Capstone Project Report

Machine Learning Engineer Nanodegree

Sunil Kumar Rajput

March 06, 2019

Table of Contents

Table of Figures

Sentiment Analysis on Amazon Reviews

I. Definition

Project Overview

Analyzing and predicting consumers behaviour has already been a promising area of study with great value of research. By analyzing the polarity of the text, decision makers like companies who sell their products can effectively inspect the strength and weakness of their products. Reviews posted on Amazon are not only related to the product that are sold but also the service given to the customers. **In this project I propose a system that performs the classification of the reviews.**

Sentimental analysis often requires to using a combination of techniques like Natural Language Processing and text analysis to identify the positive or negative sentences or emotions. The whole process has parts such as tokenization, stop word filtering, stemming, classification and prediction of sentiment. The more detailed process will be described in the methodology part of the report.

This technique holds a great hidden business value as it allows us to quantize the subjective information for further investigation. For example, by analyzing the polarity of the text, decision makers can effectively understand the strength and weakness of their product or even anticipate the complaints and the sales amount. Due to its vast potential there are a lot of researchers applying the topic of sentimental analysis.

I was always fascinated by the thought of making a machine understand the Natural Language the way human consume process and understand. It is very easy for a human to differentiate *"This product is good"* and *"This product is not so good"* but for machines it is a quite a work to understand that these are 2 opposite class of statements.

Problem Statement

My goal is to implement **sentimental analysis on the Amazon Review Dataset using transfer learning technique.** I believe the modern transfer learning technique can help achieve a higher accuracy than traditional method of training the models from scratch. The model will be based on Neural networks, the good thing about neural networks is that they are good at understanding patterns in data, but the bottleneck is data, they need a lot data to train on. This requires both huge data and longer training times.

Solution Statement

So **I will prefer using a pre-trained 3-layer AWD-LSTM model**[1] developed by Salesforce's research that is already been trained on 100 million tokens from Wikipedia[2] articles.

Dataset Amazon Reviews has been fetched from Xiang Zhang's Google Drive [3] dir. The file named amazon_review_full_csv.tar.gz will be used.

The dataset has 3 columns name Ratings, Summary and Review.

I will take advantage of transfer learning and use a pre-trained language model so; my model will already have a good enough sense of English sentences.

Using language model, I will build an RNN classifier that will predict the polarity of the review. I believe using a pre-trained LM will enhance the score of prediction.

Metrics

The performance of each classification model is evaluated using three statistical measures, classification accuracy, sensitivity and specificity.

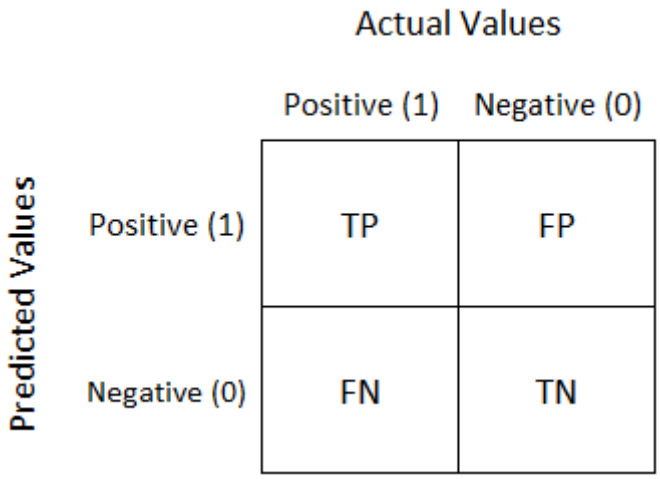

Figure 1 Confusion Matrix

In the above confusion matrix TP stands for True Positive (correctly predicted as positive), FP stands for False Positive (falsely predicted as positive), FN stands for False Negative (falsely predicted as negative) and TN stands for True Negative (correctly predicted as negative).

For any binary classification we have these 4 values calculated. Based on these values we can calculate accuracy, sensitivity and specificity.

Sensitivity (also called the true positive rate, the recall, or probability of detection in some fields) measures the proportion of actual positives that are correctly identified. It is calculated by dividing TP by total number of TP + TN.

Specificity (also called the true negative rate) measures the proportion of actual negatives that are correctly identified. It is calculated by dividing TN by total number of TN + TP.

Classification **accuracy** is defined as the ratio of the number of correctly classified cases and total number of cases.

I will use accuracy as the evaluation metric.

For further reading please see here[9].

During the training process the log loss will also tell the performance of model in real time.

Justification

In classification when the dataset is itself skewed i.e. either of the class if largely outnumbered by the other then accuracy it not really a good metric to use. I would really suggest reading more about this here.

But the dataset I am using has equal number of positive and negative reviews to accuracy is a good metric to evaluate our model.

II. Analysis

Data Exploration

Dataset Amazon Reviews has been fetched from Xiang Zhang's Google Drive [3] dir. The file named amazon_review_full_csv.tar.gz will be used.

This dataset contains 3.65 million reviews collected with ratings with their summary.

The dataset has 3 columns named Ratings, Summary and Review. The dataset is already divided into 3000000, 650000 training reviews and testing reviews respectively.

	Ratings	Summary	Review
0	3	more like funchuck	Gave this to my dad for a gag gift after direc...
1	5	Inspiring	I hope a lot of people hear this cd. We need m...
2	5	The best soundtrack ever to anything.	I'm reading a lot of reviews saying that this ...
3	4	Chrono Cross OST	The music of Yasunori Misuda is without questi...
4	5	Too good to be true	Probably the greatest soundtrack in history! U...
5	5	There's a reason for the price	There's a reason this CD is so expensive, even...
6	1	Buyer beware	This is a self-published book, and if you want...
7	4	Errors, but great story	I was a dissapointed to see errors on the back...
8	1	The Worst!	A complete waste of time. Typographical errors...
9	1	Oh please	I guess you have to be a romance novel lover f...

Figure 2 Dataset Preview

Columns

1. Ratings : type numerical values ranging between 1 to 5
2. Summary : type string
3. Review : type string

Count of columns in train dataset

```
Ratings    3000000
Summary    2999924
Review     3000000
```

Count of columns in test dataset

```
Ratings    650000
Summary    649988
Review     650000
```

We can clearly see that some values in summary column is missing but the Ratings and Review columns are fine.

Exploratory Visualization

Before proceeding towards the solving the problem lets have a look in the data.

The table shown below mentions the count of Reviews and Summary for each Rating value.

Ratings	Summary	Review
1	599988	600000
2	599977	600000
3	599977	600000
4	599987	600000
5	599995	600000

Figure 3 Count of Ratings per Review

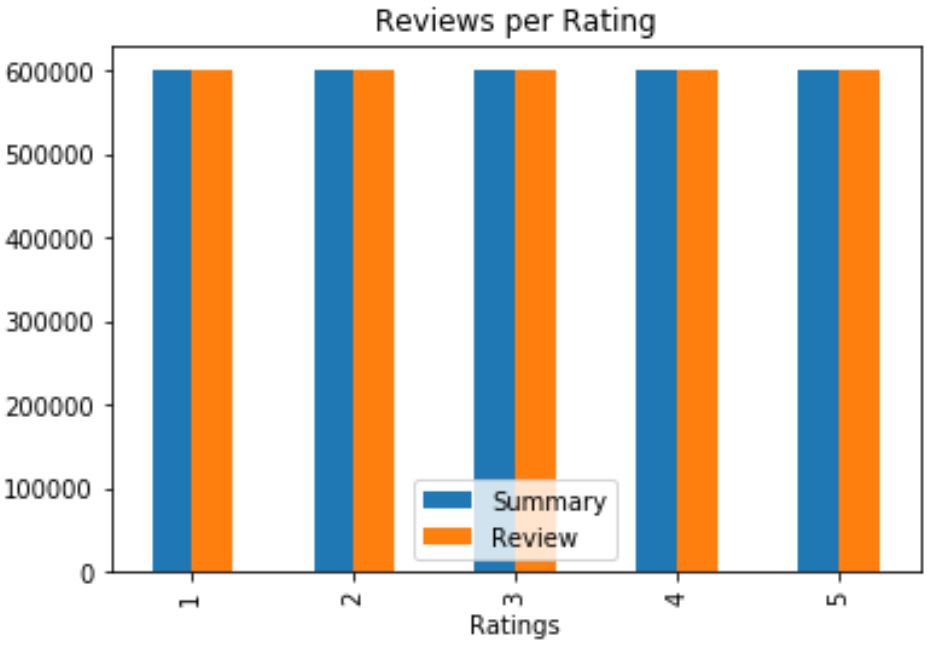

Figure 4 Graph of Rating per Review

As per the graph we can see that number of reviews in Rating category is consistently 600000, but the count of Summary is little below.

Algorithms and Techniques

Pre-trained Language Model

To use the pre-trained language model, we download the weights from here. These weights are trained and saved by Fastai and released for public use to have the advantage of Transfer Learning.

So, we load the structure of model and give them the pre-trained weights and then edit the weights according to our new vocabulary generated from reviews.

This method saves a lot of training time and gives a language model which has a very good understanding of English. Further details are explained in Implementation part.

Embedding matrix

Every word will be represented by a vector of this length. Read more about embedding matrices here.

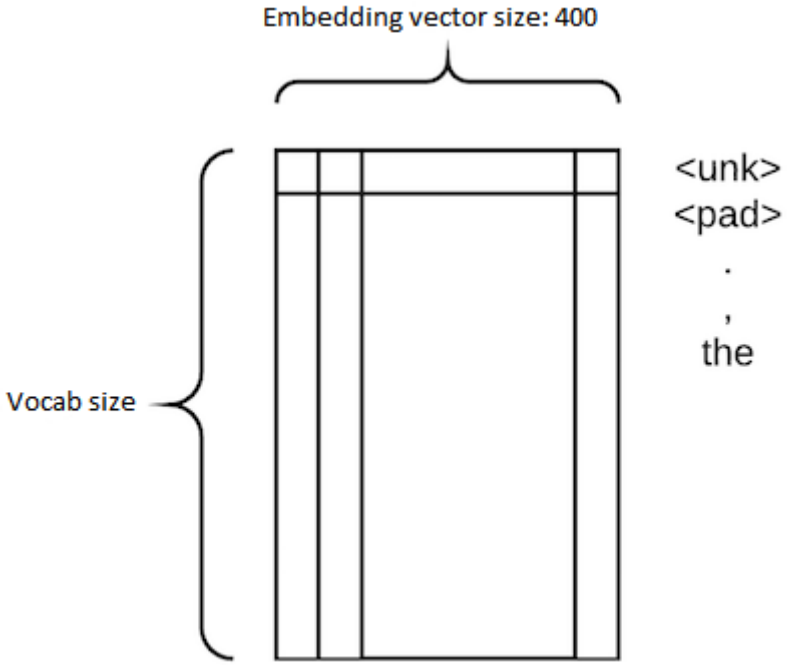

Figure 5 Embedding matrix

So, in our case the embedding vector size 400 from our AWD LSTM model.

Batch size and BPTT

What happens in a language model is even though we have lots of reviews, they all get concatenated together into one big block of text. So, we predict the next word in this huge long thing which is all the Amazon reviews concatenated together.

- We split up the concatenated reviews into batches.
- We then move each section underneath the previous one and transpose it.
- We end up with a matrix which is 1 million by n. In our case 52.
- We then grab a little chunk at time and those chunk lengths are approximately equal to BPTT. Here, we grab a little 70 long section and that is the first thing we chuck into our GPU (i.e. the batch).

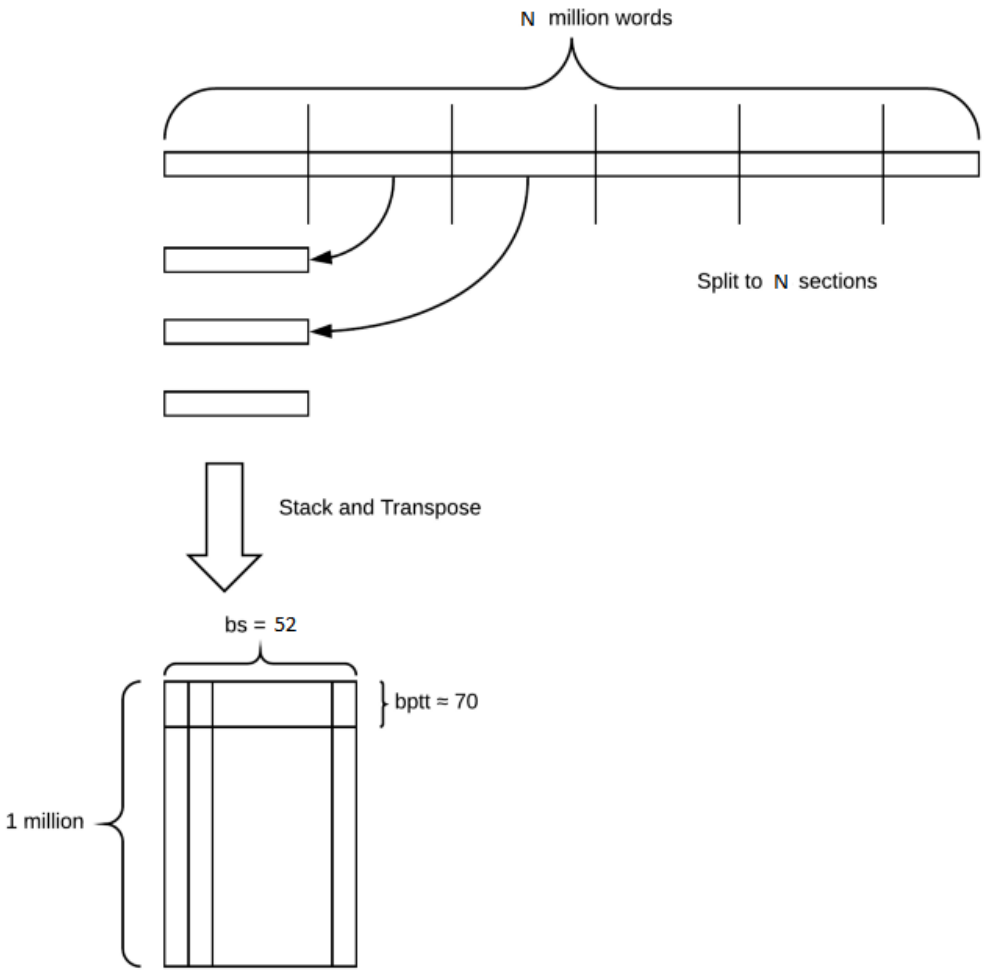

Figure 6 Batch size and BPTT

Benchmark

For a benchmark model I will first use a count vectorizer[10] which will convert the collection of words into a matrix of tokens and then build a Logistic classifier[11] on top of that. Please note in this process the model will be trained from scratch.

Read more about count vectorizer here.

Due to memory limitations I have not used the whole dataset of 3million rows. I have described the data set manipulation in data pre-processing part of the report. The exact same dataset I used for training both models.

Using this method, I got a test accuracy of 86.19%.

III. Methodology

Data Pre-processing

Now as we know that we have 3 columns named Ratings, Summary and Review. Below mentioned are all the steps that I have carried out before using the data for any training.

1. Drop the column *Summary*
2. Drop all rows having a *Ratings* value equal to 3.
3. Replace value of Rating to 1 for all *Ratings* greater than 3.
4. Replace value of Rating to 0 for all *Ratings* smaller than 3.

The reason column *Summary* is dropped because one, that is not required as we plan to predict the sentiment by reading the review and second the count of *Summary* is not equal to the count of column *Ratings* i.e. some values of *Summary* are missing.

	Ratings	Review
0	3	Gave this to my dad for a gag gift after direc...
1	5	I hope a lot of people hear this cd. We need m...
2	5	I'm reading a lot of reviews saying that this ...
3	4	The music of Yasunori Misuda is without questi...
4	5	Probably the greatest soundtrack in history! U...

Figure 7 Dataset Preview before data manipulation

The reason all the rows having a *Ratings* value equal to 3 are dropped because we don't want to deal with neutral reviews.

	Ratings	Review
1	5	I hope a lot of people hear this cd. We need m...
2	5	I'm reading a lot of reviews saying that this ...
3	4	The music of Yasunori Misuda is without questi...
4	5	Probably the greatest soundtrack in history! U...
5	5	There's a reason this CD is so expensive, even...

Figure 8 Dataset preview after removing neutral reviews

Finally, I turned this classification into a binary classification that just predicts whether the review is positive or negative. Value 1 is positive, and Value 0 is negative.

After the data manipulation this how the dataframe looks like.

	Ratings	Review
1	1	I hope a lot of people hear this cd. We need m...
2	1	I'm reading a lot of reviews saying that this ...
3	1	The music of Yasunori Misuda is without questi...
4	1	Probably the greatest soundtrack in history! U...
5	1	There's a reason this CD is so expensive, even...

Figure 9 Dataset preview after converting Ratings to 0 and 1

After these changes we are left with 2400000 training rows 520000 training rows.

Let's have a look of number of positive and negative review in our training dataset.

Ratings	Review
0	1200000
1	1200000

Figure 10 Count of positive and negative Ratings

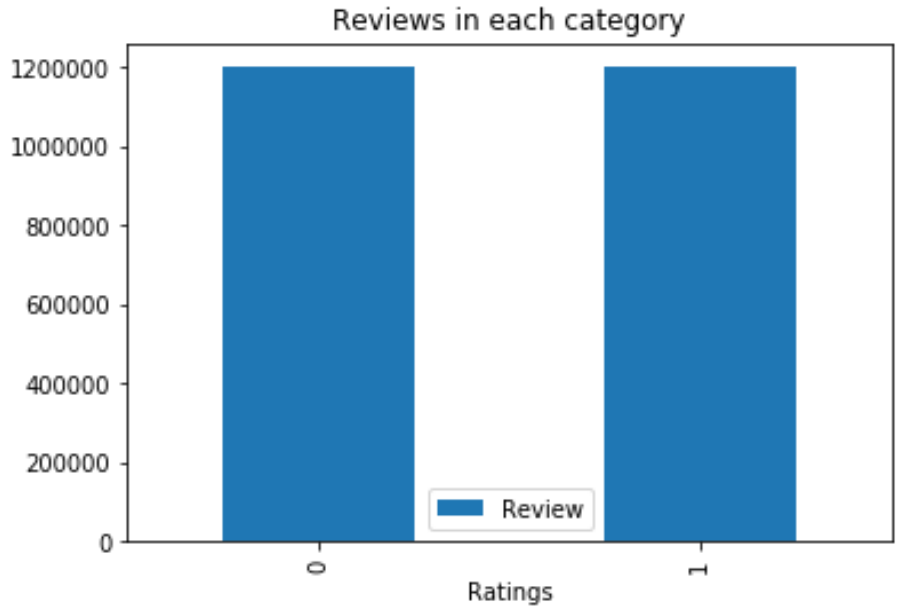

Figure 11 Graph of count of Ratings

Even after all these pre-processing the 2.4 Million rows are too much data to process to store in memory. While working on the project I faced this issue where training time was going in days and reaching my memory limits while some computations.

So, I have randomly picked 150000 training rows and 26000 testing rows. All the further steps will be taken on this data. The resultant subset of data also has the 50-50% composition of positive and negative reviews.

Final data is 150000 training rows and 26000 testing rows.

Implementation

The implementation process can be split into two main stages:

1. Language model
    a. Preparation of data for Language model.
    b. Language model data cleaning.
    c. Loading weights of pre-trained model.
    d. Training the language model.
    e. Saving the language model.
2. Classifier
    a. Preparation of data for Classifier
    b. Creating a classifier from trained encoder of language model.
    c. Training the classifier
    d. Predict the sentiment.

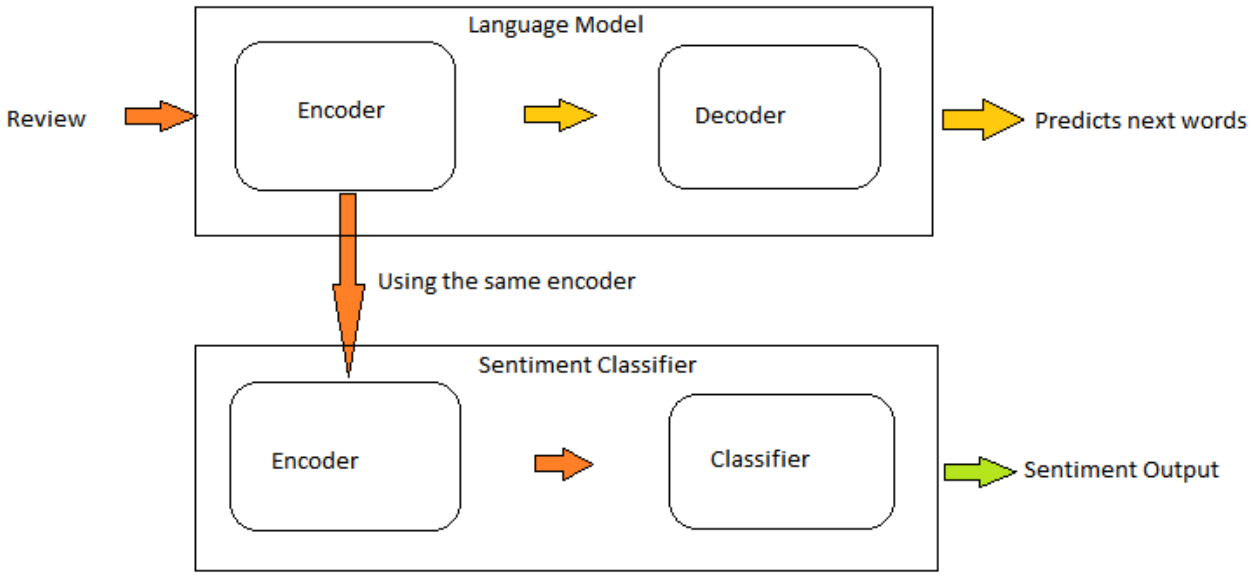

Figure 12 Model Structure

NOTE: The whole training process of this notebook took me around 20 hours. So, weights are saved after every few epochs.

During the first stage, first all the text(strings) from train and test data were extracted into a single list and then split into a train and test set of 90 and 10% respectively. The reason of merging both the was to get as

much as text possible to train the language model on. The sole purpose of language model is understanding the structure of English language and to predict the next set of words given a word or phrase as input.

The process of tokenization is to split the text into sperate tokens and assign them unique index (integers) so that our model can use that. So, we use spacy library to tokenize and clean our data. Extra spaces, tab chars, new line in chars and other characters were replaced with standard one.

Next step before re-training the pre-trained model it to match the vocab size and their indexes. For words that don't exist in the pre-trained model we assign a mean of weights.

Then we re-trained the language model with our new vocab and use accuracy as a metric to measure its performance. While training initially we freeze everything expect the last embedding layer so that the missing tokens are initialized with mean weights properly.

```
SequentialRNN(
  (0): RNN_Encoder(
    (encoder): Embedding(55702, 400, padding_idx=1)
    (encoder_with_dropout): EmbeddingDropout(
      (embed): Embedding(55702, 400, padding_idx=1)
    )
    (rnns): ModuleList(
      (0): WeightDrop(
        (module): LSTM(400, 1150)
      )
      (1): WeightDrop(
        (module): LSTM(1150, 1150)
      )
      (2): WeightDrop(
        (module): LSTM(1150, 400)
      )
    )
    (dropouti): LockedDropout()
    (dropouths): ModuleList(
      (0): LockedDropout()
      (1): LockedDropout()
      (2): LockedDropout()
    )
  )
  (1): LinearDecoder(
    (decoder): Linear(in_features=400, out_features=55702, bias=False)
    (dropout): LockedDropout()
  )
)
```

Figure 13 Language Model Layers

This is how our language model neural network look like. A sequential model with a linear decoder.

In the stage second of our implementation first we extract the tokens from the train and test dataset and then we assign the tokens the same integer values that are assigned while training the language model. This is done by reading the integer to string dictionary that was saved earlier.

The new classifier is the is a pooling classifier on top of the language model encoder. Then I trained the model initial by unfreezing only the layer and then last 2 layers and then finally all layers.

The total training time required was more than 12 hours.

```
SequentialRNN(
  (0): MultiBatchRNN(
    (encoder): Embedding(55702, 400, padding_idx=1)
    (encoder_with_dropout): EmbeddingDropout(
      (embed): Embedding(55702, 400, padding_idx=1)
    )
    (rnns): ModuleList(
      (0): WeightDrop(
        (module): LSTM(400, 1150)
      )
      (1): WeightDrop(
        (module): LSTM(1150, 1150)
      )
      (2): WeightDrop(
        (module): LSTM(1150, 400)
      )
    )
    (dropouti): LockedDropout()
    (dropouths): ModuleList(
      (0): LockedDropout()
      (1): LockedDropout()
      (2): LockedDropout()
    )
  )
  (1): PoolingLinearClassifier(
    (layers): ModuleList(
      (0): LinearBlock(
        (lin): Linear(in_features=1200, out_features=50, bias=True)
        (drop): Dropout(p=0.2)
        (bn): BatchNorm1d(1200, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      )
      (1): LinearBlock(
        (lin): Linear(in_features=50, out_features=2, bias=True)
        (drop): Dropout(p=0.1)
        (bn): BatchNorm1d(50, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      )
    )
  )
)
```

Figure 14 Classifier Layers

This is how our classifier neural network looks like. A sequential model with a pooling linear classifier.

Refinement

Techniques in effect for efficient training of the model are:

1. **Cyclical Learning Rates** for training neural networks: in this technique training with cyclical learning rate instead of fixed values achieves more classification accuracy. This is performed using *lr_find()* method.

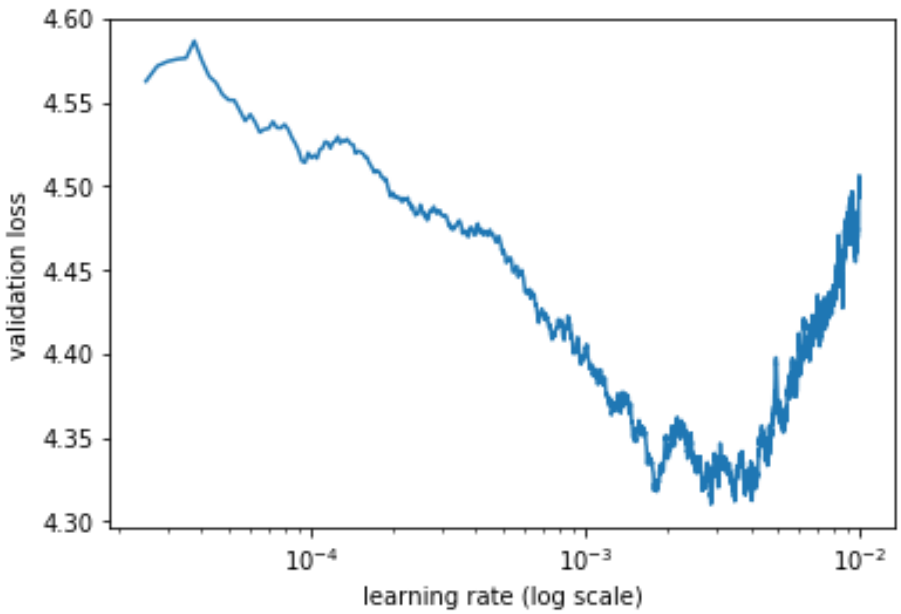

Figure 15 Graph of finding optimal Learning Rate

2. Using a **mini batch size** helps parallel processing of the GPU efficiently.
3. Use of **stochastic gradient with restart** (SGDR). The basic idea is that as you get closer and closer to the spot with minimal loss, you may want to start decrease the learning rate (taking smaller steps) to get exactly the right stop. This is specified with the *cycle_len* (int) parameter passed while calling *model.fit()* method.
4. Use of **differential learning rate** to train different layer of model with different rate results in better accuracy. This is passed as an array of learning rates while initializing model. General idea is to use a lower learning rate for the earlier layers and gradually increase it in the latter layers.

These techniques are described in various papers which are referenced in the Papers section of the report.

Also, this transfer learning technique of training is inspired from the Jeremy Howard tutorial. Link to the video is here.

IV. Results

Model Evaluation and Validation

Both models have been validated with the 26000 reviews that were separated from the dataset as Test dataset. Below are the classification results.

Logistic Classifier results

Accuracy is 86.1923076923077

	precision	recall	f1-score	support
0	0.87	0.86	0.86	13040
1	0.86	0.87	0.86	12960
micro avg	0.86	0.86	0.86	26000
macro avg	0.86	0.86	0.86	26000
weighted avg	0.86	0.86	0.86	26000

Figure 16 Classification report of Logistic Classifier

Confusion Matrix

TP	11172	FP	1868
TN	1722	FN	11238

Transfer Learning Results

Accuracy is 92.84615384615384

	precision	recall	f1-score	support
0	0.94	0.92	0.93	13040
1	0.92	0.94	0.93	12960
micro avg	0.93	0.93	0.93	26000
macro avg	0.93	0.93	0.93	26000
weighted avg	0.93	0.93	0.93	26000

Figure 17 Classification report of Transfer Learning Classifier

Confusion Matrix

TP	11981	FP	1059
TN	801	FN	12159

Justification

So, as we can see we have a test accuracy of 92% with transfer learning and test accuracy of 86% when using logistic classifier. So, the results are better than the benchmark results that were reported earlier.

Let's try simple statements as input to our model an see the sentiment output. Note: 1 is a positive sentiment and 0 a negative sentiment.

```
In [243]: 1 generate_prediction(m," This was a good product")
Out[243]: array([1])

In [244]: 1 generate_prediction(m," This was a bad product")
Out[244]: array([0])

In [245]: 1 generate_prediction(m,"Well this is the best quality")
Out[245]: array([1])
```

Figure 18 Prediction of model on general sentences

Looks its working great. Now let's try with some real picked reviews from amazon in the next section.

V. Conclusion

Free-Form Visualization

The following reviews are picked up directly from amazon website.

Let's see about this absurd comment as input to model.

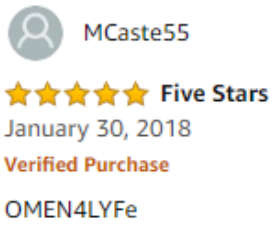

Figure 19 Amazon Review 1

Our model generates a positive sentiment. 😊 Matches the five-star rating given by customer.

```
In [249]:  1  generate_prediction(m,"OMEN4LYFe")
Out[249]: array([1])
```

Figure 20 Amazon Review 1 model output

A big review

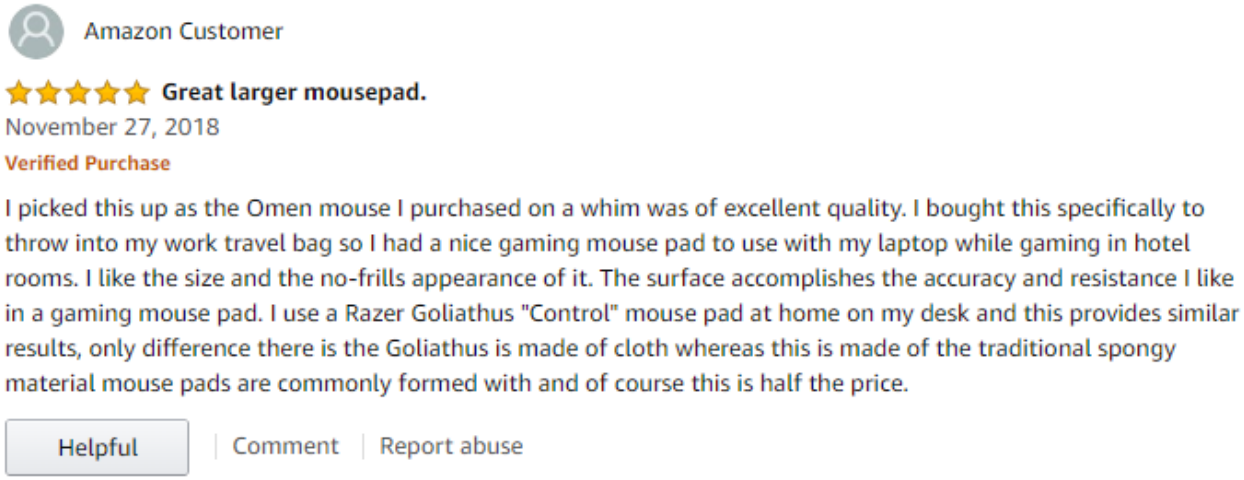

Figure 21 Amazon Review 2

Output is a positive sentiment.

```
In [248]:
generate_prediction(m,"I picked this up as the Omen mouse I purchased on \
a whim was of excellent quality. I bought this specifically to throw into my \
work travel bag so I had a nice gaming mouse pad to use with my \
laptop while gaming in hotel rooms. I like the size and the no-frills appearance \
of it. The surface accomplishes the accuracy and resistance I like in a gaming \
mouse pad. I use a Razer Goliathus \"Control\" mouse pad at home on my desk and\
this provides similar results, only difference there is the Goliathus is made of\
cloth whereas this is made of the traditional spongy material mouse pads are commonly\
formed with and of course this is half the price.")

Out[248]: array([1])
```

Figure 22 Amazon Review 2 model output

A negative Review

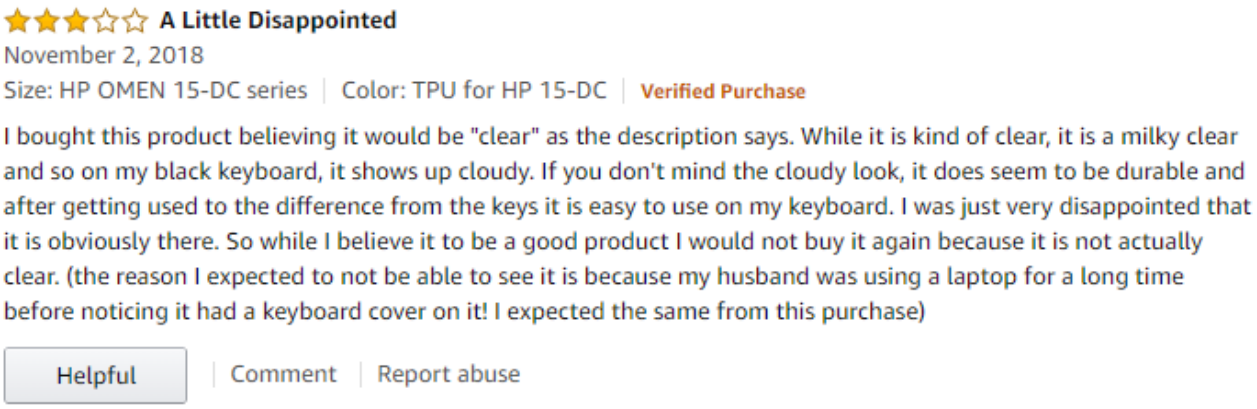

Figure 23 Amazon Review 3

```
In [256]:
generate_prediction(m,"I bought this product believing it would be \"clear\" \
as the description says. While it is kind of clear, it is a milky clear and so\
on my black keyboard, it shows up cloudy. If you don't mind the cloudy look, \
it does seem to be durable and after getting used to the difference from the \
keys it is easy to use on my keyboard. I was just very disappointed that it is \
obviously there. So while I believe it to be a good product I would not buy it\
again because it is not actually clear. (the reason I expected to not be able\
to see it is because my husband was using a laptop for a long time before \
noticing it had a keyboard cover on it! I expected the same from this purchase")

Out[256]: array([0])
```

Figure 24 Amazon Review 3 model output

Reflection

Overall the project can be summarized using the following steps:

1. Initial problem and relevant, public datasets were found.
2. The data was downloaded and pre-processed according to need.

3. A benchmark classifier was created.
4. Main solution of the problem was created using transfer learning.
    a. It involved creating a language model to understand the structure of English language
    b. Classifier to predict the sentiment based on understanding the English language.
5. Prediction were made using the test dataset with 92% accuracy.
6. Prediction were also made using reviews picked up directly from amazon website.

Overall this project summarizes a great way of using transfer learning for a wide range of application, the same language model can be used for word prediction or style text generation.

During the implementation I learnt that

1. While training large networks a GPU machine is always a big help otherwise the training time counts in days.
2. Proper data cleaning and visualization is very much necessary step as it helps you to make better decision like which metric should you use for your model evaluation.
3. Using a library like Pytorch has many advantages as saving the model states and their parts according to future needs, this helps you save time that could be wasted training the model from start every time.
4. Specially the step 4 was the most difficult step, as I had to familiarize myself with many new libraries like Fastai and Pytorch.

Improvement

1. Due to memory and training time limitation I did not use the whole dataset to model the solution, there is small probability that the model performance can increase.
2. Currently I pre-trained with a forward encoder named *"fwd_wt103.h5"*, We can create one more classifier using a backward encoder using *"bwd_wt103.h5"*. That's a backward English language model that learns to read English backward. So, if we redo this whole thing, put all the documents in reverse, and change this to backward, we now have a second classifier which classifies things by positive or negative sentiment based on the reverse document. If we then take the two predictions and take the average of them, basically we can have a bi-directional. This can result in better accuracy.
3. In dataset pre-processing we can replace of all the numerical values such prices of products and quantity of product with a specific number like 0. Those number don't really matter while predicting the next word when using the language model. Se we can get a better accuracy while training language model.

VI. Citations

1. https://github.com/salesforce/awd-lstm-lm
2. https://blog.einstein.ai/the-wikitext-long-term-dependency-language-modeling-dataset/
3. https://drive.google.com/drive/folders/0Bz8a_Dbh9Qhbfll6bVpmNUtUcFdjYmF2SEpmZUZUcVNiMUw1TWN6RDV3a0JHT3kxLVhVR2M
4. https://blog.einstein.ai/the-wikitext-long-term-dependency-language-modeling-dataset/
5. https://scikit-learn.org/stable/
6. https://scikit-learn.org/stable/
7. https://docs.fast.ai/text.models.html#AWD_LSTM
8. Paper: https://arxiv.org/pdf/1708.02182.pdf
9. https://en.wikipedia.org/wiki/Sensitivity_and_specificity
10. https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html
11. https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

VII. Papers

1. Sequence to Sequence Learning

   Despite their flexibility and power, DNNs can only be applied to problems whose inputs and targets can be sensibly encoded with vectors of fixed dimensionality. Sequences pose a challenge for DNNs because they require that the dimensionality of the inputs and outputs is known and fixed. In this paper, we show that a straightforward application of the Long Short-Term Memory (LSTM) architecture can solve general sequence to sequence problems.

2. Regularizing and Optimizing LSTM Language Models

   We explored additional regularization techniques that aimed to improve data efficiency during training. Use of randomized-length backpropagation through time (BPTT), embedding dropout, activation regularization (AR), and temporal activation regularization (TAR).

3. Cyclical Learning Rates for Training Neural Networks

   Instead of monotonically decreasing the learning rate, this method lets the learning rate cyclically vary between reasonable boundary values. Training with cyclical learning rates instead of fixed values achieves improved classification accuracy without a need to tune and often in fewer iterations.