Amazon Food Review Classification using Deep Learning and Recommender System

Zhenxiang Zhou
Department of Statistics
Stanford University
Stanford, CA 94305
zxzhou@stanford.edu

Lan Xu
Department of Statistics
Stanford University
Stanford, CA 94305
lanx@stanford.edu

Abstract

In this paper we implemented different models to solve the review usefulness classification problem. Both feed-forward neural network and LSTM were able to beat the baseline model. Performances of the models are evaluated using 0-1 loss and F-1 scores. In general, LSTM outperformed feed-forward neural network, as we trained our own word vectors in that model, and LSTM itself was able to store more information as it processes sequence of words. Besides, we built a recommender system using the user-item-rating data to further investigate this dataset and intended to make connection with review classification. The performance of recommender system is measured by RMSE in rating predictions.

1 Introduction

Most e-commerce websites like Amazon, allow users to leave review about products and services they receive during their shopping experience. These reviews are important to other users when making the decision whether or not to buy the product. Thus, understanding the meaning of the reviews and correctly classify its helpfulness would be an useful thing for the websites to do. Also, the results of the classification can be used for further application such as summarization and recommender system.

In this project, we intended to classify the usefulness of each review using deep learning models. This task can be broken down into three parts: generating a vector representation for each sentence we encounter, using those vectors to train our model, and evaluating the performance of our classification model.

Besides, we also want to further investigate the dataset by building a recommender system based on user-item-rating data. As the dataset is sparse, the baseline model would be recommending popular items overall. We'll evaluate the performance of different methods like collaborative filtering and matrix factorization based on their RMSE.

2 Background and Related Work

With the rise of online shopping and social networks, reviews and ratings has become an important way to understand the sentiment and need of users. Being able to automatically filter the reviews has become a key challenge for those companies and websites. As a result, a lot of effort has been made to the NLP tasks such as sentimental analysis[1].

As the computing power increases, neural networks have become more and more popular for language modeling. Both feed-forward neural networks[2] and recurrent neural network have been used frequently in dealing with text categorization, document tagging and sequencial word generating problems. The LSTM (long short-term memory) neural network[3] is also gaining more and more attention in their performance in speech recognition.

A lot previous work has been done in building recommender systems as well. Collaborative filtering[4], Content-based filtering[5] and Matrix Factorization[6] have been the most successful ones with broad application in industry. Recently, deep Learning has been used for Music Recommendation[7]. Recurrent Neural Network is also adopted for collaborative filtering based recommendation as well.

3 Approach

3.1 Feed-forward Neural Network

We first tried the standard feed-forward neural network[15]. Our input is the word vector representing each of the reviews x. After forward propagating the input through the deep network, the output y_i would be a 2-dimensional vector representing whether a review is helpful or not. We experimented with different numbers of hidden units, hidden layer, non-linear relationship between the levels and number of embedding size. The resulting model is a five-layer neural network.

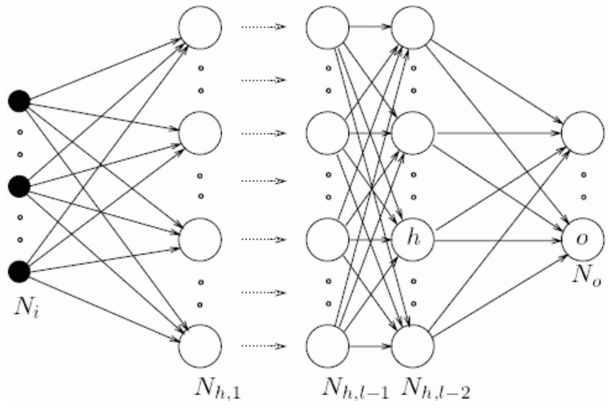

Figure 1: Work flow of Feed-forward Neural Network

The non-linearity used in the model:

$$h^{(1)} = ReLu(W^{(1)}x + b_1)$$

$$h^{(2)} = tanh(W^{(2)}h^{(1)} + b_2)$$

$$h^{(3)} = ReLu(W^{(3)}h^{(2)} + b_3)$$

$$h^{(4)} = Sigmoid(W^{(4)}h^{(3)} + b_4)$$

$$h^{(5)} = ReLu(W^{(5)}h^{(4)} + b_5)$$

$$\hat{y} = softmax(Uh^{(5)} + b_6)$$

After each hidden layer, we also added a drop out[14] layer with probability p to prevent the units from co-adapting too much. In addition, we used Xavier initialization to initialize all W,U in the network. Moreover, given that our input data class might be unbalanced, we assigned weights associated with each class's proportion to each class while calculating the cross entropy loss.

3.2 LSTM

To explore more advanced model, we adopted the LSTM model introduced by Hochreiter & Schmidhuber. In this model, we first trained our word vectors on a global word-word co- occurrence matrix(all words in all reviews) instead of using pre-trained word vectors. And then we feed our

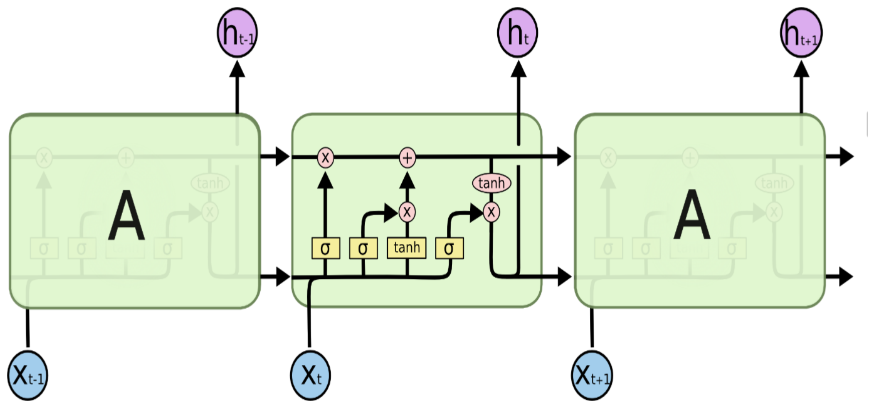

Figure 2: Work flow of LSTM From Colah's Blog[9]

representation vectors into LSTM model. This model in our case is essentially mapping sequence of words to a class and the work flow is shown below:

The mathematical equations for LSTM are:

$$\text{input gate: } \mathbf{g}^u = \sigma(\mathbf{W}^u * \mathbf{h}_{t-1} + \mathbf{I}^u * \mathbf{x}_t)$$
$$\text{forget gate: } \mathbf{g}^f = \sigma(\mathbf{W}^f * \mathbf{h}_{t-1} + \mathbf{I}^f * \mathbf{x}_t)$$
$$\text{output gate: } \mathbf{g}^o = \sigma(\mathbf{W}^o * \mathbf{h}_{t-1} + \mathbf{I}^o * \mathbf{x}_t)$$
$$\text{New memory cell: } \mathbf{g}^c = \tanh(\mathbf{W}^c * \mathbf{h}_{t-1} + \mathbf{I}^c * \mathbf{x}_t)$$
$$\text{Final memory cell: } \mathbf{m}_t = \mathbf{g}^f \odot + \mathbf{g}^u \odot \mathbf{g}^c$$
$$\text{Final hidden state: } \mathbf{h}_t = \tanh(\mathbf{g}^o \odot \mathbf{m}_{t-1})$$

After feeding input through the LSTM model, the output y_i would also be a 2-dimensional vector representing whether a review is helpful or not. And again, we considered class proportion in calculating cross entropy loss.

3.3 Recommender System

Among all the algorithms of building recommender system, the matrix factorization model is one of the most successful and widely used. Instead of using the similarity measure to recommend items to user, it characterizes both items and users by vectors of factors inferred from item rating patterns.

Denote the vector for user i as u_i, and vector for product j as p_j, our prediction of the rating of user i on product j can be written as:

$$\hat{r}_{ij} = p_j^T u_i$$

Therefore, to learn the optimal user and product vectors, our objective funtion is:

$$min_{u^*,p^*} \sum_{(i,j)} (r_{ij} - p_j^T u_i)^2 + \lambda(||p_j||^2 + ||u_i||^2)$$

4 Experiment

4.1 Dataset and Pre-processing

The Amazon Fine Food Review dataset consists of product and user information, ratings, and a plaintext review. The data span a period of more than 10 years, including all 500,000 reviews up to October 2012. After pre-processing, we used 290,000 records of review to build our model. The attributes of our data:

In our classification task, we labeled a review as helpful when:

Table 1: Attributes of Amazon food review data

Name	Description
ProductId	unique identifier for the product
UserId	unique identifier for the user
HelpfulnessNumerator	number of users who found the review helpful
HelpfulnessDenominator	number of users who indicated whether they found the review helpful
Score	rating between 1 and 5
Time	timestamp for the review
Summary	brief summary of the review
Text	text of the review

1 It has been voted by at least one user

2 More than 50% of the users find it helpful

In Feed-forward NN, we used the GloVe as our embedding for word vectors. A review is represented by the average of its word vectors of all words. In LSTM model, self-trained word vector has been used to represent reviews.

4.2 Review Usefulness Classification

For feed-forward neural network, the base line test classification accuracy is 50%, where just randomly classifies reviews. And if we use a simple softmax classifier, we obtain a base line test classification accuracy 0.6. After tuning the parameters in our neural network by changing learning rate, regularization rate etc., we have some improvements over the base line. For fixed optimal parameters: regularization rate $\rho = 0.004$ and learning rate $\alpha = 0.005$, drop out rate $p = 0.85$, we have following result:

Model	Training accuracy	Testing accuracy	F1 score
Model with unweighted loss, embedding size = 50	0.83	0.69	0.71
Model with unweighted loss, embedding size = 100	0.82	0.72	0.76
Model with weighted loss, embedding size = 50	0.85	0.71	0.7
Model with weighted loss, embedding size = 100	0.83	0.75	0.78

Table 2: Feed forward neural network

For LSTM model, after tuning parameters in network, we found optimal parameters are regularization rate $\rho = 0.001$ and learning rate $\alpha = 0.001$,drop out rate $p = 0.9$. And accuracies for different models are summarized below:

Model	Training accuracy	Testing accuracy	F1 score
Model with unweighted loss, embedding size = 50	0.85	0.79	0.8
Model with unweighted loss, embedding size = 100	0.88	0.81	0.82
Model with weighted loss, embedding size = 50	0.87	0.76	0.81
Model with weighted loss, embedding size = 100	0.89	0.85	0.86

Table 3: LSTM model

4.3 Recommender System

We intended to use a deep neural network based recommender system however due to time constraint we just implemented standard collaborative filtering and matrix factorization. To be consistent with the dataset we used in review classification, we also used that dataset and divided 70/30 as training/test set.

Our baseline model for recommender system is recommending popular product overall. RMSE on test set is used as a criteria evaluating the performance of collaborative filtering and matrix factorization model(showed in Table 4).

Table 4: Performance of different recommender systems

Model	**RMSE**
Popular(baseline)	1.7372
Collaborative Filtering	1.4538
Matrix Factorization	1.1198

As we can see from above, both collaborative filtering and matrix factorization beat the baseline model. Matrix factorization is the best-performing model, which matches our expectation.

4.4 Discussion

1. In our experiment, the LSTM model indeed outperforms the feed-forward model in terms of both testing accuracy and F1 score. The reason for this is probably the fact that LSTM allows for greater model complexity. It has the capability of bridging long time lags between inputs. In other words, it is able to remember inputs from many time steps in the past, which makes LSTM an advantage for learning long sequences with long time lags. In our case, LSTM learns to classify a review by considering sequence of words while feed-forward network learns to classify a review by a single averaged vector, where averaging vector may lose some information in original data.
2. From the accuracy table for both models, we can see that if we calculate loss with appropriate weights for unbalanced classes in training step, it will actually improve the model performance. The reason for this may be the fact that it gives right direction while back-propagating to calculate gradients.
3. Since in our project we are trying to classify a review to be useful or not, which is a binary problem, we expect to achieve higher test accuracy with such powerful techniques. After a close investigation in false positive and true negative observations, we found that there are brunch of long ”unuseful” reviews that are classified to be ”useful”. And in fact among long length reviews, they tend to be labeled as ”useful”. Hence, when a long review comes in the model, the model will probably have less power to detect its true identity. Moreover, for a review like ”`What else do you need to know? Oatmeal, instant (make it with a half cup of low-fat milk and add raisins;nuke for 90 seconds). More expensive than Kroger store brand oatmeal and maybe a little tastier or better texture or something. It's still just oatmeal. Mmm, convenient!`” is labeled as ”Useful”, for a review like `"The tiny Altoids are great for a quick breath freshening and the small box is great for portability. It's full, but compact.I haven't run out yet, and probably won't for a couple months, but I can't wait to buy it again."` is labeled as ”unuseful”. In our opinion, these two reviews are kind of similar so that it is hard to distinguish which one is useful and the other one is not. This ambiguity problem is due to our definition ”useful” mechanism and will probably in turn, hurt model performance.
4. Initially we tried to implement an recommender system combining information in the text and the rating information using neural networks. we want to accomplish something similar to the work in [7] which uses audio signal as the input space and built a content based recommender system. However, we struggled a lot to find out a proper way of representing the products with all information together. We also tried to see the possibility of building a RNN based collaborative filtering system, but the constraint in lacking buying history of each user of our dataset.

5 Conclusion

In this paper we implemented different models to solve the review usefulness classification problem. Both feed-forward neural network and LSTM were able to beat the baseline model. Adjusting the loss function helped in both models improving prediction accuracy. LSTM outperformed feed-forward neural network, as we trained our own word vectors in that model, and LSTM itself was able to store more information as it processes sequence of words. We also found out some interesting patterns in the way our classifier work: longer answers and answers with more positive words are more likely to be classified as useful.

For future work, we think there are three different parts we can work on:

1 We can keep tuning our existing neural network by choosing different parameters such as learning rate, regularization rate, embedding size etc. to further improve prediction accuracy. Besides, we can also try some state-of-arts tricks for neural network training such as Batch Normalization[12], Random Search for Hyper- Parameter Optimization[13].

2 After traditional Matrix factorization (MF), we can use deep network(convolutional, recurrent) to learn a function that maps item content features to corresponding MF latent factors. This will help solve the cold-start problem in building recommender systems[10].

3 Instead of Recurrent Neural Network, we can also try Convolution neural network model for our classification tasks[11]. In CNN, it can help us to get better feature to learn the task by alternating convolution and sub-sampling.

References

[1] Pang, Bo, and Lillian Lee. ”Opinion mining and sentiment analysis.” Foundations and trends in information retrieval 2.1-2 (2008): 1-135.

[2] Lam, Savio LY, and Dik Lun Lee. ”Feature reduction for neural network based text categorization.” Database Systems for Advanced Applications, 1999. Proceedings., 6th International Conference on. IEEE, 1999.

[3] Sundermeyer, Martin, Ralf Schlter, and Hermann Ney. ”LSTM Neural Networks for Language Modeling.” INTERSPEECH. 2012.

[4]Schafer, J. Ben, et al. ”Collaborative filtering recommender systems.” The adaptive web. Springer Berlin Heidelberg, 2007. 291-324.

[5]Lops, Pasquale, Marco De Gemmis, and Giovanni Semeraro. ”Content-based recommender systems: State of the art and trends.” Recommender systems handbook. Springer US, 2011. 73-105.

[6]Koren, Yehuda, Robert Bell, and Chris Volinsky. ”Matrix factorization techniques for recommender systems.” Computer 8 (2009): 30-37.

[7]Van den Oord, Aaron, Sander Dieleman, and Benjamin Schrauwen. ”Deep content-based music recommendation.” Advances in Neural Information Processing Systems. 2013.

[8] Pennington, J., Socher, R., & Manning, C. D. (2014, October). Glove: Global Vectors for Word Representation. In EMNLP (Vol. 14, pp. 1532-1543).

[9] Christopher Olah, Understanding LSTM Networks, retrieved from http://colah.github.io/posts/2015-08-Understanding-LSTMs/

[10]Hidasi, B., Karatzoglou, A., Baltrunas, L., & Tikk, D. (2015). Session-based Recommendations with Recurrent Neural Networks. arXiv preprint arXiv:1511.06939.

[11]Chellapilla, K., Puri, S., & Simard, P. (2006, October). High performance convolutional neural networks for document processing. In Tenth International Workshop on Frontiers in Handwriting Recognition. Suvisoft.

[12]Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167. Chicago

[13]Bergstra, J., & Bengio, Y. (2012). Random search for hyper-parameter optimization. The Journal of Machine Learning Research, 13(1), 281-305. Chicago

[14]Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. The Journal of Machine Learning Research, 15(1), 1929-1958.

[15]Derks, E. P. P. A., Pastor, M. S., & Buydens, L. M. C. (1995). Robustness analysis of radial base function and multi-layered feed-forward neural network models. Chemometrics and Intelligent Laboratory Systems, 28(1), 49-60.