Twitter Sentiment Analysis

By Reem Hesham , Rahma Hassan , Ghada Ghanem, Omar Shams , moaaz Al Akel

NLP course .spring 2021

The Idea of the project

Twitter Sentiment Analysis

What is Sentiment Analysis ?

- identify and extract subjective information in source materials by analyzing the text .
- Used Before in a purchase , to know if the product has good or bad reviews

Why Twitter Sentiment Analysis :

In Business: Companies use Twitter Sentiment Analysis to develop their business strategies, to assess customers' feelings towards products or brand, how people respond to their campaigns or product launches and also why consumers are not buying certain products.

Politics: In politics Sentiment Analysis Dataset Twitter is used to keep track of political views, to detect consistency and inconsistency between statements and actions at the government level. Sentiment Analysis Dataset Twitter is also used for analyzing election results.

Dataset

From Kaggle https://www.kaggle.com/kazanova/sentiment140

This is the sentiment140 dataset. It contains 1,600,000 tweets extracted using the twitter api . The tweets have been annotated (0 = negative, 4 = positive) and they can be used to detect sentiment .

Content

It contains the following 6 fields:

1. **target**: the polarity of the tweet (*0* = negative, *2* = neutral, *4* = positive)
2. **ids**: The id of the tweet (*2087*)
3. **date**: the date of the tweet (*Sat May 16 23:58:44 UTC 2009*)
4. **flag**: The query (*lyx*). If there is no query, then this value is NO_QUERY.
5. **user**: the user that tweeted (*robotickilldozr*)
6. **text**: the text of the tweet (*Lyx is cool*)

Approach and preprocessing

- Remove any stop word and repeating words

The classifiers used to solve the problem

- Neural Network
- Zero shot learning model

Data preprocessing steps

- Load dataset.
- Get No. of records.
- Take sample of 100,000 record to deal with.
- Check for any null or missed values in dataset to make sure that it is clean.
- Get the correlation between the features and the target.
- Drop the less correlated features with the target.
- Now data is ready!

```
data.head()
```

	targ	texts
800000	1	I LOVE @Health4UandPets u guys r the best!!
800001	1	im meeting up with one of my besties tonight! ...
800002	1	@DaRealSunisaKim Thanks for the Twitter add, S...
800003	1	Being sick can be really cheap when it hurts t...
800004	1	@LovesBrooklyn2 he has that effect on everyone

Text preprocessing steps

- Downloading stopwords list.
- Remove some useful stopwords from the list.
- Removing any links, mentions or non alphabets from the tweet.
- Replace any emoji to an equivalent word.
- Split the tweet into tokens.
- Lemmatize tweet words.
- Now the tweet is ready to be classified!

```
data['texts']
```

```
800000                                                LOVE guy best
800001    im meeting one besties tonight Cant wait GIRL ...
800002    Thanks Twitter add Sunisa got meet HIN show DC...
800003    Being sick really cheap hurt much eat real foo...
800004                                          effect everyone
                                  ...
49995     insomnia prob slept hr woke 5am nd couldnt go ...
49996     20 mintues late meeting starting know going la...
49997     super excited Are tweeting event happening Onl...
49998     WANT ANOTHER DAY OFF To much Sh today Got quot...
49999                                       jacked umbrella cake
Name: texts, Length: 100000, dtype: object
```

1- Neural Network tensorflow based

- Neural Networks have been found to work well with text data .
- By creating a very simple neural network , we could achieve a sufficient accuracy which reaches % 79 .

Build the model

The layers we decide to choose to build the model :

1- first Embedding layer to convert the tweets to vector its numbers related to each other .

2- LSTM saves the words and predict the next words based on the previous words.

3- use a Dense layer fully connected .

4- using the drop-out to avoid the overfitting

5- and in the output layer we put the sigmoid as we are in the binary classification problem .

Results in case of training all the data

```
print('Training finished !!')
```

```
Epoch 1/10
10080/10080 [==============================] - 1258s 125ms/step - loss: 0.4725 - accuracy: 0.7739 - val_loss: 0.4531 - val_accuracy: 0.7843
Epoch 2/10
10080/10080 [==============================] - 1249s 124ms/step - loss: 0.4512 - accuracy: 0.7870 - val_loss: 0.4430 - val_accuracy: 0.7908
Epoch 3/10
10080/10080 [==============================] - 1253s 124ms/step - loss: 0.4444 - accuracy: 0.7914 - val_loss: 0.4408 - val_accuracy: 0.7927
Epoch 4/10
10080/10080 [==============================] - 1259s 125ms/step - loss: 0.4408 - accuracy: 0.7941 - val_loss: 0.4388 - val_accuracy: 0.7931
Epoch 5/10
10080/10080 [==============================] - 1255s 124ms/step - loss: 0.4381 - accuracy: 0.7961 - val_loss: 0.4380 - val_accuracy: 0.7935
Epoch 6/10
10080/10080 [==============================] - 1250s 124ms/step - loss: 0.4378 - accuracy: 0.7970 - val_loss: 0.4405 - val_accuracy: 0.7944
Epoch 7/10
10080/10080 [==============================] - 1240s 123ms/step - loss: 0.4363 - accuracy: 0.7980 - val_loss: 0.4376 - val_accuracy: 0.7938
Epoch 8/10
10080/10080 [==============================] - 1248s 124ms/step - loss: 0.4525 - accuracy: 0.7917 - val_loss: 0.4734 - val_accuracy: 0.7769
Epoch 9/10
10080/10080 [==============================] - 1258s 125ms/step - loss: 0.4456 - accuracy: 0.7935 - val_loss: 0.4397 - val_accuracy: 0.7954
Epoch 10/10
10080/10080 [==============================] - 1243s 123ms/step - loss: 0.4378 - accuracy: 0.7986 - val_loss: 0.4406 - val_accuracy: 0.7943
Training finished !!
```

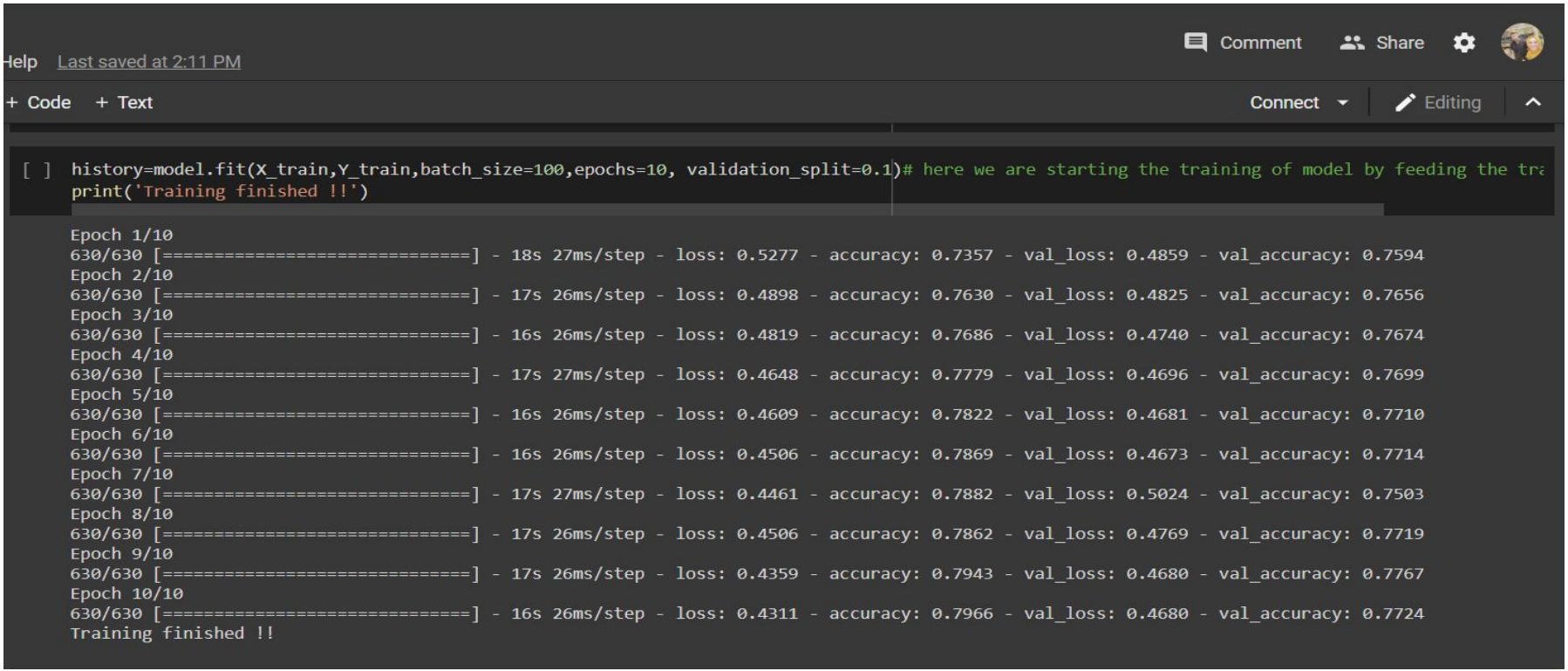
```
[ ] accr1 = model.evaluate(X_test,Y_test) #we are starting to test the model here
```

```
15000/15000 [==============================] - 506s 34ms/step - loss: 0.4429 - accuracy: 0.7938
```

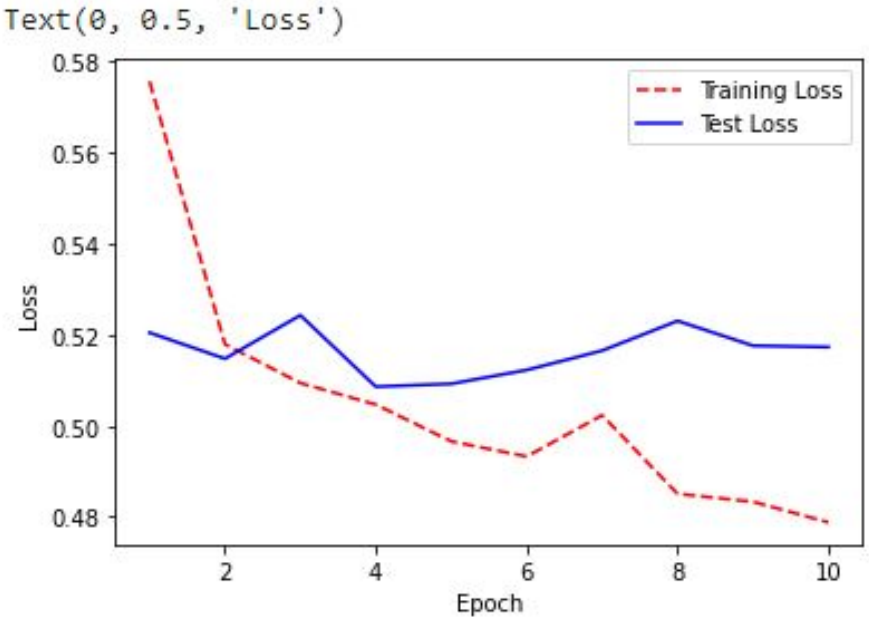
```
print('Test set\n  Accuracy: {:0.2f}'.format(accr1[1])) #the accuracy of the model on test data is given below
```

```
Test set
  Accuracy: 0.79
```

Results in case of training partial of the data



Result analysis



What is Zero Shot Learning Model

-Fancy pretrained **classification** model which can predict unseen object without seeing its label explicitly before.

-But how?

- Using a **detailed description(representation of features of inputs)** of this object, it will be able to predict it with high accuracy.

-Applications in NLP,Computer Vision

Zero shot learning model

-**Pretrained model.**

-State of art who make a progress in classification problems.

- It is a different type of multi-class classifier.

-It is still an active area of research.

Classification Example (NLP APPLICATION):

Can I predict Zebra without feeding my training data with it?

WORD EMBEDDING(Representation of words using vectors):

[1, 1, 0]

[0, 1, 1]

OUR WORK(Twitter Sentiment Analysis) :

-**GOAL**:Applying the idea of this classifier on any text(tweets) to classify the feeling of this tweet whether negative or positive.

-**HOW**:By Feeding the zero-short learning classifier by two main inputs: labels and the data itself.

STEPS OF CODE:

```
data_pos = data[data['targ'] == 1]
data_neg = data[data['targ'] == 0]
```

```
data_pos = data_pos.iloc[:int(50)]
data_neg = data_neg.iloc[:int(50)]
```

```
data = pd.concat([data_pos, data_neg])
```

```
!pip install transformers datasets
from transformers import pipeline
```

```
classifier = pipeline("zero-shot-classification", device=0)
```

Results:

```
labels = ["positive", "negative"]
#Transform texts from data frame to list:
text_list=data_nw['texts'].tolist()
len(text_list)
#Transform targets from data frame to list
classifier(text_list, labels)
```

```
[{'labels': ['positive', 'negative'],
  'scores': [0.9910650253295898, 0.008934964425861835],
  'sequence': 'I LOVE @Health4UandPets u guys r the best!! '},
 {'labels': ['positive', 'negative'],
  'scores': [0.9273909330368042, 0.07260910421609879],
  'sequence': 'im meeting up with one of my besties tonight! Cant wait!!  - GIRL TALK!!'},
 {'labels': ['positive', 'negative'],
  'scores': [0.9575332999229431, 0.04246670752763748],
  'sequence': '@DaRealSunisaKim Thanks for the Twitter add, Sunisa! I got to meet you once at a HIN show here in the DC area
 {'labels': ['negative', 'positive'],
  'scores': [0.8919116258621216, 0.10808838158845901],
  'sequence': 'Being sick can be really cheap when it hurts too much to eat real food  Plus, your friends make you soup'},
```

Advantages of using it:

-It is fancy because of its ability to classify unseen objects/examples.

-It gives high accuracy.

-It is fast because it is a pretrained model.

(Efficient and Effective)

Code

https://colab.research.google.com/drive/1vOhhxvnZHfVKWys8HgP9Szwer1-7a-_I

Thank you