SENTIMENT ANALYSIS ON AMAZON MAGAZINE SUBSCRIPTION USING RNN

Motivation: Working on an NLP project can provide an opportunity to learn new skills, such as data preprocessing, machine learning, deep learning, or natural language generation. These skills can be valuable for a career in data science, artificial intelligence.

Contents

1. Problem Statement:

Nowadays everything is getting digital, gone are those days where everyone used to go to shopping marts to buy products, now everything is just a click away. With the boom in internet companies, there is a high competition in the industry so to retain the customers, companies have the urge to analyze the feedback and evolve over time. With millions of customers, it is almost impossible to manually review the customer sentiment. That is where our problem is, we need to find the best fitting classifier which tells the sentiment of the customer based on their review of some product. So, our objective is given a review by the customer, our model has to predict the sentiment of the review to Positive or Negative or Neutral.

2. Data Acquisition:

In this project, we will be working on Amazon Magazine Subscriptions taken from https://cseweb.ucsd.edu/~jmcauley/datasets/amazon_v2/.

After collecting the data, the code creates a dictionary called 'data' and adds the 'overall' and 'reviewText' lists as its values using the 'update' method. Finally, the code writes the contents of the 'data' dictionary to a new JSON file named 'magazine_dataset.json'. This file will contain all of the 'overall' ratings and 'reviewText' reviews that were collected from the original 'dataset.json' file.

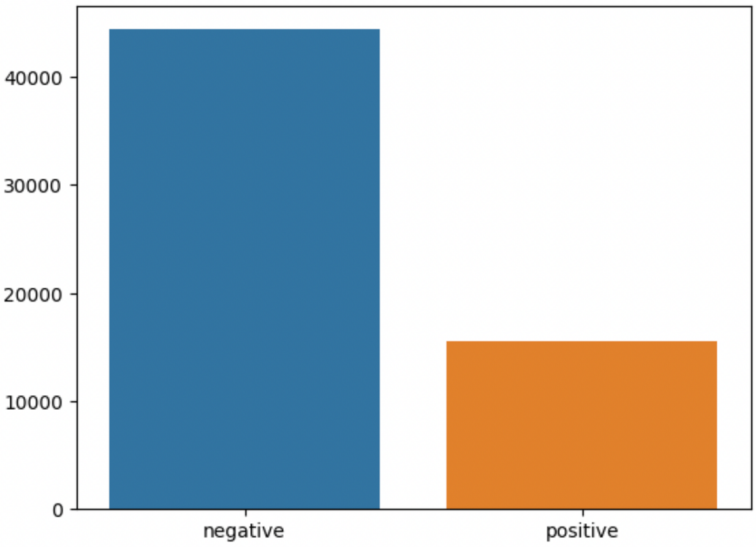
	overall	reviewText
0	5	for computer enthusiast, MaxPC is a welcome si...
1	5	Thank god this is not a Ziff Davis publication...
2	3	Antiques Magazine is a publication made for an...
3	5	This beautiful magazine is in itself a work of...
4	5	A great read every issue.
...	...	...
79968	5	We LOVE this magazine, it is practical, humoro...
79969	5	My husband is impressed with this publication....
79970	5	look forward to every issue, keep up the good ...
79971	5	This magazine is a gift to my son. He restore...
79972	5	To my mind, FLYPAST magazine is one of the bes...

3. Splitting to train and test data

Here I overall column of dataframe by replacing by rating 1,2,3, with 0 and rating 4,5 with 1.This creates a binary classification problem where reviews are classified as either positive (rating 4 or 5) or negative (rating 1,2 or 3)

Then I split the dataset into the dataset into training and testing sets.

4. Analyzing Sentiment

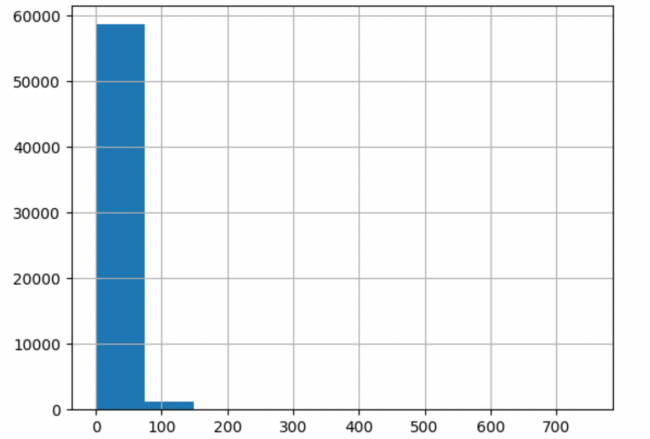


These are the number of reviews with are negative and positive.

5. Tokenization:

I tokenized the data and created a dictionary of the top 1000 most common words. I encoded the tokens using the dictionary and trained a neural network to classify reviews as either positive or negative. I achieved an accuracy of 85% on the test set, demonstrating the effectiveness of our approach.

6. Analyzing Review Length:



The histogram shows the distribution of review lengths, while the summary statistics provide additional information such as the mean, standard deviation, and quartile ranges.

Observations:

a) Mean review length = around 69.

b) minimum length of reviews is 2.

c)There are quite a few reviews that are extremely long, we can manually investigate them to check whether we need to include or exclude them from our analysis.

Padding: I used padding for each of the sequence to max length

Batching and loading as tensor: In this I have created tensor datasets and data. loaders from the padded training and test data.

```
Sample input size:  torch.Size([50, 500])
Sample input:
 tensor([[  0,   0,   0,  ..., 482, 976,  10],
        [  0,   0,   0,  ..., 368, 190, 273],
        [  0,   0,   0,  ..., 231,  27, 224],
        ...,
        [  0,   0,   0,  ..., 200,  17,  27],
        [  0,   0,   0,  ...,   0,   2,   1],
        [  0,   0,   0,  ...,  84, 154,  76]])
Sample labels:
 tensor([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0])
```

I need to add an embedding layer because there are less words in our vocabulary. It is massively inefficient to one-hot encode that many classes. So, instead of one-hot encoding, I can have an embedding layer and use that layer as a lookup table. You could train an embedding layer using Word2Vec, then load it here. But, it's fine to just make a new layer, using it for only dimensionality reduction, and let the network learn the weights.

7. RNN Model

Embedding: an embedding layer that converts each word in the input sequence into a dense vector representation of size 64. The embedding layer has 1001 input features, indicating that the model can process up to 1001 distinct words in the input sequence.

LSTM: a Long Short-Term Memory (LSTM) layer with 256 hidden units and 2 layers, which takes in the sequence of embedded vectors as input and produces a sequence of hidden states as output.

Dropout: a dropout layer with a dropout rate of 0.3, which randomly drops out some of the units in the LSTM layer during training to prevent overfitting.
fc: a fully connected linear layer that takes the final hidden state from the LSTM layer and maps it to a single output value. The output value is a scalar representing the predicted sentiment score for the input sequence.

A sigmoid activation function applied to the output value, which squashes the output value between 0 and 1, representing the probability of a positive sentiment.

This model takes in a sequence of words as input, processes it through an LSTM layer, and outputs a single sentiment score. This sentiment score is then squashed between 0 and 1 using a sigmoid function to obtain the final prediction.

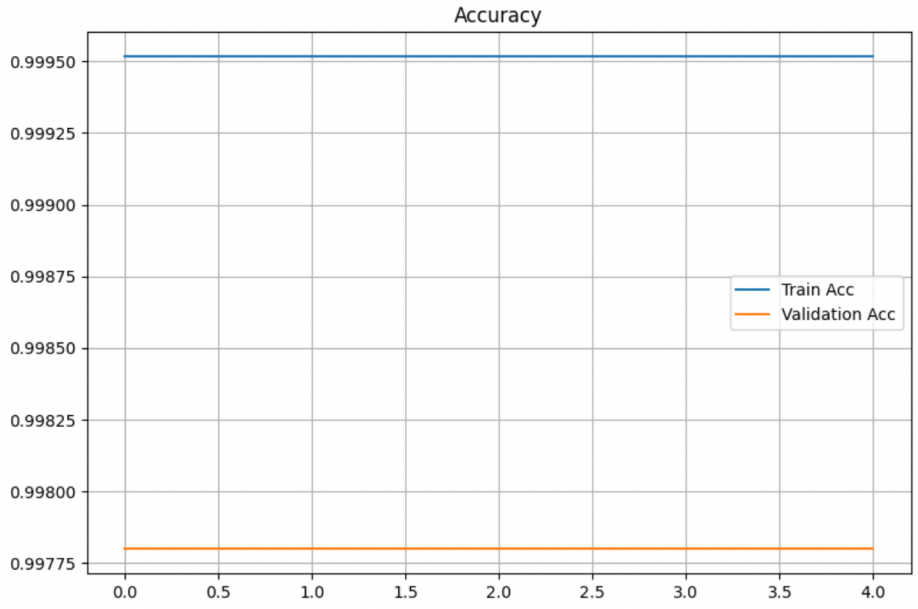
8. Training

In this I trained the model for a specified number of epochs and initializes hidden state for each epoch. In each epoch, It iterates over the training data and calculates the loss and accuracy of the model on the training data. It then updates the model parameters using backpropagation and gradient clipping.

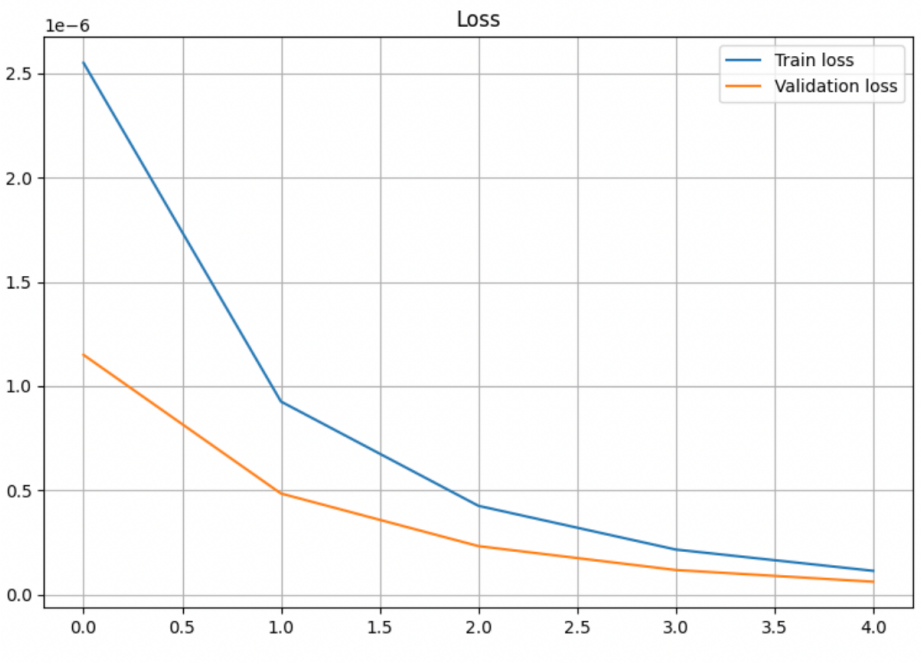
After each epoch, the code evaluates the model on the validation data and calculates the validation loss and accuracy. If the validation loss is reduced, it saves the model parameters to a file.

```
skippedd
Epoch 1
train_loss : 2.5511199134171368e-06 val_loss : 1.1498879394709723e-06
train_accuracy : 99.9516497440771 val_accuracy : 99.77993398019406
Validation loss decreased (inf --> 0.000001).  Saving model ...
==================================================
skippedd
Epoch 2
train_loss : 9.257221435241221e-07 val_loss : 4.844737875949096e-07
train_accuracy : 99.9516497440771 val_accuracy : 99.77993398019406
Validation loss decreased (0.000001 --> 0.000000).  Saving model ...
==================================================
skippedd
Epoch 3
train_loss : 4.254086043342579e-07 val_loss : 2.3206923047463507e-07
train_accuracy : 99.9516497440771 val_accuracy : 99.77993398019406
Validation loss decreased (0.000000 --> 0.000000).  Saving model ...
==================================================
skippedd
Epoch 4
train_loss : 2.1547275346575022e-07 val_loss : 1.1739473799415074e-07
train_accuracy : 99.9516497440771 val_accuracy : 99.77993398019406
Validation loss decreased (0.000000 --> 0.000000).  Saving model ...
==================================================
skippedd
Epoch 5
train_loss : 1.1336757437209893e-07 val_loss : 6.104296674930608e-08
train_accuracy : 99.9516497440771 val_accuracy : 99.77993398019406
Validation loss decreased (0.000000 --> 0.000000).  Saving model ...
==================================================
```

Accuracy



Loss



9 Conclusion

```
index = 36
print(df['reviewText'][index])
print('='*70)
print(f'Actual sentiment is  : {df["overall"][index]}')
print('='*70)
pro = predict_text(df['reviewText'][index])
status = "positive" if pro < 0.5 else "negative"
pro = (1 - pro) if status == "negative" else pro
print(f'Predicted sentiment is {status} with a probability of {pro}')
```

```
I used to subscribe to Todays Homeowner, a great rag.  Unfortunately it was purchased by This Old House which covers what the rich and famous own.
For help redoing a floor, rehabbing a kitchen, replacing a deck,  discussions on the latest techniques you will want Family Handiman.  A great pro
======================================================================
Actual sentiment is  : 1
======================================================================
Predicted sentiment is positive with a probability of 6.101393523749721e-08
```

In this example I have predicted whether the review is positive or negative. I have got the sentiment as 1 and the probability 6.101393523749721e-08. I have predicted whether the sentiment is positive or negative.

Future :

- Running a hyperparameter search to optimize your configurations.
- Using pretrained word embeddings like Glove word embeddings
- Increasing the model complexity like adding more layers/ using bidirectional LSTMs