Amazon Product Review Classification using LSTM-Variants

Syed Saad Hussain
Department of Mechanical and Industrial Engineering
Ryerson University
Toronto, Canada
saad2.hussain@ryerson.ca

***Abstract*—There have been many recent advancements in applied deep learning as it pertains to sentiment analysis. For this project, LSTM-variants and collaborative deep learning models are implemented for the purpose of classifying Amazon product reviews for sentiment extraction. Results show that CNN-biLSTM models with attention layer perform the best for the classification tasks on this data set, and that even simple bi-LSTM models outperform the base LSTM model.**

I. Introduction

In today's competitive e-commerce environment, product reviews can provide key insights on consumer sentiment of their purchases. Although a rating on a five-star scale can provide some high-level insights, it is more interesting to analyze textual feedback as it can provide the seller further context behind levels of customer satisfaction. This feedback can be used by the seller to improve their product quality and offerings, and better gage consumer sentiment. For my project, I applied deep learning methods on Amazon product review comments. My goal was twofold:

- Task 1: Predict product rating, given the textual review
- Task 2: Predict whether others will find the review useful, given the textual review

Both tasks are essentially text classification problems. These two classification tasks were performed independently, although in future works it would be interesting to see if multi-task learning can be utilized to achieve superior performance given the similarities between the tasks.

Much research has been done in recent years in the field of applied deep learning for natural language processing (NLP) applications. Many groups submit deep learning-based solutions for NLP problems for many of the annual tasks outlined by the International Workshop on Semantic Evaluation (SemEval), which poses several open problems for academics and researchers to tackle.

Before the rise of deep learning, NLP researchers applied more traditional machine learning techniques such as support vector machine (SVM) and Naïve Bayes. The drawback to using these techniques is that it requires manual feature engineering to achieve high accuracy [1]. For example, the performance of an SVM depends heavily on the kernel, and the algorithmic complexity poses computational constraints [1]. Naïve Bayes assumes independence between features and fails to model long term dependencies [1]. On the contrary, more recent publications have shown that deep learning methods are robust enough in text classification tasks to not require unnecessary data processing such as feature engineering [1].

During the literature review, I searched for papers that specifically addressed text classification tasks using deep learning methods. Li *et al.* employed a collaborative model with bidirectional long Short-term memory (bi-LSTMs) and convolutional neural networks (CNNs) to perform a classification task on Chinese news articles [2]. Kumar *et al.* also used a collaborative model with bidirectional LSTM and CNNs but added an attention mechanism for irony detection on tweets [3]. Mathapati *et al.* utilized a LSTM-CNN model for sentiment analysis on IMDB movie review data [1].

In the following section, an in-depth summary on the findings of my literature review will be provided.

II. Literature Review

In their paper, Li *et al.* leveraged a Bi-LSTM-CNN model to classify Chinese news articles. They trained their word embeddings using Word2Vec using continuous bag of words (CBOW) for their n-gram model. They utilized Word2Vec due to its ability to learn localized word associations for a given corpus. For their neural model, they opted to use bidirectional LSTMs with convolutional layers. This leverages the power of the forward LSTM, which processes the corpus in one direction, and backward LSTM that processes the corpus in the opposite direction and allows the model to account for the complete semantic structure of the text. A convolutional layer was sandwiched between the bi-LSTM layers to concatenate the context vectors. A linear transformation was applied to the tanh layer and the result was sent to the max-pooling layer. A softmax function was used to transform the output into a probability. The proposed model outperformed CNN, LSTM, SVM, and TF-IDF.

In their paper, Kumar *et al.* perform a binary text classification task to detect sarcasm on Twitter. They achieved this through the integration of an attention mechanism with their Bi-LSTM model before passing it through a convolutional neural network. The attention layer accounted for the fact that not all words have equal contribution to the overall semantic structure. They also used auxiliary features such as counts of punctuation to boost their model performance. GloVe word embeddings were used.

In their paper, Mathapati *et al.* compared several neural models to discover which one provides superior performance in binary classification tasks on consumer movie reviews. They compared traditional methods such as Naïve Bayes with multilayer perceptron (MLP), LSTM, CNN, and CNN-LSTM. In their experimentation, they found that all neural models significantly outperformed Naïve Bayes, and that the CNN model performed the best. For the convolutional filter, they discovered that limited window sizes of 3 or 4 words had the best performance, possibly due to CNN's inability to capture long term dependencies in sequential data. From their experimentation, it was concluded that MLPs do not store temporal dependencies in memory which is why they perform the poorest of all neural models.

The following section provides further detail on the dataset and techniques used for preprocessing.

III. Data Set, Variables & Preprocessing

The Amazon Customer Reviews dataset was pulled from Kaggle [4]. I limited the scope of my project to analyze only the products that are categorized as mobile electronics, which contains just over 100,000 entries between 2001 and 2015.

As 94% of all data were for products reviewed between 2010 and 2015, prior years were filtered out. The e-commerce space was not as prominent in the early 2000's as it is now, and so including those data from earlier years may have skewed the results. Furthermore, internet adoption has also accelerated recently, and so the demographics of the typical e-commerce consumer has likely changed over the past two decades.

The overall dataset included many fields such as veracity of purchase, product title, review date and more. However, the focus of this project is solely on sentiment analysis on review text. Thus, only the following fields were preserved:

- review body – contains review text, and the main interest of my analysis
- star rating – the rating left behind by the customer, on a 1-5 scale
- helpful votes – the number of people who found the review helpful
- total votes – the number of people who rated the review

Task 1 was defined in two separate ways, and the rationale behind this decision will be discussed in the results section.

1) Multi-class classification on textual reviews (Good, Fair, Poor)
2) Binary classification on textual reviews (Good, Poor)

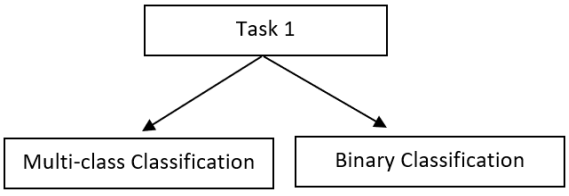

Fig. 1. Breakdown of Task 1

There were a few data preprocessing steps taken before training my deep learning models.

For Task 1, the 'star rating' field was an interval variable, representing the degree of customer satisfaction on a 1-5 scale. Instead of maintaining the five-class labels, it was redefined as an ordinal variable as follows:

TABLE I
Defining Ordinal Variables for Rating

Star Rating (processed)	Star Rating (original)
Good	[4,5]
Fair	[3]
Poor	[1,2]

This mapping was necessary because it is challenging for even a human to distinguish between ratings that are good (4 or 5) or poor (1 or 2) due to the subjective nature of the task. For instance, consider the following sample reviews from the dataset:

- "Great product. I would highly recommend": star rating – 4
- "very poor headphone. charging cable is loose": star rating – 2

For both examples, the expected ratings based on the textual feedback could very well be 5 and 1, respectively. However, that is not the case due to the subjectivity of the labels. However, the overall sentiment in each review is clear and a good model should be able to capture that.

The second task I performed was determining the usefulness of each review. The usefulness label was not directly available through the dataset, but was inferred using manual feature engineering, as follows:

$$usefulness = \frac{\text{\# of people that found review helpful}}{\text{\# of people that left feedback for review}} \quad (1)$$

For each review, the usefulness is defined as the proportion of people that found each review useful, and falls in the range [0,1]. There were many reviews where there was no feedback from others. Thus, the rows with null values for usefulness were dropped for this task.

The limitations with using usefulness as a label is the same as that of the star rating – it is challenging, even for a human, to distinguish between reviews where the usefulness is 0.9 or 0.8 because of the subjectivity of the metric. Thus, the problem was arbitrarily redefined as a binary classification task as follows:

TABLE II
Defining Ordinal Variables for Usefulness

Usefulness (processed)	Usefulness (original)
Useful	>0.7
Not Useful	<0.7

Like the star rating problem described previously, this transformation simplifies the problem for the neural model because it would only have to extract overall sentiment instead of learning individual scores, which are already highly subjective.

Before inputting any textual data into a model, it is necessary to formulate a numerical representation through vectorization. Prior to vectorization, I performed some preprocessing of the textual data as follows:

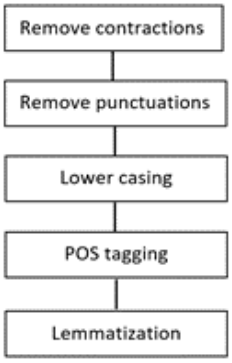

Fig. 2. Text Preprocessing

Contractions were replaced with their formal representations. For example, "haven't" was converted to "have not". Punctuation were replaced with spaces. All letters were converted to lower case. A part-of-speech tagger was used as a precursor to the lemmatization step.

Note that it is possible that further information about semantic structure in the reviews may have been destroyed during this process. For example, there may be some hidden information in reviews that are multiple sentences long or that have many exclamation marks or upper casings. Intuitively, these elements may have an impact on overall sentiment. However, analysis on the influence of these factors is outside the scope of this project although it would be prudent to study their effects in future works.

After preprocessing the text, the vectorization was done by using a pre-trained GloVe word embedding. Research has shown that the GLoVe model, which is based on global semantic similarity between words, generally outperforms Word2vec, which is based on local semantic similarity between words in the corpus [5]. The specific pre-trained GloVe model that was used in this project was trained on 42 billion tokens in a common crawl and is a row vector of length 300 [5].

IV. Methodology

Based on the results of the papers studied in the literature review, LSTM and its variants generally outperform feed-forward neural networks (FFNNs) and traditional methods in NLP classification tasks. Thus, for my experimentation I only compared performances across different LSTM-variants, with the plain LSTM being used as the base-case.

It should be noted that the hyperparameters for each model were not meticulously fine-tuned, and so there may be room for improvement in their performances.

All models were trained locally on PC with an AMD Ryzen 5 5600X 6-Core Processor, 16GB of RAM, and a NVIDIA GeForce RTX 3070 graphics processing unit (GPU). Thanks to the GPU, I was able to leverage the CUDA cores to achieve relatively quick model development times using TensorFlow.

For all classification tasks, I used the same five neural network architectures to compare performances. Here is how they performed in descending order:

- CNN-biLSTM with attention mechanism
- CNN-biLSTM
- Bi-LSTM with attention mechanism
- Bi-LSTM
- LSTM

From the experimental results, collaborative models outperformed the base-case and the attention layer boosted the performance of all models. The complete numerical analysis can be found in the results section.

There were many commonalities in model architectures, with some differences between them. For example, here is a simplified graphic showcasing how the LSTM model was structured.

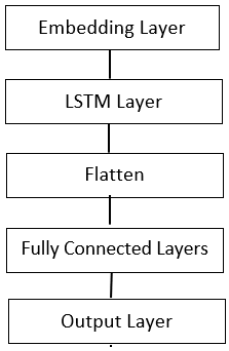

Fig. 3. LSTM-based NN Architecture

The embedding layer accepts an input of dimension 300, which is the length of the pre-trained GloVe word vectors. In the embedding layer, the trainable parameter was set to false because the word embeddings have already been trained.

The word vector is then inputted into the LSTM layer, which learns the temporal dependencies in the sequence. The output is then flattened before being passed through the FFNN for classification.

Between the dense layers, dropout layers are added as they are proven to be reliable regularization techniques [6]. They work by randomly dropping a small subset of hidden units during training to prevent units from co-adapting [6].

Since a multitask classification was being performed the output layer has 3 neurons, one for each class. A softmax activation is used to perform the classification.

The loss function used for training was categorical cross entropy and the metrics were categorical accuracy. Adam optimizer was used.

The bi-LSTM model architecture was very similar to its LSTM counterpart, except the LSTM layer was replaced

with bidirectional layers which allows it to capture sequential relationships in both directions, as seen in the figure below.

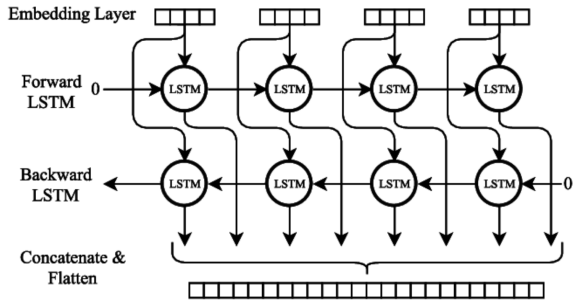

Fig. 4. Bi-directional LSTM Layers [7]

For all tasks, the model that performed the best was the CNN-biLSTM with an attention layer. Here is a simplified diagram of its general structure:

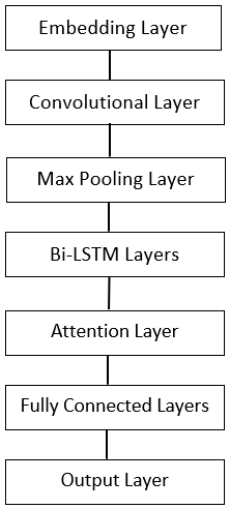

Fig. 5. CNN-biLSTM with Attention Model [7]

In this model, the word vector first passes through the convolutional layer, where key semantic structures may be learned. The convolution window size was set to 3, as suggested by the papers in the literature review section [1]. The max pooling layer extracts the most significant features, and the bidirectional LSTM layers use this information to learn the relationships of the sequence in both directions and combine the results. The attention layer extracts the most relevant inputs to the classification task.

In the following section, an in-depth analysis of the results of all experiments will be discussed.

V. Results

As mentioned earlier, Task 1 was divided into two sub-tasks. For the multi-class classification task, here are the results on the test set:

TABLE III
Task 1 - Multi-class Classification Results

Model	Test Set Accuracy
CNN-biLSTM w/ Attention	85.62%
CNN-biLSTM	84.89%
Bi-directional LSTM w/ Attention	84.08%
Bi-directional LSTM	83.68%
LSTM	83.21%

All models outperform the base-case LSTM, with the CNN-biLSTM with attention layer model performing the best. Here is a plot of validation set accuracy against number of epochs for the multi-class classification sub-task.

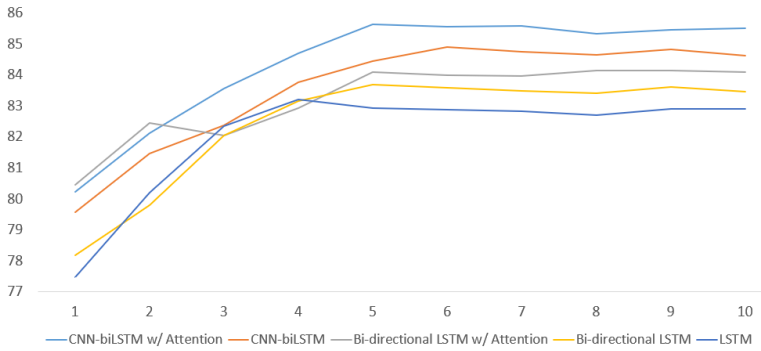

Fig. 6. Multi-class Classification - Validation Accuracy vs. Number of Epochs

As seen from the figure above, the validation set accuracy for all models begins to taper after 4-6 epochs.

The most accurate model only had a test set accuracy of around 85% and so deeper analysis of the misclassifications was conducted. The misclassified points were separated and plotted in a stacked graph for further analysis.

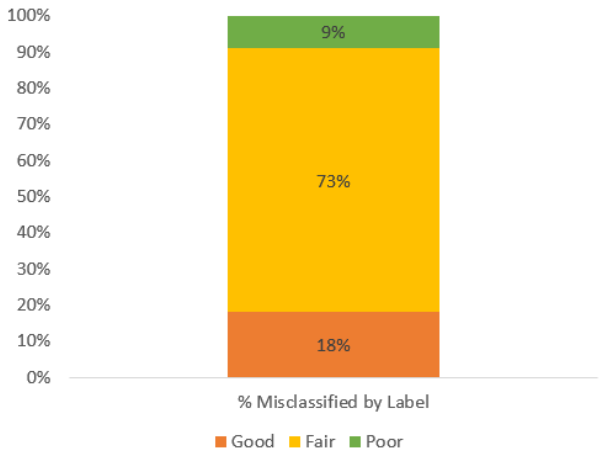

Fig. 7. Breakdown of Misclassified Points

As seen from the figure above, 73% of all misclassifications in the best performing model were for the 'Fair' label. It appears that the model does well to extract extreme sentiment, but struggles with neutrality. This may be because reviews

labelled 'Fair' have both positive and negative sentiment, however, a complete root cause analysis is beyond the scope of this project.

Here are two examples that the model misclassifies:

- "decent phone for the price. good value for money" – Prediction: 'Good' Actual; 'Fair'
- "wire build quality is poor but you get what you pay for" – Prediction: 'Poor' Actual: 'Fair'

Both samples have mixed sentiment which the model fails to correctly identify. It should be noted that some data may be mislabelled such that the sentiment is different from what the label indicates.

The analysis of the misclassifications provides motivation for the second sub-task of Task 1: How do the models perform when the 'Fair' label is completely removed from the data? The hypothesis was that the model performance would be significantly boosted due to existence of fewer classes, and removal of neutral sentiment.

Here is how the models performed in the binary classification sub-task for task 1:

TABLE IV
TASK 1 - BINARY CLASSIFICATION RESULTS

Model	Test Set Accuracy
CNN-biLSTM w/ Attention	93.89%
CNN-biLSTM	93.42%
Bi-directional LSTM w/ Attention	93.29%
Bi-directional LSTM	92.96%
LSTM	89.05%

As hypothesized, the performance of all models in the binary sub-task shows significant improvement of the multi-class subtask. The CNN-biLSTM model with the attention layer still performs the best, however, the spread between all performances (excluding the base-case) is narrower than before. All models significantly outperform the base-case LSTM.

Here is a plot of the validation accuracy against number of epochs for the binary classification sub-task.

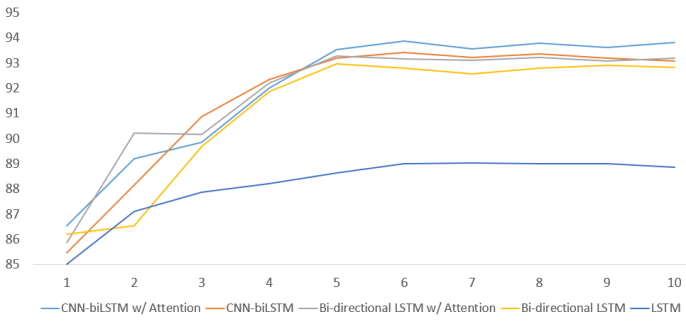

Fig. 8. Binary Classification - Validation Accuracy vs. Number of Epochs

As seen from the figure above, the validation set accuracy for all models begins to taper after 4-6 epochs.

For the binary classification task, the number of neurons in the output layer was changed to one, and the activating function was changed to sigmoid.

Here are two samples that were misclassified in the best-performing model:

- "awfully steep learning curve to use. Do not recommend for beginners but I am a pro so it works for me" Prediction: 'Poor' ; Actual: 'Good'
- "good keyboard as the lights are very bright and vibrant. Wish it was 150 tho" – Prediction: 'Good'; Actual: 'Poor'

In both samples there is a detachment between the prevailing sentiment in the text and its label, which may be why they were misclassified.

For task 2, a binary classification was performed on the usefulness of the review. As mentioned in a previous section, the 'usefulness' label is manually engineered and defined somewhat subjectively. Thus, the hypothesis is that the model performance may be poorer than for task 1. Here is how the models performed in task 2:

TABLE V
TASK 1 - BINARY CLASSIFICATION RESULTS

Model	Test Set Accuracy
CNN-biLSTM w/ Attention	67.01%
CNN-biLSTM	66.34%
Bi-directional LSTM w/ Attention	66.15%
Bi-directional LSTM	65.88%
LSTM	65.45%

As expected, the accuracy of the models on task two was much lower than that of task one. Once again, the CNN-biLSTM model with the attention layer outperforms the other models, but the spread between model performances remains narrow (excluding base-case LSTM).

Here are two samples that were misclassified in the best-performing model:

- "good value for money. Highly recommend" – Prediction: Useful ; Actual: Not Useful
- "don't buy. Not as advertised." – Prediction: Not Useful ; Actual: Useful

Both reviews lack the level of detail that should make them informative to others, however, it appears that consumers have differing opinions. One potential improvement to the model, which is outside the scope of the project, is to re-train the models before pre-processing the text as some semantic information may have been lost in that process.

Below is a chart that shows the time to train each model for one epoch (same batch size) in the multi-class classification sub-task for task 1.

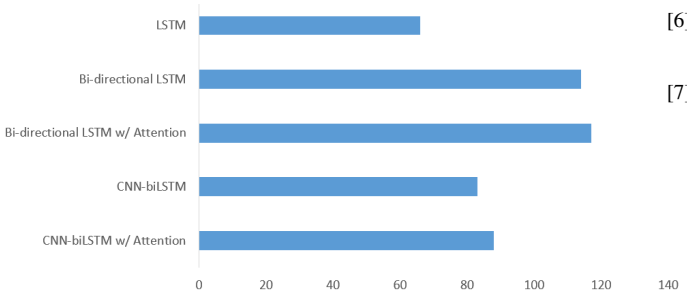

Fig. 9. Comparison of Training Time (s) - One Epoch, Same Batch Sizes

The LSTM took the least amount of time to train, followed by CNN-biLSTM models. The longest training time was for the bi-LSTM models. The attention layer seems to have a small impact on training time.

VI. Conclusion

In this project, it was shown that LSTM-variants generally perform well in text classification tasks. Furthermore, experimental results show a modest performance boost through the use of collaborative models such as CNN-biLSTM and attention layers.

For task 1, the best model struggled with extracting neutral sentiment (eg. 'Fair') but still provided over 85% accuracy on the test set. When 'Fair' reviews were excluded, redefining the task as a binary classification problem, the performance was further boosted to 93.89%. For task 2, the best model only provided an accuracy of 67.01%. This may be partly due to the subjective definition of the concept of 'usefulness', but the root cause analysis is outside the scope of this project.

Model performances may generally be improved through more meticulous fine-tuning of hyperparameters. Furthermore, in future works it would be intriguing to assess the performances of these models without text pre-processing. The inclusion of exclamatory punctuation (eg. '!') and other textual features such as capitalization (eg. 'TERRIBLE') may provide more semantic information. Lastly, pre-trained GloVe vectors on social media data (eg. Twitter) may boost performance due to the informal nature of product reviews.

References

[1] S. Mathapati, A. K. Adur, R. Tanuja, S. Manjula, and K. Venugopal, "Collaborative deep learning techniques for sentiment analysis on imdb dataset," in *2018 Tenth International Conference on Advanced Computing (ICoAC)*. IEEE, 2018, pp. 361–366.

[2] C. Li, G. Zhan, and Z. Li, "News text classification based on improved bi-lstm-cnn," in *2018 9th International Conference on Information Technology in Medicine and Education (ITME)*. IEEE, 2018, pp. 890–893.

[3] A. Kumar, S. R. Sangwan, A. Arora, A. Nayyar, M. Abdel-Basset *et al.*, "Sarcasm detection using soft attention-based bidirectional long short-term memory model with convolution network," *IEEE access*, vol. 7, pp. 23 319–23 328, 2019.

[4] "Amazon us customer reviews dataset," https://www.kaggle.com/cynthiarempel/amazon-us-customer-reviews-dataset?select=amazon$_reviews_us_Apparel_v1_00.tsv$.

[5] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.

[6] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[7] "Papers with code - bilstm explained." [Online]. Available: https://paperswithcode.com/method/bilstm