Predict the food reviews from amazon whether it is positive or negative

Avinash Yaganapu[1], Master’s Candidate in Computer Science, UNLV

ABSTRACT When we take a review by a user for food we need to expect whether the review is positive or negative. Here reviews play a role to describe the food for a new user and also company tries to improve by considering those reviews, so we apply a well-known architecture called LSTM which is based on Recurrent Neural Networks (RNN) on amazon fine food reviews dataset which contains all reviews written by users on Amazon website. These reviews are about food related items and our task is to classify them all whether they are 'Positive' or 'Negative' by using LSTM with multiple layer.

I. INTRODUCTION

One of the biggest challenges of an online food delivery system is totally based on the reviews. The main problem involved in this system is reviews are generated frequently and multiple times by a single user, so the task is to predict a review whether it is positive or negative. When we consider all the reviews we may get some thousands of words but the thing we have to know is every sentence consists of some meaning and also follow some sequence that means every letter has some dependency on the previous or later letters based on these dependencies only we are going to use the model called LSTM which uses recurrent neural networks(RNN).

It would greatly benefit for both the dealers and the end buyers if there is a way to determine a review is positive or negative, so that the company may focus on the customer needs. A simple LSTM architecture looks like this.

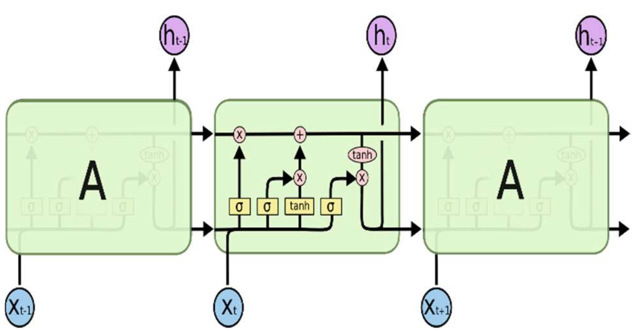


FIGURE 1: LSTM Architecture

II. DATASET DESCRIPTION

Dataset contained 10 unique features with 5,68,454 samples up to October 2012. Reviews include product and user information, ratings, and a plain text review. It also includes reviews from all other Amazon categories. 80% of the entire dataset forms the Training Data and the remaining 20% forms the Testing data.

A. FEATURE DESCRIPTION

TABLE 1

Feature Name	Definition
ID	Row Id
ProductId	Unique identifier for the product
UserId	Unique identifier for the user
Profilename	Profile name of the user
HelpfulnessNumerator	Number of the users who found the Review helpful
HelpfulnessDenominator	Number of users who indicated whether they found the review Helpful or not
Score	Rating between 1 and 5
Time	Timestamp for the review
Summary	Brief summary of the review
Text	Text of the review

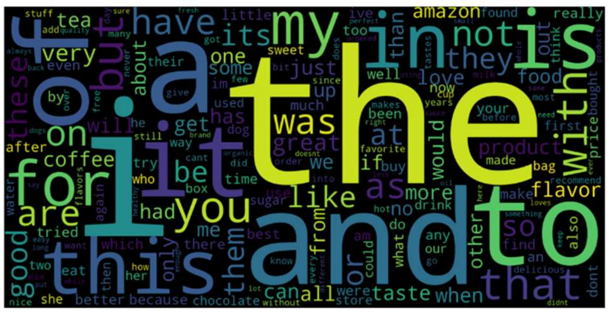
B. KEY OBSERVATIONS IN DATA

First, we concentrate on the features which provides required information. The features like ProductId, UserId, Score, Time, and Text plays a crucial role. Here we are using the data which has a timespan of more than 10 years so that we can be able to predict the score of the product in the future. Here Time is the feature which we will use for sorting and taking some sample data.

C. DATA PREPROCESSING

Data preprocessing is one of the major step here because the dataset is huge. Here first we will start with loading the data and then data cleaning which away all the html tags, punctuations and special characters later we store all the reviews in a list. As we must define either positive or negative, so we need only binary classification so here we need to consider score feature, in this we need to eliminate the reviews with score 3 and we need to convert the score which is below 3 as ‘0’ and above 3 as ‘1’.

Later we need to sort the data according to Time and then we have to eliminate the duplicate data by considering features (ProductId, Time, UserId, Text) and then we have to know the number of positive and negative reviews. Define vocabulary which says about the count of words and their frequency which is represented in below figure. The below figure says that ‘the’ is the most frequent word used. As we done sorting, so we need to take first 50000 samples and then convert the letters into numerical by giving them a rank by considering this vocabulary. By this the data preprocessing completes for saving time we need to save the result of the data in a csv file so that we can use it whenever we needed instead of doing the data preprocessing step again and again.



D. DATASET STATISTICS

Dataset statistics	
Number of reviews	568,454
Number of users	256,059
Number of products	74,258
Users with > 50 reviews	260
Median no. of words per review	56
Timespan	Oct 1999 - Oct 2012

III. CLASSIFIERS USED

The sequential classifier is used for this dataset which takes either 0 or 1 as input.

IV. TYPES OF LAYERS USED

The following layers are used:

- Embedding layer
- Dense layer
- LSTM layer

V. EVALUATION CRITERIA

For balanced data, accuracy is a good measure. And for unbalanced data that means each review may be of different size that means each review may consists of different number of words so to process the data we will use padding for all the samples so that Recurrent Neural Networks works easy.
In the evaluation criteria we will change the number of layers and calculate the accuracy.

VI. Results

1. DATAPREPROCESSING

```
In [7]: runfile('C:/Users/avina/.spyder-py3/temp.py', wdir='C:/Users/avina/.spyder-py3')
Dimension of dataset - :  (364171, 10)

_______________ Frequency of positive and negative reviews _______________
1    307061
0     57110
Name: Score, dtype: int64
Dimension of dataset - :  (50000, 10)

_______________ Frequency of positive and negative reviews _______________
1    42145
0     7855
Name: Score, dtype: int64
```

Figure 1

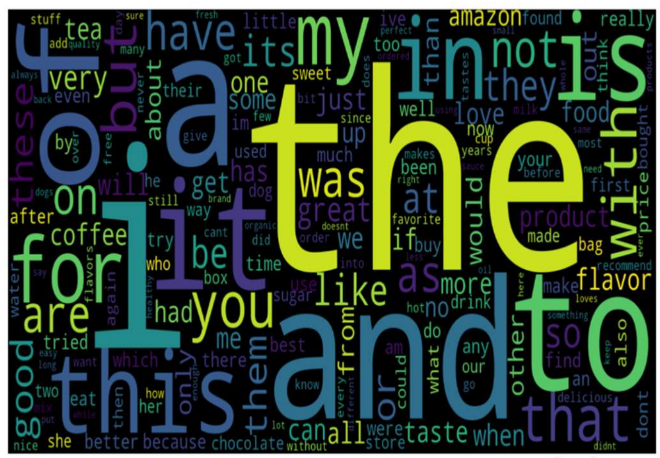


Figure 2

```
__________ SIZE OF VOCABULARY __________
41490
____________ FIRST REVIEW BEFORE CONVERTING ________

['this', 'is', 'one', 'movie', 'that', 'should', 'be', 'in', 'your', 'movie', 'collection', 'it', 'is', 'filled', 'with', 'comedy', 'action', 'and', 'whatever', 'else', 'you', 'want', 'to', 'call', 'it']
____________ FIRST REVIEW AFTER CONVERSION ________

[9, 8, 37, 1698, 13, 267, 29, 10, 72, 1698, 3013, 6, 8, 1151, 14, 11614, 4216, 3, 856, 437, 16, 147, 5, 846, 6]

In [8]:
```

Figure 3

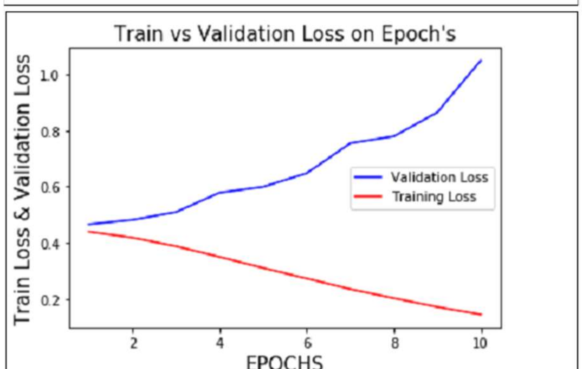
2. 1 LAYER-LSTM

```
In [11]: runfile('C:/Users/avina/.spyder-py3/1layerlstm.py', wdir='C:/Users/avina/.spyder-py3')
----------------------TRAIN DATA-----------------------------------
40000
40000
-------------------------------------------------------------------

----------------------TEST DATA------------------------------------
10000
10000
----------------------TRAIN DATA-----------------------------------
(40000, 700)
40000
-------------------------------------------------------------------

----------------------TEST DATA------------------------------------
(10000, 700)
10000
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, 700, 32)           1426560
_________________________________________________________________
lstm_1 (LSTM)                (None, 100)               53200
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 101
=================================================================
Total params: 1,479,861
Trainable params: 1,479,861
Non-trainable params: 0
```

Figure 4

```
None
Train on 40000 samples, validate on 10000 samples
Epoch 1/10
40000/40000 [==============================] - 944s 24ms/step - loss: 0.4393 - acc: 0.8448 - val_loss: 0.4656 - val_acc: 0.8243
Epoch 2/10
40000/40000 [==============================] - 1134s 28ms/step - loss: 0.4181 - acc: 0.8460 - val_loss: 0.4814 - val_acc: 0.8243
Epoch 3/10
40000/40000 [==============================] - 1116s 28ms/step - loss: 0.3876 - acc: 0.8487 - val_loss: 0.5088 - val_acc: 0.8190
Epoch 4/10
40000/40000 [==============================] - 1244s 31ms/step - loss: 0.3497 - acc: 0.8593 - val_loss: 0.5779 - val_acc: 0.8163
Epoch 5/10
40000/40000 [==============================] - 1195s 30ms/step - loss: 0.3103 - acc: 0.8759 - val_loss: 0.5988 - val_acc: 0.8017
Epoch 6/10
40000/40000 [==============================] - 1108s 28ms/step - loss: 0.2725 - acc: 0.8917 - val_loss: 0.6473 - val_acc: 0.7757
Epoch 7/10
40000/40000 [==============================] - 1126s 28ms/step - loss: 0.2343 - acc: 0.9080 - val_loss: 0.7545 - val_acc: 0.7497
Epoch 8/10
40000/40000 [==============================] - 1073s 27ms/step - loss: 0.2020 - acc: 0.9232 - val_loss: 0.7792 - val_acc: 0.7504
Epoch 9/10
40000/40000 [==============================] - 996s 25ms/step - loss: 0.1707 - acc: 0.9335 - val_loss: 0.8646 - val_acc: 0.7657
Epoch 10/10
40000/40000 [==============================] - 1155s 29ms/step - loss: 0.1441 - acc: 0.9438 - val_loss: 1.0494 - val_acc: 0.7638
```

Figure 5



Accuracy: 76.38%

Figure 6

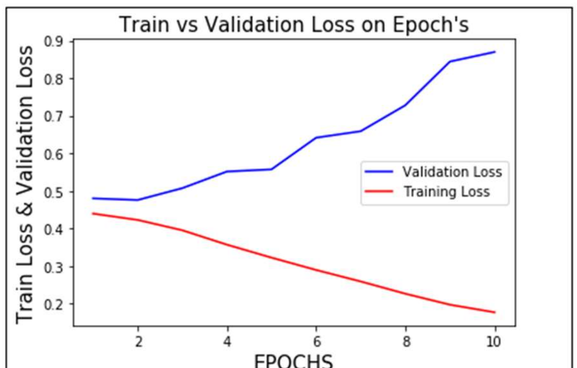
3. 1 LAYER-LSTM WITH DROPOUT

```
----------------------TRAIN DATA-----------------------------------
40000
40000
-------------------------------------------------------------------

----------------------TEST DATA------------------------------------
10000
10000
----------------------TRAIN DATA-----------------------------------
(40000, 700)
40000
-------------------------------------------------------------------

----------------------TEST DATA------------------------------------
(10000, 700)
10000
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_2 (Embedding)      (None, 700, 32)           1328000
_________________________________________________________________
dropout_1 (Dropout)          (None, 700, 32)           0
_________________________________________________________________
lstm_3 (LSTM)                (None, 100)               53200
_________________________________________________________________
dropout_2 (Dropout)          (None, 100)               0
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 101
=================================================================
Total params: 1,381,301
Trainable params: 1,381,301
Non-trainable params: 0
```

Figure 7

```
None
Train on 40000 samples, validate on 10000 samples
Epoch 1/10
40000/40000 [==============================] - 1078s 27ms/step - loss: 0.4394 - acc: 0.8451 - val_loss: 0.4800 - val_acc: 0.8243
Epoch 2/10
40000/40000 [==============================] - 1205s 30ms/step - loss: 0.4226 - acc: 0.8460 - val_loss: 0.4756 - val_acc: 0.8243
Epoch 3/10
40000/40000 [==============================] - 1185s 30ms/step - loss: 0.3951 - acc: 0.8477 - val_loss: 0.5071 - val_acc: 0.8166
Epoch 4/10
40000/40000 [==============================] - 1180s 29ms/step - loss: 0.3568 - acc: 0.8558 - val_loss: 0.5514 - val_acc: 0.8179
Epoch 5/10
40000/40000 [==============================] - 1179s 29ms/step - loss: 0.3222 - acc: 0.8711 - val_loss: 0.5572 - val_acc: 0.8008
Epoch 6/10
40000/40000 [==============================] - 1184s 30ms/step - loss: 0.2894 - acc: 0.8854 - val_loss: 0.6415 - val_acc: 0.8043
Epoch 7/10
40000/40000 [==============================] - 1186s 30ms/step - loss: 0.2588 - acc: 0.8972 - val_loss: 0.6588 - val_acc: 0.7801
Epoch 8/10
40000/40000 [==============================] - 1156s 29ms/step - loss: 0.2262 - acc: 0.9116 - val_loss: 0.7279 - val_acc: 0.7750
Epoch 9/10
40000/40000 [==============================] - 1160s 29ms/step - loss: 0.1969 - acc: 0.9231 - val_loss: 0.8442 - val_acc: 0.7798
Epoch 10/10
40000/40000 [==============================] - 1160s 29ms/step - loss: 0.1767 - acc: 0.9297 - val_loss: 0.8697 - val_acc: 0.7664
```

Figure 8



Accuracy: 76.64%

Figure 9

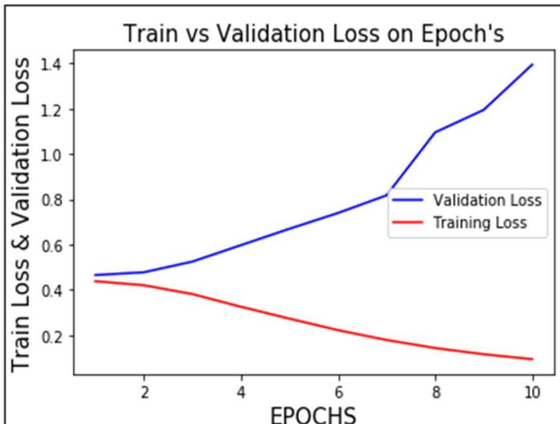
4. 2 LAYER-LSTM

```
In [4]: runfile('C:/Users/avina/.spyder-py3/2layerlsm.py', wdir='C:/Users/avina/.spyder-py3')
----------------------TRAIN DATA-----------------------------------
40000
40000
-------------------------------------------------------------------

----------------------TEST DATA------------------------------------
10000
10000
----------------------TRAIN DATA-----------------------------------
(40000, 700)
40000
-------------------------------------------------------------------

----------------------TEST DATA------------------------------------
(10000, 700)
10000
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_4 (Embedding)      (None, 700, 32)           1328000
_________________________________________________________________
lstm_6 (LSTM)                (None, 700, 100)          53200
_________________________________________________________________
lstm_7 (LSTM)                (None, 100)               80400
_________________________________________________________________
dense_4 (Dense)              (None, 1)                 101
=================================================================
Total params: 1,461,701
Trainable params: 1,461,701
Non-trainable params: 0
```

Figure 10

```
None
Train on 40000 samples, validate on 10000 samples
Epoch 1/10
40000/40000 [==============================] - 5208s 130ms/step - loss: 0.4378 - acc: 0.8449 - val_loss: 0.4651 - val_acc: 0.8243
Epoch 2/10
40000/40000 [==============================] - 5418s 135ms/step - loss: 0.4202 - acc: 0.8461 - val_loss: 0.4773 - val_acc: 0.8243
Epoch 3/10
40000/40000 [==============================] - 5433s 136ms/step - loss: 0.3811 - acc: 0.8503 - val_loss: 0.5245 - val_acc: 0.8211
Epoch 4/10
40000/40000 [==============================] - 6342s 159ms/step - loss: 0.3251 - acc: 0.8679 - val_loss: 0.5970 - val_acc: 0.8009
Epoch 5/10
40000/40000 [==============================] - 7023s 176ms/step - loss: 0.2726 - acc: 0.8915 - val_loss: 0.6695 - val_acc: 0.7520
Epoch 6/10
40000/40000 [==============================] - 9465s 237ms/step - loss: 0.2217 - acc: 0.9123 - val_loss: 0.7391 - val_acc: 0.7439
Epoch 7/10
40000/40000 [==============================] - 5738s 143ms/step - loss: 0.1783 - acc: 0.9311 - val_loss: 0.8171 - val_acc: 0.7650
Epoch 8/10
40000/40000 [==============================] - 5647s 141ms/step - loss: 0.1430 - acc: 0.9451 - val_loss: 1.0951 - val_acc: 0.7297
Epoch 9/10
40000/40000 [==============================] - 5578s 139ms/step - loss: 0.1154 - acc: 0.9550 - val_loss: 1.1944 - val_acc: 0.7143
Epoch 10/10
40000/40000 [==============================] - 5591s 140ms/step - loss: 0.0937 - acc: 0.9640 - val_loss: 1.3943 - val_acc: 0.7276
```

Figure 11



Accuracy: 72.76%

Figure 12

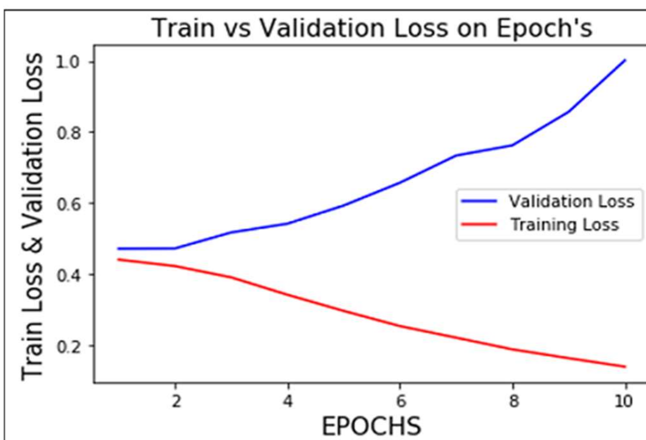
5 . 2 LAYER-LSTM WITH DROPOUT

```
----------------------TRAIN DATA-----------------------------------
40000
40000
-------------------------------------------------------------------
----------------------TEST DATA------------------------------------
10000
10000
----------------------TRAIN DATA-----------------------------------
(40000, 700)
40000
-------------------------------------------------------------------

----------------------TEST DATA------------------------------------
(10000, 700)
10000
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_5 (Embedding)      (None, 700, 32)           1328000
_________________________________________________________________
dropout_3 (Dropout)          (None, 700, 32)           0
_________________________________________________________________
lstm_8 (LSTM)                (None, 700, 100)          53200
_________________________________________________________________
dropout_4 (Dropout)          (None, 700, 100)          0
_________________________________________________________________
lstm_9 (LSTM)                (None, 100)               80400
_________________________________________________________________
dropout_5 (Dropout)          (None, 100)               0
_________________________________________________________________
dense_5 (Dense)              (None, 1)                 101
=================================================================
Total params: 1,461,701
Trainable params: 1,461,701
Non-trainable params: 0
```

Figure 13

```
None
Train on 40000 samples, validate on 10000 samples
Epoch 1/10
40000/40000 [==============================] - 5654s 141ms/step - loss: 0.4409 - acc: 0.8447 - val_loss: 0.4716 - val_acc: 0.8243
Epoch 2/10
40000/40000 [==============================] - 5712s 143ms/step - loss: 0.4227 - acc: 0.8460 - val_loss: 0.4723 - val_acc: 0.8243
Epoch 3/10
40000/40000 [==============================] - 5739s 143ms/step - loss: 0.3909 - acc: 0.8483 - val_loss: 0.5173 - val_acc: 0.8209
Epoch 4/10
40000/40000 [==============================] - 5693s 142ms/step - loss: 0.3420 - acc: 0.8631 - val_loss: 0.5419 - val_acc: 0.8101
Epoch 5/10
40000/40000 [==============================] - 5664s 142ms/step - loss: 0.2965 - acc: 0.8818 - val_loss: 0.5930 - val_acc: 0.7847
Epoch 6/10
40000/40000 [==============================] - 5632s 141ms/step - loss: 0.2538 - acc: 0.8996 - val_loss: 0.6572 - val_acc: 0.7632
Epoch 7/10
40000/40000 [==============================] - 5606s 140ms/step - loss: 0.2212 - acc: 0.9152 - val_loss: 0.7338 - val_acc: 0.7622
Epoch 8/10
40000/40000 [==============================] - 5586s 140ms/step - loss: 0.1882 - acc: 0.9275 - val_loss: 0.7625 - val_acc: 0.7421
Epoch 9/10
40000/40000 [==============================] - 5548s 139ms/step - loss: 0.1634 - acc: 0.9371 - val_loss: 0.8567 - val_acc: 0.7483
Epoch 10/10
40000/40000 [==============================] - 5542s 139ms/step - loss: 0.1395 - acc: 0.9472 - val_loss: 1.0016 - val_acc: 0.7430
```

Figure 14



Accuracy: 74.30%

Figure 15

VI. DETAILED ANALYSIS

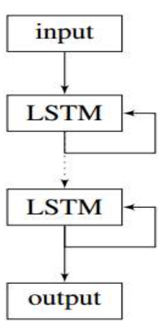
A. 1 LAYER-LSTM VS 1 LAYER-LSTM WITH DROPOUT

Accuracy is usually considered a good measure for balanced data, but for unbalanced data we ae using padding and making it a balanced data. Here through the results we can understand that 1 layer-LSTM Dropout performs well for this dataset this is due to dropping out units in a neural network during the training phase of certain set of neurons which is chosen at random. By ignoring these units are not considered during a forward or backward pass.

In a fully connected layer it occupies most of the parameters, and hence, neurons develop codependency amongst each other during training which curbs the individual power of each neuron leading to overfitting of training data, Dropout helps to prevent this overfitting of data.

B. 2 LAYER-LSTM VS 2 LAYER-LSTM WITH DROPOUT

The same thing happened when we increase the LSTM layers, the Dropout helps to gain more accuracy and tries to prevent overfitting. But the thing we must observe is the accuracy rate of 2 layer-LSTM is smaller than the 1 layer-LSTM this is due to dependency between the letters in a sentence. The deeper layers are more efficient but, in this dataset, it differs due to the dependency factors, for example in the below diagram the first LSTM layer might learn that some characters are vowels and others are consonants the second layer would build on this to learn that a vowel is most likely to follow a consonant.



VII. CONCLUSION

Using Dropout over multiple layers worked better than the normal one. This shows us that Dropout gives better results.

REFERENCES

- J. McAuley and J. Leskovec. From amateurs to connoisseurs: modeling the evolution of user expertise through online reviews. WWW, 2013.
- Dataset: https://www.kaggle.com/snap/amazon-fine-food-reviews.