Program 3 - Classification using Neural Networks

George Bennett, Spring 2022 - CS CS461

References:

https://www.tensorflow.org/tutorials/customization/custom_training_walkthrough

https://www.tensorflow.org/tutorials/load_data/pandas_dataframe

Data preparation

First we use pandas to manipulate and clean up data from the csv.

In [2]:
```
import pandas as pd
```

In [3]:
```
csv = pd.read_csv("congressional_tweet_training_data.csv", encoding = 'utf8')
csv.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 592803 entries, 0 to 592802
Data columns (total 6 columns):
 #   Column          Non-Null Count   Dtype
---  ------          --------------   -----
 0   favorite_count  592803 non-null  int64
 1   full_text       592803 non-null  object
 2   hashtags        592803 non-null  object
 3   retweet_count   592803 non-null  int64
 4   year            574091 non-null  float64
 5   party_id        592803 non-null  object
dtypes: float64(1), int64(2), object(3)
memory usage: 27.1+ MB
```

Here we are going to process the data so it's a little more friendlier for our purposes

We take the CSV of Tweets and I converting the data types of each column to the appropriate ones which match the type of data for the source.

In [4]:
```
dataset = csv.copy()

dataset["full_text"] = dataset["full_text"].astype("string")
dataset["party_id"] = dataset["party_id"].astype("category")
dataset["hashtags"] = dataset["hashtags"].astype("string")
dataset["year"] = dataset["year"].astype("string")
```

A preview of the dataset

In [5]:

```
dataset
```

Out[5]:

	favorite_count	full_text	hashtags	retweet_count	year	party_id
0	0	b"RT @KUSINews: One of our longtime viewers wa...	KUSI	10	2017.0	R
1	258	b"Today I'm urging the @CDCgov to immediately ...	Coronavirus	111	2020.0	R
2	0	b'Tomorrow, #MO03 seniors graduate from Calvar...	MO03	2	2014.0	R
3	9	b'Congrats to #TeamUSA and Canton Native @JGre...	TeamUSA WorldJuniors	3	2017.0	R
4	3	b'Pleased to support @amergateways at their Ju...	ImmigrantHeritageMonth	3	2019.0	D
...	...	...	...	...	...	...
592798	3	b'This time, it focused on careers in #publics...	publicservice publicsafety	0	2017.0	R
592799	5	b'.#StormyDaniels, #MichaelWolfe, #JamesComey ...	StormyDaniels MichaelWolfe JamesComey	1	2018.0	R
592800	33	b'@NRDems The American people deserve the trut...	CultureOfCorruption	14	2020.0	D
592801	4	b'Only 2 weeks left to submit your #app to the...	app copolitics CAC16 HouseOfCode co06	3	2016.0	R
592802	155	b'The #MuslimBan remains as un-American and of...	MuslimBan	48	2020.0	D

592803 rows × 6 columns

Then we try to extract the most useful information. I tried to pair down the information as much as possible to try to get easy training. This means I tried to convert everthing, as easily as possible, to numeric values.

First I start by getting the integer length of all tweets.

In [6]:

```
text = dataset["full_text"].str.slice(1)
length = text.str.len()
length = length.to_frame()
length.columns = ['length']
```

Then I try to get the mentions of other accounts and then count the number of occurences.

In [7]:

```
mentions = dataset["full_text"].str.findall('@(\w+)')
mentions = mentions.apply(lambda list: len(list))
mentions = mentions.to_frame()
mentions.columns = ["mentions"]
```

Seperates out retweet count

In [8]:

```
retweets = dataset["retweet_count"]
```

Extracts year and from string to integer

In [9]:

```
years = dataset["year"].str.extract(r'(\d{4})')
years = years.astype("Int64")
years.columns = ["year"]
```

Here we split hashtags into lists and then count the total number of hashtags. I use a numpy array to find the unique values and then count them hashtags into a dictionary. I then take a slice of the dictionary of the top 500 hashtags and use them for prediction.

In [42]:

```
import numpy as np
import itertools
hashtags = dataset["hashtags"].str.split(' ')

tags = []
for taglist in hashtags:
    for item in taglist:
        tags.append(item.lower())

tags = np.asarray(tags)
uniqueTags = np.unique(uniqueTags)

tagList = uniqueTags.tolist()
tagDict = {}

for tag in uniqueTags:
    count = np.where(tags == tag)[0]
    count = len(ii)
    if count != 1:
        tagDict[tag] = count

topTags = dict(itertools.islice(tagDict.items(), 500))

hashtags = hashtags.apply(lambda list: len(list))
hashtags = hashtags.to_frame()
hashtags.columns = ["hashtags"]

tagDict
```

Out[42]:

```
{}
```

In [ ]:

Convert party affiliation to integers

In [11]:

```
party_id = dataset["party_id"]
party_id = party_id.apply(lambda String: "0" if (String=="D") else "1")
party_id = party_id.astype("int")
```

Collate the information

In [12]:

```
input = pd.concat([length, mentions, retweets, years, hashtags, party_id], axis=1)
labels = party_id
input = input.fillna(input.median())
input.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 592803 entries, 0 to 592802
Data columns (total 6 columns):
 #   Column         Non-Null Count   Dtype
---  ------         --------------   -----
 0   length         592803 non-null  Int64
 1   mentions       592803 non-null  int64
 2   retweet_count  592803 non-null  int64
 3   year           592803 non-null  Int64
 4   hashtags       592803 non-null  int64
 5   party_id       592803 non-null  int32
dtypes: Int64(2), int32(1), int64(3)
memory usage: 26.0 MB
```

Training Stage

In [22]:

```
import tensorflow as tf
import tensorflow.keras.layers as l
import keras
```

Validation

In [23]:

```
p = 0.05
n = int(len(input) * p)
trainset = input.iloc[0:n]
trainlabels = labels.iloc[0:n]

testset = input.iloc[n:2*n]
testlabels = labels.iloc[n:2*n]
print("Set size: {}".format(n))
```

```
Set size: 29640
```

Convert pandas dataframe to tensor

In [24]:

```
normalizer = tf.keras.layers.Normalization(axis=-1)
trainset = tf.convert_to_tensor(trainset, dtype=tf.int64)
normalizer.adapt(trainset)
normalizer(trainset)

testlabels = tf.cast(testlabels, tf.bool)

testset = tf.convert_to_tensor(testset, dtype=tf.int64)
testlabels = tf.cast(testlabels, tf.bool)
```

Network Configuration

In [26]:

```
model = keras.Sequential()
relu = tf.nn.relu
softmax = tf.nn.softmax
model.add(normalizer)
model.add(l.GaussianNoise(0.1))
model.add(l.Dense(5, activation=relu, input_shape=(5,)))
model.add(l.Dense(2, activation=relu))
model.add(l.Dense(5, activation=relu))
model.add(l.GaussianNoise(0.1))
model.add(l.Dense(1, activation=softmax))

model.compile(optimizer='adam',
              loss=tf.keras.losses.BinaryCrossentropy(from_logits=True),
              metrics=['accuracy'])
```

In [27]:

```
model.fit(trainset, trainlabels, epochs=5, batch_size=1, shuffle=True)
```

```
Epoch 1/5
29640/29640 [==============================] - 140s 5ms/step - loss: 0.0587 - accur
acy: 0.9636
Epoch 2/5
29640/29640 [==============================] - 125s 4ms/step - loss: 9.8946e-04 - a
ccuracy: 1.0000
Epoch 3/5
29640/29640 [==============================] - 133s 4ms/step - loss: 6.4964e-04 - a
ccuracy: 1.0000
Epoch 4/5
29640/29640 [==============================] - 145s 5ms/step - loss: 5.5609e-05 - a
ccuracy: 1.0000
Epoch 5/5
29640/29640 [==============================] - 136s 5ms/step - loss: 2.4588e-08 - a
ccuracy: 1.0000
```

Out[27]:

```
<keras.callbacks.History at 0x1dc77c2dd50>
```

In [ ]:

```
accuracy = tf.keras.metrics.Accuracy()

for i in range(2000):
    x = testset[i]
    y = testlabels[i]
    prediction = tf.math.argmax(model(x, training=False), axis=1, output_type=tf.in
    accuracy(prediction, y)

print((accuracy.result()))
```

Results:

Comments:

In [ ]: