CS230

Comprehend Sentiments in Product Reviews (Natural Language Processing)

Yecheng Yang
yechengy@stanford.edu

Aditya Thakkar
adityat@stanford.edu

Huaping Gu
hpgu@stanford.edu

1 Problem Description

Social distancing during the current pandemic has prompted consumers to buy products online. One of the important factors that help consumers judge quality of a product or service without being able to tangibly review the product is customer reviews. In addition, reviews also help product seller to get feedback from their customers. We propose using Deep Learning to analyze sentiments of product reviews that can help product sellers and consumers to take focused approach on the reviews that matter to them. The input to our deep learning system will be product review comprised of product rating and text description of the review. The input will be processed to generate embedding, which will be passed to Neural Network model. The output will be rating in the range of 1-5 signifying score of the review.

2 Dataset Exploration

The product reviews are typically comprised of product ratings and text description. Product review from Amazon in following categories - Books, Movies, Electronics, Kitchen and Food, have been made openly available at [1], and [2]. We will refer to data from [1] as JHU data and data from [2] as Food Review data.

JHU Dataset comprises of 8,000 reviews, while Food Review dataset comprises of 568,454 reviews. Looking at the rating distributions, JHU data is much more evenly distributed compared to Food Review data, which has an overwhelmingly ratio for 5-star reviews as seen in [Figure 1] *Refer to Appendix*.

The mean review length for JHU data set is 135 words while the mean is only 75 words for Food Reviews *(refer to Table 1 and Figure 2 in Appendix)*. Moreover, standard deviation for JHU Dataset is 150 words compared to 46 words for Food Review Dataset. Similarly, their 75th percentile is 162 and 83 words respectively. This pattern continues to hold for even longer reviews. We also noticed there are reviews with min 1 word and max of 3393 words in case of JHU Dataset, and reviews with min 21 words and maximum 176 words in case of Food Review Dataset. As observed, the vast majority of the reviews have length between 21 and 200 words, and a closer look into review length revealed that food reviews are in general more concise compared to reviews for other products. For details please check [Table 1]. The vocabulary consists of 46,815 words.

Beyond that, we have also conducted some standard pre-processing for the reviews. All reviews are converted to lower case, all trailing spaces and punctuation are removed. In addition, the words of every review have been tokenized i.e. converted into an integer value; the result of which is list of tokens (integers) for every review.

Last but not least, instead of having only 0 and 1 value as labels for classification, product rating ranging from 1-5 are used as the label value. This will ensure model is able to predict the sentiment of the review and assign score ranging in value 1-5.

3 Proposed Methodology

Long Short Term Memory (LSTM) Neural Network will be used at the core of the algorithm. LSTM are known to regulate flow of information using mechanism called gates [3]. This helps in learning long-term dependencies needed for problems such as sentimental analysis. Python programming language will be used to pre-process the data and tokenized text reviews i.e. convert text to integer. In addition, PyTorch will be used to generate embedding of the tokenized text; specifically, torch.nn.Embedding module of the PyTorch framework will be used to generate embedding. Dimension of the embedding will be treated as an hyperparameter and selected in the range of 50-300 as this range has been commonly used in speech recognition applications such as [4] and [5]. In addition, the deep learning model using LSTM neural network will be developed using PyTorch framework as well.

3.1 Model

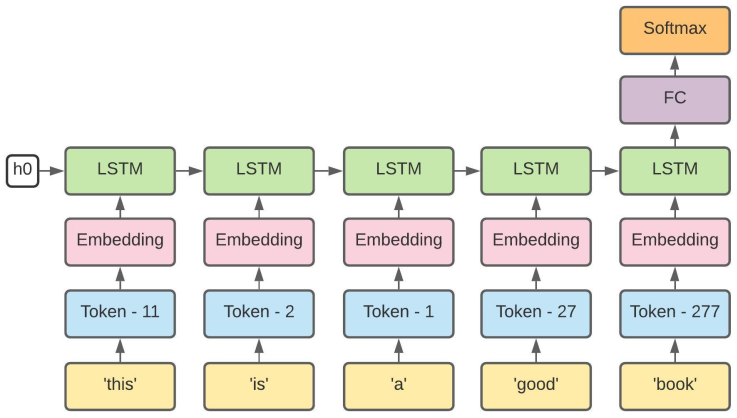


Figure 3: Sentiment Analysis Model

Refer to Figure 3. In the Forward Propagation step, embedding will be input to the LSTM neurons along with cell state from previous LSTM state. Output of LSTM will passed to a fully connected layer with Softmax activation function in order to generate the output. Softmax because we would like the output of the model to classify the outcome in the range of 1-5. Loss function used will be categorical cross entropy function, which is commonly used with Softmax activation function. In the Back Propagation step, Adam optimizer will be used with Beta1 value of 0.9 and Beta2 value of 0.999. In addition, learning rate will be an hyperparameter and will be initialized to the value 0.001. Lastly, considering the fact that there are more than 500K examples in the dataset, mini-batch algorithm will be used to accelerate learning process. Mini-batch size will be a hyperparameter. However, size of 512 will be used as the initial mini-batch size. 512 is chosen as the initial mini-batch size under assumption that 1126 batches generated, each of size 512, would be computationally inexpensive compared to smaller mini-batch size, and it would fit the 16Gb RAM available in the p2.Xlarge EC2 instance on AWS with GPU. Lastly, depending on whether there is over-fitting, dropout techniques such as L2 regularization or Dropout will be used in order to reduce the high variance.

3.2 High Level Algorithms

There are 2 folds to the architecture; first ($Algorithm1$) is to load the data from files, parse the data from original format to dictionary and pre-process the data into format that can be inputed to the model i.e. remove punctuation, convert text to lower case, shuffle the dataset (which includes labels as well), create vocabulary, convert vocabulary to tokens, tokenize review text, create label set, pad and truncate the tokenized reviews, and convert the tokenized reviews and label set into numpy array. Second step ($Algorithm2$) is to split the dataset into train, validate and test sets, batch it and convert it to PyTorch tensors, create the model, train and validate the model, and finally test the model.

```
Algorithm 1: Data Processing
Data: JHU and Food Review Datasets
Result: Parsed, Padded and shuffled tokenized dataset and label set in numpy array format
Function loadData(path)is
    Load datasets from files;
    Parse;
    Shuffle Dataset (which includes labels as well);
    Create Vocabulary;
    Tokenize Vocabulary;
    Encode (tokenize) review text;
    Create Label Set;
    Padding and Truncation;
```

```
Algorithm 2: Main Loop
Data: Training, Validate and Test Dataset
Result: Trained and validated model, and test results
Split Dataset into Train, Validate and Test set;
Batch datasets and convert to PyTorch Tensors;

Create ReviewSentimentLSTM model object;
Create Optimizer and CrossEntropy Criterion;
Create CUDA device if CUDA is available;
while epochs not finish do
    train model while batches do
        counter + 1 forward propogate on the batch;
        calculate loss();
        Adam Optimizer();
        clip_grad_norm();
        optimizer();
        if counter mod validate_counter then
            validate the model using Validate dataset;
        else
            continue training
test model;
```

4 Hyper parameters Tuning

Following are the Hyper Parameters for the architecture.

Hyper Parameter	Description	Initial Value
data_split_ratio	training data ratio (remaining is evenly divided into Validate and Test set)	0.92
batch_size	number of examples in 1 batch	512
output_dim	number of output classes	5
embedding_dim	embedding dimension, empirical data from related paper	300
dropout	dropout rate	0.1
n_layers	how many hidden layers	1
n_direction	1 = unidirectional LSTM and 2 = bidirectional LSTM	1
learning_rate	Learning rate in training	0.001
hidden_dim	number of LSTM hidden units per layer.	176

Table 2: Hyper Parameters and initial value

4.1 Hyperparameter Tuning Results

Table 3 shows the test results with various Hyper Parameter (HP) Tuning. All the testcases were executed for 2 Epochs and the results are from Test Set execution

Test No.	Description	Epoch Duration (sec)	Avg Test Set Loss	Test Set Accuracy
1.	Initial HP in Table 2	195	0.019119	0.990892
2.	Initial HP in Table 2 + n_layers = 2	340	0.021294	0.989722
3.	Initial HP in Table 2 + n_layers = 3	494	0.018231	0.991196
4.	Initial HP in Table 2 + n_layers = 4	650	0.016701	0.991890
5.	Initial HP in Table 2 + n_layers = 5	788	0.064367	0.988767
6.	Test 4. + n_direction = 2	1480	0.0187490	0.990676
7.	Test 6. + batch_size = 256	1958	0.0178740	0.990545
8.	Test 6. + embedding_dim = 400	1546	0.018730	0.990199
9.	Test 8. + hidden_dim = 256	3321	0.018730	0.990199
10.	Test 8. + hidden_dim = 75	299	0.019239	0.991196
11.	Test 8. + hidden_dim = 128	699	0.016768	0.991803
12.	Test 11. + learning_rate = 0.034	688	0.133538	0.991023
13.	Test 11. + learning_rate = 0.064	668	0.802661	0.665293
14.	Test 11. + batch_size = 256	997	0.016700	0.992150

Table 3: Hyper Parameter Tunning Results with Test Set

4.1.1 Observations

Testcase 14 - Initial HP in Table 2 + n_layers = 4 + n_direction = 2 + embedding_dim = 400 + hidden_dim = 128 + batch_size = 256, gave the most optimum results. Following Figure 4 shows Validate Set Loss trend and Validate Set Accuracy trend for Testcase 14, which gives most optimum results.

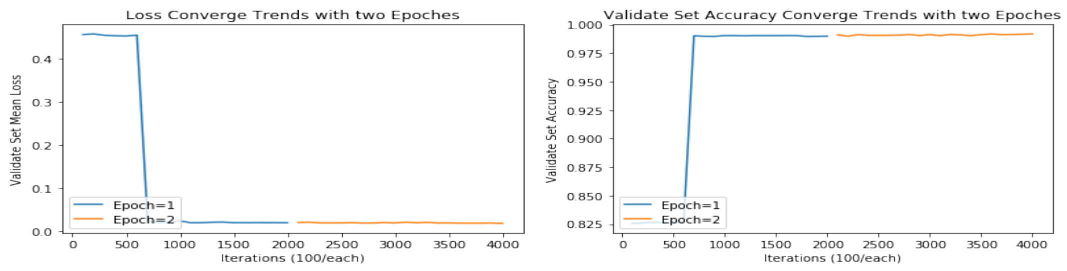


Figure 4: Validate Set Loss and Accuracy Trends

4.1.2 Analysis

Epoch: As seen from the results above Every epoch for the chosen Hyperparameters took approximately 997 seconds to complete. In other words, it took 1994 seconds or 33 minutes to train the model with 2072 Training batches and 91 Validate Set batches, each of size 256 examples. In addition, from Figure 4 it can be observed that at approximately 700 batch mark, we receive Mean Validate Set Loss of 0.023166310444639827 and accuracy of 0.9901986122131348. The loss and and accuracy marginally reduce and increase respectively for rest of the iterations. We can infer that Human Level performance in terms of accuracy is about 0.9901986122131348. Furthermore, we can infer that Early Stopping is not needed as the loss is not overwhelmingly oscillating.

Learning Rate: As seen in Table 3, with Learning Rate set to 0.034 (Testcase 13) or 0.064 (Testcase 14), we are not able to reach the optimal point. In other words, it can be inferred that loss has been oscillating. As observed in the test results, with learning rate of 0.001, model was able near optimum point for rest of the test cases.

Hidden Layers: From Table 3, in Testcases 1-5, it can be seen that performance of the model in terms of loss and accuracy starts to drop as number of LSTM layers are increased beyond 4. In addition, performance was most optimal with 4 hidden layers.

Hidden Dimensions: Hidden Dimension or number of LSTM units at each layer was chosen to be 176 in the initial tests because majority of the dataset comes from Food Review dataset and as per Table 1 99% of Food Reviews were 176 words long. From testcase 9 (where hidden_dim was 256 and batch_size was 256) and 14 (where hidden_dim was 128 and batch_size was 256) , it can be seen that as number of LSTM units within a hidden layer increase, Epoch duration also increases. It can be inferred that computation resources required to train the model increases as number of hidden_dim increases. Lastly, it can be observed that reducing the hidden_dim from 176 to 128 did not affect the performance.

Embedding Dimension: As seen in testcase 8 of Table 3, increasing Embedding Dimension from 300 to 400 did not have drastic impact on the performance of the model.

Dataset: As discussed in Dataset Exploration section, Food Reviews dataset, which is the largest dataset out of the 2 used for training the model, has overwhelming number of 4 and 5 star reviews. In addition, it can be seen from Figure 4, the model nears to the optimal point fairly quickly at approximately 700 batch mark. One of the reasons why this could be happening is because the dataset is not balance i.e. there are more number of review with 4 or 5 ratings compared to review with ratings of 1-3. It would be desireable to train this model with dataset consisting of reviews whose ratings are even distribution across the 5 classes.

4.2 Tuning Decision

Hyper-Parameter tuning is a very heuristic and empirical task. It not only requires experience and knowledge in Deep Learning field, but also requires insights into the datasets under training. In addition, methodical approach is required to fine tune the hyper parameters. We initiate hyper parameters tuning by choosing batch_size (512) that was large enough to fit in the memory and reduce number of iterations per test execution. Next we, started tuning performance by varying number of hidden_layers and then varied parameters such as hidden_dim, n_direction, embedding_dim, learning_rate and batch_size.

5 Next Steps

As a next step, the goal of the team was to implement Transformer model that relies on attention mechanism to execute dependencies between input and output and compare performance of it with that of LSTM model. Team was able to implement Transformer model using BERT; however, roadblocks were faced as it BERT code was unable to use GPU in tokenizing large set of data resulting in taking long time in tokenizing data on CPU using BERT. In addition, once tokenization using BERT was successful, there were embedding Index Out of Range that blocked the progress of training using Transformers model.

In addition, it would be desireable to use more balanced dataset as the current dataset has overwhelming number of 4 and 5 star reviews. Moreover, although current model is trained on reviews from books, DVDs, electronics, house appliances and food products, it would be desireable to use this model as transferred learning to other product sentimental analysis such as wine review, car review etc.

6 Contributions

Great team work in this final project although we are fully remote in this pandemic period. Yecheng and Aditya proposed the datasets and topic selection. Aditya did the valuable dataset distribution analysis. Aditya and Yechang proposed the overall methodology and the Deep Learning architecture, Huaping helped iterate into steps.

Aditya worked on many lines of code for the $LSTM$ solution, triaging and debugging. If we had more time, would polish and triage more on the $Transformers$ solution which both Aditya and Yecheng spent whole days to tune.

Huaping worked on the AWS GPU environment setup, and executed majority of the tests during the Hyper Parameter tuning. Test results analysis are done by all of the members.

Finally we really appreciated all the helps we got from the section videos, piazza replies and TA Office Hours. Huge thanks for the overall TA teams and professor Andrew and instructor Kian, special thanks to our project mentor Avoy Datta.

References

[1] J. Blitzer, M. Dredze, F. Pereira, "Biographies, Bollywood, Boom-boxes and Blenders: Domain Adaptation for Sentiment Classification." Association of Computational Linguistics (ACL), 2007

[2] Amazon Fine Food Review https://www.kaggle.com/snap/amazon-fine-food-reviews

[3] S. Hochreiter, and J. Schmidhuber. "LONG SHORT-TERM MEMORY." Neural ComputationNovember 1997

[4] - Yoon Kim. 2014. Convolutional neural networks for sentence classification. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). Association for Computational Linguistics, pages 1746–1751.

[5] - Aditya Joshi, Vaibhav Tripathi, Kevin Patel, Pushpak Bhattacharyya, and Mark Carman. 2016. Are word embedding-based features useful for sarcasm detection? In Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics, pages 1006–1011.

[6] T. Young, D. Hazarika, S. Poria and E. Cambria, "Recent Trends in Deep Learning Based Natural Language Processing [Review Article]," in IEEE Computational Intelligence Magazine, vol. 13, no. 3, pp. 55-75, Aug. 2018, doi: 10.1109/MCI.2018.2840738.

[7] Y. Wang, M. Huang, X. Zhu, and L. Zhao, "Attention-based LSTM for aspect-level sentiment classification," in Proc. Conf. Empirical Methods Natural Language Processing, 2016, pp. 606–615.

[8] Y. Ma, H. Peng, and E. Cambria, "Targeted aspectbased sentiment analysis via embedding commonsense knowledge into an attentive LSTM," in Proc. Association Advancement Artificial Intelligence Conf., 2018, pp. 5876–5883.

[9] D.S. Sachan, M. Zaheer, R. Salakhutdinov, "Revisiting LSTM Networks for Semi-Supervised Text Classification via Mixed Objective Function", arXiv: 2009.04007

Appendix

Review Rating Distribution

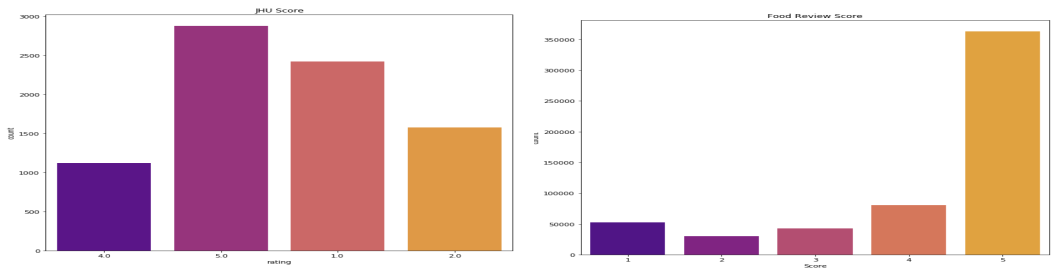


Figure 1: Review Rating Distribution

Review Length Distribution

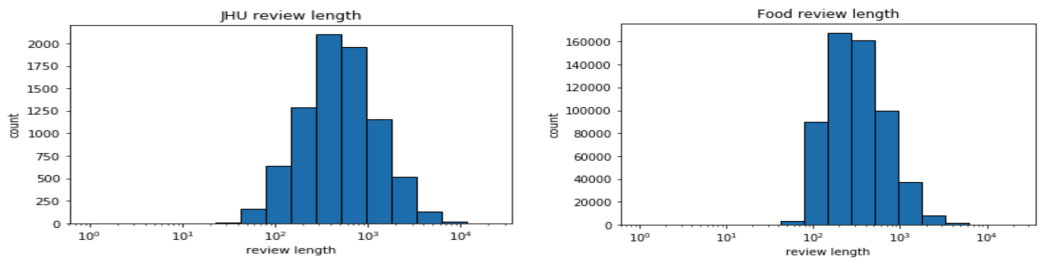


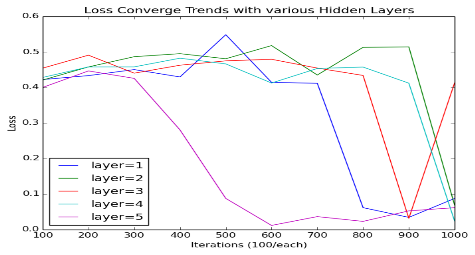
Figure 2: Review Length Distribution

Parameters	JHU Dataset	Food Review Dataset
count	8000.000000	568454.000000
mean	135.251500	75.992474
std	150.860587	46.694354
min	1.000000	21.000000
25%	49.000000	44.000000
50%	90.000000	59.000000
75%	162.000000	83.000000
99%	739.010000	176.000000
max	3393.000000	176.000000

Table 1: Statistics for JHU and Food Review Dataset

Statistics for JHU and Food Review Dataset

Hidden Layer Hyper Parameter Tuning



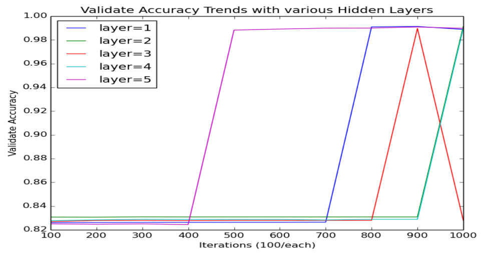
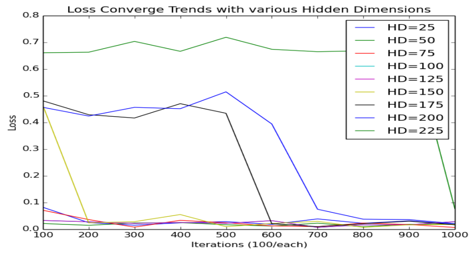


Figure 3: Hidden Layer Hyper Parameter Tuning

Hidden Dimension Hyper Parameter Tuning



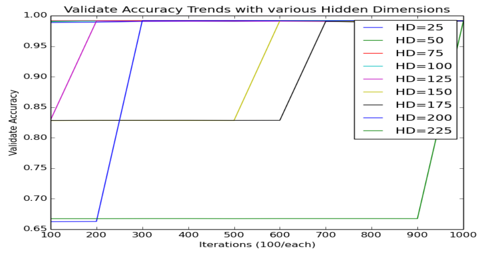
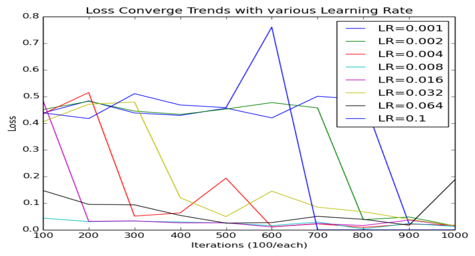
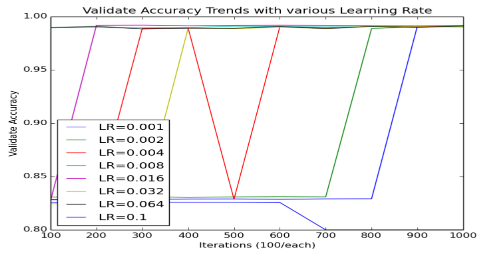


Figure 4: Hidden Dimension Hyper Parameter Tuning


Loss Converge Trends with various Learning Rate
Loss
Iterations (100/each)
LR=0.001
LR=0.002
LR=0.004
LR=0.008
LR=0.016
LR=0.032
LR=0.064
LR=0.1


Validate Accuracy Trends with various Learning Rate
Validate Accuracy
Iterations (100/each)
LR=0.001
LR=0.002
LR=0.004
LR=0.008
LR=0.016
LR=0.032
LR=0.064
LR=0.1