NLP Homework 2
Sentiment Classification

==========

Task description: Try to use Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM) for sentiment classification.

Implementation of Convolution Neural Network

Data-Preprocessing

1. Read the JSON data from the downloaded dataset into chunks of 100
2. Constructed DataFrame from dict of array-like or dicts
3. Convert it into a CSV file
4. Generated sentiment labels are as follows:

   Positive: 1

   Negative: -1

   Neutral: 0
5. Preprocessing of text data
   a. Removal of stop words
   b. Tokenization
   c. Stemming

Embedding

Word embedding is a representation of a word as a numeric vector. Word2vec is based on the idea that a word's meaning is defined by its context. Context is represented as surrounding words. For Embeddings, word2vec pre-trained embeddings are used as input to the network. Word2Vec vectors of size 500 as input. Hence for generating input tensor, Word2Vec vectors is trained with embedding size 500.

Parameter Description

```
count_parameters(cnn_model)
```

```
+----------------+------------+
|    Modules     | Parameters |
+----------------+------------+
| convs.0.weight |    5000    |
|  convs.0.bias  |     10     |
| convs.1.weight |   10000    |
|  convs.1.bias  |     10     |
| convs.2.weight |   15000    |
|  convs.2.bias  |     10     |
| convs.3.weight |   25000    |
|  convs.3.bias  |     10     |
|   fc.weight    |    120     |
|    fc.bias     |     3      |
+----------------+------------+
Total Trainable Params: 55163
55163
```

Time required to run for 1 epoch of CNN: -

```
Epoch1
Epoch ran :1
Input vector
[[  297    42   236   171     0    32     2   173    47   341   789     1
  10518   108   275    62     2  1877   108   282     2   877    36   647
     31  2320  1654   196     1   254     3   201    53     0  1720   166
     31     0   201   991   246    15   263     1   506   467  1189     0
     44    22     0    74     7   219    30  5959  1113  1113  1113  1113
   1113  1113  1113  1113  1113  1113  1113  1113  1113  1113  1113  1113
```

✓ 1m 11s completed at 6:45 PM

Hyperparameters

Epoch - 30

Loss Function - CrossEntropy

Optimizer - Adam

Base Result:

	precision	recall	f1-score	support
0	0.78	0.72	0.75	2992
1	0.62	0.61	0.61	3044
2	0.74	0.82	0.78	2964
accuracy			0.72	9000
macro avg	0.72	0.72	0.71	9000
weighted avg	0.71	0.72	0.71	9000

Ablation Study

CNN	Loss	Accuracy
Tanh	0.7071	72 %
Relu	0. 6977	72.3 %
Sigmoid	0.7213	68%

Implementation of Long Short-Term Memory

Data-Preprocessing

6. Read the JSON data from the downloaded dataset into chunks of 100
7. Constructed DataFrame from dict of array-like or dicts
8. Convert it into a CSV file
9. Generated sentiment labels are as follows:

    Positive: 1

    Negative: -1

    Neutral: 0
10. Preprocessing of text data
    a. Removal of stop words
    b. Tokenization
    c. Stemming

Embedding

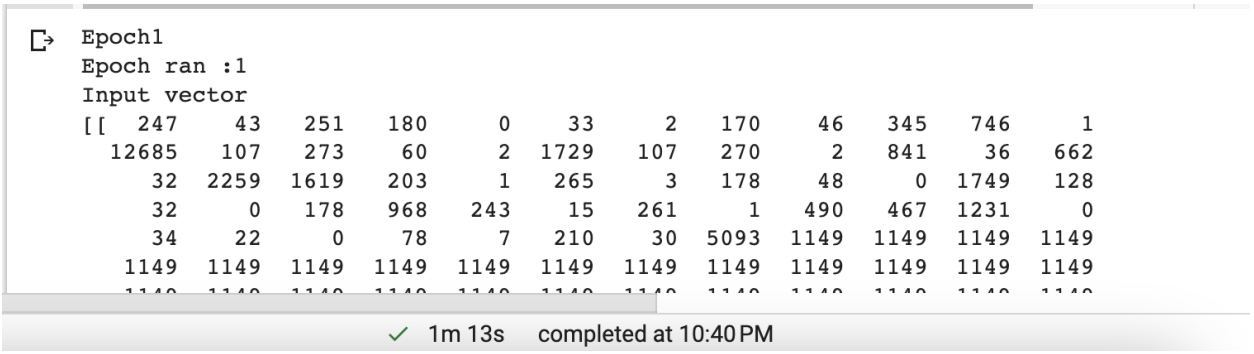
Word embedding is a representation of a word as a numeric vector. Word2vec is based on the idea that a word's meaning is defined by its context. Context is represented as surrounding words. For Embeddings, word2vec pre-trained embeddings are used as input to the network. Word2Vec vectors of size 500 as input. Hence for generating input tensor, Word2Vec vectors is trained with embedding size 500.

Parameter Description

```
count_parameters(lstm_model)
```

```
+-------------------+------------+
|      Modules      | Parameters |
+-------------------+------------+
| lstm.weight_ih_l0 |   64000    |
| lstm.weight_hh_l0 |    4096    |
|  lstm.bias_ih_l0  |    128     |
|  lstm.bias_hh_l0  |    128     |
|     fc.weight     |     64     |
|      fc.bias      |     2      |
+-------------------+------------+
Total Trainable Params: 68418
68418
```

Time required to run for 1 epoch of LSTM:



Hyperparameters

Epoch - 30

Loss Function - BCE_CrossEntropy

Optimizer - Adam

Base Result:

	precision	recall	f1-score
class 0	0.50	1.00	0.67
class 1	0.00	0.00	0.00
class 2	1.00	0.67	0.80
accuracy			0.60
macro avg	0.50	0.56	0.49
weighted avg	0.70	0.60	0.61

Ablation Study

LSTM	Loss	Accuracy
Tanh	0.2189	83 %
Relu	0. 2315	67 %
Sigmoid	0.2643	70 %

Things learned:

- Handling large datasets in pandas data frames through chunking.
- Preprocessing datasets by removing stop words and applying stemming.
- Incorporating GloVe and Word2Vec embeddings.
- Applying CNN and LSTM models for sentiment classification and training.

- Determining that activation functions such as sigmoid and tanh are more appropriate for binary classification tasks where we output only one neuron with a value between 0 and 1 (for softmax) or -1 and 1 (for tanh).
- Noting that positive and negative words have a significant impact on the sentiment value, as observed in the tests.