Natural Language Processing

Group Member Names:

Shah Wali Ullah Baig - 191980008

Accessing GPU Resources

To access GPU resources in Google Colab, go to the "Runtime" option (located in the centre of the navigationbar's options), choose "Change runtime Type", click the downward arrow, select "GPU", and click save.

We will access our datasets from Google Drive to Collaborate directly by connecting them. But before making the connection, we will create a folder in our drive and upload the dataset file (zip file) to that folder.

Connect Colab with Google Drive

To connect Google Colab with Google Drive, first we will import the drive library from Google.colab pakage and mount it, then run this code.

After running this code, he will ask for permission to connect. Simply choose the 'Connect to Google Drive' option. Then a new window will appear with a list of Google accounts (if you are logged in multiple accounts). Choose a Google account with whose drive you want to connect, and lastly, press the 'Allow' button.

In [ ]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

Unzip the Folder

To unzip the folder, we will simply use the 'unzip' keyword and give it the path of the zip file (uploaded on the drive) and run this cell. It will automatically make a folder in Google Colab files with the same name as the zip file's name.

To get the path, go to that zip file, right click on it and select 'Get link'. There will be an option to "copy link".

In [ ]:
```
!unzip '/content/drive/MyDrive/Colab Notebooks/NLP-Project/acllmdb.zip'
```

```
Streaming output truncated to the last 5000 lines.
  inflating: aclImdb/train/unsup/5504_0.txt
  inflating: aclImdb/train/unsup/5505_0.txt
  inflating: aclImdb/train/unsup/5506_0.txt
  inflating: aclImdb/train/unsup/5507_0.txt
  inflating: aclImdb/train/unsup/5508_0.txt
  inflating: aclImdb/train/unsup/5509_0.txt
  inflating: aclImdb/train/unsup/550_0.txt
  inflating: aclImdb/train/unsup/5510_0.txt
  inflating: aclImdb/train/unsup/5511_0.txt
  inflating: aclImdb/train/unsup/5512_0.txt
  inflating: aclImdb/train/unsup/5513_0.txt
  inflating: aclImdb/train/unsup/5514_0.txt
  inflating: aclImdb/train/unsup/5515_0.txt
  inflating: aclImdb/train/unsup/5516_0.txt
  inflating: aclImdb/train/unsup/5517_0.txt
  inflating: aclImdb/train/unsup/5518_0.txt
  inflating: aclImdb/train/unsup/5519_0.txt
  inflating: aclImdb/train/unsup/551_0.txt
```

Install Required Libraries

In [ ]:
```
!pip install -U spacy                                  # To import spacy library
```

```
Looking in indexes: https://pypi.org/simple, (https://pypi.org/simple,) ht
tps://us-python.pkg.dev/colab-wheels/public/simple/ (https://us-python.pk
g.dev/colab-wheels/public/simple/)
Requirement already satisfied: spacy in /usr/local/lib/python3.7/dist-pack
ages (3.4.1)
Requirement already satisfied: preshed<3.1.0,>=3.0.2 in /usr/local/lib/pyt
hon3.7/dist-packages (from spacy) (3.0.7)
Requirement already satisfied: wasabi<1.1.0,>=0.9.1 in /usr/local/lib/pyth
on3.7/dist-packages (from spacy) (0.10.1)
Requirement already satisfied: spacy-loggers<2.0.0,>=1.0.0 in /usr/local/l
ib/python3.7/dist-packages (from spacy) (1.0.3)
Requirement already satisfied: murmurhash<1.1.0,>=0.28.0 in /usr/local/li
b/python3.7/dist-packages (from spacy) (1.0.8)
Requirement already satisfied: numpy>=1.15.0 in /usr/local/lib/python3.7/d
ist-packages (from spacy) (1.21.6)
Requirement already satisfied: spacy-legacy<3.1.0,>=3.0.9 in /usr/local/li
b/python3.7/dist-packages (from spacy) (3.0.10)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.
7/dist-packages (from spacy) (21.3)
Requirement already satisfied: catalogue<2.1.0,>=2.0.6 in /usr/local/lib/p
ython3.7/dist-packages (from spacy) (2.0.8)
Requirement already satisfied: langcodes<4.0.0,>=3.2.0 in /usr/local/lib/p
ython3.7/dist-packages (from spacy) (3.3.0)
Requirement already satisfied: typer<0.5.0,>=0.3.0 in /usr/local/lib/pytho
n3.7/dist-packages (from spacy) (0.4.2)
Requirement already satisfied: setuptools in /usr/local/lib/python3.7/dist
-packages (from spacy) (57.4.0)
Requirement already satisfied: requests<3.0.0,>=2.13.0 in /usr/local/lib/p
ython3.7/dist-packages (from spacy) (2.23.0)
Requirement already satisfied: pydantic!=1.8,!=1.8.1,<1.10.0,>=1.7.4 in /u
sr/local/lib/python3.7/dist-packages (from spacy) (1.9.2)
Requirement already satisfied: thinc<8.2.0,>=8.1.0 in /usr/local/lib/pytho
n3.7/dist-packages (from spacy) (8.1.0)
Requirement already satisfied: tqdm<5.0.0,>=4.38.0 in /usr/local/lib/pytho
n3.7/dist-packages (from spacy) (4.64.1)
Requirement already satisfied: srsly<3.0.0,>=2.4.3 in /usr/local/lib/pytho
n3.7/dist-packages (from spacy) (2.4.4)
Requirement already satisfied: pathy>=0.3.5 in /usr/local/lib/python3.7/di
st-packages (from spacy) (0.6.2)
Requirement already satisfied: cymem<2.1.0,>=2.0.2 in /usr/local/lib/pytho
n3.7/dist-packages (from spacy) (2.0.6)
Requirement already satisfied: typing-extensions<4.2.0,>=3.7.4 in /usr/loc
al/lib/python3.7/dist-packages (from spacy) (4.1.1)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.7/dist-pac
kages (from spacy) (2.11.3)
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.7/dist-
packages (from catalogue<2.1.0,>=2.0.6->spacy) (3.8.1)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in /usr/local/lib/
python3.7/dist-packages (from packaging>=20.0->spacy) (3.0.9)
Requirement already satisfied: smart-open<6.0.0,>=5.2.1 in /usr/local/lib/
python3.7/dist-packages (from pathy>=0.3.5->spacy) (5.2.1)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python
3.7/dist-packages (from requests<3.0.0,>=2.13.0->spacy) (2022.6.15)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/di
st-packages (from requests<3.0.0,>=2.13.0->spacy) (2.10)
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python
3.7/dist-packages (from requests<3.0.0,>=2.13.0->spacy) (3.0.4)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in
/usr/local/lib/python3.7/dist-packages (from requests<3.0.0,>=2.13.0->spac
y) (1.24.3)
Requirement already satisfied: blis<0.8.0,>=0.7.8 in /usr/local/lib/python
```

```
3.7/dist-packages (from thinc<8.2.0,>=8.1.0->spacy) (0.7.8)
Requirement already satisfied: click<9.0.0,>=7.1.1 in /usr/local/lib/pytho
n3.7/dist-packages (from typer<0.5.0,>=0.3.0->spacy) (7.1.2)
Requirement already satisfied: MarkupSafe>=0.23 in /usr/local/lib/python3.
7/dist-packages (from jinja2->spacy) (2.0.1)
```

In [ ]:
```
!python -m spacy download en                    # en contains the language-sp
```

```
2022-09-25 18:25:42.497011: E tensorflow/stream_executor/cuda/cuda_driver.
cc:271] failed call to cuInit: CUDA_ERROR_NO_DEVICE: no CUDA-capable devic
e is detected
⚠ As of spaCy v3.0, shortcuts like 'en' are deprecated. Please use the
full pipeline package name 'en_core_web_sm' instead.
Looking in indexes: https://pypi.org/simple, (https://pypi.org/simple,) ht
tps://us-python.pkg.dev/colab-wheels/public/simple/ (https://us-python.pk
g.dev/colab-wheels/public/simple/)
Collecting en-core-web-sm==3.4.0
  Downloading https://github.com/explosion/spacy-models/releases/download/
en_core_web_sm-3.4.0/en_core_web_sm-3.4.0-py3-none-any.whl (https://githu
b.com/explosion/spacy-models/releases/download/en_core_web_sm-3.4.0/en_cor
e_web_sm-3.4.0-py3-none-any.whl) (12.8 MB)
     |████████████████████████████████| 12.8 MB 8.1 MB/s
Requirement already satisfied: spacy<3.5.0,>=3.4.0 in /usr/local/lib/pytho
n3.7/dist-packages (from en-core-web-sm==3.4.0) (3.4.1)
Requirement already satisfied: pydantic!=1.8,!=1.8.1,<1.10.0,>=1.7.4 in /u
sr/local/lib/python3.7/dist-packages (from spacy<3.5.0,>=3.4.0->en-core-we
b-sm==3.4.0) (1.9.2)
Requirement already satisfied: typing-extensions<4.2.0,>=3.7.4 in /usr/loc
al/lib/python3.7/dist-packages (from spacy<3.5.0,>=3.4.0->en-core-web-sm==
3.4.0) (4.1.1)
Requirement already satisfied: catalogue<2.1.0,>=2.0.6 in /usr/local/lib/p
ython3.7/dist-packages (from spacy<3.5.0,>=3.4.0->en-core-web-sm==3.4.0)
(2.0.8)
Requirement already satisfied: pathy>=0.3.5 in /usr/local/lib/python3.7/di
st-packages (from spacy<3.5.0,>=3.4.0->en-core-web-sm==3.4.0) (0.6.2)
Requirement already satisfied: spacy-legacy<3.1.0,>=3.0.9 in /usr/local/li
b/python3.7/dist-packages (from spacy<3.5.0,>=3.4.0->en-core-web-sm==3.4.
0) (3.0.10)
Requirement already satisfied: typer<0.5.0,>=0.3.0 in /usr/local/lib/pytho
n3.7/dist-packages (from spacy<3.5.0,>=3.4.0->en-core-web-sm==3.4.0) (0.4.
2)
Requirement already satisfied: tqdm<5.0.0,>=4.38.0 in /usr/local/lib/pytho
n3.7/dist-packages (from spacy<3.5.0,>=3.4.0->en-core-web-sm==3.4.0) (4.6
4.1)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.7/dist-pac
kages (from spacy<3.5.0,>=3.4.0->en-core-web-sm==3.4.0) (2.11.3)
Requirement already satisfied: cymem<2.1.0,>=2.0.2 in /usr/local/lib/pytho
n3.7/dist-packages (from spacy<3.5.0,>=3.4.0->en-core-web-sm==3.4.0) (2.0.
6)
Requirement already satisfied: requests<3.0.0,>=2.13.0 in /usr/local/lib/p
ython3.7/dist-packages (from spacy<3.5.0,>=3.4.0->en-core-web-sm==3.4.0)
(2.23.0)
Requirement already satisfied: wasabi<1.1.0,>=0.9.1 in /usr/local/lib/pyth
on3.7/dist-packages (from spacy<3.5.0,>=3.4.0->en-core-web-sm==3.4.0) (0.1
0.1)
Requirement already satisfied: thinc<8.2.0,>=8.1.0 in /usr/local/lib/pytho
n3.7/dist-packages (from spacy<3.5.0,>=3.4.0->en-core-web-sm==3.4.0) (8.1.
0)
Requirement already satisfied: preshed<3.1.0,>=3.0.2 in /usr/local/lib/pyt
hon3.7/dist-packages (from spacy<3.5.0,>=3.4.0->en-core-web-sm==3.4.0) (3.
0.7)
Requirement already satisfied: numpy>=1.15.0 in /usr/local/lib/python3.7/d
ist-packages (from spacy<3.5.0,>=3.4.0->en-core-web-sm==3.4.0) (1.21.6)
Requirement already satisfied: setuptools in /usr/local/lib/python3.7/dist
-packages (from spacy<3.5.0,>=3.4.0->en-core-web-sm==3.4.0) (57.4.0)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.
7/dist-packages (from spacy<3.5.0,>=3.4.0->en-core-web-sm==3.4.0) (21.3)
Requirement already satisfied: spacy-loggers<2.0.0,>=1.0.0 in /usr/local/l
ib/python3.7/dist-packages (from spacy<3.5.0,>=3.4.0->en-core-web-sm==3.4.
```

```
0) (1.0.3)
Requirement already satisfied: srsly<3.0.0,>=2.4.3 in /usr/local/lib/pytho
n3.7/dist-packages (from spacy<3.5.0,>=3.4.0->en-core-web-sm==3.4.0) (2.4.
4)
Requirement already satisfied: murmurhash<1.1.0,>=0.28.0 in /usr/local/li
b/python3.7/dist-packages (from spacy<3.5.0,>=3.4.0->en-core-web-sm==3.4.
0) (1.0.8)
Requirement already satisfied: langcodes<4.0.0,>=3.2.0 in /usr/local/lib/p
ython3.7/dist-packages (from spacy<3.5.0,>=3.4.0->en-core-web-sm==3.4.0)
(3.3.0)
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.7/dist-
packages (from catalogue<2.1.0,>=2.0.6->spacy<3.5.0,>=3.4.0->en-core-web-s
m==3.4.0) (3.8.1)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in /usr/local/lib/
python3.7/dist-packages (from packaging>=20.0->spacy<3.5.0,>=3.4.0->en-cor
e-web-sm==3.4.0) (3.0.9)
Requirement already satisfied: smart-open<6.0.0,>=5.2.1 in /usr/local/lib/
python3.7/dist-packages (from pathy>=0.3.5->spacy<3.5.0,>=3.4.0->en-core-w
eb-sm==3.4.0) (5.2.1)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in
/usr/local/lib/python3.7/dist-packages (from requests<3.0.0,>=2.13.0->spac
y<3.5.0,>=3.4.0->en-core-web-sm==3.4.0) (1.24.3)
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python
3.7/dist-packages (from requests<3.0.0,>=2.13.0->spacy<3.5.0,>=3.4.0->en-c
ore-web-sm==3.4.0) (3.0.4)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/di
st-packages (from requests<3.0.0,>=2.13.0->spacy<3.5.0,>=3.4.0->en-core-we
b-sm==3.4.0) (2.10)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python
3.7/dist-packages (from requests<3.0.0,>=2.13.0->spacy<3.5.0,>=3.4.0->en-c
ore-web-sm==3.4.0) (2022.6.15)
Requirement already satisfied: blis<0.8.0,>=0.7.8 in /usr/local/lib/python
3.7/dist-packages (from thinc<8.2.0,>=8.1.0->spacy<3.5.0,>=3.4.0->en-core-
web-sm==3.4.0) (0.7.8)
Requirement already satisfied: click<9.0.0,>=7.1.1 in /usr/local/lib/pytho
n3.7/dist-packages (from typer<0.5.0,>=0.3.0->spacy<3.5.0,>=3.4.0->en-core
-web-sm==3.4.0) (7.1.2)
Requirement already satisfied: MarkupSafe>=0.23 in /usr/local/lib/python3.
7/dist-packages (from jinja2->spacy<3.5.0,>=3.4.0->en-core-web-sm==3.4.0)
(2.0.1)
✓ Download and installation successful
You can now load the package via spacy.load('en_core_web_sm')
```

In [ ]:
```
!pip install tensorflow-text                    # Provides a collection of te
```

```
Looking in indexes: https://pypi.org/simple, (https://pypi.org/simple,)
https://us-python.pkg.dev/colab-wheels/public/simple/ (https://us-pytho
n.pkg.dev/colab-wheels/public/simple/)
Collecting tensorflow-text
  Downloading tensorflow_text-2.10.0-cp37-cp37m-manylinux_2_17_x86_64.ma
nylinux2014_x86_64.whl (5.9 MB)
     |████████████████████████████████| 5.9 MB 6.0 MB/s
Requirement already satisfied: tensorflow-hub>=0.8.0 in /usr/local/lib/p
ython3.7/dist-packages (from tensorflow-text) (0.12.0)
Collecting tensorflow<2.11,>=2.10.0
  Downloading tensorflow-2.10.0-cp37-cp37m-manylinux_2_17_x86_64.manylin
ux2014_x86_64.whl (578.0 MB)
     |████████████████████████████████| 578.0 MB 11 kB/s
Requirement already satisfied: google-pasta>=0.1.1 in /usr/local/lib/pyt
hon3.7/dist-packages (from tensorflow<2.11,>=2.10.0->tensorflow-text)
(0.2.0)
Requirement already satisfied: packaging in /usr/local/lib/python3.7/dis
t-packages (from tensorflow<2.11,>=2.10.0->tensorflow-text) (21.3)
Collecting keras<2.11,>=2.10.0
```

In [ ]:
```
!pip install --upgrade tensorflow_hub           # A repository of trained mac
```

```
Looking in indexes: https://pypi.org/simple, (https://pypi.org/simple,) ht
tps://us-python.pkg.dev/colab-wheels/public/simple/ (https://us-python.pk
g.dev/colab-wheels/public/simple/)
Requirement already satisfied: tensorflow_hub in /usr/local/lib/python3.7/
dist-packages (0.12.0)
Requirement already satisfied: numpy>=1.12.0 in /usr/local/lib/python3.7/d
ist-packages (from tensorflow_hub) (1.21.6)
Requirement already satisfied: protobuf>=3.8.0 in /usr/local/lib/python3.
7/dist-packages (from tensorflow_hub) (3.17.3)
Requirement already satisfied: six>=1.9 in /usr/local/lib/python3.7/dist-p
ackages (from protobuf>=3.8.0->tensorflow_hub) (1.15.0)
```

In [ ]:
```
!pip install transformers                    # Provides APIs to quickly do
```

```
Looking in indexes: https://pypi.org/simple, (https://pypi.org/simple,) ht
tps://us-python.pkg.dev/colab-wheels/public/simple/ (https://us-python.pk
g.dev/colab-wheels/public/simple/)
Collecting transformers
  Downloading transformers-4.22.1-py3-none-any.whl (4.9 MB)
     |████████████████████████████████| 4.9 MB 8.3 MB/s
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.7/dis
t-packages (from transformers) (1.21.6)
Requirement already satisfied: importlib-metadata in /usr/local/lib/python
3.7/dist-packages (from transformers) (4.12.0)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.
7/dist-packages (from transformers) (21.3)
Collecting huggingface-hub<1.0,>=0.9.0
  Downloading huggingface_hub-0.9.1-py3-none-any.whl (120 kB)
     |████████████████████████████████| 120 kB 57.3 MB/s
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.7/dis
t-packages (from transformers) (6.0)
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python
3.7/dist-packages (from transformers) (2022.6.2)
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.7/dist
-packages (from transformers) (4.64.1)
Requirement already satisfied: requests in /usr/local/lib/python3.7/dist-p
ackages (from transformers) (2.23.0)
Collecting tokenizers!=0.11.3,<0.13,>=0.11.1
  Downloading tokenizers-0.12.1-cp37-cp37m-manylinux_2_12_x86_64.manylinux
2010_x86_64.whl (6.6 MB)
     |████████████████████████████████| 6.6 MB 35.6 MB/s
Requirement already satisfied: filelock in /usr/local/lib/python3.7/dist-p
ackages (from transformers) (3.8.0)
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/li
b/python3.7/dist-packages (from huggingface-hub<1.0,>=0.9.0->transformers)
(4.1.1)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in /usr/local/lib/
python3.7/dist-packages (from packaging>=20.0->transformers) (3.0.9)
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.7/dist-
packages (from importlib-metadata->transformers) (3.8.1)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/di
st-packages (from requests->transformers) (2.10)
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python
3.7/dist-packages (from requests->transformers) (3.0.4)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in
/usr/local/lib/python3.7/dist-packages (from requests->transformers) (1.2
4.3)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python
3.7/dist-packages (from requests->transformers) (2022.6.15)
Installing collected packages: tokenizers, huggingface-hub, transformers
Successfully installed huggingface-hub-0.9.1 tokenizers-0.12.1 transformer
s-4.22.1
```

In [ ]:
```
!python -m pip install tensorflow_text          # Provides a collection of te
```

```
Looking in indexes: https://pypi.org/simple, (https://pypi.org/simple,) ht
tps://us-python.pkg.dev/colab-wheels/public/simple/ (https://us-python.pk
g.dev/colab-wheels/public/simple/)
Requirement already satisfied: tensorflow_text in /usr/local/lib/python3.
7/dist-packages (2.10.0)
Requirement already satisfied: tensorflow<2.11,>=2.10.0 in /usr/local/lib/
python3.7/dist-packages (from tensorflow_text) (2.10.0)
Requirement already satisfied: tensorflow-hub>=0.8.0 in /usr/local/lib/pyt
hon3.7/dist-packages (from tensorflow_text) (0.12.0)
Requirement already satisfied: termcolor>=1.1.0 in /usr/local/lib/python3.
7/dist-packages (from tensorflow<2.11,>=2.10.0->tensorflow_text) (1.1.0)
Requirement already satisfied: astunparse>=1.6.0 in /usr/local/lib/python
3.7/dist-packages (from tensorflow<2.11,>=2.10.0->tensorflow_text) (1.6.3)
Requirement already satisfied: flatbuffers>=2.0 in /usr/local/lib/python3.
7/dist-packages (from tensorflow<2.11,>=2.10.0->tensorflow_text) (2.0.7)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in /us
r/local/lib/python3.7/dist-packages (from tensorflow<2.11,>=2.10.0->tensor
flow_text) (0.26.0)
Requirement already satisfied: keras<2.11,>=2.10.0 in /usr/local/lib/pytho
n3.7/dist-packages (from tensorflow<2.11,>=2.10.0->tensorflow_text) (2.10.
0)
Requirement already satisfied: wrapt>=1.11.0 in /usr/local/lib/python3.7/d
ist-packages (from tensorflow<2.11,>=2.10.0->tensorflow_text) (1.14.1)
Requirement already satisfied: tensorflow-estimator<2.11,>=2.10.0 in /usr/
local/lib/python3.7/dist-packages (from tensorflow<2.11,>=2.10.0->tensorfl
ow_text) (2.10.0)
Requirement already satisfied: google-pasta>=0.1.1 in /usr/local/lib/pytho
n3.7/dist-packages (from tensorflow<2.11,>=2.10.0->tensorflow_text) (0.2.
0)
Requirement already satisfied: keras-preprocessing>=1.1.1 in /usr/local/li
b/python3.7/dist-packages (from tensorflow<2.11,>=2.10.0->tensorflow_text)
(1.1.2)
Requirement already satisfied: typing-extensions>=3.6.6 in /usr/local/lib/
python3.7/dist-packages (from tensorflow<2.11,>=2.10.0->tensorflow_text)
(4.1.1)
Requirement already satisfied: absl-py>=1.0.0 in /usr/local/lib/python3.7/
dist-packages (from tensorflow<2.11,>=2.10.0->tensorflow_text) (1.2.0)
Requirement already satisfied: h5py>=2.9.0 in /usr/local/lib/python3.7/dis
t-packages (from tensorflow<2.11,>=2.10.0->tensorflow_text) (3.1.0)
Requirement already satisfied: protobuf<3.20,>=3.9.2 in /usr/local/lib/pyt
hon3.7/dist-packages (from tensorflow<2.11,>=2.10.0->tensorflow_text) (3.1
7.3)
Requirement already satisfied: gast<=0.4.0,>=0.2.1 in /usr/local/lib/pytho
n3.7/dist-packages (from tensorflow<2.11,>=2.10.0->tensorflow_text) (0.4.
0)
Requirement already satisfied: opt-einsum>=2.3.2 in /usr/local/lib/python
3.7/dist-packages (from tensorflow<2.11,>=2.10.0->tensorflow_text) (3.3.0)
Requirement already satisfied: numpy>=1.20 in /usr/local/lib/python3.7/dis
t-packages (from tensorflow<2.11,>=2.10.0->tensorflow_text) (1.21.6)
Requirement already satisfied: setuptools in /usr/local/lib/python3.7/dist
-packages (from tensorflow<2.11,>=2.10.0->tensorflow_text) (57.4.0)
Requirement already satisfied: six>=1.12.0 in /usr/local/lib/python3.7/dis
t-packages (from tensorflow<2.11,>=2.10.0->tensorflow_text) (1.15.0)
Requirement already satisfied: libclang>=13.0.0 in /usr/local/lib/python3.
7/dist-packages (from tensorflow<2.11,>=2.10.0->tensorflow_text) (14.0.6)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in /usr/local/lib/pytho
n3.7/dist-packages (from tensorflow<2.11,>=2.10.0->tensorflow_text) (1.48.
1)
Requirement already satisfied: packaging in /usr/local/lib/python3.7/dist-
packages (from tensorflow<2.11,>=2.10.0->tensorflow_text) (21.3)
Requirement already satisfied: tensorboard<2.11,>=2.10 in /usr/local/lib/p
```

```
ython3.7/dist-packages (from tensorflow<2.11,>=2.10.0->tensorflow_text)
(2.10.0)
Requirement already satisfied: wheel<1.0,>=0.23.0 in /usr/local/lib/python
3.7/dist-packages (from astunparse>=1.6.0->tensorflow<2.11,>=2.10.0->tenso
rflow_text) (0.37.1)
Requirement already satisfied: cached-property in /usr/local/lib/python3.
7/dist-packages (from h5py>=2.9.0->tensorflow<2.11,>=2.10.0->tensorflow_te
xt) (1.5.2)
Requirement already satisfied: werkzeug>=1.0.1 in /usr/local/lib/python3.
7/dist-packages (from tensorboard<2.11,>=2.10->tensorflow<2.11,>=2.10.0->t
ensorflow_text) (1.0.1)
Requirement already satisfied: tensorboard-data-server<0.7.0,>=0.6.0 in /u
sr/local/lib/python3.7/dist-packages (from tensorboard<2.11,>=2.10->tensor
flow<2.11,>=2.10.0->tensorflow_text) (0.6.1)
Requirement already satisfied: requests<3,>=2.21.0 in /usr/local/lib/pytho
n3.7/dist-packages (from tensorboard<2.11,>=2.10->tensorflow<2.11,>=2.10.0
->tensorflow_text) (2.23.0)
Requirement already satisfied: google-auth-oauthlib<0.5,>=0.4.1 in /usr/lo
cal/lib/python3.7/dist-packages (from tensorboard<2.11,>=2.10->tensorflow<
2.11,>=2.10.0->tensorflow_text) (0.4.6)
Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.
7/dist-packages (from tensorboard<2.11,>=2.10->tensorflow<2.11,>=2.10.0->t
ensorflow_text) (3.4.1)
Requirement already satisfied: google-auth<3,>=1.6.3 in /usr/local/lib/pyt
hon3.7/dist-packages (from tensorboard<2.11,>=2.10->tensorflow<2.11,>=2.1
0.0->tensorflow_text) (1.35.0)
Requirement already satisfied: tensorboard-plugin-wit>=1.6.0 in /usr/loca
l/lib/python3.7/dist-packages (from tensorboard<2.11,>=2.10->tensorflow<2.
11,>=2.10.0->tensorflow_text) (1.8.1)
Requirement already satisfied: rsa<5,>=3.1.4 in /usr/local/lib/python3.7/d
ist-packages (from google-auth<3,>=1.6.3->tensorboard<2.11,>=2.10->tensorf
low<2.11,>=2.10.0->tensorflow_text) (4.9)
Requirement already satisfied: pyasn1-modules>=0.2.1 in /usr/local/lib/pyt
hon3.7/dist-packages (from google-auth<3,>=1.6.3->tensorboard<2.11,>=2.10-
>tensorflow<2.11,>=2.10.0->tensorflow_text) (0.2.8)
Requirement already satisfied: cachetools<5.0,>=2.0.0 in /usr/local/lib/py
thon3.7/dist-packages (from google-auth<3,>=1.6.3->tensorboard<2.11,>=2.10
->tensorflow<2.11,>=2.10.0->tensorflow_text) (4.2.4)
Requirement already satisfied: requests-oauthlib>=0.7.0 in /usr/local/lib/
python3.7/dist-packages (from google-auth-oauthlib<0.5,>=0.4.1->tensorboar
d<2.11,>=2.10->tensorflow<2.11,>=2.10.0->tensorflow_text) (1.3.1)
Requirement already satisfied: importlib-metadata>=4.4 in /usr/local/lib/p
ython3.7/dist-packages (from markdown>=2.6.8->tensorboard<2.11,>=2.10->ten
sorflow<2.11,>=2.10.0->tensorflow_text) (4.12.0)
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.7/dist-
packages (from importlib-metadata>=4.4->markdown>=2.6.8->tensorboard<2.11,
>=2.10->tensorflow<2.11,>=2.10.0->tensorflow_text) (3.8.1)
Requirement already satisfied: pyasn1<0.5.0,>=0.4.6 in /usr/local/lib/pyth
on3.7/dist-packages (from pyasn1-modules>=0.2.1->google-auth<3,>=1.6.3->te
nsorboard<2.11,>=2.10->tensorflow<2.11,>=2.10.0->tensorflow_text) (0.4.8)
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python
3.7/dist-packages (from requests<3,>=2.21.0->tensorboard<2.11,>=2.10->tens
orflow<2.11,>=2.10.0->tensorflow_text) (3.0.4)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python
3.7/dist-packages (from requests<3,>=2.21.0->tensorboard<2.11,>=2.10->tens
orflow<2.11,>=2.10.0->tensorflow_text) (2022.6.15)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/di
st-packages (from requests<3,>=2.21.0->tensorboard<2.11,>=2.10->tensorflow
<2.11,>=2.10.0->tensorflow_text) (2.10)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in
/usr/local/lib/python3.7/dist-packages (from requests<3,>=2.21.0->tensorbo
```

```
ard<2.11,>=2.10->tensorflow<2.11,>=2.10.0->tensorflow_text) (1.24.3)
Requirement already satisfied: oauthlib>=3.0.0 in /usr/local/lib/python3.
7/dist-packages (from requests-oauthlib>=0.7.0->google-auth-oauthlib<0.5,>
=0.4.1->tensorboard<2.11,>=2.10->tensorflow<2.11,>=2.10.0->tensorflow_tex
t) (3.2.0)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in /usr/local/lib/
python3.7/dist-packages (from packaging->tensorflow<2.11,>=2.10.0->tensorf
low_text) (3.0.9)
```

In [ ]:
```
!pip install -q -U "tensorflow-text==2.8.*"    # To install specifically 2.8
```

```
|████████████████████████████████| 4.9 MB 8.9 MB/s
|████████████████████████████████| 497.9 MB 25 kB/s
|████████████████████████████████| 5.8 MB 37.1 MB/s
|████████████████████████████████| 462 kB 42.5 MB/s
|████████████████████████████████| 1.4 MB 46.1 MB/s
```

In [ ]:
```
!pip install -q -U tf-models-official==2.7.0    # A TensorFlow Model is a Ne
```

```
|████████████████████████████████| 1.8 MB 6.8 MB/s
|████████████████████████████████| 238 kB 55.1 MB/s
|████████████████████████████████| 43 kB 1.7 MB/s
|████████████████████████████████| 99 kB 7.3 MB/s
|████████████████████████████████| 1.1 MB 50.0 MB/s
|████████████████████████████████| 116 kB 59.5 MB/s
|████████████████████████████████| 1.3 MB 65.4 MB/s
|████████████████████████████████| 352 kB 41.8 MB/s
  Building wheel for py-cpuinfo (setup.py) ... done
  Building wheel for seqeval (setup.py) ... done
```

In [ ]:
```
!pip install -U tfds-nightly                    # A library of datasets reac
```

```
Looking in indexes: https://pypi.org/simple, (https://pypi.org/simple,) ht
tps://us-python.pkg.dev/colab-wheels/public/simple/ (https://us-python.pk
g.dev/colab-wheels/public/simple/)
Collecting tfds-nightly
  Downloading tfds_nightly-4.6.0.dev202209250045-py3-none-any.whl (4.7 MB)
     |████████████████████████████████| 4.7 MB 7.6 MB/s
Requirement already satisfied: absl-py in /usr/local/lib/python3.7/dist-pa
ckages (from tfds-nightly) (1.2.0)
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packag
es (from tfds-nightly) (1.15.0)
Requirement already satisfied: tqdm in /usr/local/lib/python3.7/dist-packa
ges (from tfds-nightly) (4.64.1)
Requirement already satisfied: etils[epath] in /usr/local/lib/python3.7/di
st-packages (from tfds-nightly) (0.7.1)
Requirement already satisfied: tensorflow-metadata in /usr/local/lib/pytho
n3.7/dist-packages (from tfds-nightly) (1.10.0)
Requirement already satisfied: toml in /usr/local/lib/python3.7/dist-packa
ges (from tfds-nightly) (0.10.2)
Requirement already satisfied: dill in /usr/local/lib/python3.7/dist-packa
ges (from tfds-nightly) (0.3.5.1)
Requirement already satisfied: promise in /usr/local/lib/python3.7/dist-pa
ckages (from tfds-nightly) (2.3)
Requirement already satisfied: importlib-resources in /usr/local/lib/pytho
n3.7/dist-packages (from tfds-nightly) (5.9.0)
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-pack
ages (from tfds-nightly) (1.21.6)
Requirement already satisfied: termcolor in /usr/local/lib/python3.7/dist-
packages (from tfds-nightly) (1.1.0)
Requirement already satisfied: protobuf>=3.12.2 in /usr/local/lib/python3.
7/dist-packages (from tfds-nightly) (3.17.3)
Requirement already satisfied: requests>=2.19.0 in /usr/local/lib/python3.
7/dist-packages (from tfds-nightly) (2.23.0)
Requirement already satisfied: typing-extensions in /usr/local/lib/python
3.7/dist-packages (from tfds-nightly) (4.1.1)
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python
3.7/dist-packages (from requests>=2.19.0->tfds-nightly) (3.0.4)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/di
st-packages (from requests>=2.19.0->tfds-nightly) (2.10)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in
/usr/local/lib/python3.7/dist-packages (from requests>=2.19.0->tfds-nightl
y) (1.24.3)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python
3.7/dist-packages (from requests>=2.19.0->tfds-nightly) (2022.6.15)
Requirement already satisfied: zipp in /usr/local/lib/python3.7/dist-packa
ges (from etils[epath]->tfds-nightly) (3.8.1)
Requirement already satisfied: googleapis-common-protos<2,>=1.52.0 in /us
r/local/lib/python3.7/dist-packages (from tensorflow-metadata->tfds-nightl
y) (1.56.4)
Installing collected packages: tfds-nightly
Successfully installed tfds-nightly-4.6.0.dev202209250045
```

Import Libraries

In [ ]:
```
import pandas as pd     # Widely used for data science/data analysis and mach
import numpy as np      # Aims to provide an array object that is up to 50x f
import os, glob         # Used to return all file paths that match a specific
import os, os.path      # For merging, normalizing and retrieving path names
import seaborn as sns   # Seaborn helps you explore and understand your data
```

Check, did the drive connect successfully?

Before starting, we will check that our data can be read easily or if there is any error.

In [ ]:
```
DIR = '/content/aclImdb/train/unsup'
print(len([name for name in os.listdir(DIR) if os.path.isfile(os.path.join(D
```

```
50000
```

Output shows that there are 50000 files in the unsup folder. That means our dataset can be readable and drive is connected successfully and there are not any issues with file path or anything else.

Procedure/Scenario

Basically, all the files (which are provided to us) are in text form and are present in different folders. The core dataset contains 25,000 (positive + negative) files are for training, another 25,000 (positive + negative) files are for testing purposes, and the remaining 50,000 are for unsupervised learning.

The requirement is that the size of the dataset for training should be 5000 rows and the size of the testing dataset should be 500 rows. So we'll make dataframes out of data files. We'll take 2500 positive files and 2500 negative files, label them (pos =1 and neg =0), and merge them into a single dataframe to get a labelled dataframe with 5000 rows, which will be our training dataset, and the same procedure for testing dataset.

In the last step, we will train a selector transformer on a training dataset and for testing we will use a testing dataset. After that, we will evaluate the transformer by taking different values of epochs.

Prepration of Training Data

As our training dataset will consist of both positive and negative labelled data, we will first take 2500 positive labelled data and then take 2500 negative labelled data and perform preprocessing on them.

Positive Training Dataset

Here we store the path of our positive training data in a variable and initialise a list named of files.

In [ ]:
```
# Define relative path to folder containing the text files

files_folder = "/content/aclImdb/train/pos"
files = []
```

Now, by iterating a for loop, we extrect our (.txt format) filed data using the glob function and store it in the list in the form of a dataframe for each file data and name the column as 'text'.

In [ ]:
```
# Create a dataframe list by using a list comprehension

files = [pd.read_csv(file, delimiter='\t', names =['text'] ) for file in glo
```

Here we will combine the whole list of dataframes and make a single dataframe.

In [ ]:
```
# Concatenate the list of DataFrames into one
df_pos = pd.concat(files)
```

As the upper dataframe belongs to the positive class, we assigned it the label '1'. For this, we first made an array with a size of 12500 and then assigned it to the dataframe.

In [ ]:
```
b = np.ones(12500, dtype = int)

df_pos = df_pos.assign(label=b)
df_pos.head()
```

Out[17]:

	text	label
0	Unfortunately for myself - I stumbled onto thi...	1
0	I've seen this film literally over 100 times.....	1
0	It seems to be a perfect day for swimming. A n...	1
0	For fans of Troma or the Cyberpunk genre mixed...	1
0	This is a refreshing, enjoyable movie. If you ...	1

Here we computed the total number of rows and columns just to show the characteristics of the dataframe.

In [ ]:
```
# computing number of rows
rows = len(df_pos.axes[0])

# computing number of columns
cols = len(df_pos.axes[1])

# Simply print them
print('\n')
print("Number of Rows: ", rows)
print("Number of Columns: ", cols)
df_pos.head()
```

```
Number of Rows:  12500
Number of Columns:  2
```

Out[18]:

	text	label
0	Unfortunately for myself - I stumbled onto thi...	1
0	I've seen this film literally over 100 times.....	1
0	It seems to be a perfect day for swimming. A n...	1
0	For fans of Troma or the Cyberpunk genre mixed...	1
0	This is a refreshing, enjoyable movie. If you ...	1

As we know, we want a total of only 5000 (2500 pos + 2500 neg) rows for our training dataset, so, we split our dataframe and make a new dataframe which has a size of (2500,2).

In [ ]:
```
# splitting dataframe by row index

df_positive = df_pos.iloc[:2500,:]
print("Shape of new dataframes - {}".format(df_positive.shape))
df_positive
```

```
Shape of new dataframes - (2500, 2)
```

Out[19]:

	text	label
0	Unfortunately for myself - I stumbled onto thi...	1
0	I've seen this film literally over 100 times.....	1
0	It seems to be a perfect day for swimming. A n...	1
0	For fans of Troma or the Cyberpunk genre mixed...	1
0	This is a refreshing, enjoyable movie. If you ...	1
...	...	...
0	First things first, the female lead is too gor...	1
0	As a baseball die-hard, this movie goes contra...	1
0	This movie really rocks! Jeff Wincott is terri...	1
0	This is an EXCELLENT example of early Bette Da...	1
0	Episode two of season one is a delightful holi...	1

2500 rows × 2 columns

Negative Training Dataset

Now the above same factors are for the preparation of the negative dataset. we store the path of our negative training data in a variable and initialise a list named of files.

In [ ]:
```
# Define relative path to folder containing the text files

files_folder = "/content/aclImdb/train/neg"
files = []
```

Now, by iterating a for loop, we extrect our (.txt format) filed data using the glob function and store it in the list in the form of a dataframe for each file data and name the column as text.

In [ ]:
```
# Create a dataframe list by using a list comprehension

files = [pd.read_csv(file, delimiter='\t', names =['text'] ) for file in glo
```

Here we will combine the whole list of dataframes and make a single dataframe.

In [ ]:
```
# Concatenate the list of DataFrames into one

df_neg = pd.concat(files)
```

As the upper dataframe belongs to the negative class, we assigned it the label '0'. For this, we first made an array with a size of 12500 and then assigned it to the dataframe.

In [ ]:
```
b = [0] * 12500

df_neg = df_pos.assign(label=b)
df_neg.head()
```

Out[23]:

	text	label
0	Unfortunately for myself - I stumbled onto thi...	0
0	I've seen this film literally over 100 times.....	0
0	It seems to be a perfect day for swimming. A n...	0
0	For fans of Troma or the Cyberpunk genre mixed...	0
0	This is a refreshing, enjoyable movie. If you ...	0

Here we computed the total number of rows and columns just to show the characteristics of the neagtive labeled dataframe.

In [ ]:
```
# computing number of rows
rows = len(df_neg.axes[0])

# computing number of columns
cols = len(df_neg.axes[1])

# Simply print

print('\n')
print("Number of Rows: ", rows)
print("Number of Columns: ", cols)
df_neg.head()
```

```
Number of Rows:  12500
Number of Columns:  2
```

Out[24]:

	text	label
0	Unfortunately for myself - I stumbled onto thi...	0
0	I've seen this film literally over 100 times.....	0
0	It seems to be a perfect day for swimming. A n...	0
0	For fans of Troma or the Cyberpunk genre mixed...	0
0	This is a refreshing, enjoyable movie. If you ...	0

As we know, we want a total of only 5000 (2500 pos + 2500 neg) rows for our training dataset, so, we split our dataframe by using iloc and make a new dataframe which has a size of (2500,2).

In [ ]:
```
# splitting dataframe by row index

df_negative = df_neg.iloc[:2500,:]
print("Shape of new dataframes - {}".format(df_negative.shape))
df_negative
```

```
Shape of new dataframes - (2500, 2)
```

Out[25]:

	text	label
0	Unfortunately for myself - I stumbled onto thi...	0
0	I've seen this film literally over 100 times.....	0
0	It seems to be a perfect day for swimming. A n...	0
0	For fans of Troma or the Cyberpunk genre mixed...	0
0	This is a refreshing, enjoyable movie. If you ...	0
...	...	...
0	First things first, the female lead is too gor...	0
0	As a baseball die-hard, this movie goes contra...	0
0	This movie really rocks! Jeff Wincott is terri...	0
0	This is an EXCELLENT example of early Bette Da...	0
0	Episode two of season one is a delightful holi...	0

2500 rows × 2 columns

Combines Both datasets

Now we have our positive and negative labelled dataframes, so we will merge them into a single dataframe by using the concat function of the pandas library. Then in the results we will get our training dataset size of (5000,2).

In [ ]:
```
frames = [df_positive, df_negative]

train = pd.concat(frames)
print("Shape of new train dataframe - {}".format(train.shape))
display(train)
```

```
Shape of new train dataframe - (5000, 2)
```

	text	label
0	Unfortunately for myself - I stumbled onto thi...	1
0	I've seen this film literally over 100 times.....	1
0	It seems to be a perfect day for swimming. A n...	1
0	For fans of Troma or the Cyberpunk genre mixed...	1
0	This is a refreshing, enjoyable movie. If you ...	1
...	...	...
0	First things first, the female lead is too gor...	0
0	As a baseball die-hard, this movie goes contra...	0
0	This movie really rocks! Jeff Wincott is terri...	0
0	This is an EXCELLENT example of early Bette Da...	0
0	Episode two of season one is a delightful holi...	0

5000 rows × 2 columns

Preprocessing on Training Dataset

For text processing first we import spacy library because spacy is designed specifically for production use and helps you build applications that process and “understand” large volumes of text.

In [ ]:
```
import spacy
```

Now, by using spacy, we load "en_core_web_sm", which is a small English pipeline trained on written web text that includes vocabulary, syntax, and entities.

Then we make a new column named "tokens" in our training dataset in which we store our text data after tokenization.

In [ ]:
```
nlp = spacy.load("en_core_web_sm")
train1 = train.copy()
train1["tokens"] = train["text"].apply(lambda x: nlp.tokenizer(x))

train1.head()
```

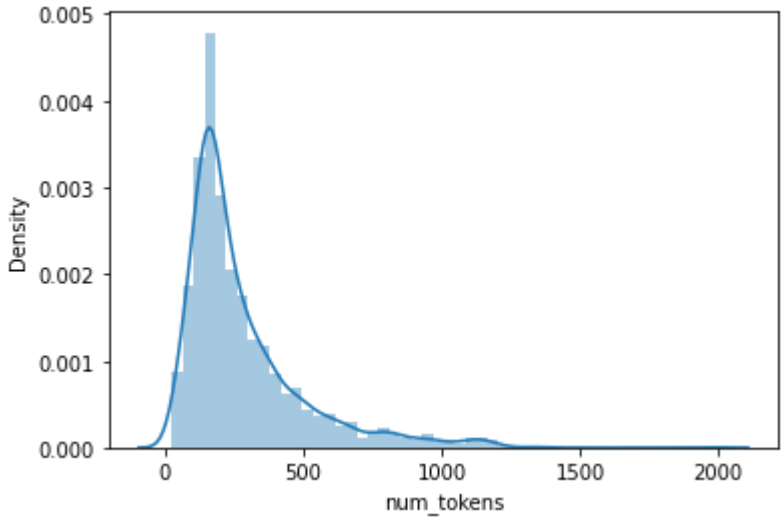
Out[28]:

	text	label	tokens
0	Unfortunately for myself - I stumbled onto thi...	1	(Unfortunately, for, myself, -, I, stumbled, o...
0	I've seen this film literally over 100 times.....	1	(I, 've, seen, this, film, literally, over, 10...
0	It seems to be a perfect day for swimming. A n...	1	(It, seems, to, be, a, perfect, day, for, swim...
0	For fans of Troma or the Cyberpunk genre mixed...	1	(For, fans, of, Troma, or, the, Cyberpunk, gen...
0	This is a refreshing, enjoyable movie. If you ...	1	(This, is, a, refreshing, ,, enjoyable, movie,...

Here we create a new column named "num_tokens", store the sum of tokens in each document, and make a histogram by using the seaborn library to visualise the total number of tokens per movie review.

In [ ]:
```
# Sum the number of tokens in each Doc

train1['num_tokens'] = [len(token) for token in train1.tokens]

# Visualize histogram of tokens per movie review

g = sns.distplot(train1.num_tokens)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: Futu
reWarning: `distplot` is a deprecated function and will be removed in a fu
ture version. Please adapt your code to use either `displot` (a figure-lev
el function with similar flexibility) or `histplot` (an axes-level functio
n for histograms).
  warnings.warn(msg, FutureWarning)
```



Here we have written three functions:

1. First is for making a spacy document in which we will join the words into a string.
2. In the second function, we are actually filtering the tokens in such a way that it will skip stopwords and only pick the non_stopwords tokens. Basically, we are removing stopwords.

3. The third one is for lemmatization, which we use . For each token, use lemma_keyword (access each token with a for loop).

In [ ]:
```
def to_doc(words:tuple) -> spacy.tokens.Doc:
    # Create SpaCy documents by joining the words into a string
    return nlp(' '.join(words))

def remove_stops(doc) -> list:
    # Filter out stop words by using the `token.is_stop` attribute
    return [token.text for token in doc if not token.is_stop]

def lemmatize(doc) -> list:
    # Take the `token.lemma_` of each non-stop word
    return [token.lemma_ for token in doc if not token.is_stop]
```

Here we are making a copy of our training dataset as it will be some kind of backup for us.

now we will map above three functions on our copy of training dataset by using map function and it will return three new columns doc, removed_stops and lemmatized.

In [ ]:
```
# create documents for all tuples of tokens
#docs = list(map(to_doc, train3.text))

train3 = train1.copy()
# apply removing stop words to all
train3['removed_stops'] = list(map(remove_stops, train3.tokens))

# apply lemmatization to all
train3['lemmatized'] = list(map(lemmatize, train3.tokens))
```

Showing the top 5 rows of are dataset

In [ ]:
```
train3.head()
```

Out[32]:

	text	label	tokens	num_tokens	removed_stops	lemmatized
0	Unfortunately for myself - I stumbled onto thi...	1	(Unfortunately, for, myself, -, I, stumbled, o...	166	[Unfortunately, -, stumbled, late, lifetime, ...	[, ,...
0	I've seen this film literally over 100 times.....	1	(I, 've, seen, this, film, literally, over, 10...	99	[seen, film, literally, 100, times, ..., absol...	[, ,...
0	It seems to be a perfect day for swimming. A n...	1	(It, seems, to, be, a, perfect, day, for, swim...	174	[perfect, day, swimming, ., normal, family, wa...	[, ,...
0	For fans of Troma or the Cyberpunk genre mixed...	1	(For, fans, of, Troma, or, the, Cyberpunk, gen...	137	[fans, Troma, Cyberpunk, genre, mixed, little,...	[, ,...
0	This is a refreshing, enjoyable movie. If you ...	1	(This, is, a, refreshing, ,, enjoyable, movie,...	68	[refreshing, ,, enjoyable, movie, ., enjoyed, ...	[, ,...

Actually, the upper to_doc function was taking too much time to complete its execution, so if we skip that function, then the result of lematization of words also changed.

Another method for cleaning the data

We have chosen another method for this purpose. We will first import re, which is actually a regular expression (or RE), that specifies a set of strings that match it, and then we will compile special characters and digits by using re.

In [ ]:
```
import re
spec_chara = re.compile('[/(){}\[\]\|@,;]')
ext_sym = re.compile('[^0-9a-z #+_]')
```

Than we will import Stop words library from spacy english language pakage and load "en_core_web_sm".

In [ ]:
```
import spacy
from spacy.lang.en.stop_words import STOP_WORDS
nlp = spacy.load("en_core_web_sm", disable=['parser', 'tagger', 'ner'])
stops = STOP_WORDS
```

Here we create a normalizer function which will take a dataset as an input, make tokens of ser

At the start, it will convert words to upper case, subract special characters, and then subract symbols and digits. After that, it will perform lamatization and remove stopwords, and lastly, it will return all the cleaned datasets by joining them.

In [ ]:
```
def normalize(comment, lowercase, remove_stopwords):
    if lowercase:
        comment = comment.lower()
        comment = spec_chara.sub(' ', comment)
        comment = ext_sym.sub('', comment)

    comment = nlp(comment)
    lemmatized = list()
    for word in comment:
        lemma = word.lemma_.strip()
        if lemma:
            if not remove_stopwords or (remove_stopwords and lemma not in st
                lemmatized.append(lemma)
    return " ".join(lemmatized)
```

Here we are simply applying the function which we have defined just in the above cell and storing clean text and text without stopwords in the dataframe by making new columns.

In [ ]:
```
# The execution time will be between Five and six minutes.

train4 = train1.copy()
train4['Text_After_Clean'] = train4['text'].apply(normalize, lowercase=True,
train4['stopwords removed'] = train4['text'].apply(normalize, lowercase=True
```

```
/usr/local/lib/python3.7/dist-packages/spacy/pipeline/lemmatizer.py:211: U
serWarning: [W108] The rule-based lemmatizer did not find POS annotation f
or one or more tokens. Check that your pipeline includes components that a
ssign token.pos, typically 'tagger'+'attribute_ruler' or 'morphologizer'.
  warnings.warn(Warnings.W108)
```

Showing the top 5 rows of our cleaned dataset

In [ ]:
```
train4.head()
```

Out[33]:

	text	label	tokens	num_tokens	Text_After_Clean	stopwords removed
0	In New York, Andy Hanson (Philip Seymour Hoffm...	1	(In, New, York, ,, Andy, Hanson, (, Philip, Se...	419	in new york andy hanson philip seymour hoffman...	new york andy hanson philip seymour hoffman ad...
0	Jonathan Demme's directorial debut for Roger C...	1	(Jonathan, Demme, 's, directorial, debut, for,...	463	jonathan demmes directorial debut for roger co...	jonathan demmes directorial debut roger corman...
0	I had never seen a silent movie until July 24,...	1	(I, had, never, seen, a, silent, movie, until,...	209	i had never seen a silent movie until july 24 ...	seen silent movie july 24 2005 seen movie mary...
0	This film pulls you in from the get-go because...	1	(This, film, pulls, you, in, from, the, get, -...	335	this film pulls you in from the getgo because ...	film pulls getgo grabs attention acknowledging...
0	Paulie sounds like the most saccharine, lachry...	1	(Paulie, sounds, like, the, most, saccharine, ...	265	paulie sounds like the most saccharine lachrym...	paulie sounds like saccharine lachrymose senti...

We are just going to see the description of the label column.

In [ ]:
```
train4.groupby('label').describe()
```

Out[34]:

	num_tokens							
	count	mean	std	min	25%	50%	75%	max
label								
0	2500.0	273.1892	217.232224	14.0	141.0	197.0	336.25	2789.0
1	2500.0	273.1892	217.232224	14.0	141.0	197.0	336.25	2789.0

Preparation of Testing Data

As our testing dataset will consist of both positive and negative labelled data, we will first take 250 positive labelled data and then take 250 negative labelled data and perform preprocessing on them.

Positive Testing Dataset

Here we store the path of our positive testing data in a variable and initialise a list named of files.

In [ ]:
```
# Define relative path to folder containing the text files

files_folder = "/content/aclImdb/test/pos"
files = []
```

Now, by iterating a for loop, we extrect our (.txt format) filed data using the glob function and store it in the list in the form of a dataframe for each file data and name the column as 'text'.

In [ ]:
```
# Create a dataframe list by using a list comprehension

files = [pd.read_csv(file, delimiter='\t', names =['text'] ) for file in glo
```

Here we will combine the whole list of dataframes and make a single dataframe.

In [ ]:
```
# Concatenate the list of DataFrames into one

df_pos_test = pd.concat(files)
```

As the upper dataframe belongs to the positive class, we assigned it the label '1'. For this, we first made an array with a size of 12500 and then assigned it to the dataframe.

In [ ]:
```
b = np.ones(12500, dtype = int)

df_pos_test = df_pos.assign(label=b)
df_pos_test.head()
```

Out[39]:

	text	label
0	Unfortunately for myself - I stumbled onto thi...	1
0	I've seen this film literally over 100 times.....	1
0	It seems to be a perfect day for swimming. A n...	1
0	For fans of Troma or the Cyberpunk genre mixed...	1
0	This is a refreshing, enjoyable movie. If you ...	1

Here we computed the total number of rows and columns just to show the characteristics of the dataframe.

In [ ]:
```
# computing number of rows
rows = len(df_pos_test.axes[0])

# computing number of columns
cols = len(df_pos_test.axes[1])

print('\n')
print("Number of Rows: ", rows)
print("Number of Columns: ", cols)
df_pos_test.head()
```

```
Number of Rows:  12500
Number of Columns:  2
```

Out[40]:

	text	label
0	Unfortunately for myself - I stumbled onto thi...	1
0	I've seen this film literally over 100 times.....	1
0	It seems to be a perfect day for swimming. A n...	1
0	For fans of Troma or the Cyberpunk genre mixed...	1
0	This is a refreshing, enjoyable movie. If you ...	1

As we know, we want a total of only 500 (250 pos + 2500 neg) rows for our testing dataset, so, we split our dataframe and make a new dataframe which has a size of (250,2).

In [ ]:
```
# splitting dataframe by row index
df_positive_test = df_pos_test.iloc[:250,:]
print("Shape of new dataframes - {}".format(df_positive_test.shape))
df_positive_test
```

```
Shape of new dataframes - (250, 2)
```

Out[41]:

	text	label
0	Unfortunately for myself - I stumbled onto thi...	1
0	I've seen this film literally over 100 times.....	1
0	It seems to be a perfect day for swimming. A n...	1
0	For fans of Troma or the Cyberpunk genre mixed...	1
0	This is a refreshing, enjoyable movie. If you ...	1
...	...	...
0	This is a classic British comedy-thriller I ha...	1
0	Once you pick your jaw up from off the floor f...	1
0	This was the first "Walking Tall" movie I saw,...	1
0	Though I did not begin to read the "Classics" ...	1
0	Did Sandra (yes, she must have) know we would ...	1

250 rows × 2 columns

Negative Testing Dataset

Now the above same factors are for the preparation of the testing dataset. we store the path of our negative testing data in a variable and initialise a list named of files.

In [ ]:
```
# Define relative path to folder containing the text files

files_folder = "/content/aclImdb/test/neg"
files = []
```

Now, by iterating a for loop, we extrect our (.txt format) filed data using the glob function and store it in the list in the form of a dataframe for each file data and name the column as text.

In [ ]:
```
# Create a dataframe list by using a list comprehension

files = [pd.read_csv(file, delimiter='\t', names =['text'] ) for file in glo
```

Here we will combine the whole list of dataframes and make a single dataframe.

In [ ]:
```
# Concatenate the list of DataFrames into one

df_neg_test = pd.concat(files)
```

As the upper dataframe belongs to the negative class, we assigned it the label '0'. For this, we first made an array with a size of 12500 and then assigned it to the dataframe.

In [ ]:
```
b = [0] * 12500

df_neg_test = df_pos.assign(label=b)
df_neg_test.head()
```

Out[45]:

	text	label
0	Unfortunately for myself - I stumbled onto thi...	0
0	I've seen this film literally over 100 times.....	0
0	It seems to be a perfect day for swimming. A n...	0
0	For fans of Troma or the Cyberpunk genre mixed...	0
0	This is a refreshing, enjoyable movie. If you ...	0

Here we computed the total number of rows and columns just to show the characteristics of the neagtive labeled dataframe.

In [ ]:
```
# computing number of rows
rows = len(df_neg_test.axes[0])

# computing number of columns
cols = len(df_neg_test.axes[1])

print('\n')
print("Number of Rows: ", rows)
print("Number of Columns: ", cols)
df_neg_test.head()
```

```
Number of Rows:  12500
Number of Columns:  2
```

Out[46]:

	text	label
0	Unfortunately for myself - I stumbled onto thi...	0
0	I've seen this film literally over 100 times.....	0
0	It seems to be a perfect day for swimming. A n...	0
0	For fans of Troma or the Cyberpunk genre mixed...	0
0	This is a refreshing, enjoyable movie. If you ...	0

As we know, we want a total of only 500 (250 pos + 250 neg) rows for our testing dataset, so, we split our dataframe by using iloc and make a new dataframe which has a size of (250,2).

In [ ]:
```
# splitting dataframe by row index
df_negative_test = df_neg_test.iloc[:250,:]
print("Shape of new dataframes - {}".format(df_negative_test.shape))
df_negative_test
```

```
Shape of new dataframes - (250, 2)
```

Out[47]:

	text	label
0	Unfortunately for myself - I stumbled onto thi...	0
0	I've seen this film literally over 100 times.....	0
0	It seems to be a perfect day for swimming. A n...	0
0	For fans of Troma or the Cyberpunk genre mixed...	0
0	This is a refreshing, enjoyable movie. If you ...	0
...	...	...
0	This is a classic British comedy-thriller I ha...	0
0	Once you pick your jaw up from off the floor f...	0
0	This was the first "Walking Tall" movie I saw,...	0
0	Though I did not begin to read the "Classics" ...	0
0	Did Sandra (yes, she must have) know we would ...	0

250 rows × 2 columns

Combines Both datasets

Now we have our positive and negative labelled dataframes, so we will merge them into a single dataframe by using the concat function of the pandas library. Then in the results we will get our testing dataset size of (500,2).

In [ ]:
```
frames = [df_positive_test, df_negative_test]

test = pd.concat(frames)
print("Shape of new train dataframe - {}".format(test.shape))
display(test)
```

```
Shape of new train dataframe - (500, 2)
```

	text	label
0	Unfortunately for myself - I stumbled onto thi...	1
0	I've seen this film literally over 100 times.....	1
0	It seems to be a perfect day for swimming. A n...	1
0	For fans of Troma or the Cyberpunk genre mixed...	1
0	This is a refreshing, enjoyable movie. If you ...	1
...	...	...
0	This is a classic British comedy-thriller I ha...	0
0	Once you pick your jaw up from off the floor f...	0
0	This was the first "Walking Tall" movie I saw,...	0
0	Though I did not begin to read the "Classics" ...	0
0	Did Sandra (yes, she must have) know we would ...	0

500 rows × 2 columns

Preprocessing on Training Dataset

By using spacy, we load "en_core_web_sm", which is a small English pipeline trained on written web text that includes vocabulary, syntax, and entities.

Then we make a new column named "tokens" in our training dataset in which we store our text data after tokenization.

In [ ]:
```
nlp = spacy.load("en_core_web_sm")
test1 = test.copy()
test1["tokens"] = test["text"].apply(lambda x: nlp.tokenizer(x))

test1.head()
```

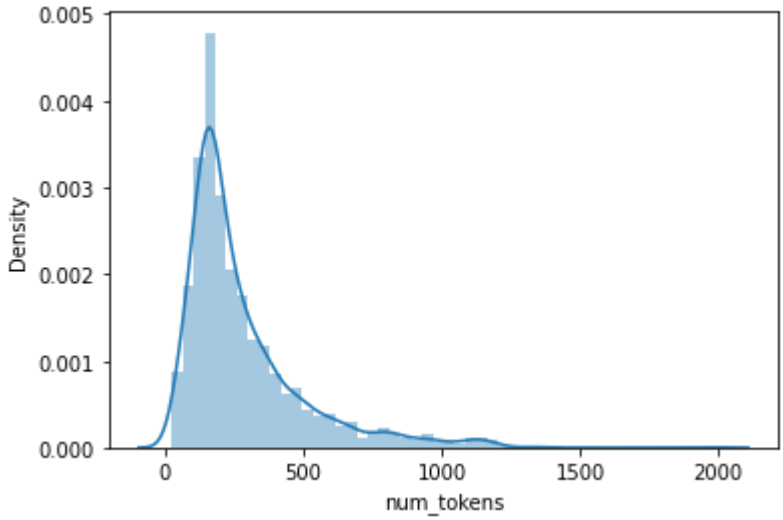
Out[49]:

	text	label	tokens
0	Unfortunately for myself - I stumbled onto thi...	1	(Unfortunately, for, myself, -, I, stumbled, o...
0	I've seen this film literally over 100 times.....	1	(I, 've, seen, this, film, literally, over, 10...
0	It seems to be a perfect day for swimming. A n...	1	(It, seems, to, be, a, perfect, day, for, swim...
0	For fans of Troma or the Cyberpunk genre mixed...	1	(For, fans, of, Troma, or, the, Cyberpunk, gen...
0	This is a refreshing, enjoyable movie. If you ...	1	(This, is, a, refreshing, ,, enjoyable, movie,...

Here we create a new column named "num_tokens", store the sum of tokens in each document, and make a histogram by using the seaborn library to visualise the total number of tokens per movie review.

In [ ]:
```
# Sum the number of tokens in each Doc

train1['num_tokens'] = [len(token) for token in train1.tokens]

# Visualize histogram of tokens per movie review

g = sns.distplot(train1.num_tokens)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: Futu
reWarning: `distplot` is a deprecated function and will be removed in a fu
ture version. Please adapt your code to use either `displot` (a figure-lev
el function with similar flexibility) or `histplot` (an axes-level functio
n for histograms).
  warnings.warn(msg, FutureWarning)
```



Here we have written three functions:

First is for making a spacy document in which we will join the words into a string. In the second function, we are actually filtering the tokens in such a way that it will skip stopwords and only pick the non_stopwords tokens. Basically, we are removing stopwords. The third

one is for lemmatization, which we use . For each token, use lemma_keyword (access each

In [ ]:
```
def to_doc(words:tuple) -> spacy.tokens.Doc:
    # Create SpaCy documents by joining the words into a string
    return nlp(' '.join(words))

def remove_stops(doc) -> list:
    # Filter out stop words by using the `token.is_stop` attribute
    return [token.text for token in doc if not token.is_stop]

def lemmatize(doc) -> list:
    # Take the `token.lemma_` of each non-stop word
    return [token.lemma_ for token in doc if not token.is_stop]
```

Here we are making a copy of our training dataset as it will be some kind of backup for us.

now we will map above three functions on our copy of training dataset by using map function and it will return three new columns doc, removed_stops and lemmatized.

In [ ]:
```
# create documents for all tuples of tokens
#docs = list(map(to_doc, train3.text))

test3 = test1.copy()
# apply removing stop words to all
test3['removed_stops'] = list(map(remove_stops, test3.tokens))

# apply lemmatization to all
test3['lemmatized'] = list(map(lemmatize, test3.tokens))
```

Showing the top 5 rows of are dataset

In [ ]:
```
test3.head()
```

Out[53]:

	text	label	tokens	removed_stops	lemmatized
0	Unfortunately for myself - I stumbled onto thi...	1	(Unfortunately, for, myself, -, I, stumbled, o...	[Unfortunately, -, stumbled, late, lifetime, ...	[, ,...
0	I've seen this film literally over 100 times.....	1	(I, 've, seen, this, film, literally, over, 10...	[seen, film, literally, 100, times, ..., absol...	[, ,...
0	It seems to be a perfect day for swimming. A n...	1	(It, seems, to, be, a, perfect, day, for, swim...	[perfect, day, swimming, ., normal, family, wa...	[, ,...
0	For fans of Troma or the Cyberpunk genre mixed...	1	(For, fans, of, Troma, or, the, Cyberpunk, gen...	[fans, Troma, Cyberpunk, genre, mixed, little,...	[, ,...
0	This is a refreshing, enjoyable movie. If you ...	1	(This, is, a, refreshing, ,, enjoyable, movie,...	[refreshing, ,, enjoyable, movie, ., enjoyed, ...	[, ,...

Actually, the upper to_doc function was taking too much time to complete its execution, so if we skip that function, then the result of lematization of words also changed.

Another method for cleaning the data

So we have chosen another method for this purpose. We will first import re, which is actually a regular expression (or RE), that specifies a set of strings that match it, and then we will compile special characters and digits by using re.

In [ ]:
```
import re
spec_chara = re.compile('[/(){}\[\]\|@,;]')
ext_sym = re.compile('[^0-9a-z #+_]')
```

Than we will import Stop words library from spacy english language pakage and load "en_core_web_sm".

In [ ]:
```
import spacy
from spacy.lang.en.stop_words import STOP_WORDS
nlp = spacy.load("en_core_web_sm", disable=['parser', 'tagger', 'ner'])
stops = STOP_WORDS
```

Here we create a normalizer function which will take a dataset as an input, make tokens of sentences, remove stopwords, and then perform lamatization.

At the start, it will convert words to upper case, subract special characters, and then subract symbols and digits. After that, it will perform lamatization and remove stopwords, and lastly, it will return all the cleaned datasets by joining them.

In [ ]:
```
def normalize(comment, lowercase, remove_stopwords):
    if lowercase:
        comment = comment.lower()
        comment = spec_chara.sub(' ', comment)
        comment = ext_sym.sub('', comment)

    comment = nlp(comment)
    lemmatized = list()
    for word in comment:
        lemma = word.lemma_.strip()
        if lemma:
            if not remove_stopwords or (remove_stopwords and lemma not in st
                lemmatized.append(lemma)
    return " ".join(lemmatized)
```

Here we are simply applying the function which we have defined just in the above cell and storing clean text and text without stopwords in the dataframe by making new columns.

In [ ]:
```
# The execution time will be between four and five minutes.

test4 = test1.copy()
test4['Text_After_Clean'] = test4['text'].apply(normalize, lowercase=True, r
test4['stopwords removed'] = test4['text'].apply(normalize, lowercase=True,
```

```
/usr/local/lib/python3.7/dist-packages/spacy/pipeline/lemmatizer.py:211: U
serWarning: [W108] The rule-based lemmatizer did not find POS annotation f
or one or more tokens. Check that your pipeline includes components that a
ssign token.pos, typically 'tagger'+'attribute_ruler' or 'morphologizer'.
  warnings.warn(Warnings.W108)
```

Showing the top 5 rows of our cleaned dataset

In [ ]:
```
train4.head()
```

Out[57]:

	text	label	tokens	num_tokens	Text_After_Clean	stopwords removed
0	In New York, Andy Hanson (Philip Seymour Hoffm...	1	(In, New, York, ,, Andy, Hanson, (, Philip, Se...	419	in new york andy hanson philip seymour hoffman...	new york andy hanson philip seymour hoffman ad...
0	Jonathan Demme's directorial debut for Roger C...	1	(Jonathan, Demme, 's, directorial, debut, for,...	463	jonathan demmes directorial debut for roger co...	jonathan demmes directorial debut roger corman...
0	I had never seen a silent movie until July 24,...	1	(I, had, never, seen, a, silent, movie, until,...	209	i had never seen a silent movie until july 24 ...	seen silent movie july 24 2005 seen movie mary...
0	This film pulls you in from the get-go because...	1	(This, film, pulls, you, in, from, the, get, -...	335	this film pulls you in from the getgo because ...	film pulls getgo grabs attention acknowledging...
0	Paulie sounds like the most saccharine, lachry...	1	(Paulie, sounds, like, the, most, saccharine, ...	265	paulie sounds like the most saccharine lachrym...	paulie sounds like saccharine lachrymose senti...

We are just going to see the description of the label column.

In [ ]:
```
test4.groupby('label').describe()
```

Out[58]:

	text				tokens				Text_After_C			
	count	unique	top	freq	count	unique	top	freq	count	unique	top	
label												
0	250	250	In New York, Andy Hanson (Philip Seymour Hoffm...	1	250	250	(In, New, York, ,, Andy, Hanson, (, Philip, Se...	1	250	250	in new york andy hanson philip seymour hoffman...
1	250	250	In New York, Andy Hanson (Philip Seymour Hoffm...	1	250	250	(In, New, York, ,, Andy, Hanson, (, Philip, Se...	1	250	250	in new york andy hanson philip seymour hoffman...

Unsupervised Learning Dataset

Here we store the path of data (which will used for unsupervised learning) in a variable and initialise a list named of files.

In [ ]:
```
# Define relative path to folder containing the text files

files_folder = "/content/aclImdb/train/unsup"
files = []
```

Now, by iterating a for loop, we extrect our (.txt format) filed data using the glob function and store it in the list in the form of a dataframe for each file data and name the column as 'text'.

In [ ]:
```
# Create a dataframe list by using a list comprehension
# The execution time will be between half and one minute.

files = [pd.read_csv(file, delimiter='\t', names =['text'] ) for file in glo
```

Here we will combine the whole list of dataframes and make a single dataframe.

In [ ]:
```
# Concatenate the list of DataFrames into one

df_unsup = pd.concat(files)
```

Here we computed the total number of rows and columns just to show the characteristics of the Un labeled dataframe.

In [ ]:
```
# computing number of rows
rows = len(df_unsup.axes[0])

# computing number of columns
cols = len(df_unsup.axes[1])

# Simply print

print('\n')
print("Number of Rows: ", rows)
print("Number of Columns: ", cols)
df_unsup.head()
```

```
Number of Rows:  50000
Number of Columns:  1
```

Out[62]:

	text
0	Heartstopper wasn't to bad for a low budget B ...
0	Originally made for the cinema, this film was ...
0	The Education of Charlie Banks was an excellen...
0	I know I'm going against the grain here, becau...
0	In this movie director Stephen Frears tries to...

We will take only 5000 rows to reduce the execution time.

In [ ]:
```
# splitting dataframe by row index

df_un = df_unsup.iloc[:5000,:]
print("Shape of new dataframes - {}".format(df_un.shape))
df_un
```

```
Shape of new dataframes - (5000, 1)
```

Out[63]:

	text
0	Heartstopper wasn't to bad for a low budget B ...
0	Originally made for the cinema, this film was ...
0	The Education of Charlie Banks was an excellen...
0	I know I'm going against the grain here, becau...
0	In this movie director Stephen Frears tries to...
...	...
0	At first, I thought this was going to be anoth...
0	WAITING FOR GUFFMAN is my favorite movie of al...
0	i would say that I've watched only the past 3 ...
0	I starting watching this on Comedy Central and...
0	Wow... It's been twelve hours since...

5000 rows × 1 columns

Preprocessing on unsup Dataset

Now, by using spacy, we load "en_core_web_sm", which is a small English pipeline trained on written web text that includes vocabulary, syntax, and entities.

Then we make a new column named "tokens" in our training dataset in which we store our text data after tokenization.

In [ ]:
```
nlp = spacy.load("en_core_web_sm")
df_unsup["tokens"] = df_unsup["text"].apply(lambda x: nlp.tokenizer(x))

df_unsup.head()
```

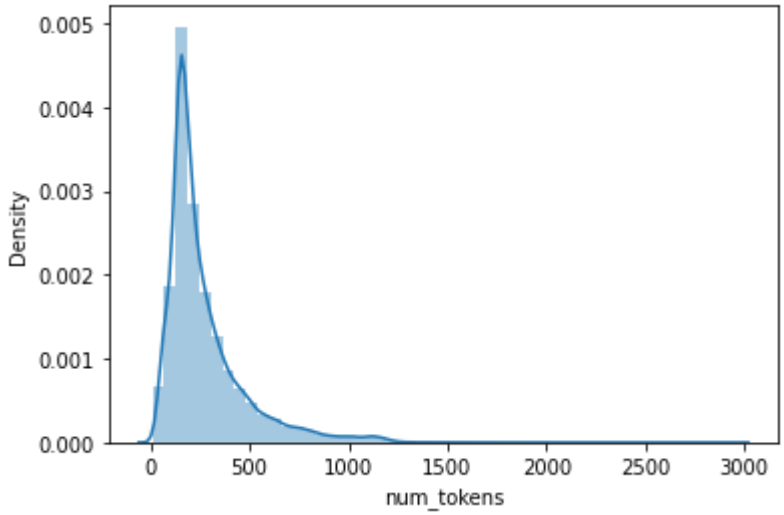
Out[64]:

	text	tokens
0	Heartstopper wasn't to bad for a low budget B ...	(Heartstopper, was, n't, to, bad, for, a, low,...
0	Originally made for the cinema, this film was ...	(Originally, made, for, the, cinema, ,, this, ...
0	The Education of Charlie Banks was an excellen...	(The, Education, of, Charlie, Banks, was, an, ...
0	I know I'm going against the grain here, becau...	(I, know, I, 'm, going, against, the, grain, h...
0	In this movie director Stephen Frears tries to...	(In, this, movie, director, Stephen, Frears, t...

Here we create a new column named "num_tokens", store the sum of tokens in each document, and make a histogram by using the seaborn library to visualise the total number of tokens per movie review.

In [ ]:
```
# Sum the number of tokens in each Doc

df_unsup['num_tokens'] = [len(token) for token in df_unsup.tokens]

# Visualize histogram of tokens per movie review

g = sns.distplot(df_unsup.num_tokens)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: Futu
reWarning: `distplot` is a deprecated function and will be removed in a fu
ture version. Please adapt your code to use either `displot` (a figure-lev
el function with similar flexibility) or `histplot` (an axes-level functio
n for histograms).
  warnings.warn(msg, FutureWarning)
```



We have chosen another method for this purpose. We will first import re, which is actually a regular expression (or RE), that specifies a set of strings that match it, and then we will compile special characters and digits by using re.

In [ ]:
```
import re
spec_chara = re.compile('[/(){}\[\]\|@,;]')
ext_sym = re.compile('[^0-9a-z #+_]')
```

Than we will import Stop words library from spacy english language pakage and load "en_core_web_sm".

In [ ]:
```
import spacy
from spacy.lang.en.stop_words import STOP_WORDS
nlp = spacy.load("en_core_web_sm", disable=['parser', 'tagger', 'ner'])
stops = STOP_WORDS
```

Here we create a normalizer function which will take a dataset as an input, make tokens of sentences, remove stopwords, and then perform lamatization.

At the start, it will convert words to upper case, subract special characters, and then subract symbols and digits. After that, it will perform lamatization and remove stopwords, and lastly, it will return all the cleaned datasets by joining them.

In [ ]:
```
def normalize(comment, lowercase, remove_stopwords):
    if lowercase:
        comment = comment.lower()
        comment = spec_chara.sub(' ', comment)
        comment = ext_sym.sub('', comment)

    comment = nlp(comment)
    lemmatized = list()
    for word in comment:
        lemma = word.lemma_.strip()
        if lemma:
            if not remove_stopwords or (remove_stopwords and lemma not in st
                lemmatized.append(lemma)
    return " ".join(lemmatized)
```

Here we are simply applying the function which we have defined just in the above cell and storing clean text and text without stopwords in the dataframe by making new columns.

In [ ]:
```
# The execution time will be between seven and eight minutes.

df_unsup = df_unsup.copy()
df_unsup['stopwords removed'] = df_unsup['text'].apply(normalize, lowercase=
```

```
/usr/local/lib/python3.7/dist-packages/spacy/pipeline/lemmatizer.py:211: U
serWarning: [W108] The rule-based lemmatizer did not find POS annotation f
or one or more tokens. Check that your pipeline includes components that a
ssign token.pos, typically 'tagger'+'attribute_ruler' or 'morphologizer'.
  warnings.warn(Warnings.W108)
```

Showing the top 5 rows of our cleaned dataset

In [ ]:
```
df_unsup.head()
```

Out[70]:

	text	tokens	num_tokens	stopwords removed
0	Heartstopper wasn't to bad for a low budget B ...	(Heartstopper, was, n't, to, bad, for, a, low,...	139	heartstopper nt bad low budget b slasher movie...
0	Originally made for the cinema, this film was ...	(Originally, made, for, the, cinema, ,, this, ...	137	originally cinema film deemed awful distributo...
0	The Education of Charlie Banks was an excellen...	(The, Education, of, Charlie, Banks, was, an, ...	144	education charlie banks excellent film indepen...
0	I know I'm going against the grain here, becau...	(I, know, I, 'm, going, against, the, grain, h...	465	know m going grain comments favourable movie p...
0	In this movie director Stephen Frears tries to...	(In, this, movie, director, Stephen, Frears, t...	680	movie director stephen frears tries tell dr je...

Transformer

Why DistilBert?

"DistilBERT is a small, fast, cheap and light Transformer model based on the BERT architecture."

Why DistilBert?

1. DistilBERT gives some extraordinary results on some downstream tasks
2. Distil-BERT has 97% of BERT's performance while being trained on half of the parameters of BERT.
3. The DistilBERT transformer is a light, small, and fast version of its bigger BERT version.
4. It has achieved 0.6% less accuracy than BERT while the model is 40% smaller.
5. It has 40% less parameters than BERT and yet 60% faster than it.

Libraries to use

In [ ]:
```
import os                                # to interact with the underlying ope
import tensorflow as tf                  # converts regular python code to a c
import tensorflow_hub as hub             # library for reusable machine learni
import tensorflow_datasets as tfds       # defines a collection of datasets re
import tensorflow_text as text           # provides a collection of text relat
```

Import Bert Algorithm

Now we will import BERT model and get embeding vectors for few sample statements

In [ ]:
```
bert_preprocess = hub.KerasLayer("https://tfhub.dev/tensorflow/bert_en_uncas
bert_encoder = hub.KerasLayer("https://tfhub.dev/tensorflow/bert_en_uncased_
```

Build Model

At the start, we will create our input layer in which we will pass the shape, which encoding bert is passing, and the name of the layer. Then we will use bert_preprocess, which will retune preprocessed text to us, and then, by using bert_encoder, we will get output as a result.

In [ ]:
```
# Bert layers
text_input = tf.keras.layers.Input(shape=(), dtype=tf.string, name='text')
preprocessed_text = bert_preprocess(text_input)
outputs = bert_encoder(preprocessed_text)
```

Now we will creat a drop out layer and pass 0.1 value that means 0.1 percent nurons to stop. And as we know it is an functional model so we will supply nuron laye a input. The input will be the pooled out of output of previous layer.

The second layer we have created it is dense laye, and pass it value 1, because 1 is telling us that movie review is positive or not. And use sigmoid as an activation layer. Also the input of this laye will be the output of privious layer.

At the last, we will create our final model.

In [ ]:
```
# Neural network layers
l = tf.keras.layers.Dropout(0.1, name="dropout")(outputs['pooled_output'])
l = tf.keras.layers.Dense(1, activation='sigmoid', name="output")(l)

# Use inputs and outputs to construct a final model
model = tf.keras.Model(inputs=[text_input], outputs = [l])
```

Now here we are defining during training which parameters (accuracy or precision) will be shown for each epoch.

In model compile the adam is an optimizer, and binary_crossentropy is used because our output is binary(1/0).

In [ ]:
```
METRICS = [
      tf.keras.metrics.BinaryAccuracy(name='accuracy'),
      tf.keras.metrics.Precision(name='precision'),
      tf.keras.metrics.Recall(name='recall')]

model.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=METRICS)
```

Spliting Datasets for training and testing

Here we are making X-train using the feature column and y_train using the label column for training purposes.

In [ ]:
```
X_train = train4['stopwords removed'].copy()
y_train = train4['label'].copy()
```

Here we are making X-test using the feature column and y_test using the label column for testing purposes.

In [ ]:
```
X_test= test4['stopwords removed'].copy()
y_test = test4['label'].copy()
```

Also we can use sklearn pakage for this purpose

In [ ]:
```
#from sklearn.model_selection import train_test_split
#X_train, X_test, y_train, y_test = train_test_split(train4['stopwords remov
```

Training of Transformer

Using 10 Epochs

Here we are training our model using only 10 epochs.

In [ ]:
```
model.fit(X_train, y_train, epochs=10)
```

```
Epoch 1/10
157/157 [==============================] - 50s 321ms/step - loss: 0.7018 -
accuracy: 0.4926 - precision: 0.4925 - recall: 0.4860
Epoch 2/10
157/157 [==============================] - 53s 339ms/step - loss: 0.6990 -
accuracy: 0.4942 - precision: 0.4942 - recall: 0.4932
Epoch 3/10
157/157 [==============================] - 53s 335ms/step - loss: 0.7005 -
accuracy: 0.4974 - precision: 0.4974 - recall: 0.4920
Epoch 4/10
157/157 [==============================] - 53s 335ms/step - loss: 0.7041 -
accuracy: 0.4844 - precision: 0.4838 - recall: 0.4656
Epoch 5/10
157/157 [==============================] - 55s 350ms/step - loss: 0.7017 -
accuracy: 0.4986 - precision: 0.4986 - recall: 0.5008
Epoch 6/10
157/157 [==============================] - 53s 340ms/step - loss: 0.7012 -
accuracy: 0.5096 - precision: 0.5095 - recall: 0.5152
Epoch 7/10
157/157 [==============================] - 53s 339ms/step - loss: 0.7023 -
accuracy: 0.4978 - precision: 0.4976 - recall: 0.4572
Epoch 8/10
157/157 [==============================] - 56s 354ms/step - loss: 0.7023 -
accuracy: 0.5032 - precision: 0.5032 - recall: 0.4960
Epoch 9/10
157/157 [==============================] - 53s 340ms/step - loss: 0.7047 -
accuracy: 0.4868 - precision: 0.4872 - recall: 0.5044
Epoch 10/10
157/157 [==============================] - 53s 338ms/step - loss: 0.7029 -
accuracy: 0.4896 - precision: 0.4895 - recall: 0.4828
```

Out[112]:
```
<keras.callbacks.History at 0x7f79ee4062d0>
```

After training, now we will evaluate our model.The evaluate() is for evaluating the already trained model using the validation or test data and the corresponding labels. Returns the loss value and metrics values for the model.

In [ ]:
```
model.evaluate(X_test, y_test)
```

```
16/16 [==============================] - 5s 325ms/step - loss: 0.6943 - ac
curacy: 0.5000 - precision: 0.5000 - recall: 0.0240
```

Out[113]:
```
[0.6943285465240479, 0.5, 0.5, 0.024000000208616257]
```

Testing of Transformer

In [ ]:
```
y_predicted = model.predict(X_test)
y_predicted = y_predicted.flatten()
```

Here we are defining a threshold such that if the predicted value is greater than 0.5, we store 1 in the array, otherwise 0.

In [ ]:
```
# defining a Threshold

y_predicted = np.where(y_predicted > 0.5, 1, 0)
```

Making confusion matrix

Using sklearn we will import confusion matrix and generate a Classification report by which we can see the precision, recall and accuracy.

In [ ]:
```
from sklearn.metrics import confusion_matrix, classification_report

cm = confusion_matrix(y_test, y_predicted)
cm
```

Out[116]:
```
array([[244,   6],
       [244,   6]])
```

Our report of our results is as below.

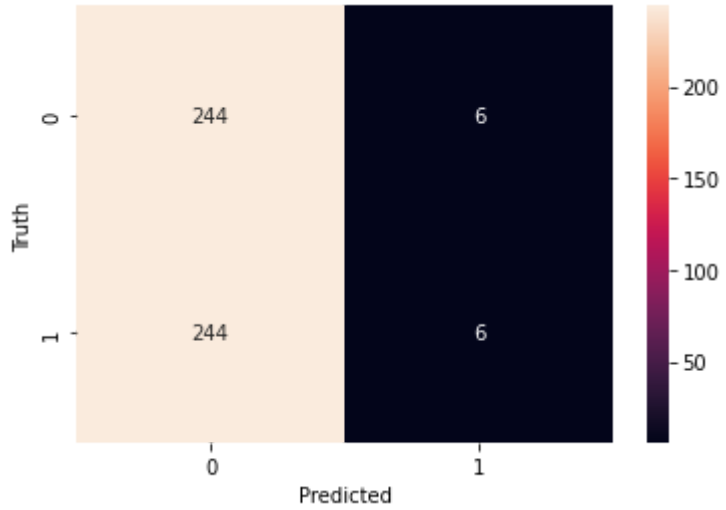
In [ ]:
```
print(classification_report(y_test, y_predicted))
```

```
              precision    recall  f1-score   support

           0       0.50      0.95      0.65       625
           1       0.39      0.03      0.06       625

    accuracy                           0.49      1250
   macro avg       0.44      0.49      0.35      1250
weighted avg       0.44      0.49      0.35      1250
```

Now we will use pyplot and seaborn libraries and visualise the results in the form of a heatmap.

In [ ]:
```
from matplotlib import pyplot as plt
import seaborn as sn
sn.heatmap(cm, annot=True, fmt='d')
plt.xlabel('Predicted')
plt.ylabel('Truth')
```

Out[117]:
```
Text(33.0, 0.5, 'Truth')
```



Using 20 Epoches

Here we are training our model using only 20 epochs.

In [ ]:
```
model.fit(X_train, y_train, epochs=20)
```

```
Epoch 1/20
157/157 [==============================] - 57s 364ms/step - loss: 0.7005 -
accuracy: 0.4848 - precision: 0.4860 - recall: 0.5280
Epoch 2/20
157/157 [==============================] - 55s 352ms/step - loss: 0.7067 -
accuracy: 0.4978 - precision: 0.4978 - recall: 0.5036
Epoch 3/20
157/157 [==============================] - 56s 358ms/step - loss: 0.6982 -
accuracy: 0.5012 - precision: 0.5012 - recall: 0.5136
Epoch 4/20
157/157 [==============================] - 55s 348ms/step - loss: 0.7016 -
accuracy: 0.4942 - precision: 0.4944 - recall: 0.5140
Epoch 5/20
157/157 [==============================] - 53s 340ms/step - loss: 0.7024 -
accuracy: 0.5032 - precision: 0.5032 - recall: 0.5072
Epoch 6/20
157/157 [==============================] - 56s 358ms/step - loss: 0.6982 -
accuracy: 0.4992 - precision: 0.4992 - recall: 0.4940
Epoch 7/20
157/157 [==============================] - 54s 342ms/step - loss: 0.7004 -
accuracy: 0.4922 - precision: 0.4926 - recall: 0.5184
Epoch 8/20
157/157 [==============================] - 55s 348ms/step - loss: 0.7005 -
accuracy: 0.4872 - precision: 0.4856 - recall: 0.4312
Epoch 9/20
157/157 [==============================] - 55s 349ms/step - loss: 0.7038 -
accuracy: 0.5048 - precision: 0.5049 - recall: 0.4984
Epoch 10/20
157/157 [==============================] - 54s 341ms/step - loss: 0.6983 -
accuracy: 0.4952 - precision: 0.4955 - recall: 0.5284
Epoch 11/20
157/157 [==============================] - 56s 356ms/step - loss: 0.7025 -
accuracy: 0.5036 - precision: 0.5036 - recall: 0.5040
Epoch 12/20
157/157 [==============================] - 53s 341ms/step - loss: 0.7000 -
accuracy: 0.4990 - precision: 0.4990 - recall: 0.4960
Epoch 13/20
157/157 [==============================] - 55s 349ms/step - loss: 0.7016 -
accuracy: 0.4980 - precision: 0.4980 - recall: 0.4880
Epoch 14/20
157/157 [==============================] - 55s 348ms/step - loss: 0.6998 -
accuracy: 0.5076 - precision: 0.5081 - recall: 0.4796
Epoch 15/20
157/157 [==============================] - 53s 341ms/step - loss: 0.7027 -
accuracy: 0.4854 - precision: 0.4852 - recall: 0.4772
Epoch 16/20
157/157 [==============================] - 56s 355ms/step - loss: 0.7026 -
accuracy: 0.4960 - precision: 0.4961 - recall: 0.5036
Epoch 17/20
157/157 [==============================] - 54s 341ms/step - loss: 0.6998 -
accuracy: 0.4908 - precision: 0.4905 - recall: 0.4744
Epoch 18/20
157/157 [==============================] - 55s 350ms/step - loss: 0.6991 -
accuracy: 0.4928 - precision: 0.4932 - recall: 0.5216
Epoch 19/20
157/157 [==============================] - 54s 345ms/step - loss: 0.7042 -
accuracy: 0.4942 - precision: 0.4936 - recall: 0.4496
Epoch 20/20
157/157 [==============================] - 53s 340ms/step - loss: 0.7029 -
accuracy: 0.4928 - precision: 0.4933 - recall: 0.5336
```

Out[121]:
```
<keras.callbacks.History at 0x7f79ee0714d0>
```

After training, now we will evaluate our model.The evaluate() is for evaluating the already trained model using the validation or test data and the corresponding labels. Returns the loss value and metrics values for the model.

In [ ]:
```
model.evaluate(X_test, y_test)
```

```
16/16 [==============================] - 5s 329ms/step - loss: 0.6939 - ac
curacy: 0.5000 - precision: 0.5000 - recall: 0.1360
```

Out[122]:
```
[0.693879246711731, 0.5, 0.5, 0.13600000739097595]
```

Testing of Transformer

In [ ]:
```
y_predicted = model.predict(X_test)
y_predicted = y_predicted.flatten()
```

Here we are defining a threshold such that if the predicted value is greater than 0.5, we store 1 in the array, otherwise 0.

In [ ]:
```
# defining a Threshold

y_predicted = np.where(y_predicted > 0.5, 1, 0)
```

Making confusion matrix

Using sklearn we will import confusion matrix and generate a Classification report by which we can see the precision, recall and accuracy.

In [ ]:
```
from sklearn.metrics import confusion_matrix, classification_report

cm = confusion_matrix(y_test, y_predicted)
cm
```

Out[125]:
```
array([[216,  34],
       [216,  34]])
```

Our report of our results is as below.

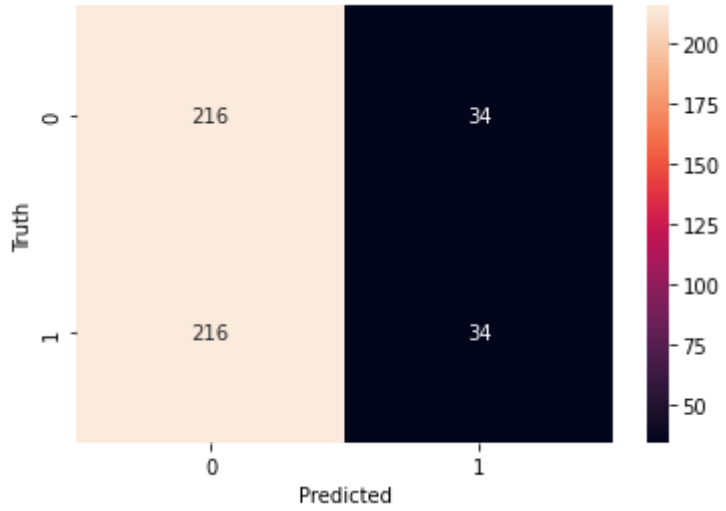
In [ ]:
```
print(classification_report(y_test, y_predicted))
```

```
              precision    recall  f1-score   support

           0       0.50      0.86      0.63       250
           1       0.50      0.14      0.21       250

    accuracy                           0.50       500
   macro avg       0.50      0.50      0.42       500
weighted avg       0.50      0.50      0.42       500
```

Now we will use pyplot and seaborn libraries and visualise the results in the form of a heatmap.

In [ ]:
```
from matplotlib import pyplot as plt
import seaborn as sn
sn.heatmap(cm, annot=True, fmt='d')
plt.xlabel('Predicted')
plt.ylabel('Truth')
```

Out[126]:
```
Text(33.0, 0.5, 'Truth')
```



In [ ]:
```

```