Sentiment Analysis of text in Mobile systems

Members:

- 18075054 Sarthak Bansal
- 18075055 Saurabh Yadav
- 18075057 Shubhagra Kasav
- 18075059 Sushant Ashutosh

Project Description

- Nowadays, mostly everyone's mobile is cluttered with many messages and no way of filtering it in an intelligent way. And with all this growing digitalisation, it has become very difficult to prevent ourselves from getting encountered with hate messages which may decrease the overall morale of a person..

- In our project, we will be focussing on solving this particular problem. We are looking to try and design a deep learning based sentiment classification model that gives sentiment scores between low and high (say 0 and 1) respectively where low will represent highly negative sentiment whereas, high will represent highly positive sentiment.

- We will look for a multiclass classification model to achieve this and will compare this with a binary classification model performance.

Dataset Used:

- For the dataset, we will use Stanford sentiment treebank data. This dataset consists of two files.

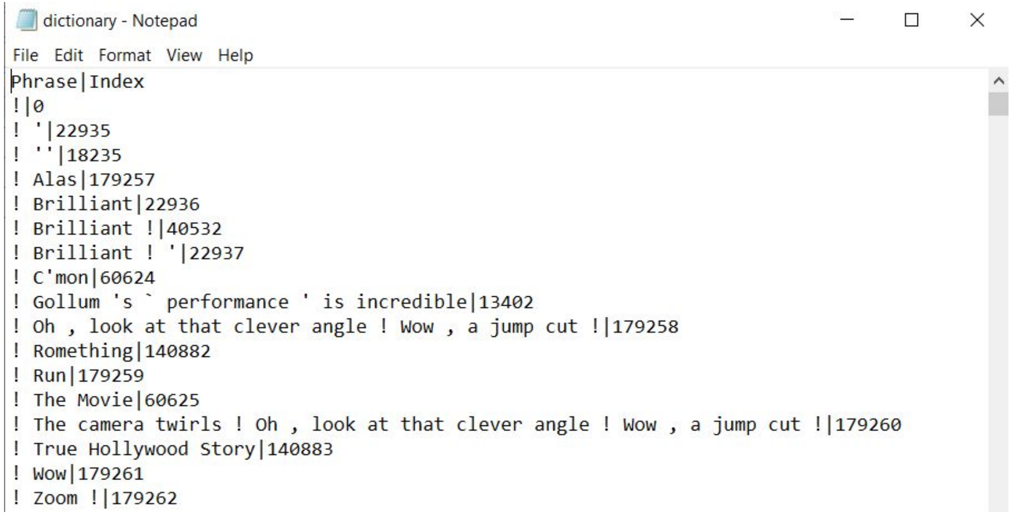
- The data set “dictionary.txt” consists of 239,233 lines of sentences with an index for each line. The index is used to match each of the sentences to a sentiment score in the file “sentiment_labels.txt”. The score ranges from 0 to 1, 0 being very negative and 1 being very positive.

Preprocessing for Text Data

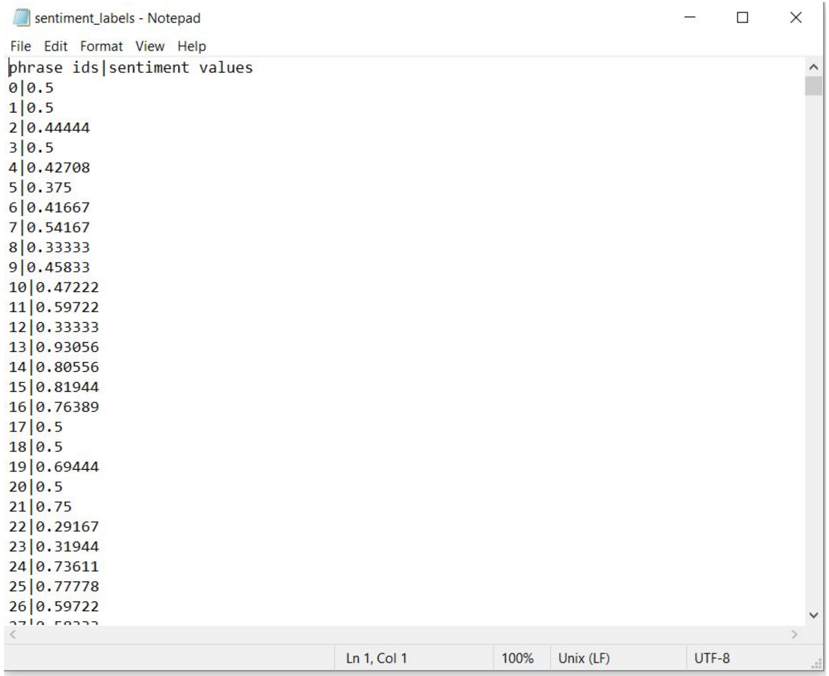
Normalization:

- Convert to Lowercase
- Remove Non ASCII, Punctuation and Special Characters
- Removing Stopwords

Flowchart of operations on Raw Dataset



```
Phrase|Index
!|0
! '|22935
! ''|18235
! Alas|179257
! Brilliant|22936
! Brilliant !|40532
! Brilliant ! '|22937
! C'mon|60624
! Gollum 's ` performance ' is incredible|13402
! Oh , look at that clever angle ! Wow , a jump cut !|179258
! Romething|140882
! Run|179259
! The Movie|60625
! The camera twirls ! Oh , look at that clever angle ! Wow , a jump cut !|179260
! True Hollywood Story|140883
! Wow|179261
! Zoom !|179262
```



```
phrase ids|sentiment values
0|0.5
1|0.5
2|0.44444
3|0.5
4|0.42708
5|0.375
6|0.41667
7|0.54167
8|0.33333
9|0.45833
10|0.47222
11|0.59722
12|0.33333
13|0.93056
14|0.80556
15|0.81944
16|0.76389
17|0.5
18|0.5
19|0.69444
20|0.5
21|0.75
22|0.29167
23|0.31944
24|0.73611
25|0.77778
26|0.59722
```

These two files were combined by taking ids common to create .csv file as processed data (shown below).

Processed Data

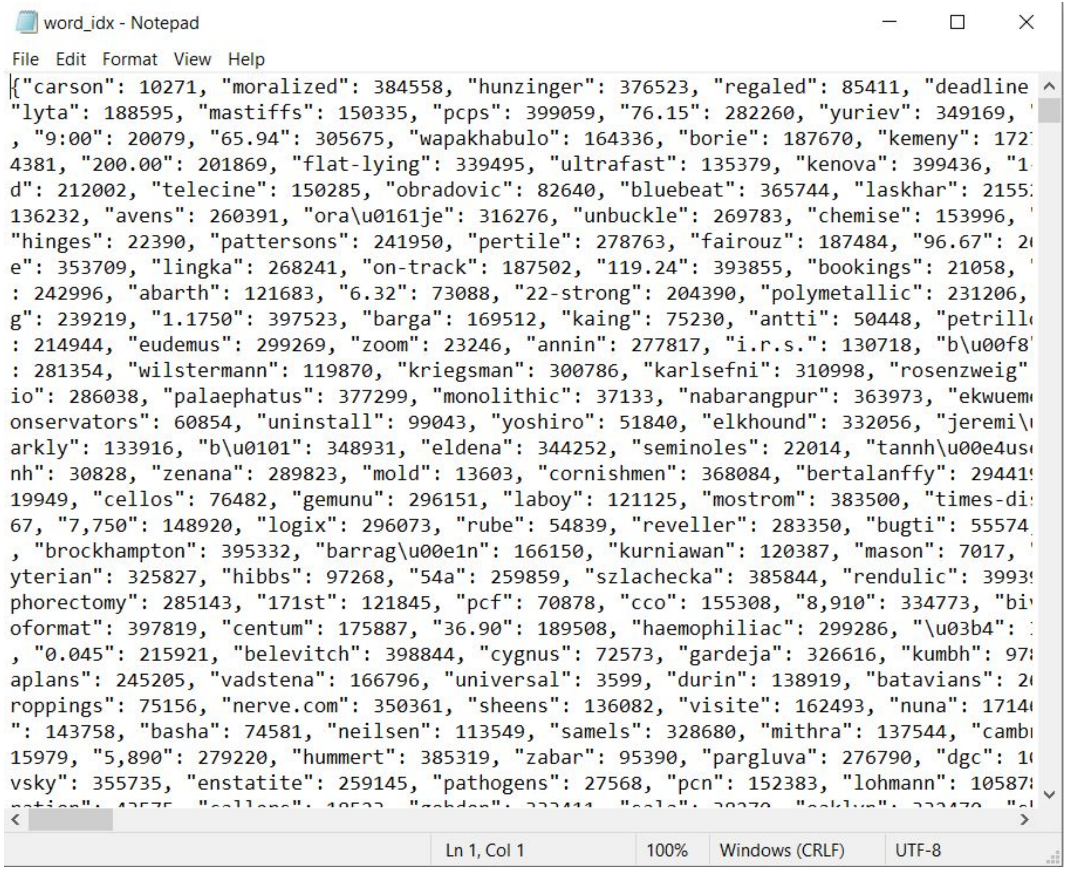
	Phrase	phrase_ids	sentiment_values
1	! '	22935	0.52778
2	! "	18235	0.5
3	! Alas	179257	0.44444
5	! Brilliant !	40532	0.93056
10	! Romething	140882	0.5
11	! Run	179259	0.43056
16	! Zoom !	179262	0.63889
17	!?	220445	0.5
21	# 3	220447	0.5
24	# 8217 ; t	140886	0.5
35	$ 20 million	60629	0.5
36	$ 20 million ticket to ride a Russian rocket	60630	0.54167
39	$ 40 million version	101300	0.43056
42	$ 7	140890	0.54167
51	%	140891	0.5
52	&	3257	0.5
53	& -	140892	0.48611
54	& - white freeze frames reminiscent of a pseudo-l	140893	0.38889
56	& - white freeze frames reminiscent of a pseudo-l	140895	0.33333
61	& Stitch " is n't the most edgy piece of Disney anir	60632	0.77778
63	& heart-rate-raising	101303	0.77778
65	'	1	0.5
68	' (Hopkins) does n't so much phone in his perforn	179268	0.33333
69	' (The Cockettes) provides a window into a subcu	43060	0.625
75	' ...	101305	0.5
77	' ... Mafia , rap stars and hood rats butt their ugly l	22938	0.48611
81	' ... both hokey and super-cool , and definitely not	22939	0.83333
85	' Anderson	179270	0.5
93	' I think ,	179271	0.5
94	' Jia	43061	0.5
97	' Society	43062	0.5

Word2Vec embedding

- The review text was first tokenized to get the vocabulary of the entire corpus and then each word was converted to a vector of 100 dimensions using 10-gram model .

- Further, the model successfully grouped the words with similar context closely in the reduced vector space.

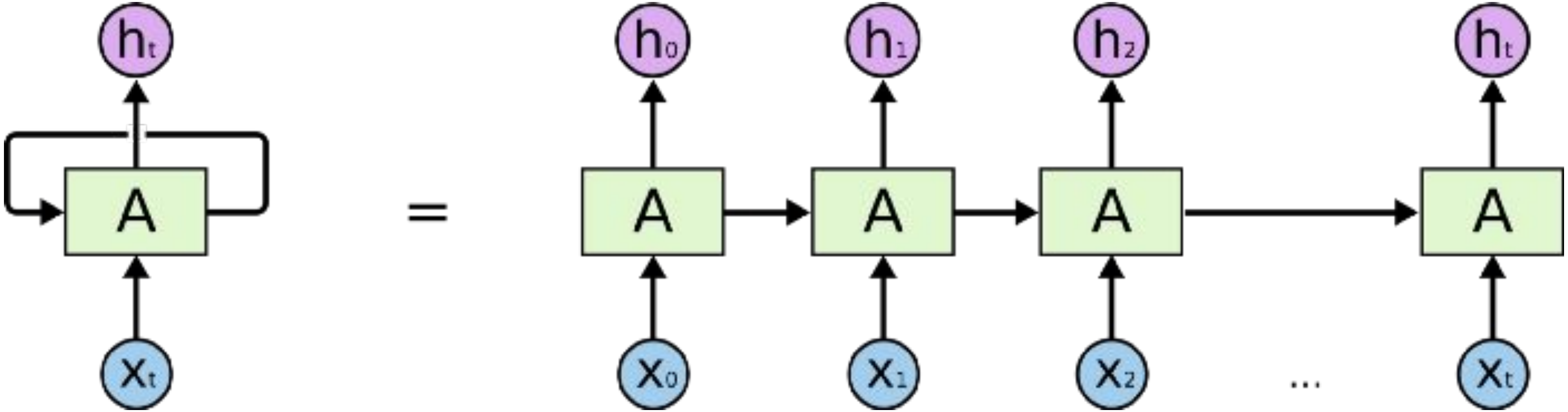
- We have used a pre-trained word embedding model know as GloVe. For our model we have represented each word using a 100 dimension embedding.



Using glove model, we did embedding of dataset i.e we generated different values for different words in a way that the words with similar meaning are grouped together and opposites are kept away.

LSTM Model

- In order to train the model we have used a type of Recurrent Neural Network, know as LSTM (Long Short Term Memory).
- The main advantage of this network is that it is able to remember the sequence of past data i.e. words in our case in order to make a decision on the sentiment of the word.
- It is basically a sequence of copies of the cells, where output of each cell is forwarded as input to the next
- Each cell allows the each cells to decide which of the past information to remember and the ones to forget



h_t
A
X_t
=
h_0 h_1 h_2 h_t
A A A A
X_0 X_1 X_2 ... X_t

Model Summary

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, 56, 100)           40000100
_________________________________________________________________
bidirectional_1 (Bidirection (None, 256)               234496
_________________________________________________________________
dense_1 (Dense)              (None, 512)               131584
_________________________________________________________________
dropout_1 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_2 (Dense)              (None, 10)                5130
=================================================================
Total params: 40,371,310
Trainable params: 371,210
Non-trainable params: 40,000,100
```

- **Layer 1:** An embedding layer of a vector size of 100 and a max length of each sentence is set to 56.
- **Layer 2:**128 cell bi-directional LSTM layers, where the embedding data is fed to the network. We add a dropout of 0.2 this is used to prevent overfitting.
- **Layer 3:**A 512 layer dense network which takes in the input from the LSTM layer. A Dropout of 0.5 is added here.
- **Layer 4:**A 10 layer dense network with softmax activation, each class is used to represent a sentiment category, with class 1 representing sentiment score between 0.0 to 0.1 and class 10 representing a sentiment score between 0.9 to 1.

Model Parameters

- **Activation Function:** I have used ReLU as the activation function. ReLU is a non-linear activation function, which helps complex relationships in the data to be captured by the model.
- **Optimiser:** We use adam optimiser, which is an adaptive learning rate optimiser.
- **Loss function:** We will train a network to output a probability over the 10 classes using Cross-Entropy loss, also called Softmax Loss. It is very useful for multi-class classification.

Training and Testing

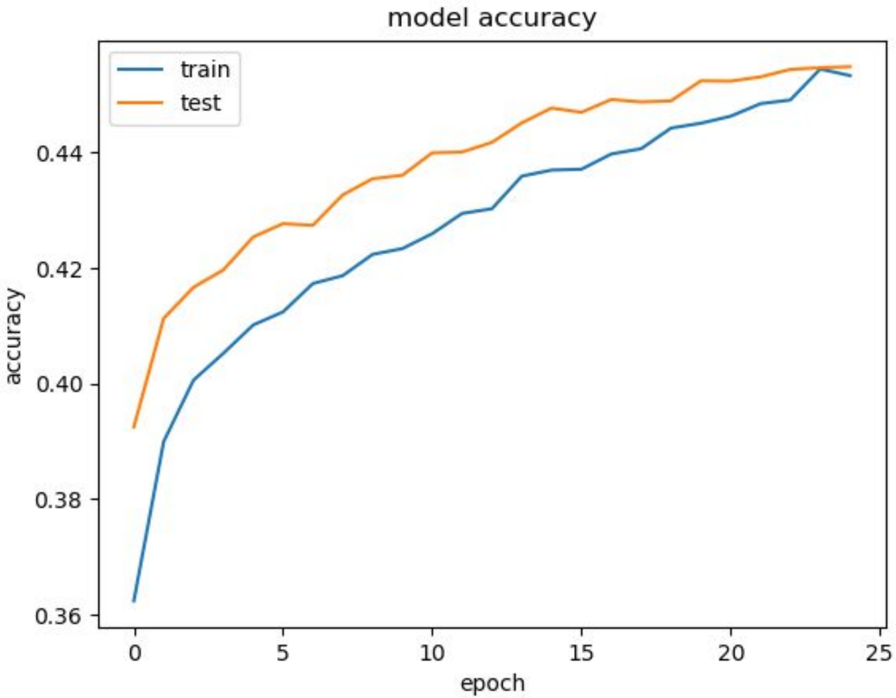
- We have used batch size of 2000 items at a time.
- We set the training set to run for 25 epochs.
- To prevent the model from overfitting we have used early stopping.
- Early stopping is a method that allows us to specify an arbitrary large number of training epochs and stop training once the model performance stops improving on a hold out/validation dataset.

We split the data in three parts:

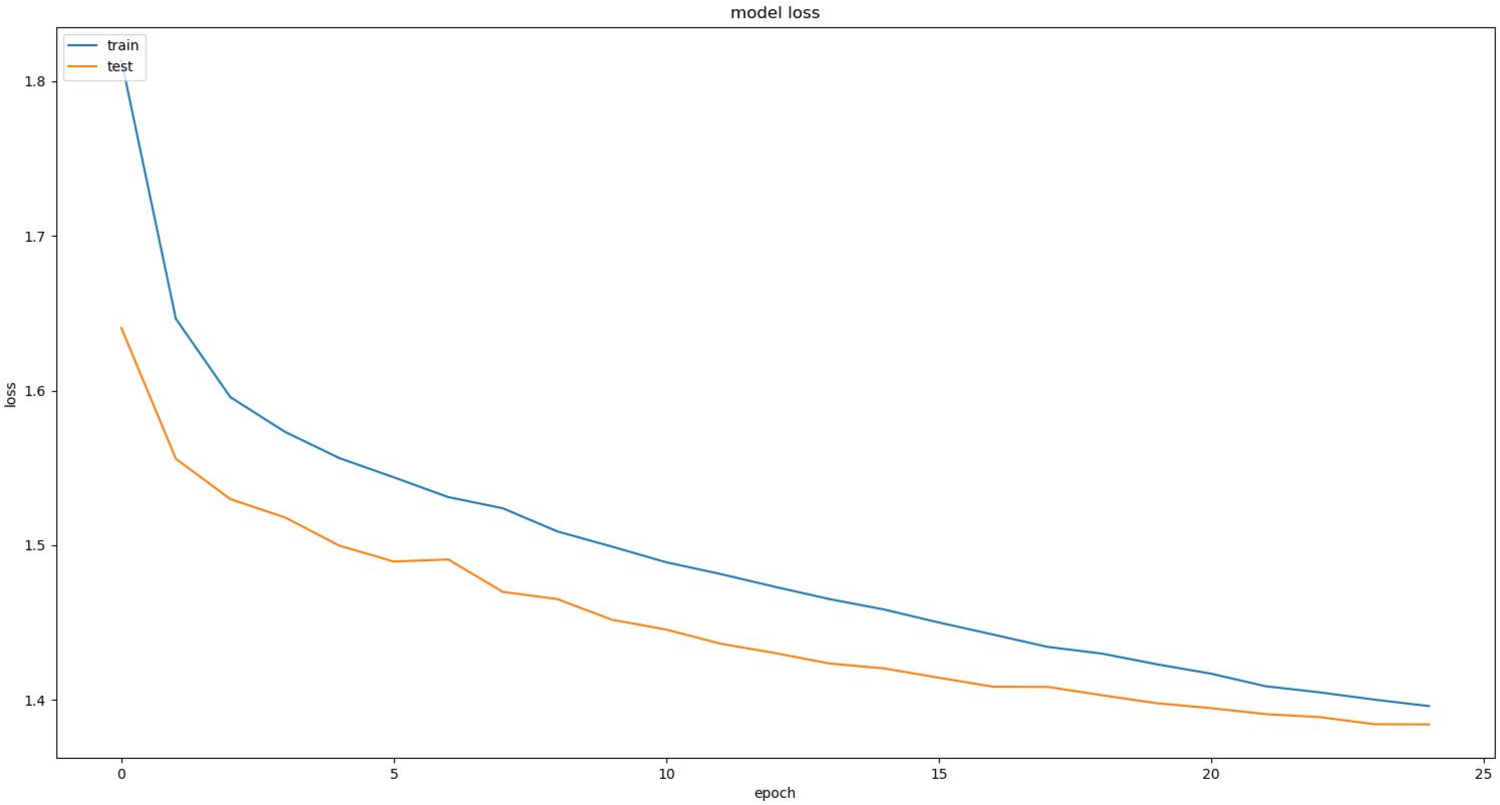
- train.csv : This is the main data which we used to train the model. This is 50% of the overall data.
- val.csv : This is a validation data which used to ensure the model does not overfit. This is 25% of the overall data.
- test.csv : This is used to test the accuracy of the model post training. This is 25% of the overall data.

```python
def train_model(model,train_x, train_y, test_x, test_y, val_x, val_y, batch_size, path) :

    # saving the best model and early stopping
    saveBestModel = keras.callbacks.ModelCheckpoint(path+'/model/best_model.hdf5', monitor='val_acc',
     verbose=0,save_best_only=True, save_weights_only=False, mode='auto', period=1)
    earlyStopping = keras.callbacks.EarlyStopping(monitor='val_loss', min_delta=0,
     patience=3, verbose=0, mode='auto')

    # Fit the model
    history = model.fit(train_x, train_y, batch_size=batch_size, epochs=25,
        validation_data=(val_x, val_y), callbacks=[saveBestModel, earlyStopping])
    # Final evaluation of the model
    score = model.evaluate(test_x, test_y, batch_size=batch_size)

    print("Test Score:", score[0])
    print("Test Accuracy:", score[1])

    # summarize history for accuracy
    plt.plot(history.history['accuracy'])
    plt.plot(history.history['val_accuracy'])
    plt.title('model accuracy')
    plt.ylabel('accuracy')
    plt.xlabel('epoch')
    plt.legend(['train','test'], loc='upper left')
    plt.show()

    # summarize history for loss
    plt.plot(history.history['loss'])
    plt.plot(history.history['val_loss'])
    plt.title('model loss')
    plt.ylabel('loss')
    plt.xlabel('epoch')
    plt.legend(['train','test'], loc='upper left')
    plt.show()

    return model
```

Training and Test plots



- From the model accuracy plot in previous slide, we can see that there is a very small difference between the training accuracy and test accuracy which means that our model is not overfitting.

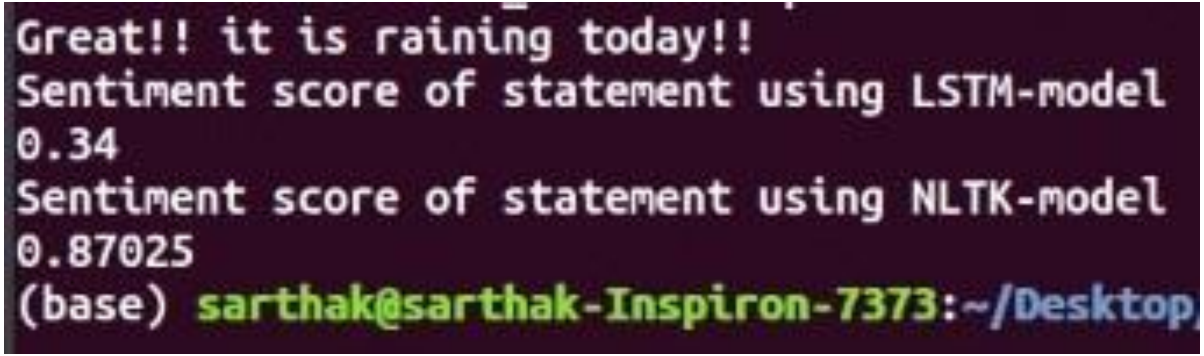

model loss
train
test
1.8
1.7
1.6
1.5
1.4
loss
0
5
10
15
20
25
epoch

Test accuracy

- Test accuracy is as shown by the below fig.

```
Test Score: 1.3891050535395693
Test Accuracy: 0.45579102635383606
```

Running The Model



This sentence “Great!! it is raining today!!” contains negative context and our model is able to predict this as seen below. it gives it a score of 0.34 while NLTK_Model shows positive sentiment.

Result of live test

We used a file live_test_data.txt file containing sentences to do live test and a file live_test_context.txt having context of sentences and show the result of both NLTK and LSTM Models in tabular format using prettytable function of numpy.

The result is as shown by fig on next slide.

Sentence	Context	NLTK_Prediction	LSTM_Prediction
Great!! I have nothing to eat	Negative	0.8446	0.42
I am not gonna fail this semester	Positive	0.6855	0.82
Great!! I have another backlog	Negative	0.8446	0.41
I am not bad at dance	Positive	0.6855	0.74
I am not so great	Negative	0.18519999999999998	0.23
I am not bad at coding	Positive	0.6855	0.75
I am not bad at academics	Positive	0.6855	0.74
I am not bad at singing	Positive	0.6855	0.74
Great!! my attendance is less	Negative	0.8446	0.32
I am not bad at sports	Positive	0.6855	0.74
You are not good but best	Positive	0.13024999999999998	0.79
Great!! we lost the match!!	Negative	0.80414999999999999	0.39

- As shown by result, in some of the statements NLTK model fails to grab the context correctly whereas our LSTM model shows same result as the expected context of the sentences.

THANK YOU