

Modelling Competition – Assignment 6

1. Pre-processing

The data was downloaded using the link given in the prompt and then split into train, eval and dev with a 85%, 10% and 5 % split. The below code was used for the splitting.

Splitting the dataset

```

In [7]: M 1 import pandas as pd
        2 import numpy as np

In [14]: M 1 tweet_file = pd.read_csv("training.full.csv")

In [15]: M 1 #shuffle and split
        2 from sklearn.utils import shuffle
        3 tweet_file = shuffle(tweet_file)

In [16]: M 1 from sklearn.model_selection import train_test_split
        2 train, test = train_test_split(tweet_file, test_size=0.15)

In [17]: M 1 dev, eval = train_test_split(test, test_size=0.3333)

In [18]: M 1 train.to_csv("train.csv", sep=',', encoding='utf-8', index = False)
        2 dev.to_csv("dev.csv", sep=',', encoding='utf-8', index = False)
        3 eval.to_csv("eval.csv", sep=',', encoding='utf-8', index = False)

In [19]: M 1 train
    
```

Out[19]:

	Sentiment	TwitterID	Date	QueryType	UserID	Tweet
1041580	4	1957099141	Thu May 28 23:30:54 PDT 2009	NO_QUERY	rehamc	Fact of the day: if I don't know you and you a...
115691	0	1827188285	Sun May 17 10:10:47 PDT 2009	NO_QUERY	MitalandMe	@siobhanvivian Haha. My mom does that all the ...
90718	0	1759124979	Sun May 10 18:31:03 PDT 2009	NO_QUERY	beckuhk	@teamalexccc oh, is he talking bout lize?
82332	0	1753020982	Sat May 09 23:49:44 PDT 2009	NO_QUERY	sheswithband	Flaw: Putting too much faith in people.
1366604	4	2050228781	Fri Jun 05 18:17:35 PDT 2009	NO_QUERY	defnatale	@calliness I think so. I have no idea. They c...
1423574	4	2058706235	Sat Jun 06 15:28:39 PDT 2009	NO_QUERY	chimmini	Clint and Emily - the wedding was beautiful. W...
735962	0	2254905111	Sun Jun 21 05:15:25 PDT 2009	NO_QUERY	dandnell	@mrmiddleweek bit worried about travelling do...

This split csv file was uploaded on the S3 bucket and then we ran crawlers on them to make tables on AWS.

Services ▾ Resource Groups ▾

Tables: A table is the metadata definition that represents your data, including its schema. A table can be used as a source or target in a job definition.

Name	Database	Location	Classification
dev	devdb	s3://group16-assignment-6/data/dev/	csv
eval	evaldb	s3://group16-assignment-6/data/eval/	csv
train	traindb	s3://group16-assignment-6/data/train/	csv

Integrating the code on our Git with TravisCI

Pylint and Pytest of the Preprocessing code

We were able to achieve a Pylint score of 10/10 after many iterations of updating the code and having the correct variables as well the correct alignment for each of the functions which are a part of our code.

```
===== 4 passed in 1.08s =====
The command "pytest --cov ./" exited with 0.
$ pylint ./twitter_file.py

-----
Your code has been rated at 10.00/10

The command "pylint ./twitter_file.py" exited with 0.
$ pylint ./twitter_file_test.py

-----
Your code has been rated at 10.00/10

The command "pylint ./twitter_file_test.py" exited with 0.
```

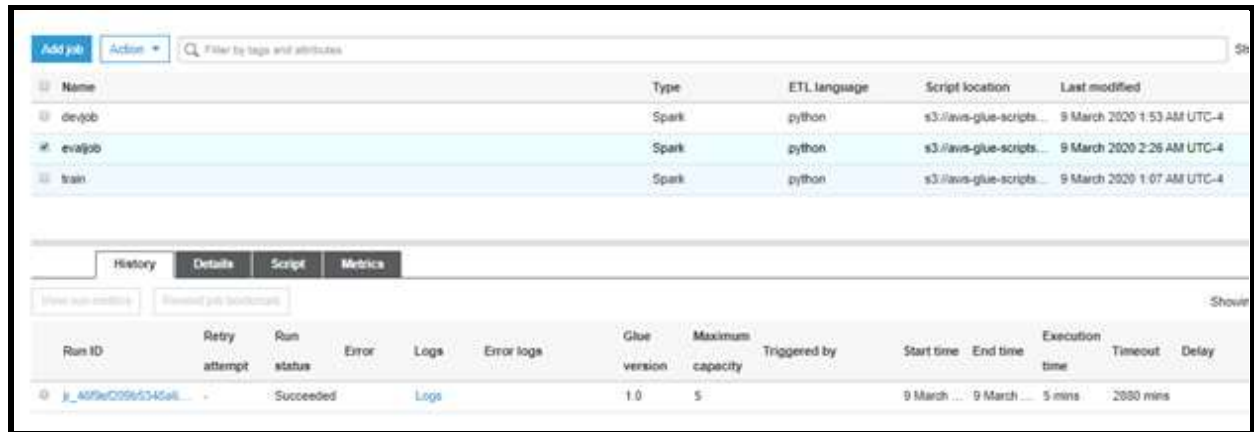
Coverage of the preprocessing code: We were able to get a coverage of 82% for our code. The images below report the coverage of our code:

```
twitter_file_test.py .... [100%]

===== 4 passed in 1.16s =====
The command "pytest" exited with 0.
$ pytest --cov ./
===== test session starts =====
platform linux -- Python 3.7.1, pytest-5.3.5, py-1.7.0, pluggy-0.13.1
rootdir: /home/travis/build/umeshbodhwani/Twitter-Sentiment-Analysis-on-AWS
plugins: cov-2.8.1
collected 4 items
```

Name	Stmts	Miss	Cover
-----	-----	-----	-----
nltoolkit.py	56	24	57%
twitter_file.py	59	2	97%
twitter_file_test.py	29	0	100%
-----	-----	-----	-----
TOTAL	144	26	82%

We then ran ETL jobs on them to make JSON files.



The screenshot displays the AWS Glue console interface. At the top, there is a search bar and a filter option. Below this, a table lists several ETL jobs. The 'evaljob' is highlighted. Below the job list, there are tabs for 'History', 'Details', 'Script', and 'Metrics'. The 'History' tab is active, showing a table of job runs. The first row in this table shows a successful run with a status of 'Succeeded' and a delay of 2080 minutes.

Name	Type	ETL language	Script location	Last modified
devjob	Spark	python	s3://avis-glue-scripts...	9 March 2020 1:53 AM UTC-4
evaljob	Spark	python	s3://avis-glue-scripts...	9 March 2020 2:26 AM UTC-4
train	Spark	python	s3://avis-glue-scripts...	9 March 2020 1:07 AM UTC-4

Run ID	Retry attempt	Run status	Error	Logs	Error logs	Glue version	Maximum capacity	Triggered by	Start time	End time	Execution time	Timeout	Delay
j_A0P9eC0565345e6...	-	Succeeded		Logs		1.0	5		9 March ...	9 March ...	5 mins	2080 mins	

2. Tensorflow model using CNN

In order to do this step, we used the following code to -

- Load embedding into a dictionary and then making an embeddings matrix.
- Made a CNN model which will be trained on our data and saved it.
- The simple CNN model using Adam optimizer was taking around 2.5 hours for just 1 epoch on local system and around 1 hour on SageMaker.
- We did some research to reduce the computation time and found other optimizers such as SGD which runs in significantly lesser time.
- However, our accuracy was compromised while fitting the model on such large dataset
- Below is the change we did for SGD and our corresponding results.

The image below shows the CNN model with SGD optimizer.

```
model = Sequential()
model.add(Embedding(config['embeddings_dictionary_size'], config['embeddings_vector_size'], weights=[matrix_emb], m
model.add(Conv1D(filters=100, kernel_size=2, activation='relu', padding='valid', strides=1,))
model.add(GlobalMaxPool1D())
model.add(Dense(100, activation='relu'))
model.add(Dense(1, activation='sigmoid', name='score'))
model.compile(optimizer='sgd', loss='binary_crossentropy', metrics=['accuracy'])
cnn_model = model
return cnn_model
```

The Result of this code on running locally was as follows:

```
(base) dym-168-39-142-39:~$ model127 umeshbodhwani python sentiment_training.py --train 'data/train/' --dev 'data/dev/' --eval 'data/eval/' --num_epoch 48 --config 'training_config.json' --model_output_dir "."
Entering training file
Entering dataset file
Entering model own file
Preparing for training...
Fetching train data...
WARNING:tensorflow:From /Users/umeshbodhwani/anaconda3/lib/python3.7/site-packages/tensorflow/python/data/util/random_seed.py:18: add_dispatch_support.<locals>.wrapper (from tensorflow.python.ops.array_ops) :
itl be removed in a future version.
Instructions for updating:
Use tf.where in 2.6, which has the same broadcast rule as np.where
WARNING:tensorflow:From /Users/umeshbodhwani/Desktop/Canvas/AI_Cloud/twittorch/twittorch_compressed/model127/sentiment_dataset.py:85: DatasetV1.make_one_shot_iterator (from tensorflow.python.data.ops.dataset_ops
d will be removed in a future version.
Instructions for updating:
Use 'for ... in Dataset:' to iterate over a dataset. If using 'tf.estimator', return the 'Dataset' object directly from your input function. As a last resort, you can use 'tf.compat.v1.data.make_one_shot_iter
Fetching dev data...
WARNING:tensorflow:From /Users/umeshbodhwani/anaconda3/lib/python3.7/site-packages/tensorflow/python/keras/initializers.py:159: calling RandomUniform.__init__ (from tensorflow.python.ops.init_ops) with dtype
will be removed in a future version.
Instructions for updating:
Call initializer instance with the dtype argument instead of passing it to the constructor
2020-03-09 11:29:31.734423: I tensorflow/core/platform/cpu_feature_guard.cc:142] Your CPU supports instructions that this TensorFlow binary was not compiled to use: AVX2 FMA
WARNING:tensorflow:From /Users/umeshbodhwani/anaconda3/lib/python3.7/site-packages/tensorflow/python/ops/init_ops.py:1251: calling VarianceScaling.__init__ (from tensorflow.python.ops.init_ops) with dtype is
1 is removed in a future version.
Instructions for updating:
Call initializer instance with the dtype argument instead of passing it to the constructor
Starting training...
Epoch 1/48
13688/13688 [=====] - 181s 7ms/step - loss: 0.6879 - acc: 0.5448 - val_loss: 0.8839 - val_acc: 0.6586
Epoch 2/48
13688/13688 [=====] - 49s 4ms/step - loss: 0.6811 - acc: 0.5695 - val_loss: 0.6795 - val_acc: 0.6743
Epoch 3/48
13688/13688 [=====] - 48s 4ms/step - loss: 0.6775 - acc: 0.5797 - val_loss: 0.6762 - val_acc: 0.5889
Epoch 4/48
13688/13688 [=====] - 47s 3ms/step - loss: 0.6745 - acc: 0.5858 - val_loss: 0.6739 - val_acc: 0.5857
Epoch 5/48
13688/13688 [=====] - 48s 4ms/step - loss: 0.6724 - acc: 0.5889 - val_loss: 0.6719 - val_acc: 0.5885
Epoch 6/48
13688/13688 [=====] - 58s 4ms/step - loss: 0.6786 - acc: 0.5919 - val_loss: 0.6782 - val_acc: 0.5915
```

```

Epoch 7/40
13600/13600 [=====] - 49s 4ms/step - loss: 0.6698 - acc: 0.5937 - val_loss: 0.6688 - val_acc: 0.5936
Epoch 8/40
13600/13600 [=====] - 49s 4ms/step - loss: 0.6676 - acc: 0.5956 - val_loss: 0.6675 - val_acc: 0.5951
Epoch 9/40
13600/13600 [=====] - 50s 4ms/step - loss: 0.6664 - acc: 0.5975 - val_loss: 0.6664 - val_acc: 0.5968
Epoch 10/40
13600/13600 [=====] - 51s 4ms/step - loss: 0.6653 - acc: 0.5987 - val_loss: 0.6654 - val_acc: 0.5981
Epoch 11/40
13600/13600 [=====] - 50s 4ms/step - loss: 0.6644 - acc: 0.5995 - val_loss: 0.6645 - val_acc: 0.5988
Epoch 12/40
13600/13600 [=====] - 51s 4ms/step - loss: 0.6636 - acc: 0.6004 - val_loss: 0.6636 - val_acc: 0.6000
Epoch 13/40
13600/13600 [=====] - 52s 4ms/step - loss: 0.6628 - acc: 0.6011 - val_loss: 0.6628 - val_acc: 0.6004
Epoch 14/40
13600/13600 [=====] - 50s 4ms/step - loss: 0.6622 - acc: 0.6016 - val_loss: 0.6624 - val_acc: 0.6012
Epoch 15/40
13600/13600 [=====] - 51s 4ms/step - loss: 0.6616 - acc: 0.6020 - val_loss: 0.6618 - val_acc: 0.6020
Epoch 16/40
13600/13600 [=====] - 51s 4ms/step - loss: 0.6611 - acc: 0.6026 - val_loss: 0.6613 - val_acc: 0.6023
Epoch 17/40
13600/13600 [=====] - 52s 4ms/step - loss: 0.6606 - acc: 0.6031 - val_loss: 0.6609 - val_acc: 0.6023
Epoch 18/40
13600/13600 [=====] - 52s 4ms/step - loss: 0.6602 - acc: 0.6035 - val_loss: 0.6605 - val_acc: 0.6030
Epoch 19/40
13600/13600 [=====] - 150s 11ms/step - loss: 0.6598 - acc: 0.6040 - val_loss: 0.6601 - val_acc: 0.603
Epoch 20/40
13600/13600 [=====] - 48s 4ms/step - loss: 0.6595 - acc: 0.6043 - val_loss: 0.6597 - val_acc: 0.6041
Epoch 21/40
13600/13600 [=====] - 53s 4ms/step - loss: 0.6591 - acc: 0.6046 - val_loss: 0.6594 - val_acc: 0.6044
Epoch 22/40
13600/13600 [=====] - 54s 4ms/step - loss: 0.6588 - acc: 0.6048 - val_loss: 0.6591 - val_acc: 0.6048
Epoch 23/40
13600/13600 [=====] - 54s 4ms/step - loss: 0.6585 - acc: 0.6052 - val_loss: 0.6588 - val_acc: 0.6052
Epoch 24/40
13600/13600 [=====] - 55s 4ms/step - loss: 0.6582 - acc: 0.6056 - val_loss: 0.6585 - val_acc: 0.6055
Epoch 25/40
13600/13600 [=====] - 55s 4ms/step - loss: 0.6580 - acc: 0.6058 - val_loss: 0.6582 - val_acc: 0.6055
Epoch 26/40
13600/13600 [=====] - 56s 4ms/step - loss: 0.6577 - acc: 0.6061 - val_loss: 0.6579 - val_acc: 0.6058
Epoch 27/40
13600/13600 [=====] - 55s 4ms/step - loss: 0.6575 - acc: 0.6063 - val_loss: 0.6577 - val_acc: 0.6061
Epoch 28/40
13600/13600 [=====] - 55s 4ms/step - loss: 0.6572 - acc: 0.6066 - val_loss: 0.6575 - val_acc: 0.6061
Epoch 29/40
13600/13600 [=====] - 55s 4ms/step - loss: 0.6570 - acc: 0.6068 - val_loss: 0.6573 - val_acc: 0.6062
Epoch 30/40
13600/13600 [=====] - 57s 4ms/step - loss: 0.6568 - acc: 0.6072 - val_loss: 0.6570 - val_acc: 0.6065
Epoch 31/40
13600/13600 [=====] - 53s 4ms/step - loss: 0.6566 - acc: 0.6074 - val_loss: 0.6568 - val_acc: 0.6067
Epoch 32/40
13600/13600 [=====] - 270s 20ms/step - loss: 0.6564 - acc: 0.6076 - val_loss: 0.6566 - val_acc: 0.6069
Epoch 33/40
13600/13600 [=====] - 49s 4ms/step - loss: 0.6562 - acc: 0.6079 - val_loss: 0.6564 - val_acc: 0.6078
Epoch 34/40
13600/13600 [=====] - 50s 4ms/step - loss: 0.6560 - acc: 0.6083 - val_loss: 0.6562 - val_acc: 0.6080
Epoch 35/40
13600/13600 [=====] - 55s 4ms/step - loss: 0.6558 - acc: 0.6085 - val_loss: 0.6561 - val_acc: 0.6083
Epoch 36/40
13600/13600 [=====] - 56s 4ms/step - loss: 0.6556 - acc: 0.6089 - val_loss: 0.6558 - val_acc: 0.6086
Epoch 37/40
13600/13600 [=====] - 57s 4ms/step - loss: 0.6554 - acc: 0.6089 - val_loss: 0.6557 - val_acc: 0.6087
Epoch 38/40
13600/13600 [=====] - 55s 4ms/step - loss: 0.6552 - acc: 0.6092 - val_loss: 0.6555 - val_acc: 0.6091
Epoch 39/40
13600/13600 [=====] - 60s 4ms/step - loss: 0.6551 - acc: 0.6094 - val_loss: 0.6553 - val_acc: 0.6093
Epoch 40/40
13598/13600 [=====] - ETA: 0s - loss: 0.6549 - acc: 0.6097WARNING:tensorflow:Your dataset iterator ran out
of steps * epochs' batches (in this case, 1601 batches). You may need to use the repeat() function when building your dataset.
13600/13600 [=====] - 52s 4ms/step - loss: 0.6549 - acc: 0.6097 - val_loss: 0.6388 - val_acc: 0.6094
Test loss:0.6535608787089586
Test accuracy:0.6125375032424927
Saving model...
2020-03-09 12:12:00.069063: W tensorflow/python/util/util.cc:280] Sets are not currently considered sequences, but this may change in
the future. To avoid a future warning, set the verbosity to 10 (on Linux, 'export AUTOGRAPH_VERBOSITY=10') and attach the full output. Cause: converting
3cd08b: AttributeError: module 'gast' has no attribute 'Num'
WARNING:tensorflow:Entity <function canonicalize_signatures.<locals>.signature_wrapper at 0x62d93cd88> could not be transformed and
e verbosity to 10 (on Linux, 'export AUTOGRAPH_VERBOSITY=10') and attach the full output. Cause: converting <function canonicalize_s
Name: 3, expecting 4
WARNING:tensorflow:Entity <function Function._initialize_uninitialized_variables.<locals>.initialize_variables at 0x647378c80> could
n filing the bug, set the verbosity to 10 (on Linux, 'export AUTOGRAPH_VERBOSITY=10') and attach the full output. Cause: converting
78c80: AttributeError: module 'gast' has no attribute 'Num'
Model successfully saved at: ./sentiment_model.h5

```

We got a Test accuracy of 61.2% with this SGD Optimization in our CNN Model.

We also tried to increase the number of layers and changed the optimizer to Adadelta this time.

Below is our modified code and the corresponding results.

```

model = Sequential()
model.add(Embedding(config['embeddings_dictionary_size'], config['embeddings_vector_size'], weights=[matrix_emb], name='embedding', input_length=
model.add(Conv1D(filters=100, kernel_size=20, activation='relu', padding='valid', strides=1,))
# model.add(GlobalMaxPool1D)
model.add(Conv1D(filters=50, kernel_size=3, activation='relu', padding='valid', strides=1,))
model.add(Conv1D(filters=20, kernel_size=2, activation='relu', padding='valid', strides=1,))
model.add(Flatten())
model.add(Dense(100, activation='relu'))
model.add(Dense(1, activation='sigmoid', name='score'))
model.compile(optimizer='adadelta', loss='binary_crossentropy', metrics=['accuracy'])
cnn_model = model
return cnn_model

```

```

(Users/umeshbodhwani@anapanda3:~/code127) umeshbodhwani$
(Users/umeshbodhwani@anapanda3:~/code127) umeshbodhwani$ python sentiment_training.py --train "data/train/" --dev "data/dev/" --eval "data/eval/" --run_epoch 40 --config "training_config.json" --model_output_dir "."
Entering training file
Entering dataset file
Entering model cnn file
Preparing for training...
Fetching train data...
WARNING:tensorflow:From /Users/umeshbodhwani/anaconda3/lib/python3.7/site-packages/tensorflow/python/data/ops/random_shuffle_v2.py:58: adw.DispatchSupport.<locals>.wrapper (from tensorflow.python.ops.array_ops) is deprecated and will be removed in a future version.
Instructions for updating:
Use tf.where in 2.8, which has the same broadcast rule as np.where
WARNING:tensorflow:From /Users/umeshbodhwani/Desktop/Canvas/AI_Cloud/twitters/twitters_compressed/model127/sentiment_dataset.py:80: DatasetV1.make_one_shot_iterator (from tensorflow.python.data.ops.dataset_ops) is deprecated and will be removed in a future version.
Instructions for updating:
Use 'for ... in dataset:' to iterate over a dataset. If using 'tf.estimator', return the 'Dataset' object directly from your input function. As a last resort, you can use 'tf.compat.v1.data.make_one_shot_iterator' to fetch data.
Fetching dev data...
WARNING:tensorflow:From /Users/umeshbodhwani/anaconda3/lib/python3.7/site-packages/tensorflow/python/ops/init_ops.py:129: calling RandomUniform.__init__ (from tensorflow.python.ops.init_ops) with dtype is deprecated and will be removed in a future version.
Instructions for updating:
Call initializer instance with the dtype argument instead of passing it to the constructor
2020-03-04 12:34:43.972077: I tensorflow/core/platform/cpu_feature_guard.cc:142] Your CPU supports instructions that this TensorFlow binary was not compiled to use: AVX2 FMA
WARNING:tensorflow:From /Users/umeshbodhwani/anaconda3/lib/python3.7/site-packages/tensorflow/python/ops/init_ops.py:1261: calling VarianceScaling.__init__ (from tensorflow.python.ops.init_ops) with dtype is deprecated and will be removed in a future version.
Instructions for updating:
Call initializer instance with the dtype argument instead of passing it to the constructor
Starting training...
Epoch 1/40
13600/13600 [=====] - 236s 10ms/step - loss: 0.6916 - acc: 0.5274 - val_loss: 0.6897 - val_acc: 0.5546
Epoch 2/40
13600/13600 [=====] - 167s 13ms/step - loss: 0.6885 - acc: 0.5565 - val_loss: 0.6871 - val_acc: 0.5596
Epoch 3/40
13600/13600 [=====] - 171s 13ms/step - loss: 0.6858 - acc: 0.5602 - val_loss: 0.6846 - val_acc: 0.5622
Epoch 4/40
13600/13600 [=====] - 172s 13ms/step - loss: 0.6836 - acc: 0.5664 - val_loss: 0.6824 - val_acc: 0.5701
Epoch 5/40
13600/13600 [=====] - 173s 13ms/step - loss: 0.6812 - acc: 0.5715 - val_loss: 0.6801 - val_acc: 0.5736
Epoch 6/40
13600/13600 [=====] - 173s 13ms/step - loss: 0.6798 - acc: 0.5753 - val_loss: 0.6779 - val_acc: 0.5774
Epoch 7/40
13600/13600 [=====] - 171s 13ms/step - loss: 0.6767 - acc: 0.5802 - val_loss: 0.6756 - val_acc: 0.5841
Epoch 8/40
13600/13600 [=====] - 172s 13ms/step - loss: 0.6745 - acc: 0.5861 - val_loss: 0.6734 - val_acc: 0.5875
Epoch 9/40
13600/13600 [=====] - 173s 13ms/step - loss: 0.6723 - acc: 0.5905 - val_loss: 0.6714 - val_acc: 0.5929
Epoch 10/40
13600/13600 [=====] - 172s 13ms/step - loss: 0.6702 - acc: 0.5936 - val_loss: 0.6695 - val_acc: 0.5982
Epoch 11/40
13600/13600 [=====] - 175s 13ms/step - loss: 0.6684 - acc: 0.5963 - val_loss: 0.6679 - val_acc: 0.5979

```

```

Epoch 12/40
13600/13600 [=====] - 176s 13ms/step - loss: 0.6667 - acc: 0.6985 - val_loss: 0.6663 - val_acc: 0.6990
Epoch 13/40
13600/13600 [=====] - 176s 13ms/step - loss: 0.6653 - acc: 0.6984 - val_loss: 0.6650 - val_acc: 0.6986
Epoch 14/40
13600/13600 [=====] - 173s 13ms/step - loss: 0.6648 - acc: 0.6919 - val_loss: 0.6639 - val_acc: 0.6924
Epoch 15/40
13600/13600 [=====] - 174s 13ms/step - loss: 0.6629 - acc: 0.6833 - val_loss: 0.6630 - val_acc: 0.6937
Epoch 16/40
13600/13600 [=====] - 182s 13ms/step - loss: 0.6620 - acc: 0.6842 - val_loss: 0.6621 - val_acc: 0.6942
Epoch 17/40
13600/13600 [=====] - 181s 13ms/step - loss: 0.6612 - acc: 0.6850 - val_loss: 0.6614 - val_acc: 0.6956
Epoch 18/40
13600/13600 [=====] - 176s 13ms/step - loss: 0.6605 - acc: 0.6857 - val_loss: 0.6600 - val_acc: 0.6958
Epoch 19/40
13600/13600 [=====] - 177s 13ms/step - loss: 0.6599 - acc: 0.6862 - val_loss: 0.6602 - val_acc: 0.6968
Epoch 20/40
13600/13600 [=====] - 176s 13ms/step - loss: 0.6594 - acc: 0.6867 - val_loss: 0.6598 - val_acc: 0.6971
Epoch 21/40
13600/13600 [=====] - 194s 14ms/step - loss: 0.6590 - acc: 0.6872 - val_loss: 0.6594 - val_acc: 0.6978
Epoch 22/40
13600/13600 [=====] - 181s 13ms/step - loss: 0.6586 - acc: 0.6876 - val_loss: 0.6590 - val_acc: 0.6980
Epoch 23/40
13600/13600 [=====] - 175s 13ms/step - loss: 0.6582 - acc: 0.6879 - val_loss: 0.6586 - val_acc: 0.6983
Epoch 24/40
13600/13600 [=====] - 183s 13ms/step - loss: 0.6579 - acc: 0.6884 - val_loss: 0.6583 - val_acc: 0.6986
Epoch 25/40
13600/13600 [=====] - 177s 13ms/step - loss: 0.6576 - acc: 0.6886 - val_loss: 0.6580 - val_acc: 0.6991
Epoch 26/40
13600/13600 [=====] - 178s 13ms/step - loss: 0.6573 - acc: 0.6889 - val_loss: 0.6577 - val_acc: 0.6992
Epoch 27/40
13600/13600 [=====] - 176s 13ms/step - loss: 0.6570 - acc: 0.6891 - val_loss: 0.6574 - val_acc: 0.6998
Epoch 28/40
13600/13600 [=====] - 176s 13ms/step - loss: 0.6568 - acc: 0.6894 - val_loss: 0.6572 - val_acc: 0.6999
Epoch 29/40
13600/13600 [=====] - 180s 13ms/step - loss: 0.6565 - acc: 0.6896 - val_loss: 0.6569 - val_acc: 0.6104
Epoch 30/40
13600/13600 [=====] - 192s 14ms/step - loss: 0.6563 - acc: 0.6899 - val_loss: 0.6567 - val_acc: 0.6108
Epoch 31/40
13600/13600 [=====] - 190s 14ms/step - loss: 0.6560 - acc: 0.6101 - val_loss: 0.6565 - val_acc: 0.6103
Epoch 32/40
13600/13600 [=====] - 181s 13ms/step - loss: 0.6558 - acc: 0.6103 - val_loss: 0.6562 - val_acc: 0.6109
Epoch 33/40
13600/13600 [=====] - 186s 14ms/step - loss: 0.6556 - acc: 0.6105 - val_loss: 0.6560 - val_acc: 0.6112
Epoch 34/40
13600/13600 [=====] - 198s 15ms/step - loss: 0.6554 - acc: 0.6107 - val_loss: 0.6558 - val_acc: 0.6113
Epoch 35/40
13600/13600 [=====] - 207s 15ms/step - loss: 0.6552 - acc: 0.6110 - val_loss: 0.6556 - val_acc: 0.6113
Epoch 36/40
13600/13600 [=====] - 247s 18ms/step - loss: 0.6550 - acc: 0.6110 - val_loss: 0.6553 - val_acc: 0.6116
Epoch 37/40
13600/13600 [=====] - 236s 17ms/step - loss: 0.6548 - acc: 0.6112 - val_loss: 0.6552 - val_acc: 0.6115
Epoch 38/40
13600/13600 [=====] - 174s 13ms/step - loss: 0.6546 - acc: 0.6114 - val_loss: 0.6549 - val_acc: 0.6119
Epoch 39/40
13600/13600 [=====] - 172s 13ms/step - loss: 0.6544 - acc: 0.6115 - val_loss: 0.6547 - val_acc: 0.6121
Epoch 40/40
13597/13600 [=====] - ETA: 0s - loss: 0.6542 - acc: 0.6118WARNING:tensorflow:Your dataset iterator ran out of data;
_steps * epochs" batches (in this case, 1601 batches). You may need to use the repeat() function when building your dataset.
13600/13600 [=====] - 171s 13ms/step - loss: 0.6542 - acc: 0.6118 - val_loss: 0.6382 - val_acc: 0.6121
Test loss:0.6533593118190766
Test accuracy:0.6151624917984809
Saving model...
2020-03-09 15:37:33.748083: W tensorflow/python/util/util.cc:280] Sets are not currently considered sequences, but this may change in the future.
WARNING:tensorflow:Entity <function Function._initialize_uninitialized_variables.<locals>.initialize_variables at 0x62f861a60> could not be transformed
n filing the bug, set the verbosity to 10 (on Linux, 'export AUTOGRAPH_VERBOSITY=10') and attach the full output. Cause: converting <function
61a60>: AttributeError: module 'gast' has no attribute 'Num'
WARNING:tensorflow:Entity <function canonicalize_signatures.<locals>.signature_wrapper at 0x62f861a60> could not be transformed and will be executed
e verbosity to 10 (on Linux, 'export AUTOGRAPH_VERBOSITY=10') and attach the full output. Cause: converting <function canonicalize_signatures.
Name: 3, expecting 4
WARNING:tensorflow:Entity <function Function._initialize_uninitialized_variables.<locals>.initialize_variables at 0x644ea6378> could not be transformed
n filing the bug, set the verbosity to 10 (on Linux, 'export AUTOGRAPH_VERBOSITY=10') and attach the full output. Cause: converting <function
a6378>: AttributeError: module 'gast' has no attribute 'Num'
Model successfully saved at: ./sentiment_model.h5
(base) dyn-160-39-142-39:~$

```

Test accuracy at local system using Adadelta optimizer: 61.5%

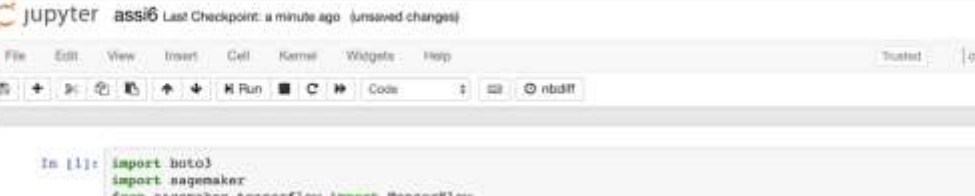
While Adadelta took more time than SGD, the difference in their results was not significant.

Our best model:

The best accuracy was obtained using the Adam optimizer with more hidden layers, but it took quite a lot time. Hence we decided to run the model on SageMaker with just 4 epochs.

- 1) We were able to run our model successfully on SageMaker.
- 2) We faced a lot of issues in running SageMaker with the Educate account. It runs when we upload all our files into SageMaker Jupyter, but even that fails given the huge amount of data for this assignment.
- 3) We did some research on this and found that the only way to run SageMaker on such huge data was that we save our data in S3 bucket and let SageMaker pick our data from there. For this, we used our personal account on AWS.

While doing this through the Educate account, we were not able to get access to the required instances and hence we used a Private account in order to deploy our model on AWS SageMaker. With this we were able to successfully run our model on SageMaker.



The screenshot shows a Jupyter Notebook titled "ass6" with a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help) and a toolbar. The notebook contains three code cells:

```
In [1]: import boto3
import sagemaker
from sagemaker.tensorflow import TensorFlow
from sagemaker.tuner import IntegerParameter
from sagemaker.tuner import CategoricalParameter
from sagemaker.tuner import ContinuousParameter
from sagemaker.tuner import HyperparameterTuner

In [2]: role = sagemaker.get_execution_role()
role

Out[2]: 'arn:aws:iam::371556535825:role/service-role/AmazonSageMaker-ExecutionRole-20200307T212761'
```

```
In [3]: estimator = TensorFlow(entry_point='model127/sentiment_training.py', \
                               source_dir='s3://twittera6/parent_folder/model127.tar.gz', \
                               role=role, \
                               framework_version='1.14.0', py_version='py3', \
                               hyperparameters={'num_epoch': 4, 'config_file': 'training_config.json'}, \
                               train_instance_count=1, train_instance_type='ml.m4.xlarge')
```

[illegible]

We used the Adam optimizer for training our model on SageMaker. We noticed that each epoch took around 1 hour to run. Hence we could only train our sagemaker model on 4 epochs and got an accuracy of around 64%.

Had we been given more computation power, we would have run our CNN model on a larger number of epochs and achieved a better accuracy

Sagemaker Inference

In this step, we then deployed our model as a SageMaker inference endpoint. While doing this, in the endpoint configuration we used the cheapest instance “ml.t2.medium”. The following screenshots show the process of creating an endpoint.

Resource Groups ▾

Success! You created a model.
To use this model, choose [Create endpoint](#).

Amazon SageMaker ▾ Models

Models

Create endpoint Create endpoint configuration Actions ▾ Create model

Search models

Name	ARN	Creation time
sentiment-model	arn:aws:sagemaker:us-east-1:299791166592:model/sentiment-model	Mar 08, 2020 01:13 UTC

Create endpoint configuration

To deploy models to Amazon SageMaker, first create an endpoint configuration. In the configuration, specify which models to deploy, and the relative traffic weighting and hardware requirements for each. See [Deploying a Model on Amazon SageMaker Hosting Services](#) or [Learn more about the API](#).

New endpoint configuration

Endpoint configuration name
sentiment-model
Maximum of 63 alphanumeric characters. Can include hyphens (-), but not spaces. Must be unique within your account in an AWS Region.

Encryption key - optional
Encrypt your data. Choose an existing KMS key or enter a key's ARN.
No Custom Encryption ▾

☒ Enable data capture
By enabling this feature, Amazon SageMaker can save prediction request and prediction response information from your endpoint to a specified location. [Learn more](#)

Production variants

Model name	Training job	Variant name	Instance type	Elastic Inference	Initial instance count	Initial weight	Actions
sentiment-model	-	default-variant-name	ml.t2.medium	none	1	1	Edit Remove

[Add model](#)

Tags - optional

Key Value Remove

[Add tag](#)

Cancel [Create endpoint configuration](#)

Endpoint name
Your application uses this name to access this endpoint.

Maximum of 63 alphanumeric characters. Can include hyphens (-), but not spaces. Must be unique within your account in an AWS Region.

Attach endpoint configuration

☒ Use an existing endpoint configuration. Use an existing endpoint configuration or clone an endpoint configuration.

☐ Create a new endpoint configuration. Add models and configure the instance and initial weight for each model.

New endpoint configuration Change Clone

Endpoint configuration name: Encryption key:

Production variants

Model name	Training job	Variant name	Instance type	Elastic Inference	Initial instance count
sentiment-model	-	default-variant-name	m1t2.medium	-	1

▼ **Tags - optional**

Key: Value: Remove

[Add tag](#)

Cancel Create endpoint

Resource Groups ▼ ★ vocstartsoft/user622654-base... N. Virginia Support

Success! You created an endpoint.
To track the status of the endpoint, view details. View details

Amazon SageMaker > Endpoints

Endpoints Refresh Update endpoint Actions Create endpoint

Name	ARN	Creation time	Status	Last updated
sentiment-model	arn:aws:sagemaker:us-east-1:299791166592:endpoint/sentiment-model	Mar 08, 2020 01:21 UTC	Creating	Mar 08, 2020 01:21 UTC

4. Model Deployment

As part of this step, we have made a Lambda Function that does the following:

- Pre-process with our code for the pipeline
- Model inference with the SageMaker endpoint
- Post processing

The lambda function takes JSON input with a “tweet” key and produces a JSON output with a “sentiment” key and a value that can either be “positive” or “negative” based on the model prediction.

In order to test this part, we are giving a sample tweet and then checking if its been identified as positive or negative.

```
lambda_function.py
from twitter_file import TwitterClass
tw = TwitterClass()

sage_maker_client = boto3.client("runtime.sagemaker")

def lambda_handler(event, context):
    now = datetime.datetime.now()
    Date_and_Time = str(now)

    initial_time = time.time_ns()

    tweet = event["tweet"]
    features = tw.processed(tweet)

    final_time = time.time_ns()
    Preprocessing_time = final_time - initial_time
    print('Preprocessing time: {}'.format(Preprocessing_time))

    model_payload = {
        'embedding_input': features,
    }

    model_initial_time = time.clock()

    model_response = sage_maker_client.invoke_endpoint(
        EndpointName="sentiment-model",
        ContentType="application/json",
        Body=json.dumps(model_payload))

    model_final_time = time.clock()
    model_inf_time = model_final_time - model_initial_time
    print('Model inference time: {}'.format(model_inf_time))

    result = json.loads(model_response["Body"].read().decode())

    response = {}

    response["tweet"] = tweet

    if result["predictions"][0][0] >= 0.5:
        response["sentiment"] = "positive"
    else:
        response["sentiment"] = "negative"

    response['Date_and_Time'] = Date_and_Time
    response['Preprocessing_time in nanoseconds'] = Preprocessing_time
    response['model_inf_time in seconds'] = model_inf_time
    response['probability'] = result["predictions"][0][0]

    print("Result: " + json.dumps(response, indent=2))
    # TODO implement
    return response
```

Payload Logging using Lambda Function

Here we modified the Lambda Function (above) created to calculate and display the following:

- Date and time of the request
- Tweet
- Sentiment
- Probability from the model
- Pre-processing time
- Model inference time

Then the lambda function logs a JSON object to the bucket in S3 with the above details. The image below shows our result on AWS.

The screenshot displays the AWS Lambda console for a function named 'lambda_function'. The 'Execution result: succeeded (logs)' section is expanded, showing a JSON object returned by the function. Below this, a 'Summary' section provides key metrics, and a 'Log output' section shows the raw log data from CloudWatch.

Execution result: succeeded (logs)

Details

The area below shows the result returned by your function execution. [Learn more](#) about returning results from your function.

```
{
  "tweet": "This cookie is awesome!",
  "sentiment": "positive",
  "Date_and_Time": "2020-03-09 00:49:45.011084",
  "Preprocessing_time in nanoseconds": 683182,
  "model_inf_time in seconds": 0.010000000000000009,
  "probability": 0.649373412
}
```

Summary

Code SHA-256	Request ID
OXU0OVGlq9YD+UF0tGx+2MW0j3MCSMamkR3hLWkNQY=	39573bba-5b30-4808-a473-248bbfda9329
Duration	Billed duration
84.80 ms	100 ms
Resources configured	Max memory used
1024 MB	313 MB

Log output

The section below shows the logging calls in your code. These correspond to a single row within the CloudWatch log group corresponding to this Lambda function. [Click here](#) to view the CloudWatch log group.

```
START RequestId: 39573bba-5b30-4808-a473-248bbfda9329 Version: $LATEST
Preprocessing time: 683182
Model inference time: 0.010000000000000009
Result: {
  "tweet": "This cookie is awesome!",
  "sentiment": "positive",
  "Date_and_Time": "2020-03-09 00:49:45.011084",
  "Preprocessing_time in nanoseconds": 683182,
  "model_inf_time in seconds": 0.010000000000000009,
  "probability": 0.649373412
}
```

REST API

Here in this step we created an API Gateway to expose our lambda function.

The gateway implemented a “/predict” resource with a “POST” request method.

We then deployed it under a “v1” stage. The following set of images shows how we went about this process:

The screenshot shows the AWS API Gateway console with the 'Create new API' page. The 'Choose the protocol' section has 'REST' selected. The 'Create new API' section has 'New API' selected. The 'Settings' section shows the API name as 'sentiment-model-api-final', the endpoint type as 'Regional', and a 'Create API' button at the bottom right.

aws Services Resource Groups vocstartsoft/user622654-base... N, Virginia Support

Amazon API Gateway APIs > Create Show all hints ?

Choose the protocol

Select whether you would like to create a REST API or a WebSocket API.

☒ REST ☐ WebSocket

Create new API

In Amazon API Gateway, a REST API refers to a collection of resources and methods that can be invoked through HTTPS endpoints.

☒ New API ☐ Import from Swagger or Open API 3 ☐ Example API

Settings

Choose a friendly name and description for your API.

API name* sentiment-model-api-final

Description

Endpoint Type Regional ⓘ ⓘ

* Required

Create API

The screenshot shows the AWS API Gateway console with the 'New Child Resource' page. The 'Resources' tab is selected, and a new resource named 'predict' is being created with the path '/predict'. The 'Enable API Gateway CORS' checkbox is checked. The 'Create Resource' button is at the bottom right.

aws Services Resource Groups vocstartsoft/user622654-base... N, Virginia Support

Amazon API Gateway APIs > sentiment-model-api-final (g2oyntbu91) > Resources > /@hvkxk7g8 > Create Show all hints ?

New Child Resource

Use this page to create a new child resource for your resource.

Configure as [proxy resource](#) ⓘ ⓘ

Resource Name* predict

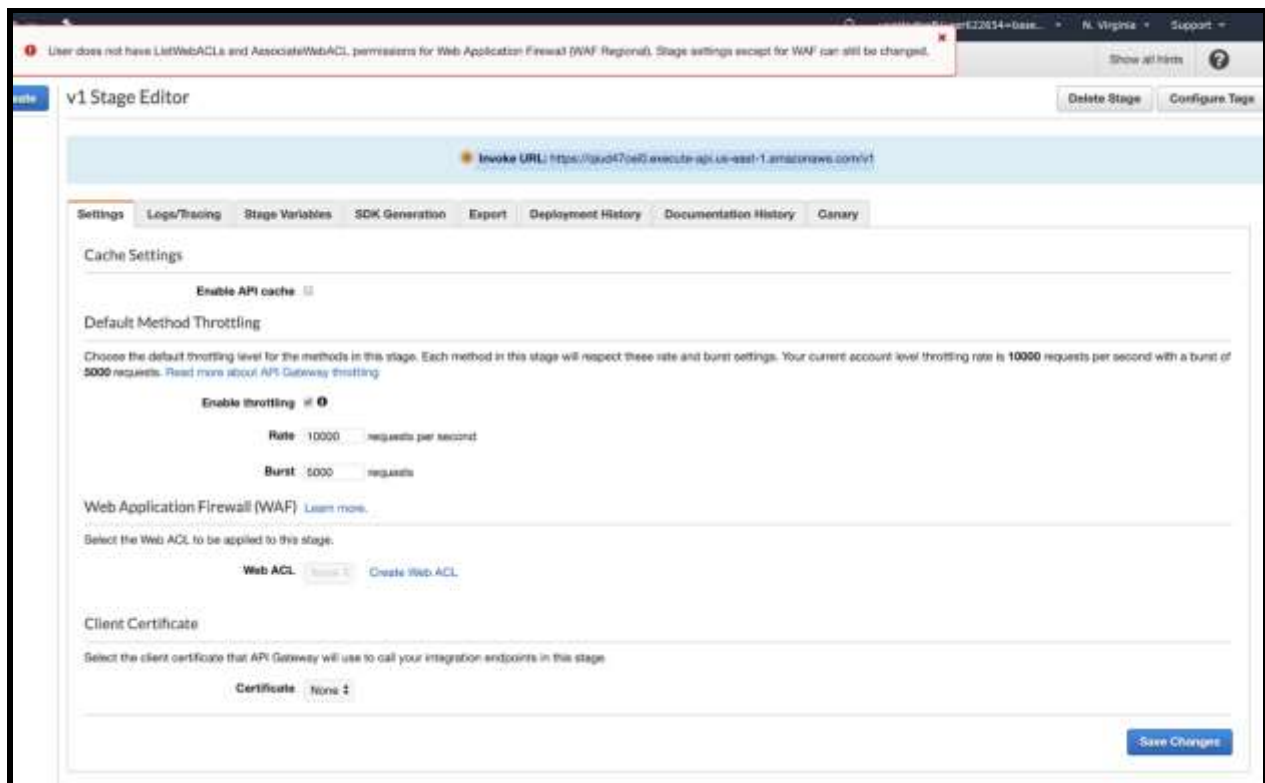
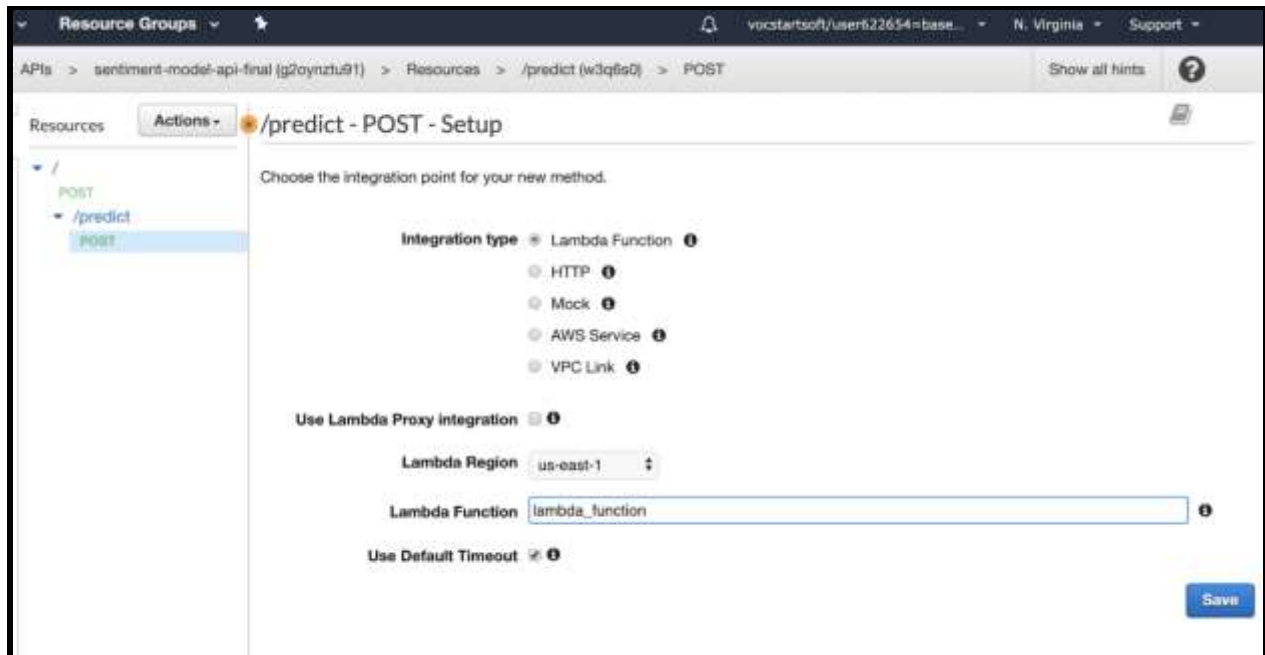
Resource Path* /predict

You can add path parameters using brackets. For example, the resource path **(username)** represents a path parameter called 'username'. Configuring **/[proxy+]** as a proxy resource catches all requests to its sub-resources. For example, it works for a GET request to /foo. To handle requests to /, add a new ANY method on the / resource.

Enable API Gateway CORS ⓘ ⓘ

* Required

Cancel Create Resource



Invoke URL: <https://qiud47cei0.execute-api.us-east-1.amazonaws.com/v1>

```
[base] dyn-168-36-143-151:~ umeshbodhwani$  
[base] dyn-168-36-143-151:~ umeshbodhwani$  
[base] dyn-168-36-143-151:~ umeshbodhwani$  
[base] dyn-168-36-143-151:~ umeshbodhwani$ curl -X POST https://qiud47cei0.execute-api.us-east-1.amazonaws.com/v1/predict --header "Content-Type:application/json" --data '{"tweet": "This cookie is awesome!"}'  
{"tweet": "This cookie is awesome!", "sentiment": "positive", "Date_and_Time": "2020-03-08 21:43:17.870634", "Preprocessing_time in nanoseconds": 458778, "model_inf_time in seconds": 0.0, "probability": 0.84726436}[base]  
[base] dyn-168-36-143-151:~ umeshbodhwani$  
[base] dyn-168-36-143-151:~ umeshbodhwani$  
[base] dyn-168-36-143-151:~ umeshbodhwani$  
[base] dyn-168-36-143-151:~ umeshbodhwani$  
[base] dyn-168-36-143-151:~ umeshbodhwani$  
[base] dyn-168-36-143-151:~ umeshbodhwani$
```

curl -X POST https://qiud47cei0.execute-api.us-east-1.amazonaws.com/v1/predict --header "Content-Type:application/json" --data '{"tweet": "This cookie is awesome!"}'

The screenshot shows the GitHub repository page for 'umeshbodhwani / Twitter-Sentiment-Analysis-on-AWS'. The repository has 14 commits, 1 branch, 0 packages, 0 releases, and 2 contributors. The latest commit is 'eac93a0' from 18 minutes ago, titled 'umeshbodhwani add SageMaker Notebook file'. The repository contains several files and folders, including 'CNN-model', 'Glue SPARK code', 'SageMaker', 'Twitter-Sentiment-Analysis-on-AWS', '_pycache_', 'Preprocessing.zip', 'README.md', and 'english'. The repository is currently on the 'master' branch.

Preprocessing.zip	Initial commit	5 days ago
README.md	Create README.md	3 days ago
english	Initial commit	5 days ago
lambda_function.py	add CNN model and lambda function	20 minutes ago
nltktoolkit.py	Initial commit	5 days ago
travis.yml	update travis	5 days ago
twitter_file.py	add zip file path	5 days ago
twitter_file_test.py	Initial commit	5 days ago
word_list.txt	Initial commit	5 days ago

README.md	
-----------	--

Twitter-Sentiment-Analysis-on-AWS

Links:

Embedding dict: https://twittera6.s3.amazonaws.com/parent_folder/embedding_list/glove.txt

Train: https://twittera6.s3.amazonaws.com/parent_folder/train/train.json

Dev: https://twittera6.s3.amazonaws.com/parent_folder/dev/dev.json

Eval: https://twittera6.s3.amazonaws.com/parent_folder/eval/eval.json

Model: https://twittera6.s3.amazonaws.com/parent_folder/model127.tar.gz

Github URL: <https://github.com/umeshbodhwani/Twitter-Sentiment-Analysis-on-AWS>

REST API Deployment link:

```
curl -X POST https://qiud47cei0.execute-api.us-east-1.amazonaws.com/v1/predict --header "Content-Type:application/json" --data '{"tweet": "This cookie is awesome!"}'
```