REPORT 2

Text Classification with Pytorch

I. Group member

- ✓ Nguyễn Huy Hoàng – 20194433 – hoang.nh194433@sis.hust.edu.vn
- ✓ Nguyễn Văn Chiến – 20198281 – chien.nv198281@sis.hust.edu.vn
- ✓ Đặng Thị Hạnh – 20194271 - hanh.dt194271@sis.hust.edu.vn

II. Problem description

Objective: The main goal of Module 2 is to resolve the Sentiment Classification using the PyTorch library (torchtext for specificity).

1. Text classification

- Text classification is a supervised learning problem, which categorized text/ tokens into organized groups (predefined categories). [1]

- Twitter is an American microblogging and social networking service on which users post and interact with messages known as “tweets”. Tweets were originally restricted to 140 characters, but the limit was doubled to 280 for non-CJK languages in November 2017. [2] Recently, people have been discussing about the Covid-19 pandemic (an ongoing global pandemic of coronavirus disease 2019, which killed billions of people around the world). The problem is to classify tweets into sentiment such as “extremely negative”, “negative”, “neutral”, “positive”, and “extremely positive”.

- Input: Representation of the content of a tweet (e.g., a vector of key word weights).
- Output: An assigned label as one of the five sentiments: extremely negative, negative, neutral, positive, extremely positive.

2. Pytorch

- Pytorch is an open source machine learning library based on the Torch library, used for applications such as computer vision and natural language processing, primarily developed by Facebook's AI Research lab (FAIR). [3]

- Pytorch provides two high-level features: [3]

- Tensor computing (like NumPy) with strong acceleration via graphic processing units (GPU).
- Deep neural networks built on a type-based automatic differentiation system.

III. Neural Network Model

1. Recurrent Neural Network (RNN)

- A recurrent neural network (RNN) is a class of artificial neural networks where connections between nodes form a directed or undirected graph along a temporal sequence. This allows it to exhibit temporal dynamic behavior. Derived from feedforward neural networks, RNNs can use their internal state (memory) to process variable-length sequences of inputs. This makes them applicable to tasks such as unsegmented, connected handwriting recognition or speech recognition. Recurrent neural networks are theoretically Turing complete and can run arbitrary programs to process arbitrary sequences of inputs. [4]

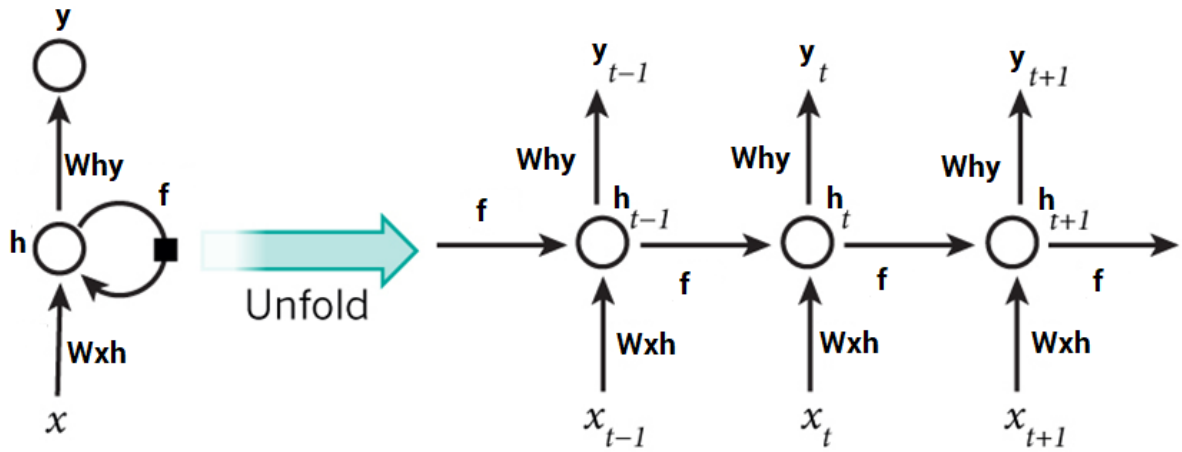


Figure 1: Recurrent Neural Networks
Source: https://iq.opengenus.org/when-to-use-recurrent-neural-networks-rnn/

Where,

$$h_t = f_W(x_t, h_{t-1})$$

h_t : new state

f_W() : function of W (can be tanh or RELU)

x_t : input at time *t*

h_{t-1} : old state

y_t : output at time t

2. Long Short Term Memory (LSTM)

- Long short-term memory (LSTM) is an artificial recurrent neural network (RNN) architecture used in the field of deep learning. Unlike standard feedforward neural networks, LSTM has feedback connections. It can process not only single data points (such as images), but also entire sequences of data (such as speech or video). [5]

- A common LSTM unit is composed of a cell, an input gate, an output gate and a forget gate. The cell remembers values over arbitrary time intervals and the three gates regulate the flow of information into and out of the cell. [5]

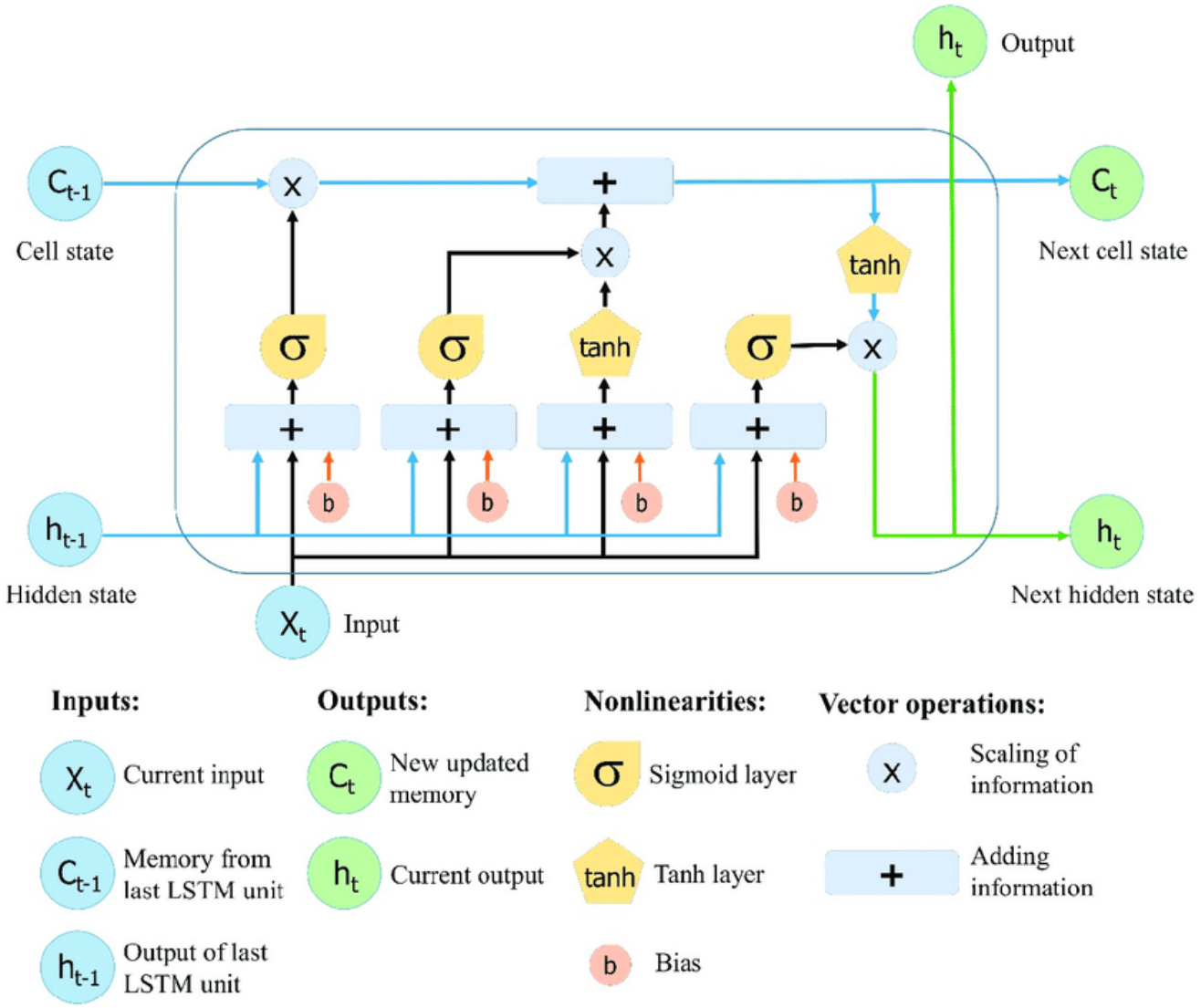


Figure 2: Long short term memory
Source: https://www.researchgate.net/figure/The-structure-of-the-Long-Short-Term-Memory-LSTM-neural-network-Reproduced-from-Yan_fig8_334268507

3. *Modelling*

a. RNN

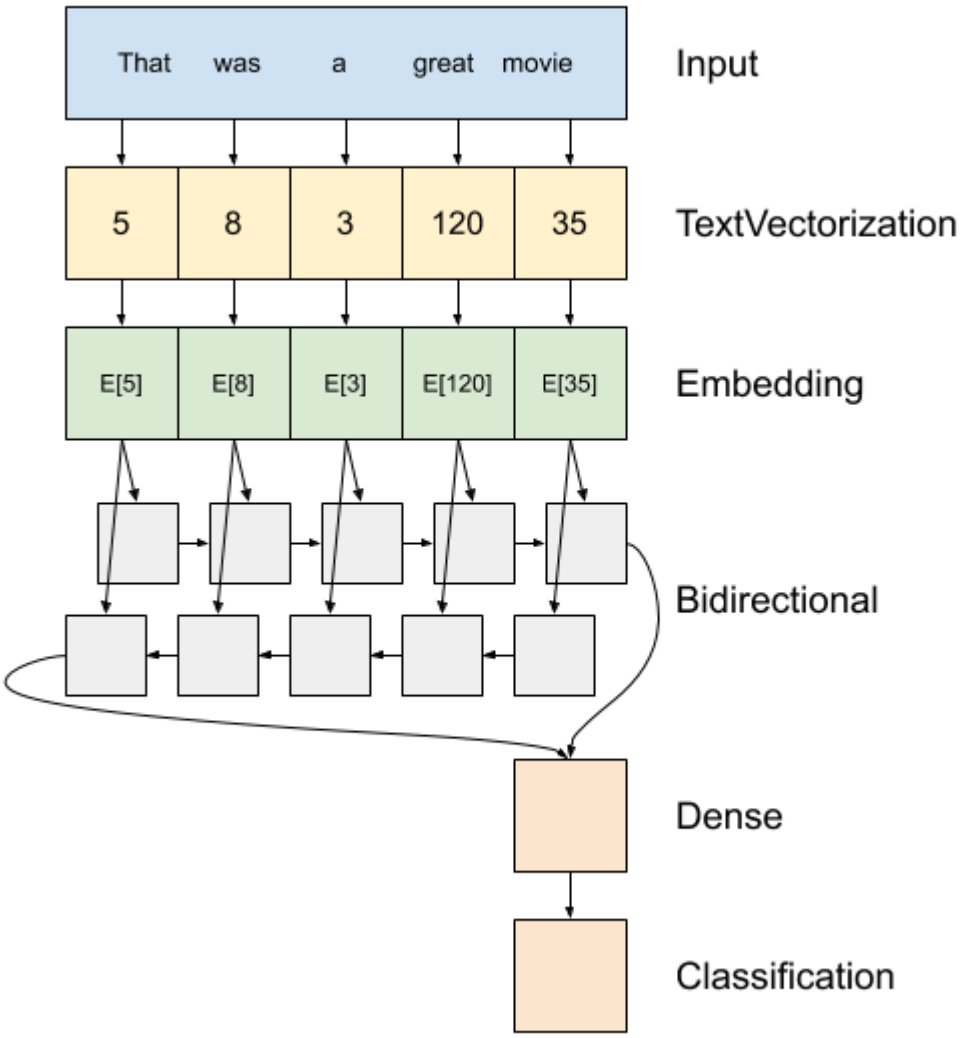


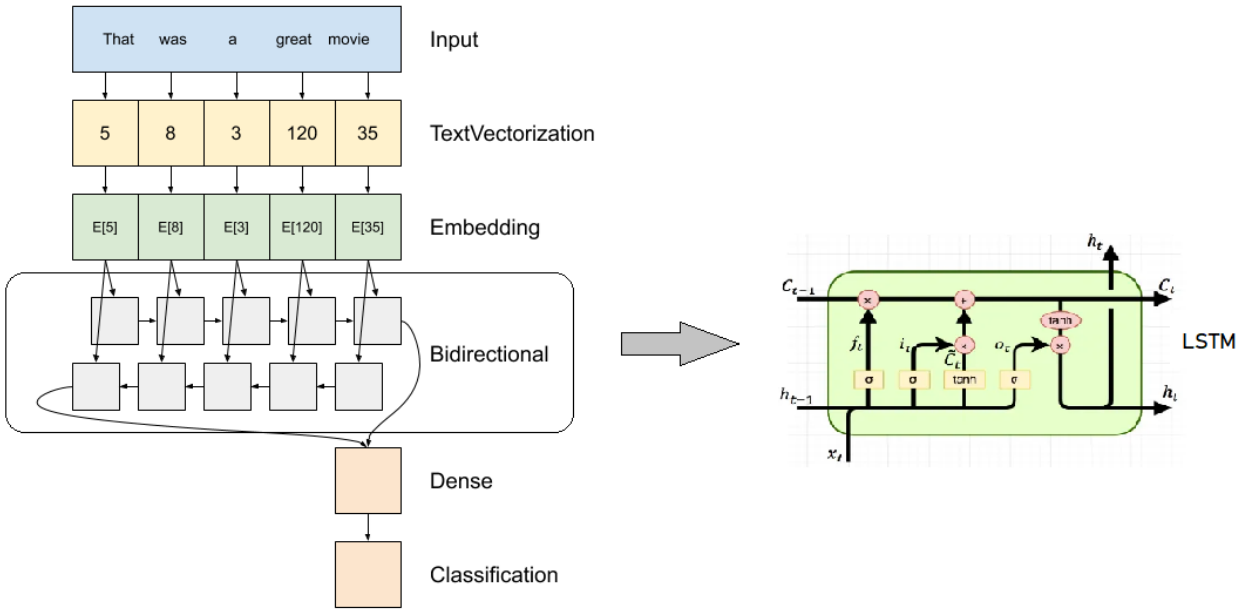
Figure 3: RNN model
Source: https://www.tensorflow.org/text/tutorials/text_classification_rnn

Above is the diagram of the model: [6]

- The first layer is the encoder, which converts the text to a sequence of token indices by building a vocabulary with each word attached to an index.
- After the encoder is an embedding layer. An embedding layer stores one vector per word. When called, it converts the sequences of word indices to sequences of vectors. These vectors are trainable. After training (on enough data), words with similar meanings often have similar vectors.

- A recurrent neural network (RNN) processes sequence input by iterating through the elements. RNNs pass the outputs from one timestep to their input on the next timestep. This propagates the input forward and backwards through the RNN layer and then concatenates the final output.
- After the RNN has converted the sequence to a single vector the Dense (Linear) layer to do some final processing, and convert from this vector representation to a single logit as the classification output.

b. LSTM



LSTM model is quiet similar to the RNN model above. The online different is the bidirectional layer in RNN model, this layer is replaced by an LSTM layer.

4. *Dataset: Coronavirus Tweets*

   Source: https://www.kaggle.com/datatattle/covid-19-nlp-text-classification

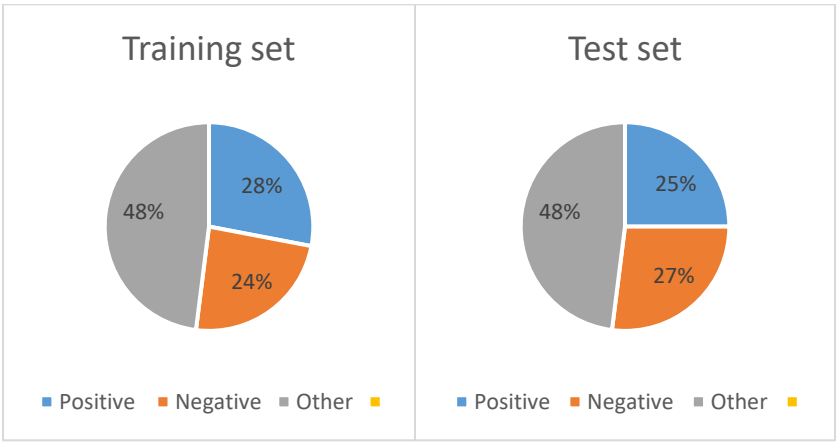
   - The tweets have been pulled from Twitter and manual tagging has been done then. The names and user name have been given codes to avoid any privacy concerns.
   - Language: English
   - Columns:

     - User name (encoded)

- Screen name (encoded)
- Location
- Tweet at
- Original Tweet
- Label (Sentiment)

- Number of classes: 5

- Extremely Negative
- Negative
- Neutral
- Positive
- Extremely Positive



- Training set: 41157 unique samples

- Test set: 3798 unique samples

5. Experiment

- Data Loading: Load data from 'csv' files (pandas)

- Parameterize dataset: Convert sentiment classes into integer classes (pandas)

+ Method 1: Original Sentiment classes: Extremely Negative, Negative, Neutral, Positive, Extremely Positive

- Extremely Negative : -2
- Negative : -1
- Neutral: 0
- Positive : 1
- Extremely Positive: 2

+ Method 2: Three-sentiment classes: Combine ‘Extremely Negative’ and ‘Negative’, ‘Extremely Positive’ and ‘Positive’

- Extremely Negative and Negative : -1
- Neutral : 0
- Extremely Positive and Positive : 1

- Data preprocessing: Cleaning the data (Natural Language Toolkit (nltk), Regular expression operation (re), pandas)

- Remove URL link
- Remove mention (@Henry, @Jenny,…)
- Remove hastags (#COVID-19, #CoronaVirus,…)
- Remove html tags
- Replace punctuation

→ Write cleaned data into new csv files.

- Generate data batch and iterator: Field, LabelField, TabularDataset, BuckerIterator from torchtext.legacy.data

- Define fields
- Load data using TabularDataset
- Build vocabulary

- Generate batch and iterator using BuckerIterator

- Define the model: torch.nn

The model will be built based on the network-layer graph above.

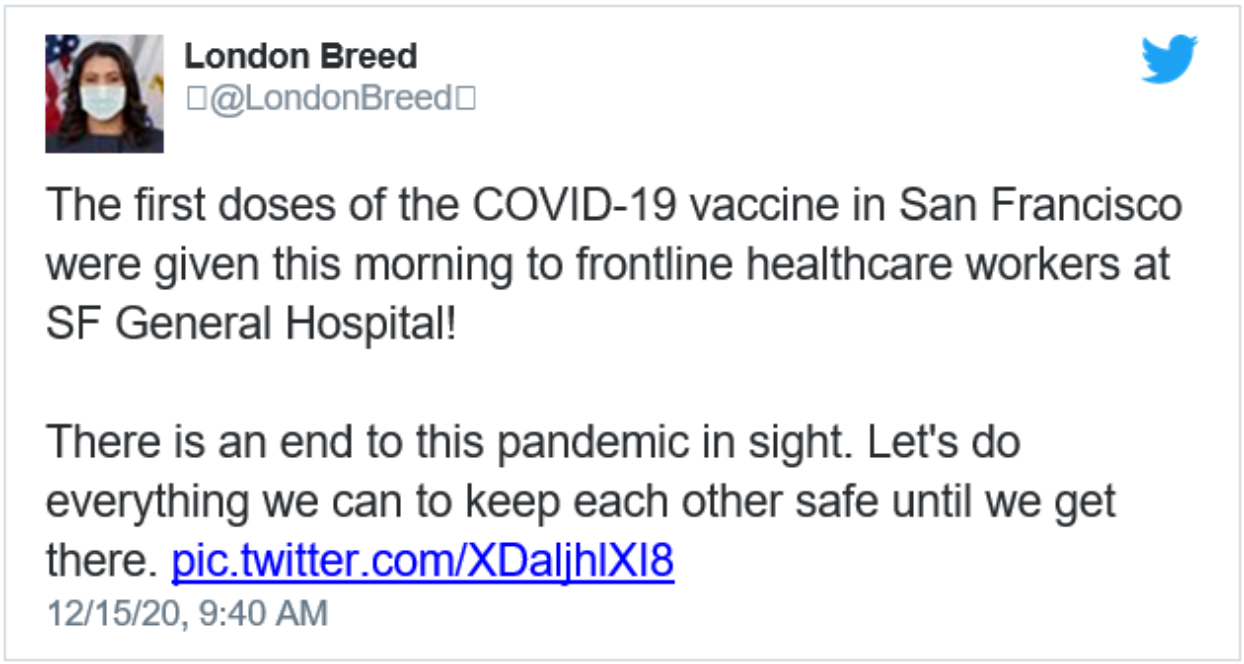
- Training and testing model on Test set:

- "with torch.no_grad()" is like a loop where every tensor inside the loop will have requires_grad set to False.
- Optimizer: Adam
- Loss function: Cross Entropy

- Predict sentiment of a random tweet about COVID-19: spacy

+ Tweet 1: from Donald Trump

+ Tweet 2: London Breed



+ Tweet 3: We create this sentence by ourselves. Just imagine some day in the future the pandemic will be over:

"The COVID-19 pandemic is finally over"

6. Result

a. RNN

+ Method 1: Original Sentiment classes: Extremely Negative, Negative, Neutral, Positive, Extremely Positive

Epoch	Time elapsed	Training accuracy(%)	Valid accuracy(%)
1	1.26'	27.87	27.85
2	2.54'	27.91	27.86
3	3.74'	28.00	27.80
4	4.90'	27.85	27.85
5	5.96'	27.66	27.54
6	6.92'	24.69	24.06
7	7.87'	27.88	27.81

8	8.81’	27.90	27.68
9	9.75’	25.58	24.92
10	10.70’	27.54	27.31
11	11.65’	27.81	27.40
12	12.60’	28.14	27.71
13	13.55’	27.96	27.37
14	14.50’	28.02	27.41
15	15.46’	28.20	27.69

Total Training Time: 15.46 min

Test accuracy: 25.20%

*Test tweet

	Tweet 1	Tweet 2	Tweet 3
Extremely Negative	0.005347067024558783	0.008389783091843128	0.01231339480727911
Negative	0.4791475534439087	0.973167896270752	0.6364898681640625
Neutral	0.2643568217754364	0.0009804184082895517	0.020838269963860512
Positive	0.24754323065280914	0.016476236283779144	0.32517752051353455
Extremely Positive	0.0036052654031664133	0.0009856411488726735	0.005180956330150366
Final Sentiment	Negative	Negative	Negative

+ Method 2: Three-sentiment classes: Combine ‘Extremely Negative’ and ‘Negative’, ‘Extremely Positive’ and ‘Positive’

Epoch	Time elapsed	Training accuracy(%)	Valid accuracy(%)
1	1.28’	37.88	26.60
2	2.49’	43.88	44.13
3	3.65’	43.76	44.06
4	4.75’	43.85	44.18
5	5.78’	43.88	44.24

6	6.77’	43.89	44.11
7	7.76’	43.99	44.15
8	8.76’	44.00	44.11
9	9.79’	42.99	42.98
10	10.78’	44.12	44.27
11	11.77’	44.01	44.28
12	12.76’	44.16	44.39
13	13.77’	44.23	44.24
14	14.78’	44.04	44.15
15	15.79’	37.71	36.54

Total Training Time: 15.79 min

Test accuracy: 42.68%

*Test tweet

	Negative	Neutral	Positive	Final Sentiment
Tweet 1	2.485761433929784e-13	0.516394317150116	0.4544287323951721	Neutral
Tweet 2	3.1402910281030927e-06	0.4221319258213043	0.5380465984344482	Positive
Tweet 3	7.58814485379844e-06	0.40283215045928955	0.45040223002433777	Positive

b. LSTM

Original Sentiment classes: Extremely Negative, Negative, Neutral, Positive, Extremely Positive

Epoch	Time elapsed	Training accuracy(%)	Valid accuracy(%)
1	1.89’	44.44	41.66
2	3.74’	70.06	60.30
3	5.59’	82.79	63.83
4	7.42’	89.53	65.98

5	9.29’	93.09	65.75
6	11.15’	95.44	66.79
7	13.03’	96.34	65.96
8	14.91’	97.39	66.70
9	16.77’	97.79	64.92
10	18.63’	98.24	65.23
11	20.47’	98.32	65.63
12	22.33’	98.59	65.52
13	24.20’	98.74	65.29
14	26.06’	98.99	65.16
15	27.93’	99.19	64.71

Total Training Time: 27.93 min

Test accuracy: 62.35%

*Test tweet

	Tweet 1	Tweet 2	Tweet 3
Extremely Negative	0.6944298148155212	0.36836737394332886	0.03578733280301094
Negative	0.00010300070425728336	0.007209158502519131	0.1893978863954544
Neutral	0.3023918867111206	0.5560702681541443	0.18501830101013184
Positive	0.0030550609808415174	0.0580272413790226	0.11182550340890884
Extremely Positive	2.0227953427820466e-05	0.010325943119823933	0.4779709279537201
Final Sentiment	Extremely Negative	Neutral	Extremely Positive

7. Conclusion

Through the results, we can see that the performance of the RNN model is very low, the test accuracy is only 20.25%. Even when we reduce the sentiment class to 3 classes, the efficiency only increases to 42.68%. Therefore, predicting sentiment of random tweets is not accurate. However, the LSTM model proved to be quite effective when test accuracy is

up to 62.35% while keeping the same 5 sentiment classes. This model also predicts random tweets relatively accurately.

Reasons for the inefficiencies of the RNN model may due to the “vanishing gradient” problem when using Back propagation through time, which can be used up to a limited number of time steps like 8 or 10. [7] However, this problem was solved when replacing the bidirectional layer with a LSTM layer. This is because LSTMs solve the problem using a unique additive gradient structure that includes direct access to the forget gate's activations, enabling the network to encourage desired behavior from the error gradient using frequent gates update on every time step of the learning process. [8]

References

[1] M. Learn, "Text Classification with Machine Learning & NLP," [Online]. Available: https://monkeylearn.com/text-classification/. [Accessed 10 2021].

[2] Wikipedia, "Twitter".

[3] Wikipedia, "PyTorch," [Online]. Available: https://en.wikipedia.org/wiki/PyTorch. [Accessed 11 2021].

[4] Wikipedia, "Recurrent neural network," [Online]. Available: https://en.wikipedia.org/wiki/Recurrent_neural_network. [Accessed 11 2021].

[5] Wikipedia, "Long short-term memory," [Online]. Available: https://en.wikipedia.org/wiki/Long_short-term_memory#:~:text=Long+short%2Dterm+memory+(LSTM)+is+an+artificial+recurrent,the+field+of+deep+learning.&text=LSTM+networks+are+well%2Dsuited,events+in+a+time+series.. [Accessed 11 2021].

[6] TendorFlow, "Text classification with an RNN," [Online]. Available: https://www.tensorflow.org/text/tutorials/text_classification_rnn. [Accessed 11 2021].

[7] GeeksforGeeks, "Back Propagation through time – RNN," [Online]. Available: https://www.geeksforgeeks.org/ml-back-propagation-through-time/. [Accessed 11 2021].

[8] N. Arbel, "How LSTM networks solve the problem of vanishing gradients," [Online]. Available: https://medium.datadriveninvestor.com/how-do-lstm-networks-solve-the-problem-of-vanishing-gradients-a6784971a577. [Accessed 11 2021].