ivyclare Tutorial on Federated Learning with LSTMs in Pytorch

History

1 contributor

Raw | Blame

1639 lines (1639 sloc) 69.4 KB

(https://colab.research.google.com/github/ivyclare/federated-learning-on-raspberry-pi/blob/ivy_branch/Federated_Learning_With_LSTM.ipynb)

The Complete Beginners Guide to Federated Learning With LSTM

on Movie Reviews Dataset

We are going to implement this in the following steps:

- Create devices (Virtual Workers)
- Distribute our data to those devices
- Create our model
- Send our model to the devices (Cause our model is located in our computer , while the data is located in their machines)
- Do normal Training -Get the smarter model back from devices

In [1]:
```
from google.colab import drive
drive.mount('/content/drive')
# Change To The WORKING DIRECTORY
%cd /content/drive/My Drive/Colab Notebooks/PrivateAI
!pwd
```

```
Go to this URL in a browser: https://accounts.google.co
m/o/oauth2/auth?client_id=947318989803-6bn6qk8qdgf4n4g3
pfee6491hc0brc4i.apps.googleusercontent.com&redirect_ur
i=urn%3Aietf%3Awg%3Aoauth%3A2.0%3Aoob&scope=email%20htt
ps%3A%2F%2Fwww.googleapis.com%2Fauth%2Fdocs.test%20http
s%3A%2F%2Fwww.googleapis.com%2Fauth%2Fdrive%20https%3A%
2F%2Fwww.googleapis.com%2Fauth%2Fdrive.photos.readonly%
20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fpeopleapi.r
eadonly&response_type=code

Enter your authorization code:
··········
Mounted at /content/drive
/content/drive/My Drive/Colab Notebooks/PrivateAI
/content/drive/My Drive/Colab Notebooks/PrivateAI
```

Installing Pysyft

Pysyft is an extension of Pytorch that is needed inorder to perform Federated Learning. Since we are using Google Colab for this project, we only need to run the command below to install Pysyft and we are good to go.

If you are not using Google Colab, please follow the instructions here , to set up your environment.

In [2]:
```
!pip install syft
```

```
Collecting syft
  Downloading https://files.pythonhosted.org/packages/5
b/25/633ddb891b3c4927bd03311a04ece038387faecb46120b8429
ed28c72c13/syft-0.1.23a1-py3-none-any.whl (251kB)
     |████████████████████████████████| 256kB 5.0MB/s
Requirement already satisfied: Flask>=1.0.2 in /usr/loc
al/lib/python3.6/dist-packages (from syft) (1.1.1)
Collecting msgpack>=0.6.1 (from syft)
  Downloading https://files.pythonhosted.org/packages/9
2/7e/ae9e91c1bb8d846efafd1f353476e3fd7309778b582d2fb4ce
a4cc15b9a2/msgpack-0.6.1-cp36-cp36m-manylinux1_x86_64.w
```

```
hl (248kB)
     |████████████████████████████████| 256kB 26.0MB/s
Collecting lz4>=2.1.6 (from syft)
  Downloading https://files.pythonhosted.org/packages/0
a/c6/96bbb3525a63ebc53ea700cc7d37ab9045542d33b4d262d0f0
408ad9bbf2/lz4-2.1.10-cp36-cp36m-manylinux1_x86_64.whl
(385kB)
     |████████████████████████████████| 389kB 45.4MB/s
Requirement already satisfied: torchvision==0.3.0 in /u
sr/local/lib/python3.6/dist-packages (from syft) (0.3.
0)
Requirement already satisfied: torch==1.1 in /usr/loca
l/lib/python3.6/dist-packages (from syft) (1.1.0)
Requirement already satisfied: numpy>=1.14.0 in /usr/lo
cal/lib/python3.6/dist-packages (from syft) (1.16.4)
Requirement already satisfied: scikit-learn>=0.21.0 in
/usr/local/lib/python3.6/dist-packages (from syft) (0.2
1.3)
Collecting websocket-client>=0.56.0 (from syft)
  Downloading https://files.pythonhosted.org/packages/2
9/19/44753eab1fdb50770ac69605527e8859468f3c0fd7dc5a76dd
9c4dbd7906/websocket_client-0.56.0-py2.py3-none-any.whl
(200kB)
     |████████████████████████████████| 204kB 35.6MB/s
Requirement already satisfied: tblib>=1.4.0 in /usr/loc
al/lib/python3.6/dist-packages (from syft) (1.4.0)
Collecting flask-socketio>=3.3.2 (from syft)
  Downloading https://files.pythonhosted.org/packages/6
6/44/edc4715af85671b943c18ac8345d0207972284a0cd630126ff
5251faa08b/Flask_SocketIO-4.2.1-py2.py3-none-any.whl
Collecting tf-encrypted!=0.5.7,>=0.5.4 (from syft)
  Downloading https://files.pythonhosted.org/packages/1
f/82/cf15aeac92525da2f794956712e7ebf418819390dec783430e
e242b52d0b/tf_encrypted-0.5.8-py3-none-manylinux1_x86_6
4.whl (2.1MB)
     |████████████████████████████████| 2.1MB 45.2MB/s
Collecting websockets>=7.0 (from syft)
  Downloading https://files.pythonhosted.org/packages/f
0/4b/ad228451b1c071c5c52616b7d4298ebcfcac5ae8515ede959d
b19e4cd56d/websockets-8.0.2-cp36-cp36m-manylinux1_x86_6
4.whl (72kB)
     |████████████████████████████████| 81kB 30.7MB/s
Collecting zstd>=1.4.0.0 (from syft)
  Downloading https://files.pythonhosted.org/packages/2
2/37/6a7ba746ebddbd6cd06de84367515d6bc239acd94fb3e0b1c8
5788176ca2/zstd-1.4.1.0.tar.gz (454kB)
     |████████████████████████████████| 460kB 44.7MB/s
Requirement already satisfied: Werkzeug>=0.15 in /usr/l
ocal/lib/python3.6/dist-packages (from Flask>=1.0.2->sy
ft) (0.15.5)
Requirement already satisfied: itsdangerous>=0.24 in /u
sr/local/lib/python3.6/dist-packages (from Flask>=1.0.2
->syft) (1.1.0)
Requirement already satisfied: Jinja2>=2.10.1 in /usr/l
ocal/lib/python3.6/dist-packages (from Flask>=1.0.2->sy
ft) (2.10.1)
Requirement already satisfied: click>=5.1 in /usr/loca
l/lib/python3.6/dist-packages (from Flask>=1.0.2->syft)
(7.0)
Requirement already satisfied: six in /usr/local/lib/py
thon3.6/dist-packages (from torchvision==0.3.0->syft)
(1.12.0)
Requirement already satisfied: pillow>=4.1.1 in /usr/lo
cal/lib/python3.6/dist-packages (from torchvision==0.3.
0->syft) (4.3.0)
Requirement already satisfied: scipy>=0.17.0 in /usr/lo
cal/lib/python3.6/dist-packages (from scikit-learn>=0.2
1.0->syft) (1.3.1)
Requirement already satisfied: joblib>=0.11 in /usr/loc
al/lib/python3.6/dist-packages (from scikit-learn>=0.2
1.0->syft) (0.13.2)
Collecting python-socketio>=4.3.0 (from flask-socketio>
```

```
=3.3.2->syft)
  Downloading https://files.pythonhosted.org/packages/3
5/b0/22c3f785f23fec5c7a815f47c55d7e7946a67ae2129ff60414
8e939d3bdb/python_socketio-4.3.1-py2.py3-none-any.whl
(49kB)
     |████████████████████████████████| 51kB 18.8MB/s
Requirement already satisfied: tensorflow<2,>=1.12.0 in
/usr/local/lib/python3.6/dist-packages (from tf-encrypt
ed!=0.5.7,>=0.5.4->syft) (1.14.0)
Collecting pyyaml>=5.1 (from tf-encrypted!=0.5.7,>=0.5.
4->syft)
  Downloading https://files.pythonhosted.org/packages/e
3/e8/b3212641ee2718d556df0f23f78de8303f068fe29cdaa7a910
18849582fe/PyYAML-5.1.2.tar.gz (265kB)
     |████████████████████████████████| 266kB 48.2MB/s
Requirement already satisfied: MarkupSafe>=0.23 in /us
r/local/lib/python3.6/dist-packages (from Jinja2>=2.10.
1->Flask>=1.0.2->syft) (1.1.1)
Requirement already satisfied: olefile in /usr/local/li
b/python3.6/dist-packages (from pillow>=4.1.1->torchvis
ion==0.3.0->syft) (0.46)
Collecting python-engineio>=3.9.0 (from python-socketio
>=4.3.0->flask-socketio>=3.3.2->syft)
  Downloading https://files.pythonhosted.org/packages/2
b/20/8e3ba16102ae2e245d70d9cb9fa48b076253fdb036dc43eea1
42294c2897/python_engineio-3.9.3-py2.py3-none-any.whl
(119kB)
     |████████████████████████████████| 122kB 46.4MB/s
Requirement already satisfied: astor>=0.6.0 in /usr/loc
al/lib/python3.6/dist-packages (from tensorflow<2,>=1.1
2.0->tf-encrypted!=0.5.7,>=0.5.4->syft) (0.8.0)
Requirement already satisfied: keras-preprocessing>=1.
0.5 in /usr/local/lib/python3.6/dist-packages (from ten
sorflow<2,>=1.12.0->tf-encrypted!=0.5.7,>=0.5.4->syft)
(1.1.0)
Requirement already satisfied: grpcio>=1.8.6 in /usr/lo
cal/lib/python3.6/dist-packages (from tensorflow<2,>=1.
12.0->tf-encrypted!=0.5.7,>=0.5.4->syft) (1.15.0)
Requirement already satisfied: tensorflow-estimator<1.1
5.0rc0,>=1.14.0rc0 in /usr/local/lib/python3.6/dist-pac
kages (from tensorflow<2,>=1.12.0->tf-encrypted!=0.5.7,
>=0.5.4->syft) (1.14.0)
Requirement already satisfied: protobuf>=3.6.1 in /usr/
local/lib/python3.6/dist-packages (from tensorflow<2,>=
1.12.0->tf-encrypted!=0.5.7,>=0.5.4->syft) (3.7.1)
Requirement already satisfied: tensorboard<1.15.0,>=1.1
4.0 in /usr/local/lib/python3.6/dist-packages (from ten
sorflow<2,>=1.12.0->tf-encrypted!=0.5.7,>=0.5.4->syft)
(1.14.0)
Requirement already satisfied: gast>=0.2.0 in /usr/loca
l/lib/python3.6/dist-packages (from tensorflow<2,>=1.1
2.0->tf-encrypted!=0.5.7,>=0.5.4->syft) (0.2.2)
Requirement already satisfied: absl-py>=0.7.0 in /usr/l
ocal/lib/python3.6/dist-packages (from tensorflow<2,>=
1.12.0->tf-encrypted!=0.5.7,>=0.5.4->syft) (0.7.1)
Requirement already satisfied: google-pasta>=0.1.6 in /
usr/local/lib/python3.6/dist-packages (from tensorflow<
2,>=1.12.0->tf-encrypted!=0.5.7,>=0.5.4->syft) (0.1.7)
Requirement already satisfied: keras-applications>=1.0.
6 in /usr/local/lib/python3.6/dist-packages (from tenso
rflow<2,>=1.12.0->tf-encrypted!=0.5.7,>=0.5.4->syft)
(1.0.8)
Requirement already satisfied: wheel>=0.26 in /usr/loca
l/lib/python3.6/dist-packages (from tensorflow<2,>=1.1
2.0->tf-encrypted!=0.5.7,>=0.5.4->syft) (0.33.4)
Requirement already satisfied: wrapt>=1.11.1 in /usr/lo
cal/lib/python3.6/dist-packages (from tensorflow<2,>=1.
12.0->tf-encrypted!=0.5.7,>=0.5.4->syft) (1.11.2)
Requirement already satisfied: termcolor>=1.1.0 in /us
r/local/lib/python3.6/dist-packages (from tensorflow<2,
>=1.12.0->tf-encrypted!=0.5.7,>=0.5.4->syft) (1.1.0)
Requirement already satisfied: setuptools in /usr/loca
```

```
Requirement already satisfied: setuptools in /usr/
l/lib/python3.6/dist-packages (from protobuf>=3.6.1->te
nsorflow<2,>=1.12.0->tf-encrypted!=0.5.7,>=0.5.4->syft)
(41.0.1)
Requirement already satisfied: markdown>=2.6.8 in /usr/
local/lib/python3.6/dist-packages (from tensorboard<1.1
5.0,>=1.14.0->tensorflow<2,>=1.12.0->tf-encrypted!=0.5.
7,>=0.5.4->syft) (3.1.1)
Requirement already satisfied: h5py in /usr/local/lib/p
ython3.6/dist-packages (from keras-applications>=1.0.6-
>tensorflow<2,>=1.12.0->tf-encrypted!=0.5.7,>=0.5.4->sy
ft) (2.8.0)
Building wheels for collected packages: zstd, pyyaml
  Building wheel for zstd (setup.py) ... done
  Created wheel for zstd: filename=zstd-1.4.1.0-cp36-cp
36m-linux_x86_64.whl size=1067106 sha256=a3fca792cca767
034ab0b2581939af5b0abacf8cf3b6792c628c5ee377bf318f
  Stored in directory: /root/.cache/pip/wheels/66/3f/e
e/ac08c81af7c1b24a80c746df669ea3cb37542d27877d66ccf4
  Building wheel for pyyaml (setup.py) ... done
  Created wheel for pyyaml: filename=PyYAML-5.1.2-cp36-
cp36m-linux_x86_64.whl size=44105 sha256=250195edbf14e1
ed69e9c1ed2681e8cec3dbe92c8608c1dc0b032afc06dfa029
  Stored in directory: /root/.cache/pip/wheels/d9/45/d
d/65f0b38450c47cf7e5312883deb97d065e030c5cca0a365030
Successfully built zstd pyyaml
Installing collected packages: msgpack, lz4, websocket-
client, python-engineio, python-socketio, flask-socketi
o, pyyaml, tf-encrypted, websockets, zstd, syft
  Found existing installation: msgpack 0.5.6
    Uninstalling msgpack-0.5.6:
      Successfully uninstalled msgpack-0.5.6
  Found existing installation: PyYAML 3.13
    Uninstalling PyYAML-3.13:
      Successfully uninstalled PyYAML-3.13
Successfully installed flask-socketio-4.2.1 lz4-2.1.10
msgpack-0.6.1 python-engineio-3.9.3 python-socketio-4.
3.1 pyyaml-5.1.2 syft-0.1.23a1 tf-encrypted-0.5.8 webso
cket-client-0.56.0 websockets-8.0.2 zstd-1.4.1.0
```

Import Required Libraries

The next step is to import the required libraries and hook Pytorch using *sy.TorchHook* which makes the extended extended functions on Pytorch tensors available to us.

In [10]:
```
import torch
import syft as sy
hook = sy.TorchHook(torch)
```

```
W0819 05:16:27.754048 140679342729088 hook.py:98] Torch
was already hooked... skipping hooking process
```

Creating New Workers

As earlier mentioned, in order to perform Federated Learning we need to have data on more different devices and we have to send our model to those devices where the training will be done. Hence, we need to create 2 devices. We will assume the person with the first device is called Bob and the second device called Alice.

In [0]:
```
bob = sy.VirtualWorker(hook, id="bob")
alice = sy.VirtualWorker(hook, id="alice")
```

The LSTM Network

A short description of lstm and how they work here

Importing Libraries for LSTM

In [12]:
```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.
g. pd.read_csv)
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import seaborn as sns

import torch
import torch.nn.functional as F
from torchtext import datasets
from torchtext import data
import torch.optim as optim
from torch import nn,optim
import torch.nn.functional as F
from torch.utils.data import *

import random
import matplotlib.pyplot as plt
%matplotlib inline

import time
import json
import copy
import os
import glob

from PIL import Image

# check if CUDA is available
train_on_gpu = torch.cuda.is_available()

if not train_on_gpu:
    device = torch.device('cuda')
    print('CUDA is not available.  Training on CPU ...'
)
else:
    device = torch.device('cpu')
    print('CUDA is available!  Training on GPU ...')

device
```

```
CUDA is not available.  Training on CPU ...
```

Out[12]:
```
device(type='cuda')
```

Loading the Data

We are going to use the IMDB dataset which is provided in Pytorch in the torchtext.data.Dataset (https://torchtext.readthedocs.io/en/latest/datasets.html#imdb). And we split the data into train and test sets.

In [0]:
```
import numpy as np

# read data from text files
with open('data/reviews.txt', 'r') as f:
    reviews = f.read()
with open('data/labels.txt', 'r') as f:
    labels = f.read()
```

We take a look at what our data and then split the training data into a train set and validation set. 80% of the data is used for training and 10% for validation.

In [14]:
```
print(reviews[:5000])
```

In [14]:
```
print(reviews[:5000])
print()
print(labels[:20])
```

```
bromwell high is a cartoon comedy . it ran at the same
time as some other programs about school life  such as
teachers  . my   years in the teaching profession lead
me to believe that bromwell high  s satire is much clos
er to reality than is  teachers  . the scramble to surv
ive financially  the insightful students who can see ri
ght through their pathetic teachers  pomp  the pettines
s of the whole situation  all remind me of the schools
i knew and their students . when i saw the episode in w
hich a student repeatedly tried to burn down the school
i immediately recalled . . . . . at . . . . . .
. . . . high . a classic line inspector i  m here to sa
ck one of your teachers . student welcome to bromwell h
igh . i expect that many adults of my age think that br
omwell high is far fetched . what a pity that it isn  t
story of a man who has unnatural feelings for a pig . s
tarts out with a opening scene that is a terrific examp
le of absurd comedy . a formal orchestra audience is tu
rned into an insane  violent mob by the crazy chantings
of it  s singers . unfortunately it stays absurd the wh
ole time with no general narrative eventually making it
just too off putting . even those from the era should b
e turned off . the cryptic dialogue would make shakespe
are seem easy to a third grader . on a technical level
it  s better than you might think with some good cinema
tography by future great vilmos zsigmond . future stars
sally kirkland and frederic forrest can be seen briefly
.
homelessness  or houselessness as george carlin stated
has been an issue for years but never a plan to help th
ose on the street that were once considered human who d
id everything from going to school  work  or vote for t
he matter . most people think of the homeless as just a
lost cause while worrying about things such as racism
the war on iraq  pressuring kids to succeed  technology
the elections  inflation  or worrying if they  ll be ne
xt to end up on the streets .  br    br   but what if y
ou were given a bet to live on the streets for a month
without the luxuries you once had from a home  the ente
rtainment sets  a bathroom  pictures on the wall  a com
puter  and everything you once treasure to see what it
s like to be homeless  that is goddard bolt  s lesson .
br    br   mel brooks  who directs  who stars as bolt p
lays a rich man who has everything in the world until d
eciding to make a bet with a sissy rival  jeffery tambo
r  to see if he can live in the streets for thirty days
without the luxuries if bolt succeeds  he can do what h
e wants with a future project of making more buildings
. the bet  s on where bolt is thrown on the street with
a bracelet on his leg to monitor his every move where h
e can  t step off the sidewalk . he  s given the nickna
me pepto by a vagrant after it  s written on his forehe
ad where bolt meets other characters including a woman
by the name of molly  lesley ann warren  an ex  dancer
who got divorce before losing her home  and her pals sa
ilor  howard morris  and fumes  teddy wilson  who are a
lready used to the streets . they  re survivors . bolt
isn  t . he  s not used to reaching mutual agreements l
ike he once did when being rich where it  s fight or fl
ight  kill or be killed .  br    br   while the love co
nnection between molly and bolt wasn  t necessary to pl
ot  i found  life stinks  to be one of mel brooks  obse
rvant films where prior to being a comedy  it shows a t
ender side compared to his slapstick work such as blazi
ng saddles  young frankenstein  or spaceballs for the m
atter  to show what it  s like having something valuabl
e before losing it the next day or on the other hand ma
king a stupid bet like all rich people do when they don
t know what to do with their money . maybe they should
```

```
t know what to do with their money . maybe they should
give it to the homeless instead of using it like monopo
ly money .  br    br   or maybe this film will inspire
you to help others .
airport    starts as a brand new luxury    plane is loa
ded up with valuable paintings  such belonging to rich
businessman philip stevens  james stewart  who is flyin
g them  a bunch of vip  s to his estate in preparation
of it being opened to the public as a museum  also on b
oard is stevens daughter julie  kathleen quinlan   her
son . the luxury jetliner takes off as planned but mid
air the plane is hi  jacked by the co  pilot chambers
robert foxworth   his two accomplice  s banker  monte m
arkham   wilson  michael pataki  who knock the passenge
rs  crew out with sleeping gas  they plan to steal the
valuable cargo  land on a disused plane strip on an iso
lated island but while making his descent chambers almo
st hits an oil rig in the ocean  loses control of the p
lane sending it crashing into the sea where it sinks to
the bottom right bang in the middle of the bermuda tria
ngle . with air in short supply  water leaking in  havi
ng flown over    miles off course the problems mount fo
r the survivor  s as they await help with time fast run
ning out . . .  br    br   also known under the sligh

positive
negative
po
```

Data Preprocessing

In [15]:
```
from string import punctuation

print(punctuation)

# get rid of punctuation
reviews = reviews.lower() # lowercase, standardize
all_text = ''.join([c for c in reviews if c not in punc
tuation])
```

```
!"#$%&'()*+,-./:;<=>?@[\]^_`{|}~
```

In [0]:
```
# split by new lines and spaces

reviews_split = all_text.split('\n')

all_text = ' '.join(reviews_split)

# create a list of words
words = all_text.split()
```

In [17]:
```
words[:30]
```

Out[17]:
```
['bromwell',
 'high',
 'is',
 'a',
 'cartoon',
 'comedy',
 'it',
 'ran',
 'at',
 'the',
 'same',
 'time',
 'as',
 'some',
 'other',
 'programs',
 'about',
 'school',
```

```
 'life',
 'such',
 'as',
 'teachers',
 'my',
 'years',
 'in',
 'the',
 'teaching',
 'profession',
 'lead',
 'me']
```

Encoding the words

In [18]:
```
# feel free to use this import
from collections import Counter

## Build a dictionary that maps words to integers
counts =  Counter(words)
vocab = sorted(counts, key=counts.get, reverse=True)
vocab_to_int = {word: ii for ii, word in enumerate(voca
b, 1)}

## use the dict to tokenize each review in reviews_spli
t
## store the tokenized reviews in reviews_ints
reviews_ints = []
for review in reviews_split:
  reviews_ints.append([vocab_to_int[word] for word in r
eview.split()])


  # stats about vocabulary
print('Unique words: ', len((vocab_to_int)))  # should
 ~ 74000+
print()

# print tokens in first review
print('Tokenized review: \n', reviews_ints[:1])
```

```
Unique words:  74072

Tokenized review:
 [[21025, 308, 6, 3, 1050, 207, 8, 2138, 32, 1, 171, 5
7, 15, 49, 81, 5785, 44, 382, 110, 140, 15, 5194, 60, 1
54, 9, 1, 4975, 5852, 475, 71, 5, 260, 12, 21025, 308,
13, 1978, 6, 74, 2395, 5, 613, 73, 6, 5194, 1, 24103,
5, 1983, 10166, 1, 5786, 1499, 36, 51, 66, 204, 145, 6
7, 1199, 5194, 19869, 1, 37442, 4, 1, 221, 883, 31, 298
8, 71, 4, 1, 5787, 10, 686, 2, 67, 1499, 54, 10, 216,
1, 383, 9, 62, 3, 1406, 3686, 783, 5, 3483, 180, 1, 38
2, 10, 1212, 13583, 32, 308, 3, 349, 341, 2913, 10, 14
3, 127, 5, 7690, 30, 4, 129, 5194, 1406, 2326, 5, 2102
5, 308, 10, 528, 12, 109, 1448, 4, 60, 543, 102, 12, 21
025, 308, 6, 227, 4146, 48, 3, 2211, 12, 8, 215, 23]]
```

Encoding the labels

In [0]:
```
# 1=positive, 0=negative label conversion
labels_split = labels.split('\n')
encoded_labels = np.array([1 if label == 'positive' els
e 0 for label in labels_split])
```

Removing outliers

In [20]:
```
# stats about vocabulary
```

```
print('Unique words: ', len((vocab_to_int)))  # should
 ~ 74000+
print()

# print tokens in first review
print('Tokenized review: \n', reviews_ints[:1])
```

```
Unique words:  74072

Tokenized review:
 [[21025, 308, 6, 3, 1050, 207, 8, 2138, 32, 1, 171, 5
7, 15, 49, 81, 5785, 44, 382, 110, 140, 15, 5194, 60, 1
54, 9, 1, 4975, 5852, 475, 71, 5, 260, 12, 21025, 308,
13, 1978, 6, 74, 2395, 5, 613, 73, 6, 5194, 1, 24103,
5, 1983, 10166, 1, 5786, 1499, 36, 51, 66, 204, 145, 6
7, 1199, 5194, 19869, 1, 37442, 4, 1, 221, 883, 31, 298
8, 71, 4, 1, 5787, 10, 686, 2, 67, 1499, 54, 10, 216,
1, 383, 9, 62, 3, 1406, 3686, 783, 5, 3483, 180, 1, 38
2, 10, 1212, 13583, 32, 308, 3, 349, 341, 2913, 10, 14
3, 127, 5, 7690, 30, 4, 129, 5194, 1406, 2326, 5, 2102
5, 308, 10, 528, 12, 109, 1448, 4, 60, 543, 102, 12, 21
025, 308, 6, 227, 4146, 48, 3, 2211, 12, 8, 215, 23]]
```

In [21]:
```
print('Number of reviews before removing outliers: ', l
en(reviews_ints))

## remove any reviews/labels with zero length from the
 reviews_ints list.
non_zero_idx = [ii for ii, review in enumerate(reviews_
ints) if len(review) != 0]

reviews_ints = [reviews_ints[ii] for ii in non_zero_idx
]
encoded_labels = np.array([encoded_labels[ii] for ii in
non_zero_idx])

print('Number of reviews after removing outliers: ', le
n(reviews_ints))
```

```
Number of reviews before removing outliers:  25001
Number of reviews after removing outliers:  25000
```

Padding Sequences

In [0]:
```
def pad_features(reviews_ints, seq_length):
    ''' Return features of review_ints, where each revi
ew is padded with 0's
        or truncated to the input seq_length.
    '''
    #getting the correct row x col
    features= np.zeros((len(reviews_ints), seq_length),
dtype=int)

    # for each review, grab that review and
    for i, row in enumerate(reviews_ints):
      features[i, -len(row):] =  np.array(row)[:seq_len
gth]

    return features
```

In [23]:
```
# Test your implementation!

seq_length = 200

features = pad_features(reviews_ints, seq_length=seq_le
ngth)

## test statements - do not change - ##
assert len(features)==len(reviews_ints), "Your features
should have as many rows as reviews."
```

```
assert len(features[0])==seq_length, "Each feature row
 should contain seq_length values."

# print first 10 values of the first 30 batches
print(features[:30,:10])
```

```
[[    0     0     0     0     0     0     0     0     0
0]
 [    0     0     0     0     0     0     0     0     0
0]
 [22382    42 46418    15   706 17139  3389    47    77
35]
 [ 4505   505    15     3  3342   162  8312  1652     6
4819]
 [    0     0     0     0     0     0     0     0     0
0]
 [    0     0     0     0     0     0     0     0     0
0]
 [    0     0     0     0     0     0     0     0     0
0]
 [    0     0     0     0     0     0     0     0     0
0]
 [    0     0     0     0     0     0     0     0     0
0]
 [   54    10    14   116    60   798   552    71   364
5]
 [    0     0     0     0     0     0     0     0     0
0]
 [    0     0     0     0     0     0     0     0     0
0]
 [    0     0     0     0     0     0     0     0     0
0]
 [    1   330   578    34     3   162   748  2731     9
325]
 [    9    11 10171  5305  1946   689   444    22   280
673]
 [    0     0     0     0     0     0     0     0     0
0]
 [    1   307 10399  2069  1565  6202  6528  3288 17946
10628]
 [    0     0     0     0     0     0     0     0     0
0]
 [   21   122  2069  1565   515  8181    88     6  1325
1182]
 [    1    20     6    76    40     6    58    81    95
5]
 [   54    10    84   329 26230 46427    63    10    14
614]
 [   11    20     6    30  1436 32317  3769   690 15100
6]
 [    0     0     0     0     0     0     0     0     0
0]
 [    0     0     0     0     0     0     0     0     0
0]
 [   40    26   109 17952  1422     9     1   327     4
125]
 [    0     0     0     0     0     0     0     0     0
0]
 [   10   499     1   307 10399    55    74     8    13
30]
 [    0     0     0     0     0     0     0     0     0
0]
 [    0     0     0     0     0     0     0     0     0
0]
 [    0     0     0     0     0     0     0     0     0
0]]
```

Splitting Data

In [24]:
```
split_frac = 0.8
```

```
## split data into training, validation, and test data
 (features and labels, x and y)
split_idx = int(len(features)*0.8)
train_x, remaining_x = features[:split_idx], features[s
plit_idx:]
train_y, remaining_y = encoded_labels[:split_idx], enco
ded_labels[split_idx:]


test_idx = int(len(remaining_x)*0.5)
val_x, test_x = remaining_x[:test_idx], remaining_x[tes
t_idx:]
val_y, test_y = remaining_y[:test_idx], remaining_y[tes
t_idx:]

## print out the shapes of your resultant feature data
print("\t\t\tFeature Shapes:")
print("Train set: \t\t{}\n".format(train_x.shape),
      "Val set: \t\t{}\n".format(val_x.shape),
      "Test set: \t\t{}\n".format(test_x.shape))
```

```
                        Feature Shapes:
Train set:              (20000, 200)
 Val set:               (2500, 200)
 Test set:              (2500, 200)
```

Distributing the Data

Our virtual workers have been created but they don't have any data on them. After loading our data, we distribute the data between Alice and Bob. We do this by splitting the training and validation datasets into 2 and sending them to the workers using the sy.BaseDataset class.

In [25]:
```
train_x.shape[0]
```

Out[25]: 20000

In [0]:
```
train_idx = int(train_x.shape[0]/2)
valid_idx = int(val_x.shape[0]/2)
test_idx = int(test_x.shape[0]/2)

# Sending toy datasets to virtual workers
bob_train_dataset = sy.BaseDataset(torch.from_numpy(tra
in_x[:train_idx]),
                                   torch.from_numpy(trai
n_y[:train_idx])).send(bob)

alice_train_dataset = sy.BaseDataset(torch.from_numpy(t
rain_x[train_idx:]),
                                     torch.from_numpy(tr
ain_y[train_idx:])).send(alice)


bob_valid_dataset = sy.BaseDataset(torch.from_numpy(val
_x[:valid_idx]),
                                   torch.from_numpy(val_
y[:valid_idx])).send(bob)

alice_valid_dataset = sy.BaseDataset(torch.from_numpy(v
al_x[valid_idx:]),
                                   torch.from_numpy(val_
y[valid_idx:])).send(alice)


bob_test_dataset = sy.BaseDataset(torch.from_numpy(test
_x[:test_idx]),
                                   torch.from_numpy(test
 y[:test_idx])).send(bob)
```

```
alice_test_dataset = sy.BaseDataset(torch.from_numpy(te
st_x[test_idx:]),
                                    torch.from_numpy(test
_y[test_idx:])).send(alice)
```

Creating Federated DataLoaders

Now, we load datasets using dataloaders. In Federated learning, we load datasets from different devices in a federated manner using **Federated DataLoaders**

In [0]:
```
# Creating federated datasets, an extension of Pytorch
 TensorDataset class
federated_train_dataset = sy.FederatedDataset([bob_trai
n_dataset, alice_train_dataset])
federated_valid_dataset = sy.FederatedDataset([bob_vali
d_dataset, alice_valid_dataset])
federated_test_dataset = sy.FederatedDataset([bob_test_
dataset, alice_test_dataset])

BATCH_SIZE = 50

# Creating federated dataloaders, an extension of Pytor
ch DataLoader class
federated_train_loader = sy.FederatedDataLoader(federat
ed_train_dataset,
                                                shuffle
=True, batch_size=BATCH_SIZE)

federated_valid_loader = sy.FederatedDataLoader(federat
ed_valid_dataset,
                                                shuffle
=True, batch_size=BATCH_SIZE)

federated_test_loader = sy.FederatedDataLoader(federate
d_test_dataset,
                                               shuffle=
False, batch_size=BATCH_SIZE)
```

Building Our Network

In [0]:
```
class SentimentRNN(nn.Module):
    """
    The RNN model that will be used to perform Sentimen
t analysis.
    """

    def __init__(self, vocab_size, output_size, embeddi
ng_dim, hidden_dim, n_layers, drop_prob=0.5):
        """
        Initialize the model by setting up the layers.
        """
        super(SentimentRNN, self).__init__()

        self.output_size = output_size
        self.n_layers = n_layers
        self.hidden_dim = hidden_dim

        # define all layers
        #embedding layer
        self.embedding = nn.Embedding(vocab_size,embedd
ing_dim)

        #lstm layer
        self.lstm = nn.LSTM(embedding_dim,hidden_dim, n
_layers, dropout=drop_prob, batch_first = True)

        self.dropout = nn.Dropout(0.3)
```

```
        self.dropout = nn.Dropout(0.3)

        #fully connected layer
        self.fc = nn.Linear(hidden_dim, output_size)

        self.sig = nn.Sigmoid()

    def forward(self, x, hidden):
        """
        Perform a forward pass of our model on some inp
ut and hidden state.
        """
        # Batch_size used for shaping data
        batch_size = x.size(0)

        # embeddings and lstm_out
        embeds = self.embedding(x)

        lstm_out, hidden = self.lstm(embeds, hidden)

        # stack up lstm outputs
        lstm_out = lstm_out.contiguous().view(-1, self.
hidden_dim)

        # dropout and fully-connected layer
        out = self.dropout(lstm_out)

        out = self.fc(out)

        # sigmoid funtion
        sig_out = self.sig(out)

        # reshape to be batch_size first
        sig_out = sig_out.view(batch_size, -1)
        sig_out = sig_out[:, -1]  # get last batch of l
abels

        # return last sigmoid output and hidden state
        return sig_out, hidden


    def init_hidden(self, batch_size):
        ''' Initializes hidden state '''
        # Create two new tensors with sizes n_layers x
 batch_size x hidden_dim,
        # initialized to zero, for hidden state and cel
l state of LSTM

        weight = next(self.parameters()).data

        if(train_on_gpu):
          hidden = (weight.new(self.n_layers, batch_siz
e, self.hidden_dim).zero_().cuda(),
                    weight.new(self.n_layers, batch_size
, self.hidden_dim).zero_().cuda())
        else:
            hidden = (weight.new(self.n_layers, batch_s
ize, self.hidden_dim).zero_(),
                      weight.new(self.n_layers, batch_s
ize, self.hidden_dim).zero_())


        return hidden
```

In [29]:
```
# Instantiate the model w/ hyperparams
vocab_size = len(vocab_to_int)+1 # +1 for the 0 padding
+ our word tokens
output_size = 1
embedding_dim = 400
hidden_dim = 256
n_layers = 2
```

```
net = SentimentRNN(vocab_size, output_size, embedding_d
im, hidden_dim, n_layers)

print(net)
```

```
SentimentRNN(
  (embedding): Embedding(74073, 400)
  (lstm): LSTM(400, 256, num_layers=2, batch_first=Tru
e, dropout=0.5)
  (dropout): Dropout(p=0.3)
  (fc): Linear(in_features=256, out_features=1, bias=Tr
```