Open in Colab
(https://colab.research.google.com/github/Sukhwinder9813/EmojiPredictionfromTweet/blob/master/EmojiPredictor

In [ ]:
```
import pandas as pd
import numpy as np
```

In [ ]:
```
import nltk
nltk.download('popular')
```

In [ ]:
```
!pip install emoji
import emoji
```

In [ ]:
```
from keras.preprocessing.text import Tokenizer # https://keras-cn.readthedocs.io/en/latest/preprocessing/text/
```

In [ ]:
```
from nltk.corpus import stopwords # https://www.nltk.org/book/ch02.html 页面搜索stopwords
```

In [ ]:
```
data=pd.read_csv('Train.csv')
```

In [ ]:
```
x_train=data['TEXT'].values
```

In [ ]:
```
y_train=data['Label'].values
```

In [ ]:
```
smoothtweets=[]
stopper=set(stopwords.words('english')) # set() 生成无序不重复集 # stopwords.words() 列出无实意词（停止词）
for tweets in x_train:
    words=tweets.split(" ") #以空格为分隔符 分割所有词
    str = ""
    for word in words:
        if word[0] != "@" and word not in stopper: # 第一个字符不是@且不是停止词
            if word[0] == "#":
                word = word[1:] # 去除词前#
        str += word + " " # 去除停止词、词前#和第一个字符是@的词 重新形成一句话
    smoothtweets.append(str) # 将经过处理的句子放入list
```

In [ ]:
```
smoothtweets[2] # 数据集中的第二句话
```

In [ ]:
```
tokenizer = Tokenizer(filters='!"#$%&()*+,-./:;<=>?@[\\]^_`{|}~\t\n', split=" ", lower=True) # filters：需要去除的符号 lower：转换成小写
tokenizer.fit_on_texts(smoothtweets) # 更新词库
newsmoothtweets=[] # 初始化序列
newsmoothtweets=tokenizer.texts_to_sequences(smoothtweets) # 将数据集序列化，就是把句中每一个单词编号
```

In [ ]:
```
print(newsmoothtweets[3])
len(newsmoothtweets)
```

In [ ]:
```
from keras.preprocessing import sequence
X_train=sequence.pad_sequences(newsmoothtweets,maxlen=12,padding='post') # 填充序列，就是使得每句话对应的序列长度都是'maxlen'
```

In [30]:
```
from keras.layers import *
from keras.models import Sequential
```

In [ ]:
```
from keras.utils import to_categorical
```

In [ ]:
```
Y_train=to_categorical(y_train) # one-hot编码
```

In [ ]:
```
Y_train.shape
```

In [ ]:
```
X_train.shape
```

In [ ]:
```

```

In [ ]:
```
wordembedd={}
f=open('glove.6B.50d.txt',"r",encoding='utf-8')
for line in f:
  words=line.split() # 默认为所有的空字符，包括空格、换行(\n)、制表符(\t)等
  word=words[0] # 词
  coefs=np.asarray(words[1:],dtype='float') # 编码
  wordembedd[word]=coefs
f.close()
```

In [ ]:
```
def populate_weight_matrix(vocab, raw_embedding):
    # Create weight matrix from pre-trained embeddings
    vocab_size = len(vocab) + 1 # 因为fit_on_texts时自动去除了1个最不常见的词
    weight_matrix = np.zeros((vocab_size, 50))
    for word, i in vocab.items(): # .items返回可遍历的元组数组
        if word in raw_embedding: # word是'glove.6B.50d.txt'中的; raw_embedding是'popular'中的
            weight_matrix[i] = raw_embedding[word]
    return weight_matrix # 以'glove.6B.50d.txt'中的规则将'popular'训练集编码
```

In [ ]:
```
vocab=tokenizer.word_index # 将单词（字符串）映射为它们的排名或者索引。仅在调用fit_on_texts之后设置。结果是dict{'word',integer}形式
```

In [28]:
```
len(vocab)
import math
```

In [ ]:
```
weight_matrix=populate_weight_matrix(vocab,wordembedd)# 以'glove.6B.50d.txt'中的规则将'popular'训练集编码
```

In [34]:
```
weight_matrix.shape
```

Out[34]:
```
(89623, 50)
```

In [33]:
```
max_length = math.ceil(sum([len(s.split(" ")) for s in smoothtweets])/len(smoothtweets)) #
vocab_size=len(vocab)+1
model=Sequential()
model.add(Embedding(vocab_size,50,weights=[weight_matrix],input_length=max_length+2,trainable=True,)) # vocab_size:
输入长度;50:输出长度;我们设置trainable=true使得这个编码层可再训练
model.add(LSTM(128,dropout=0.2,return_sequences=True))
model.add(LSTM(128,dropout=0.2))
model.add(Dense(20,activation='softmax'))
model.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['accuracy'])
```

In [ ]:
```
model.summary()
```

In [ ]:
```
model.fit(X_train,Y_train,epochs=5,batch_size=128,shuffle=True,validation_split=0.15)
```

In [ ]:
```
model.evaluate(X_train,Y_train)
```

In [ ]:
```
model.predict_classes(X_train[1:13])
```

In [ ]:
```
smoothtweets[9]
```

In [ ]:
```
df=pd.read_csv('Mapping.csv')
```

In [ ]:
```
df.head(20)
```

In [ ]:
```
data['Label'][9]
```

In [ ]:
```
df3=pd.read_csv('Test.csv')
```

In [ ]:
```
df3.head()
```

In [ ]:
```
xtest=df3['TEXT']
```

In [ ]:
```
smoothTESTtweets=[]
stopper=set(stopwords.words('english'))
for tweets in xtest:
  words=tweets.split(" ")
  str = ""
  for word in words:
      if word[0] != "@" and word not in stopper:
          if word[0] == "#":
              word = word[1:]
          str += word + " "
  smoothTESTtweets.append(str)
```

In [ ]:
```
newsmoothtweets=tokenizer.texts_to_sequences(smoothTESTtweets)
```

In [ ]:
```
newsmoothtweets[2]
```

In [ ]:
```
x_test=sequence.pad_sequences(newsmoothtweets,maxlen=12,padding='post')
```

In [ ]:
```
x_test[3]
```

In [ ]:
```
data=model.predict_classes(x_test)
```

In [ ]:
```
for i in range(20,35,1):
  print(df['emoticons'][data[i]])
  print(smoothTESTtweets[i])
```

In [ ]:
```
smoothTESTtweets[2]
```

In [ ]:
```
xtest = ["I love you"]

smoothTESTtweets=[]
stopper=set(stopwords.words('english'))
for tweets in xtest:
  words=tweets.split(" ")
  str = ""
  for word in words:
      if word[0] != "@" and word not in stopper:
          if word[0] == "#":
              word = word[1:]
          str += word + " "
  smoothTESTtweets.append(str)

newsmoothtweets=tokenizer.texts_to_sequences(smoothTESTtweets)

x_test=sequence.pad_sequences(newsmoothtweets,maxlen=12,padding='post')
data=model.predict_classes(x_test)
```

In [ ]:
```
print(df['emoticons'][data[0]])
print(smoothTESTtweets[0])
```

In [ ]:
```

```