**Project Members :** Harish Anumanchineni, Greg Araujo

**1 Description Of Problem**

User generated reviews through platforms such as Yelp, Tripadvisor and Rotten Tomatoes have become a trusted source for opinions on services ranging from food to auto mechanics and everything in between. While the user review rating can give people a sense of the quality of the service, previous research has shown that average ratings are rounded to the

nearest half a star. For example, a business with a 3.24 rating will be rounded to 3 stars, while a business with a 3.25 rating will be rounded to 3.5 stars [1]. For potential customers, it may be difficult to determine the difference between a business that received a 3 versus one that received a 3.5 average rating just from reading user reviews. Given that using customer reviews alone can be difficult in determining a customer rating, we include additional categorical data such as the type of ambience, the type of food served, price range to determine if these attributes help in accurately predicting restaurant ratings.

**2 Data Sources**

Yelp has provided a subset of businesses, reviews, and user data for use in personal, educational, and academic purposes [2]. In its inception, the Yelp dataset consisted of 1,537 business records, 29,908 user reviews and 3,873 user profiles [3]. Today’s version consists of 8,021,122 user reviews, 209,393 businesses across 10 metropolitan areas [2]. For our project, we have focused on predicting ratings for restaurants. Our revised dataset includes 4,472,431 restaurant reviews by 1,179,034 user reviews on 70,180 restaurants. After registering on the Yelp website, you can download the dataset from <https://www.yelp.com/dataset/download>. Below are the attributes and datasets that will be merged for modeling:

* **Business.json** : Contains business data including location data, attributes, and categories.
  + Relevant attributes :
    - business\_id : unique string business id
    - city : city restaurant is located in
    - star rating : average rating
    - review\_count : number of reviews
    - business categories : style/type of food served
* **Review.json** : Contains full review text data including the user\_id that wrote the review and the business\_id the review is written for.
  + Relevant attributes :
    - review id : unique review id
    - user id : unique user id, maps to the user in user.json
    - business id : maps to business in business.json
    - stars : rating
    - date : date timestamp
    - text : the customer review
    - useful : number of useful votes received
    - funny : number of funny votes received
    - cool : number of cool votes received
* **User.json** : User data including the user's friend mapping and all the metadata associated with the user.
  + Relevant attributes :
    - user\_id : unique user id, maps to the user in user.json
    - review\_count : the number of reviews they've written
    - useful : number of useful votes sent by the user

**3 Literature Review**

Since 2005, students, academics and researchers alike have experimented with the Yelp dataset. To date, Yelp has officially announced winners to 12 rounds of the Yelp Dataset Challenge [4]. With so much attention given to the dataset, it was not difficult to find research on user rating prediction specific to the Yelp dataset. Nabiha Asghar had used user reviews to build both logistic regression and support vector machines to predict review ratings [5].While Asghar broke the corpus into unigrams, bi-grams, and tri-grams during pre-processing to implement during modeling. Asghar nor any other papers we found combined both categorical data (i.e. Food Category, Restaurant Price Range) with n-gram data. For our deep learning model, we observed that Rao et al. used a variety of Recurrent Neural Network (RNN) LSTM models such as PC-LSTM, CIFG-LSTM, Bi-LSTM for Document-level Sentiment Classification [6]. Similar to the issue we experience with our machine learning research. We did not find any academic research that included both categorical and text based data within a deep learning model.

**4 Technology + Issues Experienced**

All models and preprocessing was completed within the google co-lab environment.

**4.1 Data Ingestion**

Because we were combining categorical and user review data together, our group had to utilize all three json files provided by Yelp (Business.json (132 mb), Review.json (5 gb), User.json (2 gb)). While each file was worked on separately for feature engineering and ensuring data quality, merging the respective files caused the co-lab environment to crash due to a lack of sufficient RAM. After several attempts, we were able to merge the files together. The final file size turned out to be approximately 6 gb. During the merging process we learned a new concept of loading large dataset using chunks (specifying the chunk size) which is the most common way of dealing with larger size datasets which saved us from running out of memory(RAM)

**4.2 Insufficient RAM**

After merging the file, the dataset had 4,472,431 rows along with 104 columns. Given the size of the dataset, and our google co-lab environment limitations, we were only able to use 10% of the dataset.

**4.3 LSTM Implementation**

During the LSTM implementation, the first thing that came to mind for us was why not Artificial Neural Networks (ANN) for classification. Why only RNN? Our online research made the distinction between the methods chosen [12]. RNN captures the sequential information present in the input data i.e. dependency between the words in the text while making predictions. That is the order of text is important.

LSTM (Long Short Term Memory) are used for implementing Recurrent neural networks. The lecture (in the current semester) was one that helped get the basic idea of RNNs. Later Lab work provided clarity with regards to the implementation. Adjusting the parameters was the main difficulty we faced in the implementation of neural networks. After this step execution time was the main thing that we faced a major difficulty (it is taking very long for one epoch or iteration) which is not an ideal scenario. We thought of using the GPU feature so implemented the same with CuDNNLSTM which increased the speed and reduced the execution time by a larger margin. The final hardship we faced was overfitting of the model, in order to overcome it we implemented the concept of early stopping which reduced overfitting.

**5 Data Pre-Processing**

For this project, we build three different prediction models, and utilize three different feature extraction methods for each supervised learning algorithm.

**5.1 Pre-Processing**

Our team wrote Python scripts for the business.json, review.json and user.json files. The purpose of each script was to convert all attributes/dimensions into a csv format so that they could be merged together to include attributes from all three files. Within the business.json, certain fields were stored as dictionary fields(i.e. Business attributes) and required additional scripts to extract the columns for modeling.

Given that the review form within Yelp is a free form field, users are allowed to use any combination of alphanumeric characters within a review. Prior to modeling, steps were taken to convert a user review to lower case. Also, stop words such as couldn’t, doesn’t, etc…were removed from the review along with any punctuation marks. Finally, techniques such as lemmatization were implemented so that groupings of similar words can be analysed as a single item as identified by the word's lemma. All these steps were implemented using a series of python libraries and scripts. Again, utilizing parallelization libraries such as multiprocessing enabled us to use 4 cores which reduced the execution time from 1,200 secs to 400 secs for certain tasks.

**5.2 Feature Extraction**

**5.2.1 Countvectorizer**

For this project we decided to utilize the Count Vectorizer function within sklearn to convert the user reviews into a matrix of token counts [7]. We then use the vectorizer, to transform each user into a document and term matrix. Each row represents a document and each column represents the word and each cell represents the frequencies. The vectorizer is modeled off of the training dataset and then is used to transform the test dataset into its own document and term matrix as well.

**5.3 Supervised Learning**

To train our prediction models, we used 3 supervised learning algorithms (2 machine learning, 1 deep learning).

**5.3.1 Logistic Regression**

In our logistic regression model we make the distinction between a positive review (review stars > 3) and a negative review (review stars <= 3). Before running the model, we join the document-term sparse matrix with the additional categorical attributes and predict whether a review is positive or negative.

**5.3.2 Support Vector Machines (SVM)**

For our project, we use linear SVMs for multiclass classification (Linear Support Vectors). The strength of SVMs is that they eliminate the non-uniqueness of solutions by optimizing the margin around the decision boundary, and handle non-separable data by allowing misclassifications [5]. Given the differences between user review ratings that are within 0.5 stars of each other, we thought that the use of decision boundaries would lead to an increase in accuracy.

**5.3.3 Recurrent Neural Net (RNN) - LSTM**

LSTM can capture the long dependencies (i.e. user reviews) in a sequence by introducing a memory unit and a gate mechanism which aims to decide how to utilize and update the information kept in the memory cell [8]. Given the datasets heavy reliance on user reviews, utilizing the Long Short Term Memory algorithm would produce accurate results. In the interest of time, we have decided to implement the NVIDIA CUDA® Deep Neural Network library (cuDNN). CuDNN is a GPU-accelerated library of primitives for deep neural networks. cuDNN provides highly tuned implementations for standard routines such as forward and backward convolution, pooling, normalization, and activation layers [9].

**6 Analysis**

**6.1 Exploratory Data Analysis (EDA)**

Prior to modeling, exploratory data analysis (EDA) was conducted in order to summarize (if any), the main characteristics of the dataset. Given that we were mixing categorical variables such as restaurant price range, best day of week to go to a restaurant, we had hoped that there may be some insights before modeling that would help explain some of the outputs after modeling had been completed.

**6.1.1 Overall Dataset**

While there were over 4.4 million user reviews within the dataset, the sample provided from Yelp from nearly 1.2 million users on over 70,000 restaurants. The majority of user reviews were rated as 5 (40%) while only 13.5% of reviews had a rating of 1 (Fig 1).

|  |  |  |
| --- | --- | --- |
| **User Rating** | **No. Of Reviews** | **% Of Total** |
| 1 | 530,617 | 13.5% |
| 2 | 411,891 | 10.4% |
| 3 | 590,956 | 15.0% |
| 4 | 1,164,095 | 29.5% |
| 5 | 1,774,871 | 45.0% |

*Fig 1.* User Rating Distribution - Yelp Dataset

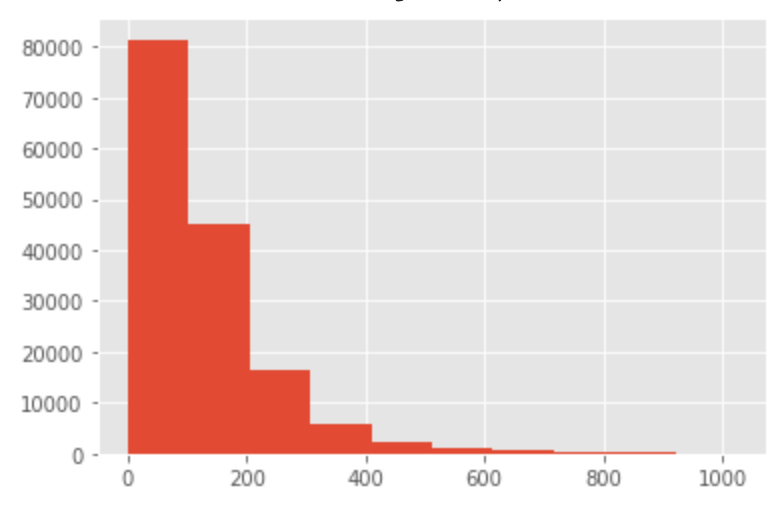
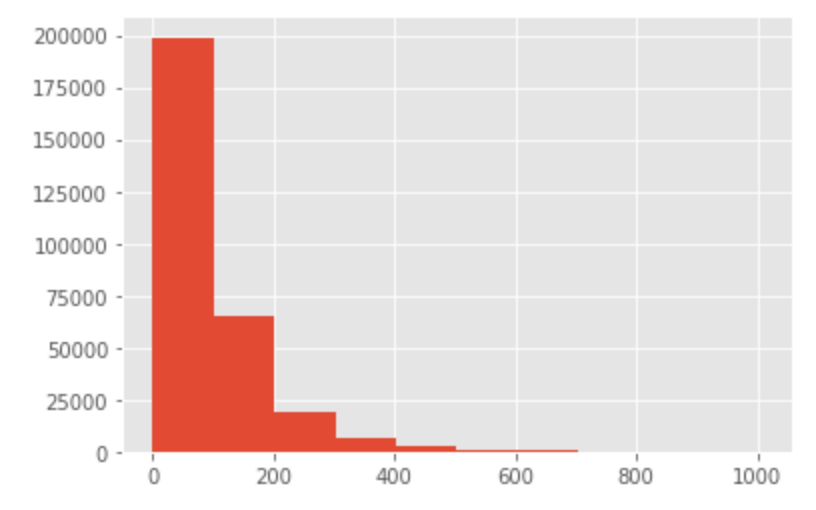
The large majority of users are infrequent reviewers, as over 50% of user reviews had only reviewed a single restaurant. It is not until you reach past the 90th percentile where you see reviewers with at least 7 reviews associated with a user id.

The Yelp sample dataset was selected from 10 metropolitan areas [2]. As a result, we observed over 850 unique geo-locations with the cities of Toronto and Las Vegas (excluding census metropolitan area) as the only locations with over 10% of restaurants within the dataset. While both cities had the most restaurants reviewed, over 30% of user reviews were from Las Vegas. No hour within the day had more than 6% of user reviews while the day of week was equally distributed with the day of week distribution being between 12% - 15%.

**6.1.2 Sample Dataset**

Due to the processing issues that we experienced, for the machine learning algorithms we used 10% of the dataset for modeling, and 5% of the dataset for deep learning modeling. The sample function within the Pandas library enabled us to return a random sample of items from an axis of object [13].

Within the machine learning dataset, on average users posted reviews with 74 words within them. While some reviews posted comments with over 1,000 words within them. Although the word count for positive and negative reviews followed similar distribution patterns (Fig 2, 3).

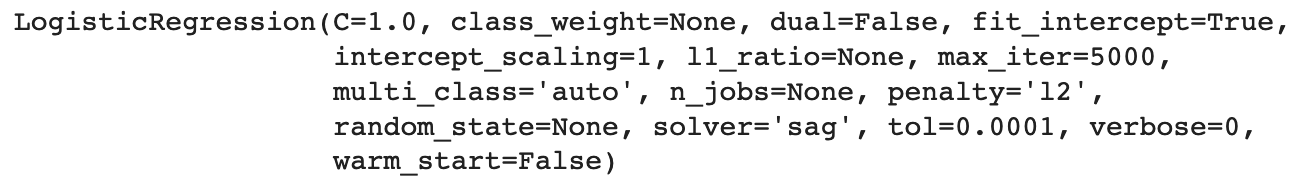
|  |  |
| --- | --- |
| *Fig 2.* Word Count Frequency - Negative Reviews | *Fig 3.* Word Count Frequency - Positive Reviews |

We observed that users who left a negative review, on average wrote an additional 30 words within their comments.

For modeling, each dataset was split into training and test datasets utilizing an 80%/20% split.

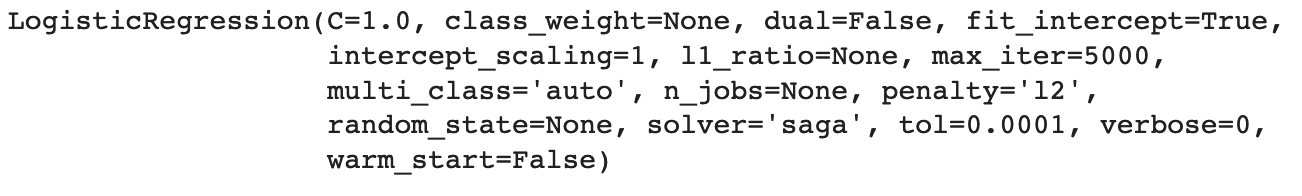
**6.2 Logistic Regression**

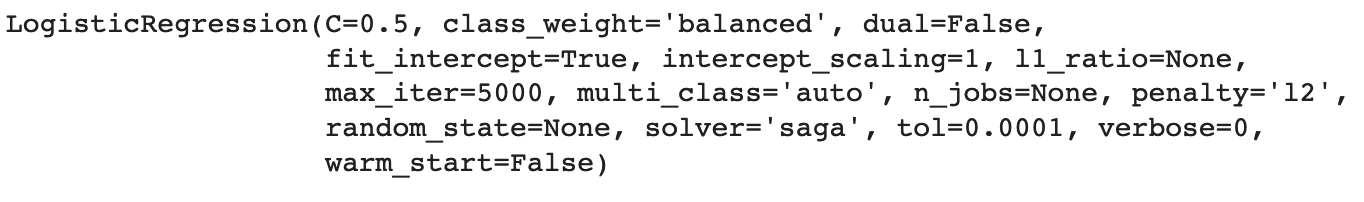
For our first attempt at predicting review ratings, we decided to go with the default settings (excl. max\_iter) within the logistic regression package within sklearn (Fig 1). Because the model did not converge within the default iteration settings, we increased it to 5,000.



*Fig 1.* Logistic Regression parameters - Attempt 1

The model resulted in an overall accuracy score of 0.5992. Given the low accuracy score, we decided to explore other solver functions within the logistic regression package. As per the sklearn documentation, ‘sag’ and ‘saga’ (stochastic average gradient descent) are more effective on larger datasets [10]. As a result, we switched to the saga solver and proceeded with additional hyperparameter tuning.

*Fig 2.* Logistic Regression parameters - Attempt 3



*Fig 3.* Logistic Regression parameters - Attempt 5

Figure 2 and 3 display some of the hyperparameters that we experimented with.

* C: Acts as a control variable that retains strength modification of regularization [11]. By lowering C (note that C must be a positive value), would strengthen the regulator used to classify between a positive and negative rating. We observed minimal impact on model results (Attempt 4, 5).
* Class\_weight: Given the imbalance between positive and negative reviews. Adjusting the class weight to balanced by specifying a class weighting configuration that is used to influence the amount that logistic regression coefficients are updated during training [12]. Results were in line with what had been observed previously (Attempt 5).
* Penalty: As we learned in class, using penalty regularization can lead to a decrease in overfitting a model and create a more generalized model. As such, we attempted to use the L2 penalty but did not see any immediate impact to the model (Attempt 3 - 5).

After going through the hyperparameter exercise, we saw that all models had very similar results (Figure 4).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Accuracy** | **Precision** | **Recall** | **F1** |
| Attempt 1 | 0.5992 |  |  |  |
| Attempt 2 | 0.6098 | 0.7026 | 0.704 | 0.7031 |
| Attempt 3 | 0.6113 | 0.6942 | 0.7293 | 0.7113 |
| Attempt 4 | 0.6113 | 0.6942 | 0.7293 | 0.7113 |
| Attempt 5 | 0.6094 | 0.6986 | 0.7126 | 0.7056 |

*Fig 4.* Logistic Regression Model Results

**6.3 Support Vector Machine**

Given that, Support Vector Machines have the ability to predict multi class labels, we decided to build models with that objective in mind. Our first attempt with max iterations set to 5,000 failed to converge. With our second attempt, we increased the max iterations to 15,000 and was able to get the model to converge. However, the results were nowhere near what we had expected. After training, the model had an accuracy score of 0.2098. By examining the confusion matrix (Fig 5), we observed that the model had issues predicting user reviews with a 1, 2, or 5 star rating. While initially we had thought that the model may perform poorly as a result of the ratings distribution, we could not explain why the model did especially poorly on predicting 5 star ratings given that it represented the largest share of ratings (39%) within the dataset.

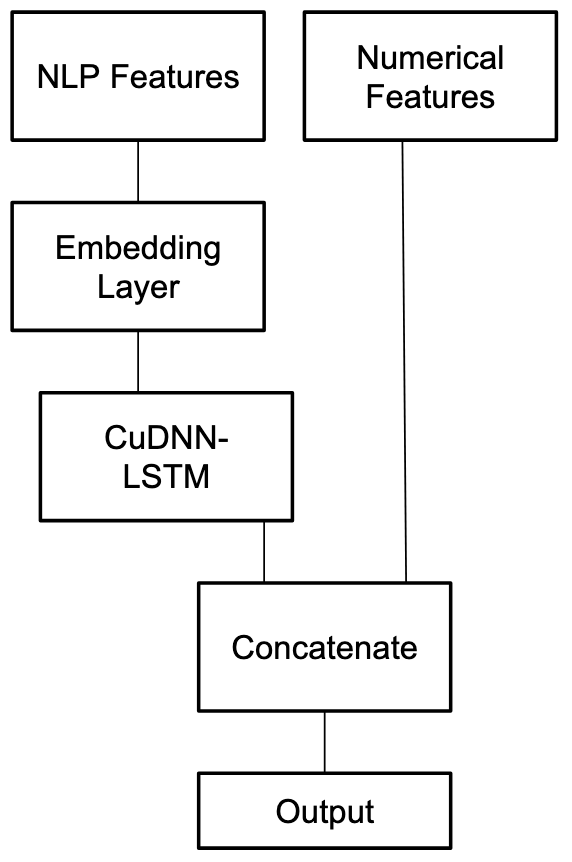
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | | **Predicted Class** | | | | |
| **1** | **2** | **3** | **4** | **5** |
| **Actual Class** | **1** | 635 | 24 | 4,177 | 5,676 | 15 |
| **2** | 491 | 28 | 3,958 | 3,858 | 35 |
| **3** | 1,027 | 64 | 5,868 | 4,744 | 108 |
| **4** | 1,883 | 86 | 9,465 | 11,910 | 237 |
| **5** | 1,340 | 23 | 7,617 | 25,859 | 321 |

*Fig 5*. Confusion Matrix - SVM Test Set

As a result of seeing such poor results from the onset, we decided to move onto our deep learning implementation.

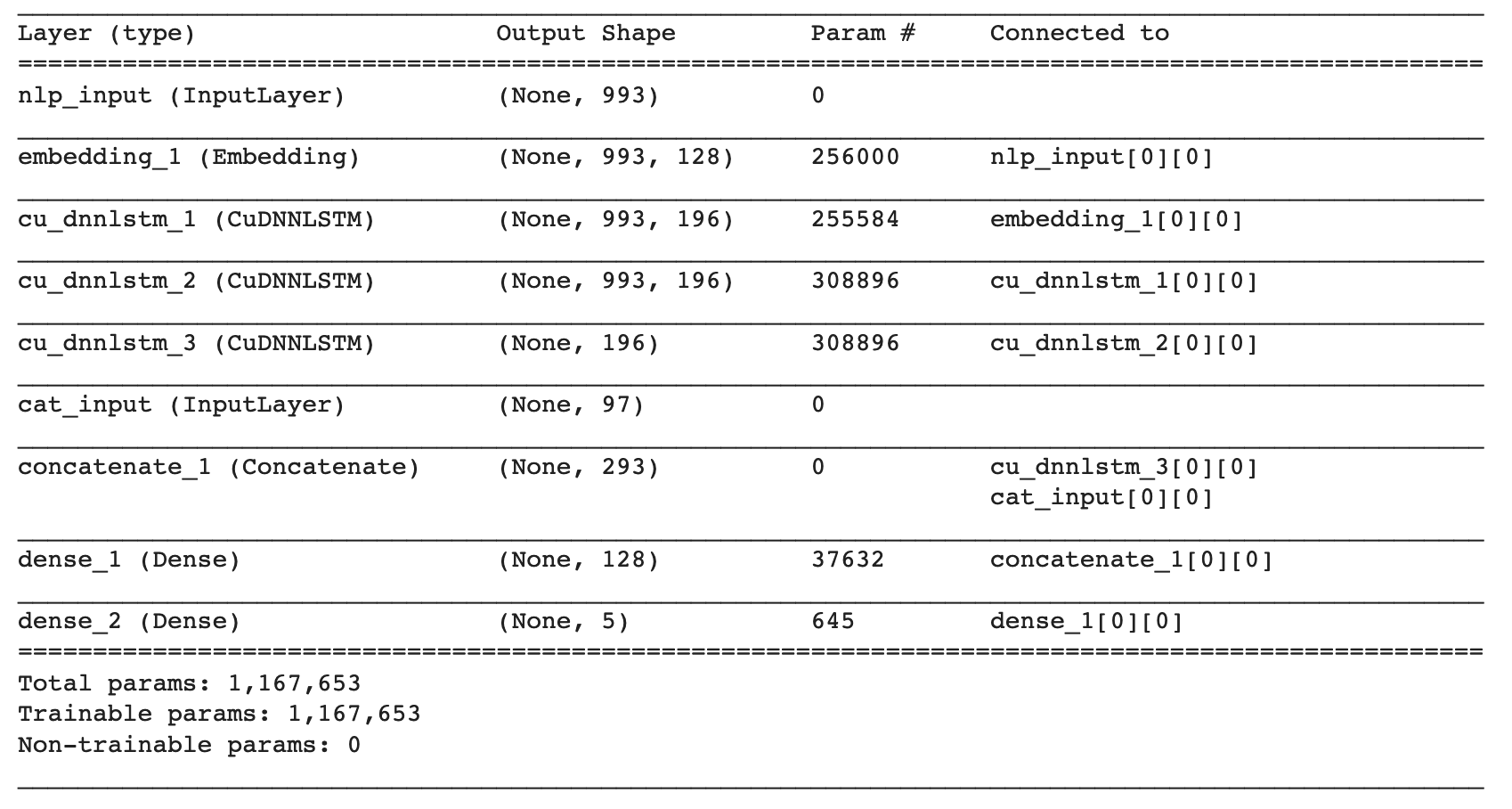
**6.4 LSTM**

As mentioned in section 6.1.2, for our deep learning implementation, we had to limit ourselves to 5% of the entire dataset which amounted to modeling off of 223,622 user reviews. Because we decided to combine, our architecture for the model would be different from the frameworks taught during lectures. Research suggested that we utilize an RNN (LSTM) network for the sentiment analysis and then combine the numerical data in a Dense network to predict the user rating [14].



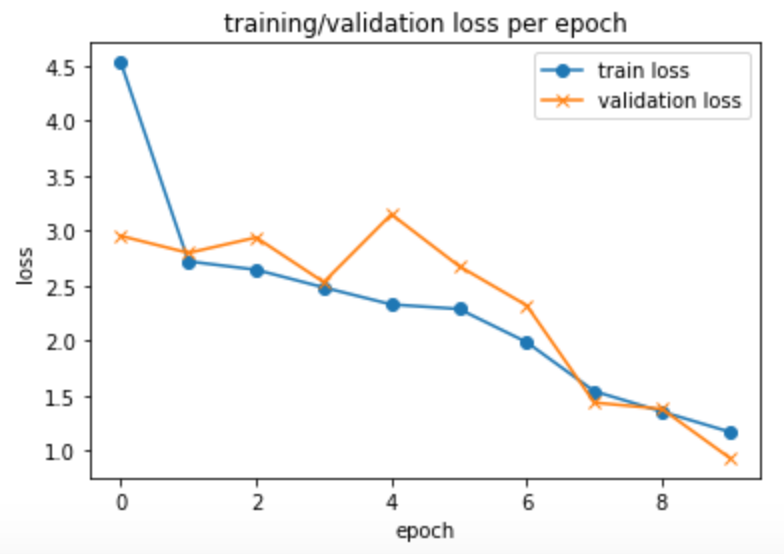
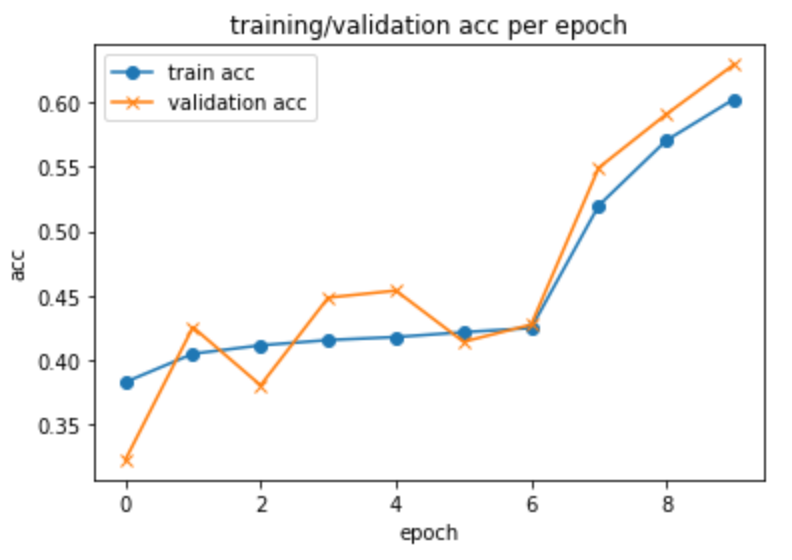
*Fig 6.* Deep Learning Framework used for predicting user ratings on Yelp Dataset.

In our first attempt, we decided to run a 3 layer CuDNN network, along with a 128 neuron Dense layer to predict user ratings. In total the model had to train over 1.1 million parameters (Fig 7).



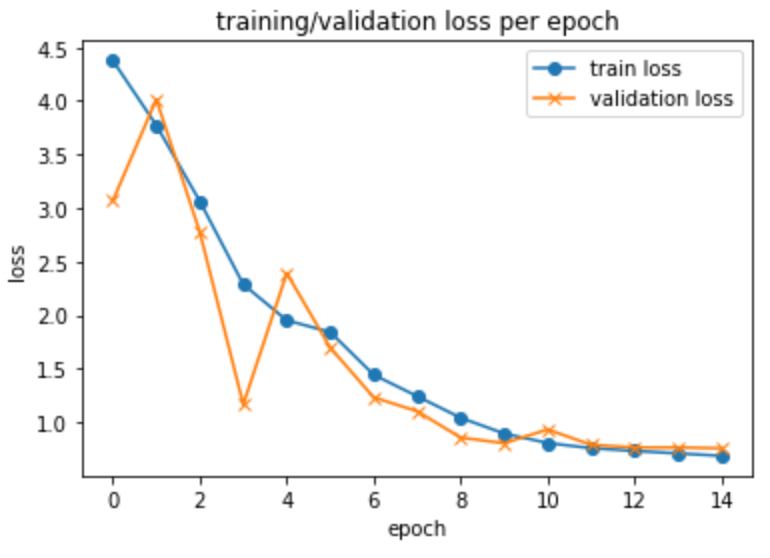
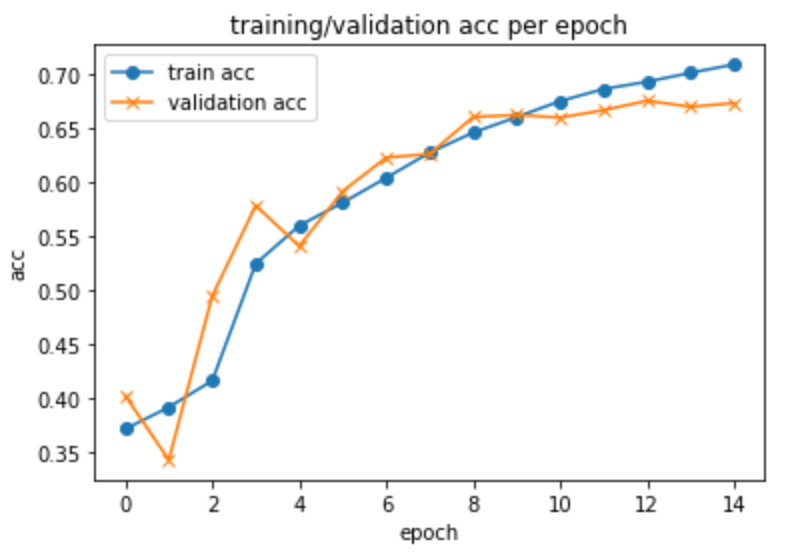
*Fig 7.* CuDNN Architecture - Attempt 1

Within the model itself, we set the kernel\_initializer equal to glorot\_uniform (to find a good variance for the distribution from which the initial parameters are drawn) [16] and the recurrent\_initializer equal to orthogonal (no matter how many times we perform repeated matrix multiplication, the resulting matrix doesn't explode or vanish) [15]. Given the time required to run each model, we set the epoch to 10 and saw an accuracy score of 0.6291. Although the model only had 10 epochs, after examining the accuracy and loss functions we had reason to believe that had the model had more time to run, we would have had a much stronger model (Fig 8, 9).



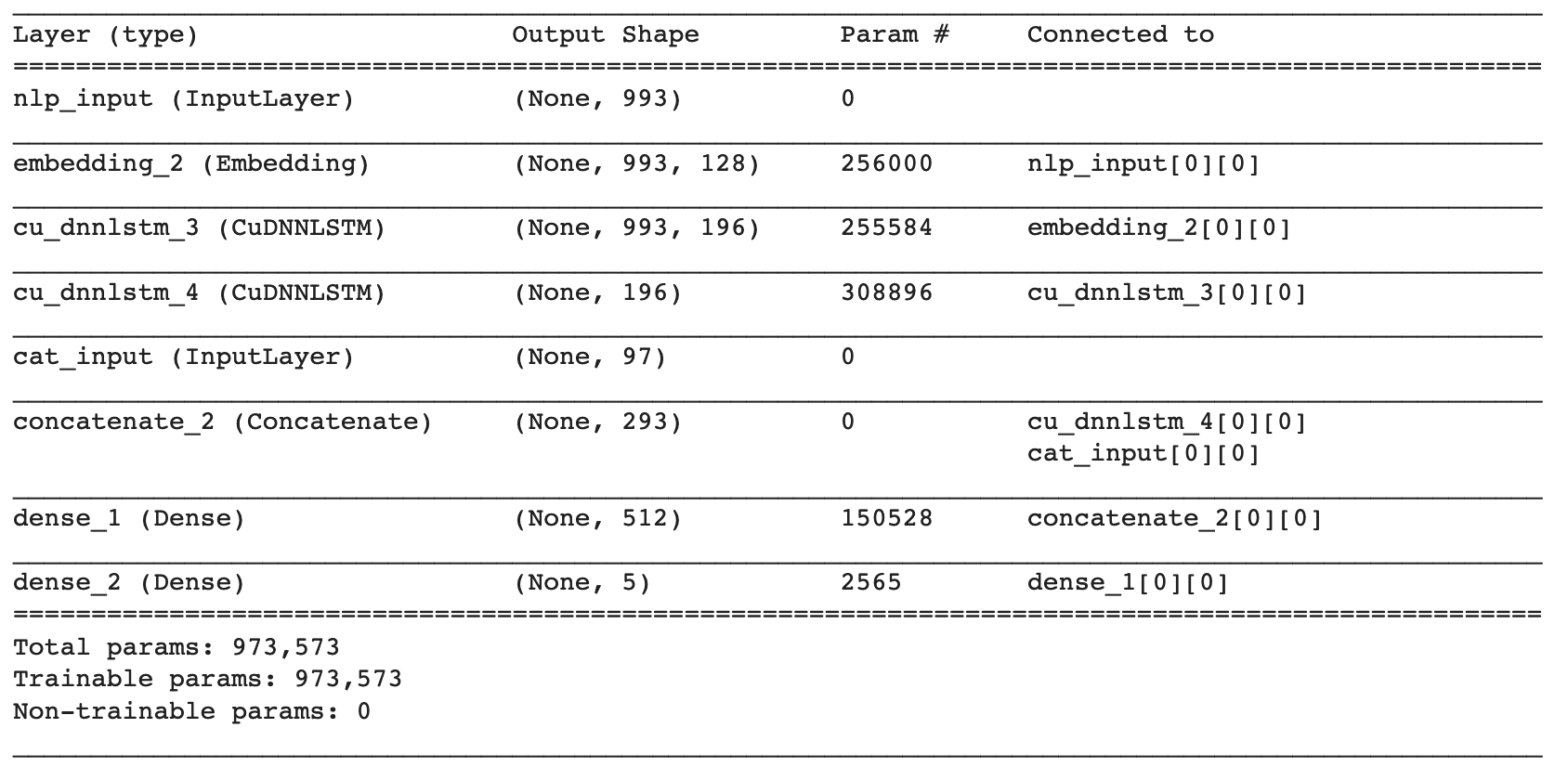
|  |  |
| --- | --- |
| *Fig 8.* Training/Validation Accuracy - Attempt 1 | *Fig 9.* Training/Validation Loss - Attempt 1 |

Seeing as the model was analyzing over 200 numerical categories, we thought to increase the number of neurons to 256 and see if that would improve accuracy results. By increasing the number of epochs to 15, this led to improvements over our first model as we saw the accuracy score improve to nearly 0.68 (Fig 10, 11). Additionally by increasing the epochs to 15, we were able to observe improvements within the model.



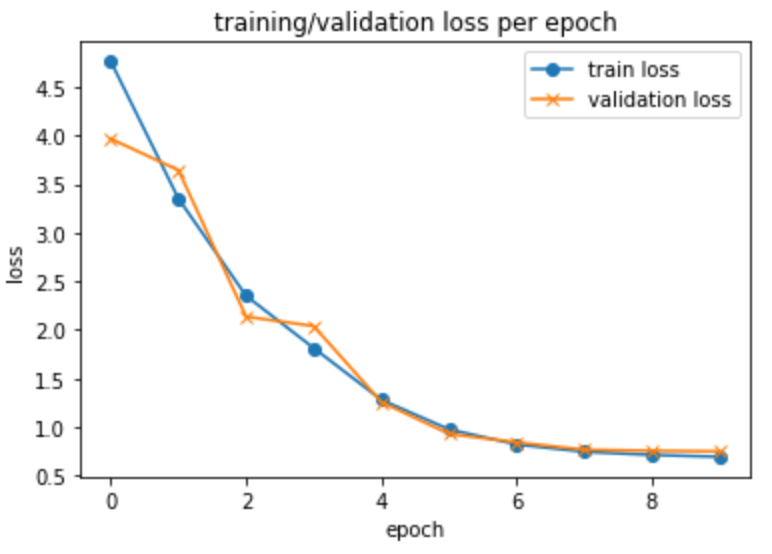
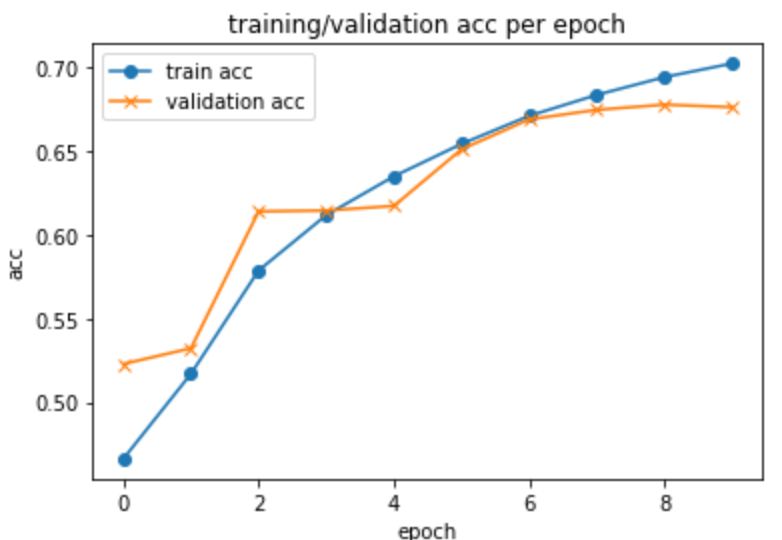
|  |  |
| --- | --- |
| *Fig 10.* Training/Validation Accuracy - Attempt 2 | *Fig 11.* Training/Validation Loss - Attempt 2 |

With the improved results, we thought to increase the neurons again to 512 to see if we continued to observe improved accuracy scores. Furthermore, seeing as there were improvements as a result of increasing the number of neurons, we hypothesized that the third hidden CuDNN-LSTM layer made did not have too much impact on the accuracy of the model. The additional benefit of removing the third layer, was that the model would also run quicker as a result of decreasing the number of parameters required by over 200k (Fig 12).



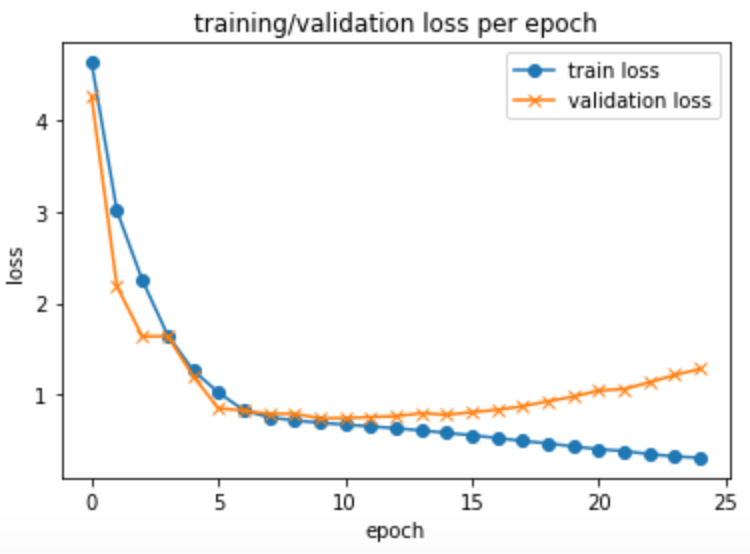
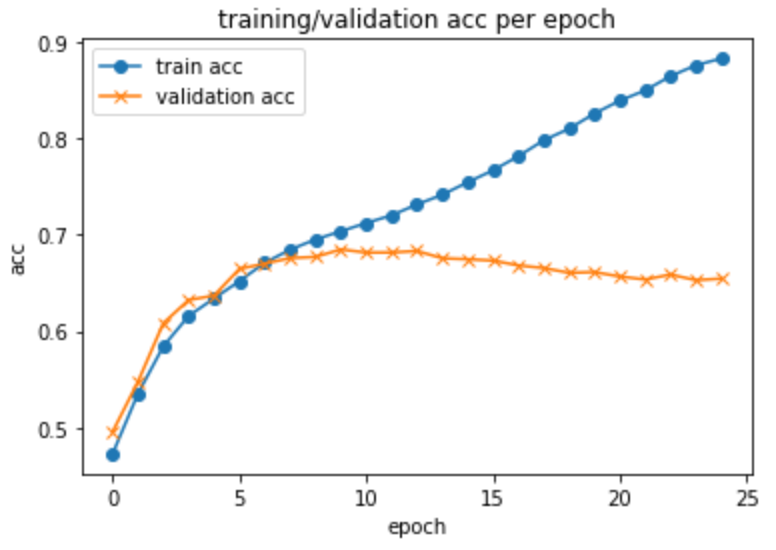
*Fig 12.* CuDNN Architecture - Attempt 3

The results of the model were very similar to those in previous iterations as we observed an accuracy score of 0.6762 and similar trends in accuracy and loss as in previous models run (Fig 13, 14).



|  |  |
| --- | --- |
| *Fig 13.* Training/Validation Accuracy - Attempt 3 | *Fig 14.* Training/Validation Loss - Attempt 3 |

Removing the third layer, we did not sacrifice any potential gains in accuracy and also had the benefit of increasing the time it took to run an epoch. Now that we were comfortable with the architecture to use before hyperparameter tuning, we ran the model overnight to see what types of gains would be made to the model in terms of accuracy if we set the epochs to 25. And while we observed marked improvements in accuracy within training, the validation accuracy plateaued around the 6th epoch.

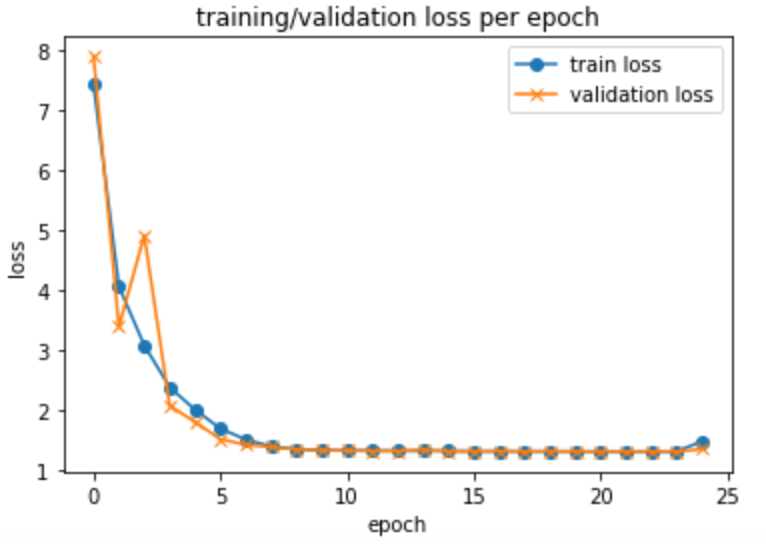
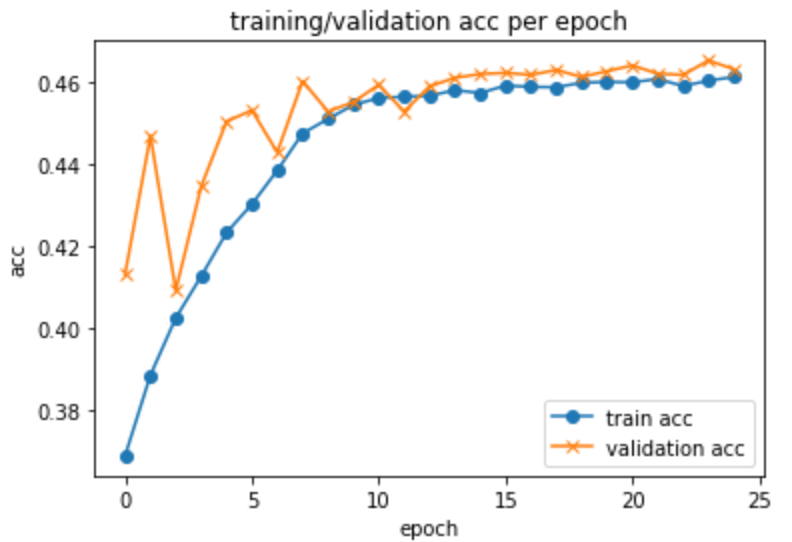


|  |  |
| --- | --- |
| *Fig 15.* Training/Validation Accuracy - Attempt 4 | *Fig 16.* Training/Validation Loss - Attempt 4 |

With the validation accuracy results plateauing so quickly, we thought that the numerical data could be introducing significant noise to the dataset (see Lessons Learned for additional insights).

One final attempt was made using a Bi-Directional LSTM model, however each epoch run time was estimated at over 2 hours so we decided to perform hyperparameter tuning with the above model.

With our first attempt at hyperparameter tuning, we looked to add an L2 regularizer to the kernel, bias term, activity and function in order to reduce the possibility of overfitting and creating a more generalized model. And while the model began to converge as we increased the epochs, accuracy plateaued at roughly the 0.46 mark.



|  |  |
| --- | --- |
| *Fig 17.* Training/Validation Accuracy - Attempt 6 | *Fig 18.* Training/Validation Loss - Attempt 6 |

Given that the models results decreased significantly we concluded experimentation and decided that the model within Attempt 3 worked best.

**7 Lessons Learned**

Throughout the entire process, there were many lessons learned from a data processing standpoint all the way to modelling.

**7.1 Data Processing**

Given the size of the dataset, there were multiple lessons learned from a data processing standpoint. Our group had to learn how to parallelize data using the multiprocessing package within python in order to circumvent running out of memory within our google colab environment.

**7.2 Deep Learning Architecture**

Within the architecture itself, our group had to learn and make use of early stopping in order to prevent the model from overfitting, and making use of neural networks for NLP tasks for sentiment analysis. Finally, because of the fact that we wanted to use both sentiment analysis and categorical data to predict we learned how to implement a model utilizing both types of attributes.

**7.3 Modeling**

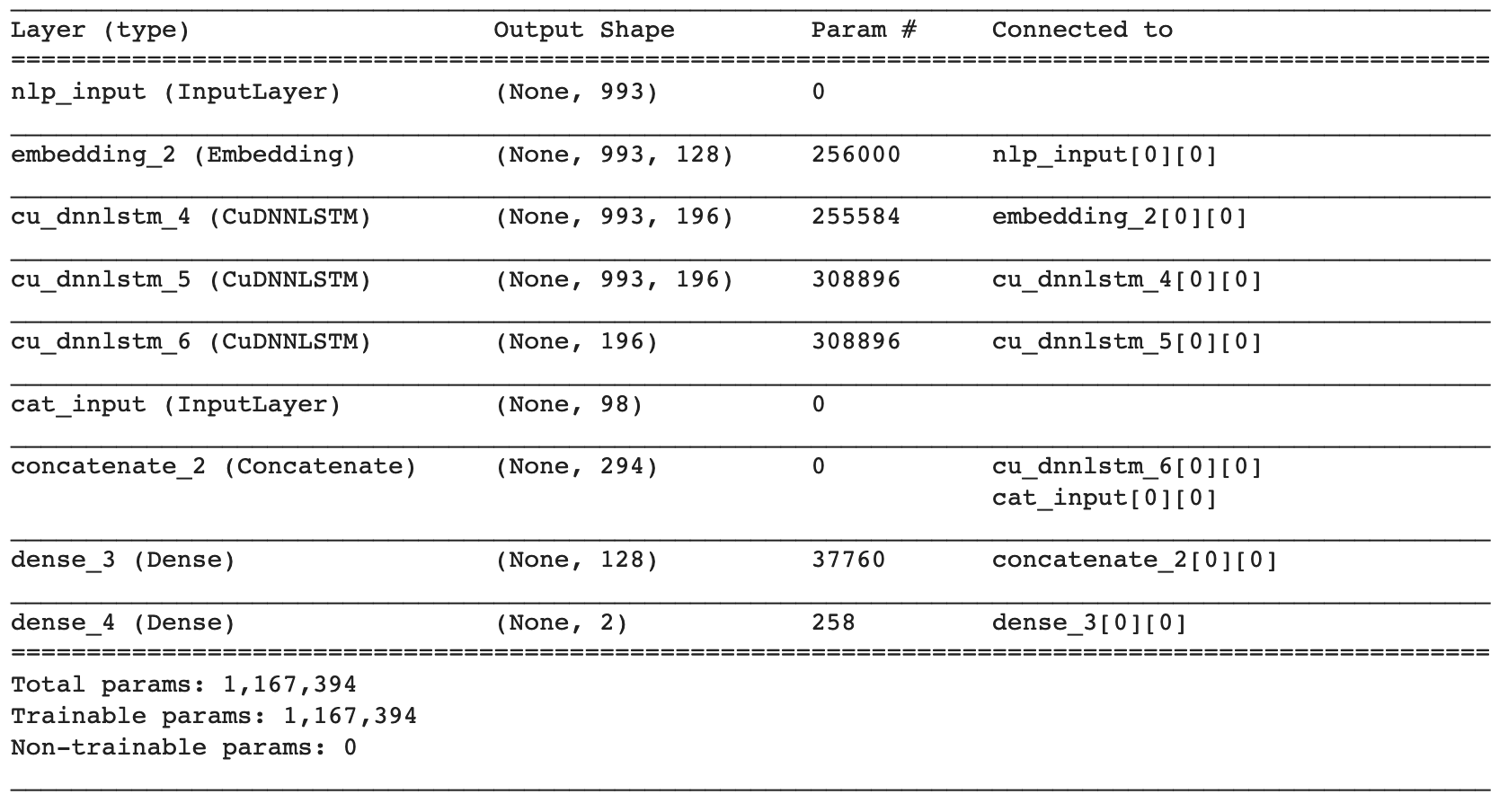
Throughout the process, we thought that the lower accuracy results could be attributed to an imbalance within the dataset as a result of an uneven distribution of user review stars. Further analysis is required to determine if a more balanced dataset would improve results. Instead of experimenting with a more balanced dataset, we decided to move towards predicting a positive review (rating > 3) versus a negative review (rating ≤ 3). The rationale for this decision is because we felt that the categorical attributes were introducing too much noise into the dataset when trying to predict multiple classes. After running binary classification models for SVM and LSTM, we concluded that given the dataset utilized, it was a better predictor for positive and negative ratings than it was for predicting the rating of a restaurant experience.

**7.3.1 Support Vector Machines**

By moving towards binary classification, we observed a significant improvement in prediction versus the multi-class model. Accuracy increased from 0.2098 to 0.6299. Precision, recall and F1 scores also improved drastically (0.7482, 0.6575, 0.7).

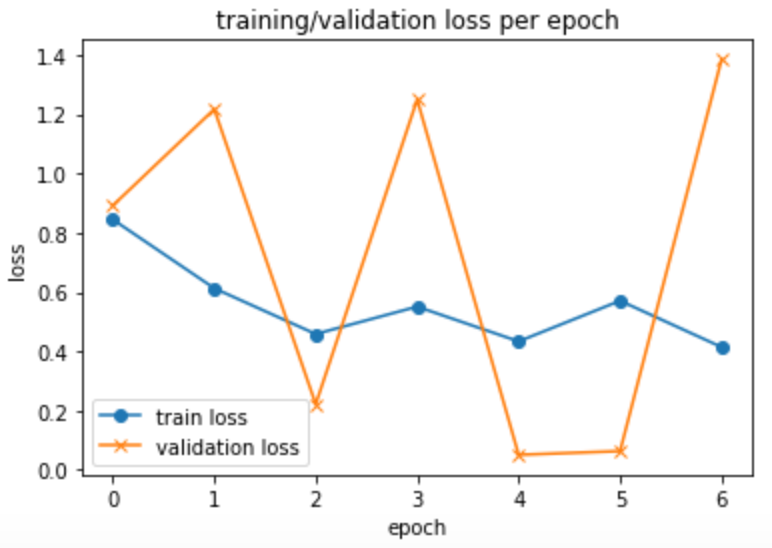
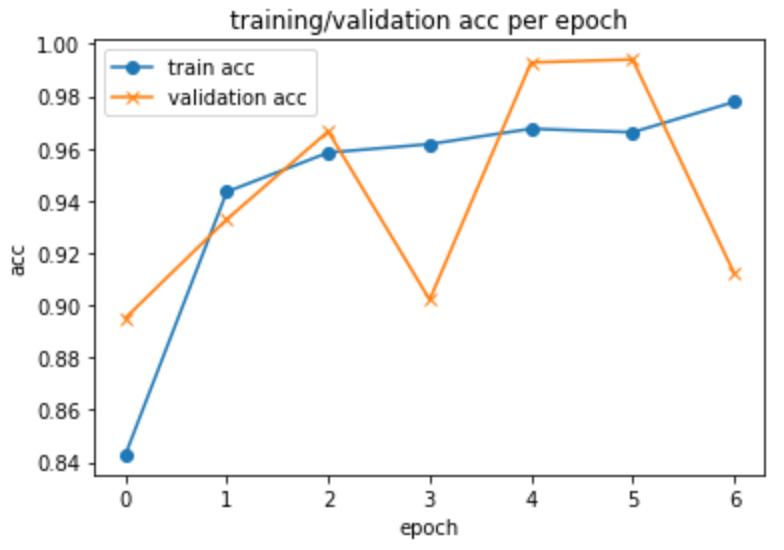
**7.3.2 LSTM**

For binary classification, we decided to move forward with a similar architecture to the one used in Attempt 3 of our earlier LSTM model. Besides the removal of the categorical data, the only other difference was that we increased the batch size from 128 to 256.



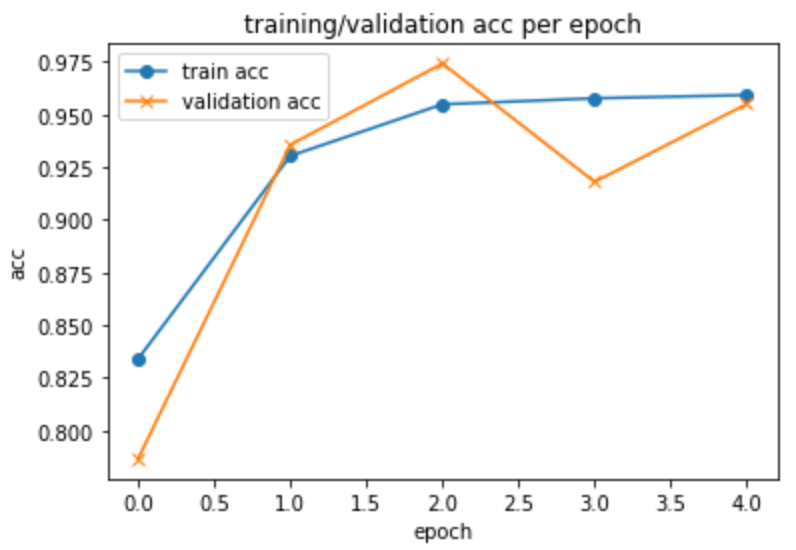
*Fig 19.* CuDNN Architecture - Removing Categorical Data

By removing the categorical attributes, we observed drastic improvements within the model as the model converged in less than 10 epochs and accuracy on the test dataset improved to 0.9144 (Fig 20, 21).



|  |  |
| --- | --- |
| *Fig 20.* Training/Validation Accuracy - NLP - Attempt 1 | *Fig 21.* Training/Validation Loss - NLP - Attempt 1 |

Given the improvements in the model when the number of neurons increased from 128 to 512 within our multiclass models, we decided to run one last model. The final model run for the project also saw improvements over our original model as accuracy on the test dataset increased to 95% and the model converged within 4 epochs (Fig 22, 23).



|  |  |
| --- | --- |
| *Fig 22.* Training/Validation Accuracy - NLP - Attempt 2 | *Fig 23.* Training/Validation Loss - NLP - Attempt 2 |

Given the improvement in accuracy, along with the fact that the model converged in less epochs and time than the original models, we concluded that the dataset we worked with is more suited for binary classification.

**8 References**

1. Michale Luca. Reviews, Reputation, and Revenue: The Case of Yelp.com

2. Yelp Open Dataset. <https://www.yelp.com/dataset>

3. Bryan Hood, Victor Hwang and Jennifer King. Inferring Future Business Attention

4. Past Winners. <https://www.yelp.com/dataset/challenge/winners>

5. Nabiha Asghar. Yelp Dataset Challenge: Review Rating Prediction . arXiv preprint arXiv:1605.05362, 2016.

6. Guozheng Rao, Weihang Huang, Zhiyong Feng, Qiong Cong. LSTM With Sentence Representations For Document-level Sentiment Classification. Neurocomputing 308 p. 49-57, 2018.

7. sklearn.feature\_extraction.text.CountVectorizer. <https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html>

8. Jiacheng Xu, Danlu Chen, Xipeng Qiu, Xuanjing Huang. Cached Long Short-Term Memory Neural Networks for Document-Level Sentiment Classification

9. <https://developer.nvidia.com/cudnn>

10. sklearn.linear\_model.LogisticRegression. <https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html>

11. Mikael Rusin. What is the C parameter in logistic regression?. <https://www.quora.com/What-is-the-C-parameter-in-logistic-regression>

12. Aravind Pai. CNN vs. RNN vs. ANN – Analyzing 3 Types of Neural Networks in Deep Learning. <https://www.analyticsvidhya.com/blog/2020/02/cnn-vs-rnn-vs-mlp-analyzing-3-types-of-neural-networks-in-deep-learning/>

13. pandas.DataFrame.sample. <https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.sample.html>

14. Christian Freischlag. [Deep Learning] Combining numerical and text features in (deep) neural networks. <http://digital-thinking.de/deep-learning-combining-numerical-and-text-features-in-deep-neural-networks/>

15.recurrent initializer == Orthogonal

<https://smerity.com/articles/2016/orthogonal_init.html>

16.glorot\_uniform

<https://datascience.stackexchange.com/questions/13061/when-to-use-he-or-glorot-normal-initialization-over-uniform-init-and-what-are>

**9 Project Contributions**

* Academic Literature Review + Online Research : Harish Anumanchineni, Greg Araujo
* Data Pre-Processing : Harish Anumanchineni, Greg Araujo
* Machine Learning Models : Harish Anumanchineni, Greg Araujo
* Deep Learning Model : Harish Anumanchineni, Greg Araujo
* Project Report : Harish Anumanchineni, Greg Araujo