PHASE 3

SENTIMENT ANALYSIS FOR MARKETING

MEMBERS

- SOUNDHAR BALAJI.B
- RADHIKA.M
- ROHANSHAJ.K.R
- VIGNESH.V
- NELSON JOSEPH.M

INTRODUCTION:

- In the realm of natural language processing and sentiment analysis, the journey to extract meaningful insights from text data commences with the critical steps of loading and preprocessing the dataset. These initial stages serve as the foundation upon which the entire sentiment analysis solution is built.

- Loading the dataset is akin to unearthing a treasure trove of textual information. It is the act of retrieving the raw data that will be the lifeblood of your analysis. The source could be diverse - from social media posts, customer reviews, or any corpus of text that holds the sentiment of interest.

- However, raw text data is rarely ready for analysis in its pristine form. Preprocessing is the transformative process that makes the data amenable to machine learning and natural language processing algorithms. It involves a series of steps like text cleaning, tokenization, removing stop words, stemming, and lemmatization. This ensures that the data is standardized, uniform, and free from noise, thus enhancing the quality of insights derived.

- In this part of the project, we will delve into the crucial tasks of loading the dataset, understanding its structure, and undertaking the necessary preprocessing steps. This groundwork sets the stage for subsequent phases, including feature engineering, model development, and sentiment analysis. With a well-prepared dataset, the journey towards understanding and harnessing sentiment within the textual data can begin.

TASK:

Phase 3: Development Part 1

In this part you will begin building your project by loading and preprocessing the dataset.
Start building the sentiment analysis solution by
loading dataset and preprocessing the data.

DATASET :https://www.kaggle.com/datasets/crowdflower/twitter-airline-sentiment

NOTEBOOK LINK :https://drive.google.com/drive/folders/1G6Gqw6_E7Cs8dfOrto3jl4_eXOZGiDt3

PROGRAM

```
In [20]: pip install nitk

Requirement already satisfied: nitk in
c:\users\sound\appdata\local\programs\python\python39\lib\site-packages (3.8.1) Requirement
already satisfied: joblib in c:\users\sound\appdata\local\programs\python\python39\lib\site-
packages (from nitk) (1.3.2)
Requirement already satisfied: tqdm in
c:\users\sound\appdata\local\programs\python\python39\lib\site-packages (from nitk) (4. 66.1)
Requirement already satisfied: regex>=2021.8.3 in
c:\users\sound\appdata\local\programs\python\python39\lib\site-packages (fro m nitk)
(2023.10.3)
Requirement already satisfied: click in
c:\users\sound\appdata\local\programs\python\python39\lib\site-packages (from nitk) (8.1.7)
Requirement already satisfied: colorama in
c:\users\sound\appdata\local\programs\python\python39\lib\site-packages (from click ->nitk)
(0.4.6)
```

```
In [21]: import numpy as np import pandas as pd
'import re 'import emoji from nitk.stem import PorterStemmer
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
-Frnm clelparn mnripl cplprtinn imnnrt train tact cnlit
```

```
In [22]: data    pd.read csv(r"C:\Users\sound\Downloads\Tweets.csv")
```

```
In [42]: data.head()
```

OUT 42:

	tweet_id	airline_sentiment	airline_sentiment_confidence	negativereason	negativereason_confidence	airline	airline_sentiment_gold	name	negativereason_gold	retweet_count	text	tweet_coord	tweet_created	tweet_location	user_timezone
0	570306133677760513	neutral	1.0000	NaN	NaN	Virgin America	NaN	cairdin	NaN	0	@VirginAmerica What @dhepburn said.	NaN	2015-02-24 11:35:52 -0800	NaN	Eastern Time (US & Canada)
1	570301130888122368	positive	0.3486	NaN	0.0000	Virgin America	NaN	jnardino	NaN	0	@VirginAmerica plus you've added commercials t...	NaN	2015-02-24 11:15:59 -0800	NaN	Pacific Time (US & Canada)
2	570301083672813571	neutral	0.6837	NaN	NaN	Virgin America	NaN	yvonnalynn	NaN	0	@VirginAmerica I didn't today... Must mean I n...	NaN	2015-02-24 11:15:48 -0800	Lets Play	Central Time (US & Canada)
3	570301031407624196	negative	1.0000	Bad Flight	0.7033	Virgin America	NaN	jnardino	NaN	0	@VirginAmerica it's really aggressive to blast...	NaN	2015-02-24 11:15:36 -0800	NaN	Pacific Time (US & Canada)
4	570300817074462722	negative	1.0000	Can't Tell	1.0000	Virgin America	NaN	jnardino	NaN	0	@VirginAmerica and it's a really big bad thing...	NaN	2015-02-24 11:14:45 -0800	NaN	Pacific Time (US & Canada)
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
14635	569587686496825344	positive	0.3487	NaN	0.0000	American	NaN	KristenReenders	NaN	0	@AmericanAir thank you we got on a different f...	NaN	2015-02-22 12:01:01 -0800	NaN	NaN
14636	569587371693355008	negative	1.0000	Customer Service Issue	1.0000	American	NaN	itsropes	NaN	0	@AmericanAir leaving over 20 minutes Late Flig...	NaN	2015-02-22 11:59:46 -0800	Texas	NaN
14637	569587242672398336	neutral	1.0000	NaN	NaN	American	NaN	sanyabun	NaN	0	@AmericanAir Please bring American Airlines to...	NaN	2015-02-22 11:59:15 -0800	Nigeria,lagos	NaN
14638	569587188687634433	negative	1.0000	Customer Service Issue	0.6659	American	NaN	SraJackson	NaN	0	@AmericanAir you have my money, you change my ...	NaN	2015-02-22 11:59:02 -0800	New Jersey	Eastern Time (US & Canada)
14639	569587140490866689	neutral	0.6771	NaN	0.0000	American	NaN	daviddtwu	NaN	0	@AmericanAir we have 8 ppl so we need 2 know h...	NaN	2015-02-22 11:58:51 -0800	dallas, TX	NaN

14640 rows × 15 columns

```
In [23]: data
```

Preprocessing

```
confidence_threshold = 0.6

data = data.drop(data.query("airline_sentiment_confidence < @confidence_threshold").index,
axis=0).reset_index(drop=True)
```

```
tweets_df = pd.concat([data['text'], data['airline_sentiment']], axis=1)
tweets_df
```

	text	airline_sentiment
0	@VirginAmerica What @dhepburn said.	neutral
1	@VirginAmerica I didn't today... Must mean I n...	neutral
2	@VirginAmerica it's really aggressive to blast...	negative
3	@VirginAmerica and it's a really big bad thing...	negative
4	@VirginAmerica seriously would pay $30 a fligh...	negative
...	...	...
14397	@AmericanAir right on cue with the delays	negative
14398	@AmericanAir leaving over 20 minutes Late Flig...	negative
14399	@AmericanAir Please bring American Airlines to...	neutral
14400	@AmericanAir you have my money, you change my ...	negative
14401	@AmericanAir we have 8 ppl so we need 2 know h...	neutral

14402 rows × 2 columns

```
tweets_df['airline_sentiment'].value_counts()
```

```
airline_sentiment
negative     9113
neutral      2997
positive     2292
Name: count, dtype: int64
```

```
tweets_df.isna().sum().sum()
```

```python
sentiment_ordering = ['negative', 'neutral', 'positive']

tweets_df['airline_sentiment'] = tweets_df['airline_sentiment'].apply(lambda x: sentiment_ordering.index(x))
```

```python
tweets_df
```

	text	airline_sentiment
0	@VirginAmerica What @dhepburn said.	1
1	@VirginAmerica I didn't today... Must mean I n...	1
2	@VirginAmerica it's really aggressive to blast...	0
3	@VirginAmerica and it's a really big bad thing...	0
4	@VirginAmerica seriously would pay $30 a fligh...	0
...	...	...
14397	@AmericanAir right on cue with the delays 👌	0
14398	@AmericanAir leaving over 20 minutes Late Flig...	0
14399	@AmericanAir Please bring American Airlines to...	1
14400	@AmericanAir you have my money, you change my ...	0
14401	@AmericanAir we have 8 ppl so we need 2 know h...	1

14402 rows × 2 columns

```python
emoji.demojize('@AmericanAir right on cue with the delays▯')
```

@AmericanAir right on cue with the delays:OK_hand:'

```python
ps = PorterStemmer()

def process_tweet(tweet):
    new_tweet = tweet.lower()
    new_tweet = re.sub(r'@\w+', '', new_tweet) # Remove @s
    new_tweet = re.sub(r'#', '', new_tweet) # Remove hashtags
    new_tweet = re.sub(r':', ' ', emoji.demojize(new_tweet)) # Turn emojis into words
    new_tweet = re.sub(r'http\S+', '',new_tweet) # Remove URLs
    new_tweet = re.sub(r'\$\S+', 'dollar', new_tweet) # Change dollar amounts to dollar
    new_tweet = re.sub(r'[^a-z0-9\s]', '', new_tweet) # Remove punctuation
    new_tweet = re.sub(r'[0-9]+', 'number', new_tweet) # Change number values to number
    new_tweet = new_tweet.split(" ")
    new_tweet = list(map(lambda x: ps.stem(x), new_tweet)) # Stemming the words
    new_tweet = list(map(lambda x: x.strip(), new_tweet)) # Stripping whitespace from the words
    if '' in new_tweet:
        new_tweet.remove('')
    return new_tweet
```

```python
tweets = tweets_df['text'].apply(process_tweet)

labels = np.array(tweets_df['airline_sentiment'])
```

```
tweets
```

```
0                                              [what, , said]
1          [i, didnt, today, must, mean, i, need, to, tak...
2          [it, realli, aggress, to, blast, obnoxi, enter...
3           [and, it, a, realli, big, bad, thing, about, it]
4          [serious, would, pay, dollar, a, flight, for, ...
                                 ...
14397              [right, on, cue, with, the, delay, hand, ]
14398      [leav, over, number, minut, late, flight, no, ...
14399      [pleas, bring, american, airlin, to, blackberr...
14400      [you, have, my, money, you, chang, my, flight,...
14401      [we, have, number, ppl, so, we, need, number, ...
Name: text, Length: 14402, dtype: object
```

```
# Get size of vocabulary
vocabulary = set()

for tweet in tweets:
    for word in tweet:
        if word not in vocabulary:
            vocabulary.add(word)

vocab_length = len(vocabulary)

# Get max length of a sequence
max_seq_length = 0

for tweet in tweets:
    if len(tweet) > max_seq_length:
        max_seq_length = len(tweet)

# Print results
print("Vocab length:", vocab_length)
print("Max sequence length:", max_seq_length)
```

```
Vocab length: 11250
Max sequence length: 90
```

```
tokenizer = Tokenizer(num_words=vocab_length)
tokenizer.fit_on_texts(tweets)

sequences = tokenizer.texts_to_sequences(tweets)

word_index = tokenizer.word_index

model_inputs = pad_sequences(sequences, maxlen=max_seq_length, padding='post')
```

```
model_inputs
```

```
array([[  49,    2,  218, ...,    0,    0,    0],
       [   5,  191,  102, ...,    0,    0,    0],
       [  15,  138, 2841, ...,    0,    0,    0],
       ...,
       [  76,  507,  435, ...,    0,    0,    0],
       [   8,   19,   12, ...,    0,    0,    0],
       [  37,   19,    4, ...,    0,    0,    0]])
```

```
model_inputs.shape
```

```
(14402, 90)
```

```
X_train, X_test, y_train, y_test = train_test_split(model_inputs, labels, train_size=0.7, random_state=22)
```

Training

```
embedding_dim = 32
inputs = tf.keras.Input(shape=(max_seq_length,))


embedding = tf.keras.layers.Embedding(
    input_dim=vocab_length,
    output_dim=embedding_dim,
    input_length=max_seq_length
)(inputs)

# Model A (just a Flatten layer)
flatten = tf.keras.layers.Flatten()(embedding)


# Model B (GRU with a Flatten layer)
gru = tf.keras.layers.GRU(units=embedding_dim)(embedding)
gru_flatten = tf.keras.layers.Flatten()(gru)


# Both A and B are fed into the output
concat = tf.keras.layers.concatenate([flatten, gru_flatten])


outputs = tf.keras.layers.Dense(3, activation='softmax')(concat)
model = tf.keras.Model(inputs, outputs)


tf.keras.utils.plot_model(model)
```

```
model.compile(
    optimizer='adam',
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy'])
batch_size = 32
epochs = 100
history = model.fit(
    X_train,
    y_train,
    validation_split=0.2,
    batch_size=batch_size,
    epochs=epochs,
    callbacks=[
        tf.keras.callbacks.EarlyStopping(
            monitor='val_loss',
            patience=3,
          restore_best_weights=True,
          verbose=1
        ),
        tf.keras.callbacks.ReduceLROnPlateau() ])
```

```
Epoch 1/100 252/252 [==============================] - 10s 30ms/step - loss: 0.7891 - accuracy: 0.6664 - val_loss: 0.6665 - val_accuracy: 0.7248 - lr: 0.0010 Epoch 2/100 252/252 [==============================] - 7s 27ms/step - loss: 0.5201 - accuracy: 0.7991 - val_loss: 0.5361 - val_accuracy: 0.7873 - lr: 0.0010 Epoch 3/100 252/252 [==============================] - 7s 27ms/step - loss: 0.3668 - accuracy: 0.8710 - val_loss: 0.5028 - val_accuracy: 0.8002 - lr: 0.0010 Epoch 4/100 252/252 [==============================] - 7s 28ms/step - loss: 0.2703 - accuracy: 0.9117 - val_loss: 0.5048 - val_accuracy: 0.8057 - lr: 0.0010 Epoch 5/100 252/252 [==============================] - 7s 28ms/step - loss: 0.2025 - accuracy: 0.9375 - val_loss: 0.5170 - val_accuracy: 0.8032 - lr: 0.0010 Epoch 6/100 252/252 [==============================] - ETA: 0s - loss: 0.1512 - accuracy: 0.9604Restoring model weights from the end of the best epoch: 3. 252/252 [==============================] - 7s 28ms/step - loss: 0.1512 - accuracy: 0.9604 - val_loss: 0.5317 - val_accuracy: 0.8012 - lr: 0.0010 Epoch 6: early stopping
```

Results

```
model.evaluate(X_test, y_test)
```

136/136 [==============================] - 1s 9ms/step - loss: 0.4885 - accuracy: 0.8051
[0.48851093649864197, 0.8051376938819885]

CONCLUSION

In this initial phase of our sentiment analysis project, we've made significant progress by successfully loading and preprocessing the dataset. This foundational step is crucial for the success of our entire project, as the quality and structure of our data will directly impact the accuracy and reliability of our sentiment analysis models. By loading the dataset, we've bridged the gap between raw data and actionable insights, making it accessible for further analysis. Our preprocessing efforts, which included tasks such as text cleaning, tokenization, and handling missing values, have improved the data's quality, making it ready for more advanced natural language processing techniques.

Loading the dataset was more than just a technical task; it marked the beginning of our journey towards understanding and predicting sentiment in text. The dataset, comprised of text data from various sources, holds the potential to reveal valuable insights about people's opinions, emotions, and attitudes. By ensuring it is correctly structured and prepared, we are one step closer to extracting meaningful information. Our diligent preprocessing work ensures that the data is consistent and free from common issues that could otherwise lead to biased or inaccurate results in our sentiment analysis.

As we move forward in this sentiment analysis project, we can build upon this solid foundation. The loaded and preprocessed dataset serves as the cornerstone for our data-driven insights, allowing us to explore different natural language processing techniques, sentiment analysis algorithms, and model development. With this groundwork in place, we are now better equipped to delve into the fascinating world of sentiment analysis and ultimately provide valuable insights that can inform decision-making, marketing strategies, and much more. Our commitment to data quality and preprocessing sets the stage for the success of our sentiment analysis solution.