Movie Reviews Sentiment Analysis Web App

1: Download the data

In [1]:
```
%mkdir ../data
!wget -O ../data/aclImdb_v1.tar.gz http://ai.stanford.edu/~amaas/data/sentimen
!tar -zxf ../data/aclImdb_v1.tar.gz -C ../data
```

```
mkdir: cannot create directory '../data': File exists
--2024-04-19 22:27:42--  http://ai.stanford.edu/~amaas/data/sentiment/aclImdb_
v1.tar.gz
Resolving ai.stanford.edu (ai.stanford.edu)... 171.64.68.10
Connecting to ai.stanford.edu (ai.stanford.edu)|171.64.68.10|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 84125825 (80M) [application/x-gzip]
Saving to: '../data/aclImdb_v1.tar.gz'

100%[======================================>] 84,125,825  22.2MB/s   in 5.5s

2024-04-19 22:27:47 (14.6 MB/s) - '../data/aclImdb_v1.tar.gz' saved [84125825/
84125825]
```

2: Data Preparation

In [2]:
```
import os
import glob

def read_imdb_data(data_dir='../data/aclImdb'):
    data = {}
    labels = {}

    for data_type in ['train', 'test']:
        data[data_type] = {}
        labels[data_type] = {}

        for sentiment in ['pos', 'neg']:
            data[data_type][sentiment] = []
            labels[data_type][sentiment] = []

            path = os.path.join(data_dir, data_type, sentiment, '*.txt')
            files = glob.glob(path)

            for f in files:
                with open(f) as review:
                    data[data_type][sentiment].append(review.read())
                    # represent a positive review by '1' and a negative review
                    labels[data_type][sentiment].append(1 if sentiment == 'pos

            assert len(data[data_type][sentiment]) == len(labels[data_type][se
                    "{}/{} data size does not match labels size".format(data_ty

    return data, labels
```

In [3]:
```
data, labels = read_imdb_data()
print("IMDB reviews: train = {} pos / {} neg, test = {} pos / {} neg".format(
            len(data['train']['pos']), len(data['train']['neg']),
            len(data['test']['pos']), len(data['test']['neg'])))
```

```
IMDB reviews: train = 12500 pos / 12500 neg, test = 12500 pos / 12500 neg
```

In [4]:
```
from sklearn.utils import shuffle

def prepare_imdb_data(data, labels):
    """Prepare training and test sets from IMDb movie reviews."""

    #Combine positive and negative reviews and labels
    data_train = data['train']['pos'] + data['train']['neg']
    data_test = data['test']['pos'] + data['test']['neg']
    labels_train = labels['train']['pos'] + labels['train']['neg']
    labels_test = labels['test']['pos'] + labels['test']['neg']

    #Shuffle reviews and corresponding labels within training and test sets
    data_train, labels_train = shuffle(data_train, labels_train)
    data_test, labels_test = shuffle(data_test, labels_test)

    # Return a unified training data, test data, training labels, test labels
    return data_train, data_test, labels_train, labels_test
```

In [5]:
```
train_X, test_X, train_y, test_y = prepare_imdb_data(data, labels)
print("IMDb reviews (combined): train = {}, test = {}".format(len(train_X), le
```

```
IMDb reviews (combined): train = 25000, test = 25000
```

In [6]:
```
print(train_X[10])
print(train_y[10])
```

```
This movie was for a while in my collection, but it wasn't before a friend of
mine reminded me about it  until I decided that I should watch it. I did not
know much about Close to Leo  just that it was supposed to be excellent comin
g out of age movie and it deals with a very serious topic  Aids. <br /><br />
Although the person who has aids  is Leo  the scenario wraps around the way
in which Marcel (the youngest brother of Leo) coupes with the sickness of his
relative. At first everyone is trying to hide the truth from Marcel  he is be
lieved to be too young to understand the sickness of his brother  the fact th
at Leo is also a homosexual contributes to the unwillingness of the parents to
discus the matter with the young Marcel. I know from experience that on many o
ccasions most older people do not want to accept the fact that sometimes even
when someone is young this does not automatically means that he will not be ab
le to accept the reality and act in more adequate manner then even themselves
. With exception of the fact that the family tried to conceal the truth from M
arcel, they have left quite an impression for me  the way they supported thei
r son  even after discovering the truth about his sexuality and his sickness.
The fact that they allowed the young Marcel to travel along with Leo to Paris
to meet his ex boyfriend was quite a gesture from them most families I know w
ill be reluctant to do that. There is a lot of warmth in the scenes in which t
he brothers spend some time together  you can see them being real friends , c
oncern about each other.<br /><br />Close to Leo is an excellent drama, which
I strongly recommend
1
```

3. Data Cleaning

In [9]:
```
import nltk
from nltk.corpus import stopwords
from nltk.stem.porter import *

import re
from bs4 import BeautifulSoup

def review_to_words(review):
    nltk.download("stopwords", quiet=True)
    stemmer = PorterStemmer()

    text = BeautifulSoup(review, "html.parser").get_text() # Remove HTML tags
    text = re.sub(r"[^a-zA-Z0-9]", " ", text.lower()) # Convert to lower case
    words = text.split() # Split string into words
    words = [w for w in words if w not in stopwords.words("english")] # Remove
    words = [PorterStemmer().stem(w) for w in words] # stem

    return words
```

In [10]:
```
review_to_words(train_X[10])
```

Out[10]:
```
['movi',
 'collect',
 'friend',
 'mine',
 'remind',
 'decid',
 'watch',
 'know',
 'much',
 'close',
 'leo',
 'suppos',
 'excel',
 'come',
 'age',
 'movi',
 'deal',
 'seriou',
 'topic',
 'aid',
 'although',
 'person',
 'aid',
 'leo',
 'scenario',
 'wrap',
 'around',
 'way',
 'marcel',
 'youngest',
 'brother',
 'leo',
 'coup',
 'sick',
 'rel',
 'first',
 'everyon',
 'tri',
 'hide',
 'truth',
 'marcel',
 'believ',
 'young',
 'understand',
 'sick',
 'brother',
 'fact',
 'leo',
 'also',
 'homosexu',
 'contribut',
 'unwilling',
 'parent',
 'discu',
 'matter',
 'young',
 'marcel',
 'know',
 'experi',
 'mani',
```

```
'occas',
'older',
'peopl',
'want',
'accept',
'fact',
'sometim',
'even',
'someon',
'young',
'automat',
'mean',
'abl',
'accept',
'realiti',
'act',
'adequ',
'manner',
'even',
'except',
'fact',
'famili',
'tri',
'conceal',
'truth',
'marcel',
'left',
'quit',
'impress',
'way',
'support',
'son',
'even',
'discov',
'truth',
'sexual',
'sick',
'fact',
'allow',
'young',
'marcel',
'travel',
'along',
'leo',
'pari',
'meet',
'ex',
'boyfriend',
'quit',
'gestur',
'famili',
'know',
'reluct',
'lot',
'warmth',
'scene',
'brother',
'spend',
'time',
'togeth',
```

```
 'see',
 'real',
 'friend',
 'concern',
 'close',
 'leo',
 'excel',
 'drama',
 'strongli',
 'recommend']
```

In [11]:
```
import pickle

cache_dir = os.path.join("../cache", "movie_review_sentiment_analysis")  # whe
os.makedirs(cache_dir, exist_ok=True)  # ensure cache directory exists

def preprocess_data(data_train, data_test, labels_train, labels_test,
                    cache_dir=cache_dir, cache_file="preprocessed_data.pkl"):
    """Convert each review to words; read from cache if available."""

    # If cache_file is not None, try to read from it first
    cache_data = None
    if cache_file is not None:
        try:
            with open(os.path.join(cache_dir, cache_file), "rb") as f:
                cache_data = pickle.load(f)
            print("Read preprocessed data from cache file:", cache_file)
        except:
            pass  # unable to read from cache, but that's okay

    # If cache is missing, then do the heavy lifting
    if cache_data is None:
        # Preprocess training and test data to obtain words for each review
        #words_train = list(map(review_to_words, data_train))
        #words_test = list(map(review_to_words, data_test))
        words_train = [review_to_words(review) for review in data_train]
        words_test = [review_to_words(review) for review in data_test]

        # Write to cache file for future runs
        if cache_file is not None:
            cache_data = dict(words_train=words_train, words_test=words_test,
                              labels_train=labels_train, labels_test=labels_te
            with open(os.path.join(cache_dir, cache_file), "wb") as f:
                pickle.dump(cache_data, f)
            print("Wrote preprocessed data to cache file:", cache_file)
    else:
        # Unpack data loaded from cache file
        words_train, words_test, labels_train, labels_test = (cache_data['words
                cache_data['words_test'], cache_data['labels_train'], cache_da

    return words_train, words_test, labels_train, labels_test
```

In [ ]:
```
# Preprocess data
train_X, test_X, train_y, test_y = preprocess_data(train_X, test_X, train_y, t
```

```
/tmp/ipykernel_7947/2137687332.py:12: MarkupResemblesLocatorWarning: The input
looks more like a filename than markup. You may want to open this file and pas
s the filehandle into Beautiful Soup.
  text = BeautifulSoup(review, "html.parser").get_text() # Remove HTML tags
```

In [17]:
```
import numpy as np
from collections import Counter

def build_dict(data, vocab_size = 5000):
    """Construct and return a dictionary mapping each of the most frequently ap

    words = []
    for sentence in data:
        word = set(sentence)
        words.extend(word)
    word_count = Counter(words) # A dict storing the words that appear in the

    sorted_words = sorted(word_count, key=word_count.get, reverse=True)

    word_dict = {} # Word dictionary that translates words into integers
    for idx, word in enumerate(sorted_words[:vocab_size - 2]): # The -2 is so
        word_dict[word] = idx + 2                                # 'infrequent'

    return word_dict
```

In [18]:
```
word_dict = build_dict(train_X)
```

In [19]:
```
data_dir = '../data/pytorch' # The folder to store the data
if not os.path.exists(data_dir): # Check that the folder exists
    os.makedirs(data_dir)
```

In [20]:
```
with open(os.path.join(data_dir, 'word_dict.pkl'), "wb") as f:
    pickle.dump(word_dict, f)
```

4. Data Transformation

In [21]:
```
def convert_and_pad(word_dict, sentence, pad=500):
    NOWORD = 0 # 0 represents the 'no word' category
    INFREQ = 1 # 1 represents the infrequent words, i.e., words not appearing i

    working_sentence = [NOWORD] * pad

    for word_index, word in enumerate(sentence[:pad]):
        if word in word_dict:
            working_sentence[word_index] = word_dict[word]
        else:
            working_sentence[word_index] = INFREQ

    return working_sentence, min(len(sentence), pad)

def convert_and_pad_data(word_dict, data, pad=500):
    result = []
    lengths = []

    for sentence in data:
        converted, leng = convert_and_pad(word_dict, sentence, pad)
        result.append(converted)
        lengths.append(leng)

    return np.array(result), np.array(lengths)
```

In [22]:
```
train_X, train_X_len = convert_and_pad_data(word_dict, train_X)
test_X, test_X_len = convert_and_pad_data(word_dict, test_X)
```

In [23]:
```
# example one of the processed reviews.
train_X[0], train_X_len[0]
```

Out[23]:
```
(array([   1,  350, 1473,  612, 1340, 1942, 1578,  228,  595,    1, 1279,
        3681,    1,    1, 4302,  403, 2720,  142,  101,  382,  292,    4,
        1285,    3,   10,   30,  571, 1221,   53,    1,  108, 1158, 3109,
        1978,    1,    1,   78, 2010, 1500, 1505,    3, 4675,    1,  552,
         382, 1479,  163,   69, 2411,   20,  445,   34,    3,  552,  196,
           1,   17,    4,  114,   60, 1001, 4483,    3,    2, 2410, 1593,
          30,  716, 3509, 2451, 1929,  212,    6,  142,  111,  159,  108,
         111,  721,  368,  563,    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,    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]),
 81)
```

5. Uploading Data to S3

In [24]:
```
import pandas as pd

pd.concat([pd.DataFrame(train_y), pd.DataFrame(train_X_len), pd.DataFrame(trai
        .to_csv(os.path.join(data_dir, 'train.csv'), header=False, index=False
```

In [25]:
```
import sagemaker

sagemaker_session = sagemaker.Session()
```

```
bucket = sagemaker_session.default_bucket()
prefix = 'movie_review/sagemaker/sentiment_data'

role = sagemaker.get_execution_role()
```

```
sagemaker.config INFO - Not applying SDK defaults from location: /etc/xdg/sage
maker/config.yaml
sagemaker.config INFO - Not applying SDK defaults from location: /home/ec2-use
r/.config/sagemaker/config.yaml
```

In [26]:
```
input_data = sagemaker_session.upload_data(path=data_dir, bucket=bucket, key_p
```

6. Build Model

In [27]:
```
!pygmentize train/model.py
```

```
import torch.nn as nn

class LSTMClassifier(nn.Module):
    """
    This is the simple RNN model we will be using to perform Sentiment Analysi
s.
    """

    def __init__(self, embedding_dim, hidden_dim, vocab_size):
        """
        Initialize the model by settingg up the various layers.
        """
        super(LSTMClassifier, self).__init__()

        self.embedding = nn.Embedding(vocab_size, embedding_dim, padding_idx=
0)
        self.lstm = nn.LSTM(embedding_dim, hidden_dim)
        self.dense = nn.Linear(in_features=hidden_dim, out_features=1)
        self.sig = nn.Sigmoid()

        self.word_dict = None

    def forward(self, x):
        """
        Perform a forward pass of our model on some input.
        """
        x = x.t()
        lengths = x[0,:]
        reviews = x[1:,:]
        embeds = self.embedding(reviews)
        lstm_out, _ = self.lstm(embeds)
        out = self.dense(lstm_out)
        out = out[lengths - 1, range(len(lengths))]
        return self.sig(out.squeeze())
```

In [29]:
```
!pip install torch
```

```
Collecting torch
  Downloading torch-2.2.2-cp310-cp310-manylinux1_x86_64.whl.metadata (26 kB)
Requirement already satisfied: filelock in /home/ec2-user/anaconda3/envs/pytho
n3/lib/python3.10/site-packages (from torch) (3.13.1)
Requirement already satisfied: typing-extensions>=4.8.0 in /home/ec2-user/anac
onda3/envs/python3/lib/python3.10/site-packages (from torch) (4.9.0)
Requirement already satisfied: sympy in /home/ec2-user/anaconda3/envs/python3/
lib/python3.10/site-packages (from torch) (1.12)
Requirement already satisfied: networkx in /home/ec2-user/anaconda3/envs/pytho
n3/lib/python3.10/site-packages (from torch) (3.2.1)
Requirement already satisfied: jinja2 in /home/ec2-user/anaconda3/envs/python
3/lib/python3.10/site-packages (from torch) (3.1.3)
Requirement already satisfied: fsspec in /home/ec2-user/anaconda3/envs/python
3/lib/python3.10/site-packages (from torch) (2024.2.0)
Collecting nvidia-cuda-nvrtc-cu12==12.1.105 (from torch)
  Downloading nvidia_cuda_nvrtc_cu12-12.1.105-py3-none-manylinux1_x86_64.whl.m
etadata (1.5 kB)
Collecting nvidia-cuda-runtime-cu12==12.1.105 (from torch)
  Downloading nvidia_cuda_runtime_cu12-12.1.105-py3-none-manylinux1_x86_64.wh
l.metadata (1.5 kB)
Collecting nvidia-cuda-cupti-cu12==12.1.105 (from torch)
  Downloading nvidia_cuda_cupti_cu12-12.1.105-py3-none-manylinux1_x86_64.whl.m
etadata (1.6 kB)
Collecting nvidia-cudnn-cu12==8.9.2.26 (from torch)
  Downloading nvidia_cudnn_cu12-8.9.2.26-py3-none-manylinux1_x86_64.whl.metada
ta (1.6 kB)
Collecting nvidia-cublas-cu12==12.1.3.1 (from torch)
  Downloading nvidia_cublas_cu12-12.1.3.1-py3-none-manylinux1_x86_64.whl.metad
ata (1.5 kB)
Collecting nvidia-cufft-cu12==11.0.2.54 (from torch)
  Downloading nvidia_cufft_cu12-11.0.2.54-py3-none-manylinux1_x86_64.whl.metad
ata (1.5 kB)
Collecting nvidia-curand-cu12==10.3.2.106 (from torch)
  Downloading nvidia_curand_cu12-10.3.2.106-py3-none-manylinux1_x86_64.whl.met
adata (1.5 kB)
Collecting nvidia-cusolver-cu12==11.4.5.107 (from torch)
  Downloading nvidia_cusolver_cu12-11.4.5.107-py3-none-manylinux1_x86_64.whl.m
etadata (1.6 kB)
Collecting nvidia-cusparse-cu12==12.1.0.106 (from torch)
  Downloading nvidia_cusparse_cu12-12.1.0.106-py3-none-manylinux1_x86_64.whl.m
etadata (1.6 kB)
Collecting nvidia-nccl-cu12==2.19.3 (from torch)
  Downloading nvidia_nccl_cu12-2.19.3-py3-none-manylinux1_x86_64.whl.metadata
(1.8 kB)
Collecting nvidia-nvtx-cu12==12.1.105 (from torch)
  Downloading nvidia_nvtx_cu12-12.1.105-py3-none-manylinux1_x86_64.whl.metadat
a (1.7 kB)
Collecting triton==2.2.0 (from torch)
  Downloading triton-2.2.0-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86
_64.whl.metadata (1.4 kB)
Collecting nvidia-nvjitlink-cu12 (from nvidia-cusolver-cu12==11.4.5.107->torc
h)
  Downloading nvidia_nvjitlink_cu12-12.4.127-py3-none-manylinux2014_x86_64.wh
l.metadata (1.5 kB)
Requirement already satisfied: MarkupSafe>=2.0 in /home/ec2-user/anaconda3/env
s/python3/lib/python3.10/site-packages (from jinja2->torch) (2.1.5)
Requirement already satisfied: mpmath>=0.19 in /home/ec2-user/anaconda3/envs/p
ython3/lib/python3.10/site-packages (from sympy->torch) (1.3.0)
Downloading torch-2.2.2-cp310-cp310-manylinux1_x86_64.whl (755.5 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 755.5/755.5 MB 405.6 kB/s eta 0:00:
```

```
0000:0100:01
Downloading nvidia_cublas_cu12-12.1.3.1-py3-none-manylinux1_x86_64.whl (410.6
MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 410.6/410.6 MB 1.5 MB/s eta 0:00:0
0:00:0100:01
Downloading nvidia_cuda_cupti_cu12-12.1.105-py3-none-manylinux1_x86_64.whl (1
4.1 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 14.1/14.1 MB 25.9 MB/s eta 0:00:00
00:0100:01
Downloading nvidia_cuda_nvrtc_cu12-12.1.105-py3-none-manylinux1_x86_64.whl (2
3.7 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 23.7/23.7 MB 18.8 MB/s eta 0:00:00
00:0100:01
Downloading nvidia_cuda_runtime_cu12-12.1.105-py3-none-manylinux1_x86_64.whl
(823 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 823.6/823.6 kB 14.9 MB/s eta 0:00:
0000:01
Downloading nvidia_cudnn_cu12-8.9.2.26-py3-none-manylinux1_x86_64.whl (731.7 M
B)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 731.7/731.7 MB 1.4 MB/s eta 0:00:0
0:00:0100:01
Downloading nvidia_cufft_cu12-11.0.2.54-py3-none-manylinux1_x86_64.whl (121.6
MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 121.6/121.6 MB 1.7 MB/s eta 0:00:0
0:00:0100:01
Downloading nvidia_curand_cu12-10.3.2.106-py3-none-manylinux1_x86_64.whl (56.5
MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 56.5/56.5 MB 6.2 MB/s eta 0:00:00:
00:0100:01
Downloading nvidia_cusolver_cu12-11.4.5.107-py3-none-manylinux1_x86_64.whl (12
4.2 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 124.2/124.2 MB 5.3 MB/s eta 0:00:0
0:00:0100:01
Downloading nvidia_cusparse_cu12-12.1.0.106-py3-none-manylinux1_x86_64.whl (19
6.0 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 196.0/196.0 MB 4.0 MB/s eta 0:00:0
0:00:0100:01
Downloading nvidia_nccl_cu12-2.19.3-py3-none-manylinux1_x86_64.whl (166.0 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 166.0/166.0 MB 4.9 MB/s eta 0:00:0
0:00:0100:01
Downloading nvidia_nvtx_cu12-12.1.105-py3-none-manylinux1_x86_64.whl (99 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 99.1/99.1 kB 827.5 kB/s eta 0:00:0
0a 0:00:01
Downloading triton-2.2.0-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_6
4.whl (167.9 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 167.9/167.9 MB 4.7 MB/s eta 0:00:0
0:00:0100:01
Downloading nvidia_nvjitlink_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl
(21.1 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 21.1/21.1 MB 28.8 MB/s eta 0:00:00
00:0100:01
Installing collected packages: triton, nvidia-nvtx-cu12, nvidia-nvjitlink-cu1
2, nvidia-nccl-cu12, nvidia-curand-cu12, nvidia-cufft-cu12, nvidia-cuda-runtim
e-cu12, nvidia-cuda-nvrtc-cu12, nvidia-cuda-cupti-cu12, nvidia-cublas-cu12, nv
idia-cusparse-cu12, nvidia-cudnn-cu12, nvidia-cusolver-cu12, torch
Successfully installed nvidia-cublas-cu12-12.1.3.1 nvidia-cuda-cupti-cu12-12.
1.105 nvidia-cuda-nvrtc-cu12-12.1.105 nvidia-cuda-runtime-cu12-12.1.105 nvidia
-cudnn-cu12-8.9.2.26 nvidia-cufft-cu12-11.0.2.54 nvidia-curand-cu12-10.3.2.106
nvidia-cusolver-cu12-11.4.5.107 nvidia-cusparse-cu12-12.1.0.106 nvidia-nccl-cu
```

```
12-2.19.3 nvidia-nvjitlink-cu12-12.4.127 nvidia-nvtx-cu12-12.1.105 torch-2.2.2
triton-2.2.0
```

In [30]:
```
import torch
import torch.utils.data

# Read in the first 250 rows
train_sample = pd.read_csv(os.path.join(data_dir, 'train.csv'), header=None, na

# Turn the input pandas dataframe into tensors
train_sample_y = torch.from_numpy(train_sample[[0]].values).float().squeeze()
train_sample_X = torch.from_numpy(train_sample.drop([0], axis=1).values).long(

# Build the dataset
train_sample_ds = torch.utils.data.TensorDataset(train_sample_X, train_sample_
# Build the dataloader
train_sample_dl = torch.utils.data.DataLoader(train_sample_ds, batch_size=50)
```

In [31]:
```
def train(model, train_loader, epochs, optimizer, loss_fn, device):
    """
    This is the training method that is called by the PyTorch training script.
    passed are as follows:
    model        - The PyTorch model that we wish to train.
    train_loader - The PyTorch DataLoader that should be used during training.
    epochs       - The total number of epochs to train for.
    optimizer    - The optimizer to use during training.
    loss_fn      - The loss function used for training.
    device       - Where the model and data should be loaded (gpu or cpu).
    """

    for epoch in range(1, epochs + 1):
        model.train()
        total_loss = 0
        for batch in train_loader:
            batch_X, batch_y = batch

            batch_X = batch_X.to(device)
            batch_y = batch_y.to(device)

            # clear the gradients
            optimizer.zero_grad()
            # forward pass
            outputs = model(batch_X)
            # prediction
#             _, preds = torch.max(outputs, 1)
            # calculate loss
            loss = loss_fn(outputs, batch_y)
            # bacward pass
            loss.backward()
            # optimization
            optimizer.step()

            total_loss += loss.data.item()
        print("Epoch: {}, BCELoss: {}".format(epoch, total_loss / len(train_loa
```

In [32]:
```
# check GPU availability
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
device
```

Out[32]:
```
device(type='cpu')
```

In [33]:
```
import torch.optim as optim
from train.model import LSTMClassifier

model = LSTMClassifier(32, 100, 5000).to(device)
optimizer = optim.Adam(model.parameters())
loss_fn = torch.nn.BCELoss()

train(model, train_sample_dl, 5, optimizer, loss_fn, device)
```

```
Epoch: 1, BCELoss: 0.6946756720542908
Epoch: 2, BCELoss: 0.6845051765441894
Epoch: 3, BCELoss: 0.675368320941925
Epoch: 4, BCELoss: 0.6648377060890198
Epoch: 5, BCELoss: 0.6511309385299683
```

In [37]:
```
from sagemaker.pytorch import PyTorch

estimator = PyTorch(entry_point="train.py",
                    source_dir="train",
                    role=role,
                    framework_version='1.10.0',
                    py_version='py38',
                    train_instance_count=1,
                    train_instance_type='ml.m4.xlarge',  # Updated to a valid 
                    hyperparameters={
                        'epochs': 10,
                        'hidden_dim': 200,
                    })
```

```
train_instance_count has been renamed in sagemaker>=2.
See: https://sagemaker.readthedocs.io/en/stable/v2.html for details.
train_instance_type has been renamed in sagemaker>=2.
See: https://sagemaker.readthedocs.io/en/stable/v2.html for details.
```

In [38]:
```
estimator.fit({'training': input_data})
```

```
INFO:sagemaker.image_uris:image_uri is not presented, retrieving image_uri bas
ed on instance_type, framework etc.
INFO:sagemaker:Creating training-job with name: pytorch-training-2024-04-20-00
-01-07-679
```

```
2024-04-20 00:01:08 Starting - Starting the training job...
2024-04-20 00:01:22 Starting - Preparing the instances for training......
2024-04-20 00:02:19 Downloading - Downloading input data...
2024-04-20 00:02:59 Downloading - Downloading the training image......
2024-04-20 00:03:49 Training - Training image download completed. Training in progress.bash: cannot set terminal process group (-1): Inappropriate ioctl for device
bash: no job control in this shell
2024-04-20 00:03:58,811 sagemaker-training-toolkit INFO     Imported framework sagemaker_pytorch_container.training
2024-04-20 00:03:58,813 sagemaker-training-toolkit INFO     No GPUs detected (normal if no gpus installed)
2024-04-20 00:03:58,825 sagemaker_pytorch_container.training INFO     Block until all host DNS lookups succeed.
2024-04-20 00:03:58,830 sagemaker_pytorch_container.training INFO     Invoking user training script.
2024-04-20 00:03:59,020 sagemaker-training-toolkit INFO     Installing dependencies from requirements.txt:
/opt/conda/bin/python3.8 -m pip install -r requirements.txt
Requirement already satisfied: pandas in /opt/conda/lib/python3.8/site-packages (from -r requirements.txt (line 1)) (1.3.4)
Requirement already satisfied: numpy in /opt/conda/lib/python3.8/site-packages (from -r requirements.txt (line 2)) (1.21.2)
Collecting nltk
Downloading nltk-3.8.1-py3-none-any.whl (1.5 MB)
Collecting beautifulsoup4
Downloading beautifulsoup4-4.12.3-py3-none-any.whl (147 kB)
Collecting html5lib
Downloading html5lib-1.1-py2.py3-none-any.whl (112 kB)
Requirement already satisfied: python-dateutil>=2.7.3 in /opt/conda/lib/python3.8/site-packages (from pandas->-r requirements.txt (line 1)) (2.8.2)
Requirement already satisfied: pytz>=2017.3 in /opt/conda/lib/python3.8/site-packages (from pandas->-r requirements.txt (line 1)) (2021.3)
Requirement already satisfied: click in /opt/conda/lib/python3.8/site-packages (from nltk->-r requirements.txt (line 3)) (8.0.3)
Collecting regex>=2021.8.3
Downloading regex-2024.4.16-cp38-cp38-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (777 kB)
Requirement already satisfied: joblib in /opt/conda/lib/python3.8/site-packages (from nltk->-r requirements.txt (line 3)) (1.1.0)
Requirement already satisfied: tqdm in /opt/conda/lib/python3.8/site-packages (from nltk->-r requirements.txt (line 3)) (4.62.3)
Collecting soupsieve>1.2
Downloading soupsieve-2.5-py3-none-any.whl (36 kB)
Requirement already satisfied: six>=1.9 in /opt/conda/lib/python3.8/site-packages (from html5lib->-r requirements.txt (line 5)) (1.16.0)
Collecting webencodings
Downloading webencodings-0.5.1-py2.py3-none-any.whl (11 kB)
Installing collected packages: webencodings, soupsieve, regex, nltk, html5lib, beautifulsoup4
Successfully installed beautifulsoup4-4.12.3 html5lib-1.1 nltk-3.8.1 regex-2024.4.16 soupsieve-2.5 webencodings-0.5.1
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the system package manager. It is recommended to use a virtual environment instead: https://pip.pypa.io/warnings/venv
2024-04-20 00:04:04,796 sagemaker-training-toolkit INFO     No GPUs detected (normal if no gpus installed)
2024-04-20 00:04:04,811 sagemaker-training-toolkit INFO     No GPUs detected (normal if no gpus installed)
2024-04-20 00:04:04,825 sagemaker-training-toolkit INFO     No GPUs detected
```

```
(normal if no gpus installed)
2024-04-20 00:04:04,837 sagemaker-training-toolkit INFO     Invoking user scri
pt
Training Env:
{
    "additional_framework_parameters": {},
    "channel_input_dirs": {
        "training": "/opt/ml/input/data/training"
    },
    "current_host": "algo-1",
    "framework_module": "sagemaker_pytorch_container.training:main",
    "hosts": [
        "algo-1"
    ],
    "hyperparameters": {
        "epochs": 10,
        "hidden_dim": 200
    },
    "input_config_dir": "/opt/ml/input/config",
    "input_data_config": {
        "training": {
            "TrainingInputMode": "File",
            "S3DistributionType": "FullyReplicated",
            "RecordWrapperType": "None"
        }
    },
    "input_dir": "/opt/ml/input",
    "is_master": true,
    "job_name": "pytorch-training-2024-04-20-00-01-07-679",
    "log_level": 20,
    "master_hostname": "algo-1",
    "model_dir": "/opt/ml/model",
    "module_dir": "s3://sagemaker-us-east-2-637423629228/pytorch-training-2024
-04-20-00-01-07-679/source/sourcedir.tar.gz",
    "module_name": "train",
    "network_interface_name": "eth0",
    "num_cpus": 4,
    "num_gpus": 0,
    "output_data_dir": "/opt/ml/output/data",
    "output_dir": "/opt/ml/output",
    "output_intermediate_dir": "/opt/ml/output/intermediate",
    "resource_config": {
        "current_host": "algo-1",
        "current_instance_type": "ml.m4.xlarge",
        "current_group_name": "homogeneousCluster",
        "hosts": [
            "algo-1"
        ],
        "instance_groups": [
            {
                "instance_group_name": "homogeneousCluster",
                "instance_type": "ml.m4.xlarge",
                "hosts": [
                    "algo-1"
                ]
            }
        ],
        "network_interface_name": "eth0"
    },
    "user_entry_point": "train.py"
```

```
}
Environment variables:
SM_HOSTS=["algo-1"]
SM_NETWORK_INTERFACE_NAME=eth0
SM_HPS={"epochs":10,"hidden_dim":200}
SM_USER_ENTRY_POINT=train.py
SM_FRAMEWORK_PARAMS={}
SM_RESOURCE_CONFIG={"current_group_name":"homogeneousCluster","current_hos
t":"algo-1","current_instance_type":"ml.m4.xlarge","hosts":["algo-1"],"instanc
e_groups":[{"hosts":["algo-1"],"instance_group_name":"homogeneousCluster","ins
tance_type":"ml.m4.xlarge"}],"network_interface_name":"eth0"}
SM_INPUT_DATA_CONFIG={"training":{"RecordWrapperType":"None","S3DistributionTy
pe":"FullyReplicated","TrainingInputMode":"File"}}
SM_OUTPUT_DATA_DIR=/opt/ml/output/data
SM_CHANNELS=["training"]
SM_CURRENT_HOST=algo-1
SM_MODULE_NAME=train
SM_LOG_LEVEL=20
SM_FRAMEWORK_MODULE=sagemaker_pytorch_container.training:main
SM_INPUT_DIR=/opt/ml/input
SM_INPUT_CONFIG_DIR=/opt/ml/input/config
SM_OUTPUT_DIR=/opt/ml/output
SM_NUM_CPUS=4
SM_NUM_GPUS=0
SM_MODEL_DIR=/opt/ml/model
SM_MODULE_DIR=s3://sagemaker-us-east-2-637423629228/pytorch-training-2024-04-2
0-00-01-07-679/source/sourcedir.tar.gz
SM_TRAINING_ENV={"additional_framework_parameters":{},"channel_input_dirs":{"t
raining":"/opt/ml/input/data/training"},"current_host":"algo-1","framework_mod
ule":"sagemaker_pytorch_container.training:main","hosts":["algo-1"],"hyperpara
meters":{"epochs":10,"hidden_dim":200},"input_config_dir":"/opt/ml/input/confi
g","input_data_config":{"training":{"RecordWrapperType":"None","S3Distribution
Type":"FullyReplicated","TrainingInputMode":"File"}},"input_dir":"/opt/ml/inpu
t","is_master":true,"job_name":"pytorch-training-2024-04-20-00-01-07-679","log
_level":20,"master_hostname":"algo-1","model_dir":"/opt/ml/model","module_di
r":"s3://sagemaker-us-east-2-637423629228/pytorch-training-2024-04-20-00-01-07
-679/source/sourcedir.tar.gz","module_name":"train","network_interface_nam
e":"eth0","num_cpus":4,"num_gpus":0,"output_data_dir":"/opt/ml/output/data","o
utput_dir":"/opt/ml/output","output_intermediate_dir":"/opt/ml/output/intermed
iate","resource_config":{"current_group_name":"homogeneousCluster","current_ho
st":"algo-1","current_instance_type":"ml.m4.xlarge","hosts":["algo-1"],"instanc
e_groups":[{"hosts":["algo-1"],"instance_group_name":"homogeneousCluster","in
stance_type":"ml.m4.xlarge"}],"network_interface_name":"eth0"},"user_entry_poi
nt":"train.py"}
SM_USER_ARGS=["--epochs","10","--hidden_dim","200"]
SM_OUTPUT_INTERMEDIATE_DIR=/opt/ml/output/intermediate
SM_CHANNEL_TRAINING=/opt/ml/input/data/training
SM_HP_EPOCHS=10
SM_HP_HIDDEN_DIM=200
PYTHONPATH=/opt/ml/code:/opt/conda/bin:/opt/conda/lib/python38.zip:/opt/conda/
lib/python3.8:/opt/conda/lib/python3.8/lib-dynload:/opt/conda/lib/python3.8/si
te-packages
Invoking script with the following command:
/opt/conda/bin/python3.8 train.py --epochs 10 --hidden_dim 200
Using device cpu.
Get train data loader.
Model loaded with embedding_dim 32, hidden_dim 200, vocab_size 5000.
[2024-04-20 00:04:07.078 algo-1:34 INFO utils.py:27] RULE_JOB_STOP_SIGNAL_FILE
NAME: None
[2024-04-20 00:04:07.148 algo-1:34 INFO profiler_config_parser.py:102] User ha
```

```
s disabled profiler.
[2024-04-20 00:04:07.148 algo-1:34 INFO json_config.py:91] Creating hook from
json_config at /opt/ml/input/config/debughookconfig.json.
[2024-04-20 00:04:07.149 algo-1:34 INFO hook.py:200] tensorboard_dir has not b
een set for the hook. SMDebug will not be exporting tensorboard summaries.
[2024-04-20 00:04:07.149 algo-1:34 INFO hook.py:255] Saving to /opt/ml/output/
tensors
[2024-04-20 00:04:07.149 algo-1:34 INFO state_store.py:77] The checkpoint conf
ig file /opt/ml/input/config/checkpointconfig.json does not exist.
[2024-04-20 00:04:07.156 algo-1:34 INFO hook.py:591] name:embedding.weight cou
nt_params:160000
[2024-04-20 00:04:07.156 algo-1:34 INFO hook.py:591] name:lstm.weight_ih_l0 co
unt_params:25600
[2024-04-20 00:04:07.156 algo-1:34 INFO hook.py:591] name:lstm.weight_hh_l0 co
unt_params:160000
[2024-04-20 00:04:07.156 algo-1:34 INFO hook.py:591] name:lstm.bias_ih_l0 coun
t_params:800
[2024-04-20 00:04:07.156 algo-1:34 INFO hook.py:591] name:lstm.bias_hh_l0 coun
t_params:800
[2024-04-20 00:04:07.156 algo-1:34 INFO hook.py:591] name:dense.weight count_p
arams:200
[2024-04-20 00:04:07.156 algo-1:34 INFO hook.py:591] name:dense.bias count_par
ams:1
[2024-04-20 00:04:07.157 algo-1:34 INFO hook.py:593] Total Trainable Params: 3
47401
[2024-04-20 00:04:07.157 algo-1:34 INFO hook.py:424] Monitoring the collection
s: losses
[2024-04-20 00:04:07.160 algo-1:34 INFO hook.py:488] Hook is writing from the
hook with pid: 34
Epoch: 1, BCELoss: 0.6682921623697087
Epoch: 2, BCELoss: 0.6077206718678377
Epoch: 3, BCELoss: 0.5563158514548321
Epoch: 4, BCELoss: 0.4570438880093244
Epoch: 5, BCELoss: 0.4085615812515726
Epoch: 6, BCELoss: 0.3655042715218602
Epoch: 7, BCELoss: 0.34548429627807775
Epoch: 8, BCELoss: 0.32394267649066694
Epoch: 9, BCELoss: 0.29707914955761966

2024-04-20 00:46:43 Uploading - Uploading generated training modelEpoch: 10, B
CELoss: 0.3147812634706497
2024-04-20 00:46:36,974 sagemaker-training-toolkit INFO     Reporting training
SUCCESS

2024-04-20 00:46:54 Completed - Training job completed
Training seconds: 2675
Billable seconds: 2675
```

8. Deploy the Model

In [39]:
```
predictor = estimator.deploy(initial_instance_count=1, instance_type='ml.m4.xl
```

```
INFO:sagemaker:Repacking model artifact (s3://sagemaker-us-east-2-637423629228/pytorch-training-2024-04-20-00-01-07-679/output/model.tar.gz), script artifact (s3://sagemaker-us-east-2-637423629228/pytorch-training-2024-04-20-00-01-07-679/source/sourcedir.tar.gz), and dependencies ([]) into single tar.gz file located at s3://sagemaker-us-east-2-637423629228/pytorch-training-2024-04-20-00-52-36-559/model.tar.gz. This may take some time depending on model size...
INFO:sagemaker:Creating model with name: pytorch-training-2024-04-20-00-52-36-559
INFO:sagemaker:Creating endpoint-config with name pytorch-training-2024-04-20-00-52-36-559
INFO:sagemaker:Creating endpoint with name pytorch-training-2024-04-20-00-52-36-559
------!
```

9. Model Testing

In [40]:
```
test_X = pd.concat([pd.DataFrame(test_X_len), pd.DataFrame(test_X)], axis=1)
```

In [41]:
```
# We split the data into chunks and send each chunk seperately, accumulating t

def predict(data, rows=512):
    split_array = np.array_split(data, int(data.shape[0] / float(rows) + 1))
    predictions = np.array([])
    for array in split_array:
        predictions = np.append(predictions, predictor.predict(array))

    return predictions
```

In [42]:
```
predictions = predict(test_X.values)
predictions = [round(num) for num in predictions]
```

In [43]:
```
from sklearn.metrics import accuracy_score
accuracy_score(test_y, predictions)
```

Out[43]:
```
0.8556
```

In [44]:
```
test_review = 'This an Amazing Movie. Very good story and plot'
```

In [45]:
```
# Convert test_review into a form usable by the model and save the results in 
test_review_words = review_to_words(test_review)
test_review_words, test_review_len = convert_and_pad(word_dict, test_review_wo
test_data = np.hstack((test_review_len, test_review_words))
test_data = test_data.reshape(1, -1)
test_data.shape, test_data[0, :8]
```

Out[45]:
```
((1, 501), array([  5, 355,   2,   7,  15,  41,   0,   0]))
```

In [46]:
```
predictor.predict(test_data)
```

Out[46]:
```
array(0.66853648)
```

10. Inference Code for Model

In [99]:
```
!pygmentize serve/predict.py
```

```
import argparse
import json
import os
import pickle
import sys
import pandas as pd
import numpy as np
import torch
import torch.nn as nn
import torch.optim as optim
import torch.utils.data

from model import LSTMClassifier

from utils import review_to_words, convert_and_pad

def model_fn(model_dir):
    """Load the PyTorch model from the `model_dir` directory."""
    print("Loading model.")

    # First, load the parameters used to create the model.
    model_info = {}
    model_info_path = os.path.join(model_dir, 'model_info.pth')
    with open(model_info_path, 'rb') as f:
        model_info = torch.load(f)

    print("model_info: {}".format(model_info))

    # Determine the device and construct the model.
    device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
    model = LSTMClassifier(model_info['embedding_dim'], model_info['hidden_di
m'], model_info['vocab_size'])

    # Load the store model parameters.
    model_path = os.path.join(model_dir, 'model.pth')
    with open(model_path, 'rb') as f:
        model.load_state_dict(torch.load(f))

    # Load the saved word_dict.
    word_dict_path = os.path.join(model_dir, 'word_dict.pkl')
    with open(word_dict_path, 'rb') as f:
        model.word_dict = pickle.load(f)

    model.to(device).eval()

    print("Done loading model.")
    return model

def input_fn(serialized_input_data, content_type):
    print('Deserializing the input data.')
    if content_type == 'text/plain':
        data = serialized_input_data.decode('utf-8')
        return data
    raise Exception('Requested unsupported ContentType in content_type: ' + co
ntent_type)

def output_fn(prediction_output, accept):
    print('Serializing the generated output.')
    return str(prediction_output)
```

```
def predict_fn(input_data, model):
    print('Inferring sentiment of input data.')

    device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

    if model.word_dict is None:
        raise Exception('Model has not been loaded properly, no word_dict.')

    #         data_X   - A sequence of length 500 which represents the convert
ed review
    #         data_len - The length of the review
    data_X = review_to_words(input_data)
    data_X, data_len = convert_and_pad(model.word_dict, data_X)

    # Using data_X and data_len we construct an appropriate input tensor. Reme
mber
    # that our model expects input data of the form 'len, review[500]'.
    data_pack = np.hstack((data_len, data_X))
    data_pack = data_pack.reshape(1, -1)

    data = torch.from_numpy(data_pack)
    data = data.to(device)

    # Make sure to put the model into evaluation mode
    model.eval()

    # TODO: Compute the result of applying the model to the input data. The va
riable `result` should
    #       be a numpy array which contains a single integer which is either 1
or 0

    output = model(data).detach().cpu().numpy()
    result = np.round(output).astype(np.int)

    return result
```

11. Deploy the Model

In [101…

```
from sagemaker.predictor import RealTimePredictor
from sagemaker.pytorch import PyTorchModel

class StringPredictor(RealTimePredictor):
    def __init__(self, endpoint_name, sagemaker_session):
        super(StringPredictor, self).__init__(endpoint_name, sagemaker_session
```

In [103…

```
model = PyTorchModel(model_data=estimator.model_data,
                     role=role,
                     framework_version='1.10.0',
                     py_version='py38',
                     entry_point='predict.py',
                     source_dir='serve',
                     predictor_cls=StringPredictor)

predictor = model.deploy(initial_instance_count=1, instance_type='ml.m4.xlarge
```

```
INFO:sagemaker:Repacking model artifact (s3://sagemaker-us-east-2-637423629228/pytorch-training-2024-04-20-00-01-07-679/output/model.tar.gz), script artifact (serve), and dependencies ([]) into single tar.gz file located at s3://sagemaker-us-east-2-637423629228/pytorch-inference-2024-04-20-02-41-37-624/model.tar.gz. This may take some time depending on model size...
INFO:sagemaker:Creating model with name: pytorch-inference-2024-04-20-02-41-38-385
INFO:sagemaker:Creating endpoint-config with name pytorch-inference-2024-04-20-02-41-38-968
INFO:sagemaker:Creating endpoint with name pytorch-inference-2024-04-20-02-41-38-968
------!
```

```
WARNING:sagemaker.deprecations:The class RealTimePredictor has been renamed in sagemaker>=2.
See: https://sagemaker.readthedocs.io/en/stable/v2.html for details.
WARNING:sagemaker.deprecations:content_type is a no-op in sagemaker>=2.
See: https://sagemaker.readthedocs.io/en/stable/v2.html for details.
```

12. Testing the Model

In [109…

```
import glob

def test_reviews(data_dir='../data/aclImdb', stop=250):

    results = []
    ground = []

    # We make sure to test both positive and negative reviews
    for sentiment in ['pos', 'neg']:

        path = os.path.join(data_dir, 'test', sentiment, '*.txt')
        files = glob.glob(path)

        files_read = 0

        print('Starting ', sentiment, ' files')

        # Iterate through the files and send them to the predictor
        for f in files:
            with open(f) as review:
                # First, we store the ground truth (was the review positive or
                if sentiment == 'pos':
                    ground.append(1)
                else:
                    ground.append(0)
                # Read in the review and convert to 'utf-8' for transmission vi
                review_input = review.read().encode('utf-8')
                # Send the review to the predictor and store the results
                results.append(int(predictor.predict(review_input, initial_args

            # Sending reviews to our endpoint one at a time takes a while so we
            # only send a small number of reviews
            files_read += 1
            if files_read == stop:
                break

    return ground, results
```

In [110…
```
ground, results = test_reviews()
```

```
Starting  pos  files
Starting  neg  files
```

In [111…
```
from sklearn.metrics import accuracy_score
accuracy_score(ground, results)
```

Out[111]:
```
0.872
```

In [115…
```
test_review = 'The Movie was very bad! poor!'
```

In [116…
```
predictor.predict(test_review, initial_args={'ContentType':'text/plain'})
```

Out[116]:
```
b'0'
```

In [114…
```
predictor.endpoint
```

```
WARNING:sagemaker.deprecations:The endpoint attribute has been renamed in sage
maker>=2.
See: https://sagemaker.readthedocs.io/en/stable/v2.html for details.
```

Out[114]:
```
'pytorch-inference-2024-04-20-02-41-38-968'
```

In [117…
```
# https://vtfg80f45l.execute-api.us-east-2.amazonaws.com/Prod
```

In [120…
```
# zip -r -X INFO6105-Project.zip INFO6105-Project
```

13. Model Use for Web App

Setting up a Lambda function

1. Create an IAM Role for the Lambda function
2. Create a Lambda function
3. Setting up API Gateway
4. Deploying the web app

In [123…
```
!pygmentize lambda_function.py
```

```
import boto3

def lambda_handler(event, context):

    # The SageMaker runtime is what allows us to invoke the endpoint that we've created.
    runtime = boto3.Session().client('sagemaker-runtime')

    # Now we use the SageMaker runtime to invoke our endpoint, sending the review we were given
    response = runtime.invoke_endpoint(EndpointName = 'pytorch-inference-2024-04-20-02-41-38-968',    # The name of the endpoint we created
                                       ContentType = 'text/plain',
# The data format that is expected
                                       Body = event['body'])
# The actual review

    # The response is an HTTP response whose body contains the result of our inference
    result = response['Body'].read().decode('utf-8')

    return {
        'statusCode' : 200,
        'headers' : { 'Content-Type' : 'text/plain', 'Access-Control-Allow-Origin' : '*' },
        'body' : result
    }
```

In [ ]: