Multiclass Sentiment Prediction using Yelp Business Reviews

April Yu
Department of Computer Science
Stanford University
Stanford, CA 94305
aprilyu@stanford.edu

Daryl Chang
Department of Computer Science
Stanford University
Stanford, CA 94305
dlchang@stanford.edu

Abstract

Many e-commerce and related sites allow text reviews, which provide much more detail than the quantitative metric of star ratings. However, star ratings provide insight on a establishment or product very quickly. Seeding new e-commerce sites with this quick quantitative information from pre-existing text reviews scattered throughout the Internet would greatly improve the information provided about an item or establishment. Is it possible to translate the detail of text reviews into the quick usefulness of a numerical rating? We hope to explore this through multiple deep learning techniques.

1 Introduction

Sites like Yelp and Amazon allow users to formalize their thoughts and opinions of businesses and products in the form of text reviews and numerical ratings. These text reviews provide the highest granularity of detail that restaurants are able to receive directly from their customers. However, when expanding to a new area, Yelp often lacks reviews for local businesses, a problem that could be ameliorated by seeding the business page with relevant reviews scraped from the web. If a scraped review does not have a numerical rating, one needs to be generated from the review text itself.

Using the Yelp Challenge Academic Dataset and pre-trained GloVe word vectors, we run various neural networks on text reviews to predict the star rating associated with that particular review. We set a preliminary baseline for our work by using softmax regression on the mean of the word vectors for a review. With this baseline metric in mind, we implement a recurrent neural network with LSTM and a convolutional neural network. In both approaches, we use pre-trained word vectors instead of randomly initializing and then training word vectors.

2 Background/Related Work

One simple model for sentiment analysis is the bag-of-words model, which vectorizes a document as the occurrence counts for each word, sometimes using a reweighting scheme such as tf-idf to place more importance on unique words. However, the bag-of-words model does not account for more complex interactions, such as word order and long-range dependencies. Socher gives the example of "white blood cells destroying an infection" and "an infection destroying white blood cells" having the same bag-of-words representation, despite the former being positive and the latter being negative (Socher et al., 2011).

Vector space models (VSMs) learned from co-occurrence counts, as in GloVe and word2vec, have achieved higher performance on sentiment analysis. These models represent words in a high dimensional space, encoding both syntactic and semantic information, with a bias towards the latter as

the window size grows larger. These vectors can be used as inputs for different tasks; in sentiment analysis, for example, we can use softmax regression on the word vectors to predict the polarity of the document. We can also use the word vectors as inputs for a variety of neural networks.

2.1 Long Short-Term Memory

Hochreiter illuminates the problem that with conventional recurrent neural learning, the back-propagated error tends to either "blow up" or "vanish" (1997). He proposes long short-term memory, which is designed to overcome these error back-flow problems by learning to bridge time intervals in excess of 1000 steps even in the case of noisy, incompressible input sequences. With the case of Yelp Reviews, the sentiment of a sentence can be largely dependent upon words that are far away from each other. A traditional recurrent Neural Network (RNN) would be ineffective at learning these long-range dependencies.

2.2 Convolutional Neural Networks

Both Kim and Kalchbrenner et al. have successfully used convolutional neural networks for a variety of sentiment analysis tasks. Kalchbrenner et al. (2014) employs a CNN with the following layers:

- Convolutional layers using wide convolution in order to ensure that all weights of the filter reach words at the margin of each sentence.
- k-max pooling, in which the k largest values in a sequence are returned in the order they appear in the sequence.
- Folding, in which every x rows are summed up component wise to reduce feature dimensionality.
- Dropout, in which a binary mask of Bernoulli random variables is applied to prevent co-adaptation of the hidden units.

Kalchbrenner combines multiple sets of these layers to create neural networks of varying depth. The neural network achieves 48.5% accuracy on 5-way sentiment classification on the movie review dataset from the Stanford Sentiment Treebank, and 86.8% on binary sentiment classification (Kalchbrenner et al., 2014).

However, Kim achieves equivalent results with a simpler convolutional neural network that uses max-over-time pooling and a single convolutional layer (Kim, 2014). He also demonstrates the use of multichannel word vectors, in which both static and non-static pretrained word vectors are used, with the intermediate feature map being the sum of the two resulting feature maps. With the multichannel architecture, Kim achieves 47.4% accuracy on the 5-way movie review sentiment classification task and 88.1% on the binary classification task (Kim, 2014).

3 Approach

We experiment with the two aforementioned models: recurrent neural networks with long short-term memory, and convolutional neural networks.

3.1 Recurrent Neural Network with Long Short-Term Memory

We implement a variation of the LSTM proposed by Hochreiter et al, which introduces a new structure called a memory cell. To allow for constant error flow, a multiplicative input gate unit protects the memory contents from perturbation by irrelevant inputs. Similarly, a multiplicative output gate unit protects other units from perturbation by currently irrelevant memory contents. This creates a memory cell, which is built around a central linear unit with a fixed self-connection. We implement a variant of the Hochreiter LSTM with the following equations:

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \tag{1}$$

$$\widetilde{C}_t = tanh(W_c x_t + U_c h_{t-1} + b_c) \tag{2}$$

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \tag{3}$$

$$C_t = i_t * \widetilde{C}_t + f_t * C_{t-1} \tag{4}$$

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_1) \tag{5}$$

$$h_t = o_t * tanh(C_t) \tag{6}$$

where

- x_t is the input to the memory cell layer at time t
- $W_i, W_f, W_c, W_o, U_i, U_f, U_c, U_o$ are weight matrices
- $b_i, b_f, b_c and b_o$ are bias vectors
- i_t is the value for the input gate
- $\widetilde{C}_t$ is the candidate value for the states of the memory cells at time t
- f_t is the activation of the memory cells' forget gates at time t
- C_t is the memory cells' new state at time t
- o_t is the value for the output gates
- h_t is the output

The gates control the error flow to and from the memory cell c_j by learning when to capture and release the errors and appropriately scaling them.

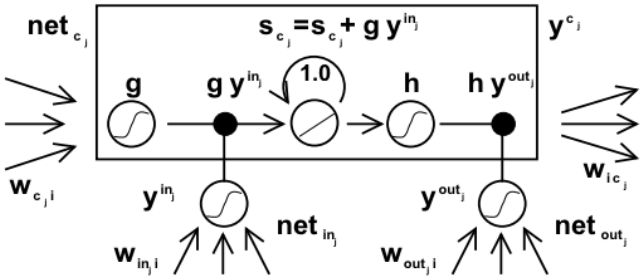


Figure 1: Architecture of memory cell c_j (the box) and its gate units in_j, out_j

We use a network with one input layer, one hidden layer and one output layer. The hidden layer contains the memory cells and the corresponding gate units.

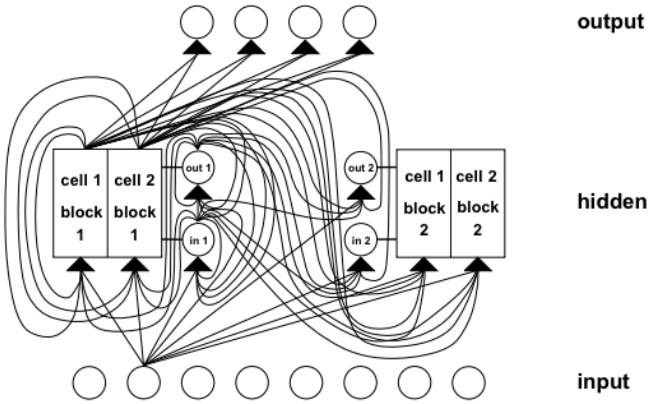


Figure 2: Example of a net with 8 input units, 4 output units and 2 memory cell blocks of size 2

We implement a recurrent neural net similar to one depicted in the above diagram with different dimensionality for the input, hidden and output layers. All results are a product of 128 hidden units, unless otherwise specified. Our model is composed of a single LSTM layer followed by an average pooling and a logistic regression layer. From the input sequence $x_0, x_1, ..., x_n$, the memory cells in the LSTM layer produces a representation sequence $h_0, h_1, ..., h_n$. This sequence is averaged over all timesteps, thus resulting in representation h. This representation is fed to a logistic regression layer, whose output is the class label associated with the input sequence.

All diagrams are borrowed from Hochreiter et al.

3.2 Convolutional Neural Network

We implement a version of Kim's convolutional neural network, with our key contribution being the use of chunking instead of zero-padding for the feature matrices.

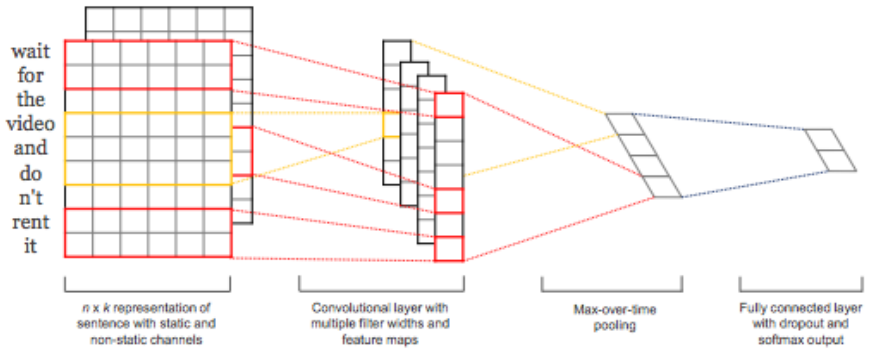


Figure 3: Convolutional neural network. Adapted from Kim.

As shown in Figure 3, we represent each review by concatenating the word vectors of the words in the review to form a feature matrix. The feature matrix is then convolved using multiple feature filters to form a feature map. Max-over-time pooling is then applied to the feature maps to form a one-dimensional vector, on which dropout is then applied to prevent feature co-adaptation. Then a hidden fully-connected layer and non-linearity are applied to the vector to produce the resulting output.

Our implementation differs from Kim's in three key aspects:

- We use GloVe vectors instead of word2vec vectors.
- Since each input sentence is of a different length, Kim pads each matrix with zeros, standardizing the dimensionality of the input matrices (Kim, 2014). While we implement zero-padding, we find that it does not translate well to our dataset, both in terms of computational efficiency and accuracy. Consequently, we instead segment each review into chunks of a fixed length and use the CNN to predict the sentiment of each one, as shown in Figure 4. The overall sentiment is then calculated by averaging the chunk sentiments.
- Instead of using softmax and cross-entropy as the non-linearity and cost function, respectively, we change the hidden layer to output just one number, effectively changing the model from a classifier to a regressor. Consequently, we use mean squared error as our cost function. We do so because categorical classification does not include any gradation of errors; for example, predicting a '2' on a review with 4 stars is just as bad as predicting a '3'. Using regression allows us to account for the distance between the predicted and actual labels. To calculate label accuracy, we round the regression prediction to the nearest integer.

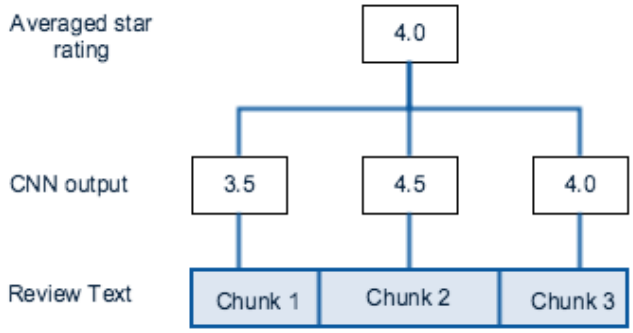


Figure 4: An illustration of the chunking process applied to each review.

4 Experiments

We use the Yelp Academic dataset, which consists of 1.6 million reviews by 366,000 users over 61,000 businesses. We segment our dataset into 2 sections, split 85/15: development and testing. We run cross-validation of our models on the development set throughout development and test on the test set after parameter tuning and model refinement.

As shown in the figure, the Yelp dataset is highly skewed toward positive ratings. Initially, we adjusted for the class imbalance by sampling equally from each class for the training examples; however, we then decided to keep the class distribution in order to better reflect the real-world context in which such a model would be used.

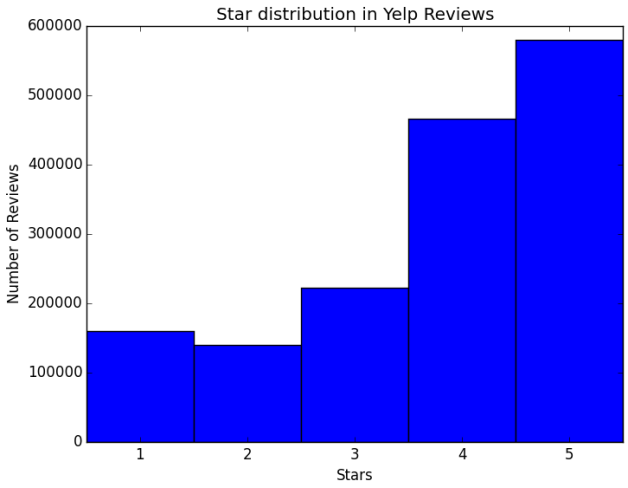


Figure 5: The Yelp dataset is highly skewed towards 4- and 5-star ratings.

Our baseline metric for the project is softmax regression on the averaged word vectors for a review, which achieves 45% accuracy on the test set.

With an LSTM implemented as stated above and a hidden layer dimensionality of 128, we achieve 51% accuracy on the dev set. We then begin looking into the effect the hidden layer dimensionality has on the accuracy achieved by the model:

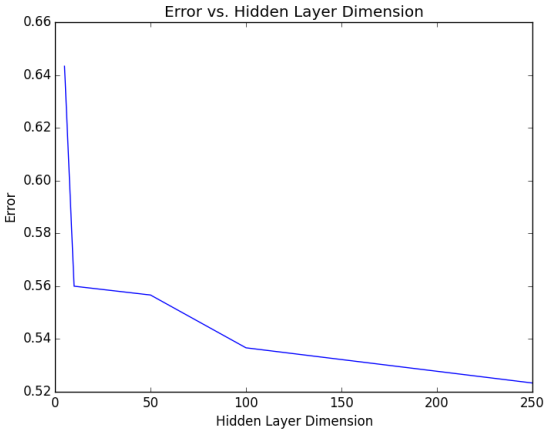


Figure 6: We performed experiments that test the effect of the hidden layer dimensionality on the overall model accuracy

As the dimension of the hidden layer increases, the accuracy of the model as a whole increased. However, as expected, the overall time spent training the model with higher and higher dimensionality exponentially increased. The last experiment, where the hidden layer dimension was 250, took about 10 hours to train.

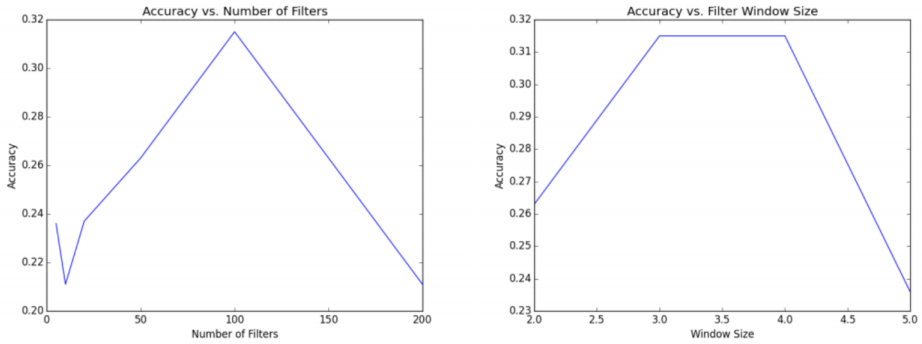
We then compare the performance of CNNs with zero padding and chunking, optimizing each using a grid search over the parameters. While the CNN with zero padding achieves a maximum accuracy of 28% after grid search, the CNN with chunking achieves 34%. We hypothesize that the zero padded CNN performs poorly because the majority of the features in the feature matrix for shorter reviews would consist of zeros, thereby creating spurious features in the feature maps. This issue is magnified in review classification due to the increased variance in review length. Chunking performs surprisingly poorly as well, given that it avoids the zero padding problem.

In order to ensure that our CNN implementation did not contain a bug, we run Kim's implementation on the Yelp dataset for comparison. However, it also performs quite poorly on the dataset, indicating that its performance does not translate well from sentence classification to review classification.

Table 1: Model Comparison

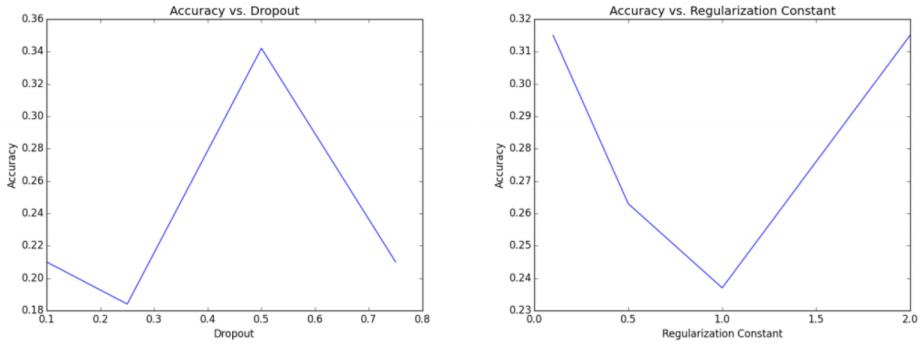
Model	Accuracy
Softmax regression	0.45
RNN with LSTM	0.51
CNN with zero padding	0.28
CNN with zero padding (Kim)	0.32
CNN with chunking	0.34

Finally, we examine the effect of varying each parameter on the accuracy of the convolutional network, as shown below in Figure 6. We find that using 100 filters, a window size of 4, dropout of 0.5, and a regularization constant of 1.0 achieves the optimum accuracy, the same parameters used by Kim (2014).



(a) Filters

(b) Filter window size



(c) Dropout

(d) Regularization

Figure 7: We performed experiments varying the parameters of the CNN to optimize accuracy and explore the effects of each parameter.

5 Conclusion

Through the various experiments, we learned much about the models we implemented and the overall application of neural networks to Natural Language Processing tasks. We were optimistic when first beginning the project, but later realized that the models did not meet our initial expectations. This is partially a result of the fact that the papers we referenced focused on single sentence binary classification, while our task was multi-sentence sentiment analysis. Our task had 5 output possibilities, which greatly affected the accuracy of the implemented models. The multi-sentence structure of the reviews, as opposed to individual sentences, also negatively affected the accuracy of both the RNN and CNN, in comparison to the results reported by Kim et al and Hochreiter et al respectively. Particularly with the CNN, we learned that zero padding and chunking were ineffective in the context in which we used them.

In the future, we hope to implement a recursive neural network on top of the CNN for each chunk, which we anticipate will improve the performance of the overall model.

6 References

S. Hochreiter and J. Schmidhuber. 1997. Long Short-Term Memory. In *Neural Computation*, 9(8):1735-1780

N. Kalchbrenner, E. Grefenstette, P. Blunsom. A Convolutional Neural Network for Modelling Sentences. *In Proceedings of ACL 2014*.

Y. Kim. 2014. Convolutional Neural Networks for Sentence Classification. *In Proceedings of EMNLP 2014*.

R. Socher, J. Pennington, E. Huang, A. Ng, C. Manning. 2011. Semi-Supervised Recursive Autoencoders for Predicting Sentiment Distributions. *In Proceedings of EMNLP 2011*.