Sentence Classification on Yelp Reviews

Zhezhong Jiang
Department of Computer Science
Boston University
Boston, MA 02115
`zzjiang@bu.edu`

Rui Pang
Department of Computer Science
Boston University
Boston, MA 02115
`ruipang@bu.edu`

Yiyan Zhou
Department of Computer Science
Boston University
Boston, MA 02115
`zhou482@bu.edu`

May 9, 2019

Abstract

Many e-commerce sites allow users to input detailed reviews as well as specific star ratings. We seek to find if it's possible to correlate the detail of text reviews with the numerical rating through multiple deep learning techniques. For a local business that does not have reviews on Yelp, it is possible to scrape the relevant reviews found on web and predict corresponding star rating using this model. We use convolutional neural networks (CNN) to perform sentence classification on Yelp reviews. Two label classification(positive/negative) and five label classification(1 to 5) are implemented to identify positive/negative reviews and the corresponding stars. The result is promising on 2-label classification task and our models are performing reasonably good on 5-label classification as well. For 2-label classification, we successfully develop a model that has 94.49% accuracy rate. And for five-label classification, among 7 models we have tested, the highest accuracy rate we have achieved is 60.41%.

1 Introduction

Neural network has been the hottest research area in the field of machine learning recently. As the name suggests, it is a brain-inspired system which is intended to replicate the way that we humans learn. When we human process image or other types of recognition, our brain neurons get stimulated layer by layer, for example, one layer of brain cells recognize the shape and the other layer of brain cells recognize the color, and after getting all the information from each layer, our brain recognize the item we see. Similar to this process, neural network algorithm consists of input and output layers, as well as (in most cases) a hidden layer consisting of units that transform the input into something that the output layer can use. Comparing to other classification algorithms, neural networks require much lower pre-processing of input and have been designed to solve problems in different areas such as but not limited to image classification, object localization, natural language processing etc.

Unlike traditional neuron networks, which are often computational expensive and requires a lot of storage. Convolutional neural network has three important ideas that can help improve a machine learning system: sparse interactions, parameter sharing and equivariant representations[6].

By making the kernel smaller than the input, convolutional networks typically have sparse interactions(also referred to as sparse connectivity or sparse weights).Fewer parameters are needed to be stored. Convolutional neural network also uses the same parameter for more than one function in a model(parameter sharing).Each member of the kernel is used at every position of the input. Therefore,rather than learning a different set of parameters for each location, the model learns only one set.

Moreover,because of the parameter sharing characteristic, the layers of convolutional neural network have a property called equivariance to translation. Specifically, a function $f(x)$ is equivariant to a function g if $f(g(x)) = g(f(x))$. For example, let I and g be two functions in our network, such that $I' = g(I)$ is the image function with $I'(x, y) = I(x - 1, y)$. Equivariance property gives the same result if we apply transformation to I, then apply convolution, as if we apply convolution to I', then apply the transformation to the output.

Those three unique properties reduce the memory requirements of the CNN model and improves its statistical efficiency(usually quite large improvement),which become the advantages of CNN.It is a class of deep neural networks widely used in computer vision and proven to be quite effective when tackling Natural Language Processing (NLP) problems [2].

In the present work, we apply the Convolutional Neural Networks algorithm to sentence classification of yelp reviews. In Yelp, customers can share their reviews about products and services they receive from local/online shop in two formats: detailed sentence reviews and star ratings (1 to 5 stars). We try to explore the possibility to correlate the detail of text reviews with numerical ratings as well as the overall attitude of customers. We trained our neural network models on Yelp data set using GloVe word vectors[3] which is trained on 100 billion words on Google News. The goal of our work is to replicate the CNN models proposed by Yoon Kim[2] as well as exploring possible improvements on models in terms of prediction accuracy.

2 Data Overview and Pre-processing

The original data set has over 6,685,900 reviews on different categories of business. Our primary focus here is on the restaurant subset which contains roughly 1,000,000 user reviews. Before feeding the data into neural network, each review is processed in the following ways : remove the punctuation, change all characters to lower case, and with the help of Python nltk library , all the stop words such as "is" "are" "the" is removed since they don't contain any contextual information. The numbers are also removed for two reasons: first, they only convey a sense of quantitative information i.e. the phrase "I spent 15 dollars on a sandwich totally not worth it" , with the number removed, the sentiment is not affected at all; Second, the existence of numbers will increase the size of the dictionary when we build the vocabulary used in word embedding later on, which makes the data modeling process harder.
For the task of two label classification, we removed the neutral reviews (3-star) and categorize reviews with four and five star as positive and reviews with one and two stars as negative. A total of 512913 entries with 391928 positive reviews and 120985 negative reviews are obtained after the pre-processing steps. As in standard approach, we randomly removed 65% positive reviews to make a balanced data set. The final data set then has a total entry of 258160 entries with 137175 positive reviews and 120985 negative reviews. Thus the baseline accuracy for our project is roughly 53%.
For the five label task, the same pre-processing step is performed but we keep reviews with original star rating. The five label data set, after balancing number of reviews for each rating, has a total of 250,000 reviews.
For both two label and five label task, a hold-out test set is used to evaluate the performance in real time. The hold-out test set is obtained by randomly drawing 20% from the training set. The test sets are balanced as well.

To feed the text data into neural network, it needs to be converted into numerical data. For all the experiments, we have adopted the glove word embedding algorithm to find the vector representation for each word. Each word then is converted to a 100 dimensional vector, and we have restricted the length of each review to be 130 either by cutting down the long reviews or padding zero vector for short reviews. At the end ,each review is a (130,100) matrix and ready to be fed in to the network.

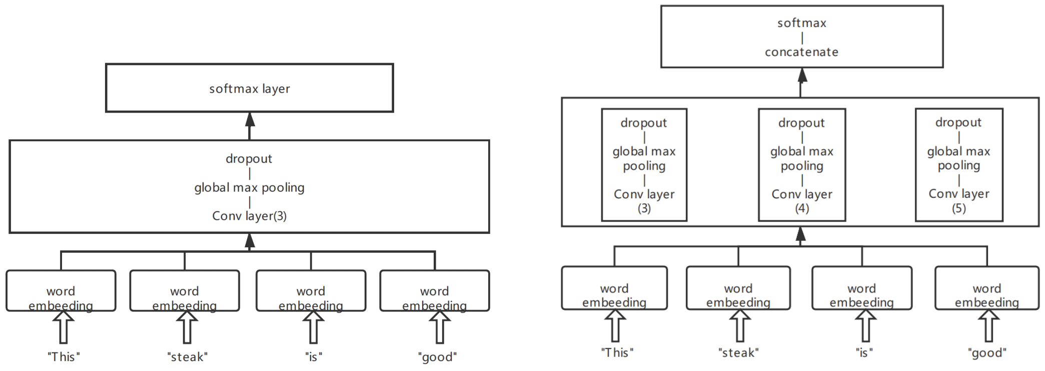
3 Model Specification

The primary model we investigated is as follows , 3 different sizes(3,4,5) of convolutional filters are used, each filter size has 128 feature maps, for example the filter with size 3 will produce a feature map of (128,128), the first dimension is the number of convolution operations and the second is the number of feature maps used for that filter size, then a global max pooling is conducted to extract the largest entry in for each feature map , resulting in a vector of length 128.The feature maps obtained from different filter sizes are then concatenated and feed into a fully connected neuron network.
For the purpose of comparison we have designed a baseline model, to examine if the network structure proposed by the author is indeed more effective that single layer convolutional network.The baseline model and original model are shown in Figure 1.

4 Model Testing

4.1 Motivation

Rather than applying some of the state of art techniques directly into our model, we decide to first examine those techniques in a simulation test.The simulation takes in randomly generated data and the overall network structure is very similar to the one we used in real data modeling but with much less parameters. Thus it is possible to keep all other parameters fixed and see if introducing an additional technique indeed helps the learning process.



(a) baseline model (b) original yoon kim model

Figure 1: baseline model and yoon kim model

4.2 Model with Drop out

First, we fit our data using CNN proposed by Yoon Kim [2]. Besides convolutional layer, Yoon Kim [2] adds an additional layer of dropout[4] before the final dense layer. The idea of dropout can be viewed as a way of regularization which it randomly shuts down a proportion p of the neurons during training. Intuitively it forces some of the week neurons (i.e. the neurons that are not being updated too much during back propagation) to learn. Specifically given a input layer $\mathbf{z} = [\hat{c_1}, \hat{c_2}, \cdots, \hat{c_m}]$ where m is the number of concatenated feature maps obtained from different filter sizes, instead of using :

$$y = \mathbf{w} \cdot \mathbf{z} + b$$

in the forward propagation , dropout uses :

$$y = \mathbf{w} \cdot (\mathbf{z} \circ \mathbf{r}) + b$$

where $\circ$ is the element-wise multiplication operator and $\mathbf{r} \in \mathbb{R}^m$ is a vector of Bernoulli random variables with probability p of being 1. During back propagation a weight w_i is updated only if the corresponding $(\mathbf{z} \circ \mathbf{r})_i$ is non zero. Moreover,during testing time, all the weights are being used to make prediction and are re-scaled as $W^l = pW^l$. Although Kim does not specifically mention why such re-scaling is used, our guess is that if the weights have high probability of being dropped out (a low p), it simply does not learn much useful information because it is only being updated a very limited number of times.Thus the weights that are being updated more often should have more influences than the one that doesn't.

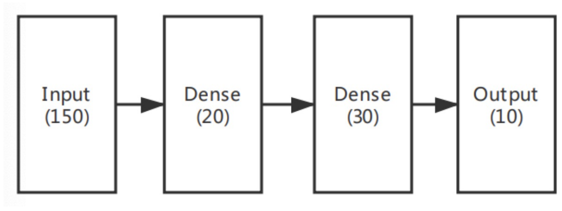
Simulation

To investigate the effect of dropout, 300 examples are generated i.e $\mathcal{T} = \{x_i, y_i\}_{i=1}^{300}$, each x_i has 150 features , that is $x_i \in \mathcal{R}^{150}$, and each of the feature value comes from a standard normal distribution. Each response variable y_i can take on one of 10 different integer values ranging from 1 to 10, indicating 10 different classes. Each response variable y_i is a chosen based on uniform distribution from 1 to 10. A hold-out validation set is used for tracking the accuracy of the model on unseen data during training. The network is a simple two-layer neural network with 20 neurons in the first hidden layer and 30 neurons in second hidden layer. The activation function is chosen to be sigmoid. The model structure is shown in Figure 2 :

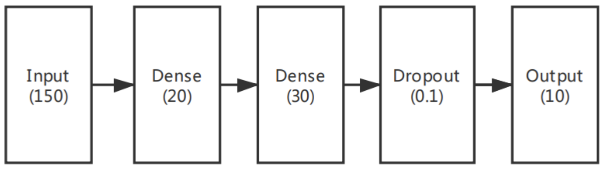
We have employed an extreme value $p = 0.1$- on average 90% of the neurons will be dropped during training to see the effects of this method. As it can be observed from Figure 3, even though the model without dropout achieves much higher accuracy during training (which is reasonable since only part of the neurons are used in the dropped out model during training), the model with dropout at test time (when all features are used and are scaled appropriately) is more robust and can lead to higher testing accuracy.

4.3 Model With Batch Normalization

Besides the vanishing gradient problem described previously, another potential difficulty of training neural network is that the distribution of outputs in the middle layers change as the training proceeded. This phenomenon is referred to as

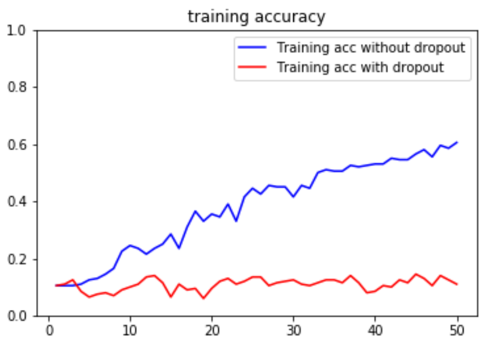


(a) baseline simulation model

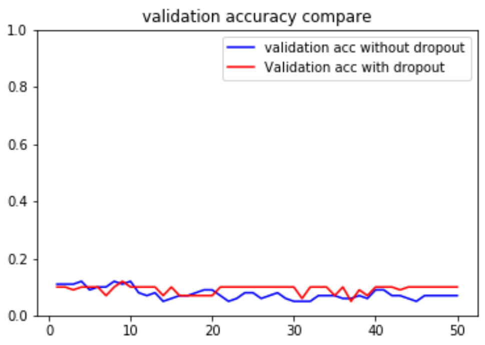


(b) simulation with dropout

Figure 2: Simulation models for dropout



(a) training accuracy comparison



(b) validation accuracy comparison

Figure 3: Dropout training accuracy and validation accuracy comparison

covariate shift[5]. This shift in distribution is not a desirable feature. During feed forward process , the activations of previous layer is the input of the next layer. Since the network changes its parameters during training, the activation of each layer will also change, which means the input distribution changes with each step of weight update. Therefore the intermediate layer has to continuously adapt to changing inputs. Batch Normalization is the way to reduce the covariance shift by normalizing the activations of each layer and therefore ensures a more stable input distribution and thus accelerate the learning process. Batch normalization introduces two parameters that can be learned by the network to convert the mean and variance to any value the network desires.
The algorithm is taken from the original batch norm paper[5], let $\mathcal{B} = x_1 ... m$, the mini-batch data of size m, and γ, β be the two addition parameters to be learned:

$$\mu_{\mathcal{B}} = \frac{1}{m}\sum_{i=1}^{m} x_i \quad \text{mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 = \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu_{\mathcal{B}})^2 \quad \text{mini-batch variance}$$

$$\hat{x}_i = \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \quad \text{normalize}$$

$$y_i = \gamma \hat{x}_i + \beta \quad \text{scale and shift}$$

Simulation
The simulation data and model used is exactly the same as the one in dropout section, except now a batch norm layer is added right after each dense layer. The resulting comparison for training accuracy is shown in Figure 4:

As Figure 4 shows, the model with batch norm learns faster than the one without(higher training accuracy), showing normalizing inputs to hidden layers can indeed accelerate the learning process.

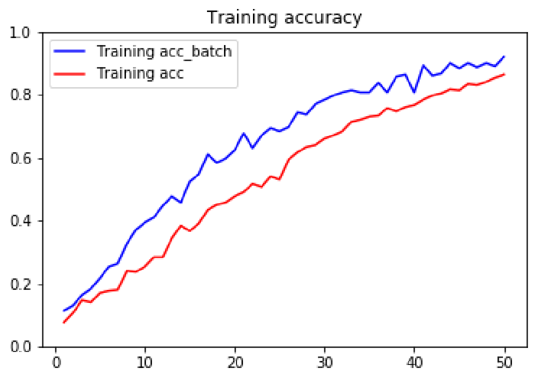
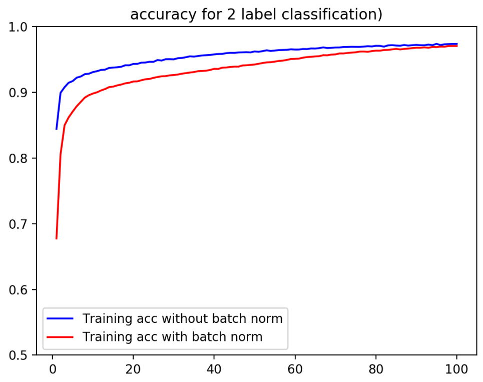


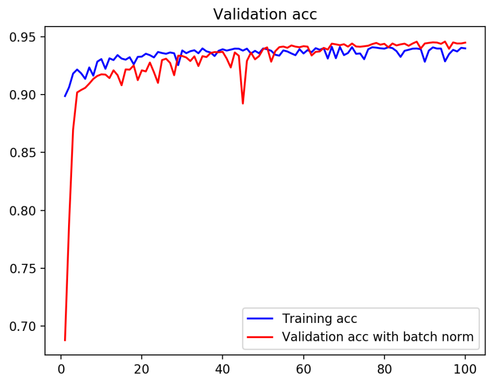
Figure 4

4.4 Effect of Batch Norm and Dropout in Yelp Sentiment Classification Challenge

Here we present the effect of batch normalization in action. We add a batch normalization layer right after the convolution layer and the dense layer but before the activation layer in our original network structured presented in Kim[2].Figure 5 is the comparison of accuracy curves and validation accuracy curve over time on the 2 label classification task:



(a) accuracy



(b) validation accuracy

Figure 5: Training accuracy comparison for batch norm

As Figure 5 shows, model without batch norm (blue curve) appears to learn faster than the model with batch normalization (red curve). This may first seem contrary to the the results presented in the original paper and simulation stage. We deduce that this is due to the dropout layer used before the last dense fully connected layer (reason explained in the dropout section). Another potential reason is the number of parameters for original yoon kim model is significantly larger(around 340,000) than the toy network we build in the simulation stage. With number of layers increase , the number of additional parameters of batch normalization also increases. However , examining the validation accuracy, we can clearly see that it converges faster than original model ,with much less wiggle after 50 epochs and leads to better performance. After training with 100 epochs , the model with batch normalization is able to achieve a 94% on the hold-out validation set , which is 1.5 % more than the original model.

4.5 Challenge : 5 label classification

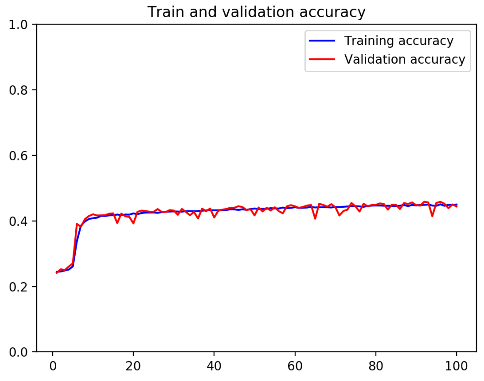
We also employ our model in 5-label classification by classifying the detailed rating for each review ranging from 1 star(the worst) to 5 star (the best). For the network, in order to adapt to multi-label classification ,the last layer are changed to a softmax layer ,specifically output neuron j is computed the following way :

$$softmax(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}$$

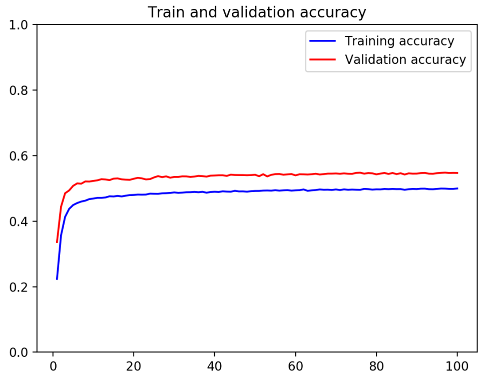
Softmax layer takes in a vector size K and normalize it to output a probability distribution. The loss function is still cross entropy loss but with multiple labels, slight modification has to be made :

$$-\sum_{c=1}^{M} y_{o,c} \log(P_{o,c})$$

where M is the number of classes (5 in this case), log is the natural log as usual, y is a binary indicator takes on value 0 if class label c is the correct prediction on observation o , and $P_{o,c}$ is the predicted probability that observation o is of class c.
The baseline model achieves only 44 % accuracy, and the original yoon kim model achieves 54 %. Plots of training and validation accuracy for both model is shown in Figure 6:



(a) baseline model



(b) original yoon kim

Figure 6

As can be observed both the base line model and original yoon kim model stops improving after the first 10 epochs or so due to its over simplified network structure.Again the effect of dropout can be clearly observed from the original yoon kim model by noticing that the validation accuracy is higher than the training accuracy. The network structure are enhanced in the hope to capture more structural information within sentences. Thus we increased the number of filters to 128 , and instead of filters with size 3,4,5 , we include one more filter size of 2 to capture contextual information between neighbouring words such as "not bad". Batch normalization is added right after convolutional layer and dense layer , and dropout layer is added after the concatenation of feature maps and after the dense layers, for this particular experiment we have chosen the parameter $p = 1 - 0.3 * N$, where $N \sim U(0, 1)$. The model achieves 59% accuracy. The model diagram is given in Figure 7 and accuracy and loss plots are given in Figure 8.

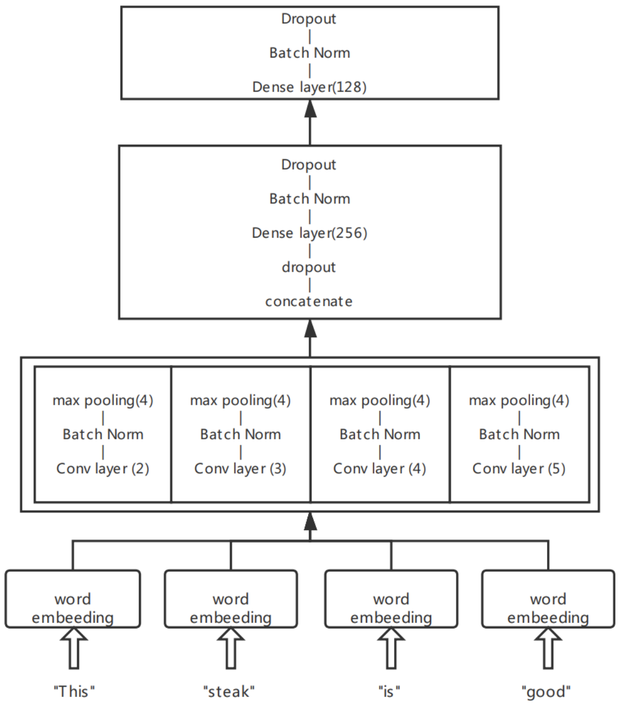
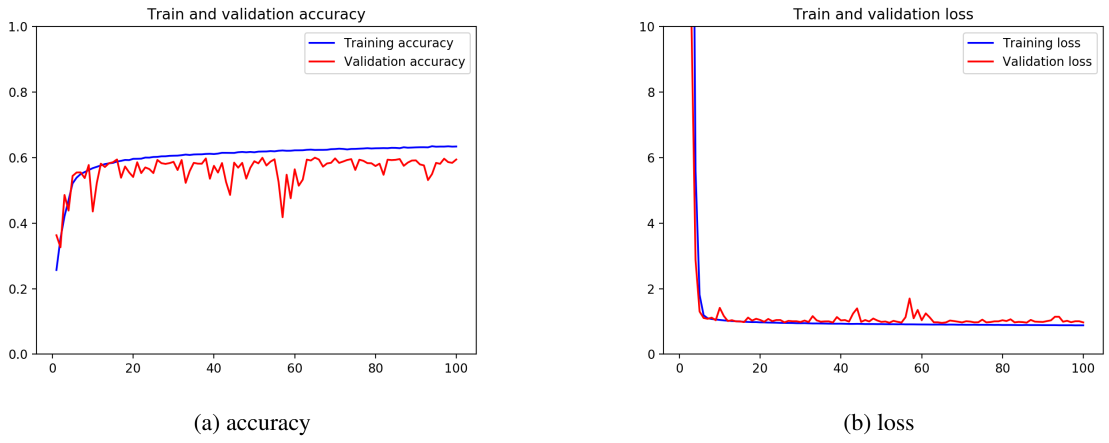


Figure 7



(a) accuracy

(b) loss

Figure 8

5 Results

Here we show the results of our experiments, apart from CNN models, we also experimented several recurrent neural network models to make comparisons. Recurrent neural network models are another kind of network structure designed specifically for dealing with sequential data. It utilizes the idea of "gates" to learn which part of information should be forgot and which part should be remembered to pass on. The most popular RNN structure used nowadays includes LSTM[8](named lstm in Table 1) and its variant Bidirectional LSTM[9](named bi-lstm in Table 1) As can be observed from the table, our improved model(named yoon-kim-2 in Table 1) based on the yoon-kim structure achieves the best performance is two-label classification task whereas the hybrid mode LSTM-CNN beats our model with a fraction of accuracy score in the five label task. The red dot ● indicates that the model uses drop out and the green dot ● indicates the model uses batch normalization

model \ task	2-label	5-label	
baseline	89%	44%	
yoon-kim	92.4 %	54%	🔴
yoon-kim-2	**94.49%**	59.56%	🔴🟢
lstm	92.02%	59.70%	🔴🟢
bi-lstm	92.96%	60.1%	🔴🟢
cnn-lstm	90.74%	51.28%	🔴🟢
lstm-cnn	91.6%	**60.41%**	🔴🟢

Table 1

6 Conclusion

The performance is quite good considering the baseline accuracy is only 20% in the 5 label classification task. To gain more insights , the confusion matrix for the final model (yoon-kim-2) is given below :

prediction \ actual	1	2	3	4	5
1	7732	1712	326	118	112
2	2644	4424	2325	481	126
3	475	1727	4676	2814	308
4	90	163	1314	6115	2318
5	80	49	166	2940	6765

Table 2

As can be observed , the model is doing the right thing on the "big picture".Even in the much more complicated setting of 5 labels , it has learned , though not in the perfect sense , to identify a positive review and negative review. For example for reviews with 1 star(the first row), 7732 test examples are classified correctly , and 1712 test samples are classified as a 2-star review, so a total of 9444 test samples are predicted as negative review. The same can be observed for reviews with 5-star and 4-star. Not surprisingly the machine is worst at predicting neutral reviews (with less than 50% precision score). Though interestingly the distribution of 3-star prediction seems to follow a a shape pf Gaussian distribution. Our conclusion is that the performance of this model is restricted due to the confusion between reviews that fall in to the same category but with different level of sentiment. The root cause of this confusion though , has a lot to do with different judge criteria of people who wrote the reviews, for the same food and service , it's very possible that two persons come up with completely and opposite reviews. To eliminate such problem , one possible fix is to spend more time in the pre-process step to manually select reviews for each category based on some fixed criteria.While RNN performs the best in the more general setting of 5-label task, it should be mentioned that it takes considerably amount of time to train, the hybrid model takes about 9 hours on a Tesla P100 GPU whereas our model(yoon-kim-2) only takes 25-30 minutes to train. Thus it is our opinion that CNN points a promising direction in dealing with natural language tasks such as sentiment analysis.All code including data pre processing, simulation and machine learning models is available at here

References

[1] LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11):2278–2324, November 1998a.

[2] Kim, Yoon. Convolutional Neural Networks for Sentence Classification. arXiv preprint arXiv:1408.5882.

[3] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation.

[4] Srivastava, Nitish, Hinton, Geoffrey, Krizhevsky, Alex, Sutskever, Ilya, and Salakhutdinov, Ruslan. Dropout: A simple way to prevent neural networks from overfitting. J. Mach. Learn. Res., 15(1):1929–1958, January 2014.

[5] S. Ioffe and C. Szegedy. Dropout: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In ICML, 2015.

[6] Goodfellow.I,Bengio.Y and Courville.A. Deep Learning. MIT Press, 2016

[7] Kafunah,Jefkine. Backpropagation In Convolutional Neural Networks. DeepGrid, 2016

[8] Sepp Hochreiter and Jurgen Schmidhuber. Long short-term memory. Neural computation 9(8):1735–1780, 1997

[9] Schuster, M. and Paliwal, K. K. Bidirectional recurrent neural networks. Signal Processing, IEEE Transactions on, 45(11), 2673–2681, 1997