CIS 520 Fall 2019, Project Proposal

Y3RP - Yelp Restaurant Review Rating Prediction

Team Members:

- Mihir Parmar; Email: `mihir98@seas.upenn.edu`
- Pooja Consul; Email: `pconsul@seas.upenn.edu`
- Tejas Srivastava; Email: `tjss@seas.upenn.edu`

1 Abstract

We work on the Yelp Dataset[1] to predict ratings and sentiments (positive and negative) for restaurant reviews. There are 1 million reviews written in free text making the problem challenging. Our project explores two main ideas, first, performance of CNNs for text classification. We show that a CNN model works at par with LSTMs traditionally used for text classification tasks specifically rating prediction in the context of this project. Secondly for the sentiment classification task we use only the aspect descriptors. This method works at par with using the entire review for CNNs.

2 Motivation

With the rise in e-commerce platforms and other platforms offering consumer services to end-users, the importance of reviews and ratings of the services and products has also increased significantly; thus, making these reviews and ratings an integral part of these platforms. Not only these reviews help the users take informed and quick decisions about the products and services being offered, but also offer insights to the companies about the acceptance of their products and the success of these products among the user base, thus having an impact on the revenue generated and helping to develop business intelligence.

The complexity of textual reviews and the fact that the relationship between reviews and ratings is circuitous make this problem a challenging one. Each review has a basic structure that describes the aspects or features of the object being reviewed, and their descriptors. These two values should be in close proximity. One way is to utilise bigrams. We try to leverage CNN to extract this information with Conv1D filters to make the rating predictions. CNNs perform excellently in extracting n-gram features at different positions of a sentence through convolutional filters, and can learn short and long-range relations through pooling operations. Secondly for sentiment classification each aspect we use, adjectives as our descriptor. To allow the adjectives to contain the context or the describes we create embeddings on the reviews and use the embeddings for adjectives.

3 Related Work

Most of the past work in review rating prediction utilises sentiment analysis. This involves aspect extraction and determining the correct sentiment label(generally positive and negative) for it. Other methods make use of domain specific embedding to understand the context of aspect. Recent methods involve CNN, LSTM for sequence learning and thus predicting the review rating. For the purpose of this project we did an extensive literature review. The approaches used in the past for review rating prediction and relevant techniques for aspect extraction and sentiment analysis have been elicited below.

[1]https://www.yelp.com/dataset

The winners of 2018 Yelp Dataset challenge[2] proposed a method [2] utilising both general and domain specific embedding. They used CNN for sequence learning task to predict whether a word is an aspect word or not. Their architecture consisted of 2 embedding layers, 4 CNN layers and a fully connected layer. The paper threw light on some key ideas for this project. Using combination of general embeddings and domain specific embeddings(even though corpus not large) and letting network decide which one to use (e.g. speed miles per hour, speed in context of electronic device) works well. Embedding layer was the first layer in the network, choosing right embedding determines the ability of deeper layers to extract useful information. CNNs are used for sequence learning instead of LSTMs in order to overcome the drawback of LSTMS i.e. sequentially dependant cells, for each forward and back pass have to go through entire sequence which makes training slow training. Since CNNs work well for this problem, it opens door to try out other CNN based architectures. Another noteworthy detail in the paper was that max pooling wasn't used in the CNN layers. This was because it mixes the representation of words. For this task each word requires its own representation.

Jiangtao Qiu and Yinghong Li in their work on review rating prediction [3] proposed a method based on semantic analysis. They proposed that rating of review depends on sentiment of aspects, positive and negative words in the review. Sentiment score of aspect is defined by its context. They identified term pairs occurring in the context of an aspect and assigning scores for such term pairs. They reasoned based on observation that a review involves at least one aspect; we may roughly get positive sentiments from all aspects of a 5-star rating review whereas all aspects in a review with 1-star rating tend to be negative sentiments. They used the idea of positive and negative word count for giving ratings. The 3-star rating for instance should have roughly equal number of positive and negative words. The performance of their method is useful in utilising the idea to the methods we develop.

Seo, Huang, Yang and Liu proposed a different approach [4] towards predicting ratings from reviews to model user preferences and item properties by a hybrid deep learning method in which they aggregated reviews from a user and reviews for an item, trying to learn the unique features of the users and the items, and further using these features to predict ratings. Their model architecture is based upon a user network and an item network, both of which take input as reviews aggregated by a user and reviews for an item, respectively. Both the user network and the item network utilize the dual attention model, a local attention module to capture details about user preference or business properties, and a global attention module which helped in capturing the semantic meaning of the whole review. The global and local representation of words, is then passed through convolutional neural networks, followed by maxpooling. The outputs are further concatenated and passed through fully connected layers to get a semantic vector from the user network and the item network (latent representations), and the final rating is then estimated by taking the inner product.

Nabiha Asgar in her work[1] treated review rating prediction as a multi-class classification problem and built 16 different models using a combination of feature extraction methods and machine learning algorithms The four different feature extraction methods used were unigrams, unigrams + bigrams, unigrams + bigrams + trigrams, and latent semantic indexing. The four machine learning algorithms used were logistic regression, naive bayes classification, perceptrons, linear support vector classification (SVC). This resulted in 16 different models with each being a combination of a feature extraction method used with a machine learning method. The best two results were obtained for unigrams + bigrams feature extraction.

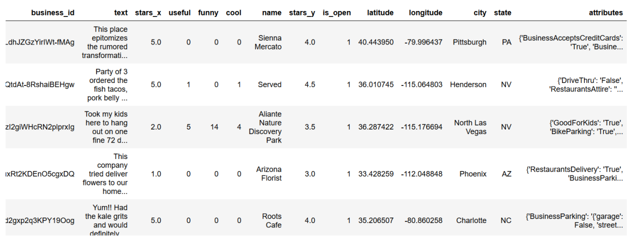
4 Dataset

The dataset used for this problem is provided by Yelp[3] as part of their Dataset Challenge 2019. The dataset consists of six files, one for each object type: business, review, user, checkin, tip, and photo. A business is represented by the business.json file which contains its business ID, name, location review counts, etc. The text review is represented by the review.json file which includes the business ID, user ID, ratings, review text, date and votes. **The necessary data for our problem is included in the business and review**

[2]https://www.yelp.com/dataset/challenge/winners
[3]https://www.yelp.com/dataset

files and hence we do not use rest of the data.

However, we constrained our problem to the domain of restaurant reviews. To extract restaurant reviews we first joined review.json and business.json, based on the business id of the businesses, to get the location data (mainly city and state) about each business for which the review is written.



business_id	text	stars_x	useful	funny	cool	name	stars_y	is_open	latitude	longitude	city	state	attributes
_dhJZGzYirlWt-fMAg	This place epitomizes the rumored transformati...	5.0	0	0	0	Sienna Mercato	4.0	1	40.443950	-79.996437	Pittsburgh	PA	{'BusinessAcceptsCreditCards': 'True', 'Busine...
QtdAt-8RshaiBEHgw	Party of 3 ordered the fish tacos, pork belly ...	5.0	1	0	1	Served	4.5	1	36.010745	-115.064803	Henderson	NV	{'DriveThru': 'False', 'RestaurantsAttire': "...
zl2giWHcRN2plprxlg	Took my kids here to hang out on one fine 72 d...	2.0	5	14	4	Aliante Nature Discovery Park	3.5	1	36.287422	-115.176694	North Las Vegas	NV	{'GoodForKids': 'True', 'BikeParking': 'True',...
ıxRt2KDEnO5cgxDQ	This company tried deliver flowers to our home...	1.0	0	0	0	Arizona Florist	3.0	1	33.428259	-112.048848	Phoenix	AZ	{'RestaurantsDelivery': 'True', 'BusinessParki...
d2gxp2q3KPY19Oog	Yum!! Had the kale grits and would definitely	5.0	0	0	0	Roots Cafe	4.0	1	35.206507	-80.860258	Charlotte	NC	{'BusinessParking': '{'garage': False, 'street...

Figure 1: Joined dataframe from reviews.json and business.json

Further, we used the location for each business for each review written with the Yelp Fusion business search API, (which takes location as a required parameter), to search all the restaurants present at a location (by also filtering on the category of each business as a restaurant) and get their business ids. We filtered our review dataset using the obtained business ids to get the restaurant reviews. We gathered about 2.8 million restaurant reviews. We will work with the first 1 million reviews given the comptutational and memory constraints available to us.

On analysis of review lengths it was observed that majority of the reviews consisted of 100-150 words and stopwords constituted the major part of it. The distribution was as follows:

Property	Value
Mean	105
Standard Deviation	98
Maximum value	1007

Table 1: Distribution of review length in terms of words after pre-processing

After pre-processing, which involved word tokenization, removal of stopwords, and handling punctuations, the distribution was as follows:

Property	Value
Mean	56
Standard Deviation	52
Maximum value	905

Table 2: Distribution of review length in terms of words after pre-processing

From the analysis of cleaned dataset it can be observed that the review lengths is skewed in the 0-200 region. On further analysis is it can be seen that 97% of reviews have less that or equal to 200 words. Thus we set our cutoff of maximum words as 200 resulting in a dataset of size $(1000000, 200)$.

No. of words	No. of reviews
0-100	857,951
101-200	117,928
201-300	18,275
301-400	4,097
401-500	1,530
501-600	205

Table 3: Distribution of number of words in reviews

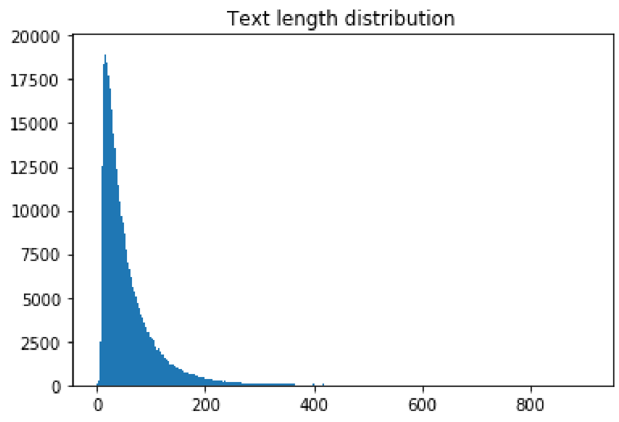


Figure 2: Distribution of review length in terms of words after pre-processing

Adjectives are extracted by utlizing the **nltk** library for parts of speech tagging. We visualised the characteristics of adjectives present in each review of our dataset.

Property	Value
Mean	9
Standard Deviation	8
Maximum value	123

Table 4: Distribution of adjectives count in reviews

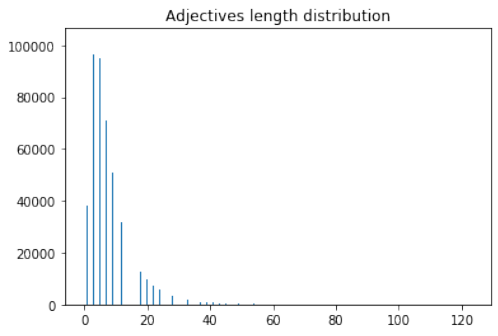
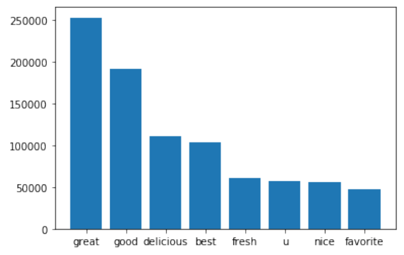


Figure 3: Distribution of adjectives count for a review in terms of words after pre-processing

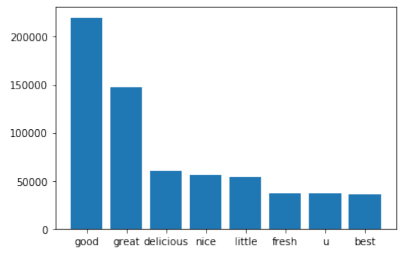
There were words whose tags where misclassified as adjectives. For instance 'pizza' was being tagged as adjective. In order to fix these anomalies and visualize the sentiment values of adjectives nltk's **sentiwordnet** library is utilised. Sentiwordnet[4] provides a positive and negative sentiment score for only true adjectives. For non adjective words no values are returned. Utilizing word embeddings (described in Methodoly section) our dataset is of size $(1000000, 20)$.

On average, a review consisted of 9 adjective words and overall distribution of adjectives was as follows:

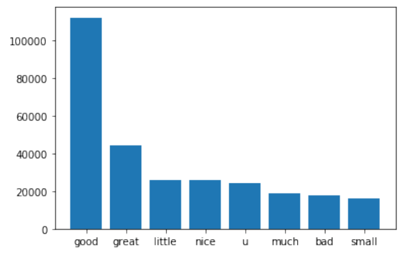
The top 8 adjectives based on the frequency of their occurrence for each of the 5 ratings class are given below:



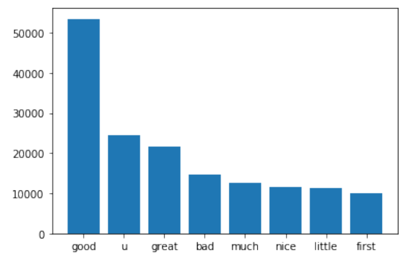
(a) Top 8 adjectives in 5 stars rating class



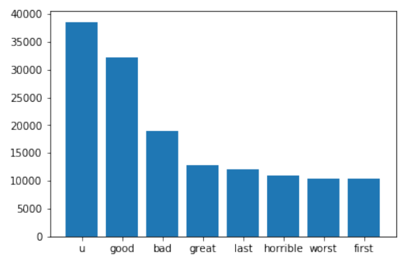
(b) Top 8 adjectives in 4 stars rating class



(c) Top 8 adjectives in 3 stars rating class



(d) Top 8 adjectives in 2 stars rating class



(e) Top 8 adjectives in 1 stars rating class

Figure 4: Top 8 adjectives in each of the ratings class

5 Problem Formulation

The task is to predict a user's star rating, on a scale of 1-5, for a business given the free-form text review written by a user for that business. Formally, this can be stated as:
Given a set $S = \{(t_1, r_1), (t_2, r_2), \ldots, (t_n, r_n)\}$ for a specific business, where t is the text review of the user and r is the corresponding rating, **we want our machine learned model to learn the best mapping from review t to numeric rating s**.

Since, the set S used to learn the mapping consists of examples of the reviews t as well as their corresponding numeric ratings s, **this problem falls into the category of supervised learning problem**. The ratings are out of 5 and hence any rating r_i corresponding to the i^{th} review t_i would belong to the set $R1 = \{1, 2, 3, 4, 5\}$. Also, the prediction corresponding to each review will be a single value from *R1*, which makes this problem a **5-class, single label classification problem**, where each value in *R1* corresponds to a class.

The second task is to predict the sentiment of the review. Star ratings for 4 and 5 are labelled with positive sentiment and star rating with labels 1, 2 and 3 are labelled with negative sentiment. The number of reviews with positive sentiment are roughly 70%.
$R2 = \{positive, negative\}$ where $positive = \{5, 4\}$ and $negative = \{1, 2, 3\}$

[4]https://www.nltk.org/$_{m}odules/nltk/corpus/reader/sentiwordnet.html$

In this task we just focus on the adjectives that appear in the review. Many methods mentioned in the literature review utilize positive and negative sentiments for aspects by using bi-grams, identifying aspects and using a window of words around it a descriptors, using sentiment analysis to identify words with high positive and negative scores. We propose a new approach where we use adjectives occurring in the review as descriptors. We reason that adjectives fit well with the definition of descriptors and capture the sentiment of the aspect for which they are used. We use this idea for rating prediction task as well to observe how well the ratings relate to the adjectives just for exploration as part of the scope of this project.

Yelp dataset is biased, the number of reviews that have numerical ratings of either 4 or 5 are significantly higher than those reviews with ratings less than 4. **About 69% reviews rate the corresponding restaurants highly**. Hence, the machine learning model needs to take into consideration this class imbalance to ensure that the accuracy for reviews belonging to the lower rating classes is not very less due to the imbalance.

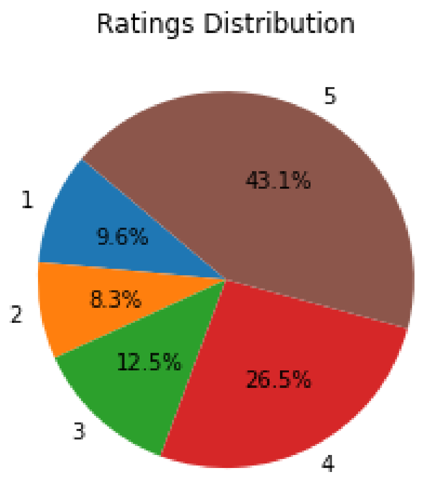


Figure 5: Distribution of ratings

6 Methodology

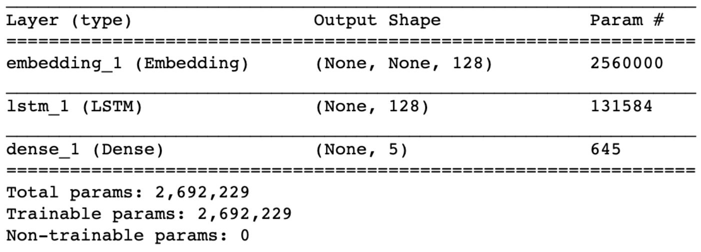
Loss function:

Since the task in hand is a classification problem for which our model output will be probability values between 0 and 1, we use the cross-entropy loss function for our deep learning models for rating prediction task. We have used binary cross-entropy for sentiment classification task. The choice of loss function was based on the problem formulation, multi class and binary classification.

Baseline Models

- **Naive Bayes:** Text classification is a supervised learning problem. Naive Bayes can be applied to multiclass classification problem. Naive Bayes assumes a very simple bag of words representation. This assumption is coherent with our task since the overall order of words is not vital. Further the conditional Independence assumption ensures fast training that is a useful property for a baseline model.

- **Vanilla LSTM:** Apart from the bag of words models like Naive Bayes we will also use deep learning models for our baseline. Deep learning models surpass the need to create handcrafted features like sentiment scores, positive negative words counts, aspect extraction etc. LSTMs are very popular for text classification task occurring in various different modifications. We will be using Vanilla LSTM.



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, None, 128)         2560000
_________________________________________________________________
lstm_1 (LSTM)                (None, 128)               131584
_________________________________________________________________
dense_1 (Dense)              (None, 5)                 645
=================================================================
Total params: 2,692,229
Trainable params: 2,692,229
Non-trainable params: 0
```

Figure 6: Our LSTM model summary

Implementation

Preprocessing

Preprocessing methods done by word tokenization, removing stopword, captilization, handling punctuations. The review words are tokenized using **keras.preprocessing.text** library. Top 20,000 words are used and number of words kept is 200 since more than 97% reviews contain 200 or less words. The labels are converted to one-hot encoding using **keras.utils.to_categorical** function.

Word Embedding

We used GloVe[5] to create embeddings of 20 features on the entire review datset consisting of all review words after preprocessing. We extract the embeddings for adjectives in each review comments. In order to represent the review by a single vector we average the embeddings of all the adjectives in the review. This results in a review of of dimension 20 resulting in a dataset of size $(1000000, 20)$. This method helps in reducing the dimension of the datset as well as retain the general meaning from all the adjectives.

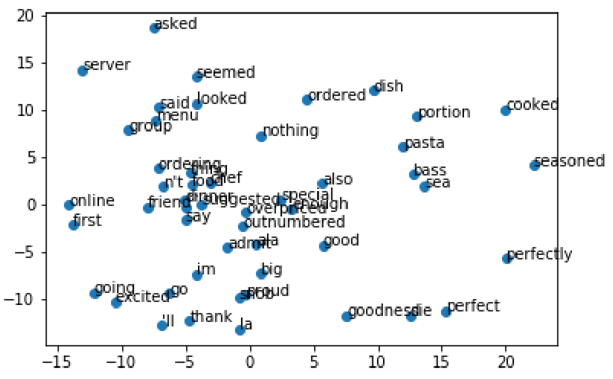


Figure 7: 2D projection of words with 100 features from GloVe embeddings

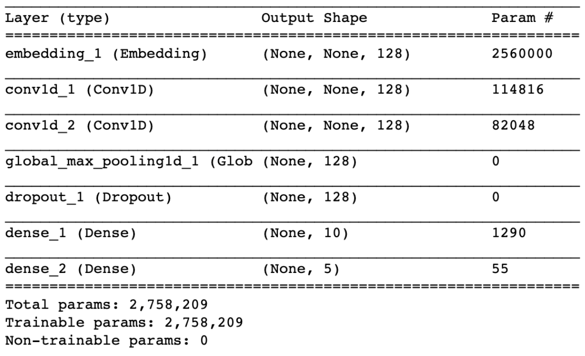
Preparing Dataset

One of the key challenges was to prepare data so that the computations don't exceed RAM size. Using the entire vocabulary or the number of maximum number of words for tokenizing eats more than 35GB of RAM. This makes training impossible. In order to make computations fast and efficient data analysis was used to set the right cutoffs and data type for our inputs. Current dataset size uses 10-12 GB RAM.

[5]https://nlp.stanford.edu/projects/glove/

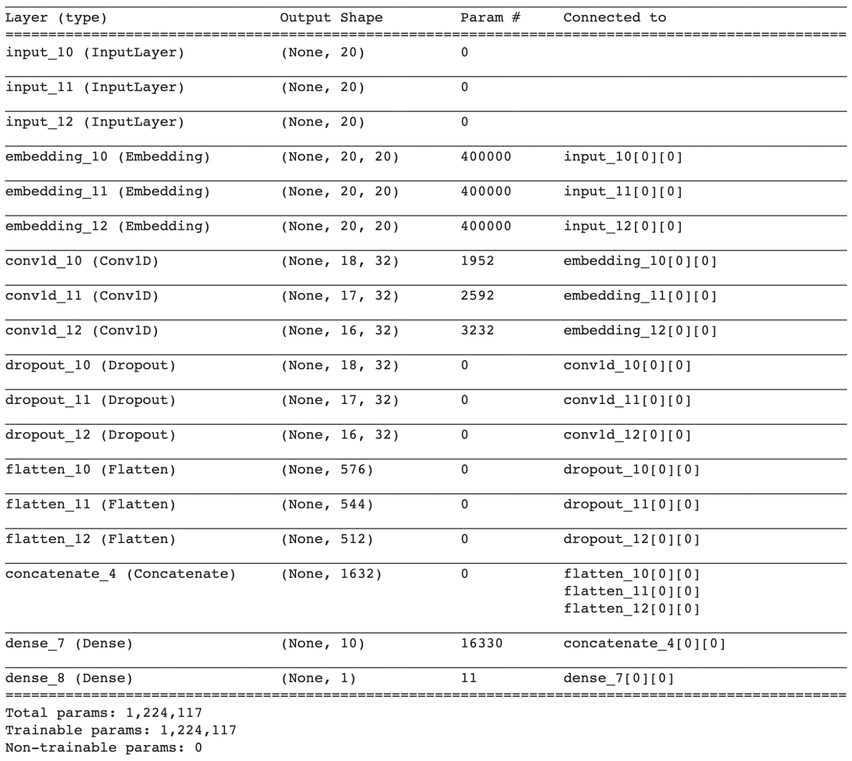
Convoluting Solutions with Convolution

CNNs use 1-d filters over the sentence to capture contextual information. We can use more than one filter to learn different features. We use the **keras** library. For the multiclass classification task an embedding layer is used to get a more feature space of tokenized words passed to the network. The embedding layer is followed by convolution layers to extract n-gram information from reviews, followed by dense layer and a softmax layer for classification. The network learns to identify and associate aspects and descriptors using n-grams. For binary classifcation task similar architecture however there is difference in the filter size. For dataset with 200 features filters of size 11,7 are used whereas for dataset with 20 features the filter size used are 3,4,5. On experimentation we found that single channel CNN architectures perform better. Thus we continued our experiments with single channel CNNs.



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, None, 128)         2560000
_________________________________________________________________
conv1d_1 (Conv1D)            (None, None, 128)         114816
_________________________________________________________________
conv1d_2 (Conv1D)            (None, None, 128)         82048
_________________________________________________________________
global_max_pooling1d_1 (Glob (None, 128)               0
_________________________________________________________________
dropout_1 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_1 (Dense)              (None, 10)                1290
_________________________________________________________________
dense_2 (Dense)              (None, 5)                 55
=================================================================
Total params: 2,758,209
Trainable params: 2,758,209
Non-trainable params: 0
```

Figure 8: Single channel CNN architecture



```
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_10 (InputLayer)           (None, 20)           0
__________________________________________________________________________________________________
input_11 (InputLayer)           (None, 20)           0
__________________________________________________________________________________________________
input_12 (InputLayer)           (None, 20)           0
__________________________________________________________________________________________________
embedding_10 (Embedding)        (None, 20, 20)       400000      input_10[0][0]
__________________________________________________________________________________________________
embedding_11 (Embedding)        (None, 20, 20)       400000      input_11[0][0]
__________________________________________________________________________________________________
embedding_12 (Embedding)        (None, 20, 20)       400000      input_12[0][0]
__________________________________________________________________________________________________
conv1d_10 (Conv1D)              (None, 18, 32)       1952        embedding_10[0][0]
__________________________________________________________________________________________________
conv1d_11 (Conv1D)              (None, 17, 32)       2592        embedding_11[0][0]
__________________________________________________________________________________________________
conv1d_12 (Conv1D)              (None, 16, 32)       3232        embedding_12[0][0]
__________________________________________________________________________________________________
dropout_10 (Dropout)            (None, 18, 32)       0           conv1d_10[0][0]
__________________________________________________________________________________________________
dropout_11 (Dropout)            (None, 17, 32)       0           conv1d_11[0][0]
__________________________________________________________________________________________________
dropout_12 (Dropout)            (None, 16, 32)       0           conv1d_12[0][0]
__________________________________________________________________________________________________
flatten_10 (Flatten)            (None, 576)          0           dropout_10[0][0]
__________________________________________________________________________________________________
flatten_11 (Flatten)            (None, 544)          0           dropout_11[0][0]
__________________________________________________________________________________________________
flatten_12 (Flatten)            (None, 512)          0           dropout_12[0][0]
__________________________________________________________________________________________________
concatenate_4 (Concatenate)     (None, 1632)         0           flatten_10[0][0]
                                                                 flatten_11[0][0]
                                                                 flatten_12[0][0]
__________________________________________________________________________________________________
dense_7 (Dense)                 (None, 10)           16330       concatenate_4[0][0]
__________________________________________________________________________________________________
dense_8 (Dense)                 (None, 1)            11          dense_7[0][0]
==================================================================================================
Total params: 1,224,117
Trainable params: 1,224,117
Non-trainable params: 0
```

Figure 9: Mutli channel CNN architecture

Data Imbalance

To handle class imbalance we oversample our training set to make the number of samples belonging to each class equal. Oversampling was preferred over undersampling because undersampling the data might lead to potential loss of useful training data. We use Synthetic Minority Oversampling Technique (SMOTE) for oversampling instead of random over-sampling with replacement since the latter might increase the chances of overfitting due to repetition of existing samples. We use the **imblearn**[6] library to apply the SMOTE technique. In order to handle class imbalance we have also used weighted loss for training some models mentioned before where the weight of each class is the inverse of it's frequency in the ratings. We used **sklearn.utils.class_weight** function for this task.

7 Experiments and Results

We evaluate our results based on the accuracy of classification for all the classes. The test set split is 20%. Therefore, there are 800,000 training examples and 200,000 test examples.

Hyperparameters:

The CNN model architecture is given below.
Batch Size: 32, 64
Classification: softmax
Optimizer: Adam, learning_rate=0.001, beta_1=0.9, beta_2=0.999, amsgrad=False
Loss function: categorical cross entropy (for multi-class classification), binary cross entropy (for sentiment classification)
Epochs: 20
Per Epoch Time: 55 seconds

The LSTM model uses the same hyperparameters.
Epochs: 5
Per Epoch Time: 2000 seconds

The LSTM model converges faster than CNN however the training time is 10 times higher.

7.1 Test accuracies for CNN compared to other models:

- **Multi-class classification:**

Model	Test Accuracy
Multinomial Naive Bayes	40.59%
Decision Tree	43.55%
LSTM	63.26%
CNN	66.02%

Table 5: Test accuracies of different models for multi-class classification problem

- **Sentiment classification:**

Model	Test Accuracy
LSTM	88.325%
CNN	82.58%

Table 6: Test accuracies of LSTM and CNN for binary classification problem

[6]https://imbalanced-learn.readthedocs.io/en/stable/generated/imblearn.over$_{s}ampling.SMOTE.html$

7.2 Comparison of test accuracies obtained using only adjectives to that using the entire review:

- **Sentiment classification:**

Model	Test Accuracy using entire review	Test accuracy using only adjectives
LSTM	88.325%	82.228%
CNN	82.58%	82.087%

Table 7: Test accuracies using entire review and only adjectives for binary classification problem

- **Multi-class classification:**

Model	Test Accuracy using entire review	Test accuracy using only adjectives
LSTM	63.26%	53.47%
CNN	66.02%	53.46%

Table 8: Test accuracies using entire review and only adjectives for multi-class classification problem

7.3 F1 scores for binary classification after applying SMOTE technique to handle class imbalance:

In this case, since our aim is to remove the effect of class imbalance of our dataset from the trained model and thereby from our prediction on the test set, we use the F1 score metric to compare the results since it ensures an optimal trade-off between precision and recall. The F1 score metric was obtained using the **sklearn.metrics.classfication_report** library.

Model	Sentiment class	F1 score without SMOTE	F1 score with SMOTE
LSTM	Positive	0.79	0.85
	Negative	0.92	0.87
CNN	Positive	0.63	0.68
	Negative	0.89	0.86

Table 9: F1 scores for positive and negative sentiment classes using entire review and only adjectives

8 Conclusion and Discussion

In this project we proposed using adjectives as descriptors. We use CNN for text classification and compare the results with our LSTM and Naive Bayes baselines. We perform these tests for both rating prediction and sentiment classification task. Secondly we propose a method utilizing adjectives as descriptors for aspects. We propose this approach primarily for sentiment classifcation. Based on the results from the previous section following conclusions can be drawn -

- Using entire review to predict star ratings gives better accuracy than using adjectives from the reviews. This can be attributed to the fact that the the language of the review as a whole is more deterministic for identifying the subtlety in the ratings.
- CNN models are performing as good as LSTM model for rating prediction for multiclass using entire review and using adjectives for binary classification task. Thus CNN due to their ability for extracting n-gram features at different positions of a sentence through convolution filters are able to work well on this problem.
- Using adjectives embeddings to learn the sentiment works at par in terms of accuracy using the entire review comment for CNNs. This is because the feature engineering process prior to training the model

enables the model to learn on the right information. The model is also simpler in form as it doesn't need to learn long term dependencies.

Some additional observations that can be made are as follows -

- Additional experiments (included in the above section) indicate that LSTMs outperform CNNs for binary classification task. LSTMs are learning the dependencies between the aspect and descriptors as well as learning the long term dependencies capturing the entire semantics of the sentence.
- The experiments suggest that altough using adjectives works as good as using the entire reviews for sentiment classification, for LSTMs the accuracy improves by a significant amount. For both unbalances and balances datasets the improvement in accuracy is about 6% which on a test set size of 200,000 reviews essentially means that 12,000 more being correctly classified than before.
- For problems with class imbalance if the model is not properly tuned deep learning models no matter how complex tend to predict the majority class.
- From the distribution of ratings for restaurants it can be observed that people rated restaurants favorably. Manual review revealed that most the negative reviews were due to poor service, unforseen problems that are not likely to occur in routine.
- The effects of data balancing are not very uniform. We thus believe that this technique is not very reliable for the scope project.

For this project we tried several methods and preprocessing steps to improve the accuracy on rating prediction task. We tried classical machine learning methods like Random Forest, Decision Tree Classifer. The input to these methods were Tf-idf weights of the reviews. However these methods performed poorly. We then explored deep learning methods namely CNNs and TCNs. TCN have a more robust architecture than CNNs and we expected it to work better than CNN however we got the opposite results. While using CNNs we tried different architectures, notably multi filter CNNs for text classification, CNNs with trainable embedding. For handling class imbalance we tried weighted loss functions, upsampling and downsampling data as well. Weighted loss function helped increase accuracy by minute amount.

Future work in this project will involve methods for better handling class imbalance. Another area of future work is based on our idea of using adjectives as descriptors. We can work on identification of aspect words. One more approach that could be tried is to build a sentiment classifier model from our dataset and use the sentiments of adjectives related to our aspects in non Deep Learning ML model like Random Forest.

References

[1] Nabiha Asghar. Yelp dataset challenge: Review rating prediction. *arXiv preprint arXiv:1605.05362*, 2016.

[2] Lei Shu Hu Xu, Bing Liu1 and Philip S. Yu. Double embeddings and cnn-based sequence labeling for aspect extraction. *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, 2018.

[3] Jiangtao Qiu and Yinghong Li. Leveraging sentiment analysis to predict ratings of reviews. *Pacific Asia Conference on Information Systems*, 2016.

[4] Hao Yang Sungyong Seo, Jing Huang and Yan Liu. Interpretable convolutional neural networks with dual local and global atention for review rating prediction. *Proceedings of the Eleventh ACM Conference on Recommender Systems*, 2017.