By: Yousef Khalil

```python
import kagglehub
# Download latest version
path = kagglehub.dataset_download("jp797498e/twitter-entity-sentiment-
analysis")
print("Path to dataset files:", path)
```

```
Downloading from
https://www.kaggle.com/api/v1/datasets/download/jp797498e/twitter-
entity-sentiment-analysis?dataset_version_number=2...

100%|██████████| 1.99M/1.99M [00:00<00:00, 99.4MB/s]

Extracting files...
Path to dataset files:
/root/.cache/kagglehub/datasets/jp797498e/twitter-entity-sentiment-
analysis/versions/2
```

```python
import pandas as pd
import numpy as np
import os
import re
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")

dataset=pd.read_csv(os.path.join(path,os.listdir(path)[1]))
dataset.head()
```

```
{"summary":"{\n  \"name\": \"dataset\",\n  \"rows\": 74681,\n
\"fields\": [\n    {\n      \"column\": \"2401\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
3740,\n        \"min\": 1,\n        \"max\": 13200,\n
\"num_unique_values\": 12447,\n        \"samples\": [\n
1616,\n          2660,\n          2335\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Borderlands\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 32,\n        \"samples\": [\n
\"Cyberpunk2077\",\n          \"Microsoft\",\n
\"TomClancysRainbowSix\"\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"Positive\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 4,\n        \"samples\":
[\n          \"Neutral\",\n          \"Irrelevant\",\n
\"Positive\"\n        ],\n        \"semantic_type\": \"\",\n
```

\"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"im getting on borderlands and i will murder you all ,\",\n   \"properties\": {\n   \"dtype\": \"string\",\n   \"num_unique_values\": 69490,\n   \"samples\": [\n   \"so how does my stained glass open facebook account girl already have 200 likes!!!! and i sure am so!!??? oh thankful!??!?!\",\n   \"How not to get bored about every damn thing in life.\",\n   \"The Best Perfect Way to Protect All the Planet Samsung Galaxy Note10 + By buff. ly / The 2zkjIhU..\"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   }\n]\n}","type":"dataframe","variable_name":"dataset"}

```
dataset.rename(columns={dataset.columns[3]:"reviews",dataset.columns[0
]:"index",dataset.columns[2]:"label",dataset.columns[1]:"company"},inp
lace=True)

dataset.sample(3)
```

{"summary":"{\n \"name\": \"dataset\",\n \"rows\": 3,\n \"fields\": [\n   {\n   \"column\": \"index\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 3720,\n   \"min\": 1386,\n   \"max\": 8809,\n   \"num_unique_values\": 3,\n   \"samples\": [\n   8809,\n   5544,\n   1386\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"company\",\n   \"properties\": {\n   \"dtype\": \"string\",\n   \"num_unique_values\": 3,\n   \"samples\": [\n   \"Nvidia\",\n   \"Hearthstone\",\n   \"Battlefield\"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"label\",\n   \"properties\": {\n   \"dtype\": \"category\",\n   \"num_unique_values\": 1,\n   \"samples\": [\n   \"Neutral\"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"reviews\",\n   \"properties\": {\n   \"dtype\": \"string\",\n   \"num_unique_values\": 3,\n   \"samples\": [\n   \"The latest The GST Daily! paper.li / GKConsultants2... Thanks to @ LiquorMarts\"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   }\n]\n}","type":"dataframe"}

```
dataset.isna().sum()

index        0
company      0
label        0
reviews    686
dtype: int64

dataset=dataset.dropna()
dataset.duplicated().sum()
```

```
np.int64(2340)
```

```
dataset=dataset.drop_duplicates()
dataset.company.value_counts()
```

```
company
TomClancysRainbowSix                  2328
Verizon                               2319
MaddenNFL                             2315
CallOfDuty                            2314
Microsoft                             2304
WorldOfCraft                          2300
NBA2K                                 2299
LeagueOfLegends                       2296
TomClancysGhostRecon                  2291
Facebook                              2289
ApexLegends                           2278
johnson&johnson                       2257
Battlefield                           2255
Amazon                                2249
CallOfDutyBlackopsColdWar             2242
FIFA                                  2238
Dota2                                 2225
Overwatch                             2220
Hearthstone                           2219
HomeDepot                             2216
GrandTheftAuto(GTA)                   2208
Borderlands                           2205
Xbox(Xseries)                         2201
Google                                2199
Nvidia                                2198
CS-GO                                 2195
PlayStation5(PS5)                     2183
Fortnite                              2176
Cyberpunk2077                         2175
AssassinsCreed                        2156
RedDeadRedemption(RDR)                2155
PlayerUnknownsBattlegrounds(PUBG)     2150
Name: count, dtype: int64
```

```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 71655 entries, 0 to 74680
Data columns (total 4 columns):
 #   Column   Non-Null Count  Dtype 
---  ------   --------------  ----- 
 0   index    71655 non-null  int64 
 1   company  71655 non-null  object
 2   label    71655 non-null  object
 3   reviews  71655 non-null  object
dtypes: int64(1), object(3)
memory usage: 2.7+ MB
```

```
dataset.label.value_counts()
```

```
label
Negative      21698
Positive      19712
Neutral       17708
Irrelevant    12537
Name: count, dtype: int64
```

Data Prerocessing

```
def extract_emoji(text_string):                         # For EXtract
All Emoji From Text
    emoji = []
    for char in text_string:
        if len(char.encode()) >=3 :
            emoji.append(char)
    return " ".join(emoji)

def extract_profile_name(text):                        # For EXtract
Profile Names From Text
    result=[]
    rev=text.split('@')
    try :
      for i in range(len(rev)-1):
        t=rev[i+1]
        split_name=t.split()
        result.append(f"@{split_name[0]}")
      return result
    except:
      pass

dataset['emoji']=dataset['reviews'].apply(extract_emoji)
# create a table for emojis
dataset['reviews']=dataset['reviews'].apply(lambda x:x.lower())
# convert all reviews to lower case
dataset['natural_review']=dataset['reviews']
# create a column for natural reviews to compare later
dataset['reviews']=dataset['reviews'].apply(lambda
x:re.sub(extract_emoji(x),'',x))                # remove emojis from
reviews
dataset['profile_name']=dataset['reviews'].apply(extract_profile_name)
# create a table for profile names
dataset['reviews']=dataset['reviews'].apply(lambda x:re.sub('[^a-zA-
Z]',' ',x))                    # remove any special characters and
```

```
numbers from reviews
dataset['label']=dataset['label'].apply(lambda x:x.lower())

dataset.sample(10)
```

```
{"summary":"{\n  \"name\": \"dataset\",\n  \"rows\": 10,\n
\"fields\": [\n    {\n      \"column\": \"index\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
3547,\n        \"min\": 2064,\n        \"max\": 13150,\n
\"num_unique_values\": 10,\n        \"samples\": [\n          10289,\n
8486,\n          13150\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"company\",\n      \"properties\": {\n        \"dtype\": \"string\",\
n        \"num_unique_values\": 9,\n        \"samples\": [\n
\"PlayerUnknownsBattlegrounds(PUBG)\",\n          \"NBA2K\",\n
\"RedDeadRedemption(RDR)\"\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"label\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 3,\n        \"samples\":
[\n          \"neutral\",\n          \"negative\",\n
\"irrelevant\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"reviews\",\n      \"properties\": {\n        \"dtype\": \"string\",\
n        \"num_unique_values\": 10,\n        \"samples\": [\n
\"rmb the tl lost it over tae playing pubg  but only to be
disappointed when they learned he was playing over the cocoa server
\",\n          \"yo  nba k fuck cat fake ass garbage ass game\",\n
\"     keemstar ya know  the social media sti       is going after the
video game blogger  thequartering       why   who fuckin  cares
just wanted to point out where keem is at in life    painfully lame
\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"emoji\",\n      \"properties\": {\n        \"dtype\": \"category\",\
n        \"num_unique_values\": 4,\n        \"samples\": [\n
\"\",\n          \"\\ud83e\\udd23 \\ud83e\\udd21 \\u2019 \\ud83e\\
udd26 \\u200d \\ud83e\\udd23\",\n          \"\\u2013\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"natural_review\",\n
\"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 10,\n        \"samples\": [\n          \"rmb
the tl lost it over tae playing pubg, but only to be disappointed when
they learned he was playing over the cocoa server.\",\n          \"yo
@nba2k fuck cat fake ass garbage ass game\",\n          \"\\ud83e\\
udd23.  @keemstar ya know, the social media sti?. . \\ud83e\\udd21 is
going after the video game blogger @thequartering . . . why?! who
fuckin\\u2019 cares. \\ud83e\\udd26\\u200d.  just wanted to point out
where keem is at in life. \\ud83e\\udd23 painfully lame.. .  \"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"profile_name\",\n
\"properties\": {\n        \"dtype\": \"object\",\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    }\n  ]\n}","type":"dataframe"}
```

```
data_for_model=dataset[['reviews','label']]
```

Build Model

```
import torch.nn as nn
import torch
import torch.optim as optim
from torch.utils.data import DataLoader, Dataset
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences

import nltk
nltk.download('stopwords')
nltk.download('punkt')
from nltk.corpus import stopwords
stopwords=stopwords.words('english')
data_for_model['reviews']=data_for_model['reviews'].apply(lambda x:'
'.join([word for word in x.split() if word not in stopwords]))
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
```

```
PERCENTILE = 97
print( f"{PERCENTILE}th percentile length Reviews:
{np.percentile([len(x) for x in data_for_model['reviews'].values],
PERCENTILE)}" )
```

```
97th percentile length Reviews: 193.0
```

```
max_len=200
reviews=data_for_model['reviews']
label=data_for_model['label']
label_encoder=LabelEncoder()
label=label_encoder.fit_transform(label)

tokenizer=Tokenizer(num_words=10000)
tokenizer.fit_on_texts(reviews)
tokenized_reviews=tokenizer.texts_to_sequences(reviews)
seq_reviews=pad_sequences(tokenized_reviews,maxlen=max_len)
x_train,x_test,y_train,y_test=train_test_split(seq_reviews,label,test_
size=0.25,random_state=42,shuffle=True)

print(reviews.values[0])
print(seq_reviews[0])
```

```
coming borders kill
[    0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0  286 6345  311]

class DataClass(Dataset):
  def __init__(self,review,label):
    self.review=torch.tensor(review)
    self.label=torch.tensor(label)

  def __len__(self):
    return len(self.review)

  def __getitem__(self,idx):
    return self.review[idx],self.label[idx]

def batch(batch):
   texts, labels = zip(*batch)
x_train,x_test,y_train,y_test=train_test_split(seq_reviews,label,test_
size=0.25,random_state=42,shuffle=True)

x_train=torch.tensor(x_train)
y_train=torch.tensor(y_train)
x_test=torch.tensor(x_test)
y_test=torch.tensor(y_test)

train_loader=DataLoader(DataClass(x_train,y_train),batch_size=32)
valied_loader=DataLoader(DataClass(x_test,y_test),batch_size=32)

class LSTMClassifier(nn.Module):
    def __init__(self, vocab_size, embed_dim, hidden_dim, output_dim):
        super().__init__()
        self.embedding = nn.Embedding(vocab_size, embed_dim)
        self.lstm = nn.LSTM(embed_dim, hidden_dim, batch_first=True)
        self.fc = nn.Linear(hidden_dim, output_dim)
        self.dropout = nn.Dropout(0.3)

    def forward(self, x):
        embedded = self.embedding(x)
        _, (hidden, _) = self.lstm(embedded)
        out = self.dropout(hidden[-1])
        return self.fc(out)

vocab=len(tokenizer.index_word)
number_of_class=len(label_encoder.classes_)
model = LSTMClassifier(vocab_size=vocab, embed_dim=100, hidden_dim=64,
output_dim=number_of_class)
criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.parameters(), lr=0.001)

# Train Model

for epoch in range(10):
    model.train()
    total_loss = 0
    for texts, labels in train_loader:
        optimizer.zero_grad()
        outputs = model(texts)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()
        total_loss += loss.item()
    print(f'Epoch {epoch+1}, Loss: {total_loss:.4f}')
```

```
Epoch 1, Loss: 1855.7926
Epoch 2, Loss: 1276.4628
Epoch 3, Loss: 907.4885
Epoch 4, Loss: 670.0378
Epoch 5, Loss: 527.2335
Epoch 6, Loss: 438.6749
Epoch 7, Loss: 374.1906
Epoch 8, Loss: 333.5186
Epoch 9, Loss: 306.5507
Epoch 10, Loss: 275.7264
```

```
# Evaluate the Model
model.eval()
correct, total = 0, 0
with torch.no_grad():
    for texts, labels in valied_loader:
        outputs = model(texts)
        preds = torch.argmax(outputs, dim=1)
        correct += (preds == labels).sum().item()
        total += labels.size(0)
print(f'Validation Accuracy: {correct/total:.2f}')
```

```
Validation Accuracy: 0.85
```