Lab 4: RNN model for Text classification

In [1]:
```
# importing the dataset
import pandas as pd
dataset = pd.read_json('News_Category_Dataset_v3.json', lines=True)
dataset.head()
```

Out[1]:

	link	headline	category	short_description	authors	date
0	https://www.huffpost.com/entry/covid-boosters-...	Over 4 Million Americans Roll Up Sleeves For O...	U.S. NEWS	Health experts said it is too early to predict...	Carla K. Johnson, AP	2022-09-23
1	https://www.huffpost.com/entry/american-airlin...	American Airlines Flyer Charged, Banned For Li...	U.S. NEWS	He was subdued by passengers and crew when he ...	Mary Papenfuss	2022-09-23
2	https://www.huffpost.com/entry/funniest-tweets...	23 Of The Funniest Tweets About Cats And Dogs ...	COMEDY	"Until you have a dog you don't understand wha...	Elyse Wanshel	2022-09-23
3	https://www.huffpost.com/entry/funniest-parent...	The Funniest Tweets From Parents This Week (Se...	PARENTING	"Accidentally put grown-up toothpaste on my to...	Caroline Bologna	2022-09-23
4	https://www.huffpost.com/entry/amy-cooper-lose...	Woman Who Called Cops On Black Bird-Watcher Lo...	U.S. NEWS	Amy Cooper accused investment firm Franklin Te...	Nina Golgowski	2022-09-22

Data Cleaning & Preprocessing

In [2]:
```
dataset['News'] = dataset['headline'] + dataset['short_description']
dataset.drop(['headline','short_description','link','authors','date'], inplace=True, axis=1)
dataset['len_news'] = dataset['News'].map(lambda x: len(x))
dataset.head()
```

Out[2]:

	category	News	len_news
0	U.S. NEWS	Over 4 Million Americans Roll Up Sleeves For O...	230
1	U.S. NEWS	American Airlines Flyer Charged, Banned For Li...	248
2	COMEDY	23 Of The Funniest Tweets About Cats And Dogs ...	133
3	PARENTING	The Funniest Tweets From Parents This Week (Se...	215
4	U.S. NEWS	Woman Who Called Cops On Black Bird-Watcher Lo...	233

- Remove extra whitespaces
- Remove mentions (@username)
- Remove any text inside square brackets []
- Remove digits
- Remove any character that is not a letter, number, or whitespace
- Remove URLs, mentions, and hashtags (improve regex)
- Convert text to lowercase
- Remove stop words
- Lemmatization

In [3]:

```
import re
import nltk
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
nltk.download('stopwords')
nltk.download('wordnet')


STOPWORDS = set(stopwords.words('english'))

def datacleaning(text):
    text = re.sub(r'\s+', ' ', text)
    text = re.sub(r'(?i)@[a-z0-9_]+', '', text)
    text = re.sub(r'\[.*?\]', '', text)
    text = re.sub(r'\d+', '', text)
    text = re.sub(r'[^a-zA-Z0-9\s]', '', text)
```

```
    text = re.sub(r'(?:@[\w_]+|#\w+|http\S+)', '', text)
    text = text.lower()

    # Remove stop words
    text = [word for word in text.split() if word not in STOPWORDS]

    # Lemmatization
    lemmatizer = WordNetLemmatizer()
    sentence = [lemmatizer.lemmatize(word, 'v') for word in text]

    return ' '.join(sentence)

dataset['News'] = dataset['News'].apply(datacleaning)
dataset.head()
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\alens\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\alens\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
```

Out[3]:

	category	News	len_news
0	U.S. NEWS	million americans roll sleeves omicrontargeted...	230
1	U.S. NEWS	american airlines flyer charge ban life punch ...	248
2	COMEDY	funniest tweet cat dog week sept dog dont unde...	133
3	PARENTING	funniest tweet parent week sept accidentally p...	215
4	U.S. NEWS	woman call cop black birdwatcher lose lawsuit ...	233

In [4]:

```
# Import necessary libraries
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from sklearn.preprocessing import LabelEncoder
import numpy as np
```

In [5]:

```
# Define tokenizer
max_words = 100000
```

```
tokenizer = Tokenizer(num_words=max_words, oov_token="")

tokenizer.fit_on_texts(dataset['News'])

# print(tokenizer.word_counts)
sequences = tokenizer.texts_to_sequences(dataset['News'])

max_length = max(len(seq) for seq in sequences)
# print(sequences)
padded_sequences = pad_sequences(sequences, maxlen=max_length)
```

In [6]:
```
# Converting categorical value to numerical
label_encoder = LabelEncoder()
X = padded_sequences
y = label_encoder.fit_transform(dataset['category'])

print("Shape of X:", X.shape)
print("Shape of y:", y.shape)
```

```
Shape of X: (209527, 141)
Shape of y: (209527,)
```

In [7]:
```
print(max(y), min(y))
```

```
41 0
```

Train & Test Split

In [8]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state= 42)

word_index = tokenizer.word_index
total_words = len(word_index)
print("Length of word index:", total_words)
```

```
Length of word index: 223792
```

In [9]:
```
# Converting label to one-hot encoding (for categorical classification)
from keras.utils import to_categorical
num_classes = len(label_encoder.classes_)
# print(num_classes)
```

```
y_train = to_categorical(y_train, num_classes=num_classes)
y_test = to_categorical(y_test, num_classes=num_classes)

print("Shape of X_train:", X_train.shape)
print("Shape of y_train:", y_train.shape)
print("Shape of X_test:", X_test.shape)
print("Shape of y_test:", y_test.shape)
```

```
Shape of X_train: (167621, 141)
Shape of y_train: (167621, 42)
Shape of X_test: (41906, 141)
Shape of y_test: (41906, 42)
```

Building RNN Model

In [10]:
```
# import necessary libraries
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, SimpleRNN, Dense, Dropout, Bidirectional
```

In [11]:
```
# Define the RNN model using list format
model = Sequential([
    Embedding(input_dim=total_words, output_dim=70, input_length=max_length),  # Word embedding layer
    Bidirectional(SimpleRNN(64, activation='tanh', dropout=0.1, recurrent_dropout=0.2, return_sequences=True)),  # F
    Bidirectional(SimpleRNN(64, activation='tanh', dropout=0.1, recurrent_dropout=0.3, return_sequences=True)),  # Se
    SimpleRNN(32, activation='tanh'),  # Final RNN layer
    Dropout(0.2),  # Dropout for regularization
    Dense(num_classes, activation='softmax')  # Output layer
])

model.build(input_shape=(None, max_length))

# Print model summary
model.summary()
```

```
C:\Users\alens\anaconda3\Lib\site-packages\keras\src\layers\core\embedding.py:90: UserWarning: Argument `input_length
` is deprecated. Just remove it.
  warnings.warn(
```

Model: "sequential"

Layer (type)	Output Shape	Param #
embedding (Embedding)	(None, 141, 70)	15,665,440
bidirectional (Bidirectional)	(None, 141, 128)	17,280
bidirectional_1 (Bidirectional)	(None, 141, 128)	24,704
simple_rnn_2 (SimpleRNN)	(None, 32)	5,152
dropout (Dropout)	(None, 32)	0
dense (Dense)	(None, 42)	1,386

Total params: 15,713,962 (59.94 MB)

Trainable params: 15,713,962 (59.94 MB)

Non-trainable params: 0 (0.00 B)

In [13]:
```
# Compile the model
model.compile(optimizer='rmsprop',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
# fit model to the data
history = model.fit(X_train, y_train,
    batch_size=128,
    epochs=15,
    validation_split=0.2
)

# evalute the model
test_loss, test_acc = model.evaluate(X_test, y_test, verbose=0)
print("Test loss and accuracy:", test_loss, test_acc)
```

```
Epoch 1/15
1048/1048 ━━━━━━━━━━━━━━━━━━━━ 728s 651ms/step - accuracy: 0.3240 - loss: 2.6849 - val_accuracy: 0.3614 - val_loss:
2.4654
Epoch 2/15
1048/1048 ━━━━━━━━━━━━━━━━━━━━ 735s 701ms/step - accuracy: 0.3749 - loss: 2.3946 - val_accuracy: 0.3836 - val_loss:
2.3833
Epoch 3/15
1048/1048 ━━━━━━━━━━━━━━━━━━━━ 784s 741ms/step - accuracy: 0.4139 - loss: 2.2540 - val_accuracy: 0.4123 - val_loss:
2.2988
Epoch 4/15
1048/1048 ━━━━━━━━━━━━━━━━━━━━ 802s 741ms/step - accuracy: 0.4495 - loss: 2.1269 - val_accuracy: 0.4399 - val_loss:
2.1941
Epoch 5/15
1048/1048 ━━━━━━━━━━━━━━━━━━━━ 780s 745ms/step - accuracy: 0.4725 - loss: 2.0459 - val_accuracy: 0.4565 - val_loss:
2.1490
Epoch 6/15
1048/1048 ━━━━━━━━━━━━━━━━━━━━ 788s 731ms/step - accuracy: 0.4893 - loss: 1.9775 - val_accuracy: 0.4617 - val_loss:
2.1285
Epoch 7/15
1048/1048 ━━━━━━━━━━━━━━━━━━━━ 786s 750ms/step - accuracy: 0.5024 - loss: 1.9191 - val_accuracy: 0.4662 - val_loss:
2.1165
Epoch 8/15
1048/1048 ━━━━━━━━━━━━━━━━━━━━ 792s 756ms/step - accuracy: 0.5184 - loss: 1.8586 - val_accuracy: 0.4692 - val_loss:
2.1009
Epoch 9/15
1048/1048 ━━━━━━━━━━━━━━━━━━━━ 800s 763ms/step - accuracy: 0.5280 - loss: 1.8144 - val_accuracy: 0.4704 - val_loss:
2.0959
Epoch 10/15
1048/1048 ━━━━━━━━━━━━━━━━━━━━ 948s 904ms/step - accuracy: 0.5397 - loss: 1.7631 - val_accuracy: 0.4765 - val_loss:
2.0926
Epoch 11/15
1048/1048 ━━━━━━━━━━━━━━━━━━━━ 1554s 1s/step - accuracy: 0.5489 - loss: 1.7220 - val_accuracy: 0.4787 - val_loss: 2.0
798
Epoch 12/15
1048/1048 ━━━━━━━━━━━━━━━━━━━━ 1582s 2s/step - accuracy: 0.5556 - loss: 1.6998 - val_accuracy: 0.4599 - val_loss: 2.1
973
Epoch 13/15
1048/1048 ━━━━━━━━━━━━━━━━━━━━ 626s 597ms/step - accuracy: 0.5586 - loss: 1.6841 - val_accuracy: 0.4685 - val_loss:
2.1073
Epoch 14/15
1048/1048 ━━━━━━━━━━━━━━━━━━━━ 594s 567ms/step - accuracy: 0.5712 - loss: 1.6439 - val_accuracy: 0.4780 - val_loss:
2.0842
```

```
Epoch 15/15
1048/1048 ━━━━━━━━━━━━━━━━━━━━ 599s 571ms/step - accuracy: 0.5725 - loss: 1.6314 - val_accuracy: 0.4704 - val_loss:
2.1146
Test loss and accuracy: 2.0895652770996094 0.47628024220466614
```