Sentiment Analysis on Amazon Movie and TV Reviews

Shashank Tyagi
University of California San Diego
s1tyagi@ucsd.edu

Sai Gullapally
University of California San Diego
scgullap@ucsd.edu

Ashwin Ramesh
University of California San Diego
a5ramesh@ucsd.edu

ABSTRACT

Learning good feature representations from text is a problem of paramount importance in Natural language processing. Good feature representations of a review text can be used to obtain the sentiment of a given text. In this paper, we propose a LSTM based neural network architecture for sentiment analysis and test it on Amazon Movie & TV reviews dataset. We compare our model with Bag-of-Words, tf-idf, multilayer dense and 1d-convolutional neural network models. Our best model achieves 0.7557 and 0.7591 RMSE on validation and test set respectively. We train all our models with stochastic gradient descent algorithm.

KEYWORDS

Sentiment analysis, ratings prediction, neural networks, Tf-idf, LSTM,

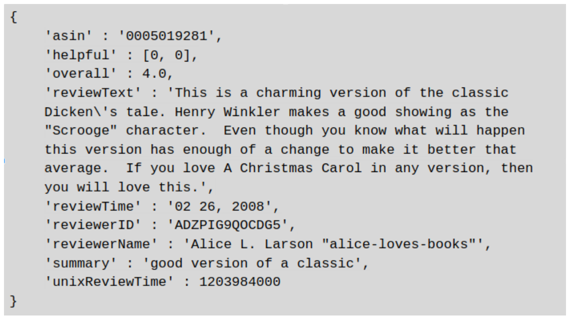
1 INTRODUCTION

Sentiment analysis can be defined as the task of inferring the reviewer's opinion on a product from the review he/she writes. This can then be used later for purposes like recommending it to other users, when a new customer plans to buy a new product, the reviews of the people who have already used that product play an important role in his/her decision making, so sentiment analysis can be used to get the summary from review. This behavior of customers is very important for e-commerce websites like amazon, e-bay etc. Consequently, sentiment analysis has become a major area of research in the field of data mining.

As an example, consider the review text shown in Fig.1. There is a lot information that can be garnered from this text viz. the review is about a movie based on Charles Dickens' novel, it is "better than average", etc. The occurrence of the phrases like "better than average" and "love" gives an idea about the sentiment of the reviewer. Thus, we can learn to predict the sentiment of the reviewer from the text. This is the problem that we're trying to solve in this paper.

In this work, we propose a model that makes use of the local correlation in the review text. We hypothesize that by exploiting this correlation we can come up with a better model for sentiment analysis. LSTM [7] lends itself naturally to this task. They have proved their utility in learning long term dependencies in natural language processing [3]. Our best model shown in Fig.13 also uses LSTM to model the local dependencies in the text.

In the next section, we review the dataset that we used to evaluate all the models. Section 3, provides a brief overview of previous work done in this field. In section 4, we describe all the models that we used. Section 5 provides analysis of the results. Finally in section 6, we provide conclusive remarks on this work.



```
{
    'asin' : '0005019281',
    'helpful' : [0, 0],
    'overall' : 4.0,
    'reviewText' : 'This is a charming version of the classic
    Dicken\'s tale. Henry Winkler makes a good showing as the
    "Scrooge" character.  Even though you know what will happen
    this version has enough of a change to make it better that
    average.  If you love A Christmas Carol in any version, then
    you will love this.',
    'reviewTime' : '02 26, 2008',
    'reviewerID' : 'ADZPIG9QOCDG5',
    'reviewerName' : 'Alice L. Larson "alice-loves-books"',
    'summary' : 'good version of a classic',
    'unixReviewTime' : 1203984000
}
```

Figure 1: A sample review form the data set [11]

Table 2: Initial statistics

Metric	Count
Unique users	112400
Number of reviews	1,697,533
Number of words in reviews	274,245,313
Number of unique words in reviews	5,185,241

2 DATASET

For our problem we make use of the Amazon Movies and TV 5-core reviews dataset [6, 11], which consists of 1,697,533 reviews. The data is in json format. Each user in the dataset has at least 5 reviews and each item has at least 5 ratings. Each review has the 9 features as shown in table 1. Fig.1 shows a sample review from the dataset.

Table 1: Features in a Review

Attribute	Description
reviewerID	ID of the reviewer
asin	ID of the product
reviewerName	name of the reviewer
helpful	helpfulness rating of the review
reviewText	text of the review
overall	rating of the product
summary	summary of the review
unixReviewTime	time of the review (unix time)
reviewTime	time of the review (raw)

2.1 Exploratory Analysis

In this section, we analyze the dataset thoroughly. Table 2 shows initial statistics gathered from the dataset.

Figure 2 shows the distribution of overall ratings. According to this histogram, the range of overall ratings is from 1 to 5, all of which are integers, and we find that the ratings are accumulated around 4 and 5 i.e., the reviews are mostly positive.

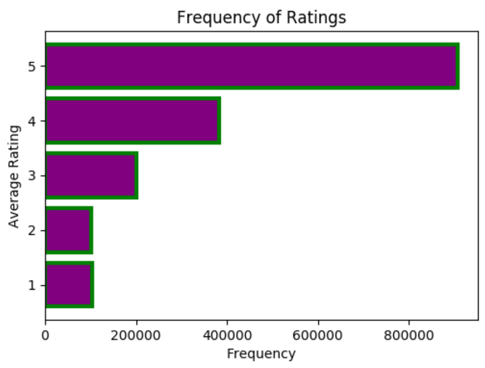


Figure 2: Frequency of Ratings

For this reason, we tried balancing the ratings by using uniform distribution. But this effort did not lead to better predictions, so we rolled back to the original distribution.

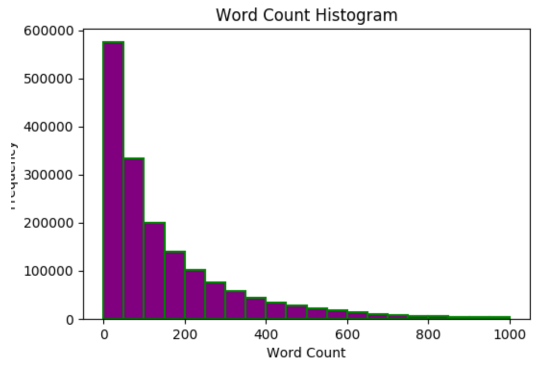


Figure 3: Word Count Histogram

Figure 3 shows the distribution of number of words in the reviews. From the histogram, it can be inferred that majority of the reviews have less than 500 words. Hence, we truncate all the reviews to 500 words in order to reduce the complexity of our model.

Figure 4 depicts the relationship between number of words in the review and the rating accompanying it. It can be observed that longer reviews are generally accompanied with higher ratings. There are extremely few cases where the rating is low for a relatively longer review. On digging deeper into the longer reviews, we found that people tend to express in detail if they liked the movie or the TV shows as opposed to when they don't like it, where they use strong negative words and keep the review short.

From the word clouds of figures 6 and 5, we see that there is a clear distinction between reviews with high and review with low ratings in terms of frequency of certain words in them. "Passion", "well", "best", "great", "good" are some of the common words in reviews with high rating, which express positive sentiment. Whereas

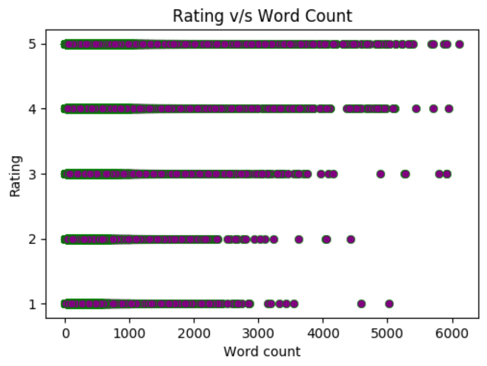


Figure 4: Rating vs Word Count

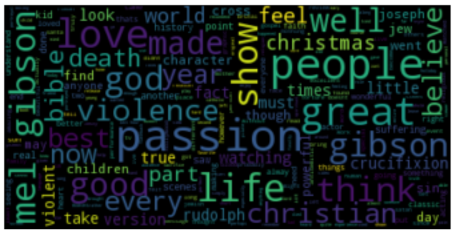


Figure 5: Positively weighted words

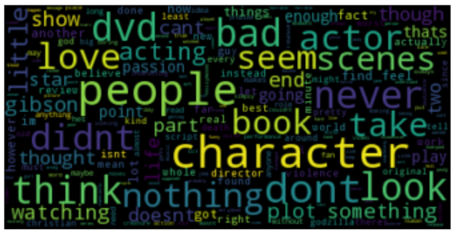


Figure 6: Negatively weighted words

words like "bad", "never", "don't", "didn't" which express negative sentiment appear more frequently in reviews with low rating. There are some frequently occurring neutral words like "people", "character", "actor", "love" that appear frequently in reviews with high as well as reviews with low ratings. These words don't express any particular sentiment. For example, if we consider the word 'love', some people may like romantic movies and some may not. Hence it can appear in both positive and negative reviews.

Given that the data is mainly distributed across the time period starting from the year 1999 to the year 2013, we analyzed how the average rating varies from year to year in Figure 7. It was observed that the there was a sharp increase in the average rating starting from 2011. This can as well be because the total number of ratings also increased sharply during the same time, as seen in the figure 8, which depicts how number of reviews submitted each year.

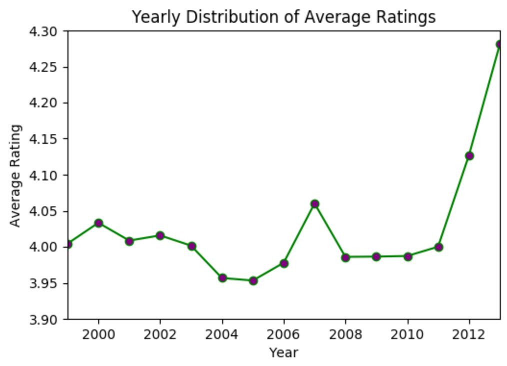


Figure 7: Yearly Distribution of Average Ratings

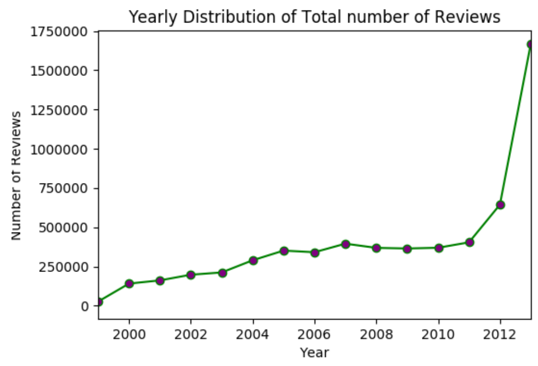


Figure 8: Yearly Distribution of Total number of Reviews

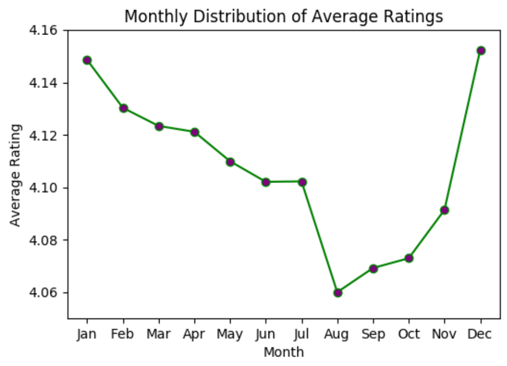


Figure 9: Monthly Distribution of Average Ratings

We also plotted the distribution of average ratings across the months of a year as can be seen in Figure 9. We found that the average ratings tend to be higher during December, January and February and lower during August and September. It can be seen in Figure 10 that the number of reviews per month also follows a similar distribution. One reason could be that due to this being the holiday season around December and January, people tend to watch

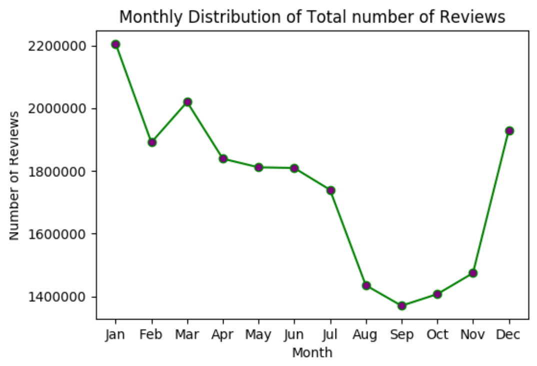


Figure 10: Monthly Distribution of Total number of Reviews

more movies and TV shows and also find time to write reviews. And since there are always more positive reviews compared to negative reviews at any given point of time, there is a corresponding increase in ratings when the review count increases.

2.2 Pre-processing

We shuffle the data and divide it into two parts - a training set of 1,497,533 reviews (20 % of which is used as validation data) and a test set of 200,000 reviews. The review text is also pre-processed for every model. We remove all punctuations and stop words in English language. We also truncate/pad all the reviews to 500 words as suggested by our findings in section 2.1.

2.3 Predictive Task

This paper models the problem of predicting the rating based on the user review text. In particular, given a review test, we predict the corresponding rating given by the user. Towards this end, we use a regression model that minimizes the mean squared loss with L2 regularization. The loss function is shown below.

$$J = \frac{1}{|\mathcal{T}|} \sum_{t \in \mathcal{T}} (r_t - \hat{r}_t(\Theta))^2 + \lambda ||\Theta||^2 \tag{1}$$

where $\mathcal{T}$ is the review corpus, r_t is the target rating, $\hat{r}_t$ is the predicted rating parametrized by Θ and λ is regularization weight. The value of the hyper parameter λ is tuned based on the results on validation set.

3 RELATED WORKS

Rating prediction is a very popular and important task in Recommender systems. An accurate prediction system would be highly profitable for companies especially e-commerce companies like Amazon, Netflix etc. Matrix factorization method by Koren et al. [9] which uses latent representations of the users and items is one of the most popular methods to predict the rating using the interaction matrix $R_{u,i}$ and latent representations γ_i,γ_u of user and item. But this method ignores the text review in the reviews. Sentiment classification is another approach where a review is analyzed for its sentiment and the rating is predicted based on the text because of the assumption that that there should a correlation between the sentiment expressed by the review and the rating given to the item. There are works which incorporate the information from the review text while predicting the rating. McAuley and Leskovec [10] propose a 'Hidden Factors as Topics' or HFT models. While a latent factor model can be used to uncover hidden dimensions in the ratings, Latent Dirichlet Allocation(LDA) can be used to uncover the hidden dimensions in review text. McAuley and Leskovec [10] use both of these together to predict the ratings, the LDA part of the model acts like a regularizer. In another work, Wang et al. [15] try to make use of the text in a different way, the proposed model is a linear combination of three different models, a regression model which rates a review based on the review's latent representation, the usual matrix factorization model without the biases and a k-NN model, it is a convex combination where the weights are not fixed instead they are learned while training. Our model is different from each of these two as it uses only the review text to predict the rating.

Other works like Tang et al. [13] use neural networks to predict the rating from the reviews, they argue that each word in a review must be interpreted differently depending on the user who gave the review, for example a critical user might give a rating of 5 while using the word "good" in the review. But some other user may give a rating of 3 and say "good", so they propose to make the representations of reviews to depend on the user who wrote them(similar to having a separate embedding for each user), once this is achieved they apply a simple multi-layer neural network to predict the ratings. Wadbude et al. [14] try to take care of the 'user bias' Wadbude et al. [14] before using a simple regression model, they remove the user bias in two possible ways, by normalizing the ratings based on the mean and standard deviations of the ratings given by each user (or) by the average deviations of the user's rating from each item's average rating. They use the same dataset as ours. Our model surpasses theirs in terms of RMSE. Hadad [5] also use amazon movies dataset (much bigger one) and try to predict the ratings from the reviews they make representations of each user and each item based on the all reviews a user gives at each level(1 through 5) of rating and the reviews each item gets at each level of rating(1 through 5) , once this is done they calculate the rating by a similarity measure(cosine similarity and its variations) of each level for user and item. Although many of them above use the Amazon reviews dataset they do not use the the same dataset (Amazon reviews, category:Movies and TV). As explained in section 5 our best model uses RNN's to capture the summary of the review in the state vector efficiently before using a dense layer to predict the rating.

4 MODELS

4.1 Baseline - Bag of words

We chose the unigram version of Bag of words model as the baseline method. In this model, after calculating word stems, we removed the stopwords and then considered the count of 1000 most frequent unigrams in the reviews as the feature vector. We then used simple linear regression model to predict the ratings. To avoid overfitting we used regularization, we used scikit-learn's implementation of Ridge regression model [1] for this purpose. In Ridge regression, the regression coefficients are usually non-zero and that is ideal for us as we don't want to eliminate any more of our input variables. The RMSE obtained when using this model is reported in table 3. This model does not consider contextual correlations and also a few words might be common to most of the reviews so they would not be informative but could adversely affect the rating.

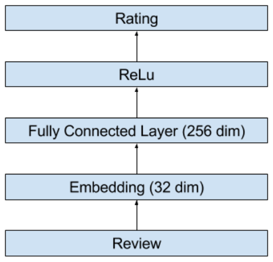


Figure 11: Multi-Layer Neural Network.

4.2 Tf-Idf vectorization with Ridge Regression

For a term t, Term Frequency (TF) measures how frequently the term t occurs in a document and Inverse Document Frequency (IDF) measures how important the term is by dividing the total number of documents by number of documents that contain the term t. Tf-idf weights are used to evaluate how important a word is to a document in a collection or corpus. This takes care of the words which are common to many reviews and thus may not be informative. We used the scikit-learn's implementation of TfidfVectorizer [2] to extract the tf-idf feature matrix. We experimented with vectorizers that broke text into both unigrams and bi-grams and then calculated the tf-idf representation. The unigram model ended up doing better and the values of RMSE have been reported in table 3.

To avoid overfitting and we use ridge regression in scikit-learn [1] for the same reasons as explained in 4.1. It applies L_2 norm constraint on the estimated coefficients:

$$\hat{\beta} = \underset{\beta}{\operatorname{argmin}} \, ||X\beta - y||_2{}^2 + \alpha||\beta||_2{}^2$$

We used the validation set to tune the hyperparameter α.

4.3 Multi-Layer Neural Network

In this model, we use a multi-layer neural network shown in Fig.11 with embeddings [12] in the input layer. The embedding layer learns a 32 dimensional feature vector corresponding to every word in the vocabulary. We hypothesize that embedding layer would perform better than tf-idf representation of the review text since it learns the representation of the text in supervised way. On top of embedding layer, we add a 256 dimensional fully-connected layer with ReLU activation. The final layer is again a fully-connected layer that gives out the predicted rating. We use Adam optimizer [8] with learning rate of 0.001 for training the network. We also employ weight decay of 0.001 and use early stopping on validation set to stop the training. This model out performs the previous model. The results can be seen in table 3.

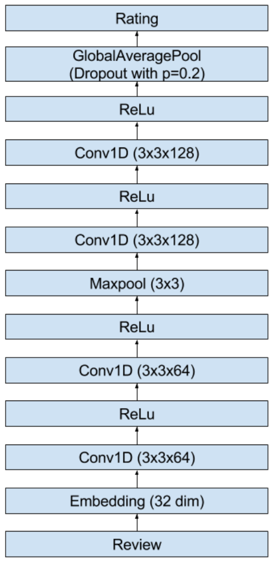


Figure 12: 1D-Convolutional Neural Network

4.4 1D-Convolutional Neural Network

In the previous model, the local neighborhood dependencies in the review text are not taken into account viz. "not bad" would be considered as "not" and "bad". We model this correlation using 1D convolution layers. The whole model is shown in Fig.12. The input layer is kept as embedding layer. In lieu of fully connected layers, this model consists of 4 1D convolutional layers with ReLU activation, a max pooling layer and global average pooling layer before the final prediction layer. Since, this model is significantly more complex and has more parameters than previous models, we use dropout with 0.2 drop probability on the penultimate layer and weight decay of 0.001. The model is trained using Adam optimizer with learning rate of 0.001. Again, we use early stopping on validation set to stop the training. This model outperforms the previous model. The results can be seen in table 3. We use Keras [4] for all the implementations henceforth.

4.5 LSTM model

We observe in the 1D convolutional model that modeling the local correlation in the review text boosted the performance of the model. We take this idea further by modeling even longer dependencies and also the hidden inherent structures in the review text. LSTM's have been quite successful at modeling long term dependencies the sequences. By considering a review as sequence of words, we can employ LSTM to model the text. The whole model is shown in Fig.13. It consists of a single LSTM layer on top of which a 64 dimensional fully connected layer with ReLU activation is used before prediction layer. The model is trained using Adam optimizer with learning rate of 0.001, a weight decay of 0.001. The training is stopped based in early stopping criterion on validation set. This model outperforms all the previous models. The results are shown in table 3.

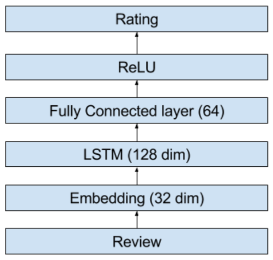


Figure 13: LSTM based Neural Network

Table 3: Results of different models

Model	RMSE Validation	RMSE Test
Baseline - Bag of Words model	1.0060	1.0841
Ridge regression with unigram Tf-idf	0.9103	0.9476
Ridge regression with bi-gram Tf-idf	0.9745	0.9882
Multi-layer neural network	0.8216	0.8358
1D-Convolutional Network	0.8090	0.8131
LSTM	0.7557	0.7591

5 RESULTS AND ANALYSIS

In this section we analyze the results obtained by the various models proposed in section 4. The unigram word count model which is the baseline model achieves an RMSE of 1.0060 on validation set and 1.0841 on test set. This model does not perform well enough as expected because we are loosing out on a lot of contextual information when we consider only the counts. One another factor would be the presence of words which are common to all the reviews and thus are not informative but could play a major part in the rating. To overcome the latter issue we moved on to the tf-idf representations, we calculate the tf-idf representations separately twice, once using unigrams and then using bigrams. To avoid overfitting, we used ridge regression. The unigram model achieves RMSE of 0.9103 and 0.9476 on validation set and test set respectively. On the other hand, the bigram model achieves RMSE of 0.9745 and 0.9882 on validation set and test set respectively. To better represent the words rather than using their counts or tf-idf values, we try to learn the word representations from the data in a supervised manner. For this we use word embeddings, using embeddings each word is given a 32-dimensional vector representation and the representation is assigned in such a way that contextually similar words (words which occur together) have similar i.e close-by vector representations. We then pass these representations through a Multi-Layer neural network, this enhance the results considerably, the achieved RMSE is 0.8216 and 0.8358 on validation set and test set respectively. These models did not consider the contextual information in the review text. To better capture the context we used a 1D-Convolutional Network which gives an RMSE of 0.8090 and 0.8131 on validation set. As expected this lowered the RMSE. The CNN would not be able to capture longer dependencies as the CNN's are intended to capture close local correlations but as we needed to capture longer dependencies we decided to use LSTM architecture. The resulting model summarizes the review text in the state vector and then predict the rating by passing this review through a dense neural network. This improved the results considerably as expected and achieved the best RMSE of 0.7557 and 0.7591 on validation and test set respectively. This shows that the sentiment can be captured well by the LSTM's hidden state vector.

6 CONCLUSIONS

In this paper we have put forth several models that we designed for inferring the sentiment of the reviewer form the review text. We started off with the traditional approaches like the bag of words representation, unigram-bigram with tf-idf representation. We then used word embeddings with a multi layer neural network to improve the predictions. To learn contextual information we used a 1-D CNN architecture and as expected the results improved. Subsequently, to learn much longer patterns and dependencies we used LSTM architecture and as expected this turned out to be the best model achieving an RMSE of 0.7557 on validation and an RMSE of 0.7591 on test set respectively. Thus we have shown our model does a good job at inferring the sentiment and predicting the ratings.

REFERENCES

[1] Scikit-learn ridge regression model implementation. http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Ridge.html, .

[2] Scikit-learn tdf-idf vectorizer implementation. http://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html, .

[3] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[4] François Chollet et al. Keras. https://github.com/fchollet/keras, 2015.

[5] Tal Hadad. Review based rating prediction. *CoRR*, abs/1607.00024, 2016. URL http://arxiv.org/abs/1607.00024.

[6] Ruining He and Julian McAuley. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. *CoRR*, abs/1602.01585, 2016. URL http://arxiv.org/abs/1602.01585.

[7] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.8.1735. URL http://dx.doi.org/10.1162/neco.1997.9.8.1735.

[8] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[9] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8), 2009.

[10] Julian McAuley and Jure Leskovec. Hidden factors and hidden topics: understanding rating dimensions with review text. In *Proceedings of the 7th ACM conference on Recommender systems*, pages 165–172. ACM, 2013.

[11] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. Image-based recommendations on styles and substitutes. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 43–52. ACM, 2015.

[12] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[13] Duyu Tang, Bing Qin, Ting Liu, and Yuekui Yang. User modeling with neural network for review rating prediction.

[14] Rahul Wadbude, Vivek Gupta, Dheeraj Mekala, Janish Jindal, and Harish Karnick. User bias removal in fine grained sentiment analysis. *arXiv preprint arXiv:1612.06821*, 2016.

[15] Bingkun Wang, Yongfeng Huang, and Xing Li. Combining review text content and reviewer-item rating matrix to predict review rating. *Intell. Neuroscience*, 2016:28:28–28:28, January 2016. ISSN 1687-5265. doi: 10.1155/2016/5968705. URL https://doi.org/10.1155/2016/5968705.