Text Classification using TensorFlow/Keras on AI Platform

This notebook illustrates:

1. Creating datasets for AI Platform using BigQuery
2. Creating a text classification model using the Estimator API with a Keras model
3. Training on Cloud AI Platform
4. Rerun with pre-trained embedding

In [ ]:

```
!sudo chown -R jupyter:jupyter /home/jupyter/training-data-analyst
```

In [ ]:

```
!pip install --user google-cloud-bigquery==1.25.0
```

Note: Restart your kernel to use updated packages.

Kindly ignore the deprecation warnings and incompatibility errors related to google-cloud-storage.

In [16]:

```
# change these to try this notebook out
BUCKET = 'qwiklabs-gcp-01-989eaccf099b'
PROJECT = 'qwiklabs-gcp-01-989eaccf099b'
REGION = 'us-central1'
```

In [17]:

```
import os
os.environ['BUCKET'] = BUCKET
os.environ['PROJECT'] = PROJECT
os.environ['REGION'] = REGION
os.environ['TFVERSION'] = '2.6'

if 'COLAB_GPU' in os.environ:  # this is always set on Colab, the value is 0 or
  from google.colab import auth
  auth.authenticate_user()
  # download "sidecar files" since on Colab, this notebook will be on Drive
  !rm -rf txtclsmodel
  !git clone --depth 1 https://github.com/GoogleCloudPlatform/training-data-anal
  !mv  training-data-analyst/courses/machine_learning/deepdive/09_sequence/txtcl
  !rm -rf training-data-analyst
  # downgrade TensorFlow to the version this notebook has been tested with
  #!pip install --upgrade tensorflow==$TFVERSION
```

In [18]:

```
import tensorflow as tf
print(tf.__version__)
```

```
2.6.0
```

We will look at the titles of articles and figure out whether the article came from the New York Times, TechCrunch or GitHub.

We will use hacker news as our data source. It is an aggregator that displays tech related headlines from various sources.

Creating Dataset from BigQuery

Hacker news headlines are available as a BigQuery public dataset. The dataset contains all headlines from the sites inception in October 2006 until October 2015.

Here is a sample of the dataset:

In [19]:

```
%load_ext google.cloud.bigquery
```

```
The google.cloud.bigquery extension is already loaded. To reload it, use:
  %reload_ext google.cloud.bigquery
```

In [20]:

```
%%bigquery --project $PROJECT
SELECT
  url, title, score
FROM
  `bigquery-public-data.hacker_news.stories`
WHERE
  LENGTH(title) > 10
  AND score > 10
  AND LENGTH(url) > 0
LIMIT 10
```

Out[20]:

	url	title	score
0	http://www.dumpert.nl/mediabase/6560049/3eb18e...	Calling the NSA: "I accidentally deleted an e-...	258
1	http://blog.liip.ch/archive/2013/10/28/hhvm-an...	Amazing performance with HHVM and PHP with a S...	11
2	http://www.gamedev.net/page/resources/_/techni...	A Journey Through the CPU Pipeline	11
3	http://jfarcand.wordpress.com/2011/02/25/atmos...	Atmosphere Framework 0.7 released: GWT, Wicket...	11
4	http://tech.gilt.com/post/90578399884/immutabl...	Immutable Infrastructure with Docker and EC2 [...	11
5	http://thechangelog.com/post/501053444/episode...	Changelog 0.2.0 - node.js w/Felix Geisendorfer	11
6	http://openangelforum.com/2010/09/09/second-bo...	Second Open Angel Forum in Boston Oct 13th--fr...	11
7	http://bredele.github.io/async	A collection of JavaScript asynchronous patterns	11
8	http://www.smashingmagazine.com/2007/08/25/20-...	20 Free and Fresh Icon Sets	11
9	http://www.cio.com/article/147801/Study_Finds_...	Study: Only 1 in 5 Workers is "Engaged" in The...	11

Let's do some regular expression parsing in BigQuery to get the source of the newspaper article from the URL. For example, if the url is http://mobile.nytimes.com/...., I want to be left with *nytimes*

In [21]:

```
%%bigquery --project $PROJECT
SELECT
  ARRAY_REVERSE(SPLIT(REGEXP_EXTRACT(url, '.*://(.[^/]+)/'), '.'))[OFFSET(1)] AS
  COUNT(title) AS num_articles
FROM
  `bigquery-public-data.hacker_news.stories`
WHERE
  REGEXP_CONTAINS(REGEXP_EXTRACT(url, '.*://(.[^/]+)/'), '.com$')
  AND LENGTH(title) > 10
GROUP BY
  source
ORDER BY num_articles DESC
LIMIT 10
```

Out[21]:

	source	num_articles
0	blogspot	41386
1	github	36525
2	techcrunch	30891
3	youtube	30848
4	nytimes	28787
5	medium	18422
6	google	18235
7	wordpress	17667
8	arstechnica	13749
9	wired	12841

Now that we have good parsing of the URL to get the source, let's put together a dataset of source and titles. This will be our labeled dataset for AI Platform.

In [22]:

```
from google.cloud import bigquery
bq = bigquery.Client(project=PROJECT)

query="""
SELECT source, LOWER(REGEXP_REPLACE(title, '[^a-zA-Z0-9 $.-]', ' ')) AS title FR
  (SELECT
    ARRAY_REVERSE(SPLIT(REGEXP_EXTRACT(url, '.*://(.[^/]+)/'), '.'))[OFFSET(1)]
    title
  FROM
    `bigquery-public-data.hacker_news.stories`
  WHERE
    REGEXP_CONTAINS(REGEXP_EXTRACT(url, '.*://(.[^/]+)/'), '.com$')
    AND LENGTH(title) > 10
  )
WHERE (source = 'github' OR source = 'nytimes' OR source = 'techcrunch')
"""

df = bq.query(query + " LIMIT 5").to_dataframe()
df.head()
```

Out[22]:

	source	title
0	github	django outbox
1	github	webscrapper using node.js deferred cheerio...
2	techcrunch	flashnotes picks up another $3.6m
3	github	a git user s guide to svn because at least 10...
4	github	show hn cmake module to take care of git subm...

For ML training, we will need to split our dataset into training and evaluation datasets (and perhaps an independent test dataset if we are going to do model or feature selection based on the evaluation dataset).

A simple, repeatable way to do this is to use the hash of a well-distributed column in our data (See https://www.oreilly.com/learning/repeatable-sampling-of-data-sets-in-bigquery-for-machine-learning).

In [23]:

```
traindf = bq.query(query + " AND ABS(MOD(FARM_FINGERPRINT(title), 4)) > 0").to_d
evaldf  = bq.query(query + " AND ABS(MOD(FARM_FINGERPRINT(title), 4)) = 0").to_d
```

Below we can see that roughly 75% of the data is used for training, and 25% for evaluation.

We can also see that within each dataset, the classes are roughly balanced.

In [24]:

```
traindf['source'].value_counts()
```

Out[24]:

```
github        27445
techcrunch    23131
nytimes       21586
Name: source, dtype: int64
```

In [25]:

```
evaldf['source'].value_counts()
```

Out[25]:

```
github        9080
techcrunch    7760
nytimes       7201
Name: source, dtype: int64
```

Finally we will save our data, which is currently in-memory, to disk.

In [26]:

```
import os, shutil
DATADIR='data/txtcls'
shutil.rmtree(DATADIR, ignore_errors=True)
os.makedirs(DATADIR)
traindf.to_csv( os.path.join(DATADIR,'train.tsv'), header=False, index=False, en
evaldf.to_csv( os.path.join(DATADIR,'eval.tsv'), header=False, index=False, enco
```

In [27]:

```
!head -3 data/txtcls/train.tsv
```

```
github  this guy just found out how to bypass adblocker
github  show hn  dodo   command line task management for developers
github  show hn  webservicemock   mock out external calls for local development
```

In [28]:

```
!wc -l data/txtcls/*.tsv
```

```
 24041 data/txtcls/eval.tsv
 72162 data/txtcls/train.tsv
 96203 total
```

TensorFlow/Keras Code

Please explore the code in this directory: `model.py` contains the TensorFlow model and `task.py` parses command line arguments and launches off the training job.

In particular look for the following:

1. tf.keras.preprocessing.text.Tokenizer.fit_on_texts() to generate a mapping from our word vocabulary to integers
2. tf.keras.preprocessing.text.Tokenizer.texts_to_sequences() to encode our sentences into a sequence of their respective word-integers
3. tf.keras.preprocessing.sequence.pad_sequences() to pad all sequences to be the same length

The embedding layer in the keras model takes care of one-hot encoding these integers and learning a dense emedding represetation from them.

Finally we pass the embedded text representation through a CNN model pictured below

Run Locally (optional step)

Let's make sure the code compiles by running locally for a fraction of an epoch. This may not work if you don't have all the packages installed locally for gcloud (such as in Colab). This is an optional step; move on to training on the cloud.

In [ ]:

```
%%bash
pip install google-cloud-storage
rm -rf txtcls_trained
gcloud ai-platform local train \
   --module-name=trainer.task \
   --package-path=${PWD}/txtclsmodel/trainer \
   -- \
   --output_dir=${PWD}/txtcls_trained \
   --train_data_path=${PWD}/data/txtcls/train.tsv \
   --eval_data_path=${PWD}/data/txtcls/eval.tsv \
   --num_epochs=0.1
```

Train on the Cloud

Let's first copy our training data to the cloud:

In [33]:

```
%%bash
```

```
gsutil cp data/txtcls/*.tsv gs://${BUCKET}/txtcls/
```

```
Copying file://data/txtcls/eval.tsv [Content-Type=text/tab-separated-values]...
Copying file://data/txtcls/train.tsv [Content-Type=text/tab-separated-values]...
- [2 files][  5.4 MiB/  5.4 MiB]
Operation completed over 2 objects/5.4 MiB.
```

In [40]:

```
%%bash
OUTDIR=gs://${BUCKET}/txtcls/trained_fromscratch
JOBNAME=txtcls_$(date -u +%y%m%d_%H%M%S)
gsutil -m rm -rf $OUTDIR
gcloud ai-platform jobs submit training $JOBNAME \
 --region=$REGION \
 --module-name=trainer.task \
 --package-path=${PWD}/txtclsmodel/trainer \
 --job-dir=$OUTDIR \
 --scale-tier=BASIC_GPU \
 --runtime-version 2.3 \
 --python-version 3.7 \
 -- \
 --output_dir=$OUTDIR \
 --train_data_path=gs://${BUCKET}/txtcls/train.tsv \
 --eval_data_path=gs://${BUCKET}/txtcls/eval.tsv \
 --num_epochs=5
```

```
jobId: txtcls_211026_184543
state: QUEUED
```

```
Removing gs://qwiklabs-gcp-01-989eaccf099b/txtcls/trained_fromscratch/packages/c
c984fd50990c1720ab24b1e74c0e92b181ff765fae7b0d6be92b19b736c766c/text_classificat
ion-1.0.tar.gz#1635273581586929...
/ [1/1 objects] 100% Done
Operation completed over 1 objects.
Job [txtcls_211026_184543] submitted successfully.
Your job is still active. You may view the status of your job with the command

  $ gcloud ai-platform jobs describe txtcls_211026_184543

or continue streaming the logs with the command

  $ gcloud ai-platform jobs stream-logs txtcls_211026_184543
```

Change the job name appropriately. View the job in the console, and wait until the job is complete.

In [41]:

```
!gcloud ai-platform jobs describe txtcls_211026_184543
```

```
createTime: '2021-10-26T18:45:47Z'
endTime: '2021-10-26T19:01:02Z'
etag: AZUCOSlHSPc=
jobId: txtcls_211026_184543
startTime: '2021-10-26T18:56:00Z'
state: SUCCEEDED
trainingInput:
  args:
  - --output_dir=gs://qwiklabs-gcp-01-989eaccf099b/txtcls/trained_fromscratch
  - --train_data_path=gs://qwiklabs-gcp-01-989eaccf099b/txtcls/train.tsv
  - --eval_data_path=gs://qwiklabs-gcp-01-989eaccf099b/txtcls/eval.tsv
  - --num_epochs=5
  jobDir: gs://qwiklabs-gcp-01-989eaccf099b/txtcls/trained_fromscratch
  packageUris:
  - gs://qwiklabs-gcp-01-989eaccf099b/txtcls/trained_fromscratch/packages/e95356
d41ff562b38a91d71b36548188f00edfabe96250863f06cecf0875cda2/text_classification-
1.0.tar.gz
  pythonModule: trainer.task
  pythonVersion: '3.7'
  region: us-central1
  runtimeVersion: '2.3'
  scaleTier: BASIC_GPU
trainingOutput:
  consumedMLUnits: 0.28

View job in the Cloud Console at:
https://console.cloud.google.com/mlengine/jobs/txtcls_211026_184543?project=qwik
labs-gcp-01-989eaccf099b

View logs at:
https://console.cloud.google.com/logs?resource=ml_job%2Fjob_id%2Ftxtcls_211026_1
84543&project=qwiklabs-gcp-01-989eaccf099b
```

Results

What accuracy did you get? You should see around 80%.

Rerun with Pre-trained Embedding

We will use the popular GloVe embedding which is trained on Wikipedia as well as various news sources like the New York Times.

You can read more about Glove at the project homepage: https://nlp.stanford.edu/projects/glove/

You can download the embedding files directly from the stanford.edu site, but we've rehosted it in a GCS bucket for faster download speed.

In [42]:

```
!gsutil cp gs://cloud-training-demos/courses/machine_learning/deepdive/09_sequen
```

```
Copying gs://cloud-training-demos/courses/machine_learning/deepdive/09_sequence/
text_classification/glove.6B.200d.txt [Content-Type=text/plain]...
/ [1 files][661.3 MiB/661.3 MiB]
Operation completed over 1 objects/661.3 MiB.
```

Once the embedding is downloaded re-run your cloud training job with the added command line argument:

```
--embedding_path=gs://${BUCKET}/txtcls/glove.6B.200d.txt
```

While the final accuracy may not change significantly, you should notice the model is able to converge to it much more quickly because it no longer has to learn an embedding from scratch.

References

- This implementation is based on code from: https://github.com/google/eng-edu/tree/master/ml/guides/text_classification.

- See the full text classification tutorial at: https://developers.google.com/machine-learning/guides/text-classification/

Next step

Client-side tokenizing in Python is hugely problematic. See Text classification with native serving for how to carry out the preprocessing in the serving function itself.