Russia-Ukraine War Twitter Sentiment Analysis

Julia Ivaniv
CS 584
Illinois Institute of Technology
A20418282

Abstract

This project aims to explore the sentiment of unlabeled Russia-Ukraine Twitter data using transfer learning. There are other methods that could be used for this problem such as existing models BERT and GPT-2, but this project explores using a custom RNN. An RNN LSTM model was fit to labeled Twitter data and then used to make predictions on the unlabeled Russia-Ukraine Twitter data. The model accuracy during training was acceptable, but the loss was high for both the training and test data. This RNN model was compared to an existing sentiment analyzer called VADER. All code was written in Python and uploaded to GitHub.

1. Introduction

Recurrent neural networks (RNNs) are the ideal type of neural network for data that is in the form of text. This is because text is a type of sequential data. RNNs are able to predict what comes next in sequential data by using layers that mimic short-term memory [11]. RNNs can be used for a machine-learning problem called sentiment analysis. Sentiment analysis involves predicting the sentiment of a piece of text. Sentiment analysis is also known as opinion mining [4]. Companies may use sentiment analysis to improve their brand based on customer feedback. This report will walk through a project that uses sentiment analysis to evaluate the public's opinion on the current Russia-Ukraine war using two different models.

1.1. Problem

In times of conflict, it can be interesting to evaluate what the public is thinking and who they are siding with. That is why this project will focus on analyzing the sentiment of Tweets written about the Russia-Ukraine war. The problem is that all of the Russia-Ukraine war Twitter data is unlabeled. Unlabeled data can be difficult to work with because of limited methods to evaluate the accuracy of models that are used to make predictions on unlabeled data. This project will focus on the use of transfer learning. Transfer learning is a method that describes when a model created for one problem can be reused for a different problem [7]. An RNN model will be created and pre-trained with existing labeled Twitter data. It makes sense to use Twitter data because it should act similarly to the unlabeled Russia-Ukraine war Twitter data. Then, this model will be used to make predictions about the unlabeled data. Although the accuracy of the model using the labeled data may not be the same as the accuracy of that same model using the unlabeled data, it is still an interesting project to explore. For further evaluation, an existing sentiment analysis model called VADER will be used on the labeled and unlabeled data.

1.2. Previous Work

There are many different ways to approach unsupervised sentiment analysis. These include the lexicon-based approach, clustering-based approach, deep learning-based approach, and rule-based approach. Another way is to use transfer learning. This involves pre-training a model on a labeled data set and then fine-tuning it to a similar unlabeled data set. There are existing models that have been used for this type of learning. Some examples are BERT and GPT-2. BERT stands for Bidirectional Encoder Representations from Transformers. This model is known for using bidirectional training instead of being limited to evaluating text sequences in only one direction [8]. GPT-2 is a very large model with 1.5 billion parameters trained on data in websites [10]. These models are great to use because of their extensive training and newer and better features. Although there are great existing models, this project will explore creating a custom RNN model.

1.3. Concepts

An RNN model was chosen for this project because of its ability to process sequential data well. An input enters the model at a specific time. The input is usually a word of a sentence in this application. Next, there is a hidden layer that acts like a memory. This layer takes into account the previous step's hidden state [9]. Weights are then applied to the connections. Finally, the output is calculated. Figure 1 shows a simple diagram of an RNN and its forward pass formulas. An RNN can have different unique layers.

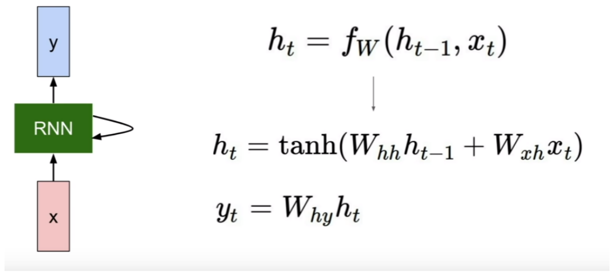

y
RNN
x
$h_t = f_W(h_{t-1}, x_t)$
$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$
$y_t = W_{hy}h_t$

Figure 1. The structure and formulas of a simple RNN.

One of them is called the long short-term memory network (LSTM). This network involves a cell state that has gates that control which inputs and outputs are relevant or not [6]. A diagram of an LSTM network is shown in Figure 2 below. The RNN model that will be used for training will include

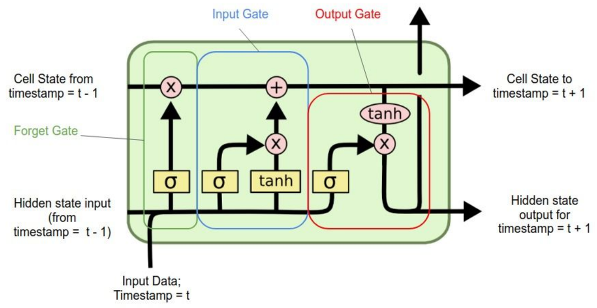

Input Gate
Output Gate
Cell State from timestamp = t - 1
Cell State to timestamp = t + 1
Forget Gate
tanh
σ
σ
tanh
σ
Hidden state input (from timestamp = t - 1)
Hidden state output for timestamp = t + 1
Input Data; Timestamp = t

Figure 2. The structure of an LSTM layer. Photo aquired from Medium.

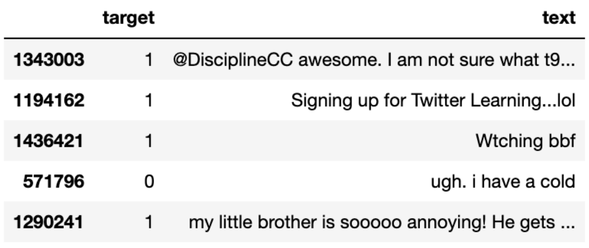
an LSTM layer to increase its capabilities and accuracy.

2. Methods

Sentiment analysis involves Natural Language Processing (NLP). To achieve this, an existing Twitter data set from Kaggle was used to train an LSTM model. The data set is composed of 1.6 million tweets that are labeled negative, neutral, or positive. The LSTM model was then fit to the unlabeled Russia-Ukraine Twitter data. This model was then compared to an existing sentiment analyzer from the NLTK Python library called the SentimentIntensityAnalyzer. This is VADER Sentiment Analysis, where VADER means Valence Aware Dictionary and sEntiment Reasoner.

2.1. Data Pre-Processing

NLP involves a lot of data preparation. First, the label column and the text column of the labeled Twitter data set were isolated. Next, the rows with a neutral (2) label were removed from the data set. This project solely focuses on binary classification. Since the positive texts were labeled with a 4, this value needed to be replaced with 1. The data was ordered by label, so a function was used to shuffle the data. The only labels in the data set from this point are 1 (positive) and 0 (negative). For the sake of time, 10,000 data points were chosen for analysis. The raw data is shown in Figure 3. X was set to the 'text' column and y was set



	target	text
1343003	1	@DisciplineCC awesome. I am not sure what t9...
1194162	1	Signing up for Twitter Learning...lol
1436421	1	Wtching bbf
571796	0	ugh. i have a cold
1290241	1	my little brother is sooooo annoying! He gets ...

Figure 3. The first five rows of the label column and the text column from the Kaggle labeled Twitter data set.

to the 'target' column. The y value was reformatted so that after the X value was pre-processed they would both be of the same format for training and testing purposes. Next, the data was checked for any blank rows. This was done for all of the data sets and none of them had blank rows. In addition, it was important to find out if the categories in the data were skewed. A function was called to count the number of positive and negative labels in the data as shown in Figure 4. The is pretty even between positive and negative sentiment.

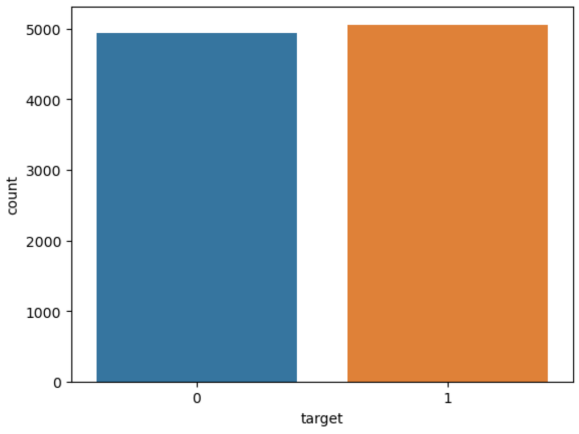
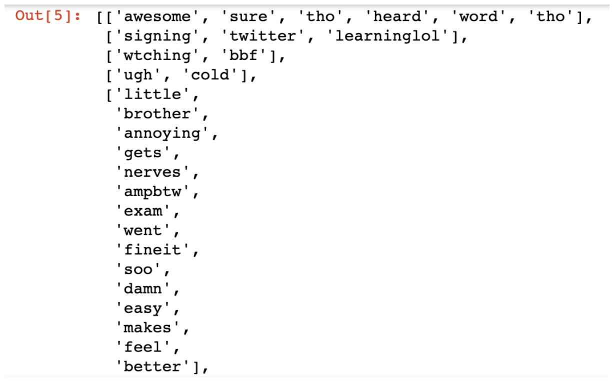


Figure 4. A plot of the count of each sentiment in the labeled Twitter data set. The data is not skewed in either direction.

The data needed to be cleaned. This was done by using regex commands. Each tweet was changed to lowercase. Then, all mentions, links, extra white spaces, special characters, numbers, duplicate words, letters that repeated more than two times, and stop words were removed. The tweets were tokenized by being split into individual words as shown in Figure 5. Next, the words in each tweet were



```
Out[5]: [['awesome', 'sure', 'tho', 'heard', 'word', 'tho'],
         ['signing', 'twitter', 'learninglol'],
         ['wtching', 'bbf'],
         ['ugh', 'cold'],
         ['little',
          'brother',
          'annoying',
          'gets',
          'nerves',
          'ampbtw',
          'exam',
          'went',
          'fineit',
          'soo',
          'damn',
          'easy',
          'makes',
          'feel',
          'better'],
```

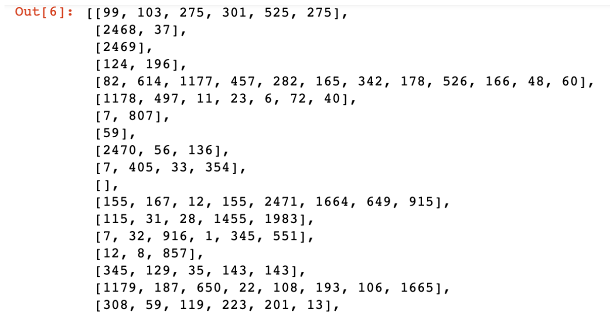
Figure 5. Text data from the labeled Twitter data set tokenized by being split into individual words.

mapped to an index. This created a sequence of numbers. These numbers make up a unique dictionary. This dictionary's length was limited to 3000 words. An example of the sequences is shown in Figure 6.

Next, the maximum and minimum lengths of the list of sequences were calculated. This is important for the preparation of training the LSTM model. Each sequence that is passed through the model needs to be of the same length. A length was chosen based on the maximum and minimum values, but it usually ranged around 20. Since not all of the sequences in the list were this long, they needed to be padded. This means that any sequences below the length of 20 would get as many zeros appended to their sequence as needed. These act as null values and do not influence the sentiment. An example of the padding is shown in Figure 7.

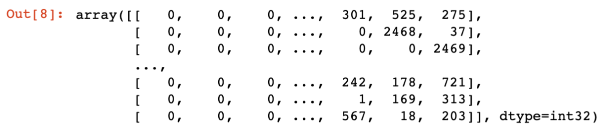
2.2. Building the Model

The model was created with a few important layers. The first layer is the embedding layer. This layer turns the vectorized input into a dense vector of fixed size [2]. The



```
Out[6]: [[99, 103, 275, 301, 525, 275],
         [2468, 37],
         [2469],
         [124, 196],
         [82, 614, 1177, 457, 282, 165, 342, 178, 526, 166, 48, 60],
         [1178, 497, 11, 23, 6, 72, 40],
         [7, 807],
         [59],
         [2470, 56, 136],
         [7, 405, 33, 354],
         [],
         [155, 167, 12, 155, 2471, 1664, 649, 915],
         [115, 31, 28, 1455, 1983],
         [7, 32, 916, 1, 345, 551],
         [12, 8, 857],
         [345, 129, 35, 143, 143],
         [1179, 187, 650, 22, 108, 193, 106, 1665],
         [308, 59, 119, 223, 201, 13],
```

Figure 6. Mapped sequence data from the labeled Twitter data set.



```
Out[8]: array([[   0,    0,    0, ...,  301,  525,  275],
               [   0,    0,    0, ...,    0, 2468,   37],
               [   0,    0,    0, ...,    0,    0, 2469],
               ...,
               [   0,    0,    0, ...,  242,  178,  721],
               [   0,    0,    0, ...,    1,  169,  313],
               [   0,    0,    0, ...,  567,   18,  203]], dtype=int32)
```

Figure 7. An example of the padded X data.

next layer is the LSTM layer. As mentioned before this layer chooses what to remember and what to forget in a sequence. After the LSTM layer, a dense layer was added. All of this layer's neurons accept the output from the previous layer [3]. In addition, dropout layers were added to help prevent overfitting [1]. Both the embedding layer and the LSTM layers had a low number of units to try and prevent from overfitting, too. The final model looked very similar to the diagram shown in Figure 8.

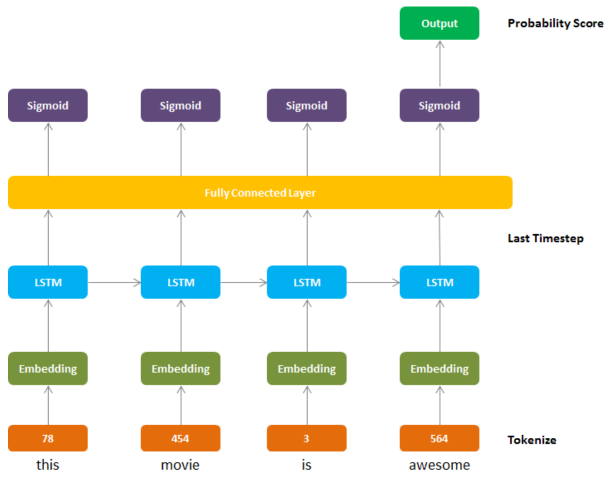


Figure 8. An example of the layers in a simple RNN model with an LSTM layer. Acquired from Towards Data Science.

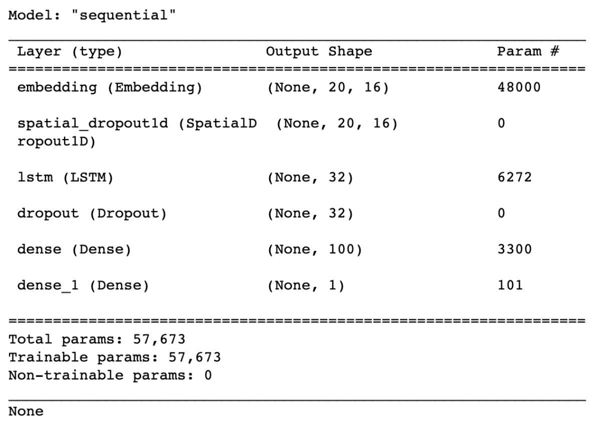
The layers were added to the model and final number of parameters was 57,673 as shown in Figure 9. The model was compiled with binary cross entropy loss because this is a classification problem. The optimizer picked for this model was Adam.

2.3. Model Training and Testing

In order to train the model the X and y values were split into train and test data. 80% of the data was used for training and 20% was used for testing. The model was fit to this data with a batch_size of 50 and an epoch size of 8.

2.4. Results: RNN LSTM Model

After the model was fitted to the data an accuracy plot and loss plot were created for both the training and test data. These plots are shown in Figures 10 and 11.



```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 20, 16)            48000

 spatial_dropout1d (SpatialD  (None, 20, 16)           0
 ropout1D)

 lstm (LSTM)                 (None, 32)                6272

 dropout (Dropout)           (None, 32)                0

 dense (Dense)               (None, 100)               3300

 dense_1 (Dense)             (None, 1)                 101

=================================================================
Total params: 57,673
Trainable params: 57,673
Non-trainable params: 0
_________________________________________________________________
None
```

Figure 9. Custom RNN model showing the different layers and the number of parameters.

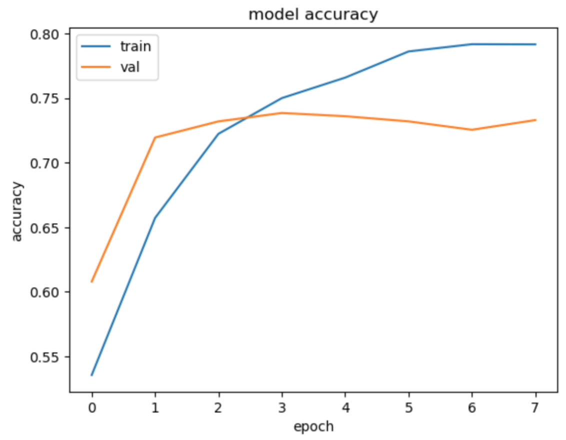


Figure 10. The custom RNN model accuracy vs epochs plot for train and test data.

The accuracy of the model is about 73%. As shown in the figures above the model loss is really high for both the train and test data. These plots can be compared to Figures 12 and 13. These figures show what different losses and accuracy can say about the RNN.

A confusion matrix was created from the Sequential model's predictions as shown in Figure 14.

This model was then fit to three different sets of Twitter data about the Russia-Ukraine war. The first set contained tweets from April 2022. The second set contained tweets from September 2022. The last set contained tweets from March 2023. These dates were specifically chosen because notable events concerning the war occurred during each one of these months. The data was pre-processed the same way as the labeled Twitter data. Since this data is unlabeled, though, the accuracy was not verified. The model was fit to the data and the predictions were outputted in the form of counts of positive and negative sentiment. A visual of how

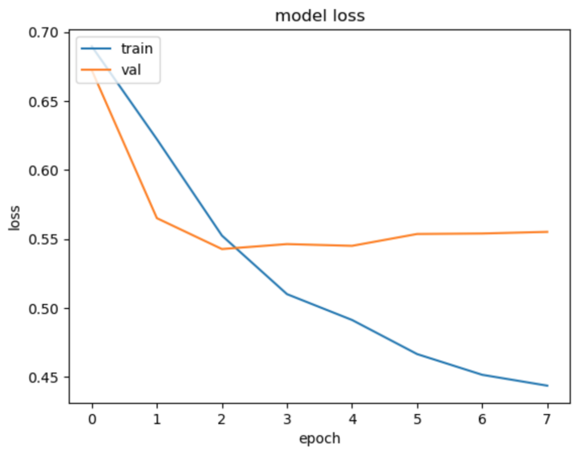


Figure 11. The custom RNN model loss vs epochs plot for train and test data.

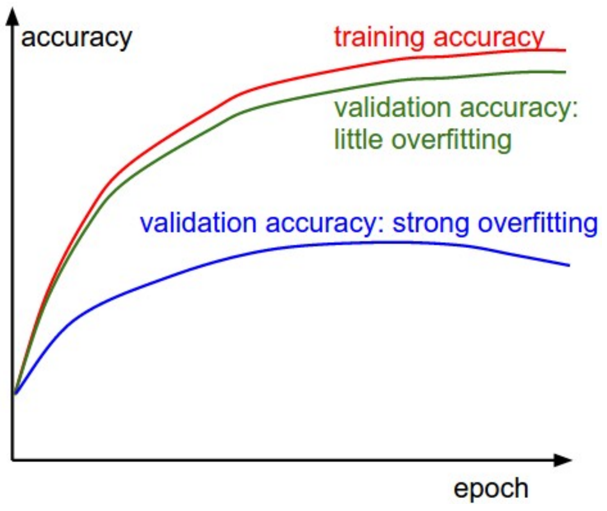


Figure 12. An accuracy vs epoch plot to help diagnose issues in neural network model. Acquired from Towards Data Science.

the data was analyzed is shown in Figure 15. For the April data, the model predicted 4,362 positive records and 5,638 negative records for a particular run. For the September data, the model predicted 4,474 positive records and 5,526 negative records. For the March data, the model predicted 4,443 positive records and 5,557 negative records.

2.5. Results: VADER Model

VADER is a sentiment analysis tool that uses a large vocabulary from words used in media to predict the polarity of its input [5]. This tool output four scores: a positive score, a negative score, a neutral score, and a combined score. This tool was imported into Python and used on the labeled Twitter data to predict the sentiment. The compound score from the sentiment analysis was evaluated and the text was given

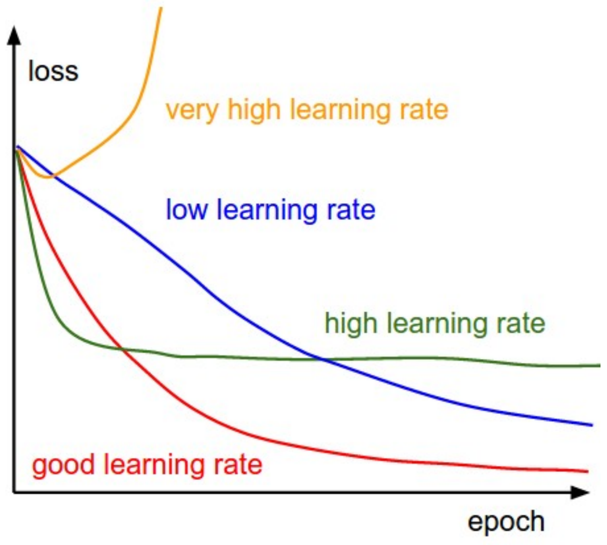


Figure 13. An accuracy vs epoch plot to help diagnose issues in neural network model. Acquired from Towards Data Science.

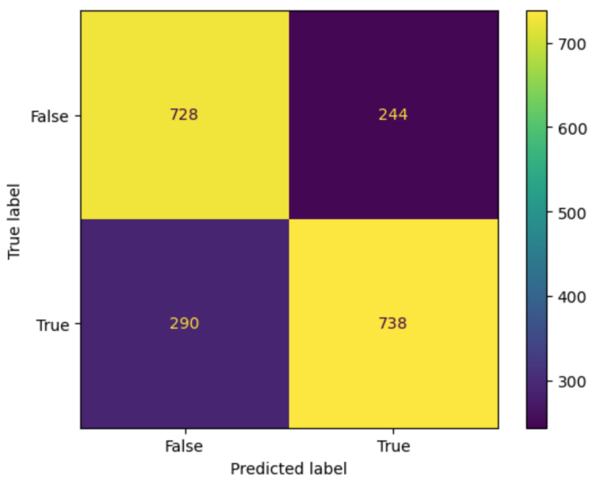
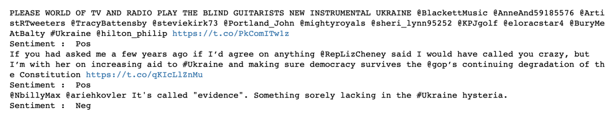


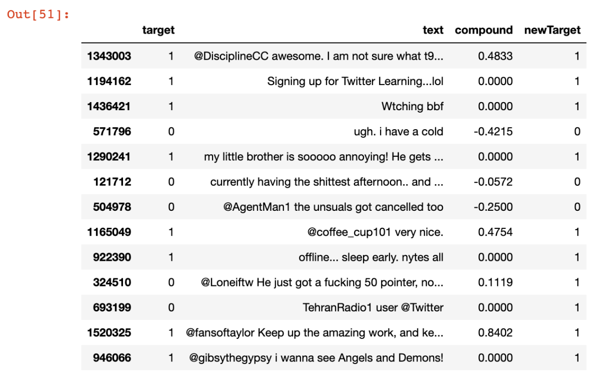
Figure 14. Confusion matrix for the RNN model's predictions.



```
PLEASE WORLD OF TV AND RADIO PLAY THE BLIND GUITARISTS NEW INSTRUMENTAL UKRAINE @BlackettMusic @AnneAnd59185576 @Arti
stRTweeters @TracyBattensby @steviekirk73 @Portland_John @mightyroyals @sheri_lynn95252 @KPJgolf @eloracstar4 @BuryMe
AtBalty #Ukraine @hilton_philip https://t.co/PkComITw1z
Sentiment :  Pos
If you had asked me a few years ago if I'd agree on anything @RepLizCheney said I would have called you crazy, but
I'm with her on increasing aid to #Ukraine and making sure democracy survives the @gop's continuing degradation of th
e Constitution https://t.co/qKIcLlZnMu
Sentiment :  Pos
@NbillyMax @ariehkovler It's called "evidence". Something sorely lacking in the #Ukraine hysteria.
Sentiment :  Neg
```

Figure 15. A sample of the labeled Russia-Ukraine war Twitter data using the custom RNN model.

a score of 0 if the compound score was negative and a score of 1 if the compound score was positive. An example of this is shown in Figure 16. A confusion matrix was created based on the results from above as shown in Figure 17. This model was then fit to the unlabeled Russia-Ukraine war data. For the April data, the model predicted 5,562 positive records and 4,438 negative records for a particular run. For the September data, the model predicted 6,414 positive records and 3,586 negative records. For the March data, the model predicted 5,608 positive records and 4,392 negative records.



Out[51]:

	target	text	compound	newTarget
1343003	1	@DisciplineCC awesome. I am not sure what t9...	0.4833	1
1194162	1	Signing up for Twitter Learning...lol	0.0000	1
1436421	1	Wtching bbf	0.0000	1
571796	0	ugh. i have a cold	-0.4215	0
1290241	1	my little brother is sooooo annoying! He gets ...	0.0000	1
121712	0	currently having the shittest afternoon.. and ...	-0.0572	0
504978	0	@AgentMan1 the unsuals got cancelled too	-0.2500	0
1165049	1	@coffee_cup101 very nice.	0.4754	1
922390	1	offline... sleep early. nytes all	0.0000	1
324510	0	@Loneiftw He just got a fucking 50 pointer, no...	0.1119	1
693199	0	TehranRadio1 user @Twitter	0.0000	1
1520325	1	@fansoftaylor Keep up the amazing work, and ke...	0.8402	1
946066	1	@gibsythegypsy i wanna see Angels and Demons!	0.0000	1

Figure 16. Labeled Twitter data with new prediction column from VADER model's predictions.

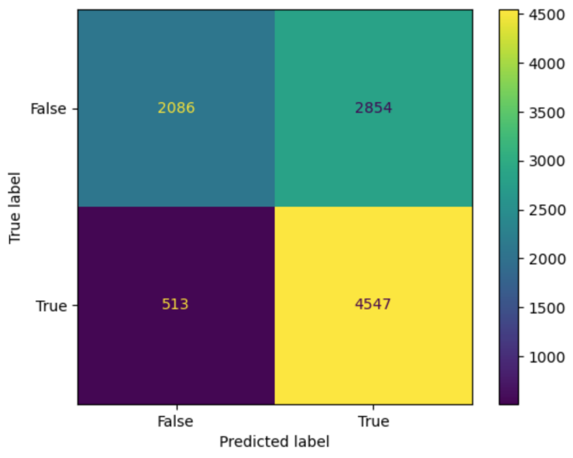


Figure 17. Confusion matrix for the VADER model's predictions.

3. Conclusion

The results of the RNN model are promising. The accuracy is acceptable, but the loss is high. When comparing the accuracy between the results in Figure 10 and the plot in Figure 12 it can be concluded that there is some slight overfitting of the model. When comparing the loss between the results in Figure 11 and the plot in Figure 13 it can be concluded that the learning rate of the model may be too high. During the conception of the model, the number of parameters was increased to try and lower the training loss, but this ultimately decreased the validation accuracy. The layers, number of parameters, batch size, and number of epochs were all played around until the model resulted in the best accuracy and loss. The final model is the best considering the scope of this project.

The results of the RNN model are very consistent with each month that was used for the Russia-Ukraine war sentiment analysis. It was hypothesized that each month would show a more drastic difference in sentiment because of the events that occurred during each month. The results all show a pretty even split between positive and negative sentiment. The results of the RNN model were compared to the results of the VADER model. The months of April and March had very similar results. There was a slight difference in the results in the month of September. The VADER model predicted much more positive sentiment than the RNN model. This could be worth exploring in the future.

The biggest thing that was learned from this project is that a topic like this cannot be analyzed easily. In most cases, people have a negative sentiment when it comes to war. Even if most of the results show a split between positive and negative sentiment, it does not mean that half of the public sides with Ukraine and the other half with Russia. It must mean that there are mixed feelings about the war. It would be interesting to control some of the features that are chosen for the model and indicate which words correspond to which sentiment in terms of support for a country.

3.1. Future Work

There could be many improvements made to the RNN model. It could be better fine-tuned to the Russia-Ukraine Twitter data by trying to train in on a small portion of hand-labeled data. This would take a while, but it could be worth exploring. In addition, some of the common data pre-processing tips were left out. For example, lemmatization was left out because it did not seem to make a difference in the final results. It could be worth exploring adding it back in. In addition, a common pre-processing tool is the use of N-grams. N-grams can help capture the meaning of words that may be misspelled or unclear. It could be useful in pre-processing of the data.

References

[1] Dropout layer. Available at https://keras.io/api/layers/regularization_layers/dropout/. 3

[2] Embedding layer. Available at https://keras.io/api/layers/core_layers/embedding/. 3

[3] Keras dense. Available at https://www.educba.com/keras-dense/. 3

[4] What is sentiment analysis? Available at https://aws.amazon.com/what-is/sentiment-analysis/. 1

[5] Aryan Bajaj. Can python understand human feelings through words? – a brief intro to nlp and vader sentiment analysis. Available at https://www.analyticsvidhya.com/blog/2021/06/vader-for-sentiment-analysis/ (2023/04/19). 4

[6] Jason Brownlee. A gentle introduction to long short-term memory networks by the experts. Available at https://machinelearningmastery.com/gentle-introduction-long-short-term-memory-networks-experts/ (2017/05/24). 2

[7] Jason Brownlee. A gentle introduction to transfer learning for deep learning. Available at https://machinelearningmastery.com/transfer-learning-for-deep-learning/ (2017/12/20). 1

[8] Rani Horev. Bert explained: State of the art language model for nlp. Available at https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270 (2018/11/10). 1

[9] Javaid Nabi. Recurrent neural networks (rnns). Available at https://towardsdatascience.com/recurrent-neural-networks-rnns-3f06d7653a85 (2019/07/11). 1

[10] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Tlanguage models are unsupervised multitask learners. page 1, 2018. 1

[11] Editorial Team. Text classification with rnn. Available at https://towardsai.net/p/deep-learning/text-classification-with-rnn (2020/11/20). 1