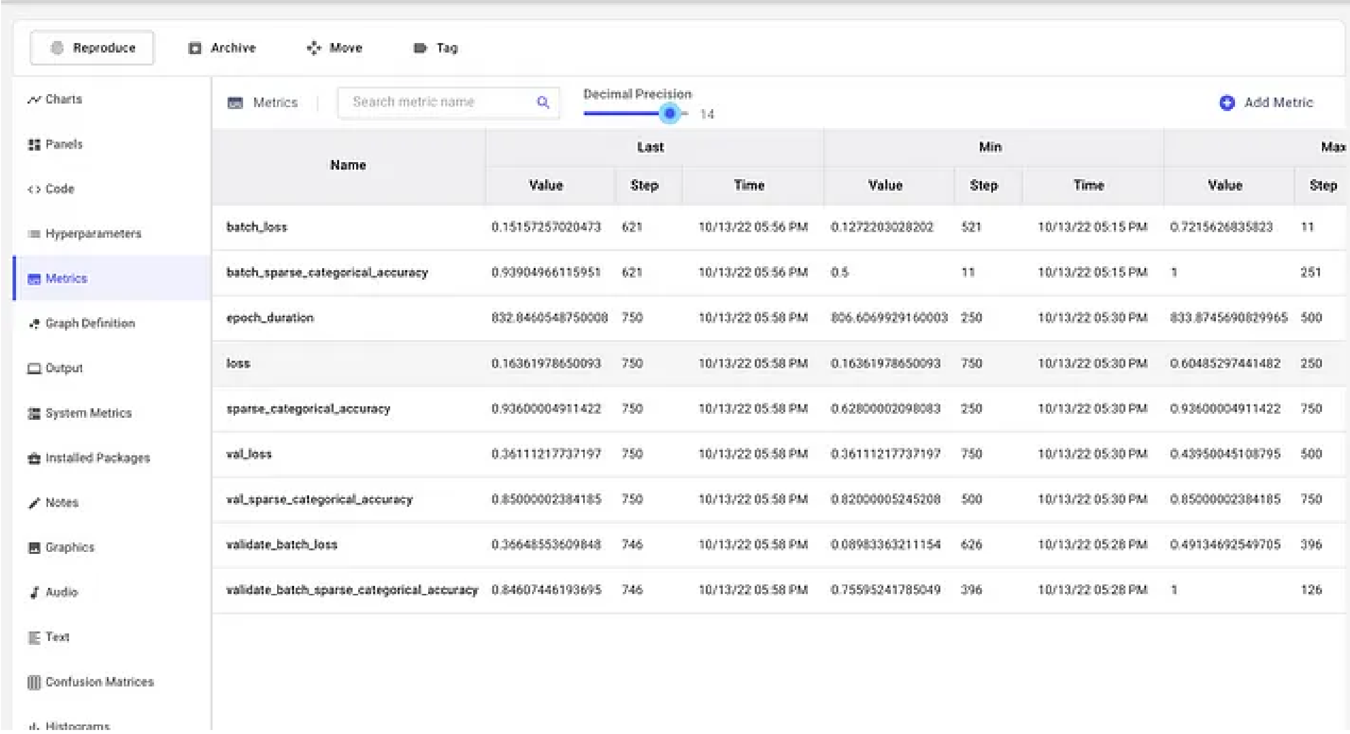
- Loss
- Accuracy
- Epoch duration



The **Code** tab will show the code used in this experiment. On the hyperparameters tab, you will see all the logged parameters.

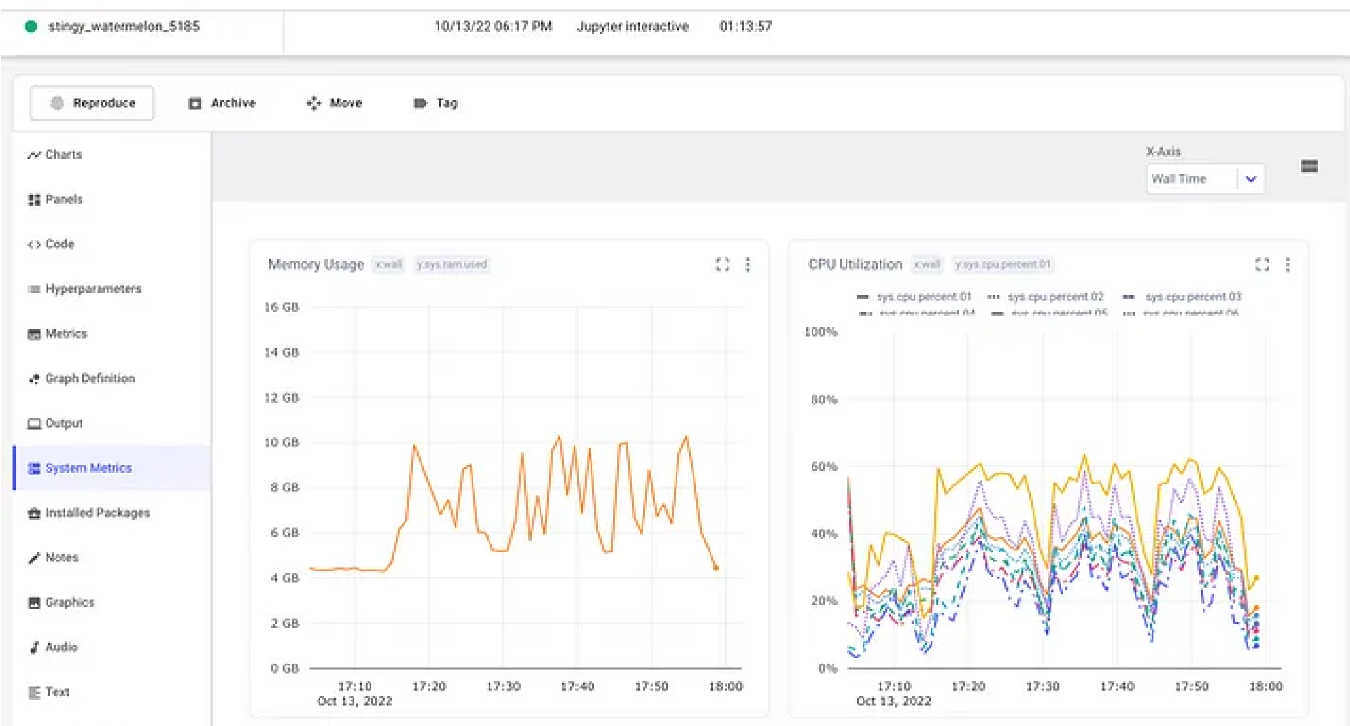


All model metrics can be viewed from the **Metrics** tab.

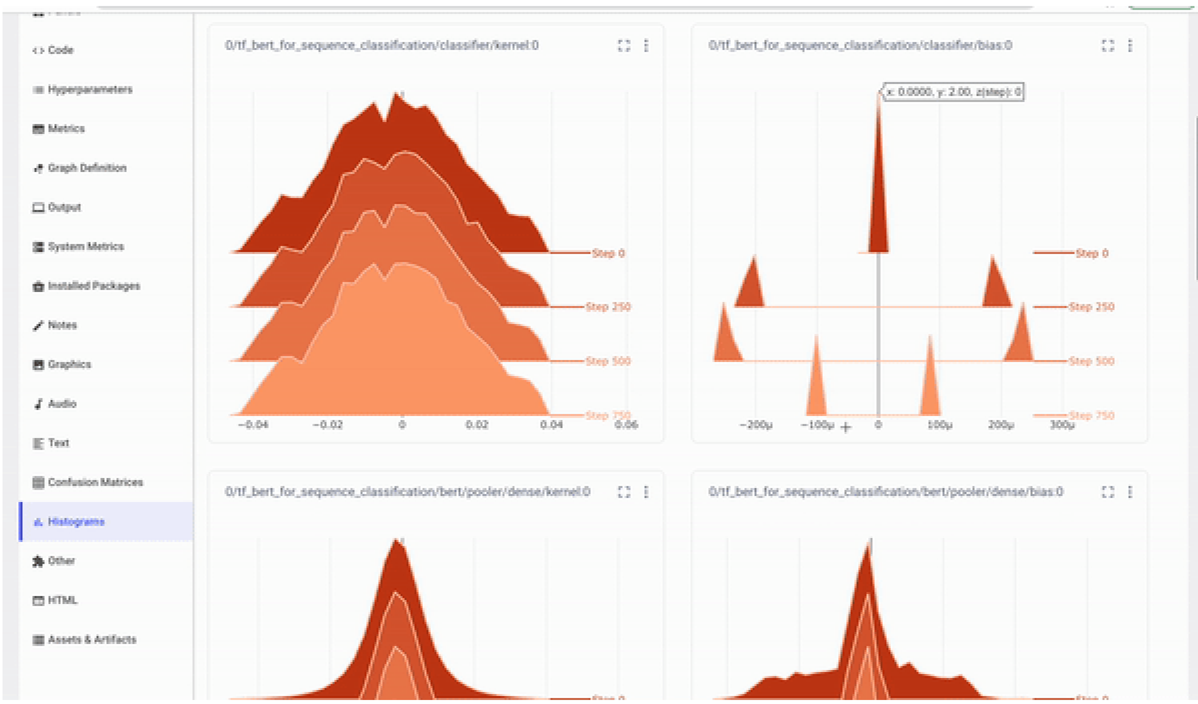


Name	Last			Min			Max	
	Value	Step	Time	Value	Step	Time	Value	Step
batch_loss	0.15157257020473	621	10/13/22 05:56 PM	0.1272203028202	521	10/13/22 05:15 PM	0.7215626835823	11
batch_sparse_categorical_accuracy	0.93904966115951	621	10/13/22 05:56 PM	0.5	11	10/13/22 05:15 PM	1	251
epoch_duration	832.8460548750008	750	10/13/22 05:58 PM	806.6069929160003	250	10/13/22 05:30 PM	833.8745690829965	500
loss	0.16361978650093	750	10/13/22 05:58 PM	0.16361978650093	750	10/13/22 05:30 PM	0.60485297441482	250
sparse_categorical_accuracy	0.93600004911422	750	10/13/22 05:58 PM	0.62800002098083	250	10/13/22 05:30 PM	0.93600004911422	750
val_loss	0.36111217737197	750	10/13/22 05:58 PM	0.36111217737197	750	10/13/22 05:30 PM	0.43950045108795	500
val_sparse_categorical_accuracy	0.85000002384185	750	10/13/22 05:58 PM	0.82000005245208	500	10/13/22 05:30 PM	0.85000002384185	750
validate_batch_loss	0.36648553609848	746	10/13/22 05:58 PM	0.08983363211154	626	10/13/22 05:28 PM	0.49134692549705	396
validate_batch_sparse_categorical_accuracy	0.84607446193695	746	10/13/22 05:58 PM	0.75595241785049	396	10/13/22 05:28 PM	1	126

Click the **System Metrics** tab to see the Memory Usage and CPU Utilization for the model training process.



Click the **Histograms** tab to see histograms for the weights and biases, activations, and gradients.



Test model on new data

Check how the BERT model performs on new data. You can also log the test sentence to Comet. First, tokenize the input data, then pass it to the BERT model. It will output logits which you will need to decode.

```
input_sequence = "I hated that movie, it was too slow"
experiment.log_text(input_sequence)
# encode context the generation is conditioned on
input_ids = tokenizer.encode(input_sequence, return_tensors='tf')
output = bert(input_ids)
logits = output.logits
```

test.py hosted with ❤ by GitHub view raw

```
input_sequence = "I hated that movie, it was too slow"
experiment.log_text(input_sequence)
# encode context the generation is conditioned on
input_ids = tokenizer.encode(input_sequence, return_tensors='tf')
output = bert(input_ids)
logits = output.logits
```

```
logits
```

```
<tf.Tensor: shape=(1, 2), dtype=float32, numpy=array([[ 0.87333816, -0.3748475 ]], dtype=float32)>
```

Let's interpret the prediction and log it as well. You can get the predicted class by passing the logits to `tf.math.argmax`. Passing the predicted class to `bert.config.id2label` will give you the predicted label.

```
predicted_class_id = int(tf.math.argmax(logits, axis=-1)[0])
prediction = bert.config.id2label[predicted_class_id]
experiment.log_text(prediction)
prediction
```



```
import tensorflow as tf
predicted_class_id = int(tf.math.argmax(logits, axis=-1)[0])
prediction = bert.config.id2label[predicted_class_id]
prediction
```

```
'LABEL_0'
```

End the experiment to make sure all items are logged as expected.

```
experiment.end()
```

end.py hosted with ❤ by GitHub view raw

Final thoughts

This article has shown you how to fine-tune a BERT model for text classification while tracking the model using Comet. You can improve this model by increasing the amount of training data. You can also swap the BERT model with another Hugging Face transformer model and compare the performance.

Follow me on LinkedIn for more technical resources.

Resources

Comet experiment

Notebook

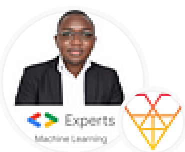
Editor's Note: Heartbeat is a contributor-driven online publication and community dedicated to providing premier educational resources for data science, machine learning, and deep learning practitioners. We're committed to supporting and inspiring developers and engineers from all walks of life.

Editorially independent, Heartbeat is sponsored and published by Comet, an MLOps platform that enables data scientists & ML teams to track, compare, explain, & optimize their experiments. We pay our contributors, and we don't sell ads.

If you'd like to contribute, head on over to our call for contributors. You can also sign up to receive our weekly newsletter (Deep Learning Weekly), check out the Comet blog, join us on Slack, and follow Comet on Twitter and

LinkedIn for resources, events, and much more that will help you build better ML models, faster.

Hugging Face Bert Comet Machine Learning Text Classification



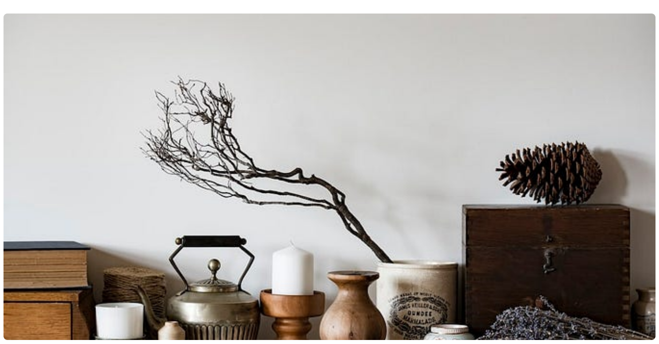
Written by Derrick Mwiti

Following

2.6K Followers · Writer for Heartbeat

Google D. E. — Machine Learning. Follow me at https://twitter.com/themwiti

More from Derrick Mwiti and Heartbeat



Derrick Mwiti in Towards Data Science

Adhing'a Fredrick in Heartbeat

Object Detection with TensorFlow 2 Object Detection API

Object detection with Mask R-CNN in TensorFlow

8 min read · Jun 26, 2022

9

Kangas: The Pandas of Computer Vision

Introduction

7 min read · May 1

464 7

Tirendaz AI in Heartbeat

Building a Text Classifier App with Hugging Face, BERT, and Comet

Implementing end-to-end deep learning projects has never been easier with these…

10 min read · Sep 12

50 2

Derrick Mwiti in Heartbeat

Dealing with Imbalanced Data in Machine Learning

Tools & techniques for handling data when it's imbalanced

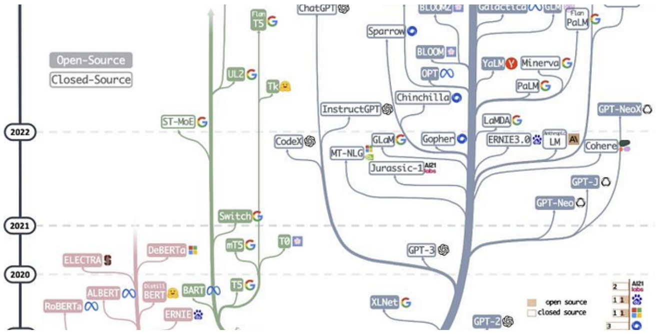
5 min read · Aug 4, 2020

103

See all from Derrick Mwiti

See all from Heartbeat

Recommended from Medium



Haifeng Li

A Tutorial on LLM

Generative artificial intelligence (GenAI), especially ChatGPT, captures everyone's...

15 min read · Sep 14

568

David O Anifowoshe

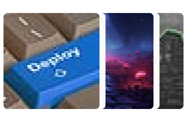
Fine-Tuning RoBERTa for COVID-19 Tweet Sentiment Classification

An NLP project using Transformers and Hugging Face for state-of-the-art results.

4 min read · Jul 24

4

Lists



Predictive Modeling w/ Python

20 stories · 464 saves

Practical Guides to Machine Learning

10 stories · 535 saves

Natural Language Processing

683 stories · 299 saves

The New Chatbots: ChatGPT, Bard, and Beyond

13 stories · 136 saves

The Python Lab

How to Perform Sentiment Analysis using BERT in Python

Sentiment analysis, also known as opinion mining, is a field within natural language...

· 5 min read · May 23

26

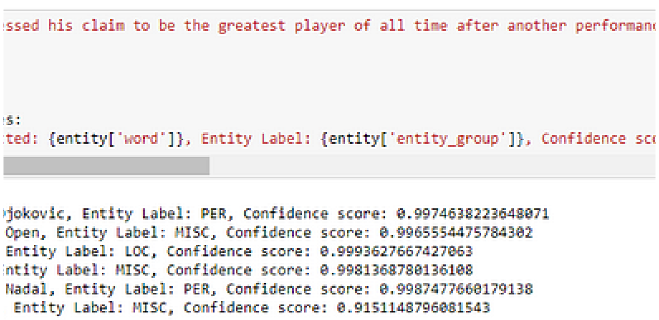
Alidu Abubakari in AI Science

Taking Sentiment Analysis to the Next Level with Huggingface's...

Introduction

17 min read · May 31

104 1



Seffa B

Named Entity Recognition with Transformers: Extracting Metadata

3 min read · Jun 12

7

Bright Eshun

Sentiment Analysis (Part 1): Finetuning DistilBert for Text...

I. Introduction

9 min read · May 8

2

See more recommendations