Text_Emotions_Classification_using_Python

In [1]:
```python
import pandas as pd
import numpy as np
import keras
import tensorflow
from keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from keras.models import Sequential
from keras.layers import Embedding, Flatten, Dense
```

In [2]:
```python
data = pd.read_csv("train.txt", sep = ';')
data.columns = ["Text", "Emotions"]
print(data.head())
```

```
                                                Text Emotions
0  i can go from feeling so hopeless to so damned...  sadness
1   im grabbing a minute to post i feel greedy wrong    anger
2  i am ever feeling nostalgic about the fireplac...     love
3                               i am feeling grouchy    anger
4  ive been feeling a little burdened lately wasn...  sadness
```

As this is a problem of natural language processing, I'll start by tokenizing the data:

In [3]:
```python
texts = data["Text"].tolist()
labels = data["Emotions"].tolist()

#Tokenize the text data
tokenizer = Tokenizer()
tokenizer.fit_on_texts(texts)
```

Now we need to pad the sequences to the same length to feed them into a neural network. Here's how we can pad the sequences of the texts to have the same length:

In [4]:
```python
sequences = tokenizer.texts_to_sequences(texts)
max_length = max([len(seq) for seq in sequences])
padded_sequences = pad_sequences(sequences, maxlen = max_length)
```

Now I'll use the label encoder method to convert the classes from strings to a numerical representation:

In [5]:
```python
# Encode the string labels to integers
label_encoder = LabelEncoder()
labels = label_encoder.fit_transform(labels)
```

We are now going to One-hot encode the labels. One hot encoding refers to the transformation of categorical labels into a binary representation where each label is represented as a vector of all zeros except a single 1. This is necessary because machine learning algorithms work with numerical data. So here is how we can One-hot encode the labels:

In [6]:
```python
# One-hot encode the labels
```

```
one_hot_labels = keras.utils.to_categorical(labels)
```

Text Emotions Classification Model

In [7]:
```
# Split the data into training and testing sets
xtrain, xtest, ytrain, ytest = train_test_split(padded_sequences,
                                                one_hot_labels,
                                                test_size=0.2)
```

Now let's define a neural network architecture for our classification problem and use it to train a model to classify emotions:

In [8]:
```
# Define the model
model = Sequential()
model.addd(Embedding(input_dim = len(tokenizer.word_index) + 1,
                    output_dim = 128, input_length = max_length))

model.add(Flatten())
model.add(Dense(units = 128, activation = "relu"))
model.add(Dense(units = len(one_hot_labels[0]), activation = "softmax"))

model.compile(optimizer = "adam", loss = "categorical_crossentropy", metrics = ["ac
model.fit(xtrain, ytrain, epochs = 10, batch_size = 32, validation_data = (xtest, y
```

```
Epoch 1/10
400/400 [==============================] - 31s 76ms/step - loss: 1.3850 - accurac
y: 0.4727 - val_loss: 0.9033 - val_accuracy: 0.7063
Epoch 2/10
400/400 [==============================] - 27s 68ms/step - loss: 0.3747 - accurac
y: 0.8855 - val_loss: 0.5067 - val_accuracy: 0.8334
Epoch 3/10
400/400 [==============================] - 23s 58ms/step - loss: 0.0578 - accurac
y: 0.9847 - val_loss: 0.5420 - val_accuracy: 0.8309
Epoch 4/10
400/400 [==============================] - 23s 59ms/step - loss: 0.0260 - accurac
y: 0.9955 - val_loss: 0.5458 - val_accuracy: 0.8397
Epoch 5/10
400/400 [==============================] - 29s 73ms/step - loss: 0.0177 - accurac
y: 0.9970 - val_loss: 0.5659 - val_accuracy: 0.8341
Epoch 6/10
400/400 [==============================] - 42s 106ms/step - loss: 0.0153 - accurac
y: 0.9972 - val_loss: 0.6522 - val_accuracy: 0.8303
Epoch 7/10
400/400 [==============================] - 39s 98ms/step - loss: 0.0145 - accurac
y: 0.9973 - val_loss: 0.6316 - val_accuracy: 0.8309
Epoch 8/10
400/400 [==============================] - 30s 76ms/step - loss: 0.0116 - accurac
y: 0.9976 - val_loss: 0.6530 - val_accuracy: 0.8325
Epoch 9/10
400/400 [==============================] - 30s 75ms/step - loss: 0.0111 - accurac
y: 0.9977 - val_loss: 0.7316 - val_accuracy: 0.8225
Epoch 10/10
400/400 [==============================] - 33s 82ms/step - loss: 0.0100 - accurac
y: 0.9977 - val_loss: 0.7419 - val_accuracy: 0.8184
```

Out[8]:
```
<keras.src.callbacks.History at 0x16d4b83fd30>
```

Now let's take a sentence as an input text and see how the model performs:

In [9]:
```
input_text = "She didn't come today because she lost her dog yestertay!"
```

```
# Preprocess the input text
input_sequence = tokenizer.texts_to_sequences([input_text])
padded_input_sequence = pad_sequences(input_sequence, maxlen=max_length)
prediction = model.predict(padded_input_sequence)
predicted_label = label_encoder.inverse_transform([np.argmax(prediction[0])])
print(predicted_label)
```

```
1/1 [==============================] - 0s 156ms/step
['sadness']
```

So this is how you can use Machine Learning for the task of text emotion classification using the Python programming language.

THANK YOU!

GitHub Link: https://github.com/anujtiwari21?tab=repositories