Out-of-the-box NLP functionalities for your pr using Transformers Library!

Sharan Babu — January 9, 2021

Advanced Deep Learning NLP Project Python Technique Text

This article was published as a part of the Data Science Blogathon.

Introduction

In this tutorial, you will learn how you can integrate common Natural Language Processing (NLP) functionalities into your application with minimal effort. We will be doing this using the '*transformers*' library provided by **Hugging Face**.

1. First, Install the **transformers** library.

```
# Install the library
!pip install transformers
```

```
[ ]  # Install the library
     !pip install transformers

     Collecting transformers
       Downloading https://files.pythonhosted.org/packages/50/0c/7d5950fcd80b029be0a8891727ba21e0cd27692c407c51261c3c921f6da3/transformers-4.1.1-py3-none-any.whl (1.5MB)
          |████████████████████████████████| 1.5MB 13.9MB/s
     Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.6/dist-packages (from transformers) (2019.12.20)
     Requirement already satisfied: requests in /usr/local/lib/python3.6/dist-packages (from transformers) (2.23.0)
     Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.6/dist-packages (from transformers) (4.41.1)
     Requirement already satisfied: packaging in /usr/local/lib/python3.6/dist-packages (from transformers) (20.8)
     Requirement already satisfied: numpy in /usr/local/lib/python3.6/dist-packages (from transformers) (1.19.4)
     Collecting tokenizers==0.9.4
       Downloading https://files.pythonhosted.org/packages/0f/1c/e789a8b12e28be5bc1ce2156cf87cb522b379be9cadc7ad8091a4cc107c4/tokenizers-0.9.4-cp36-cp36m-manylinux2010_x86_64.whl (2.9MB)
          |████████████████████████████████| 2.9MB 25.0MB/s
     Requirement already satisfied: dataclasses; python_version < "3.7" in /usr/local/lib/python3.6/dist-packages (from transformers) (0.8)
```

2. Next, import the necessary functions.

```
# Necessary imports
from transformers import pipeline
```

```
[ ]  # Necessary imports
     from transformers import pipeline
```

3. Irrespective of the task that we want to perform using this library, we have to first create a **pipeline** object which will intake other parameters and give an appropriate output. The required model weights will be downloaded the first time when the code is run.

Sentiment Analysis

Here is an example of how you can easily perform **sentiment analysis**. Sentiment analysis is predicting what sentiment, a sentence falls in. In other words, the model tries to classify whether the sentence was positive or negative.

```
# Create pipeline instance
sentiment = pipeline('sentiment-analysis')
# Positive sentiment example
result = sentiment('I am really happy today')
print(result)
print()
# Negative sentiment example
result = sentiment('I am feeling kinda sad')
print(result)
```



```
[ ]  # Create pipeline instance
     sentiment = pipeline('sentiment-analysis')

     # Positive sentiment example
     result = sentiment('I am really happy today')
     print(result)
     print()
     # Negative sentiment example
     result = sentiment('I am feeling kinda sad')
     print(result)
```

```
[{'label': 'POSITIVE', 'score': 0.999879777431488}]

[{'label': 'NEGATIVE', 'score': 0.9992819428443909}]
```

Question Answering

Code for performing **Question-Answering tasks**. Use the template in the image given below. Enter your question in the 'question' key of the dictionary passed into the pipeline object and the reference material in the 'context' key.

```
q_and_a = pipeline('question-answering')
result = q_and_a({
                  'question': 'What is the purpose of this tutorial?',
                  'context': 'This tutorial lets you implement NLP functionalities with ease'
                })
print(result)
print()
print(f"The answer to your question: {result['answer']}")
```

```
[ ]  q_and_a = pipeline('question-answering')
     result = q_and_a({
                       'question': 'What is the purpose of this tutorial?',
                       'context': 'This tutorial lets you implement NLP functionalities with ease'
                     })
     print(result)
     print()
     print(f"The answer to your question: {result['answer']}")
```

```
{'score': 0.5186614990234375, 'start': 14, 'end': 62, 'answer': 'lets you implement NLP functionalities with ease'}

The answer to your question: lets you implement NLP functionalities with ease
```

Masking

Code for **masking**, i.e., filling missing words in sentences. The missing word to be predicted is to be represented using '<mask>' as shown in the code execution image below.

```
mask = pipeline('fill-mask')
result = mask('Deep <mask> networks work amazingly well on images!')
print(result)
print()
```

```
[ ] mask = pipeline('fill-mask')
    result = mask('Deep <mask> networks work amazingly well on images!')
    print(result)
    print()
    for x in result:
        print(f"Predicted string: {x['token_str']}, score:{x['score']}")
```

```
Some weights of RobertaForMaskedLM were not initialized from the model checkpoint at distilrobe
You should probably TRAIN this model on a down-stream task to be able to use it for predictions
[{'sequence': '<s>Deep neural networks work amazingly well on images!</s>', 'score': 0.95252579

Predicted string: Ġneural, score:0.952525794506073
Predicted string: ĠNeural, score:0.02083066664636135
Predicted string: Ġlearning, score:0.007833791896700859
Predicted string: Ġpacket, score:0.0016278893454000354
Predicted string: Ġquantum, score:0.0013087533880025148
```


Named Entity Recognition

Implementing **Named Entity Recognition** (NER). Named Entity Recognition deals with extracting entities from a given sentence. For Example, '*Adam*' would be extracted as a 'name', and '*19* would be extracted as a 'number'.

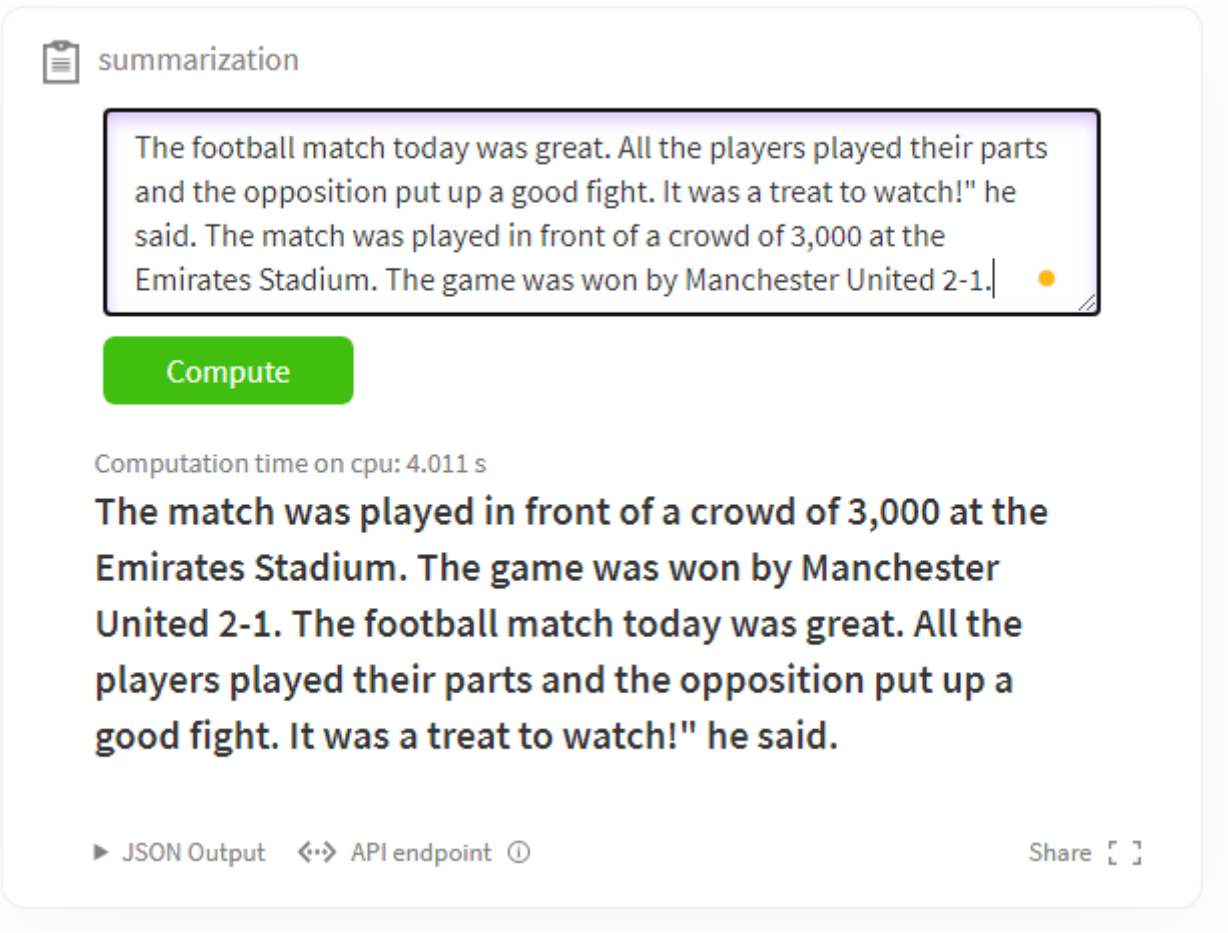
```
ner = pipeline('ner')
result = ner('Analytics Vidhya is a company in India')
print(result)
```

```
ner = pipeline('ner')
result = ner('Analytics Vidhya is a company in India.')
print(result)
```

```
[{'word': 'Ana', 'score': 0.9955880045890808, 'entity': 'I-ORG', 'index': 1, 'start': 0, 'end': 3}, {'word': '##ly', 'score': 0.9708830118179321, 'entity': 'I-ORG', 'index': 2, 'start': 3, 'end': 5}, {'word
```

Text Summarization

These were some of the common out-of-the-box NLP functionalities that you can easily implement using the transformers library. There are many other functionalities, and you can check them out at the Hugging Face website. Here is an example of '**Text Summarization**'. Text Summarization takes in a passage as input and tries to summarize it.



Great! Now, you can integrate NLP functionalities with high performance directly in your applications.

Related Links:

1. Notebook Link for directly running the code in Google Colaboratory.
2. You can watch almost all the functionalities shown in this tutorial in this youtube video made by '*MLNerdie Delhi*'.
3. You can have a look at all the models provided by Hugging face and try them on their website.

The media shown in this article are not owned by Analytics Vidhya and is used at the Author's discretion.

blogathon transformers

About the Author

Sharan Babu

Our Top Authors

view more

Previous Post
Inferential Statistics – Sampling Distribution, Central Limit Theorem and Confidence Interval

Next Post
Exploratory Data Analysis on NYC Taxi Trip Duration Dataset

Leave a Reply

Your email address will not be published. Required fields are marked *

Comment

Name*

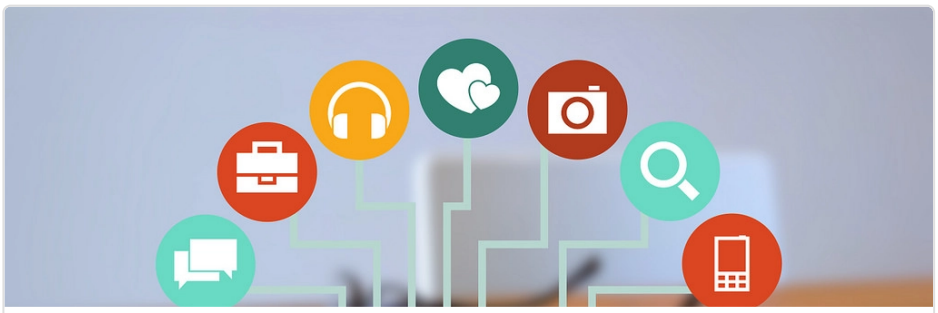
Email*

Website

Notify me of follow-up comments by email.

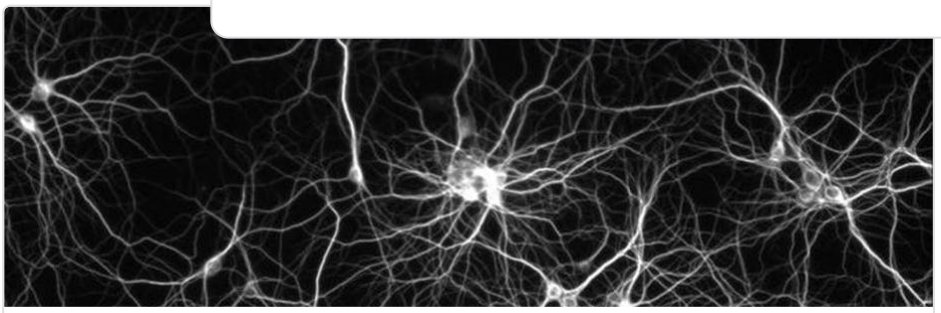
Notify me of new posts by email.

Top Resources



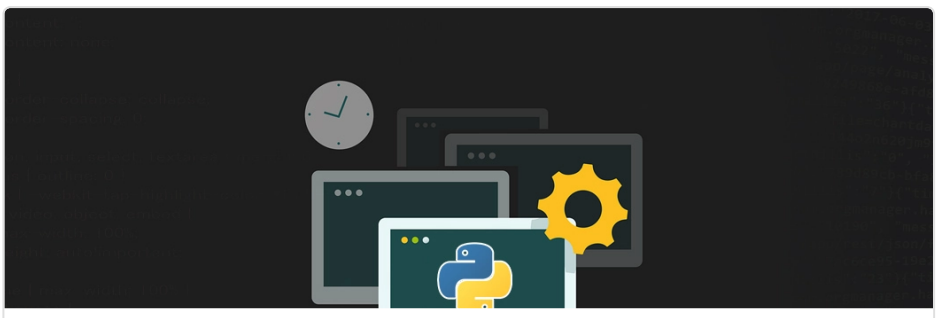
Youtube Video Downloader using Python

Atulya Khatri - FEB 01, 2022



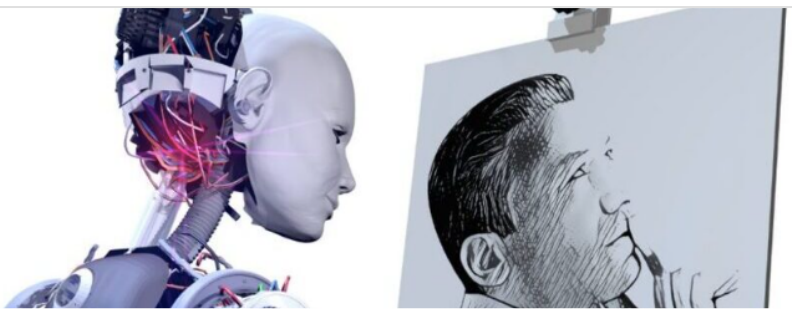
CONVOLUTIONAL NEURAL NETWORK(CNN)

Shreya12@ - JAN 28, 2022



Python Tutorial: Working with CSV file for Data Science

Harika Bonthu - AUG 21, 2021



An Introduction to Synthetic Image Generation from Text Data

Suvojit Hore - JAN 28, 2022