Sarcastic Newspaper Headline Detector

Introduction

Sarcasm is a figure of speech or speech comment which is extremely difficult to define. It is often a statement or **comment which means the opposite of what it says.** It may be made with the intent of humour, or it may be made to be hurtful. The basic meaning is to be hostile under the cover of friendliness.

Sarcasm detection is a interestring application of natural language processing (NLP) and deep learning. Sarcasm is like a hidden treasure in the vast world of language. It adds a whole new level of complexity that can really test traditional language processing models. To truly understand sarcasm, you need to not only understand the literal meaning of words, but also appreciate the subtle nuances that can turn a simple statement into a sarcastic remark. As we venture into the realm of natural language processing, we dive into the exciting world of detecting sarcasm using the incredible power of deep learning. In this project, Aim is to build a robust sarcasm detection model using deep learning techniques.
The project involves various steps, including data analysis, data cleaning, model building, testing, and predicting user's inputs. From the very beginning, where we analyze data, to the final destination of creating a user-friendly model, we navigate through the ups and downs of integrating deep learning into the fascinating domain of linguistic wit.

In [276…
```
# import all the basic / required libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# Remove Warnings
import warnings
warnings.filterwarnings("ignore")
```

In [277…
```
# import tensorflow for model buliding
import tensorflow as tf
```

In [278…
```
from tensorflow.keras.preprocessing.text import Tokenizer
```

In [279…
```
# Load the dsataset
df = pd.read_json("sarcasm.json", lines= True)
```

In [280…
```
df.shape
```

Out[280]:
```
(28619, 3)
```

Data Cleaning

Cleaning the data is crucial to ensure the effectiveness of the model. This step involves handling missing values, removing irrelevant information, and addressing any noise in the dataset. Additionally, preprocessing steps like tokenization, removing stop words, and stemming/lemmatization may be applied to convert the raw text into a format suitable for deep learning models.

In [281…
```
df['headline'][3]
```

```
'inclement weather prevents liar from getting to work'
```

Out[281]:

In [282…
```
df
```

Out[282]:

	is_sarcastic	headline	article_link
0	1	thirtysomething scientists unveil doomsday clo...	https://www.theonion.com/thirtysomething-scien...
1	0	dem rep. totally nails why congress is falling...	https://www.huffingtonpost.com/entry/donna-edw...
2	0	eat your veggies: 9 deliciously different recipes	https://www.huffingtonpost.com/entry/eat-your-...
3	1	inclement weather prevents liar from getting t...	https://local.theonion.com/inclement-weather-p...
4	1	mother comes pretty close to using word 'strea...	https://www.theonion.com/mother-comes-pretty-c...
...	...	...	...
28614	1	jews to celebrate rosh hashasha or something	https://www.theonion.com/jews-to-celebrate-ros...
28615	1	internal affairs investigator disappointed con...	https://local.theonion.com/internal-affairs-in...
28616	0	the most beautiful acceptance speech this week...	https://www.huffingtonpost.com/entry/andrew-ah...
28617	1	mars probe destroyed by orbiting spielberg-gat...	https://www.theonion.com/mars-probe-destroyed-...
28618	1	dad clarifies this not a food stop	https://www.theonion.com/dad-clarifies-this-no...

28619 rows × 3 columns

Since the **df['article_link']** column is dispensable , will drop it from the dataset

In [283…
```
df = df.drop(labels='article_link', axis=1)
```

In [284…
```
df['is_sarcastic']
```

Out[284]:
```
0        1
1        0
2        0
3        1
4        1
        ..
28614    1
28615    1
28616    0
28617    1
28618    1
Name: is_sarcastic, Length: 28619, dtype: int64
```

In [285…
```
df['is_sarcastic'].value_counts()
```

Out[285]:
```
is_sarcastic
0    14985
1    13634
Name: count, dtype: int64
```

In [329…
```
df['is_sarcastic'].unique()
```

Out[329]:
```
array([1, 0], dtype=int64)
```

In [330…
```
df['headline']
```

Out[330]:
```
0        thirtysomething scientists unveil doomsday clo...
1        dem rep. totally nails why congress is falling...
2        eat your veggies: 9 deliciously different recipes
3        inclement weather prevents liar from getting t...
4        mother comes pretty close to using word 'strea...
                              ...
28614         jews to celebrate rosh hashasha or something
28615    internal affairs investigator disappointed con...
28616    the most beautiful acceptance speech this week...
28617    mars probe destroyed by orbiting spielberg-gat...
28618                  dad clarifies this not a food stop
Name: headline, Length: 28619, dtype: object
```

In [ ]:

In [ ]:

'is_sarcastic' coloumn of dataframe consist of labels/ output indicating whether respective headlines are sarcastic or not. Here df['is_sarcastic'] column have two viz. 1 and 0 where, **1** indicates that the given headline **is sarcastic** and **0** indicates headline is **not sarcastic.**

Before performing tokenization on original dataset i tried same process on the smaller dataset to understand it more throughly

I have created one sample list with four semtences in it.
I have followed following steps to tikenize the data

- Firstly we will try to tokenize the whole list using tensorflow's Tokenizer method. It will assign the index to each unique word in the list.
- In next step we will create index sequence list for the each sentance in the defined sample list.
- Next step to perform the paading on the obatined sequnces to avoid discrepancy in the results.

In [331…
```
# lets use natural language processing for string preprocessing

import nltk
```

In [332…
```
# removing stopwords
from nltk.corpus import stopwords
nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\HP\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping corpora\stopwords.zip.
```

Out[332]:
```
True
```

In [333…
```
eng_stopwords = set(stopwords.words())
```

In [ ]:
```
# Removing stopwords
#x_data = x.apply(lambda review : [ i       for i in review.split()       if i not in en
```

In [286…
```
sent = ["I am learning pythyon", 'I am learning deep learning', 'I love dogs', 'I love c
```

In [287…
```
type(sent)
```

Out[287]:
```
list
```

In [288…
```
sent
```

Out[288]:
```
['I am learning pythyon',
 'I am learning deep learning',
 'I love dogs',
 'I love cats']
```

In [289…
```
token = Tokenizer(10, )
```

In [290…
```
token.fit_on_texts(sent)
```

In [291…
```
token.word_index
```

Out[291]:
```
{'i': 1,
 'learning': 2,
 'am': 3,
 'love': 4,
 'pythyon': 5,
 'deep': 6,
 'dogs': 7,
 'cats': 8}
```

In [292…
```
sent_sq = token.texts_to_sequences(sent)
```

In [293…
```
print(sent_sq)
```

```
[[1, 3, 2, 5], [1, 3, 2, 6, 2], [1, 4, 7], [1, 4, 8]]
```

Model Building

The heart of the project lies in building a robust deep learning model for sarcasm detection. Common architectures for natural language processing tasks include recurrent neural networks (RNNs)and long short-term memory networks (LSTMs).

In [294…
```
from tensorflow.keras.preprocessing.sequence import pad_sequences
```

In [295…
```
sent_final = pad_sequences(sent_sq, maxlen=10, truncating= 'post', padding='post')
```

In [296…
```
sent_final
```

Out[296]:
```
array([[1, 3, 2, 5, 0, 0, 0, 0, 0, 0],
       [1, 3, 2, 6, 2, 0, 0, 0, 0, 0],
       [1, 4, 7, 0, 0, 0, 0, 0, 0, 0],
       [1, 4, 8, 0, 0, 0, 0, 0, 0, 0]])
```

In [ ]:
```
```

Now, time to tokenise the origanl dataset

In [297…
```
# for the ease , will convert the dataset into the list
hl = df['headline'].tolist()
```

In [298…
```
type(hl)
```

Out[298]:
```
list
```

In [299…
```
#hl
```

In [300…
```
labels = df['is_sarcastic'].to_list()
```

In [301…

```
labels
```

Out[301]:

```
[1,
 0,
 0,
 1,
 1,
 0,
 0,
 1,
 1,
 0,
 0,
 0,
 0,
 0,
 0,
 1,
 0,
 1,
 1,
 0,
 0,
 0,
 1,
 0,
 0,
 0,
 1,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 1,
 1,
 0,
 1,
 0,
 0,
 1,
 0,
 0,
 0,
 0,
 1,
 0,
 1,
 0,
 0,
 1,
 0,
 1,
 1,
 1,
 1,
 1,
 1,
 0,
 0,
 0,
 1,
 1,
 0,
 0,
 1,
```

1,
1,
0,
0,
1,
0,
0,
0,
0,
0,
1,
1,
0,
0,
0,
0,
0,
1,
0,
0,
0,
0,
1,
1,
1,
0,
0,
0,
1,
1,
1,
0,
0,
1,
0,
0,
0,
1,
1,
0,
0,
1,
0,
0,
1,
1,
1,
0,
0,
0,
1,
1,
0,
0,
0,
1,
0,
0,
1,
0,
1,
1,
1,
0,

1,
1,
0,
0,
0,
1,
1,
0,
1,
0,
0,
0,
0,
0,
1,
1,
0,
0,
1,
0,
0,
0,
1,
1,
1,
1,
0,
1,
0,
0,
1,
1,
0,
1,
1,
1,
0,
1,
0,
0,
1,
1,
0,
0,
0,
0,
1,
0,
0,
1,
1,
1,
1,
0,
0,
1,
1,
1,
1,
0,
1,
1,
0,
1,
0,
1,
0,
0,
0,
1,

0,
0,
1,
0,
0,
0,
0,
1,
0,
0,
1,
0,
1,
1,
1,
0,
0,
1,
0,
0,
0,
0,
1,
0,
1,
0,
0,
0,
1,
1,
0,
0,
0,
1,
1,
1,
0,
1,
0,
1,
1,
1,
1,
0,
0,
0,
0,
1,
1,
1,
1,
0,
1,
0,
1,
0,
1,
0,
1,
1,
1,
0,
0,
1,
1,
0,
0,

1,
0,
0,
0,
0,
1,
1,
0,
1,
1,
0,
0,
1,
1,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
1,
1,
1,
1,
0,
0,
0,
1,
1,
1,
0,
0,
1,
0,
1,
1,
0,
0,
1,
1,
1,
0,
0,
1,
1,
1,
1,
0,
0,
1,
0,
1,
1,
0,
1,
0,

0,
0,
0,
1,
1,
0,
1,
1,
0,
0,
0,
0,
1,
1,
0,
1,
1,
1,
1,
1,
1,
0,
0,
1,
0,
1,
0,
1,
0,
0,
0,
1,
1,
1,
1,
0,
1,
0,
0,
0,
0,
0,
0,
0,
1,
1,
1,
0,
0,
1,
1,
0,
1,
0,
1,
0,
0,
1,
0,
0,
0,
1,
1,
1,
0,
0,
0,
0,
1,
1,

0,
0,
0,
0,
1,
1,
0,
0,
1,
0,
1,
0,
0,
0,
1,
0,
0,
0,
0,
1,
0,
1,
1,
0,
1,
0,
0,
1,
1,
1,
1,
1,
0,
0,
1,
1,
1,
0,
0,
1,
1,
0,
1,
0,
1,
0,
0,
0,
1,
0,
0,
0,
1,
0,
1,
0,
1,
0,
1,
1,
0,
1,
1,
1,
1,
1,
0,
0,

0,
1,
1,
0,
0,
1,
0,
0,
1,
1,
1,
0,
1,
1,
1,
1,
0,
1,
0,
1,
0,
1,
0,
1,
1,
0,
1,
0,
0,
0,
0,
1,
0,
1,
1,
0,
1,
0,
1,
0,
0,
0,
1,
1,
1,
0,
0,
1,
0,
0,
0,
1,
0,
0,
0,
0,
1,
0,
1,
0,
0,
1,
0,
0,
0,
1,

1,
1,
0,
0,
0,
0,
0,
0,
1,
0,
0,
0,
1,
1,
0,
1,
1,
0,
1,
0,
0,
0,
1,
0,
0,
1,
0,
1,
0,
0,
0,
1,
0,
1,
1,
1,
1,
1,
0,
0,
1,
1,
0,
0,
1,
1,
1,
0,
0,
1,
1,
1,
1,
1,
1,
0,
1,
1,
1,
1,
1,
0,
0,
0,
0,
0,
0,
0,
0,
1,

1,
1,
0,
1,
0,
0,
0,
0,
0,
1,
1,
0,
1,
1,
1,
0,
1,
0,
0,
0,
0,
0,
0,
1,
0,
0,
1,
0,
1,
1,
0,
0,
1,
1,
0,
0,
0,
0,
0,
0,
0,
0,
1,
0,
0,
0,
0,
0,
0,
0,
1,
0,
1,
1,
1,
1,
0,
0,
1,
0,
0,
0,
0,
0,
0,
0,
1,
0,
0,
1,
0,
1,

0,
0,
1,
0,
1,
1,
0,
0,
0,
1,
0,
1,
1,
0,
1,
1,
0,
0,
1,
0,
0,
0,
0,
0,
0,
1,
1,
1,
0,
0,
0,
1,
1,
1,
0,
0,
1,
0,
0,
0,
0,
0,
0,
1,
0,
1,
0,
1,
1,
1,
1,
0,
0,
1,
0,
0,
0,
1,
0,
0,
0,
1,
1,
1,
1,
1,
0,
0,
0,

1,
0,
0,
0,
1,
1,
0,
1,
0,
1,
0,
1,
1,
1,
0,
1,
1,
0,
1,
1,
1,
0,
1,
0,
1,
0,
1,
1,
1,
0,
0,
1,
0,
1,
0,
0,
1,
1,
0,
1,
1,
1,
0,
0,
1,
0,
0,
0,
0,
0,
1,
0,
1,
1,
1,
1,
0,
0,
0,
0,
0,
0,
0,

1,
1,
0,
1,
0,
1,
0,
1,
1,
0,
0,
0,
0,
1,
0,
1,
1,
1,
1,
0,
1,
0,
1,
1,
0,
0,
0,
1,
1,
0,
1,
1,
0,
0,
1,
1,
1,
0,
1,
0,
1,
0,
1,
1,
0,
1,
0,
0,
1,
0,
0,
0,
1,
1,
0,
0,
0,
1,
1,
0,
1,
1,
0,

1,
0,
1,
1,
1,
1,
0,
0,
0,
0,
1,
1,
0,
1,
0,
0,
0,
0,
0,
0,
1,
0,
1,
0,
1,
1,
1,
1,
1,
1,
0,
1,
1,
0,
0,
1,
0,
0,
1,
0,
0,
1,
1,
1,
1,
1,
0,
0,
1,
1,
0,
0,
1,
0,
1,
0,
0,
0,
1,
0,
1,
1,
0,
0,
1,
1,
0,
0,

1,
1,
0,
1,
1,
1,
1,
0,
1,
1,
1,
0,
0,
1,
0,
0,
1,
1,
0,
1,
0,
0,
0,
0,
0,
1,
1,
1,
0,
1,
1,
0,
0,
1,
0,
0,
1,
1,
1,
1,
1,
1,
0,
1,
1,
1,
1,
1,
1,
0,
1,
1,
0,
1,
1,
1,
1,
1,
1,
0,
1,
0,
0,
1,
0,
0,
0,
1,
0,

```
 1,
 1,
 0,
 0,
 1,
 0,
 1,
 0,
 0,
 1,
 0,
 1,
 ...]
```

Model Training

Train the model using the preprocessed dataset. Split the data into training and validation sets to evaluate the model's performance during training. Utilize appropriate loss functions and optimization algorithms. Monitor key metrics such as accuracy,loss to assess the model's performance.

In [302…
```
# create training and testing dataset =>> 9:1
```

In [303…
```
train_ind = df.shape[0]*90//100
train_ind
```

Out[303]:
```
25757
```

In [304…
```
#training dataset
headlines_train = hl[ : train_ind]
labels_train = labels[ : train_ind]
```

In [305…
```
#testing dataset
headlines_test = hl[train_ind : ]
labels_test = labels[train_ind : ]
```

In [306…
```
# create word oindex
```

In [307…
```
token = Tokenizer(num_words=1000, oov_token= 'UNK' )
```

In [308…
```
token.fit_on_texts(headlines_train)
```

In [309…
```
token.word_index
```

Out[309]:
```
{'UNK': 1,
 'to': 2,
 'of': 3,
 'the': 4,
 'in': 5,
 'for': 6,
 'a': 7,
 'on': 8,
 'and': 9,
 'with': 10,
 'is': 11,
 'new': 12,
 'man': 13,
 'trump': 14,
 'at': 15,
 'from': 16,
 'about': 17,
 'by': 18,
```

```
'you': 19,
'after': 20,
'this': 21,
'be': 22,
'out': 23,
'up': 24,
'as': 25,
'that': 26,
'it': 27,
'how': 28,
'not': 29,
'he': 30,
'his': 31,
'what': 32,
'your': 33,
'are': 34,
'just': 35,
'who': 36,
'has': 37,
'all': 38,
'will': 39,
'report': 40,
'into': 41,
'more': 42,
'have': 43,
'one': 44,
'year': 45,
'over': 46,
'u': 47,
'why': 48,
'area': 49,
'woman': 50,
'can': 51,
'day': 52,
's': 53,
'says': 54,
'first': 55,
'time': 56,
'donald': 57,
'like': 58,
'no': 59,
'get': 60,
'her': 61,
'old': 62,
'off': 63,
'people': 64,
'life': 65,
"trump's": 66,
"'": 67,
'now': 68,
'house': 69,
'an': 70,
'obama': 71,
'white': 72,
'still': 73,
'back': 74,
'make': 75,
'was': 76,
'down': 77,
'than': 78,
'women': 79,
'if': 80,
'my': 81,
'could': 82,
'when': 83,
'clinton': 84,
```

```
'i': 85,
'5': 86,
'they': 87,
'way': 88,
'before': 89,
'world': 90,
'their': 91,
'americans': 92,
'him': 93,
'family': 94,
'we': 95,
'do': 96,
'only': 97,
'study': 98,
'would': 99,
'most': 100,
'black': 101,
'being': 102,
'gop': 103,
'so': 104,
'school': 105,
'years': 106,
'bill': 107,
'best': 108,
'know': 109,
'finds': 110,
'really': 111,
"it's": 112,
'should': 113,
'last': 114,
"can't": 115,
'3': 116,
'american': 117,
'nation': 118,
'watch': 119,
'she': 120,
'10': 121,
'but': 122,
'police': 123,
'going': 124,
'home': 125,
'video': 126,
'president': 127,
'death': 128,
'during': 129,
'good': 130,
'say': 131,
'state': 132,
'or': 133,
'show': 134,
'health': 135,
'against': 136,
'mom': 137,
'getting': 138,
'right': 139,
'campaign': 140,
'every': 141,
"'the": 142,
'too': 143,
'things': 144,
'big': 145,
'some': 146,
'000': 147,
'gets': 148,
'2': 149,
'party': 150,
```

```
'self': 151,
'work': 152,
'hillary': 153,
'while': 154,
'need': 155,
'parents': 156,
'may': 157,
'love': 158,
'never': 159,
'where': 160,
'little': 161,
'own': 162,
'through': 163,
'john': 164,
'child': 165,
'other': 166,
'take': 167,
'court': 168,
'kids': 169,
"doesn't": 170,
'calls': 171,
'these': 172,
'news': 173,
'makes': 174,
'next': 175,
'high': 176,
'change': 177,
'dead': 178,
'election': 179,
'local': 180,
'stop': 181,
'4': 182,
'7': 183,
"he's": 184,
'gay': 185,
"don't": 186,
'want': 187,
'see': 188,
'takes': 189,
'war': 190,
'our': 191,
'even': 192,
'go': 193,
'real': 194,
"here's": 195,
'look': 196,
'around': 197,
'its': 198,
'baby': 199,
'america': 200,
'them': 201,
'bush': 202,
"nation's": 203,
'guy': 204,
'made': 205,
'two': 206,
'office': 207,
'again': 208,
'got': 209,
'dog': 210,
'sex': 211,
'6': 212,
'million': 213,
'plan': 214,
'dad': 215,
'ever': 216,
```

```
'college': 217,
'much': 218,
'been': 219,
'help': 220,
'another': 221,
'debate': 222,
'finally': 223,
'week': 224,
'wants': 225,
'announces': 226,
'long': 227,
'job': 228,
'gun': 229,
'thing': 230,
'night': 231,
'reveals': 232,
'care': 233,
'there': 234,
'1': 235,
'live': 236,
'actually': 237,
'money': 238,
'under': 239,
'couple': 240,
'9': 241,
'us': 242,
'congress': 243,
'senate': 244,
'better': 245,
'shows': 246,
'sexual': 247,
"man's": 248,
'national': 249,
'god': 250,
'without': 251,
'everyone': 252,
'north': 253,
'trying': 254,
'any': 255,
'had': 256,
'star': 257,
'me': 258,
'facebook': 259,
'8': 260,
'paul': 261,
'bad': 262,
'top': 263,
'enough': 264,
'face': 265,
'anti': 266,
'shooting': 267,
'food': 268,
'ways': 269,
'season': 270,
'give': 271,
"won't": 272,
'climate': 273,
'teen': 274,
'making': 275,
'men': 276,
'20': 277,
'game': 278,
'media': 279,
'part': 280,
'history': 281,
'business': 282,
```

```
'movie': 283,
'free': 284,
'body': 285,
'supreme': 286,
'end': 287,
'away': 288,
'single': 289,
'york': 290,
'introduces': 291,
'city': 292,
'law': 293,
'story': 294,
'think': 295,
'fight': 296,
'son': 297,
'deal': 298,
'students': 299,
'pope': 300,
'second': 301,
'11': 302,
'friend': 303,
'already': 304,
'tell': 305,
'children': 306,
'friends': 307,
'attack': 308,
'releases': 309,
'fire': 310,
'tv': 311,
'car': 312,
'found': 313,
'same': 314,
'entire': 315,
'presidential': 316,
'girl': 317,
'must': 318,
'wedding': 319,
'power': 320,
'line': 321,
'company': 322,
'public': 323,
'book': 324,
'pretty': 325,
'former': 326,
'film': 327,
'run': 328,
'support': 329,
'great': 330,
'government': 331,
'having': 332,
'come': 333,
'sanders': 334,
'find': 335,
'talk': 336,
"didn't": 337,
'coming': 338,
'morning': 339,
'scientists': 340,
'unveils': 341,
'call': 342,
'room': 343,
'does': 344,
'behind': 345,
'social': 346,
'use': 347,
'name': 348,
```

```
'security': 349,
'doing': 350,
'between': 351,
'republican': 352,
'open': 353,
'photos': 354,
'middle': 355,
'keep': 356,
'student': 357,
'james': 358,
'looking': 359,
'might': 360,
'asks': 361,
'case': 362,
'because': 363,
'future': 364,
'fans': 365,
'email': 366,
'ceo': 367,
'republicans': 368,
'speech': 369,
'full': 370,
'fucking': 371,
'admits': 372,
'christmas': 373,
'group': 374,
'win': 375,
'thinks': 376,
'human': 377,
'poll': 378,
"world's": 379,
'claims': 380,
'12': 381,
'michael': 382,
'rights': 383,
'tax': 384,
'person': 385,
'put': 386,
'marriage': 387,
'control': 388,
'once': 389,
'voters': 390,
'team': 391,
'vote': 392,
'department': 393,
'violence': 394,
'2016': 395,
'something': 396,
'eating': 397,
'ryan': 398,
'country': 399,
'female': 400,
'killed': 401,
'ad': 402,
'forced': 403,
'always': 404,
'dies': 405,
'sure': 406,
'until': 407,
'used': 408,
'goes': 409,
'head': 410,
'ban': 411,
'inside': 412,
'hot': 413,
'father': 414,
```

```
'secret': 415,
'super': 416,
'plans': 417,
'judge': 418,
'bernie': 419,
'photo': 420,
'thousands': 421,
'political': 422,
'post': 423,
'democrats': 424,
'minutes': 425,
'water': 426,
'past': 427,
'save': 428,
'meet': 429,
'let': 430,
'each': 431,
'taking': 432,
'candidate': 433,
'living': 434,
'very': 435,
'many': 436,
'running': 437,
'left': 438,
'music': 439,
'here': 440,
'mother': 441,
'boy': 442,
'warns': 443,
'perfect': 444,
'race': 445,
'teacher': 446,
'list': 447,
'15': 448,
'pay': 449,
'george': 450,
'service': 451,
'missing': 452,
'were': 453,
'red': 454,
'days': 455,
'art': 456,
'summer': 457,
"'i": 458,
'wall': 459,
'times': 460,
'idea': 461,
'working': 462,
'california': 463,
"you're": 464,
'month': 465,
'heart': 466,
'reports': 467,
'states': 468,
'record': 469,
'tells': 470,
'secretary': 471,
'looks': 472,
'wife': 473,
'age': 474,
'wrong': 475,
'twitter': 476,
'start': 477,
'hours': 478,
'mike': 479,
'place': 480,
```

```
'everything': 481,
'class': 482,
'comes': 483,
'wearing': 484,
'lost': 485,
'lives': 486,
'set': 487,
'russia': 488,
'shot': 489,
'employee': 490,
'phone': 491,
'three': 492,
'texas': 493,
'needs': 494,
'yet': 495,
'town': 496,
'meeting': 497,
'ready': 498,
'someone': 499,
'50': 500,
'thought': 501,
'gives': 502,
'kill': 503,
'daughter': 504,
'obamacare': 505,
'young': 506,
'talks': 507,
'30': 508,
'iran': 509,
'believe': 510,
'did': 511,
'cat': 512,
'isis': 513,
'probably': 514,
'small': 515,
'shit': 516,
'drug': 517,
'prison': 518,
'chief': 519,
'cancer': 520,
'street': 521,
"i'm": 522,
'together': 523,
'cruz': 524,
'king': 525,
'ice': 526,
'giving': 527,
'kim': 528,
'internet': 529,
'dream': 530,
'letter': 531,
'fan': 532,
'democratic': 533,
'south': 534,
'justice': 535,
'ex': 536,
'earth': 537,
'crisis': 538,
'questions': 539,
'third': 540,
'talking': 541,
'korea': 542,
'few': 543,
'half': 544,
'officials': 545,
'breaking': 546,
```

```
'today': 547,
'nothing': 548,
'attacks': 549,
'girlfriend': 550,
'mark': 551,
'using': 552,
'less': 553,
'romney': 554,
'those': 555,
"she's": 556,
'personal': 557,
't': 558,
'hard': 559,
'months': 560,
'percent': 561,
'word': 562,
'chris': 563,
'wins': 564,
'watching': 565,
'administration': 566,
'feel': 567,
'director': 568,
'restaurant': 569,
'biden': 570,
'community': 571,
'air': 572,
'owner': 573,
'nuclear': 574,
'hour': 575,
'latest': 576,
'move': 577,
'system': 578,
'rock': 579,
"what's": 580,
"women's": 581,
'tips': 582,
'sleep': 583,
'gift': 584,
'abortion': 585,
'birthday': 586,
'leaves': 587,
'military': 588,
'outside': 589,
'whole': 590,
'order': 591,
'fbi': 592,
'buy': 593,
'education': 594,
'since': 595,
'kind': 596,
'kid': 597,
'waiting': 598,
'knows': 599,
'told': 600,
'special': 601,
'leave': 602,
'bar': 603,
'florida': 604,
'tweets': 605,
'well': 606,
'offers': 607,
'minute': 608,
'excited': 609,
'guide': 610,
'washington': 611,
'happy': 612,
```

```
'100': 613,
'different': 614,
'non': 615,
'called': 616,
'thinking': 617,
'ted': 618,
'lot': 619,
'box': 620,
'mueller': 621,
'march': 622,
'francis': 623,
'response': 624,
'following': 625,
'straight': 626,
'federal': 627,
'worried': 628,
'immigration': 629,
'stephen': 630,
'majority': 631,
'millions': 632,
'fox': 633,
'career': 634,
'muslim': 635,
'issues': 636,
'celebrates': 637,
'front': 638,
'store': 639,
'beautiful': 640,
'assault': 641,
'hit': 642,
'read': 643,
'online': 644,
'david': 645,
'cover': 646,
'reason': 647,
'late': 648,
'trailer': 649,
'2015': 650,
'russian': 651,
'ask': 652,
'problem': 653,
'himself': 654,
'anything': 655,
'rise': 656,
'rules': 657,
'billion': 658,
'leaders': 659,
'taylor': 660,
'fun': 661,
'hate': 662,
'congressman': 663,
'china': 664,
'drunk': 665,
'visit': 666,
'chinese': 667,
'feels': 668,
'birth': 669,
'huge': 670,
'series': 671,
'scott': 672,
'opens': 673,
'moment': 674,
'girls': 675,
'holiday': 676,
'investigation': 677,
'relationship': 678,
```

```
'spends': 679,
'senator': 680,
'hair': 681,
"isn't": 682,
'protest': 683,
"america's": 684,
'40': 685,
'travel': 686,
'break': 687,
'experts': 688,
'hollywood': 689,
'union': 690,
'al': 691,
'cop': 692,
'pence': 693,
'candidates': 694,
'victims': 695,
'accused': 696,
'die': 697,
'host': 698,
'whether': 699,
"obama's": 700,
'across': 701,
'huffpost': 702,
'play': 703,
'policy': 704,
'tom': 705,
'bring': 706,
'experience': 707,
'massive': 708,
'hands': 709,
'apple': 710,
'politics': 711,
'early': 712,
'starting': 713,
'light': 714,
'discover': 715,
'favorite': 716,
'center': 717,
'dating': 718,
'killing': 719,
'date': 720,
'totally': 721,
'stars': 722,
'leader': 723,
'weekend': 724,
'mass': 725,
'c': 726,
'hurricane': 727,
'trip': 728,
'k': 729,
'final': 730,
'united': 731,
'learned': 732,
'message': 733,
'fall': 734,
'iraq': 735,
'joe': 736,
'point': 737,
'struggling': 738,
'reasons': 739,
'j': 740,
'oil': 741,
'powerful': 742,
'clearly': 743,
'become': 744,
```

```
'least': 745,
'turns': 746,
"they're": 747,
'dance': 748,
'crash': 749,
'check': 750,
'adds': 751,
'key': 752,
'stand': 753,
'reality': 754,
'lessons': 755,
'sick': 756,
'signs': 757,
'worst': 758,
'true': 759,
'fuck': 760,
'role': 761,
'begins': 762,
'anniversary': 763,
'puts': 764,
'feeling': 765,
'lose': 766,
'abuse': 767,
'song': 768,
'completely': 769,
'fashion': 770,
'turn': 771,
'employees': 772,
'sports': 773,
'apartment': 774,
'moving': 775,
'adorable': 776,
'global': 777,
'keeps': 778,
'hoping': 779,
'almost': 780,
'returns': 781,
'wishes': 782,
'prince': 783,
'words': 784,
'vows': 785,
'hope': 786,
'jimmy': 787,
'sign': 788,
'un': 789,
'far': 790,
'driving': 791,
'dinner': 792,
'number': 793,
'hall': 794,
'announce': 795,
'space': 796,
'risk': 797,
'side': 798,
'longer': 799,
'return': 800,
'hand': 801,
'breaks': 802,
'amazon': 803,
'murder': 804,
'nfl': 805,
'robert': 806,
'lgbt': 807,
'mind': 808,
'seen': 809,
'syrian': 810,
```

```
'west': 811,
'd': 812,
'bus': 813,
'interview': 814,
'apologizes': 815,
'worth': 816,
'official': 817,
'syria': 818,
'kills': 819,
'oscar': 820,
'cops': 821,
'mental': 822,
'audience': 823,
'evidence': 824,
'low': 825,
'halloween': 826,
'press': 827,
'demands': 828,
'cut': 829,
'biggest': 830,
'five': 831,
'playing': 832,
'lead': 833,
'data': 834,
'coffee': 835,
'weird': 836,
'governor': 837,
'stage': 838,
'important': 839,
'accidentally': 840,
'conversation': 841,
'jr': 842,
'chance': 843,
'cool': 844,
'schools': 845,
'which': 846,
"there's": 847,
'reportedly': 848,
'suspect': 849,
'near': 850,
'users': 851,
'success': 852,
'decision': 853,
'program': 854,
'style': 855,
'elizabeth': 856,
'names': 857,
'reveal': 858,
'workers': 859,
'advice': 860,
'iowa': 861,
'football': 862,
'surprise': 863,
'13': 864,
'trans': 865,
'planned': 866,
'queer': 867,
'church': 868,
'asking': 869,
'test': 870,
'steve': 871,
'weight': 872,
'executive': 873,
'spot': 874,
'door': 875,
'oscars': 876,
```

```
'cnn': 877,
'kardashian': 878,
'university': 879,
'allegations': 880,
'learn': 881,
'boys': 882,
'defense': 883,
'williams': 884,
'anyone': 885,
'demand': 886,
'dying': 887,
'shop': 888,
'train': 889,
'blood': 890,
'paris': 891,
'possible': 892,
'2014': 893,
'hear': 894,
'members': 895,
'band': 896,
'rubio': 897,
'album': 898,
'uses': 899,
'apparently': 900,
'transgender': 901,
'building': 902,
'coworker': 903,
'pro': 904,
'awards': 905,
'plane': 906,
'brings': 907,
'major': 908,
'chicago': 909,
'queen': 910,
'reform': 911,
'general': 912,
'doctor': 913,
'eat': 914,
'grandma': 915,
'leading': 916,
'tour': 917,
'pick': 918,
'homeless': 919,
'economy': 920,
'14': 921,
'teens': 922,
'table': 923,
'died': 924,
'remember': 925,
'culture': 926,
'throws': 927,
'bowl': 928,
'easy': 929,
'bathroom': 930,
'prevent': 931,
'simple': 932,
'act': 933,
'urges': 934,
'moms': 935,
'michelle': 936,
'peace': 937,
'green': 938,
'explains': 939,
'hits': 940,
'push': 941,
'private': 942,
```

```
'voice': 943,
'pregnant': 944,
'suggests': 945,
'google': 946,
'suicide': 947,
'avoid': 948,
'supporters': 949,
'2017': 950,
'passes': 951,
'fear': 952,
'quietly': 953,
'christian': 954,
'try': 955,
'website': 956,
'address': 957,
'wait': 958,
'25': 959,
'road': 960,
'sound': 961,
'review': 962,
'leads': 963,
'beauty': 964,
'industry': 965,
'reading': 966,
'recalls': 967,
'spend': 968,
'staff': 969,
'fighting': 970,
'voting': 971,
'rest': 972,
'voter': 973,
"mother's": 974,
'picture': 975,
'carolina': 976,
'mayor': 977,
"hasn't": 978,
'hopes': 979,
'force': 980,
'racist': 981,
'epa': 982,
'walking': 983,
'finding': 984,
'families': 985,
'onion': 986,
'driver': 987,
'poor': 988,
'crowd': 989,
'happens': 990,
'reminds': 991,
'receives': 992,
'problems': 993,
'lets': 994,
'netflix': 995,
'card': 996,
'desperate': 997,
'magazine': 998,
'ben': 999,
'protesters': 1000,
...}
```

In [310…
```
train_seq = pad_sequences(token.texts_to_sequences(headlines_train), maxlen=50, padding=
```

In [311…
```
train_seq
```

Out[311]:
```
array([[  1, 340,   1, ...,   0,   0,   0],
```

```
       [  1,   1, 721, ...,   0,   0,   0],
       [914,  33,   1, ...,   0,   0,   0],
       ...,
       [  1,   1,   1, ...,   0,   0,   0],
       [212, 757, 464, ...,   0,   0,   0],
       [  1, 240,   1, ...,   0,   0,   0]])
```

In [312…]
```
test_seq = pad_sequences(token.texts_to_sequences(headlines_test), maxlen=50, padding='p
```

In [313…]
```
test_seq
```

Out[313]:
```
array([[ 32,   1, 390, ...,   0,   0,   0],
       [  1,   1,   1, ...,   0,   0,   0],
       [300, 623,   1, ...,   0,   0,   0],
       ...,
       [  4, 100, 640, ...,   0,   0,   0],
       [  1,   1,   1, ...,   0,   0,   0],
       [215,   1,  21, ...,   0,   0,   0]])
```

In [314…]
```
# convert labels into array
train_labels = np.array(labels_train)
test_labels = np.array(labels_test)
```

In [315…]
```
# build the model
```

In [316…]
```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Embedding, Flatten
```

In [317…]
```
model = Sequential()
```

In [318…]
```
# input layer
model.add(Embedding(1000, input_length=50, output_dim = 16))

#first hidden layer
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.25))

#second hidden layer
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.25))
model.add(Flatten())
#model.add(GlobalAveragePooling2D())
#output layer
model.add(Dense(1, activation='sigmoid'))
```

In [319…]
```
model.summary()
```

```
Model: "sequential_4"
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 embedding_6 (Embedding)      (None, 50, 16)            16000

 dense_15 (Dense)             (None, 50, 128)           2176

 dropout_10 (Dropout)         (None, 50, 128)           0

 dense_16 (Dense)             (None, 50, 64)            8256

 dropout_11 (Dropout)         (None, 50, 64)            0

 flatten_4 (Flatten)          (None, 3200)              0

 dense_17 (Dense)             (None, 1)                 3201
```

```
=================================================================
Total params: 29,633
Trainable params: 29,633
Non-trainable params: 0
_________________________________________________________________
```

In [320…
```
# compile model
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
```

Model Testing

After training, evaluate the model on a separate test set to ensure its generalization to unseen data. Analyze the confusion matrix and performance metrics to understand how well the model is distinguishing between sarcastic and non-sarcastic sentences.

In [321…
```
#train the model
model.fit(train_seq,train_labels, epochs=10, validation_data=(test_seq,test_labels))
```

```
Epoch 1/10
805/805 [==============================] - 11s 11ms/step - loss: 0.4465 - accuracy: 0.77
21 - val_loss: 0.3965 - val_accuracy: 0.8071
Epoch 2/10
805/805 [==============================] - 9s 11ms/step - loss: 0.3598 - accuracy: 0.835
5 - val_loss: 0.3767 - val_accuracy: 0.8246
Epoch 3/10
805/805 [==============================] - 9s 11ms/step - loss: 0.3478 - accuracy: 0.841
7 - val_loss: 0.3722 - val_accuracy: 0.8284
Epoch 4/10
805/805 [==============================] - 9s 11ms/step - loss: 0.3344 - accuracy: 0.848
3 - val_loss: 0.3674 - val_accuracy: 0.8312
Epoch 5/10
805/805 [==============================] - 8s 10ms/step - loss: 0.3253 - accuracy: 0.851
8 - val_loss: 0.3716 - val_accuracy: 0.8312
Epoch 6/10
805/805 [==============================] - 9s 11ms/step - loss: 0.3218 - accuracy: 0.853
6 - val_loss: 0.3733 - val_accuracy: 0.8309
Epoch 7/10
805/805 [==============================] - 8s 10ms/step - loss: 0.3148 - accuracy: 0.858
7 - val_loss: 0.3699 - val_accuracy: 0.8302
Epoch 8/10
805/805 [==============================] - 9s 11ms/step - loss: 0.3096 - accuracy: 0.860
3 - val_loss: 0.3719 - val_accuracy: 0.8267
Epoch 9/10
805/805 [==============================] - 9s 11ms/step - loss: 0.3051 - accuracy: 0.862
7 - val_loss: 0.3783 - val_accuracy: 0.8284
Epoch 10/10
805/805 [==============================] - 9s 11ms/step - loss: 0.2984 - accuracy: 0.866
3 - val_loss: 0.3740 - val_accuracy: 0.8239
```

Out[321]:
```
<keras.callbacks.History at 0x25789bd9f10>
```

Predicting User Inputs

In [322…
```
test = ['Where are women judges in highcourts ?']
test = pad_sequences(token.texts_to_sequences(test), maxlen=50, padding='post', truncati
```

In [323…
```
model.predict(test).round()
```

```
1/1 [==============================] - 0s 260ms/step
```

Out[323]:
```
array([[0.]], dtype=float32)
```

In [324…
```
test2 = ['teacher strikes idle kids']
test2 = pad_sequences(token.texts_to_sequences(test2), maxlen=50, padding='post',truncat
```

In [325…
```
model.predict(test2).round()
```

```
1/1 [==============================] - 0s 39ms/step
```
Out[325]:
```
array([[1.]], dtype=float32)
```

In [326…
```
int(model.predict(test2).round())
```

```
1/1 [==============================] - 0s 78ms/step
```
Out[326]:
```
1
```

Taking headline as an input from the user and predicting whether headline is sarcastic or not

Now that the model is trained and tested, it's time to deploy it for real-world use. Create a simple user interface to takeinputs from users. Tokenize and preprocess the user input, then feed it into the trained model for prediction. Provide users with a clear indication of whether the input is sarcastic or not.

In [328…
```
while True:
    head1 = []
    str1 = str(input("Enter headline for prediction (Type 'stop' to break the loop): "))
    # Check if the user wants to stop
    if str1.lower() == 'stop':
        break
    else:

        head1.extend([str1])
        head1 = pad_sequences(token.texts_to_sequences(head1), maxlen=50, padding='post'
        temp = (model.predict(head1)).round()
        #int(temp) = temp.round()
        #print(head1, type(head1))


        if int(temp) == 1:
            print("**************************************")
            print("Provided headline is Sarcastic")
            print("**************************************")
        else:
            print("**************************************")
            print("Provided headline is NOT Sarcastic")
            print("**************************************")
```

```
Enter headline for prediction (Type 'stop' to break the loop): War Dims Hope for Peace
1/1 [==============================] - 0s 65ms/step
**************************************
Provided headline is Sarcastic
**************************************
Enter headline for prediction (Type 'stop' to break the loop): Mahua Moitra: Expelled fr
om Parliament over cash-for-query row
1/1 [==============================] - 0s 47ms/step
**************************************
Provided headline is NOT Sarcastic
**************************************
Enter headline for prediction (Type 'stop' to break the loop): stop
```

In [ ]:
```
```

Sentences to test :

War Dims Hope for Peace

Mahua Moitra: Expelled from Parliament over cash-for-query row

Cold Wave Linked to Temperatures

In [ ]:

In [ ]: