UG - Elevate Final Project Write up

Aakash Madabhushi
A20416408

For this write up, I have chosen a dataset containing tweets with user profile information. I will be using this dataset to train the LSTM model to be able to predict user gender based on twitter profile information. By doing this, we want our model to be able to answer an important question, which is: From the bag of words model, what are the words that strongly predict male or female gender?

This is a form of sentiment classification which is a supervised learning that is trying to model the sentiment of tweets. The term 'supervised' means that we would require a labelled dataset. This dataset would consist of 4 classes (male, gender, brand, unknown).

↓

The end product is to be able to predict the user's gender based on their tweet. The dataset consists of 26 columns..

I will only require only the text of the tweet and the gender label → 2 columns

The dataset used has a lot of missing and irrelevant information, in which we have to clean the data. I have used a pip package that is used to get rid of urls, tags and hashtags as well as lowering all the case of the tweets and removal of stopwords. Stopwords are insignificant words of the sentence such as 'is', 'am', 'at', 'on', etc.

After the cleaning of the data, I will only require 2 columns that is each assigned to x and y variable as shown:

```
x = df['text'].values
y = df['gender'].values
```

Next, the tweets from the dataset gets represented in a bag of words model which is a "way of extracting features from text for use in modelling"

This "representation of text that describes the occurrences of words within a document, which involves two things:
- A vocabulary of known words
- A measure of the presence of known words

Cite source: https://machinelearningmastery.com/gentle-introduction-bag-words-model/

I can still see from our y values (gender) that 97 values are missing and are returned as 'Nan'. In order to eliminate this, a while-loop is built that searches through y, with the use of numpy, and deletes the NaN entries. Also, this deletes the corresponding x entries.

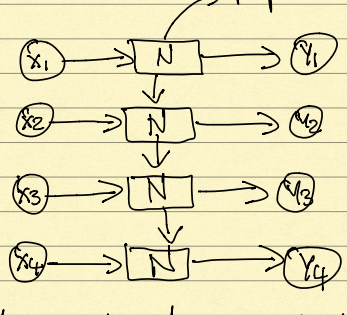
After that, I would have to reshape the y vector in order to fit the train-test split.

Since the gender category has 4 classes (male, female, brand, unknown), ordinal encoder can be used, which will produce a 4 dimension output in an integer form

In the final steps I have to split the corpus (collection of written texts) into a training and testing set. For the testing set, I will use 20% of the dataset.

After that, I will need to pad the sequences of my training and testing data. Before the padding of sequence, I noticed that after representing the model in a bag of words, it outputted a series of numbers in a list. These numbers represents the words used in the text. After padding, the matrix shown is represented at a length of 280 (maximum characters in twitter).

Finally, we use LSTM (Long short-term memory) model to predict user's gender from the tweets. LSTM is one of the branch of a Recurrent Neural Network. This is made for input that should be on a sequence which is why this is a suitable model in working with twitter data. This model can be illustrated in a diagram shown below:



* For this case we must have 4 input length as it represents the 4 classes within the gender category.

* Using an ordinal encoder for the output, it should give 4 integers which each integer represents each gender.

* This method of learning is able to train the twitter data and stores the information for a long time due to that:
  - → It can handle variable length sequences.
  - → It has long term dependencies
  - → It can maintain order information

→ It can share parameters across sequences

* The input layer is an embedding layer that takes in the input matrix and converts in vector form. In the first argument, the max features used is the max number of words that is being used for the model. The second argument is the output dimension of the embedding and the third argument is the input length which is the maximum number of characters that twitter takes.

* The LSTM layer is used in the second layer with a 64 dimension output. This provides a sequence output.

* The output layer is a dense layer that has a 1 dimensional output and has sigmoid activation function

→ Sigmoid function, stays within [0,1] and thus can either let flow of information or not.

* I have chosen RMSPROP (Root mean-squared) as an optimizer for the model. This is useful in modifying gradient descent by normalizing the gradients based on their magnitude.

→ This is a good method for a deep network, especially an LSTM as it has a low memory requirement and it is slightly faster than Adam ('Adaptive momentum')

→ This is recommended more than Adam because of its recurrence relation where the information goes through the model once.

* I have chosen binary cross-entropy as a loss function

$$C(x) = -y \log(\hat{y}) - (1-y) \log(1-\hat{y})$$

→ This is used to validate the testing data by outputting probabilistic values.

→ In each dimension of the encoding, the binary cross entropy is calculated to output the total error. That is why it is important for the data to be vectorized and encoded.

* By fitting the model, I have enforced a batch that takes a subset of 500 data points of the max features. The process of gradient descent is performed on each batch, which is randomly selected. This allows it to add noise to the gradient descent process. This is beneficial as it helps reduce the likelihood of the gradient descent

settling at a local minimum.

⇒ This repeats the process when fitting the model to get a potentially better validation accuracy.

* An epoch of 9 was chosen as you can see on the graph at the bottom, where there is a higher validation accuracy at approximately 45%.

→ During the epoch process, I noticed that at about every 2 epochs, there is a drastic change in the validation accuracy. Also, there has been a fall in the validation loss to −4.6996. This is probably due to that there is a significant amount of Nan values within the gender category.

→ In the testing set, I have gotten an accuracy of approximately 44.9% with the test loss at −4.7.

* I wouldn't use the model in the current form to make predictions as I might require a better or larger dataset to use with effective methods in eliminating the Nan values. I had thought about using something else different from the bag of words model such as TFidf vectorizer which can analyze the co-occurrence and the frequency of words. Also, I had thought of using a baseline model such as a support vector machine with an LSTM in order to help improve the overall performance and reduce the bias.

* Note: Code is attached in a seperate ipynb file