In [2]:
```
import pandas as pd, numpy as np, json, re, pickle

from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
```

In [3]:
```
# convert to lower case
def to_lower(string: str) -> str:
    return string.lower()

# Load libraries
import unicodedata
import sys

def remove_puncs(text):

    # Create a dictionary of punctuation characters
    #punctuation = dict.fromkeys(i for i in range(sys.maxunicode)
                            #if unicodedata.category(chr(i)).startswith
('P'))

    #text = text.translate(punctuation)
    text=re.sub('</?.*?>',' <>', text)
    text=re.sub('\\d|\\W+|_',' ',text)
    text=re.sub('[^a-zA-Z]'," ", text)

    return text
```

# Preprocessing

In [4]:
```
file ='controversial-comments.jsonl'
# possible orient value: split, records, index, columns, and values)
# The following file is a subset of above file with the 1st 233537 lines.
comments = pd.read_json("controversial-comments_small.jsonl",orient="colu
mns",lines=True)
comments.head()
```

Out[4]:

| | con | txt |
|---|---|---|
| **0** | 0 | Well it's great that he did something about th... |
| **1** | 0 | You are right Mr. President. |
| **2** | 0 | You have given no input apart from saying I am... |
| **3** | 0 | I get the frustration but the reason they want... |
| **4** | 0 | I am far from an expert on TPP and I would ten... |

In [5]:
```
print('Size: ', len(comments), '\n',
      'Shape: ', comments.info(), '\n',
      'Categories: ', comments.con.unique())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 233538 entries, 0 to 233537
Data columns (total 2 columns):
 #   Column  Non-Null Count   Dtype 
---  ------  --------------   ----- 
 0   con     233538 non-null  int64 
 1   txt     233538 non-null  object
dtypes: int64(1), object(1)
memory usage: 3.6+ MB
Size:  233538
 Shape:  None
 Categories:  [0 1]
```

In [6]:
```
# since the size is humongus, I will take sample of the 2 categories.
# by trial, sample of 50000 from each category can be easily handled by m
y machine

size = 50000    # sample size
replace = True  # with replacement
fn = lambda obj: obj.loc[np.random.choice(obj.index, size, replace),:]

controversy = comments.groupby('con', as_index=False).apply(fn)
del comments
controversy.shape
```

Out[6]:
```
(100000, 2)
```

In [7]:
```
controversy['txt'] = controversy['txt'].apply(lambda x:to_lower(x))
controversy['txt'] = controversy['txt'].apply(lambda x:remove_puncs(x))

controversy.reset_index(drop=True, inplace=True)

controversy.head()
```

Out[7]:

| | con | txt |
|---|---|---|
| **0** | 0 | removed |
| **1** | 0 | gt then just treat these people as non person... |
| **2** | 0 | liberals are the only ones crying about rigged... |
| **3** | 0 | gt the power of nuclear the devastation is ve... |
| **4** | 0 | he wanted to abolish the backbone of americas ... |

In [8]:
```
controversy['txt'] = [word_tokenize(string) for string in controversy['txt']]
controversy.head()
```

Out[8]:

| | con | txt |
|---|---|---|
| **0** | 0 | [removed] |
| **1** | 0 | [gt, then, just, treat, these, people, as, non... |
| **2** | 0 | [liberals, are, the, only, ones, crying, about... |
| **3** | 0 | [gt, the, power, of, nuclear, the, devastation... |
| **4** | 0 | [he, wanted, to, abolish, the, backbone, of, a... |

In [9]:
```
# Load stop words
stop_words = stopwords.words('english')
controversy['txt'] = controversy['txt'].apply(lambda x: [item for item in x if item not in stop_words])
controversy.head()
```

Out[9]:

| | con | txt |
|---|---|---|
| **0** | 0 | [removed] |
| **1** | 0 | [gt, treat, people, non, persons, treat, say, ... |
| **2** | 0 | [liberals, ones, crying, rigged, systems, cons... |
| **3** | 0 | [gt, power, nuclear, devastation, important, h... |
| **4** | 0 | [wanted, abolish, backbone, americas, economy,... |

In [10]:
```
controversy.to_pickle('./Data/pkld_controversy.pkl')
```

In [11]:
```
X=controversy.txt.values
y=controversy.con.values
```

In [12]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_state=40)
print(X_train.shape)
print(X_test.shape)
```

```
(70000,)
(30000,)
```

In [13]:

```
# Load libraries
import numpy as np
from keras.datasets import imdb
from keras.preprocessing.text import Tokenizer
from keras import models
from keras import layers

# Set random seed
np.random.seed(0)

# Set the number of features we want
number_of_features = 100

# Load data and target vector from IMDB movie data
#(data_train, target_train), (data_test, target_test) = imdb.load_data(
                #num_words=number_of_features)

# Convert IMDB data to a one-hot encoded feature matrix
tokenizer = Tokenizer(num_words=number_of_features)
features_train = tokenizer.sequences_to_matrix(X_train, mode="freq")
features_test = tokenizer.sequences_to_matrix(X_test, mode="freq")

# Start neural network
network = models.Sequential()

# Add fully connected layer with a ReLU activation function
network.add(layers.Dense(units=16,
                         activation="relu",
                         input_shape=(number_of_features,)))

# Add fully connected layer with a ReLU activation function
network.add(layers.Dense(units=16, activation="relu"))

# Add fully connected layer with a sigmoid activation function
network.add(layers.Dense(units=1, activation="sigmoid"))

# Compile neural network
network.compile(loss="binary_crossentropy", # Cross-entropy
                optimizer="rmsprop", # Root Mean Square Propagation
                metrics=["accuracy"]) # Accuracy performance metric

# Train neural network
history = network.fit(features_train, # Features
                      y_train, # Target vector
                      epochs=3, # Number of epochs
                      verbose=0, # No output
                      batch_size=100, # Number of observations per batch
                      validation_data=(features_test, y_test)) # Test dat
a

# Predict classes of test set
predicted_target = network.predict(features_test)

predicted_target
```

```
Using TensorFlow backend.
```

```
---------------------------------------------------------------------
--
TypeError                                 Traceback (most recent call las
t)
<ipython-input-13-9559053e609b> in <module>
     18 # Convert IMDB data to a one-hot encoded feature matrix
     19 tokenizer = Tokenizer(num_words=number_of_features)
---> 20 features_train = tokenizer.sequences_to_matrix(X_train, mode="fre
q")
     21 features_test = tokenizer.sequences_to_matrix(X_test, mode="freq"
)
     22

c:\users\safar\documents\github\safarie1103\bellevue university\courses\d
sc550\week9and10\venv\lib\site-packages\keras_preprocessing\text.py in se
quences_to_matrix(self, sequences, mode)
    417             counts = defaultdict(int)
    418             for j in seq:
--> 419                 if j >= num_words:
    420                     continue
    421                 counts[j] += 1

TypeError: '>=' not supported between instances of 'str' and 'int'
```

In [14]:
```
tokenizer.sequences_to_matrix?
```

In [ ]:
```

```