Project Report
ILS-Z 534 Search
Fall 2019

Chrislin Priscilla
Dhruuv Agarwal
Siddharth Kothari
Varun Miranda

December 16, 2019

1 Problem Statement

Our project goal is to recommend businesses to users. There are various techniques to implement this task. We have divided our approaches into Task 1 and Task 2. In the first task we aim to recommend businesses to users based on Non-Negative Matrix Factorization and Neural Attentional Rating Regression with Review-level Explanations and compare the results between them. In the second task, we decided to predict whether a review is useful or not.

2 Task 1A: Non-Negative Matrix Factorization

2.1 Motivation

The users have not rated many businesses and hence the data is very sparse. For sparse data model based collaborative filtering is usually preferred over memory based collaborative filtering. The reason being that memory based approach finds similar users based on cosine similarity which will not perform well because of the data not being well populated. On the other hand, model based approach specifically uses machine learning methods to find user ratings of unrated businesses. In model based approaches we have neural networks, SVD and matrix factorization. We have chosen non-negative matrix factorization to fill the user-business matrix.

2.2 Data Processing

We have used the same dataset for all the models to efficiently compare the different models in Task 1.

Data Files and features that were used for Task 1:

- business.json
    - business_id
    - state
    - categories

- review.json
    - review_id
    - user_id
    - business_id
    - stars
    - text

Filters applied:

1. business/categories: We have filtered for restaurants using the categories column. Any business that has restaurants or food in the category is classified as a restaurant
2. business/state: Restaurants were then filtered for the state of Nevada
3. Sparsity Reduction: We have only chosen business ids with minimum 50 reviews and users who have given minimum 20 reviews for reducing sparsity in the matrix. This is to ensure that the data is not too sparse and at the same time maintaining enough businesses and users in the matrix.

Number	**Filters**	**Samples**
0	Initial Dataset	6,685,902
1	Restaurants in Nevada	1,391,673
2	Business > 50 reviews and Users > 20 reviews	278,425

After dropping unrelated columns we have a data frame with user id, business id, and ratings. The dataframe was transformed into a matrix which resulted in 6113 users and 3940 business which had a decent ratio of 2:1 for users:business.
Note: Ratings were averaged for multiple entries for a user business pair.

- business.json
    - business_id
    - state
    - categories
- review.json
    - review_id
    - user_id
    - business_id
    - stars
    - text

2.3 Testing Set: User-Business pairs

We randomly chose 20 percent of the values in the populated matrix and replaced them with zeros in the matrix. While sampling this 20 percent cells, we made sure of the following:

- Corresponding users have rated atleast one other business
- Corresponding restaurants are rated by atleast one other user

2.4 Algorithm Implementation

The user-business matrix, denoted by R will be decomposed into two matrices P and Q where P will have the dimensions of all users and K latent features and Q will have the dimensions of all businesses and K latent features (where the value of K is tuned). The product of P and Q^T should approximate the value of R (denoted by $\hat{R}$).

$$R \approx P * Q^T = \hat{R}$$

The best value of K is chosen to minimize the difference between R and $\hat{R}$. To minimize the error, we need to know the gradient at the current values and update the gradient in the opposite direction. The learning rate α influences the change in the gradient.

Regularization is also performed to avoid overfitting. It penalizes the loss function when the model complexity increases. The value β denotes the regularization parameter which can also be tuned.

$$\boxed{\begin{aligned} \frac{\partial}{\partial p_{ik}} e_{ij}^2 &= -2(r_{ij} - \hat{r}_{ij})(q_{kj}) = -2e_{ij}q_{kj} \\ \frac{\partial}{\partial q_{ik}} e_{ij}^2 &= -2(r_{ij} - \hat{r}_{ij})(p_{ik}) = -2e_{ij}p_{ik} \end{aligned}}$$

Figure 1: Gradient Descent

Once the algorithm has been implemented, we utilize different values of K and calculate the RMSE between R and the predicted R. Whichever K value gives the least RMSE, we consider that reconstructed matrix as our final prediction. With our reconstructed matrix we can calculate our test RMSE.

2.5 Conclusion

As the ratings given by users for multiple businesses is sparse, hence we cannot expect a great accuracy to come out of it. The good thing about this model though is that it can be learnt online. That means, when a user visits another business in the future, we can expect the rating to be populated and the accuracy of the model to improve, thereby giving better recommendations to users.

3 Task 1B: NARRE

3.1 Motivation

Matrix factorization makes a recommendation based on previous ratings alone. The recommendations can be improved by using the reviews data. Reviews provide much more information regarding the user sentiment towards the business than ratings. However, all reviews are not the same. Some reviews can be extremely useful, and some can provide no information at all. Not only that some reviews provide no useful information but it can be argued that these reviews can impede the model's performance. Most of the models use reviews to build a better recommendation system consider all reviews the same and don't try to explore how useful is the review. Neural Attentional

Regression model with Review-level Explanations (NARRE) tries addressing this problem of recommendation system. NARRE works well in predicting the ratings accurately and additionally learning the importance of each review.

3.2 Data Processing

Yelp data has different business data. We have filtered for restaurants using the categories column. Any business that has 'restaurants' or 'food' in the category is classified as a restaurant. These restaurants were then filtered for the state of Nevada. These business ids were used to filter the reviews data leaving us with total of 1.3 million reviews.

We created two datasets after filtering for the state:

1. Dataset I: We took the entire dataset without filtering it further. This data is extremely sparse and has many users with only one review. The unique business to unique user ratio is 0.1
2. Dataset II: As we need to compare the model with nonnegative matrix factorization, we have only chosen a business with atleast 50 reviews and users with atleast 20 reviews. This data is comparatively less sparse with unique business to unique user ratio around 0.5

Both the datasets are then treated the same way to generate input data. After dropping unrelated columns, we have a data frame with user id, business id, reviews, and ratings. The user id and business id are alphanumerical therefore we converted this to just numerical ids. The reviews are then tokenized using the English module of the spacy dictionary. The tokenized review is cleaned by removing stop words, punctuation, numbers, and special characters.

In the input data every datapoint needs to have the same size therefore we padded the reviews and list of all corresponding business/user ids. As there are few users who have an extremely large number of reviews and the same for business, therefore, we have taken 90 percentile to set as the limit for all the things mentioned above. To pad the review token we have added 'PAD/' to end of reviews which were shorter than the max length and if a review token has more token then maximum limit we removed the extra ones. To pad list of business/user ids we have added '0' to end of list which is shorter than the max length and if a list is bigger than maximum limit we removed the extra ones. To build the vocabulary we used itertools and Counter to get the index and the respective vocabulary.

The input data to the model are all the reviews for the user, all the reviews for the business, all the business ids for the user, all the user ids for business, user id and business id. The model predicts the review which ranges between 1 and 5.

3.3 Train Test Split

The data is split into test and train. For dataset I, the data was divided using test_train_split function provided by scikit learn library. The test is 20% of the total data present. For dataset II, we have takem only 5% as test as it was kept consistent with the non negative matrix factorization task to make them comparable.

For each user in the train data, we found all the reviews they have given and all the business they have reviewed. For valid data, if the user is not present in the train data (indicating a case of cold start) then the reviews of the data are kept as 'PAD/' and business id as 0.

3.4 Model Architecture

The model has two identical structures, one for user-level and other for business/item level. After the networks, we learn the latent feature embedding for each, and finally, pass them to the prediction layer. Each of the two blocks is built in a similar fashion and consists of two critical elements: CNN block and Attention Pooling.

1. Text CNN based Convolution layer
2. Review Attention Pooling
3. Prediction Layer

3.4.1 Text CNN based Convolution

The first step in the model is to process the review information. Unlike DeepCoNN, where they concatenated all review for a user as one entity and then passed it to the convolution layer, here in NARRE, each review has a separate entity and which results in separate features. Intuitively, this makes the feature extraction process more granular, as we get features from each review, and don't merge them together. Merging them together can lead to some reviews dominating the features learned from all reviews. The process of using these separate features to learn one set of features will be discussed further in the Attention Pooling section later.

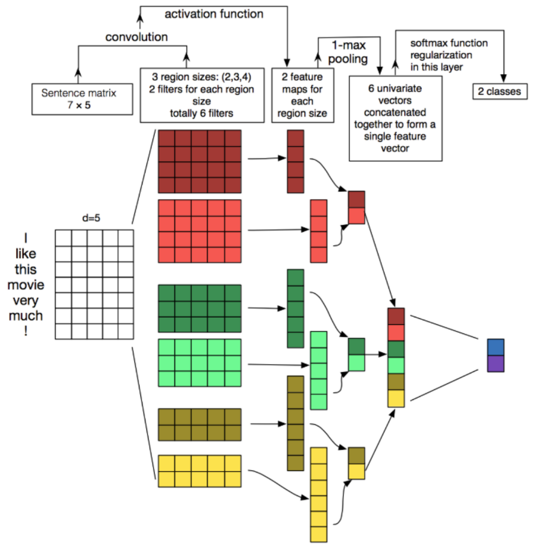
activation function
convolution
1-max pooling
softmax function regularization in this layer
Sentence matrix 7 × 5
3 region sizes: (2,3,4) 2 filters for each region size totally 6 filters
2 feature maps for each region size
6 univariate vectors concatenated together to form a single feature vector
2 classes
d=5
I
like
this
movie
very
much
!

Figure 2: TextCNN architecture for Text Classification

Above we have the image for the TextCNN architecture, as it was originally proposed. Hereafter we have the word to vector matrix for the sentence, we apply convolution with different filter sizes. The intuition of filter size here is the same as n-gram language models. Filter size of 3 will look at 3 words at one pass, like n-gram where one word depends on past n-1 words.

As the model keeps the entire features to be passed to Attention pooling later, it doesn't have a maxpooling layer or softmax layer as shown in the diagram(diagram for text classification, the task in original TextCNN paper).

3.4.2 Attention Based Pooling

The main distinguishing point of this approach was using one of the most promising concepts in Deep learning, namely, Attention.

Below is the diagram for the Narre model as described in the paper.

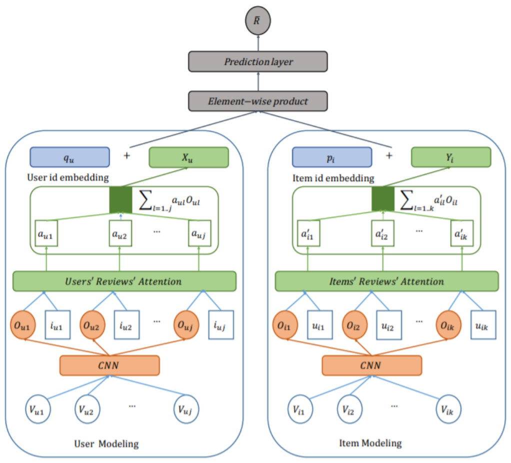

Figure 3: NARRE architecture as described in original paper

Consider one block, say for a user. We have V (i-¿j) set of reviews, which correspond to item id (i-¿j). We also have features from the convolution layer O (i-¿j).

The attention equation for this setup as discussed in the paper is,

$$a^*_{il} = h^T ReLU(W_O O_{il} + W_u u_{il} + b_1) + b_2$$

Figure 4: Attention architecture for User and Business level

Now, to implement this there is a two-layer network. The first layer, brings features and item id info to latent dimension, and then adds them and feeds to Relu activation(to ensure positive values). Then this is again passed to the Dense layer, to finally get attention weight vector.

Pseudo code for the above is below:

$$Dense(Relu(Add(Dense(latent)(O), Dense(latent)(i)))) \tag{1}$$

After this attention weights are multiplied to corresponding review features and aggregate them.

Hence, instead of just adding all features or taking an average of them, attention network helps to give proper importance to selected reviews and items for a user. (Vice versa for business network)

3.4.3 Prediction Layer

The paper makes predictions by using the concept of the latent factor model. This involves learning a score from the latent user and business embedding.

Here, the Score can be calculated as the dot product of the latent representation of user and business. But in the implementation, this process of the dot product(Hadamard product and sum) is further divided, by first multiplying the two vectors and then inserting a dropout layer in between before eventually taking the sum.

Then user bias, item bias and global bias are added to this score to get the final score.

$$Finalscore = Score + userbias + businessbias + globalbias \tag{2}$$

In our implementation, we have found that the bias addition, given our dataset, doesn't lead to better training. We got better scores for train and validation sets, without the addition of these biases.

3.5 Experiments And Findings

Initial parameters: Convolution Filter sizes = [3,4], filters=100, Adam(lr=0.001) Dropout=0.4

Dataset1

Based on our experimentation, this model can suffer from overfitting, especially for small datasets, with a size of less than a million samples. This is due to a large number of parameters in Embedding layers on id's, compared to the convolution layer applied to reviews. This number of embedding parameters directly depends on max number of users, and as it had very fewer reviews per user and a large ratio of Users count to samples count, this was bound to overfit for Dataset1.

To handle overfitting we experimented with increasing Dropout, adding regularization on Attention network and Regularization on embedding layers. L2 regularization added to layers directly with lambda 0.001. Reducing the latent dimension from 32 to 16 also helped in reducing overfitting.

Regularization on the embedding layer, helped drastically in Dataset1, thus confirming the fact that embedding layer was the mainly responsible for the overfit for Dataset1.

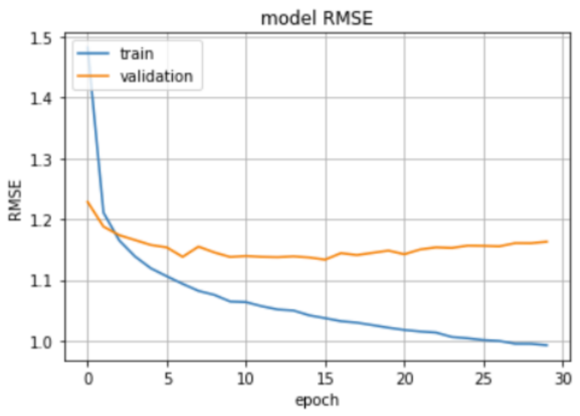

Figure 5: Dataset 1 Train/Test Graph

Dataset2

Dataset2 differs from Dataset 1 in some key aspects, like the user to sample ratio and user to business ratio.

In Dataset2, when the number of samples per user is much more, naturally we expect the model to learn more info about users, and not base the estimate on say 1-2 reviews like it would in Dataset1.

Hence, we found as we had multiple samples of reviews for a user and similarly, for a business, we end up having a lot more balanced dataset. This balance prevents learning noisy embeddings for users and businesses. (Fewer samples – more noise estimate).

The latent dimension was kept at 32 for this dataset, as increasing it resulted in sharp decreasing trend in training loss, which was undesirable.

Thus, Dataset2 didn't suffer from overfitting much, and so we experimented with removing regularization for the embedding layer and also reducing the Dropout applied.

Below is the final Test validation RMSE graph with Regularization removed from embedding layer.

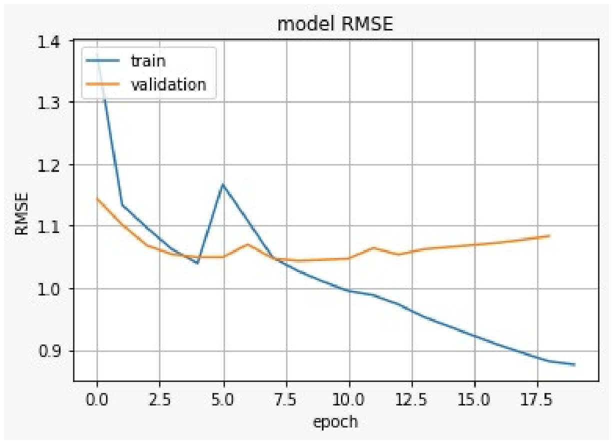

Figure 6: Dataset 2 Train/Test Graph (No embedding regularization)

Below is the final Test validation RMSE graph with Dropout reduced to 0.2. As we can see both graphs lead to same validation scores eventually. However, we can see that removing regulaization on embedding layer, results in much sharper decrease in Train RMSE, and then it starts to overfit.

A reduction in Dropout from 0.4 to 0.2, keeping the regularization, however keeps the training more stable.

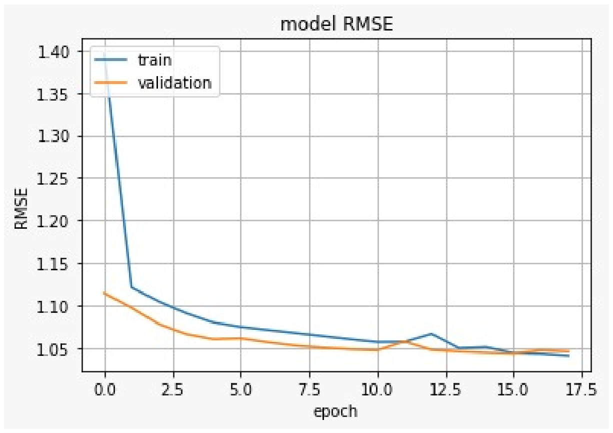

Figure 7: Dataset 2 Train/Test Graph (Dropout 0.2)

4 Task 1: Evaluation

Root Mean Squared Error: It's the square root of the average of squared differences between prediction and actual observation.Since the errors are squared before they are averaged, the RMSE gives a relatively high weight to large errors. This means the RMSE should be more useful when large errors are particularly undesirable.

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(y_j - \hat{y}_j)^2}$$

Figure 8: RMSE

Task	Method	Test RMSE
1A	Non Negative Matrix Factorization	1.5
1B	Neural Attentional Regression model with Review-level	1.04

5 Task 1: Limitations

1. Limitation 1: Yelp Dataset is a real world dataset, hence its prone to a lot of sparsity. Sparsity is especially bad for Collobrative filtering methods like the ones used in Part 1.
2. Limitation 2: Limited Resources to work and test the model on less filtered data.

6 Task 2: Usefulness of Reviews

6.1 Motivation

The topic we have selected to implement for Task 2 is to predict the usefulness of reviews. It is a classification problem that will predict if a given review will be useful or not based on certain factors.

Knowing the usefulness of a review in advance, businesses can recommend high quality and fresh reviews to their customers and gain insights into their products and services.

There are two aspects that we considered that are strong motivation points for this:

How this task is helpful to the user: Usually when the users wants to browse for possible restaurants to visit. They will probably read so many reviews, that there is a very high chance most of the reviews they encounter does not convey the information they are looking for. In other words - It is not useful.

By tagging the review as useful or not, we aid the users by telling in advance about the quality of information of the review and save them the trouble of reading not useful reviews. Thus, user will save a lot of time

How does the business benefit from this task: This task is not only beneficial to the users, it also helps the business. The users aren't the only ones who read the reviews, the business also read reviews to understand positive and negative feed backs about the restaurants.

Suppose a user complains that the air conditioner is not working. The business will know exactly what problem is the customer facing and act accordingly. This task will impact the business as it can direct business to important reviews and save time by not going through all the reviews to find useful ones.

So, just like for the user, by tagging the review as useful or not the business will analyze their shortcomings faster, for example and hence this task is essential for review filtering.

6.2 Data Processing

We have used the same dataset for all the models to efficiently compare the different approaches in Task 2.

Data Files and features that were used for Task 2:

- business.json
  - business_id
  - stars
  - review_count
  - stars
  - categories
- review.json
  - review_id
  - user_id
  - business_id
  - stars
  - date
  - text
  - useful
- user.json
  - user_id
  - review_count
  - useful
  - average_stars

6.2.1 Target Variable for Classification

:

- class_useful = 1 if useful votes for review > 0
- class_useful = 0 if useful votes for review $= 0$

6.2.2 Filters applied:

1. review/date: Reviews before 2017 were taken because it takes time for a good review to accumulate a large number of useful votes
2. business/business_id: Reviews filtered for US based Restaurants
3. review/class_useful: Down-sampled data for an easier computation
4. review/text: Reviews were tokenized and stop words and special characters were removed using spaCy. And then only the reviews with length $>= 5$ were taken into account to account for good quality reviews

Number	Filters on Review Data	Samples
0	Initial Dataset	6,685,900
1	Review Year < 2017	2,231,445
2	US Restaurants	4,290,565
3	Down-sample based on Useful Class	400,000 (Class 1: 200,000; Class 0: 200,000)
4	Length of Review >= 5	399,649 (Class 1: 199,877; Class 0: 199,772)

6.2.3 Train Test Split:

Train dataset: 267764 rows and 107 columns

Test dataset: 131885 rows and 107 columns is achieved using train_test_split() in sklearn after one hot encoding of categorical variables.

Note: Same train test files are used across all models mentioned below.

6.3 Algorithm Implementation

Our initial hypothesis is to analyze whether the review text alone can decide if the review is useful or not. So that is our baseline model. In addition to the baseline model, the second implementation will add an additional layer of input to the existing baseline model. That layer would be a top-100 category vector. The third implementation will use a machine learning approach to generate a new vector which will also be fed as an input on top of the existing baseline.

6.3.1 Model 1 (Traditional Machine Learning):

We considered two different supervised machine learning models to predict the usefulness of reviews.

KNN: One of the traditional machine learning algorithm that assumes similar things exist in close proximity. In other words, similar things are near to each other.
After tuning parameter K with GridSearchCV, we found that K=14 best suited our dataset and we achieved an AUC of 66% on test set.

Random Forest: Random Forest is a tree based approach to try and classify the data points. The Decision Tree structure will be a set of decisions that will try to send the incoming data point to the left or the right children. A maxvoting of Decision Trees would be the Random Forest network. We have plotted a feature importance plot to understand which features are important.

Following parameters were trained in GridSearchCV and used in final model:

- n_estimators = 150
- criterion = Gini
- max_depth = 15
- max_features = sqrt

and we achieved an AUC of 68% on test set.

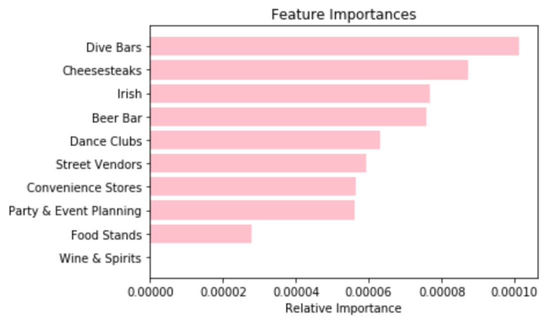

Figure 9: Feature Importance Plot

6.3.2 Model 2 (Bidirectional LSTM using Review Text):

With the reviews, the embedding matrix is created. The embedding matrix is a list of vectors where each review is a vector and thereby each text has a number associated with it. These numbers are pulled from GloVe and Crawl. The numbers are helpful because words like good and great will be given a similar number value.

Once the matrix is generated it is fed into a bidirectional LSTM. LSTM is Long Short Term Memory which aims to not forget the old information in the review. That means when the matrix is scanned there are some key words in the beginning of the review which are important later on and traditional RNN tend to forget this information. We use bidirectional LSTM because it is important to see what appears before and after the text when it is being processed in the network.

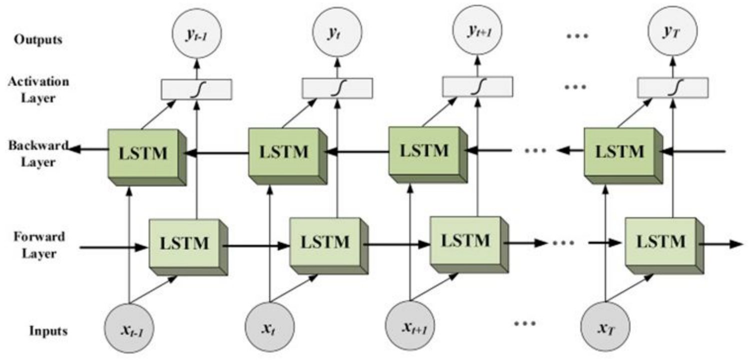

Figure 10: Bidirectional LSTM

To avoid overfitting we can introduce dropouts. Dropouts randomly removes some neurons from the network. There are two kinds of dropouts we have introduced. Spatial Dropout always drops the neuron at the same index and Dropout drops neurons at random indices.

A combination of Max Pooling and Average Pooling is also done. The concept of pooling happens to best generalize the input while down sampling the data. An example to depict Max Pooling and Average Pooling on a matrix of values is depicted in the image below:

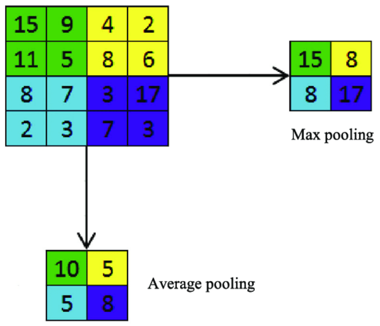

Figure 11: Max Pooling and Average Pooling

The activation function used in this network are of two types: ReLU and Sigmoid. ReLU converts all negative values to zero and they are used in all neurons except the final dense layer. The reason ReLU is used in the network is because we need it to be sparsely activated. The zero values will void the activation, which is good. The linear part of ReLU on differentiating will give a constant value so it will activate and not suffer from the vanishing gradient problem.

Sigmoid functions on the last layer makes sense as it is a classification problem and it should activate to 1 or 0. The sigmoid function computes the probability of the class and sets the activation threshold to 0.5. If the probability of the class is above 0.5 it will classify that as 1 and if the probability of the class is below 0.5 it will classify as 0. 1 depicting that the review is useful and 0 depicting that the review is not useful.

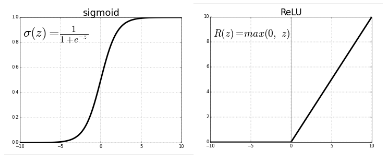

Figure 12: Activation Functions

The final dense layer will train a fully connected network on the output of the Bidirectional LSTM network and finally feed the output to a sigmoid activation function to predict usefulness or not.

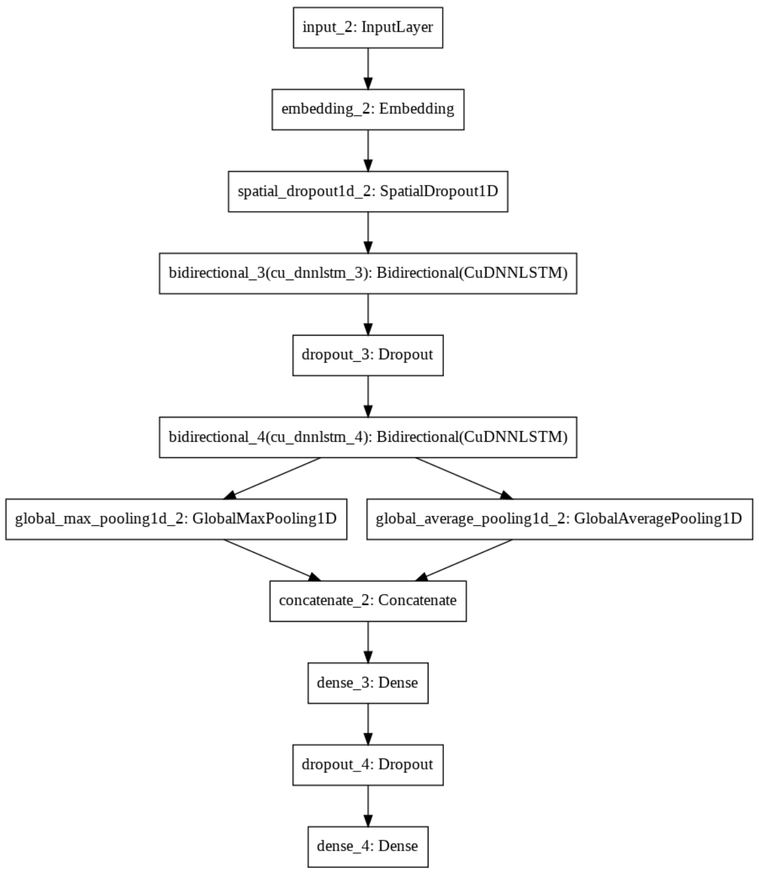
input_2: InputLayer
embedding_2: Embedding
spatial_dropout1d_2: SpatialDropout1D
bidirectional_3(cu_dnnlstm_3): Bidirectional(CuDNNLSTM)
dropout_3: Dropout
bidirectional_4(cu_dnnlstm_4): Bidirectional(CuDNNLSTM)
global_max_pooling1d_2: GlobalMaxPooling1D
global_average_pooling1d_2: GlobalAveragePooling1D
concatenate_2: Concatenate
dense_3: Dense
dropout_4: Dropout
dense_4: Dense

Figure 13: Model 2 Architecture

6.3.3 5

Final Model (Bidirectional LSTM with Review Text, Categories and Extra Features): Here we are adding extra input layers to our existing Deep Learning network which is defined by:

1. Category vector: In order to create the category vector, we chose the categories column from the business dataset. As there were as many as around 690 categories we chose the top 100 most common categories. It was meaningful to provide that filter because a lot of categories occur very few times in the dataset and it won't be advisable to use that in our model. So we created a binary vector for each data point in the business dataset where the vector position is populated with 1 and the remaining positions of that vector are filled with 0.

2. Extra features (User and Business): We felt the need to include additional features because we had an intuition that the credibility of the user matters for instance, when it comes the usefulness of the review. So we wanted to test that hypothesis. We used the following features in creating a new vector so that this vector can also be added as an input to our architecture.

1. User Rating of Business
2. Business Review Count
3. Business Rating

4. User Rating
5. User Review Count
6. Useful Votes User
7. Fans

These will be added to new dense layers respectively where the network will try to learn the equation from the category vector. The output of the dense layer will be concatenated with the output of the embedding matrix after the latter passes through the LSTM layers. Post that it will pass through the final dense layer where the sigmoid activation function will try to predict usefulness as 1 or 0.

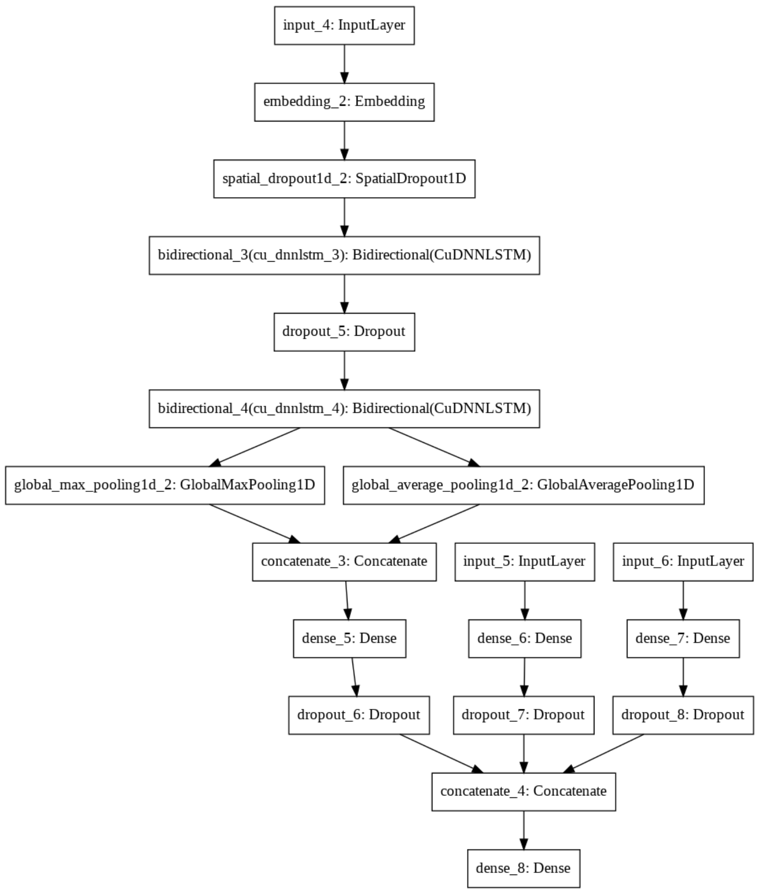
input_4: InputLayer
embedding_2: Embedding
spatial_dropout1d_2: SpatialDropout1D
bidirectional_3(cu_dnnlstm_3): Bidirectional(CuDNNLSTM)
dropout_5: Dropout
bidirectional_4(cu_dnnlstm_4): Bidirectional(CuDNNLSTM)
global_max_pooling1d_2: GlobalMaxPooling1D
global_average_pooling1d_2: GlobalAveragePooling1D
concatenate_3: Concatenate
input_5: InputLayer
input_6: InputLayer
dense_5: Dense
dense_6: Dense
dense_7: Dense
dropout_6: Dropout
dropout_7: Dropout
dropout_8: Dropout
concatenate_4: Concatenate
dense_8: Dense

Figure 14: Final Model Architecture

6.4 Model Evaluation:

Accuracy Metric: As the task is a classification problem and the classes are balanced, we have chosen AUC as the accuracy metric to evaluate our model.

The AUC (Area under the curve) or ROC (Receiver operating characteristic) is the graph plotted between the True Positive Rate (Y-axis) and the False Positive Rate (X-axis). Considering the entire area of the enclosed region as 1, the area covered under the curve plotted depicts the AUC. A pictoral representation is given below:

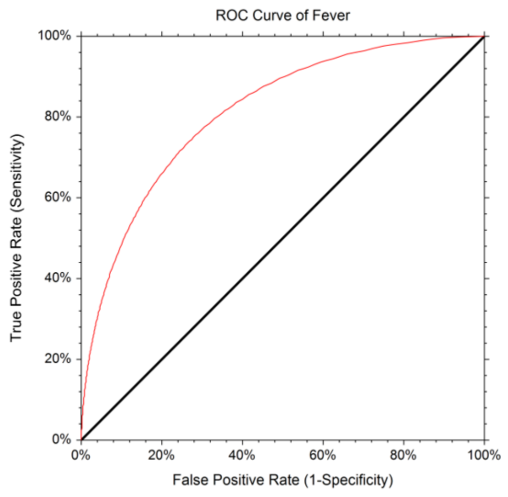

Figure 15: AUC-ROC Curve

Number	Algorithm	Train AUC (%)	Test AUC (%)
1	k-Nearest Neighbors	73	66
2	Random Forest	75	68
3	Bi-directional LSTM on Only Reviews	69.17	69.7
4	Bi-directional LSTM on Reviews and Other Features	75.6	77.1

6.5 Conclusion:

Classifying reviews as useful can turn out to be a slightly challenging task as the two reviews below are similar in sentence construct but will be tagged differently due to the lack of useful votes in the second review. For example:

1. We decided to go here for the $32 summer special. We went early to have a glass of wine and an appetizer before dinner. It was very comfortable and enjoyable sitting and listening to the live music. The food and service was amazing both upstairs and downstairs. We would highly recommend the entire experience. What a great find!!" ["votes": "funny":0, "useful":70, "cool":70]

2. I love this place! The food is excellent; the atmosphere is fabulous! Went there for a girls night out and we were treated to a wait staff of all very handsome men! They were courtesy and attentive and that, along with the decor, music and food made it a terrific night out!" ["votes": "funny":0, "useful":0, "cool":0]

Apart from the business category, the attributes can be explored as well. Business attributes (ambience etc.) can be incorporated in the model to provide a better prediction of the usefulness of reviews.

7 Contribution:

The contribution by the team is as follows:

Task	Data Creation	Model
Task 1A: NMF	Varun Miranda	Chrislin Priscilla
Task 1B: NARRE	Siddharth Kothari	Dhruuv Agarwal
Task 2: Machine Learning	Dhruuv Agarwal	Siddharth Kothari
Task 2: Deep Learning	Chrislin Priscilla	Varun Miranda

References

[1] https://www.michelle9luo.com/uploads/1/1/4/5/114540565/yelp_data_challenge.pdf

[2] https://s3-us-west-2.amazonaws.com/auto-ap-media/Final_Presentation.pdf

[3] Luke de Oliveira and Alfredo Lainez Rodrigo. 2015. Humor detection in yelp reviews.

[4] https://towardsdatascience.com/recommendation-system-series-part-2-the-10-categories-of-deep-recommendation-systems-that-189d60287b58

[5] Lei Zheng, Vahid Noroozi, and Philip S. Yu. Joint deep modeling of users and items using reviews.

[6] Chong Chen, Min Zhang, Yiqun Liu, and Shaoping Ma. 2018. Neural Attentional Rating Regression with Review-level Explanations. In Proceedings of the 2018 World Wide Web Conference on World Wide Web. International World Wide Web Conferences Steering Committee, 1583–1592. for recommendation. In WSDM, pages 425–434, 2017.

[7] http://nicolas-hug.com/blog/matrix_facto_3

[8] http://www.quuxlabs.com/blog/2010/09/matrix-factorization-a-simple-tutorial-and-implementation-in-python/

[9] https://blog.insightdatascience.com/explicit-matrix-factorization-als-sgd-and-all-that-jazz-b00e4d9b21ea

[10] https://scikit-learn.org/stable/

[11] https://keras.io/

[12] David Zhan Liu. Understanding and Predicting the Usefulness of Yelp Reviews.

[13] Yoon Kim. Convolutional neural networks for sentence classification. EMNLP, 2014.