Installing Dependencies

In [1]:

```
!pip install tensorflow==2.15.0
!pip install scikit-learn
!pip install wordcloud
!pip install nltk
```

```
dcloud) (3.0.9)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\asus\anaconda3\lib\site-packages (from matplotlib->wo
rdcloud) (1.4.4)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\asus\anaconda3\lib\site-packages (from matplotlib-
>wordcloud) (2.8.2)
Requirement already satisfied: cycler>=0.10 in c:\users\asus\anaconda3\lib\site-packages (from matplotlib->wordclo
ud) (0.11.0)
Requirement already satisfied: packaging>=20.0 in c:\users\asus\anaconda3\lib\site-packages (from matplotlib->word
cloud) (23.2)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\asus\anaconda3\lib\site-packages (from matplotlib->wo
rdcloud) (4.25.0)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\asus\anaconda3\lib\site-packages (from matplotlib->wor
dcloud) (1.0.5)
Requirement already satisfied: six>=1.5 in c:\users\asus\anaconda3\lib\site-packages (from python-dateutil>=2.7->m
atplotlib->wordcloud) (1.16.0)
Installing collected packages: wordcloud
Successfully installed wordcloud-1.9.3

Requirement already satisfied: nltk in c:\users\asus\anaconda3\lib\site-packages (3.7)
Requirement already satisfied: regex>=2021.8.3 in c:\users\asus\anaconda3\lib\site-packages (from nltk) (2022.7.9)
```

Loading Dependencies

In [2]:

```
# dl packages
from keras.models import Sequential
from keras.layers import Embedding, LSTM, Dense, Dropout
from keras.callbacks import EarlyStopping
from keras.preprocessing.text import one_hot
from keras.preprocessing.sequence import pad_sequences
from keras.utils import to_categorical

# ml packages
from sklearn.preprocessing import LabelEncoder
import numpy as np
import pandas as pd
import pickle
import nltk
import re
from nltk.stem import PorterStemmer

import seaborn as sns
import matplotlib.pyplot as plt

from wordcloud import WordCloud
```

```
WARNING:tensorflow:From C:\Users\Asus\anaconda3\lib\site-packages\keras\src\losses.py:2976: The name tf.losses.spars
e_softmax_cross_entropy is deprecated. Please use tf.compat.v1.losses.sparse_softmax_cross_entropy instead.
```

Loading Dataset

In [4]:

```
train_data = pd.read_csv("train.txt", header=None, sep=";", names=["Comment", "Emotion"], encoding="utf-8")
# get all words length in comment
train_data['length'] = [len(x) for x in train_data['Comment']]
```

In [5]:
```
train_data
```

Out[5]:

	Comment	Emotion	length
0	i didnt feel humiliated	sadness	23
1	i can go from feeling so hopeless to so damned...	sadness	108
2	im grabbing a minute to post i feel greedy wrong	anger	48
3	i am ever feeling nostalgic about the fireplac...	love	92
4	i am feeling grouchy	anger	20
...	...	...	...
15995	i just had a very brief time in the beanbag an...	sadness	101
15996	i am now turning and i feel pathetic that i am...	sadness	102
15997	i feel strong and good overall	joy	30
15998	i feel like this was such a rude comment and i...	anger	59
15999	i know a lot but i feel so stupid because i ca...	sadness	62

16000 rows × 3 columns

In [6]:
```
train_data.shape
```

Out[6]:
```
(16000, 3)
```

In [7]:
```
train_data.isnull().sum()
```

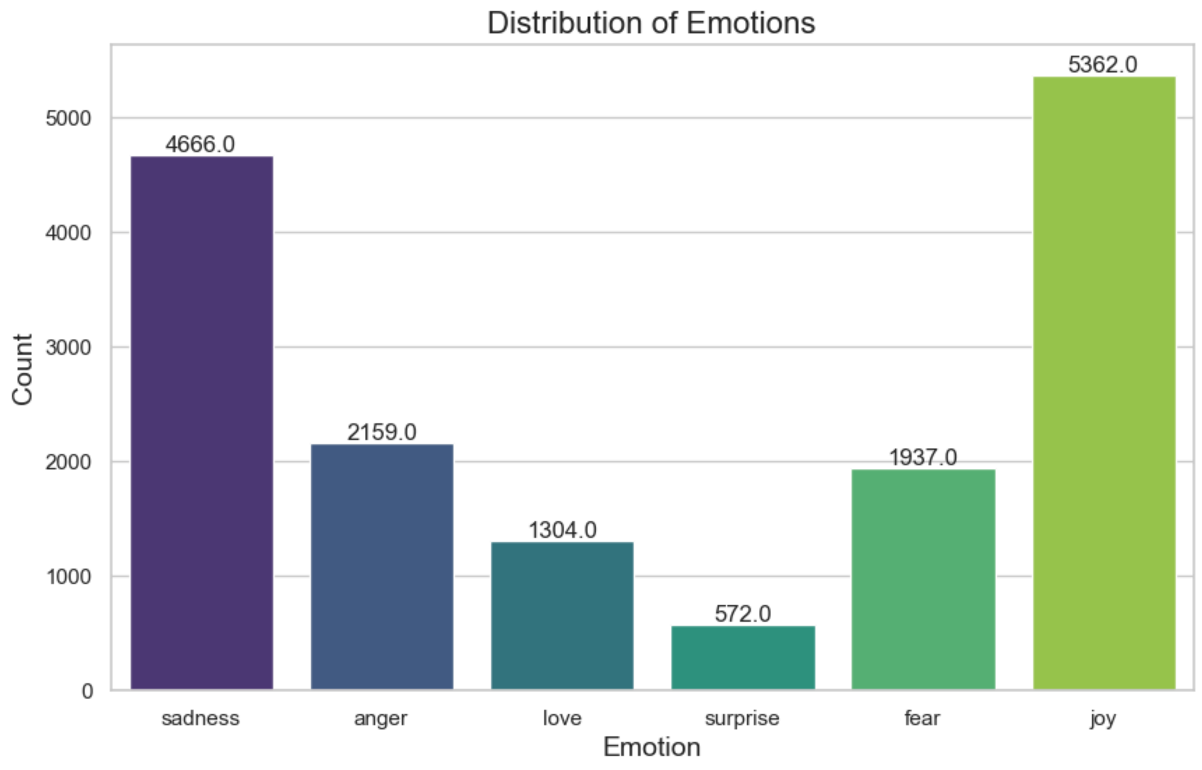
Out[7]:
```
Comment     0
Emotion     0
length      0
dtype: int64
```

In [8]:
```
train_data.duplicated().sum()
```

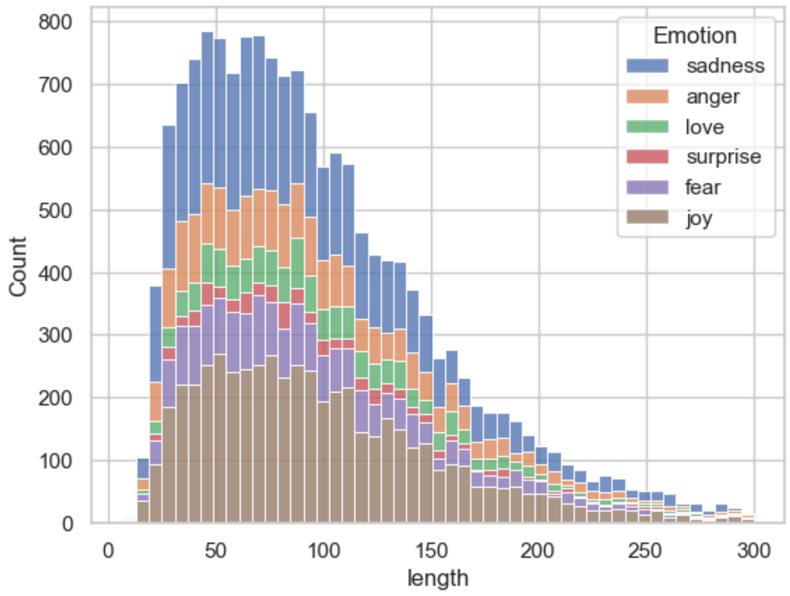
Out[8]:
```
1
```

E D A

In [9]:
```python
sns.set_theme(style="whitegrid")

# Create the count plot
plt.figure(figsize=(10, 6))
ax = sns.countplot(x='Emotion', data=train_data, palette='viridis')

# Add title and labels
plt.title('Distribution of Emotions', fontsize=16)
plt.xlabel('Emotion', fontsize=14)
plt.ylabel('Count', fontsize=14)

# Annotate bars with counts
for p in ax.patches:
    ax.annotate(f'{p.get_height()}', (p.get_x() + p.get_width() / 2, p.get_height()),
                ha='center', va='center', xytext=(0, 5), textcoords='offset points', fontsize=12)

# Show the plot
plt.show()
```

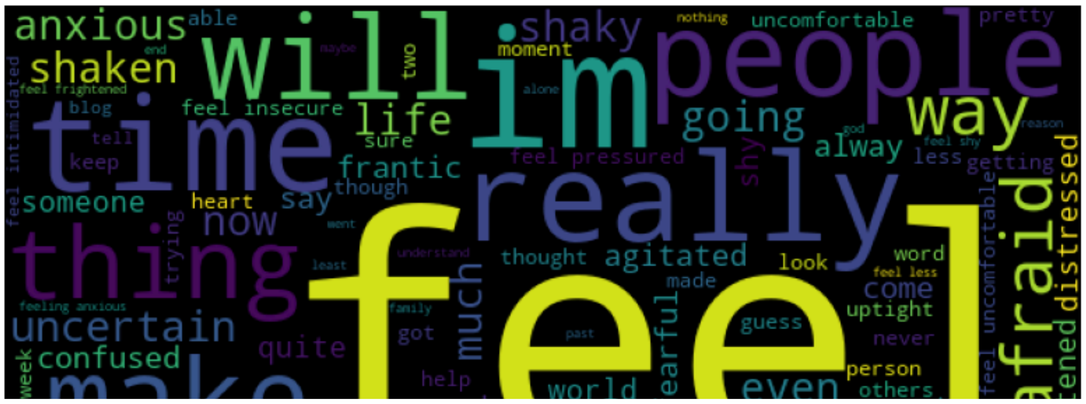


In [10]:
```python
# data distribution
df2 = train_data.copy()
df2['length'] = [len(x) for x in df2['Comment']]

# Convert the 'length' column to a numpy array
length_values = df2['length'].values

# Use sns.histplot instead of sns.kdeplot for simplicity
sns.histplot(data=df2, x='length', hue='Emotion', multiple='stack')

plt.show()
```



In [11]:
```python
#Words cloud for each emotions
def words_cloud(wordcloud, df):
    plt.figure(figsize=(10, 10))
    plt.title(df+' Word Cloud', size = 16)
    plt.imshow(wordcloud)
    # No axis details
    plt.axis("off");
emotions_list = train_data['Emotion'].unique()
for emotion in emotions_list:
    text = ' '.join([sentence for sentence in train_data.loc[train_data['Emotion'] == emotion,'Comment']])
    wordcloud = WordCloud(width = 600, height = 600).generate(text)
    words_cloud(wordcloud, emotion)
```

fear Word Cloud



Data Pre-processing

ENCODE EMOTIONS

In [12]:
```
lb = LabelEncoder()
train_data['Emotion'] = lb.fit_transform(train_data['Emotion'])
```

In [13]:
```
train_data
```

Out[13]:

	Comment	Emotion	length
0	i didnt feel humiliated	4	23
1	i can go from feeling so hopeless to so damned...	4	108
2	im grabbing a minute to post i feel greedy wrong	0	48
3	i am ever feeling nostalgic about the fireplac...	3	92
4	i am feeling grouchy	0	20
...	...	...	...
15995	i just had a very brief time in the beanbag an...	4	101
15996	i am now turning and i feel pathetic that i am...	4	102
15997	i feel strong and good overall	2	30
15998	i feel like this was such a rude comment and i...	0	59
15999	i know a lot but i feel so stupid because i ca...	4	62

16000 rows × 3 columns

Machine Learning Techniques

In [14]:
```
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report
```

In [15]:
```
df = train_data.copy()
```

In [16]:
```
df
```

Out[16]:

	Comment	Emotion	length
0	i didnt feel humiliated	4	23
1	i can go from feeling so hopeless to so damned...	4	108
2	im grabbing a minute to post i feel greedy wrong	0	48
3	i am ever feeling nostalgic about the fireplac...	3	92
4	i am feeling grouchy	0	20
...	...	...	...
15995	i just had a very brief time in the beanbag an...	4	101
15996	i am now turning and i feel pathetic that i am...	4	102
15997	i feel strong and good overall	2	30
15998	i feel like this was such a rude comment and i...	0	59
15999	i know a lot but i feel so stupid because i ca...	4	62

16000 rows × 3 columns

In [17]:
```
# Data cleaning and preprocessing
# Download NLTK stopwords
nltk.download('stopwords')
stopwords = set(nltk.corpus.stopwords.words('english'))
def clean_text(text):
    stemmer = PorterStemmer()
    text = re.sub("[^a-zA-Z]", " ", text)
    text = text.lower()
    text = text.split()
    text = [stemmer.stem(word) for word in text if word not in stopwords]
    return " ".join(text)

df['cleaned_comment'] = df['Comment'].apply(clean_text)
```

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

In [18]:
```
X_train, X_test, y_train, y_test = train_test_split(df['cleaned_comment'],df['Emotion'],test_size=0.2,random_state=42
```

In [19]:
```
# Vectorization using TF-IDF
tfidf_vectorizer = TfidfVectorizer()
X_train_tfidf = tfidf_vectorizer.fit_transform(X_train)
X_test_tfidf = tfidf_vectorizer.transform(X_test)
```

In [20]:
```python
# Multi-class classification using different algorithms
classifiers = {
    "Multinomial Naive Bayes": MultinomialNB(),
    "Logistic Regression": LogisticRegression(),
    "Random Forest": RandomForestClassifier(),
    "Support Vector Machine": SVC(),
}

for name, clf in classifiers.items():
    print(f"\n===== {name} =====")
    clf.fit(X_train_tfidf, y_train)
    y_pred_tfidf = clf.predict(X_test_tfidf)
    accuracy_tfidf = accuracy_score(y_test, y_pred_tfidf)
    print(f"\nAccuracy using TF-IDF: {accuracy_tfidf}")
    print("Classification Report:")
    print(classification_report(y_test, y_pred_tfidf))
```

```
===== Multinomial Naive Bayes =====

Accuracy using TF-IDF: 0.655
Classification Report:
              precision    recall  f1-score   support

           0       0.93      0.31      0.46       427
           1       0.91      0.24      0.38       397
           2       0.58      0.98      0.73      1021
           3       1.00      0.03      0.06       296
           4       0.70      0.91      0.79       946
           5       1.00      0.01      0.02       113

    accuracy                           0.66      3200
   macro avg       0.85      0.41      0.41      3200
weighted avg       0.76      0.66      0.58      3200


===== Logistic Regression =====

C:\Users\Asus\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:458: ConvergenceWarning: lbfgs failed to
converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://scikit-learn.org/stable/modules/preprocessin
g.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression (https://scikit-learn.org/stable/m
odules/linear_model.html#logistic-regression)
  n_iter_i = _check_optimize_result(
Accuracy using TF-IDF: 0.829375
Classification Report:
              precision    recall  f1-score   support

           0       0.88      0.79      0.83       427
           1       0.84      0.73      0.78       397
           2       0.78      0.94      0.85      1021
           3       0.80      0.49      0.61       296
           4       0.88      0.92      0.90       946
           5       0.77      0.45      0.57       113

    accuracy                           0.83      3200
   macro avg       0.82      0.72      0.76      3200
weighted avg       0.83      0.83      0.82      3200


===== Random Forest =====

Accuracy using TF-IDF: 0.8471875
Classification Report:
              precision    recall  f1-score   support

           0       0.80      0.85      0.82       427
           1       0.84      0.83      0.83       397
           2       0.84      0.89      0.87      1021
           3       0.79      0.64      0.71       296
           4       0.91      0.88      0.89       946
           5       0.75      0.71      0.73       113

    accuracy                           0.85      3200
   macro avg       0.82      0.80      0.81      3200
weighted avg       0.85      0.85      0.85      3200


===== Support Vector Machine =====

Accuracy using TF-IDF: 0.8190625
Classification Report:
              precision    recall  f1-score   support

           0       0.86      0.79      0.83       427
           1       0.84      0.71      0.77       397
           2       0.76      0.93      0.84      1021
           3       0.81      0.45      0.58       296
           4       0.88      0.91      0.89       946
           5       0.79      0.47      0.59       113

    accuracy                           0.82      3200
   macro avg       0.82      0.71      0.75      3200
weighted avg       0.82      0.82      0.81      3200
```

In [21]:
```python
#selecting model
lg = LogisticRegression()
lg.fit(X_train_tfidf, y_train)
lg_y_pred = lg.predict(X_test_tfidf)
```

```
C:\Users\Asus\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:458: ConvergenceWarning: lbfgs failed to
converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://scikit-learn.org/stable/modules/preprocessin
g.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression (https://scikit-learn.org/stable/m
odules/linear_model.html#logistic-regression)
  n_iter_i = _check_optimize_result(
```

In [22]:
```python
def predict_emotion(input_text):
    cleaned_text = clean_text(input_text)
    input_vectorized = tfidf_vectorizer.transform([cleaned_text])

    # Predict emotion
    predicted_label = lg.predict(input_vectorized)[0]
    predicted_emotion = lb.inverse_transform([predicted_label])[0]
    label =  np.max(lg.predict(input_vectorized))

    return predicted_emotion,label

# Example usage
sentences = [
            "i didnt feel humiliated",
            "i feel strong and good overall",
            "im grabbing a minute to post i feel greedy wrong",
            "He was speechles when he found out he was accepted to this new job",
            "This is outrageous, how can you talk like that?",
            "I feel like im all alone in this world",
            "He is really sweet and caring",
            "You made me very crazy",
            "i am ever feeling nostalgic about the fireplace i will know that it is still on the property",
            "i am feeling grouchy",
            "He hates you"
            ]
for sentence in sentences:
    print(sentence)
    pred_emotion, label = predict_emotion(sentence)
    print("Prediction :",pred_emotion)
    print("Label :",label)
    print("================================================================")
```

```
i didnt feel humiliated
Prediction : sadness
Label : 4
================================================================
i feel strong and good overall
Prediction : joy
Label : 2
================================================================
im grabbing a minute to post i feel greedy wrong
Prediction : anger
Label : 0
================================================================
He was speechles when he found out he was accepted to this new job
Prediction : joy
Label : 2
================================================================
This is outrageous, how can you talk like that?
Prediction : anger
Label : 0
================================================================
I feel like im all alone in this world
Prediction : sadness
Label : 4
================================================================
He is really sweet and caring
Prediction : love
Label : 3
================================================================
You made me very crazy
Prediction : sadness
Label : 4
================================================================
i am ever feeling nostalgic about the fireplace i will know that it is still on the property
Prediction : love
Label : 3
================================================================
i am feeling grouchy
Prediction : anger
Label : 0
================================================================
He hates you
Prediction : anger
Label : 0
================================================================
```

In [23]:
```
# save files
import pickle
pickle.dump(lg,open("logistic_regresion.pkl",'wb'))
pickle.dump(lb,open("label_encoder.pkl",'wb'))
pickle.dump(tfidf_vectorizer,open("tfidf_vectorizer.pkl",'wb'))
```

In [29]:
```
import sklearn
print(sklearn.__version__)
```

```
1.2.1
```

Applying Deep learning Using LSTM

Text Cleaning, Ecoding, and Padding

In [24]:
```
# Text cleaning function
def text_cleaning(df, column, vocab_size, max_len):
    stemmer = PorterStemmer()
    corpus = []

    for text in df[column]:
        text = re.sub("[^a-zA-Z]", " ", text)
        text = text.lower()
        text = text.split()
        text = [stemmer.stem(word) for word in text if word not in stopwords]
        text = " ".join(text)
        corpus.append(text)

    one_hot_word = [one_hot(input_text=word, n=vocab_size) for word in corpus]
    pad = pad_sequences(sequences=one_hot_word, maxlen=max_len, padding='pre')
    return pad

# Text cleaning and encoding
x_train = text_cleaning(train_data, "Comment", vocab_size=11000, max_len=300)
y_train = to_categorical(train_data["Emotion"])
```

Model Building and Training

In [25]:
```python
# Build and compile the model
model = Sequential()
model.add(Embedding(input_dim=11000, output_dim=150, input_length=300))
model.add(Dropout(0.2))
model.add(LSTM(128))
model.add(Dropout(0.2))
model.add(Dense(64, activation='sigmoid'))
model.add(Dropout(0.2))
model.add(Dense(6, activation='softmax'))
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

# Train the model
callback = EarlyStopping(monitor="val_loss", patience=2, restore_best_weights=True)
model.fit(x_train, y_train, epochs=10, batch_size=64, verbose=1, callbacks=[callback])
```

```
WARNING:tensorflow:From C:\Users\Asus\anaconda3\lib\site-packages\keras\src\backend.py:873: The name tf.get_default_
graph is deprecated. Please use tf.compat.v1.get_default_graph instead.

WARNING:tensorflow:From C:\Users\Asus\anaconda3\lib\site-packages\keras\src\optimizers\__init__.py:309: The name tf.
train.Optimizer is deprecated. Please use tf.compat.v1.train.Optimizer instead.

Epoch 1/10
WARNING:tensorflow:From C:\Users\Asus\anaconda3\lib\site-packages\keras\src\utils\tf_utils.py:492: The name tf.ragge
d.RaggedTensorValue is deprecated. Please use tf.compat.v1.ragged.RaggedTensorValue instead.

WARNING:tensorflow:From C:\Users\Asus\anaconda3\lib\site-packages\keras\src\engine\base_layer_utils.py:384: The name
tf.executing_eagerly_outside_functions is deprecated. Please use tf.compat.v1.executing_eagerly_outside_functions in
stead.

250/250 [==============================] - ETA: 0s - loss: 1.4396 - accuracy: 0.4332WARNING:tensorflow:Early stoppin
g conditioned on metric `val_loss` which is not available. Available metrics are: loss,accuracy
250/250 [==============================] - 84s 316ms/step - loss: 1.4396 - accuracy: 0.4332
Epoch 2/10
250/250 [==============================] - ETA: 0s - loss: 0.6337 - accuracy: 0.8018WARNING:tensorflow:Early stoppin
g conditioned on metric `val_loss` which is not available. Available metrics are: loss,accuracy
250/250 [==============================] - 88s 351ms/step - loss: 0.6337 - accuracy: 0.8018
Epoch 3/10
250/250 [==============================] - ETA: 0s - loss: 0.3008 - accuracy: 0.9068WARNING:tensorflow:Early stoppin
g conditioned on metric `val_loss` which is not available. Available metrics are: loss,accuracy
250/250 [==============================] - 90s 359ms/step - loss: 0.3008 - accuracy: 0.9068
Epoch 4/10
250/250 [==============================] - ETA: 0s - loss: 0.2027 - accuracy: 0.9324WARNING:tensorflow:Early stoppin
g conditioned on metric `val_loss` which is not available. Available metrics are: loss,accuracy
250/250 [==============================] - 91s 364ms/step - loss: 0.2027 - accuracy: 0.9324
Epoch 5/10
250/250 [==============================] - ETA: 0s - loss: 0.1545 - accuracy: 0.9479WARNING:tensorflow:Early stoppin
g conditioned on metric `val_loss` which is not available. Available metrics are: loss,accuracy
250/250 [==============================] - 94s 378ms/step - loss: 0.1545 - accuracy: 0.9479
Epoch 6/10
250/250 [==============================] - ETA: 0s - loss: 0.1228 - accuracy: 0.9592WARNING:tensorflow:Early stoppin
g conditioned on metric `val_loss` which is not available. Available metrics are: loss,accuracy
250/250 [==============================] - 94s 376ms/step - loss: 0.1228 - accuracy: 0.9592
Epoch 7/10
250/250 [==============================] - ETA: 0s - loss: 0.1010 - accuracy: 0.9650WARNING:tensorflow:Early stoppin
g conditioned on metric `val_loss` which is not available. Available metrics are: loss,accuracy
250/250 [==============================] - 96s 383ms/step - loss: 0.1010 - accuracy: 0.9650
Epoch 8/10
250/250 [==============================] - ETA: 0s - loss: 0.0923 - accuracy: 0.9693WARNING:tensorflow:Early stoppin
g conditioned on metric `val_loss` which is not available. Available metrics are: loss,accuracy
250/250 [==============================] - 97s 390ms/step - loss: 0.0923 - accuracy: 0.9693
Epoch 9/10
250/250 [==============================] - ETA: 0s - loss: 0.0780 - accuracy: 0.9735WARNING:tensorflow:Early stoppin
g conditioned on metric `val_loss` which is not available. Available metrics are: loss,accuracy
250/250 [==============================] - 95s 381ms/step - loss: 0.0780 - accuracy: 0.9735
Epoch 10/10
250/250 [==============================] - ETA: 0s - loss: 0.0666 - accuracy: 0.9766WARNING:tensorflow:Early stoppin
g conditioned on metric `val_loss` which is not available. Available metrics are: loss,accuracy
250/250 [==============================] - 93s 372ms/step - loss: 0.0666 - accuracy: 0.9766
```

Out[25]:
```
<keras.src.callbacks.History at 0x24c3629df00>
```

In [26]:
```
# Text cleaning function
def sentence_cleaning(sentence):
    stemmer = PorterStemmer()
    corpus = []
    text = re.sub("[^a-zA-Z]", " ", sentence)
    text = text.lower()
    text = text.split()
    text = [stemmer.stem(word) for word in text if word not in stopwords]
    text = " ".join(text)
    corpus.append(text)
    one_hot_word = [one_hot(input_text=word, n=11000) for word in corpus]
    pad = pad_sequences(sequences=one_hot_word, maxlen=300, padding='pre')
    return pad

# load model and predict
sentences = [
            "i feel strong and good overall",
            "im grabbing a minute to post i feel greedy wrong",
            "He was speechles when he found out he was accepted to this new job",
            "This is outrageous, how can you talk like that?",
            "I feel like im all alone in this world",
            "He is really sweet and caring",
            "You made me very crazy",
            "i am ever feeling nostalgic about the fireplace i will know that it is still on the property",
            "i am feeling grouchy",
            "He hates you"
            ]
for sentence in sentences:
    print(sentence)
    sentence = sentence_cleaning(sentence)
    result = lb.inverse_transform(np.argmax(model.predict(sentence), axis=-1))[0]
    proba =  np.max(model.predict(sentence))
    print(f"{result} : {proba}\n\n")
```

```
i feel strong and good overall
1/1 [==============================] - 1s 1s/step
1/1 [==============================] - 0s 47ms/step
joy : 0.9993389248847961


im grabbing a minute to post i feel greedy wrong
1/1 [==============================] - 0s 47ms/step
1/1 [==============================] - 0s 31ms/step
anger : 0.9985910058021545


He was speechles when he found out he was accepted to this new job
1/1 [==============================] - 0s 47ms/step
1/1 [==============================] - 0s 31ms/step
joy : 0.4203015863895416


This is outrageous, how can you talk like that?
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 43ms/step
anger : 0.9345224499702454


I feel like im all alone in this world
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 47ms/step
sadness : 0.9973914623260498


He is really sweet and caring
1/1 [==============================] - 0s 47ms/step
1/1 [==============================] - 0s 31ms/step
love : 0.8503820896148682


You made me very crazy
1/1 [==============================] - 0s 48ms/step
1/1 [==============================] - 0s 31ms/step
joy : 0.5420773029327393


i am ever feeling nostalgic about the fireplace i will know that it is still on the property
1/1 [==============================] - 0s 47ms/step
1/1 [==============================] - 0s 32ms/step
love : 0.9941604733467102


i am feeling grouchy
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 47ms/step
anger : 0.9965582489967346


He hates you
1/1 [==============================] - 0s 47ms/step
1/1 [==============================] - 0s 31ms/step
anger : 0.9705713987350464
```

In [27]:
```python
model.save('model1.h5')

# Save the LabelEncoder
with open('lb1.pkl', 'wb') as f:
    pickle.dump(lb, f)

# Save vocabulary size and max length
vocab_info = {'vocab_size': 11000, 'max_len': 300}
with open('vocab_info.pkl', 'wb') as f:
    pickle.dump(vocab_info, f)
```

```
C:\Users\Asus\anaconda3\lib\site-packages\keras\src\engine\training.py:3103: UserWarning: You are saving your model
as an HDF5 file via `model.save()`. This file format is considered legacy. We recommend using instead the native Ker
as format, e.g. `model.save('my_model.keras')`.
  saving_api.save_model(
```

In [28]:
```
# use this version
import tensorflow
import keras
print(keras.__version__)
print(tensorflow.__version__)
```

```
2.15.0
2.15.0
```

In [ ]: