Topic Detection and Topic Tracking Using DistilBERT Model

January 1, 2024

Prepared by :

- chaimaa bouabd

Supervised By : * Pr. Khadija BOUZAACHANE

1 Import data from kaggle

[ ]:
```
# install Kaggle
!pip install -q kaggle
```

[ ]:
```
from google.colab import files
files.upload()
```

```
<IPython.core.display.HTML object>
```

```
Saving kaggle.json to kaggle.json
```

[ ]:
```
{'kaggle.json':
b'{"username":"chaimabouab","key":"4c9dba2231ee394879b8af8eafc26cdc"}'}
```

[ ]:
```
#Creat a kaggle folder
!mkdir ~/.kaggle
#copy the kaggle.json to folder created
!cp kaggle.json  ~/.kaggle/
#permission for the json to act
!chmod 600 ~/.kaggle/kaggle.json
```

[ ]:
```
!kaggle datasets download -d rmisra/news-category-dataset
```

```
Downloading news-category-dataset.zip to /content
 64% 17.0M/26.5M [00:00<00:00, 173MB/s]
100% 26.5M/26.5M [00:00<00:00, 201MB/s]
```

[ ]:
```
!unzip news-category-dataset.zip
```

```
unzip:  cannot find or open news-category-dataset.zip, news-category-
dataset.zip.zip or news-category-dataset.zip.ZIP.
```

2 EDA

[ ]:
```
!pip install nltk
```

```
Requirement already satisfied: nltk in /usr/local/lib/python3.10/dist-packages
(3.8.1)
Requirement already satisfied: click in /usr/local/lib/python3.10/dist-packages
(from nltk) (8.1.7)
Requirement already satisfied: joblib in /usr/local/lib/python3.10/dist-packages
(from nltk) (1.3.2)
Requirement already satisfied: regex>=2021.8.3 in
/usr/local/lib/python3.10/dist-packages (from nltk) (2023.6.3)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages
(from nltk) (4.66.1)
```

[ ]:
```
!pip install seaborn
!pip install transformers
!pip install tensorflow-addons
!pip install wordcloud
!pip install --upgrade pip
!pip install tensorflow
```

```
Requirement already satisfied: seaborn in /usr/local/lib/python3.10/dist-
packages (0.12.2)
Requirement already satisfied: numpy!=1.24.0,>=1.17 in
/usr/local/lib/python3.10/dist-packages (from seaborn) (1.23.5)
Requirement already satisfied: pandas>=0.25 in /usr/local/lib/python3.10/dist-
packages (from seaborn) (1.5.3)
Requirement already satisfied: matplotlib!=3.6.1,>=3.1 in
/usr/local/lib/python3.10/dist-packages (from seaborn) (3.7.1)
Requirement already satisfied: contourpy>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.1->seaborn)
(1.2.0)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-
packages (from matplotlib!=3.6.1,>=3.1->seaborn) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in
/usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.1->seaborn)
(4.46.0)
Requirement already satisfied: kiwisolver>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.1->seaborn)
(1.4.5)
Requirement already satisfied: packaging>=20.0 in
/usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.1->seaborn)
(23.2)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-
packages (from matplotlib!=3.6.1,>=3.1->seaborn) (9.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.1->seaborn)
```

```
(3.1.1)
Requirement already satisfied: python-dateutil>=2.7 in
/usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.1->seaborn)
(2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-
packages (from pandas>=0.25->seaborn) (2023.3.post1)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-
packages (from python-dateutil>=2.7->matplotlib!=3.6.1,>=3.1->seaborn) (1.16.0)
Requirement already satisfied: transformers in /usr/local/lib/python3.10/dist-
packages (4.35.2)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-
packages (from transformers) (3.13.1)
Requirement already satisfied: huggingface-hub<1.0,>=0.16.4 in
/usr/local/lib/python3.10/dist-packages (from transformers) (0.19.4)
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.10/dist-
packages (from transformers) (1.23.5)
Requirement already satisfied: packaging>=20.0 in
/usr/local/lib/python3.10/dist-packages (from transformers) (23.2)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.10/dist-
packages (from transformers) (6.0.1)
Requirement already satisfied: regex!=2019.12.17 in
/usr/local/lib/python3.10/dist-packages (from transformers) (2023.6.3)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-
packages (from transformers) (2.31.0)
Requirement already satisfied: tokenizers<0.19,>=0.14 in
/usr/local/lib/python3.10/dist-packages (from transformers) (0.15.0)
Requirement already satisfied: safetensors>=0.3.1 in
/usr/local/lib/python3.10/dist-packages (from transformers) (0.4.1)
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.10/dist-
packages (from transformers) (4.66.1)
Requirement already satisfied: fsspec>=2023.5.0 in
/usr/local/lib/python3.10/dist-packages (from huggingface-
hub<1.0,>=0.16.4->transformers) (2023.6.0)
Requirement already satisfied: typing-extensions>=3.7.4.3 in
/usr/local/lib/python3.10/dist-packages (from huggingface-
hub<1.0,>=0.16.4->transformers) (4.5.0)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.10/dist-packages (from requests->transformers) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-
packages (from requests->transformers) (3.6)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.10/dist-packages (from requests->transformers) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.10/dist-packages (from requests->transformers)
(2023.11.17)
Collecting tensorflow-addons
  Downloading tensorflow_addons-0.23.0-cp310-cp310-manylinux_2_17_x86_64.manylin
ux2014_x86_64.whl (611 kB)
```

```
                                    611.8/611.8
kB 3.6 MB/s eta 0:00:00
Requirement already satisfied: packaging in
/usr/local/lib/python3.10/dist-packages (from tensorflow-addons) (23.2)
Collecting typeguard<3.0.0,>=2.7 (from tensorflow-addons)
  Downloading typeguard-2.13.3-py3-none-any.whl (17 kB)
Installing collected packages: typeguard, tensorflow-addons
Successfully installed tensorflow-addons-0.23.0 typeguard-2.13.3
Requirement already satisfied: wordcloud in /usr/local/lib/python3.10/dist-
packages (1.9.3)
Requirement already satisfied: numpy>=1.6.1 in /usr/local/lib/python3.10/dist-
packages (from wordcloud) (1.23.5)
Requirement already satisfied: pillow in /usr/local/lib/python3.10/dist-packages
(from wordcloud) (9.4.0)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-
packages (from wordcloud) (3.7.1)
Requirement already satisfied: contourpy>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib->wordcloud) (1.2.0)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-
packages (from matplotlib->wordcloud) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in
/usr/local/lib/python3.10/dist-packages (from matplotlib->wordcloud) (4.46.0)
Requirement already satisfied: kiwisolver>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib->wordcloud) (1.4.5)
Requirement already satisfied: packaging>=20.0 in
/usr/local/lib/python3.10/dist-packages (from matplotlib->wordcloud) (23.2)
Requirement already satisfied: pyparsing>=2.3.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib->wordcloud) (3.1.1)
Requirement already satisfied: python-dateutil>=2.7 in
/usr/local/lib/python3.10/dist-packages (from matplotlib->wordcloud) (2.8.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-
packages (from python-dateutil>=2.7->matplotlib->wordcloud) (1.16.0)
Requirement already satisfied: pip in /usr/local/lib/python3.10/dist-packages
(23.1.2)
Collecting pip
  Downloading pip-23.3.2-py3-none-any.whl (2.1 MB)
                                    2.1/2.1 MB
10.6 MB/s eta 0:00:00
Installing collected packages: pip
  Attempting uninstall: pip
    Found existing installation: pip 23.1.2
    Uninstalling pip-23.1.2:
      Successfully uninstalled pip-23.1.2
Successfully installed pip-23.3.2
Requirement already satisfied: tensorflow in /usr/local/lib/python3.10/dist-
packages (2.12.0)
Requirement already satisfied: absl-py>=1.0.0 in /usr/local/lib/python3.10/dist-
```

```
packages (from tensorflow) (1.4.0)
Requirement already satisfied: astunparse>=1.6.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (1.6.3)
Requirement already satisfied: flatbuffers>=2.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (23.5.26)
Requirement already satisfied: gast<=0.4.0,>=0.2.1 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (0.4.0)
Requirement already satisfied: google-pasta>=0.1.1 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (0.2.0)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (1.60.0)
Requirement already satisfied: h5py>=2.9.0 in /usr/local/lib/python3.10/dist-
packages (from tensorflow) (3.9.0)
Requirement already satisfied: jax>=0.3.15 in /usr/local/lib/python3.10/dist-
packages (from tensorflow) (0.3.25)
Requirement already satisfied: keras<2.13,>=2.12.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (2.12.0)
Requirement already satisfied: libclang>=13.0.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (16.0.6)
Requirement already satisfied: numpy<1.24,>=1.22 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (1.23.5)
Requirement already satisfied: opt-einsum>=2.3.2 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (3.3.0)
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-
packages (from tensorflow) (23.2)
Requirement already satisfied:
protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!=4.21.5,<5.0.0dev,>=3.20.3
in /usr/local/lib/python3.10/dist-packages (from tensorflow) (3.20.3)
Requirement already satisfied: setuptools in /usr/local/lib/python3.10/dist-
packages (from tensorflow) (67.7.2)
Requirement already satisfied: six>=1.12.0 in /usr/local/lib/python3.10/dist-
packages (from tensorflow) (1.16.0)
Requirement already satisfied: tensorboard<2.13,>=2.12 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (2.12.0)
Requirement already satisfied: tensorflow-estimator<2.13,>=2.12.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (2.12.0)
Requirement already satisfied: termcolor>=1.1.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (2.4.0)
Requirement already satisfied: typing-extensions>=3.6.6 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (4.5.0)
Requirement already satisfied: wrapt<1.15,>=1.11.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (1.14.1)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (0.34.0)
Requirement already satisfied: wheel<1.0,>=0.23.0 in
/usr/local/lib/python3.10/dist-packages (from astunparse>=1.6.0->tensorflow)
(0.42.0)
Requirement already satisfied: scipy>=1.5 in /usr/local/lib/python3.10/dist-
```

```
packages (from jax>=0.3.15->tensorflow) (1.11.4)
Requirement already satisfied: google-auth<3,>=1.6.3 in
/usr/local/lib/python3.10/dist-packages (from
tensorboard<2.13,>=2.12->tensorflow) (2.17.3)
Requirement already satisfied: google-auth-oauthlib<0.5,>=0.4.1 in
/usr/local/lib/python3.10/dist-packages (from
tensorboard<2.13,>=2.12->tensorflow) (0.4.6)
Requirement already satisfied: markdown>=2.6.8 in
/usr/local/lib/python3.10/dist-packages (from
tensorboard<2.13,>=2.12->tensorflow) (3.5.1)
Requirement already satisfied: requests<3,>=2.21.0 in
/usr/local/lib/python3.10/dist-packages (from
tensorboard<2.13,>=2.12->tensorflow) (2.31.0)
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in
/usr/local/lib/python3.10/dist-packages (from
tensorboard<2.13,>=2.12->tensorflow) (0.7.2)
Requirement already satisfied: tensorboard-plugin-wit>=1.6.0 in
/usr/local/lib/python3.10/dist-packages (from
tensorboard<2.13,>=2.12->tensorflow) (1.8.1)
Requirement already satisfied: werkzeug>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from
tensorboard<2.13,>=2.12->tensorflow) (3.0.1)
Requirement already satisfied: cachetools<6.0,>=2.0.0 in
/usr/local/lib/python3.10/dist-packages (from google-
auth<3,>=1.6.3->tensorboard<2.13,>=2.12->tensorflow) (5.3.2)
Requirement already satisfied: pyasn1-modules>=0.2.1 in
/usr/local/lib/python3.10/dist-packages (from google-
auth<3,>=1.6.3->tensorboard<2.13,>=2.12->tensorflow) (0.3.0)
Requirement already satisfied: rsa<5,>=3.1.4 in /usr/local/lib/python3.10/dist-
packages (from google-auth<3,>=1.6.3->tensorboard<2.13,>=2.12->tensorflow) (4.9)
Requirement already satisfied: requests-oauthlib>=0.7.0 in
/usr/local/lib/python3.10/dist-packages (from google-auth-
oauthlib<0.5,>=0.4.1->tensorboard<2.13,>=2.12->tensorflow) (1.3.1)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.10/dist-packages (from
requests<3,>=2.21.0->tensorboard<2.13,>=2.12->tensorflow) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-
packages (from requests<3,>=2.21.0->tensorboard<2.13,>=2.12->tensorflow) (3.6)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.10/dist-packages (from
requests<3,>=2.21.0->tensorboard<2.13,>=2.12->tensorflow) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.10/dist-packages (from
requests<3,>=2.21.0->tensorboard<2.13,>=2.12->tensorflow) (2023.11.17)
Requirement already satisfied: MarkupSafe>=2.1.1 in
/usr/local/lib/python3.10/dist-packages (from
werkzeug>=1.0.1->tensorboard<2.13,>=2.12->tensorflow) (2.1.3)
Requirement already satisfied: pyasn1<0.6.0,>=0.4.6 in
```

```
/usr/local/lib/python3.10/dist-packages (from pyasn1-modules>=0.2.1->google-
auth<3,>=1.6.3->tensorboard<2.13,>=2.12->tensorflow) (0.5.1)
Requirement already satisfied: oauthlib>=3.0.0 in
/usr/local/lib/python3.10/dist-packages (from requests-oauthlib>=0.7.0->google-
auth-oauthlib<0.5,>=0.4.1->tensorboard<2.13,>=2.12->tensorflow) (3.2.2)
WARNING: Running pip as the 'root' user can result in broken permissions
and conflicting behaviour with the system package manager. It is recommended to
use a virtual environment instead: https://pip.pypa.io/warnings/venv
```

[ ]:
```python
import pandas as pd
import numpy as np
import tensorflow as tf
import transformers
import sklearn
import nltk
import seaborn as sns
import matplotlib.pyplot as plt
import string
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import confusion_matrix
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer
import tensorflow_addons as tfa
from collections import Counter
from wordcloud import WordCloud

nltk.download('stopwords')
nltk.download('wordnet')
nltk.download('punkt')

lemmatizer = WordNetLemmatizer()
stop_words = stopwords.words('english')
```

```
/usr/local/lib/python3.10/dist-
packages/tensorflow_addons/utils/tfa_eol_msg.py:23: UserWarning:

TensorFlow Addons (TFA) has ended development and introduction of new features.
TFA has entered a minimal maintenance and release mode until a planned end of
life in May 2024.
Please modify downstream libraries to take dependencies from other repositories
in our TensorFlow community (e.g. Keras, Keras-CV, and Keras-NLP).

For more information see: https://github.com/tensorflow/addons/issues/2807
```

```
  warnings.warn(
/usr/local/lib/python3.10/dist-
packages/tensorflow_addons/utils/ensure_tf_install.py:53: UserWarning:
Tensorflow Addons supports using Python ops for all Tensorflow versions above or
equal to 2.13.0 and strictly below 2.16.0 (nightly versions are not supported).
 The versions of TensorFlow you are currently using is 2.12.0 and is not
supported.
Some things might work, some things might not.
If you were to encounter a bug, do not file an issue.
If you want to make sure you're using a tested and supported configuration,
either change the TensorFlow version or the TensorFlow Addons's version.
You can find the compatibility matrix in TensorFlow Addon's readme:
https://github.com/tensorflow/addons
  warnings.warn(
[nltk_data] Downloading package stopwords to /root/nltk_data…
[nltk_data]   Unzipping corpora/stopwords.zip.
[nltk_data] Downloading package wordnet to /root/nltk_data…
[nltk_data] Downloading package punkt to /root/nltk_data…
[nltk_data]   Unzipping tokenizers/punkt.zip.
```

[ ]:
```
try:
    tpu = tf.distribute.cluster_resolver.TPUClusterResolver()
    print('Running on TPU ', tpu.master())
except ValueError:
    tpu = None


if tpu:
    tf.config.experimental_connect_to_cluster(tpu)
    tf.tpu.experimental.initialize_tpu_system(tpu)
    strategy = tf.distribute.TPUStrategy(tpu)  # Updated line
else:
    strategy = tf.distribute.get_strategy()

print("REPLICAS: ", strategy.num_replicas_in_sync)
```

```
Running on TPU  grpc://10.121.1.106:8470
REPLICAS:  8
```

2.1 Reading and Displaying the file of news articles using pandas

[ ]:
```
df = pd.read_json('./News_Category_Dataset_v3.json',lines=True)
# reads a JSON file from a given path and converts it into a pandas DataFrame␣
  ↪object
df.head()
```

[ ]:
```
                                                link  \
0  https://www.huffpost.com/entry/covid-boosters-…
1  https://www.huffpost.com/entry/american-airlin…
2  https://www.huffpost.com/entry/funniest-tweets…
3  https://www.huffpost.com/entry/funniest-parent…
4  https://www.huffpost.com/entry/amy-cooper-lose…

                                           headline   category  \
0  Over 4 Million Americans Roll Up Sleeves For O…  U.S. NEWS
1  American Airlines Flyer Charged, Banned For Li…  U.S. NEWS
2  23 Of The Funniest Tweets About Cats And Dogs …     COMEDY
3  The Funniest Tweets From Parents This Week (Se…  PARENTING
4  Woman Who Called Cops On Black Bird-Watcher Lo…  U.S. NEWS

                                 short_description               authors  \
0  Health experts said it is too early to predict…  Carla K. Johnson, AP
1  He was subdued by passengers and crew when he …        Mary Papenfuss
2  "Until you have a dog you don't understand wha…         Elyse Wanshel
3  "Accidentally put grown-up toothpaste on my to…      Caroline Bologna
4  Amy Cooper accused investment firm Franklin Te…        Nina Golgowski

        date
0 2022-09-23
1 2022-09-23
2 2022-09-23
3 2022-09-23
4 2022-09-22
```

```
Error: Runtime no longer has a reference to this dataframe, please re-run this
cell and try again.
Error: Runtime no longer has a reference to this dataframe, please re-run this
cell and try again.
Error: Runtime no longer has a reference to this dataframe, please re-run this
cell and try again.
Error: Runtime no longer has a reference to this dataframe, please re-run this
cell and try again.
```

```
<google.colab._quickchart_helpers.SectionTitle at 0x7c56a4374340>
```

```
from matplotlib import pyplot as plt
import seaborn as sns
_df_0.groupby('link').size().plot(kind='barh', color=sns.palettes.
  ↪mpl_palette('Dark2'))
plt.gca().spines[['top', 'right',]].set_visible(False)
```

```
from matplotlib import pyplot as plt
import seaborn as sns
_df_1.groupby('headline').size().plot(kind='barh', color=sns.palettes.
  ↪mpl_palette('Dark2'))
```

```
plt.gca().spines[['top', 'right',]].set_visible(False)
```

```
from matplotlib import pyplot as plt
import seaborn as sns
_df_2.groupby('category').size().plot(kind='barh', color=sns.palettes.
 ↪mpl_palette('Dark2'))
plt.gca().spines[['top', 'right',]].set_visible(False)
```

```
from matplotlib import pyplot as plt
import seaborn as sns
_df_3.groupby('short_description').size().plot(kind='barh', color=sns.palettes.
 ↪mpl_palette('Dark2'))
plt.gca().spines[['top', 'right',]].set_visible(False)
```

```
<google.colab._quickchart_helpers.SectionTitle at 0x7c52db217b50>
```

```
from matplotlib import pyplot as plt
import seaborn as sns
def _plot_series(series, series_name, series_index=0):
  from matplotlib import pyplot as plt
  import seaborn as sns
  palette = list(sns.palettes.mpl_palette('Dark2'))
  counted = (series['date']
               .value_counts()
             .reset_index(name='counts')
             .rename({'index': 'date'}, axis=1)
             .sort_values('date', ascending=True))
  xs = counted['date']
  ys = counted['counts']
  plt.plot(xs, ys, label=series_name, color=palette[series_index % len(palette)])

fig, ax = plt.subplots(figsize=(10, 5.2), layout='constrained')
df_sorted = _df_4.sort_values('date', ascending=True)
for i, (series_name, series) in enumerate(df_sorted.groupby('link')):
  _plot_series(series, series_name, i)
  fig.legend(title='link', bbox_to_anchor=(1, 1), loc='upper left')
sns.despine(fig=fig, ax=ax)
plt.xlabel('date')
_ = plt.ylabel('count()')
```

```
from matplotlib import pyplot as plt
import seaborn as sns
def _plot_series(series, series_name, series_index=0):
  from matplotlib import pyplot as plt
  import seaborn as sns
  palette = list(sns.palettes.mpl_palette('Dark2'))
  counted = (series['date']
               .value_counts()
             .reset_index(name='counts')
             .rename({'index': 'date'}, axis=1)
```

```
              .sort_values('date', ascending=True))
  xs = counted['date']
  ys = counted['counts']
  plt.plot(xs, ys, label=series_name, color=palette[series_index % len(palette)])

fig, ax = plt.subplots(figsize=(10, 5.2), layout='constrained')
df_sorted = _df_5.sort_values('date', ascending=True)
for i, (series_name, series) in enumerate(df_sorted.groupby('headline')):
  _plot_series(series, series_name, i)
  fig.legend(title='headline', bbox_to_anchor=(1, 1), loc='upper left')
sns.despine(fig=fig, ax=ax)
plt.xlabel('date')
_ = plt.ylabel('count()')
```

```
from matplotlib import pyplot as plt
import seaborn as sns
def _plot_series(series, series_name, series_index=0):
  from matplotlib import pyplot as plt
  import seaborn as sns
  palette = list(sns.palettes.mpl_palette('Dark2'))
  counted = (series['date']
               .value_counts()
              .reset_index(name='counts')
              .rename({'index': 'date'}, axis=1)
              .sort_values('date', ascending=True))
  xs = counted['date']
  ys = counted['counts']
  plt.plot(xs, ys, label=series_name, color=palette[series_index % len(palette)])

fig, ax = plt.subplots(figsize=(10, 5.2), layout='constrained')
df_sorted = _df_6.sort_values('date', ascending=True)
for i, (series_name, series) in enumerate(df_sorted.groupby('category')):
  _plot_series(series, series_name, i)
  fig.legend(title='category', bbox_to_anchor=(1, 1), loc='upper left')
sns.despine(fig=fig, ax=ax)
plt.xlabel('date')
_ = plt.ylabel('count()')
```

```
from matplotlib import pyplot as plt
import seaborn as sns
def _plot_series(series, series_name, series_index=0):
  from matplotlib import pyplot as plt
  import seaborn as sns
  palette = list(sns.palettes.mpl_palette('Dark2'))
  counted = (series['date']
               .value_counts()
              .reset_index(name='counts')
              .rename({'index': 'date'}, axis=1)
              .sort_values('date', ascending=True))
```

```
  xs = counted['date']
  ys = counted['counts']
  plt.plot(xs, ys, label=series_name, color=palette[series_index % len(palette)])

fig, ax = plt.subplots(figsize=(10, 5.2), layout='constrained')
df_sorted = _df_7.sort_values('date', ascending=True)
for i, (series_name, series) in enumerate(df_sorted.
  ↪groupby('short_description')):
  _plot_series(series, series_name, i)
  fig.legend(title='short_description', bbox_to_anchor=(1, 1), loc='upper left')
sns.despine(fig=fig, ax=ax)
plt.xlabel('date')
_ = plt.ylabel('count()')
```

```
<google.colab._quickchart_helpers.SectionTitle at 0x7c52db284820>
```

```
from matplotlib import pyplot as plt
import seaborn as sns
import pandas as pd
plt.subplots(figsize=(8, 8))
df_2dhist = pd.DataFrame({
    x_label: grp['headline'].value_counts()
    for x_label, grp in _df_8.groupby('link')
})
sns.heatmap(df_2dhist, cmap='viridis')
plt.xlabel('link')
_ = plt.ylabel('headline')
```

```
from matplotlib import pyplot as plt
import seaborn as sns
import pandas as pd
plt.subplots(figsize=(8, 8))
df_2dhist = pd.DataFrame({
    x_label: grp['category'].value_counts()
    for x_label, grp in _df_9.groupby('headline')
})
sns.heatmap(df_2dhist, cmap='viridis')
plt.xlabel('headline')
_ = plt.ylabel('category')
```

```
from matplotlib import pyplot as plt
import seaborn as sns
import pandas as pd
plt.subplots(figsize=(8, 8))
df_2dhist = pd.DataFrame({
    x_label: grp['short_description'].value_counts()
    for x_label, grp in _df_10.groupby('category')
})
sns.heatmap(df_2dhist, cmap='viridis')
```

```
plt.xlabel('category')
_ = plt.ylabel('short_description')
```

```
from matplotlib import pyplot as plt
import seaborn as sns
import pandas as pd
plt.subplots(figsize=(8, 8))
df_2dhist = pd.DataFrame({
    x_label: grp['authors'].value_counts()
    for x_label, grp in _df_11.groupby('short_description')
})
sns.heatmap(df_2dhist, cmap='viridis')
plt.xlabel('short_description')
_ = plt.ylabel('authors')
```

[ ]:
```
df.shape
```

[ ]:
```
(209527, 6)
```

[ ]:
```
print(df.isnull().sum())
```

```
link                0
headline            0
category            0
short_description   0
authors             0
date                0
dtype: int64
```

[ ]:
```
print(df.describe())
```

```
                                                  link        headline  \
count                                           209527          209527
unique                                          209486          207996
top     https://www.huffingtonpost.comhttps://www.wash…  Sunday Roundup
freq                                                 2              90
first                                              NaN             NaN
last                                               NaN             NaN

        category short_description authors                 date
count     209527            209527  209527               209527
unique        42            187022   29169                 3890
top     POLITICS                              2014-03-25 00:00:00
freq       35602             19712   37418                  100
first        NaN               NaN     NaN  2012-01-28 00:00:00
last         NaN               NaN     NaN  2022-09-23 00:00:00
```

```
<ipython-input-13-772c287cbb5c>:1: FutureWarning: Treating datetime data as
categorical rather than numeric in `.describe` is deprecated and will be removed
```

```
in a future version of pandas. Specify `datetime_is_numeric=True` to silence
this warning and adopt the future behavior now.
  print(df.describe())
```

[ ]:
```
print('Number of Duplicates:', len(df[df.duplicated()]))
```

```
Number of Duplicates: 13
```

[ ]:
```
df = df.drop_duplicates(keep="last")
```

[ ]:
```
df['category'].unique()
```

[ ]:
```
array(['U.S. NEWS', 'COMEDY', 'PARENTING', 'WORLD NEWS', 'CULTURE & ARTS',
       'TECH', 'SPORTS', 'ENTERTAINMENT', 'POLITICS', 'WEIRD NEWS',
       'ENVIRONMENT', 'EDUCATION', 'CRIME', 'SCIENCE', 'WELLNESS',
       'BUSINESS', 'STYLE & BEAUTY', 'FOOD & DRINK', 'MEDIA',
       'QUEER VOICES', 'HOME & LIVING', 'WOMEN', 'BLACK VOICES', 'TRAVEL',
       'MONEY', 'RELIGION', 'LATINO VOICES', 'IMPACT', 'WEDDINGS',
       'COLLEGE', 'PARENTS', 'ARTS & CULTURE', 'STYLE', 'GREEN', 'TASTE',
       'HEALTHY LIVING', 'THE WORLDPOST', 'GOOD NEWS', 'WORLDPOST',
       'FIFTY', 'ARTS', 'DIVORCE'], dtype=object)
```

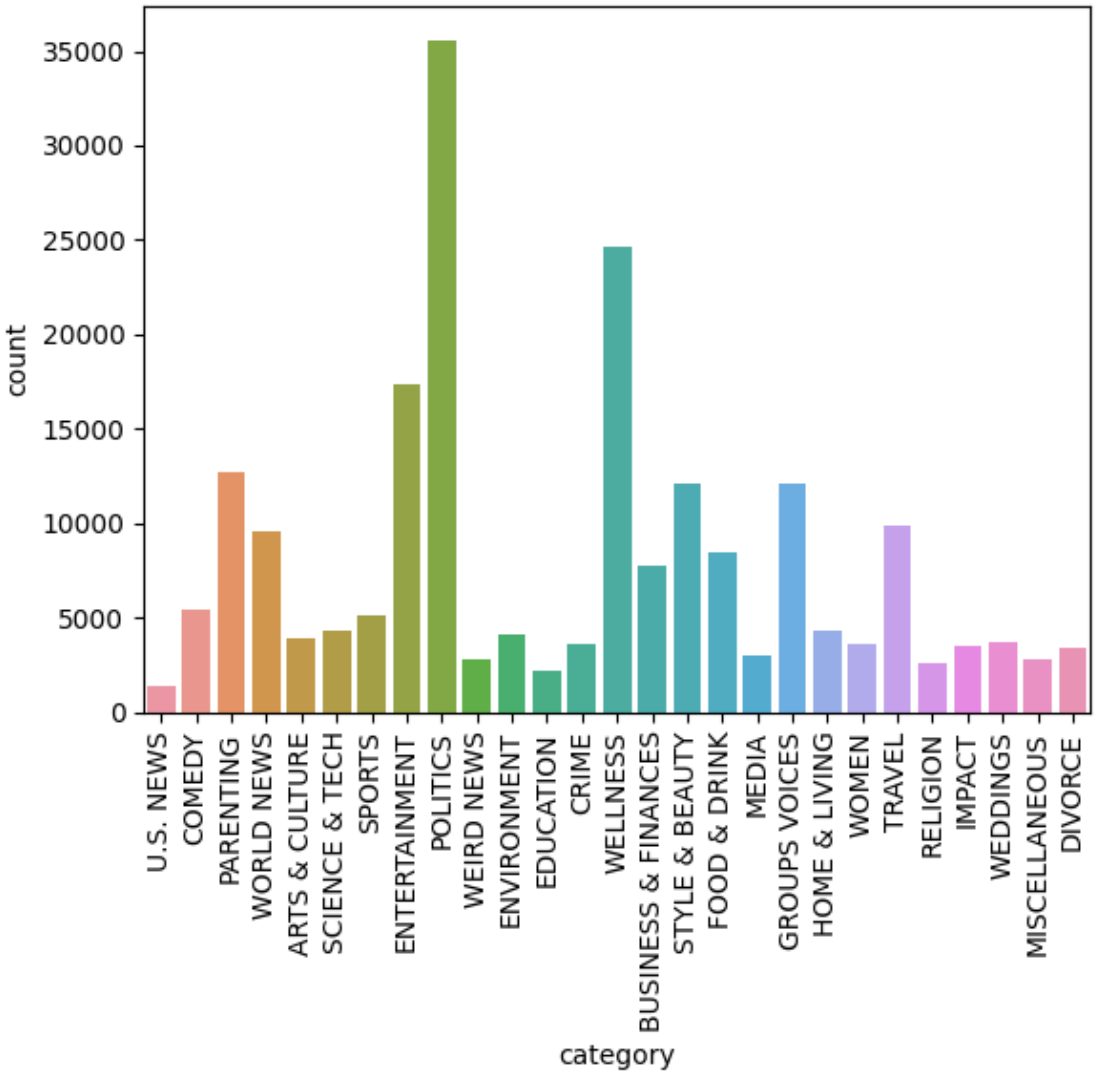
[ ]:
```
df['category']=df['category'].replace({"HEALTHY LIVING": "WELLNESS",
"QUEER VOICES": "GROUPS VOICES",
"BUSINESS": "BUSINESS & FINANCES",
"PARENTS": "PARENTING",
"BLACK VOICES": "GROUPS VOICES",
"THE WORLDPOST": "WORLD NEWS",
"STYLE": "STYLE & BEAUTY",
"GREEN": "ENVIRONMENT",
"TASTE": "FOOD & DRINK",
"WORLDPOST": "WORLD NEWS",
"SCIENCE": "SCIENCE & TECH",
"TECH": "SCIENCE & TECH",
"MONEY": "BUSINESS & FINANCES",
"ARTS": "ARTS & CULTURE",
"COLLEGE": "EDUCATION",
"LATINO VOICES": "GROUPS VOICES",
"CULTURE & ARTS": "ARTS & CULTURE",
"FIFTY": "MISCELLANEOUS",
"GOOD NEWS": "MISCELLANEOUS"})

# Merging similar kinds of categories into a single category
```

[ ]:
```
df['category'].unique()
```

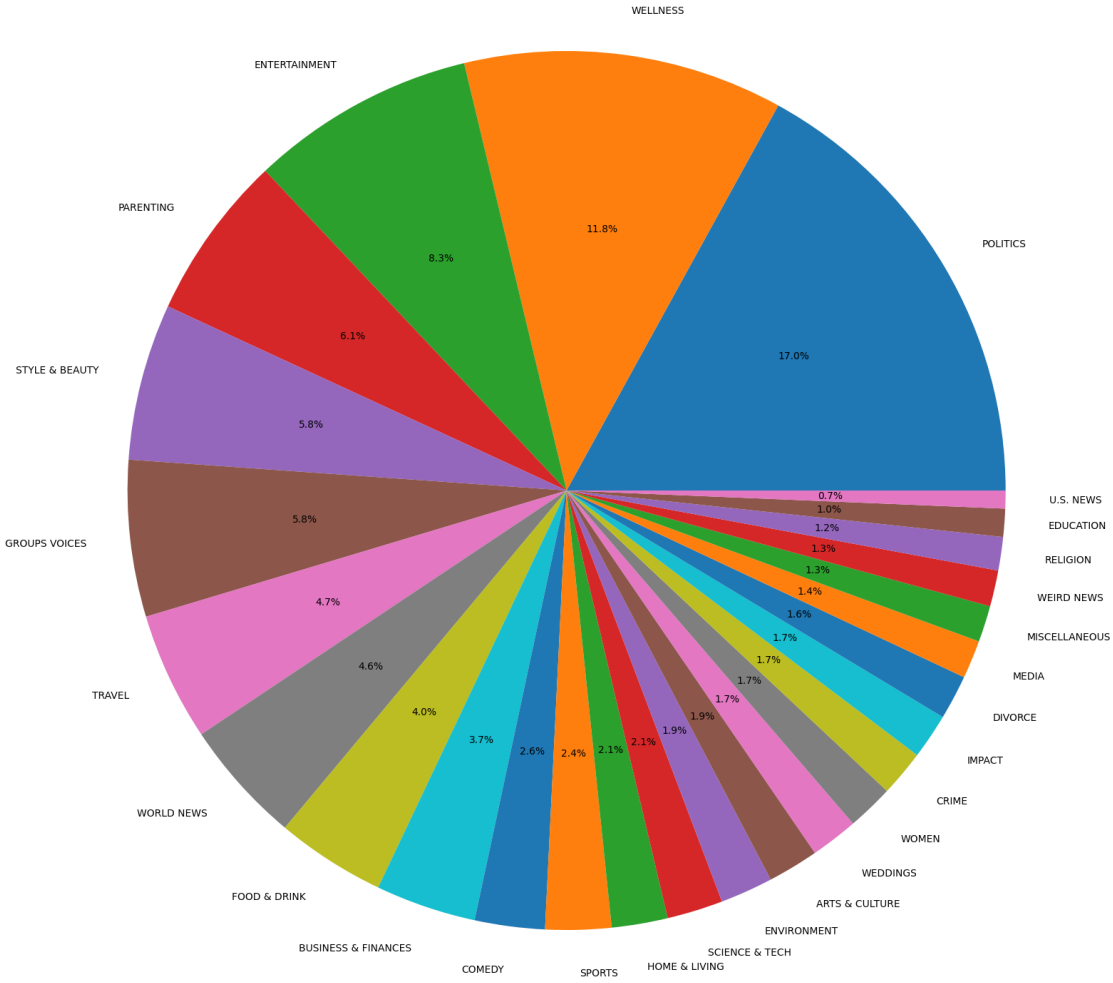
[ ]:
```
array(['U.S. NEWS', 'COMEDY', 'PARENTING', 'WORLD NEWS', 'ARTS & CULTURE',
       'SCIENCE & TECH', 'SPORTS', 'ENTERTAINMENT', 'POLITICS',
       'WEIRD NEWS', 'ENVIRONMENT', 'EDUCATION', 'CRIME', 'WELLNESS',
       'BUSINESS & FINANCES', 'STYLE & BEAUTY', 'FOOD & DRINK', 'MEDIA',
       'GROUPS VOICES', 'HOME & LIVING', 'WOMEN', 'TRAVEL', 'RELIGION',
       'IMPACT', 'WEDDINGS', 'MISCELLANEOUS', 'DIVORCE'], dtype=object)
```

[ ]:
```
# Count of News belonging to each category
print(df.groupby("category")["headline"].count().sort_values(ascending=False)) # Prints the count of news headlines in each category, sorted in descending order

sns.countplot(x="category", data=df)  # Plots a bar graph showing the count of news in each category
plt.xticks(rotation=90)  # Rotates the x-axis labels by 90 degrees for better readability
plt.show()  # Displays the plot
```

```
category
POLITICS                35601
WELLNESS                24636
ENTERTAINMENT           17362
PARENTING               12746
STYLE & BEAUTY          12065
GROUPS VOICES           12060
TRAVEL                   9900
WORLD NEWS               9542
FOOD & DRINK             8436
BUSINESS & FINANCES      7748
COMEDY                   5400
SPORTS                   5077
HOME & LIVING            4320
SCIENCE & TECH           4306
ENVIRONMENT              4065
ARTS & CULTURE           3922
WEDDINGS                 3653
WOMEN                    3571
CRIME                    3562
IMPACT                   3484
DIVORCE                  3426
MEDIA                    2944
MISCELLANEOUS            2799
WEIRD NEWS               2777
RELIGION                 2577
EDUCATION                2158
U.S. NEWS                1377
Name: headline, dtype: int64
```



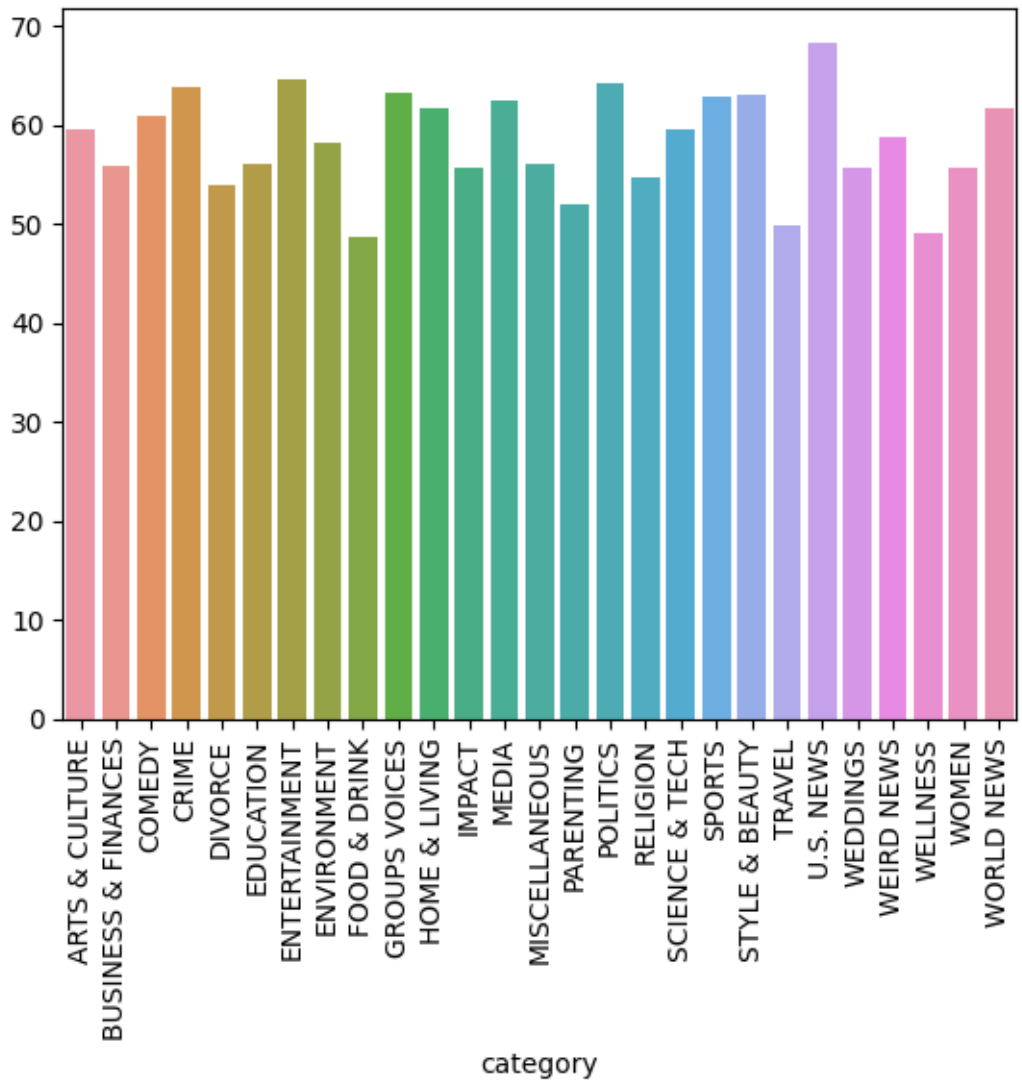
[ ]:
```
plt.figure(figsize=(20,20))  # Sets the figure size for the pie chart
plt.pie(df["category"].value_counts(), labels=df["category"].value_counts().
  ↪index, autopct="%1.1f%%", textprops={'fontsize': 10})  # Plots a pie chart␣
  ↪showing the percentage of news in each category
plt.show()  # Displays the plot
```



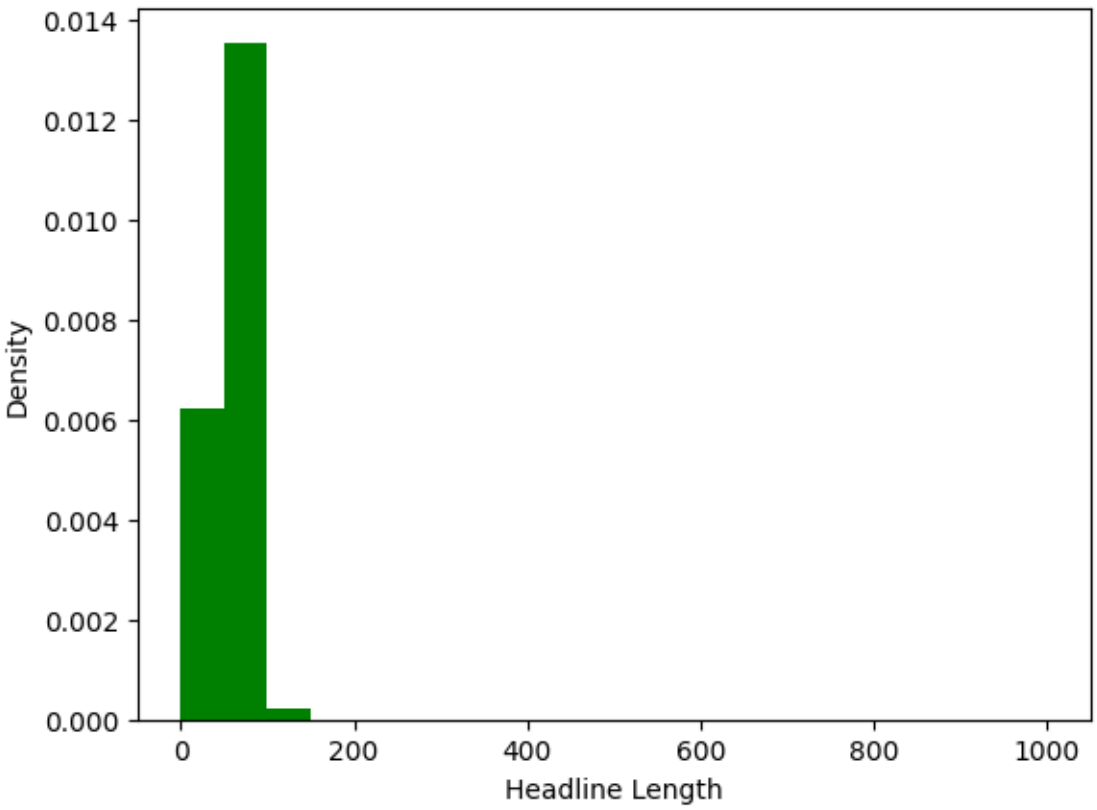
[ ]:
```
#AVERAGE LENGTH OF TITLE FOR EACH CATEGORY
df["headline_length"] = df["headline"].str.len()
print(df.groupby("category")["headline_length"].mean())
sns.barplot(x=df.groupby("category")["headline_length"].mean().index, y=df.
  ↪groupby("category")["headline_length"].mean().values)
plt.xticks(rotation=90)
plt.show()
```

```
category
ARTS & CULTURE          59.607853
BUSINESS & FINANCES     55.826794
COMEDY                  60.841111
CRIME                   63.750702
```

```
DIVORCE                53.980736
EDUCATION              55.991196
ENTERTAINMENT          64.620781
ENVIRONMENT            58.284379
FOOD & DRINK           48.651375
GROUPS VOICES          63.257380
HOME & LIVING          61.607639
IMPACT                 55.661596
MEDIA                  62.549932
MISCELLANEOUS          56.032512
PARENTING              51.893300
POLITICS               64.250190
RELIGION               54.672875
SCIENCE & TECH         59.501161
SPORTS                 62.849714
STYLE & BEAUTY         63.146705
TRAVEL                 49.904040
U.S. NEWS              68.381990
WEDDINGS               55.697235
WEIRD NEWS             58.836874
WELLNESS               49.076717
WOMEN                  55.587511
WORLD NEWS             61.665269
Name: headline_length, dtype: float64
```



[ ]:
```
#Density of News vs Paragraph Length
plt.hist(df["headline_length"], bins=20, color="green", range=(0, 1000),␣
 ↪density=True)
plt.xlabel("Headline Length")
plt.ylabel("Density")
plt.show()
```



2.2 Preprocessing of Data

[ ]:
```
# Combining the 'headline' and 'short_description' columns into a new 'text'␣
 ↪column
df['text'] = df['headline'] + ' ' + df['short_description']
```

[ ]:
```
# Function to Preprocess Text by Removing Stopwords and Punctuations
def remove_stopwords_and_punctuations(text):
    words = nltk.word_tokenize(text)  # Tokenizes the input text into␣
 ↪individual words
    words = [lemmatizer.lemmatize(word) for word in words if word.lower() not␣
 ↪in stop_words]  # Lemmatizes the words and removes stopwords
    words_without_punctuations = [''.join(c for c in word if c not in string.
 ↪punctuation) for word in words]  # Removes punctuation from the words
    words_preprocessed = [word.replace("‘", "").replace("’", "").replace("""","␣
 ↪").replace("""," ") for word in words_without_punctuations if len(word)>2] ␣
 ↪# Removes certain special characters and words of length 2 or less
    return ' '.join(words_preprocessed)  # Joins the preprocessed words back␣
 ↪into a single string
```

```
[ ]: df['text'] = df['text'].apply(remove_stopwords_and_punctuations)  # Applies the␣
      ↪'remove_stopwords_and_punctuations' function to each entry in the 'text'␣
      ↪column of the dataframe
```

2.3 Tokenizing Text, Counting Top Words by Category, and Generating Word Clouds

```
[ ]: def tokenize_text(text):
         return text.lower().split()  # Splits the input text into individual words␣
      ↪(tokens)

     df['tokenized_text'] = df['text'].apply(tokenize_text)  # Applies the␣
      ↪'tokenize_text' function to each entry in the 'text' column of the dataframe␣
      ↪'df'

     from wordcloud import WordCloud

     # Function to count the occurrences of words in each category
     def count_top_words_by_category(category_list):
         all_words = [word for words in category_list for word in words]  # Flattens␣
      ↪the list of lists into a single list
         word_counts = Counter(all_words)  # Counts the occurrences of each word
         return word_counts.most_common(50)  # Returns the 50 most common words

     # Groups the DataFrame by 'category' and applies the␣
      ↪count_top_words_by_category function
     top_words_by_category = df.groupby('category')['tokenized_text'].
      ↪apply(count_top_words_by_category).reset_index()

     # Creates word maps for each category
     for idx, row in top_words_by_category.iterrows():
         category = row['category']  # Current category
         top_words = row['tokenized_text']  # Top words for the current category

         # Creates a dictionary from the top_words list
         word_freq_dict = dict(top_words)

         # Generates a word cloud for the current category
         wordcloud = WordCloud(width=800, height=400, background_color='white',␣
      ↪max_words=50, stopwords=stopwords).generate_from_frequencies(word_freq_dict)

         # Plots the word cloud
         plt.figure(figsize=(10, 6))
         plt.imshow(wordcloud, interpolation='bilinear')
         plt.title(f"Word Cloud for Category: {category}")  # Sets the title of the␣
      ↪plot
```

```
    plt.axis('off')  # Hides the axis
    plt.show()  # Displays the plot
```

[ ]:
```
# Function to count the occurrences of words in each category
def count_top_words_by_category(category_list):
    all_words = [word for words in category_list for word in words]  # Flattens␣
 ↪the list of lists into a single list
    word_counts = Counter(all_words)  # Counts the occurrences of each word
    return word_counts.most_common(10)  # Returns the 10 most common words

# Groups the DataFrame by 'category' and applies the␣
 ↪count_top_words_by_category function
top_words_by_category = df.groupby('category')['tokenized_text'].
 ↪apply(count_top_words_by_category).reset_index()

# Displays the top 10 words for each category
for idx, row in top_words_by_category.iterrows():
    category = row['category']  # Current category
    top_words = row['tokenized_text']  # Top words for the current category
    print(f"Category: {category}")
    print(f"Top 10 words and their count:")
    for word, count in top_words:
        print(f"{word}: {count}")  # Prints each word and its count
    print()  # Prints a newline for readability
```

```
Category: ARTS & CULTURE
Top 10 words and their count:
art: 848
artist: 490
new: 484
photos: 353
one: 321
world: 295
year: 265
work: 260
show: 221
book: 212

Category: BUSINESS & FINANCES
Top 10 words and their count:
business: 881
new: 849
year: 709
company: 633
one: 632
time: 610
people: 600
```

```
money: 522
make: 492
get: 490

Category: COMEDY
Top 10 words and their count:
trump: 1298
donald: 676
video: 618
show: 470
jimmy: 422
snl: 422
like: 409
colbert: 406
new: 369
one: 347

Category: CRIME
Top 10 words and their count:
police: 866
man: 604
shooting: 413
said: 358
say: 317
suspect: 311
woman: 291
allegedly: 277
killed: 273
officer: 271

Category: DIVORCE
Top 10 words and their count:
divorce: 2470
marriage: 571
one: 403
child: 389
divorced: 377
life: 372
relationship: 358
time: 351
family: 327
year: 320

Category: EDUCATION
Top 10 words and their count:
college: 598
student: 552
school: 524
```

```
education: 461
university: 264
teacher: 218
new: 200
students: 191
one: 172
campus: 154

Category: ENTERTAINMENT
Top 10 words and their count:
new: 2057
star: 1196
show: 1117
film: 1052
one: 1025
trump: 1015
movie: 965
first: 903
like: 876
year: 789

Category: ENVIRONMENT
Top 10 words and their count:
climate: 903
change: 542
week: 393
new: 385
animal: 381
world: 336
one: 308
year: 305
dog: 268
photos: 267

Category: FOOD & DRINK
Top 10 words and their count:
food: 1377
make: 1171
recipe: 982
recipes: 957
best: 929
day: 782
photos: 731
one: 709
like: 652
new: 612

Category: GROUPS VOICES
```

```
Top 10 words and their count:
gay: 2076
black: 1741
new: 1368
people: 1165
one: 882
lgbt: 743
year: 714
love: 653
time: 645
life: 643

Category: HOME & LIVING
Top 10 words and their count:
home: 1729
photos: 1375
make: 500
day: 499
house: 429
video: 427
new: 416
one: 401
ideas: 342
craft: 318

Category: IMPACT
Top 10 words and their count:
people: 524
world: 508
day: 463
one: 436
year: 381
life: 344
new: 327
child: 322
help: 307
time: 298

Category: MEDIA
Top 10 words and their count:
news: 693
trump: 691
new: 403
fox: 377
donald: 268
media: 239
host: 206
cnn: 203
```

```
said: 183
york: 183

Category: MISCELLANEOUS
Top 10 words and their count:
life: 353
one: 314
time: 264
day: 260
year: 251
like: 243
people: 231
woman: 191
new: 188
make: 185

Category: PARENTING
Top 10 words and their count:
child: 2856
kid: 1872
kids: 1667
mom: 1657
parent: 1632
baby: 1548
one: 1507
day: 1506
time: 1396
like: 1168

Category: POLITICS
Top 10 words and their count:
trump: 14287
donald: 4653
president: 3719
new: 2822
gop: 2767
clinton: 2635
said: 2633
house: 2478
obama: 2405
state: 2220

Category: RELIGION
Top 10 words and their count:
pope: 358
church: 268
people: 245
francis: 226
```

```
god: 215
daily: 214
religious: 198
one: 195
world: 180
faith: 173

Category: SCIENCE & TECH
Top 10 words and their count:
new: 654
apple: 415
video: 358
facebook: 340
may: 339
week: 313
one: 303
google: 289
space: 278
could: 277

Category: SPORTS
Top 10 words and their count:
game: 518
nfl: 511
team: 456
player: 390
football: 357
world: 347
first: 343
new: 297
video: 284
one: 278

Category: STYLE & BEAUTY
Top 10 words and their count:
photos: 4425
style: 2353
fashion: 2330
look: 1639
week: 1590
new: 1485
check: 1251
want: 1126
dress: 1115
beauty: 1011

Category: TRAVEL
Top 10 words and their count:
```

```
travel: 1863
photos: 1718
world: 1408
new: 1242
one: 1092
best: 1056
city: 967
hotel: 927
time: 809
day: 792

Category: U.S. NEWS
Top 10 words and their count:
said: 219
new: 176
police: 154
people: 128
california: 111
state: 108
shooting: 108
school: 96
death: 91
year: 87

Category: WEDDINGS
Top 10 words and their count:
wedding: 3011
marriage: 775
day: 603
weddings: 530
couple: 513
bride: 476
one: 473
married: 445
video: 438
love: 431

Category: WEIRD NEWS
Top 10 words and their count:
man: 321
one: 161
woman: 151
people: 140
like: 139
new: 136
dog: 135
police: 135
watch: 133
```

```
say: 107

Category: WELLNESS
Top 10 words and their count:
life: 4013
health: 3024
one: 2724
time: 2715
people: 2680
new: 2312
study: 2104
make: 2075
day: 2043
get: 1916

Category: WOMEN
Top 10 words and their count:
women: 997
woman: 919
one: 302
day: 293
like: 291
life: 259
week: 258
men: 247
time: 241
sexual: 228

Category: WORLD NEWS
Top 10 words and their count:
people: 813
trump: 741
president: 725
world: 725
new: 712
attack: 665
said: 664
country: 641
year: 603
one: 579
```

[ ]:
```python
def tokenize_text(text):
    return text.split()  # Splits the input text into individual words (tokens)
```

```
df['tokenized_text'] = df['text'].apply(tokenize_text)  # Applies the␣
 ↪'tokenize_text' function to each entry in the 'text' column of the dataframe␣
 ↪'df'

def get_top_ten_words(category):
    # Filters the dataframe by category
    df_category = df[df["category"] == category]
    # Concatenates all the tokens into one list
    all_tokens = []
    for tokens in df_category["tokenized_text"]:
        all_tokens.extend(tokens)
    # Counts the word frequencies using nltk.FreqDist
    freq_dist = nltk.FreqDist(all_tokens)
    # Gets the top ten words and their counts
    top_ten_words = freq_dist.most_common(10)
    return top_ten_words

# Loops through the unique categories and prints the results
for category in df["category"].unique():
    print(f"Top ten common words in {category}:")
    print(get_top_ten_words(category))  # Prints the top ten common words in␣
 ↪the current category
```

```
Top ten common words in U.S. NEWS:
[('said', 218), ('New', 133), ('California', 111), ('people', 95), ('Police',
92), ('Trump', 79), ('year', 68), ('COVID19', 68), ('Dead', 66), ('state', 64)]
Top ten common words in COMEDY:
[('Trump', 1297), ('Donald', 675), ('VIDEO', 499), ('Jimmy', 422), ('SNL', 422),
('Colbert', 406), ('Show', 357), ('Stephen', 325), ('Bill', 320), ('Maher',
279)]
Top ten common words in PARENTING:
[('child', 2372), ('Kids', 1667), ('kid', 1548), ('parent', 1374), ('time',
1132), ('one', 1106), ('Parents', 949), ('Mom', 949), ('like', 905), ('life',
847)]
Top ten common words in WORLD NEWS:
[('Trump', 741), ('said', 656), ('people', 623), ('country', 564), ('President',
535), ('North', 527), ('Korea', 525), ('China', 509), ('year', 493), ('New',
435)]
Top ten common words in ARTS & CULTURE:
[('Art', 507), ('art', 339), ('New', 310), ('PHOTOS', 254), ('artist', 246),
('Artist', 244), ('work', 231), ('year', 216), ('one', 214), ('new', 173)]
Top ten common words in SCIENCE & TECH:
[('New', 416), ('Apple', 413), ('Facebook', 340), ('Google', 288), ('new', 234),
('NASA', 209), ('Scientists', 203), ('Space', 198), ('May', 192), ('one', 190)]
Top ten common words in SPORTS:
[('NFL', 511), ('World', 276), ('NBA', 266), ('Olympic', 264), ('game', 258),
('team', 256), ('Game', 256), ('Football', 233), ('New', 225), ('Olympics',
```

```
211)]
Top ten common words in ENTERTAINMENT:
[('New', 1382), ('Trump', 1014), ('The', 723), ('new', 666), ('Star', 657),
('film', 656), ('one', 614), ('Show', 589), ('said', 579), ('year', 577)]
Top ten common words in POLITICS:
[('Trump', 14273), ('Donald', 4651), ('GOP', 2767), ('Clinton', 2632), ('said',
2502), ('House', 2444), ('Obama', 2403), ('president', 2033), ('New', 1918),
('Hillary', 1872)]
Top ten common words in WEIRD NEWS:
[('Man', 264), ('Woman', 122), ('Police', 121), ('one', 101), ('New', 100),
('Trump', 100), ('Dog', 95), ('said', 93), ('Watch', 87), ('Weird', 83)]
Top ten common words in ENVIRONMENT:
[('Climate', 554), ('climate', 349), ('Change', 288), ('change', 254), ('year',
246), ('New', 246), ('one', 221), ('week', 213), ('animal', 211), ('California',
208)]
Top ten common words in EDUCATION:
[('student', 419), ('school', 336), ('College', 313), ('college', 285),
('education', 231), ('Education', 230), ('Students', 191), ('School', 188),
('University', 184), ('teacher', 150)]
Top ten common words in CRIME:
[('Police', 619), ('Man', 447), ('said', 350), ('Shooting', 281), ('police',
247), ('say', 201), ('Cops', 197), ('New', 196), ('Killed', 184), ('Suspect',
178)]
Top ten common words in WELLNESS:
[('life', 2949), ('time', 2271), ('one', 2060), ('people', 1977), ('way', 1596),
('health', 1564), ('make', 1523), ('Health', 1459), ('year', 1425), ('like',
1380)]
Top ten common words in BUSINESS & FINANCES:
[('year', 596), ('company', 516), ('time', 467), ('one', 457), ('business',
444), ('people', 437), ('Business', 437), ('New', 423), ('new', 413), ('make',
359)]
Top ten common words in STYLE & BEAUTY:
[('PHOTOS', 4306), ('Style', 1807), ('Fashion', 1573), ('Week', 1171), ('New',
1046), ('Want', 957), ('look', 918), ('sure', 915), ('check', 912), ('Twitter',
868)]
Top ten common words in FOOD & DRINK:
[('Recipes', 938), ('Food', 694), ('PHOTOS', 692), ('food', 682), ('Best', 640),
('make', 596), ('Make', 575), ('one', 569), ('Day', 543), ('recipe', 492)]
Top ten common words in MEDIA:
[('Trump', 690), ('News', 558), ('Fox', 373), ('New', 333), ('Donald', 268),
('Media', 239), ('CNN', 203), ('York', 181), ('Times', 175), ('said', 171)]
Top ten common words in GROUPS VOICES:
[('Gay', 1430), ('Black', 1319), ('New', 952), ('people', 755), ('LGBT', 743),
('gay', 646), ('Trump', 583), ('year', 577), ('one', 572), ('said', 561)]
Top ten common words in HOME & LIVING:
[('PHOTOS', 1342), ('Home', 1081), ('home', 647), ('Day', 380), ('VIDEO', 369),
('Ideas', 342), ('DIY', 297), ('one', 290), ('Make', 286), ('House', 284)]
Top ten common words in WOMEN:
```

```
[('Women', 995), ('woman', 698), ('Woman', 221), ('one', 208), ('Trump', 200),
('time', 195), ('life', 192), ('year', 184), ('like', 179), ('Tweets', 161)]
Top ten common words in TRAVEL:
[('PHOTOS', 1619), ('Travel', 1085), ('one', 881), ('New', 823), ('travel',
778), ('World', 775), ('time', 685), ('Best', 644), ('world', 633), ('city',
624)]
Top ten common words in RELIGION:
[('Pope', 339), ('Francis', 226), ('Daily', 210), ('God', 209), ('people', 188),
('Church', 178), ('Muslim', 163), ('Christian', 149), ('one', 146), ('Jesus',
142)]
Top ten common words in IMPACT:
[('people', 388), ('year', 303), ('one', 300), ('world', 272), ('life', 270),
('Day', 269), ('woman', 255), ('child', 255), ('World', 236), ('time', 217)]
Top ten common words in WEDDINGS:
[('Wedding', 1712), ('wedding', 1297), ('Weddings', 530), ('Marriage', 503),
('couple', 368), ('one', 343), ('bride', 319), ('VIDEO', 308), ('Day', 307),
('day', 295)]
Top ten common words in MISCELLANEOUS:
[('one', 232), ('year', 222), ('life', 220), ('time', 218), ('like', 180),
('people', 163), ('would', 145), ('day', 139), ('Life', 133), ('make', 129)]
Top ten common words in DIVORCE:
[('Divorce', 1598), ('divorce', 870), ('marriage', 343), ('child', 321), ('one',
308), ('time', 303), ('year', 283), ('life', 277), ('relationship', 257),
('like', 242)]
```

3 Encoding Category Labels and Creating a Mapping Dictionary

[ ]:
```
encoder = LabelEncoder()  # Initializes a LabelEncoder object
df['categoryEncoded'] = encoder.fit_transform(df['category'])  # Encodes the␣
 ↪'category' column and stores the result in a new 'categoryEncoded' column
category_dict = dict(zip(encoder.classes_, encoder.transform(encoder.
 ↪classes_)))  # Creates a dictionary mapping the original category names to␣
 ↪their encoded values

# Prints the dictionary
print(category_dict)  # Prints the mapping of category names to their encoded␣
 ↪values
```

```
{'ARTS & CULTURE': 0, 'BUSINESS & FINANCES': 1, 'COMEDY': 2, 'CRIME': 3,
'DIVORCE': 4, 'EDUCATION': 5, 'ENTERTAINMENT': 6, 'ENVIRONMENT': 7, 'FOOD &
DRINK': 8, 'GROUPS VOICES': 9, 'HOME & LIVING': 10, 'IMPACT': 11, 'MEDIA': 12,
'MISCELLANEOUS': 13, 'PARENTING': 14, 'POLITICS': 15, 'RELIGION': 16, 'SCIENCE &
TECH': 17, 'SPORTS': 18, 'STYLE & BEAUTY': 19, 'TRAVEL': 20, 'U.S. NEWS': 21,
'WEDDINGS': 22, 'WEIRD NEWS': 23, 'WELLNESS': 24, 'WOMEN': 25, 'WORLD NEWS': 26}
```

[ ]:
```
def regular_encode(texts, tokenizer, maxlen=512):
    enc_di = tokenizer.batch_encode_plus(
        texts,
        return_token_type_ids=False,
        pad_to_max_length=True,
        max_length=maxlen
    )  # Encodes a batch of texts using the provided tokenizer, padding all␣
 ↪texts to a specified maximum length
    return np.array(enc_di['input_ids'])  # Returns the encoded texts as a␣
 ↪numpy array
```

4 Splitting Data into Train,Test,Validation Dataset

[ ]:
```
X_train, X_temp, y_train, y_temp = train_test_split(df['text'],␣
 ↪df['categoryEncoded'], random_state = 2020, test_size = 0.3)  # Splits the␣
 ↪data into a training set and a temporary set (for validation and testing)␣
 ↪with a 70-30 split
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, random_state =␣
 ↪2020, test_size = 0.5)  # Splits the temporary set into validation and␣
 ↪testing sets with a 50-50 split

#70 15 15
```

5 DistilBERT

5.1 Tokenization

[ ]:
```
# Tokenization
tokenizer = transformers.DistilBertTokenizer.
 ↪from_pretrained('distilbert-base-uncased')
# Initializes a DistilBert tokenizer with a pre-trained model
# The tokenizer returns dictionaries of input ids, attention masks, and token␣
 ↪type ids for each text
```

```
tokenizer_config.json:   0%|          | 0.00/28.0 [00:00<?, ?B/s]
vocab.txt:   0%|          | 0.00/232k [00:00<?, ?B/s]
tokenizer.json:   0%|          | 0.00/466k [00:00<?, ?B/s]
config.json:   0%|          | 0.00/483 [00:00<?, ?B/s]
```

[ ]:
```
"""
The tokenizer function is a tool that converts text into numerical␣
 ↪representations that can be fed into a machine learning model.
It usually performs tasks such as splitting the text into tokens, mapping the␣
 ↪tokens to unique ids, and adding special tokens such as [CLS] and [SEP].
```

```
[CLS] token stands for "classification" [SEP] token stands for "separation"
"""

Xtrain_encoded = regular_encode(X_train.astype('str'), tokenizer, maxlen=512)
# Encodes the training data using the 'regular_encode' function and the␣
 ↪tokenizer
Xval_encoded = regular_encode(X_val.astype('str'), tokenizer, maxlen=512)
# Encodes the validation data using the 'regular_encode' function and the␣
 ↪tokenizer
Xtest_encoded = regular_encode(X_test.astype('str'), tokenizer, maxlen=512)
# Encodes the testing data using the 'regular_encode' function and the tokenizer
```

```
Truncation was not explicitly activated but `max_length` is provided a specific
value, please use `truncation=True` to explicitly truncate examples to max
length. Defaulting to 'longest_first' truncation strategy. If you encode pairs
of sequences (GLUE-style) with the tokenizer you can select this strategy more
precisely by providing a specific strategy to `truncation`.
/usr/local/lib/python3.10/dist-
packages/transformers/tokenization_utils_base.py:2614: FutureWarning: The
`pad_to_max_length` argument is deprecated and will be removed in a future
version, use `padding=True` or `padding='longest'` to pad to the longest
sequence in the batch, or use `padding='max_length'` to pad to a max length. In
this case, you can give a specific length with `max_length` (e.g.
`max_length=45`) or leave max_length to None to pad to the maximal input size of
the model (e.g. 512 for Bert).
  warnings.warn(
```

[ ]:
```
ytrain_encoded = tf.keras.utils.to_categorical(y_train, num_classes=27,␣
 ↪dtype='int32')
# Converts the training labels to a binary matrix representation (one-hot␣
 ↪encoding)
yval_encoded = tf.keras.utils.to_categorical(y_val, num_classes=27,␣
 ↪dtype='int32')
# Converts the validation labels to a binary matrix representation (one-hot␣
 ↪encoding)
ytest_encoded = tf.keras.utils.to_categorical(y_test, num_classes=27,␣
 ↪dtype='int32')
# Converts the testing labels to a binary matrix representation (one-hot␣
 ↪encoding)
```

6 Building the Model with Transformer Layer and Compiling it

[ ]:
```
def build_model(transformer, loss='categorical_crossentropy', max_len=512):
    input_word_ids = tf.keras.layers.Input(shape=(max_len,), dtype=tf.int32,␣
 ↪name="input_word_ids")  # Input layer for the model
```

```
    sequence_output = transformer(input_word_ids)[0]  # Transformer layer that␣
↪processes the input
    cls_token = sequence_output[:, 0, :]  # Extracts the [CLS] token's outputs␣
↪for classification
    x = tf.keras.layers.Dropout(0.5)(cls_token)  # Dropout layer to prevent␣
↪overfitting
    x = tf.keras.layers.Dense(128, activation='relu')(x)  # Dense layer with␣
↪ReLU activation function
    out = tf.keras.layers.Dense(27, activation='softmax')(x)  # Output layer␣
↪with softmax activation function for multi-class classification
    model = tf.keras.Model(inputs=input_word_ids, outputs=out)  # Builds the␣
↪model
    optimizer = transformers.AdamWeightDecay(learning_rate=2e-5,␣
↪weight_decay_rate=0.01)  # Optimizer with weight decay for regularization
    f1_score = tfa.metrics.F1Score(num_classes=27, average='macro')  # F1 score␣
↪metric for multi-class classification
    model.compile(optimizer, loss=loss, metrics=[f1_score])  # Compiles the␣
↪model with the specified optimizer, loss function, and metrics
    return model  # Returns the model
```

[ ]:
```
with strategy.scope():
    transformer_layer = transformers.TFAutoModel.
↪from_pretrained('distilbert-base-uncased')  # Loads the pre-trained␣
↪DistilBERT model
    model = build_model(transformer_layer, max_len=512)  # Builds the model␣
↪using the build_model function and the transformer layer
model.summary()  # Prints a summary of the model
```

```
model.safetensors:   0%|          | 0.00/268M [00:00<?, ?B/s]
```

```
Some weights of the PyTorch model were not used when initializing the TF 2.0
model TFDistilBertModel: ['vocab_transform.bias', 'vocab_layer_norm.weight',
'vocab_projector.bias', 'vocab_transform.weight', 'vocab_layer_norm.bias']
- This IS expected if you are initializing TFDistilBertModel from a PyTorch
model trained on another task or with another architecture (e.g. initializing a
TFBertForSequenceClassification model from a BertForPreTraining model).
- This IS NOT expected if you are initializing TFDistilBertModel from a PyTorch
model that you expect to be exactly identical (e.g. initializing a
TFBertForSequenceClassification model from a BertForSequenceClassification
model).
All the weights of TFDistilBertModel were initialized from the PyTorch model.
If your task is similar to the task the model of the checkpoint was trained on,
you can already use TFDistilBertModel for predictions without further training.
```

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
```

```
==================================================================
 input_word_ids (InputLayer)  [(None, 512)]            0

 tf_distil_bert_model (TFDis  TFBaseModelOutput(last_h  66362880
 tilBertModel)               idden_state=(None, 512,
                             768),
                              hidden_states=None, att
                             entions=None)

 tf.__operators__.getitem (S  (None, 768)              0
 licingOpLambda)

 dropout_19 (Dropout)        (None, 768)              0

 dense (Dense)               (None, 128)              98432

 dense_1 (Dense)             (None, 27)               3483

==================================================================
Total params: 66,464,795
Trainable params: 66,464,795
Non-trainable params: 0
------------------------------------------------------------------
```

7 Creating TensorFlow Datasets for Training, Validation, and Testing

[ ]:
```
BATCH_SIZE = 32*strategy.num_replicas_in_sync  # Sets the batch size based on␣
 ↪the number of replicas in the strategy
#BATCH_SIZE = 6  # Choose an appropriate batch size for GPU training
AUTO = tf.data.experimental.AUTOTUNE  # Constant that represents automatic␣
 ↪tuning of performance-related options
# This constant is used to indicate that the decision for the optimal amount of␣
 ↪resources (like the number of threads, buffer sizes etc.) to allocate for␣
 ↪loading and preprocessing data should be left to TensorFlow.
# It dynamically adjusts these based on the system's available resources, thus␣
 ↪improving efficiency.

train_dataset = (
    tf.data.Dataset
    .from_tensor_slices((Xtrain_encoded, ytrain_encoded))  # Creates a dataset␣
 ↪from the training data and labels
    .repeat()  # Repeats the dataset indefinitely
    .shuffle(2048)  # Shuffles the dataset
    .batch(BATCH_SIZE)  # Batches the dataset
```

```
    .prefetch(AUTO)  # Prefetches elements from the dataset to improve␣
 ↪performance
)

val_dataset = (
    tf.data.Dataset
    .from_tensor_slices((Xval_encoded, yval_encoded))  # Creates a dataset from␣
 ↪the validation data and labels
    .batch(BATCH_SIZE)  # Batches the dataset
)

test_dataset = (
    tf.data.Dataset
    .from_tensor_slices(Xtest_encoded)  # Creates a dataset from the testing␣
 ↪data
    .batch(BATCH_SIZE)  # Batches the dataset
)

"""
The tf.data.Dataset.from_tensor_slices method is a way to create a TensorFlow␣
 ↪dataset from an array or a list of tensors.
It slices the input tensors along the first dimension and returns a dataset of␣
 ↪tensor slices.
Each slice has the same shape as the original tensor, except for the first␣
 ↪dimension, which is reduced by one.
For example, if you have a tensor of shape (3, 2), the method will return a␣
 ↪dataset of three slices, each of shape (2,).
You can use this method to create datasets from in-memory data that fit in␣
 ↪memory, such as images and labels. You can also use this method to create␣
 ↪datasets from other datasets by applying transformations.
"""
```

[ ]:
```
'\nThe tf.data.Dataset.from_tensor_slices method is a way to create a TensorFlow
dataset from an array or a list of tensors. \nIt slices the input tensors along
the first dimension and returns a dataset of tensor slices. \nEach slice has the
same shape as the original tensor, except for the first dimension, which is
reduced by one.\nFor example, if you have a tensor of shape (3, 2), the method
will return a dataset of three slices, each of shape (2,). \nYou can use this
method to create datasets from in-memory data that fit in memory, such as images
and labels. You can also use this method to create datasets from other datasets
by applying transformations.\n'
```

[ ]:
```
n_steps = Xtrain_encoded.shape[0] // BATCH_SIZE  # Calculates the number of␣
 ↪steps per epoch for the training data
```

[ ]:
```
# Use early stopping and model checkpointing
early_stopping = tf.keras.callbacks.EarlyStopping(patience=2,␣
 ↪restore_best_weights=True)  # Early stopping callback to stop training when␣
 ↪the model stops improving
model_checkpoint = tf.keras.callbacks.ModelCheckpoint('./drive/MyDrive/
 ↪best_model.h5', save_best_only=True)  # Model checkpoint callback to save␣
 ↪the best model during training

"""
EarlyStopping is a callback provided by Keras that can be used to stop the␣
 ↪training process if the model's performance has stopped improving on a␣
 ↪validation dataset.
In this case, 'patience=2' means that we will stop training if there is no␣
 ↪improvement in the model's validation loss for 2 consecutive epochs.
'restore_best_weights=True' means that the model weights from the epoch with␣
 ↪the best monitored metric (in this case, validation loss) will be restored.

ModelCheckpoint is another callback provided by Keras that can be used to save␣
 ↪the model at different points during training.
It can be configured to save the model after every epoch, only when the model␣
 ↪improves, or at specific intervals.
In this case, 'save_best_only=True' means that the latest best model according␣
 ↪to the monitored metric (in this case, validation loss) will not be␣
 ↪overwritten.
The advantage of using this callback is that you can resume training from the␣
 ↪saved models, which can be very helpful if a long-running training process␣
 ↪is interrupted.

The advantage of using these callbacks is that they can save computational␣
 ↪resources by stopping the training process early if the model is no longer␣
 ↪improving,
and they can ensure that the best model found during the training process is␣
 ↪saved and can be reused later.
"""
```

[ ]:
```
"\nEarlyStopping is a callback provided by Keras that can be used to stop the
training process if the model's performance has stopped improving on a
validation dataset. \nIn this case, 'patience=2' means that we will stop
training if there is no improvement in the model's validation loss for 2
consecutive epochs. \n'restore_best_weights=True' means that the model weights
from the epoch with the best monitored metric (in this case, validation loss)
will be restored.\n\nModelCheckpoint is another callback provided by Keras that
can be used to save the model at different points during training. \nIt can be
configured to save the model after every epoch, only when the model improves, or
at specific intervals. \nIn this case, 'save_best_only=True' means that the
latest best model according to the monitored metric (in this case, validation
```

loss) will not be overwritten. \nThe advantage of using this callback is that you can resume training from the saved models, which can be very helpful if a long-running training process is interrupted.\n\nThe advantage of using these callbacks is that they can save computational resources by stopping the training process early if the model is no longer improving, \nand they can ensure that the best model found during the training process is saved and can be reused later.\n"

[ ]:
```
lr_scheduler = tf.keras.callbacks.ReduceLROnPlateau(monitor='val_loss',␣
 ↪factor=0.2, patience=2, min_lr=1e-6)  # Learning rate scheduler callback to␣
 ↪reduce the learning rate when the validation loss stops improving

"""
The ReduceLROnPlateau callback is a way to reduce the learning rate when the␣
 ↪model's performance stops improving.
It monitors a specified metric (in this case, 'val_loss'), and if no␣
 ↪improvement is seen for a 'patience' number of epochs,
the learning rate is reduced by a factor (in this case, 0.2).
The learning rate will never be reduced below 'min_lr' (in this case, 1e-6).
This is useful in scenarios where the learning rate might be too high to allow␣
 ↪the model to converge,
and reducing it can help the model to continue improving.
"""
```

[ ]: "\nThe ReduceLROnPlateau callback is a way to reduce the learning rate when the model's performance stops improving. \nIt monitors a specified metric (in this case, 'val_loss'), and if no improvement is seen for a 'patience' number of epochs, \nthe learning rate is reduced by a factor (in this case, 0.2). \nThe learning rate will never be reduced below 'min_lr' (in this case, 1e-6).\nThis is useful in scenarios where the learning rate might be too high to allow the model to converge, \nand reducing it can help the model to continue improving.\n"

[ ]:
```
train_history = model.fit(
    train_dataset,  # Training data
    steps_per_epoch=n_steps,  # Number of steps per epoch
    validation_data=val_dataset,  # Validation data
    epochs=30,  # Number of epochs to train for
    callbacks=[early_stopping, model_checkpoint, lr_scheduler]  # List of␣
 ↪callbacks to apply during training
)  # Trains the model for a specified number of epochs and returns a History␣
 ↪object
```

```
Epoch 1/30
572/572 [==============================] - 304s 458ms/step - loss: 1.6398 -
f1_score: 0.4399 - val_loss: 1.1200 - val_f1_score: 0.5668 - lr: 2.0000e-05
Epoch 2/30
```

```
572/572 [==============================] - 254s 444ms/step - loss: 1.1136 -
f1_score: 0.5820 - val_loss: 1.0380 - val_f1_score: 0.5960 - lr: 2.0000e-05
Epoch 3/30
572/572 [==============================] - 256s 448ms/step - loss: 0.9807 -
f1_score: 0.6234 - val_loss: 0.9976 - val_f1_score: 0.6190 - lr: 2.0000e-05
Epoch 4/30
572/572 [==============================] - 255s 447ms/step - loss: 0.8731 -
f1_score: 0.6599 - val_loss: 0.9880 - val_f1_score: 0.6263 - lr: 2.0000e-05
Epoch 5/30
572/572 [==============================] - 248s 433ms/step - loss: 0.7796 -
f1_score: 0.6927 - val_loss: 1.0075 - val_f1_score: 0.6289 - lr: 2.0000e-05
Epoch 6/30
572/572 [==============================] - 267s 467ms/step - loss: 0.6915 -
f1_score: 0.7243 - val_loss: 1.0442 - val_f1_score: 0.6269 - lr: 2.0000e-05
```

8 Loading the Best Model, Making Predictions on the Test Set, and Calculating Evaluation Metrics

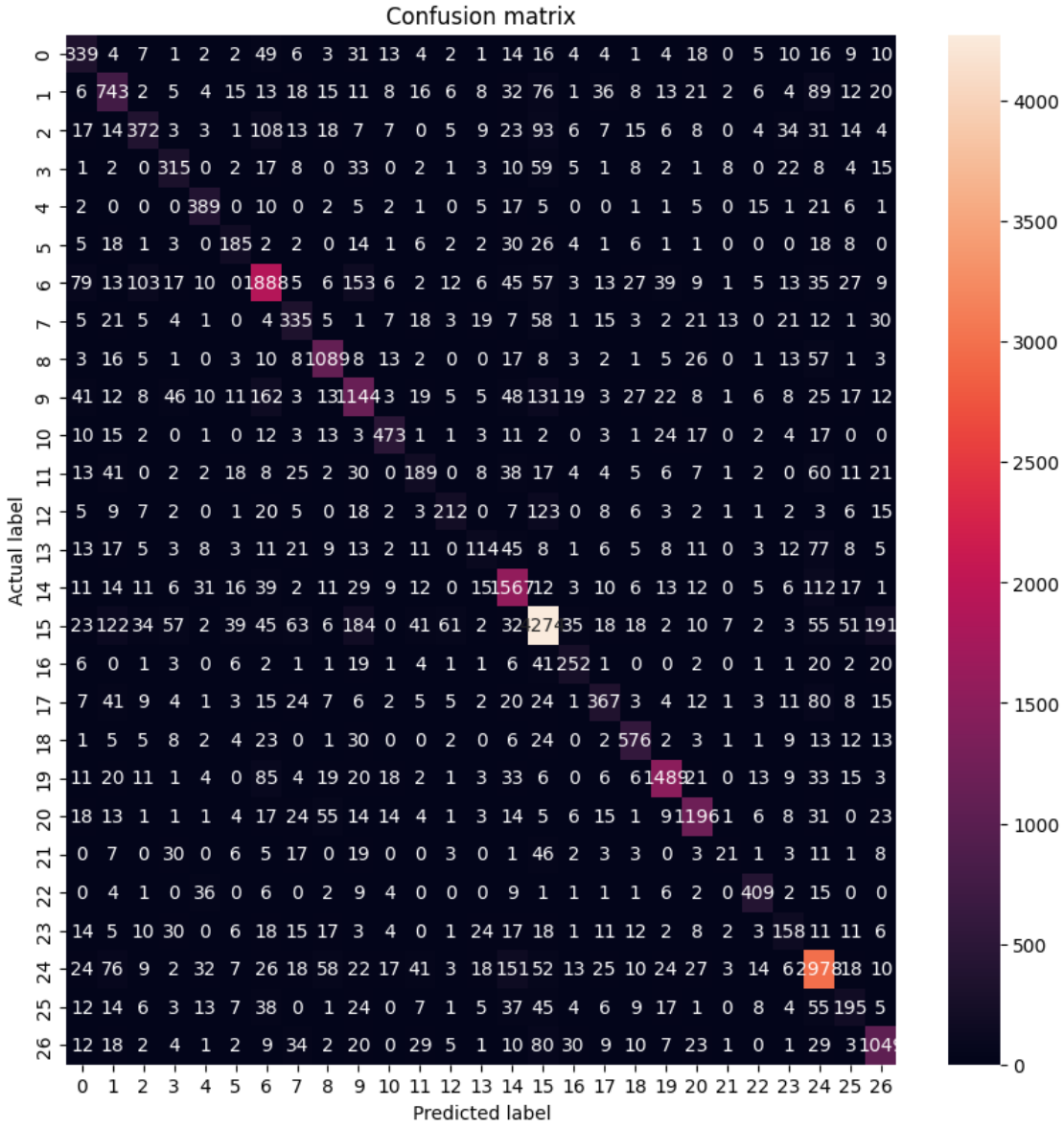
[ ]:
```python
from transformers import TFDistilBertModel, AdamWeightDecay
from tensorflow.keras.models import load_model
from tensorflow.keras.utils import custom_object_scope

with custom_object_scope({'TFDistilBertModel': TFDistilBertModel,␣
 ↪'AdamWeightDecay': AdamWeightDecay}):
    model = load_model('./drive/MyDrive/best_model.h5')  # Loads the best model␣
 ↪saved during training

# Predicts on the test set
preds = model.predict(test_dataset, verbose=1)
pred_classes = np.argmax(preds, axis=1)  # Gets the class with the highest␣
 ↪predicted probability for each example

# Calculates the metrics
print(f"Accuracy: {sklearn.metrics.accuracy_score(y_test, pred_classes)}")  #␣
 ↪Prints the accuracy of the model
print(f"Precision: {sklearn.metrics.precision_score(y_test, pred_classes,␣
 ↪average='macro')}")  # Prints the macro-averaged precision of the model
print(f"Recall: {sklearn.metrics.recall_score(y_test, pred_classes,␣
 ↪average='macro')}")  # Prints the macro-averaged recall of the model
print(f"F1-score: {sklearn.metrics.f1_score(y_test, pred_classes,␣
 ↪average='macro')}")  # Prints the macro-averaged F1 score of the model
```

```
123/123 [==============================] - 2657s 22s/step
Accuracy: 0.7101310932926054
Precision: 0.6336685983848206
Recall: 0.6161008706396148
```

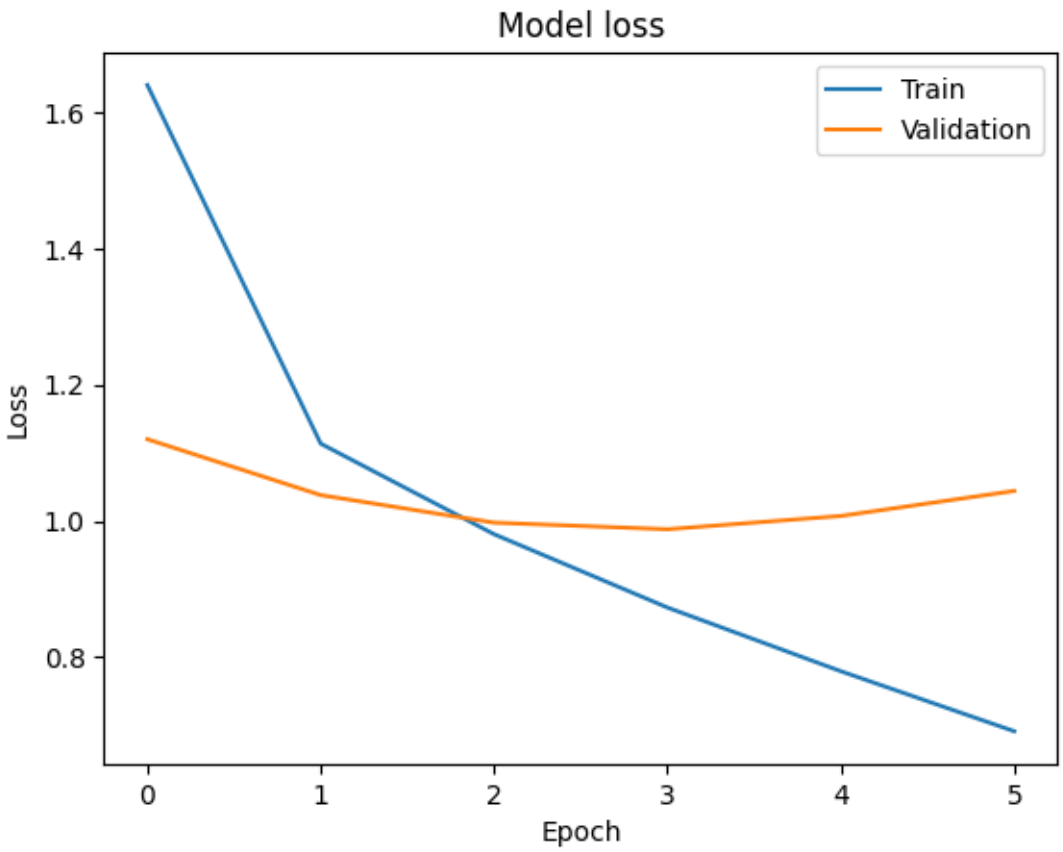
```
F1-score: 0.6202455346546011
```

9 Confusion Matrix

[ ]:
```python
from sklearn.metrics import confusion_matrix
import seaborn as sns

# Calculate the confusion matrix
cm = confusion_matrix(y_test, pred_classes)

# Display the confusion matrix
plt.figure(figsize=(10, 10))  # Sets the figure size
sns.heatmap(cm, annot=True, fmt="d")  # Plots the confusion matrix as a heatmap
plt.title('Confusion matrix')  # Sets the title of the plot
plt.ylabel('Actual label')  # Sets the label of the y-axis
plt.xlabel('Predicted label')  # Sets the label of the x-axis
plt.show()  # Displays the plot
```



[ ]:
```
# Plotting the training and validation loss
plt.plot(train_history.history['loss'])
plt.plot(train_history.history['val_loss'])
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc='upper right')
plt.show()  # Displays the plot
```



[ ]:
```
# Subtract the diagonal elements from the row sums to get the number of␣
↪incorrect predictions for each class
incorrect_predictions = cm.sum(axis=1) - np.diag(cm)

# Find the class with the most incorrect predictions
most_incorrect_class = np.argmax(incorrect_predictions)

print(f"The category that was predicted most incorrectly is: {encoder.
↪inverse_transform([most_incorrect_class])[0]}")
```

```
The category that was predicted most incorrectly is: POLITICS
```

[ ]:
```
print(f"Accuracy: {sklearn.metrics.accuracy_score(y_test, pred_classes)}")  #␣
↪Prints the accuracy of the model on the test data
print(f"Precision: {sklearn.metrics.precision_score(y_test, pred_classes,␣
↪average='micro')}")  # Prints the micro-averaged precision of the model on␣
↪the test data
print(f"Recall: {sklearn.metrics.recall_score(y_test, pred_classes,␣
↪average='micro')}")  # Prints the micro-averaged recall of the model on the␣
↪test data
```

```
print(f"F1-score: {sklearn.metrics.f1_score(y_test, pred_classes,␣
 ↪average='micro')}")  # Prints the micro-averaged F1 score of the model on␣
 ↪the test data
```

```
Accuracy: 0.7101310932926054
Precision: 0.7101310932926054
Recall: 0.7101310932926054
F1-score: 0.7101310932926054
```

[ ]:
```
print(f"Accuracy: {sklearn.metrics.accuracy_score(y_test, pred_classes)}")  #␣
 ↪Prints the accuracy of the model on the test data
print(f"Precision: {sklearn.metrics.precision_score(y_test, pred_classes,␣
 ↪average='weighted')}")  # Prints the weighted precision of the model on the␣
 ↪test data
print(f"Recall: {sklearn.metrics.recall_score(y_test, pred_classes,␣
 ↪average='weighted')}")  # Prints the weighted recall of the model on the␣
 ↪test data
print(f"F1-score: {sklearn.metrics.f1_score(y_test, pred_classes,␣
 ↪average='weighted')}")  # Prints the weighted F1 score of the model on the␣
 ↪test data
```

```
Accuracy: 0.7101310932926054
Precision: 0.7062991742141501
Recall: 0.7101310932926054
F1-score: 0.7063519606634536
```

10 convert h5 model format to tflite

[ ]:
```
import tensorflow as tf
from transformers import TFDistilBertModel  # Replace with your actual custom␣
 ↪layer
from transformers import optimization_tf as optimization  # Import the custom␣
 ↪optimizer

# Register the custom layer and optimizer before loading the model
tf.keras.utils.get_custom_objects()["TFDistilBertModel"] = TFDistilBertModel
tf.keras.utils.get_custom_objects()["AdamWeightDecay"] = optimization.
 ↪AdamWeightDecay

# Load the saved model in HDF5 format
model = tf.keras.models.load_model('./drive/MyDrive/best_model.h5')

# Convert the model to TensorFlow Lite format
converter = tf.lite.TFLiteConverter.from_keras_model(model)
tflite_model = converter.convert()
```

```
# Save the TensorFlow Lite model to a file
with open('converted_model.tflite', 'wb') as file:
    file.write(tflite_model)
```

```
WARNING:tensorflow:Skipping full serialization of Keras layer
<keras.layers.regularization.dropout.Dropout object at 0x7c52d87bed10>, because
it is not built.
WARNING:tensorflow:Skipping full serialization of Keras layer
<keras.layers.regularization.dropout.Dropout object at 0x7c52d724da80>, because
it is not built.
WARNING:tensorflow:Skipping full serialization of Keras layer
<keras.layers.regularization.dropout.Dropout object at 0x7c52d87bbd00>, because
it is not built.
WARNING:tensorflow:Skipping full serialization of Keras layer
<keras.layers.regularization.dropout.Dropout object at 0x7c52d87b8be0>, because
it is not built.
WARNING:tensorflow:Skipping full serialization of Keras layer
<keras.layers.regularization.dropout.Dropout object at 0x7c52d72292a0>, because
it is not built.
WARNING:tensorflow:Skipping full serialization of Keras layer
<keras.layers.regularization.dropout.Dropout object at 0x7c52d72b2650>, because
it is not built.
WARNING:absl:Found untraced functions such as serving, embeddings_layer_call_fn,
embeddings_layer_call_and_return_conditional_losses, transformer_layer_call_fn,
transformer_layer_call_and_return_conditional_losses while saving (showing 5 of
165). These functions will not be directly callable after loading.
```

2- EDA:

```
import pandas as pd
import tensorflow as tf
from tensorflow import keras
import numpy as np
import matplotlib.pyplot as plt
import os

from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix
from tensorflow.keras.layers import Embedding
import seaborn as sns
pd.set_option('display.max_colwidth', -1)
```

```
<ipython-input-55-6892832884e5>:14: FutureWarning: Passing a negative integer is deprecated in version 1.0 and will not be supported in future version. Instead, use None to not limit the column width.
  pd.set_option('display.max_colwidth', -1)
```

```
dataset = pd.read_json('/content/drive/MyDrive/topicDetection/News_Category_Dataset_v3.json', lines=True)
dataset.drop(['authors', 'link', 'date'], axis = 1, inplace = True)
dataset.head()
```

```
                                                                                headline  \
0  Over 4 Million Americans Roll Up Sleeves For Omicron-Targeted COVID Boosters
1  American Airlines Flyer Charged, Banned For Life After Punching Flight Attendant On Video
2  23 Of The Funniest Tweets About Cats And Dogs This Week (Sept. 17-23)
3  The Funniest Tweets From Parents This Week (Sept. 17-23)
4  Woman Who Called Cops On Black Bird-Watcher Loses Lawsuit Against Ex-Employer

    category  \
0  U.S. NEWS
1  U.S. NEWS
2  COMEDY
3  PARENTING
4  U.S. NEWS
```

```
short_description
0  Health experts said it is too early to predict whether demand would
match up with the 171 million doses of the new boosters the U.S.
ordered for the fall.
1  He was subdued by passengers and crew when he fled to the back of
the aircraft after the confrontation, according to the U.S. attorney's
office in Los Angeles.
2  "Until you have a dog you don't understand what could be eaten."

3  "Accidentally put grown-up toothpaste on my toddler’s toothbrush
and he screamed like I was cleaning his teeth with a Carolina Reaper
dipped in Tabasco sauce."
4  Amy Cooper accused investment firm Franklin Templeton of unfairly
firing her and branding her a racist after video of the Central Park
encounter went viral.
```

```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 209527 entries, 0 to 209526
Data columns (total 3 columns):
 #   Column             Non-Null Count   Dtype
---  ------             --------------   -----
 0   headline           209527 non-null  object
 1   category           209527 non-null  object
 2   short_description  209527 non-null  object
dtypes: object(3)
memory usage: 4.8+ MB
```

```
dataset.describe()
```

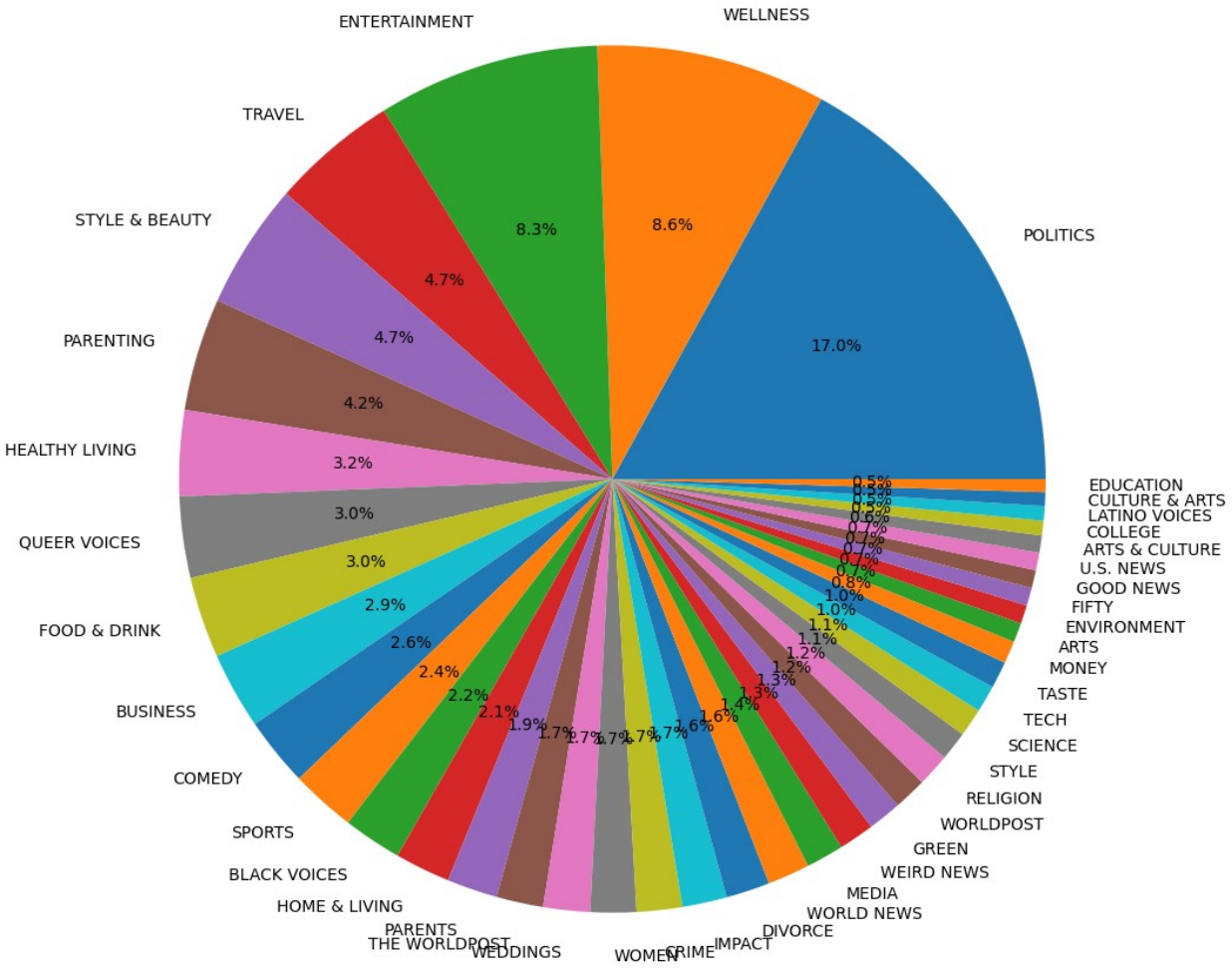
```
              headline  category short_description
count   209527           209527    209527
unique  207996           42        187022
top     Sunday Roundup   POLITICS
freq    90               35602     19712
```

```
print("We have a total of {}
categories".format(dataset['category'].nunique()))
dataset['category'].value_counts()
```

```
We have a total of 42 categories

POLITICS          35602
WELLNESS          17945
ENTERTAINMENT     17362
TRAVEL            9900
STYLE & BEAUTY    9814
PARENTING         8791
HEALTHY LIVING    6694
```

```
QUEER VOICES       6347
FOOD & DRINK       6340
BUSINESS           5992
COMEDY             5400
SPORTS             5077
BLACK VOICES       4583
HOME & LIVING      4320
PARENTS            3955
THE WORLDPOST      3664
WEDDINGS           3653
WOMEN              3572
CRIME              3562
IMPACT             3484
DIVORCE            3426
WORLD NEWS         3299
MEDIA              2944
WEIRD NEWS         2777
GREEN              2622
WORLDPOST          2579
RELIGION           2577
STYLE              2254
SCIENCE            2206
TECH               2104
TASTE              2096
MONEY              1756
ARTS               1509
ENVIRONMENT        1444
FIFTY              1401
GOOD NEWS          1398
U.S. NEWS          1377
ARTS & CULTURE     1339
COLLEGE            1144
LATINO VOICES      1130
CULTURE & ARTS     1074
EDUCATION          1014
Name: category, dtype: int64
```

```
fig = plt.figure(figsize=(12,12))
plt.pie(dataset['category'].value_counts().values,
        labels=dataset['category'].value_counts().index,
        autopct='%1.1f%%');
```




```
categories = dataset['category'].value_counts().index

def groupper(grouplist,name):
    for ele in categories:
        if ele in grouplist:
            dataset.loc[dataset['category'] == ele, 'category'] = name

groupper( grouplist= ['WELLNESS', 'HEALTHY LIVING','HOME &
LIVING','STYLE & BEAUTY' ,'STYLE'] , name =  'LIFESTYLE AND WELLNESS')

groupper( grouplist= [ 'PARENTING', 'PARENTS' ,'EDUCATION' ,'COLLEGE']
, name =  'PARENTING AND EDUCATION')

groupper( grouplist= ['SPORTS','ENTERTAINMENT' , 'COMEDY','WEIRD
NEWS','ARTS'] , name =  'SPORTS AND ENTERTAINMENT')

groupper( grouplist= ['TRAVEL', 'ARTS & CULTURE','CULTURE &
```

```
ARTS','FOOD & DRINK', 'TASTE'] , name =  'TRAVEL-TOURISM & ART-
CULTURE')

groupper( grouplist= ['WOMEN','QUEER VOICES', 'LATINO VOICES', 'BLACK
VOICES'] , name =  'EMPOWERED VOICES')

groupper( grouplist= ['BUSINESS' ,  'MONEY'] , name =  'BUSINESS-
MONEY')

groupper( grouplist= ['THE WORLDPOST' , 'WORLDPOST' , 'WORLD NEWS'] ,
name =  'WORLDNEWS')

groupper( grouplist= ['ENVIRONMENT' ,'GREEN'] , name =  'ENVIRONMENT')

groupper( grouplist= ['TECH', 'SCIENCE'] , name =  'SCIENCE AND TECH')

groupper( grouplist= ['FIFTY' , 'IMPACT' ,'GOOD NEWS','CRIME'] , name
=  'GENERAL')

groupper( grouplist= ['WEDDINGS', 'DIVORCE',  'RELIGION','MEDIA'] ,
name =  'MISC')

print("We have a total of {} categories
now".format(dataset['category'].nunique()))
dataset['category'].value_counts()
```

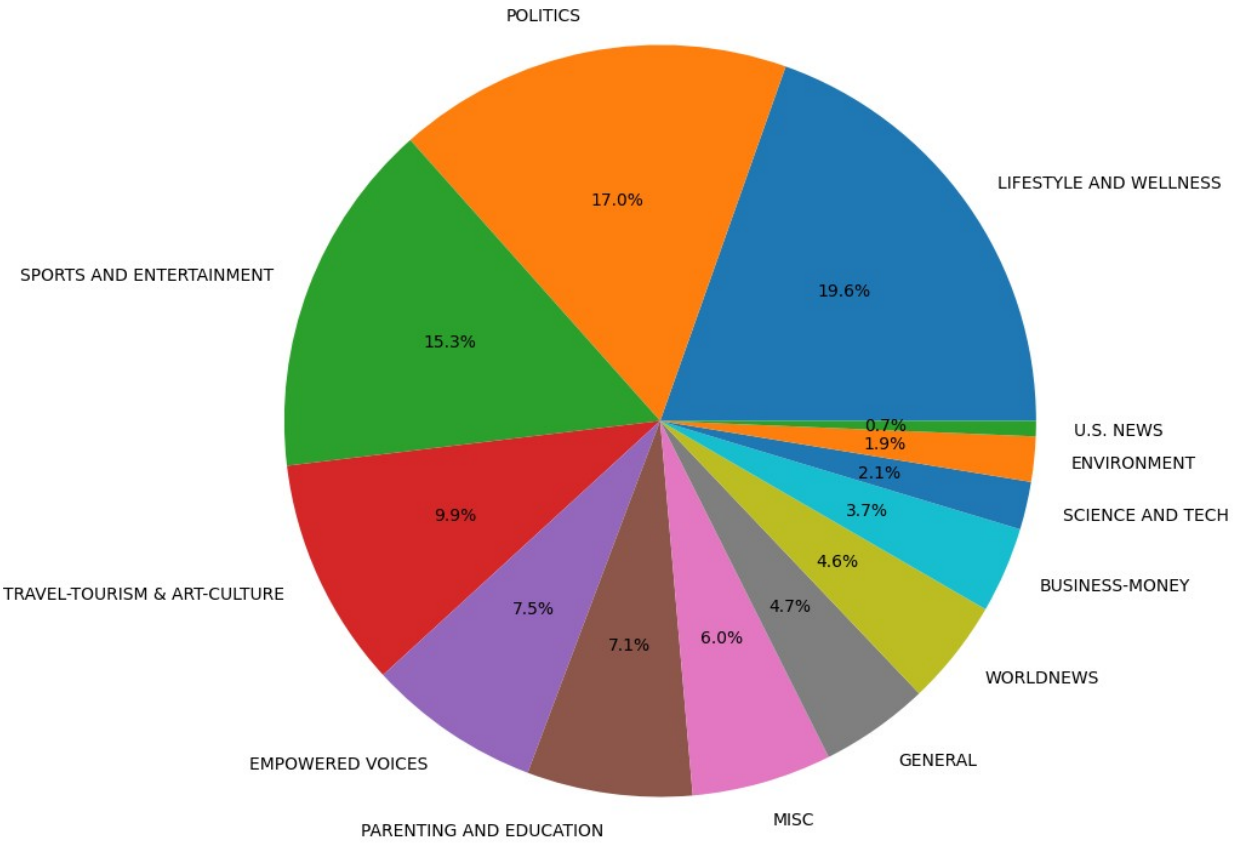
```
We have a total of 13 categories now
```

```
LIFESTYLE AND WELLNESS          41027
POLITICS                        35602
SPORTS AND ENTERTAINMENT        32125
TRAVEL-TOURISM & ART-CULTURE    20749
EMPOWERED VOICES                15632
PARENTING AND EDUCATION         14904
MISC                            12600
GENERAL                         9845
WORLDNEWS                       9542
BUSINESS-MONEY                  7748
SCIENCE AND TECH                4310
ENVIRONMENT                     4066
U.S. NEWS                       1377
Name: category, dtype: int64
```

```
dataset['category'].unique()
```

```
array(['U.S. NEWS', 'SPORTS AND ENTERTAINMENT', 'PARENTING AND
EDUCATION',
       'WORLDNEWS', 'TRAVEL-TOURISM & ART-CULTURE', 'SCIENCE AND
TECH',
       'POLITICS', 'ENVIRONMENT', 'GENERAL', 'LIFESTYLE AND WELLNESS',
       'BUSINESS-MONEY', 'MISC', 'EMPOWERED VOICES'], dtype=object)
```

```
fig = plt.figure(figsize=(10,20))
plt.pie(dataset['category'].value_counts().values,
        labels=dataset['category'].value_counts().index,
        autopct='%1.1f%%');
```




```
df = dataset.copy()

df.duplicated().sum()  #total duplicates

474

df.drop_duplicates(keep='last', inplace=True)

df.duplicated(subset=['short_description','headline']).sum()
#duplicates under 'short_description' and 'headline'

15

df.drop_duplicates(subset=['short_description','headline'],keep='last'
,inplace=True)
```

```
print(len(df[df['headline'] == ""]))
```

```
2
```

```
df.loc[df['headline'] == "", 'headline'] = np.nan
df.dropna(subset=['headline'], inplace=True)
print(len(df[df['headline'] == ""]))
```

```
0
```

```
print(len(df[df['short_description'] == ""]))
```

```
19610
```

```
df.loc[df['short_description'] == "", 'short_description'] = np.nan
df.dropna(subset=['short_description'], inplace=True)
print(len(df[df['short_description'] == ""]))
```

```
0
```

```
from sklearn.utils import shuffle
df = shuffle(df)
df.reset_index(inplace=True, drop=True)
```

```
df.head()
```

```
                                                headline  \
0  Trump's Ban On Trans People In The Armed Forces Is A Call To Arms

1  Women's Group Shines Light On Trump Sexual Assault Allegations
Before State Of The Union
2  Janet Napolitano Discusses How Schools Should Handle Campus Rape

3  A Photoshopped Picture Of Donald Trump Is Freaking Everyone Out

4  Barbara Corcoran, Real Estate Mogul And 'Shark Tank' Judge, On The
Challenge Of Selling Her Home (PHOTOS, VIDEO)

                    category  \
0  EMPOWERED VOICES
1  POLITICS
2  PARENTING AND EDUCATION
3  SPORTS AND ENTERTAINMENT
4  LIFESTYLE AND WELLNESS


                                short_description
0  President Donald Trump set the American LGBT community ablaze
Wednesday with a series of tweets that communicated his intent

1  Twenty-one women have accused the president of sexual misconduct.
```

```
2  University of California system President Janet Napolitano credited sexual assault survivors and their advocacy groups with raising the issue of rape on campus to a point where school leaders can no longer avoid the issue.
3  🤔🤔🤔

4  Take a look through our slideshow to see photos of Corcoran's charming estate and head over to Gillian Stewart Real Estate
```

```
df['desc'] = df['headline'].astype(str)+"-"+df['short_description']
df.drop(columns =['headline','short_description'],axis = 1, inplace=True)
df.astype(str)
df.head()
```

```
                    category  \
0  EMPOWERED VOICES
1  POLITICS
2  PARENTING AND EDUCATION
3  SPORTS AND ENTERTAINMENT
4  LIFESTYLE AND WELLNESS


desc
0  Trump's Ban On Trans People In The Armed Forces Is A Call To Arms-President Donald Trump set the American LGBT community ablaze Wednesday with a series of tweets that communicated his intent

1  Women's Group Shines Light On Trump Sexual Assault Allegations Before State Of The Union-Twenty-one women have accused the president of sexual misconduct.

2  Janet Napolitano Discusses How Schools Should Handle Campus Rape-University of California system President Janet Napolitano credited sexual assault survivors and their advocacy groups with raising the issue of rape on campus to a point where school leaders can no longer avoid the issue.
3  A Photoshopped Picture Of Donald Trump Is Freaking Everyone Out-🤔🤔🤔

4  Barbara Corcoran, Real Estate Mogul And 'Shark Tank' Judge, On The Challenge Of Selling Her Home (PHOTOS, VIDEO)-Take a look through our slideshow to see photos of Corcoran's charming estate and head over to Gillian Stewart Real Estate
```

2- Model training:

```
X,Y = df['desc'],df['category']

#80% to train , 10% for validation , 10% for testing
X_train, X_val, y_train, y_val = train_test_split(X,Y, test_size=0.2,
random_state=42)
X_val, X_test , y_val, y_test= train_test_split(X_val,y_val,
test_size=0.5, random_state=42)

vocab_size =20000
max_length = 150
trunc_type='post'
padding_type='post'
oov_tok = "<OOV>"

vocab_size =20000
max_length = 150
trunc_type='post'
padding_type='post'
oov_tok = "<OOV>"
tokenizer = tf.keras.preprocessing.text.Tokenizer(num_words =
vocab_size, oov_token=oov_tok)
tokenizer.fit_on_texts(X_train)

word_index = tokenizer.word_index

X_train = tokenizer.texts_to_sequences(X_train)
X_train = pad_sequences(X_train,maxlen=
max_length,padding=padding_type, truncating=trunc_type)
y_train = np.asarray(y_train)
y_train = pd.get_dummies(y_train)

X_val = tokenizer.texts_to_sequences(X_val)
X_val = pad_sequences(X_val,maxlen= max_length,padding=padding_type,
truncating=trunc_type)
y_val = np.asarray(y_val)
y_val = pd.get_dummies(y_val)

train_set = np.array(X_train)
val_set = np.array(X_val)

train_label = np.array(y_train)
val_label = np.array(y_val)


y_test = pd.get_dummies(y_test)
y_test = np.asarray(y_test)
y_test = np.argmax(y_test,axis=1)   #this would be our ground truth
label while testing
```

```
print(train_set.shape)
print(train_label.shape)


print(val_set.shape)
print(val_label.shape)
```

```
(151540, 150)
(151540, 13)
(18943, 150)
(18943, 13)
```

```
# prompt: save tokenizer
import pickle
with open('tokenizer.pickle', 'wb') as handle:
    pickle.dump(tokenizer, handle, protocol=pickle.HIGHEST_PROTOCOL)
```

```
!wget http://nlp.stanford.edu/data/glove.6B.zip -P
/content/drive/MyDrive/topicDetection
```

```
--2023-12-31 20:19:52--  http://nlp.stanford.edu/data/glove.6B.zip
Resolving nlp.stanford.edu (nlp.stanford.edu)... 171.64.67.140
Connecting to nlp.stanford.edu (nlp.stanford.edu)|171.64.67.140|:80...
connected.
HTTP request sent, awaiting response... 302 Found
Location: https://nlp.stanford.edu/data/glove.6B.zip [following]
--2023-12-31 20:19:52--  https://nlp.stanford.edu/data/glove.6B.zip
Connecting to nlp.stanford.edu (nlp.stanford.edu)|
171.64.67.140|:443... connected.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: https://downloads.cs.stanford.edu/nlp/data/glove.6B.zip
[following]
--2023-12-31 20:19:52--
https://downloads.cs.stanford.edu/nlp/data/glove.6B.zip
Resolving downloads.cs.stanford.edu (downloads.cs.stanford.edu)...
171.64.64.22
Connecting to downloads.cs.stanford.edu (downloads.cs.stanford.edu)|
171.64.64.22|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 862182613 (822M) [application/zip]
Saving to: ‘/content/drive/MyDrive/topicDetection/glove.6B.zip’

glove.6B.zip        100%[===================>] 822.24M  5.01MB/s    in
2m 39s

2023-12-31 20:22:31 (5.19 MB/s) -
‘/content/drive/MyDrive/topicDetection/glove.6B.zip’ saved
[862182613/862182613]
```

```
!unzip /content/drive/MyDrive/topicDetection/glove.6B.zip -d
/content/drive/MyDrive/topicDetection
```

```
Archive:  /content/drive/MyDrive/topicDetection/glove.6B.zip
  inflating: /content/drive/MyDrive/topicDetection/glove.6B.50d.txt
  inflating: /content/drive/MyDrive/topicDetection/glove.6B.100d.txt
  inflating: /content/drive/MyDrive/topicDetection/glove.6B.200d.txt
  inflating: /content/drive/MyDrive/topicDetection/glove.6B.300d.txt
```

```
path_to_glove_file =
'/content/drive/MyDrive/topicDetection/glove.6B.100d.txt'

#Initialising the embedding matrix with glove vec embeddings

num_tokens = len(tokenizer.word_index.items()) + 2
embedding_dim = 100
hits = 0
misses = 0


embeddings_index = {}
with open(path_to_glove_file) as f:
    for line in f:
        word, coefs = line.split(maxsplit=1)
        coefs = np.fromstring(coefs, "f", sep=" ")
        embeddings_index[word] = coefs

print("Found %s word vectors." % len(embeddings_index))


# Prepare embedding matrix
embedding_matrix = np.zeros((num_tokens, embedding_dim))
for word, i in tokenizer.word_index.items():
    embedding_vector = embeddings_index.get(word)
    if embedding_vector is not None:
        # Words not found in embedding index will be all-zeros.
        # This includes the representation for "padding" and "OOV"
        embedding_matrix[i] = embedding_vector
        hits += 1
    else:
        misses += 1
print("Converted %d words (%d misses)" % (hits, misses))
```

```
Found 400000 word vectors.
Converted 66483 words (39750 misses)
```

```
early_stop=tf.keras.callbacks.EarlyStopping(monitor='val_loss',
                                            patience=3,
min_delta=0.0001)
```

```
tf.keras.backend.clear_session()
embed_size = 100
model = keras.models.Sequential([

        Embedding(num_tokens,
        embedding_dim,

embeddings_initializer=keras.initializers.Constant(embedding_matrix),
        mask_zero=True,input_shape=[None],trainable=False),
        keras.layers.Bidirectional(keras.layers.LSTM(256, dropout =
0.4)),
        keras.layers.Dense(13, activation="softmax")

        ])


model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, None, 100)         10623500

 bidirectional (Bidirection  (None, 512)               731136
 al)

 dense (Dense)               (None, 13)                6669

=================================================================
Total params: 11361305 (43.34 MB)
Trainable params: 737805 (2.81 MB)
Non-trainable params: 10623500 (40.53 MB)
_________________________________________________________________
```

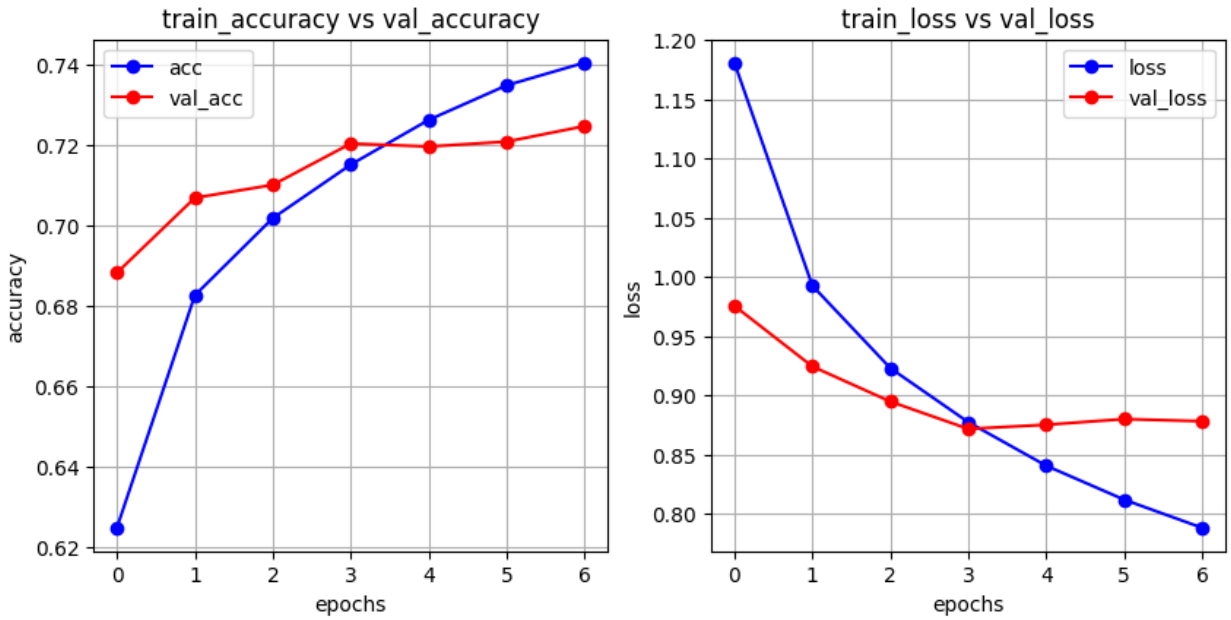
```
opt = keras.optimizers.Adam(learning_rate=0.001)
model.compile(loss="categorical_crossentropy", optimizer=opt,
metrics=["accuracy"])
history = model.fit( train_set,train_label,
                     batch_size = 32,
                     steps_per_epoch=len(X_train) // 32,
                     validation_data = (val_set , val_label),
                     validation_steps = len(val_set)//32, epochs=20,
                     callbacks=  early_stop )
```

```
Epoch 1/20
4735/4735 [==============================] - 100s 18ms/step - loss:
1.1807 - accuracy: 0.6248 - val_loss: 0.9756 - val_accuracy: 0.6881
Epoch 2/20
4735/4735 [==============================] - 81s 17ms/step - loss:
```

```
0.9926 - accuracy: 0.6825 - val_loss: 0.9244 - val_accuracy: 0.7067
Epoch 3/20
4735/4735 [==============================] - 84s 18ms/step - loss:
0.9230 - accuracy: 0.7017 - val_loss: 0.8948 - val_accuracy: 0.7100
Epoch 4/20
4735/4735 [==============================] - 79s 17ms/step - loss:
0.8774 - accuracy: 0.7150 - val_loss: 0.8717 - val_accuracy: 0.7202
Epoch 5/20
4735/4735 [==============================] - 80s 17ms/step - loss:
0.8406 - accuracy: 0.7261 - val_loss: 0.8752 - val_accuracy: 0.7195
Epoch 6/20
4735/4735 [==============================] - 79s 17ms/step - loss:
0.8120 - accuracy: 0.7347 - val_loss: 0.8800 - val_accuracy: 0.7207
Epoch 7/20
4735/4735 [==============================] - 78s 17ms/step - loss:
0.7882 - accuracy: 0.7403 - val_loss: 0.8782 - val_accuracy: 0.7246

model.save('/content/drive/MyDrive/topicDetection/model_lstm.h2')
```

3- Model Evaluation:

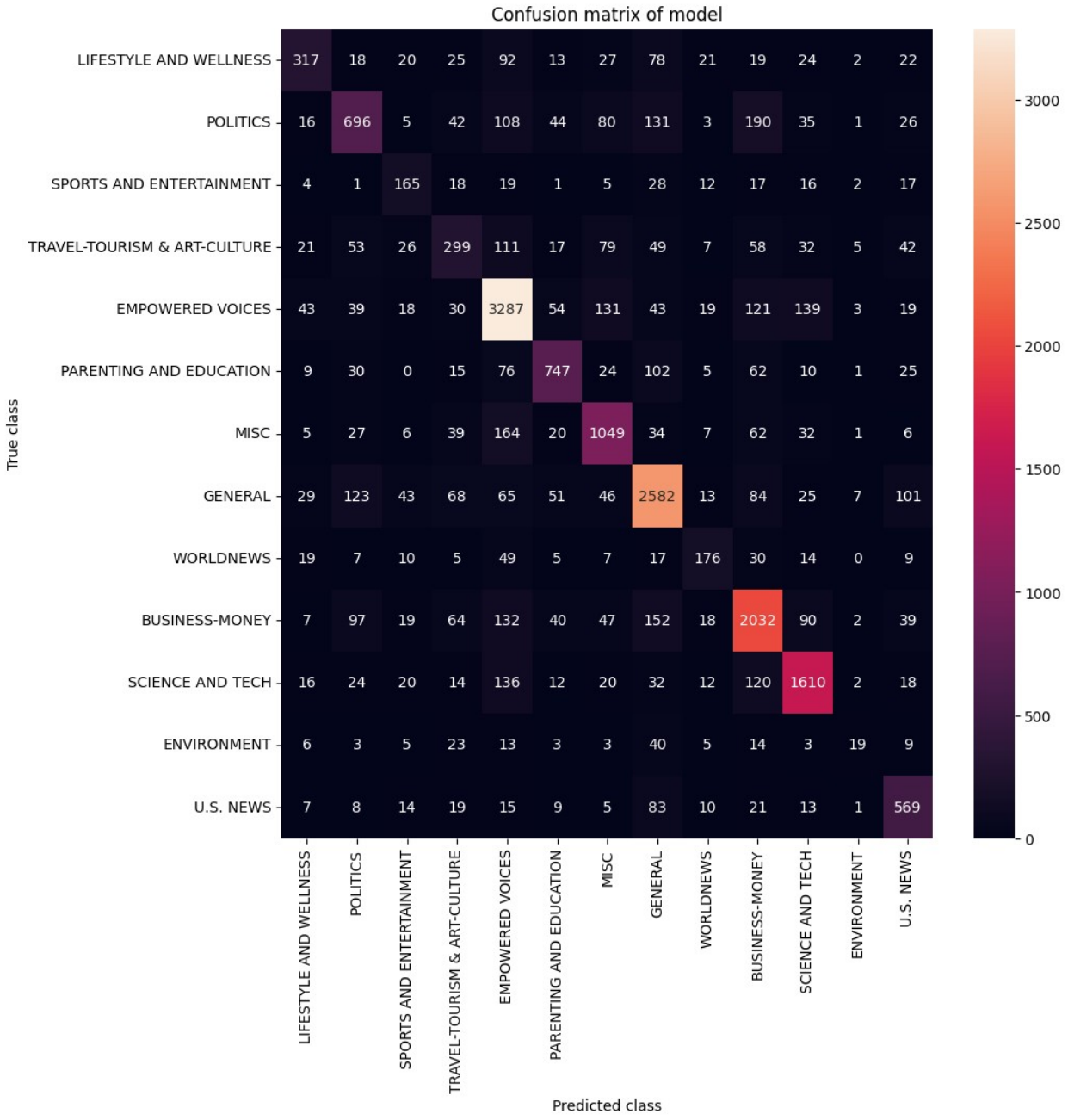
```
fig = plt.figure(figsize=(10,10))

# Plot accuracy
plt.subplot(221)
plt.plot(history.history['accuracy'],'bo-', label = "acc")
plt.plot(history.history['val_accuracy'], 'ro-', label = "val_acc")
plt.title("train_accuracy vs val_accuracy")
plt.ylabel("accuracy")
plt.xlabel("epochs")
plt.grid(True)
plt.legend()

# Plot loss function
plt.subplot(222)
plt.plot(history.history['loss'],'bo-', label = "loss")
plt.plot(history.history['val_loss'], 'ro-', label = "val_loss")
plt.title("train_loss vs val_loss")
plt.ylabel("loss")
plt.xlabel("epochs")
plt.grid(True)
plt.legend()

<matplotlib.legend.Legend at 0x7be0f0cc12a0>
```




```
classes = dataset['category'].value_counts().index

def prediction(inference_data):
    X = tokenizer.texts_to_sequences(inference_data)
    X = pad_sequences(X,maxlen= max_length,padding=padding_type,
truncating=trunc_type)
    pred = model.predict(X)
    pred_value = tf.argmax(pred,axis =1).numpy()
    pred_class = classes[pred_value]
    return pred_class,pred_value

y_pred_class, y_pred = prediction(X_test)

print(classification_report(np.asarray(y_test),np.asarray( y_pred)))
cf_matrix = confusion_matrix(y_test, y_pred)


plt.figure(figsize=(10,10))
heatmap = sns.heatmap(cf_matrix, xticklabels=classes,
                      yticklabels=classes,
                      annot=True, fmt='d', color='blue')
plt.xlabel('Predicted class')
plt.ylabel('True class')
plt.title('Confusion matrix of model')
```

	precision	recall	f1-score	support
0	0.64	0.47	0.54	678
1	0.62	0.51	0.56	1377
2	0.47	0.54	0.50	305
3	0.45	0.37	0.41	799

```
           4       0.77      0.83      0.80      3946
           5       0.74      0.68      0.70      1106
           6       0.69      0.72      0.71      1452
           7       0.77      0.80      0.78      3237
           8       0.57      0.51      0.54       348
           9       0.72      0.74      0.73      2739
          10       0.79      0.79      0.79      2036
          11       0.41      0.13      0.20       146
          12       0.63      0.74      0.68       774

    accuracy                           0.72     18943
   macro avg       0.64      0.60      0.61     18943
weighted avg       0.71      0.72      0.71     18943


Text(0.5, 1.0, 'Confusion matrix of model')
```



4- Model Predictions:

```
classes = ['U.S. NEWS', 'SPORTS AND ENTERTAINMENT', 'PARENTING AND
EDUCATION',
       'WORLDNEWS', 'TRAVEL-TOURISM & ART-CULTURE', 'SCIENCE AND
TECH',
       'POLITICS', 'ENVIRONMENT', 'GENERAL', 'LIFESTYLE AND WELLNESS',
       'BUSINESS-MONEY', 'MISC', 'EMPOWERED VOICES']
```

```
import tensorflow as tf


model_path = '/content/drive/MyDrive/topicDetection/model_lstm.h2'

# Load the model
loaded_model = tf.keras.models.load_model(model_path)

from pickle import load
tokenizer = load(open('/content/tokenizer.pickle', 'rb'))

from tensorflow.keras.preprocessing.sequence import pad_sequences
import tensorflow as tf

max_length = 150
trunc_type='post'
padding_type='post'

inference_data = "Politicians from different parties debated the new
environmental policy in the parliament."
X = tokenizer.texts_to_sequences([inference_data])
X = pad_sequences(X,maxlen= max_length,padding=padding_type,
truncating=trunc_type)
pred = loaded_model.predict(X)
pred_value = tf.argmax(pred,axis =1).numpy()
pred_class = classes[pred_value[0]]
print(pred_class)
```

```
1/1 [==============================] - 0s 34ms/step
POLITICS
```