Predicting the Helpfulness of Consumer Reviews [*]

Team member
Jun Chen, Fanglin Chen, Qianyun (Poppy) Zhang
Team Advisor
Bonnie Ryan

May 12, 2017

Abstract

Online product reviews are essential for customers to evaluate how good the product is. However, not all the reviews are helpful in deciding whether to buy or not. Nowadays, online reviews are ordered by the number of total votes, but what if a new review is generated? We hope the most helpful reviews are displayed at the top of the list, not those with the most votes. We apply both machine learning and recurrent neural network to this problem, and the final accuracy is about 0.71.

1 Introduction

Increasingly, online reviews have become one of the most important information sources for consumers. Online review platforms are burgeoning with all types of products: book, music and home supplies (Amazon.com; ebay.com); movies (IMDB.com; rottentomatoes.com); and travel and various services (Expedia.com; Tripadvisor.com; Yelp.com). On one hand, consumers rely on reviews to reduce search costs and risks associated with products that they are not familiar with (Pavlou 2006). They also enjoy seeking information from product platforms and appreciate the availability of information (Kohli, Srikanth, and Bachhawat (2004)). On the other hand, online reviews are of high economic value to business: Ba and Pavlou (2002) showed reviews could differentiate sellers by creating price premium; Chevalier and Mayzlin (2006) demonstrated that reviews can drive sales using two data sets from online book markets; Resnick et al. (2006) found that seller reviews in eBay influence the probability of a sale; Chen, Dhanasobhon, and Smith (2008) and Zhu and Zhang (2006) found that reviews have a stronger impact on sales for niche products and less popular products.

Consumer-generated online reviews contain two types of information: auxiliary information, such as product overall rating, reviewer characteristics, motivation and disclosure in the review; and core information (review itself), such as review text, review star rating, review time. Information processing theory suggests that individuals integrate both of them into consideration when they make decisions (C.-K. Lee, Yoon, and Lee (2007); Jumin Lee, Park, and Han (2008); Jung Lee and Lee (2009); Gupta and Harris (2010); Chan and Ngai (2011); L.-Y. Pan and Chiou (2011)). The underlying reason that accounts for why consumers adopt reviews in their decision-making processes is that reviews can provide diagnostic information to consumers about the products and services (Kempf and Smith (1998); Jiang and Benbasat (2004); Pavlou and Fygenson (2006)). This diagnostic value can be measured using the 'helpfulness vote' that is available from online platforms. For instance, on Amazon.com, after each customer review, there is a question: 'Was this review helpful to you?' Individuals can vote on whether the review is helpful and can also see the summary statistics of the helpful votes of other users, such as '4 out of 5 people found this useful'. Helpfulness vote has been conceptualized as the peer-evaluation that facilitates a consumer's purchase decision-making (Mudambi and Schuff (2010)), and previous studies used the helpfulness vote as a proxy to measure the perceived diagnostic value of reviews to consumers.

However, little has been done to analyze the review text and how the textual elements impact the helpfulness of reviews. This project aims to analyze the textual information embedded in

[*]The codes of our models can be accessed at https://github.com/BruceChaun/DS-GA1003_project.

Table 1: Scope of the data:data variables

Product	Review	Reviewer
Product ID	Review ID	Reviewer ID
Category	Review Time	Reviewer Name
Sub category	Review Text	Total Reviews
Total Reviews	Review Title	
Average Rating	Review Rating	
Price	Reviewer	
	Helpful Votes	
	Total Votes	

reviews to fill in the gap in previous literature. Our findings can help both platform managers and business owners in managing their reviews and predicting the helpfulness of new reviews.

The following section will first present the data used in the project and then discuss our data pre-processing steps. After that, we will present the model we used in prediction and our findings. We will conclude with error analysis and discussion.

2 Data

We use all the reviews from 'Electronics' category from Amazon.com[1]. To get a robust and clean sample, we exclude reviews with fewer than 5 total votes, which gives us slightly over 240,000 reviews. Most of the reviews in our dataset are for computer accessories, TV, and camera. The scope of the data is outlined in Table 1. For each product, we observe the category it belongs to, and the total number of reviews as well as the average rating. For each review, we know the time it is posted, the title and textual content, the rating, vote and reviewer ID. For each reviewer, we observe how many reviews this reviewer has written and his/her name. Using the reviewer name, we can infer the reviewer's gender.

2.1 Data Pre-processing

We use two columns – *helpful* which indicates how many consumers support this review, and *total* which indicates the total number of votes – to construct the ratio of helpfulness. To slice the ratio into two buckets, helpful and not helpful, we need to use a threshold. The threshold can be regarded as a hyperparameter, but we hope the number of positive and negative samples are well balanced. In our experiment, we set the threshold as 0.8, so that the number of helpful reviews and not helpful reviews are very close to 1:1.

2.2 Feature Augmentation

To capture the variance in the text and generate meaningful features to use for analysis, we take the following approach: first, we generate basic features such as word count per sentence, word count per sentence and character count per word. Then we use standard packages to extract the readability and sentiment from the text. Next, we use packages to generate various text features, including but not limited to counts of punctuations, pronouns, prepositions, auxiliary verbs, numbers, quantifiers, articles, and swear words from review. Then, we apply pos-tagging and tf-idf, classical part-of-speech techniques, to generate grammatical features.

There are five major readability indexes we use: SMOG, Automated Readability Index, Coleman–Liau Index, Flesch readabilit and Cunning-Fog [2]. The readability index represents the grade level needed to understand a certain text corpus. It is negatively correlate with the actual readability of the text. To generate the sentiment, we do not simply match the negative words and positive words from the text. We also consider the functional words such as but, and functional punctuation such as semicolon. Following an approach developed by Rinker, T. W. (2017), we identify the amplifiers and negators that are close to words and consider them in computing the

[1]This is a subset of the data used in a large scale study by Professor Vishal Singh and Qianyun (Poppy) Zhang
[2]Formula used in computing each readability index can be found at friedman2006systematic

sentiment of text. Our sentiment index is generated on sentence level. For each review, we get an average index and also a standard deviation index on review level.

Relying on the pos-tagging techniques, we are able to identify the grammatical use of words in review text. We tokenize the words in each review, and use the NLTK package in Python to generate pos tags for each word. Then we calculate the percentage of each possible tag in the whole review text, such as the percentage of words tagged as PRP (pronoun, personal) or CD (numeral, cardinal).

2.3 Principal Component Analysis

In total, we have over 80 features before we conduct any analysis. This amount of features not only make it hard to interpret the results but also gives us a lot of computation burden on our local laptop. In addition, according to figure 1, a lot of features are correlated with each other.

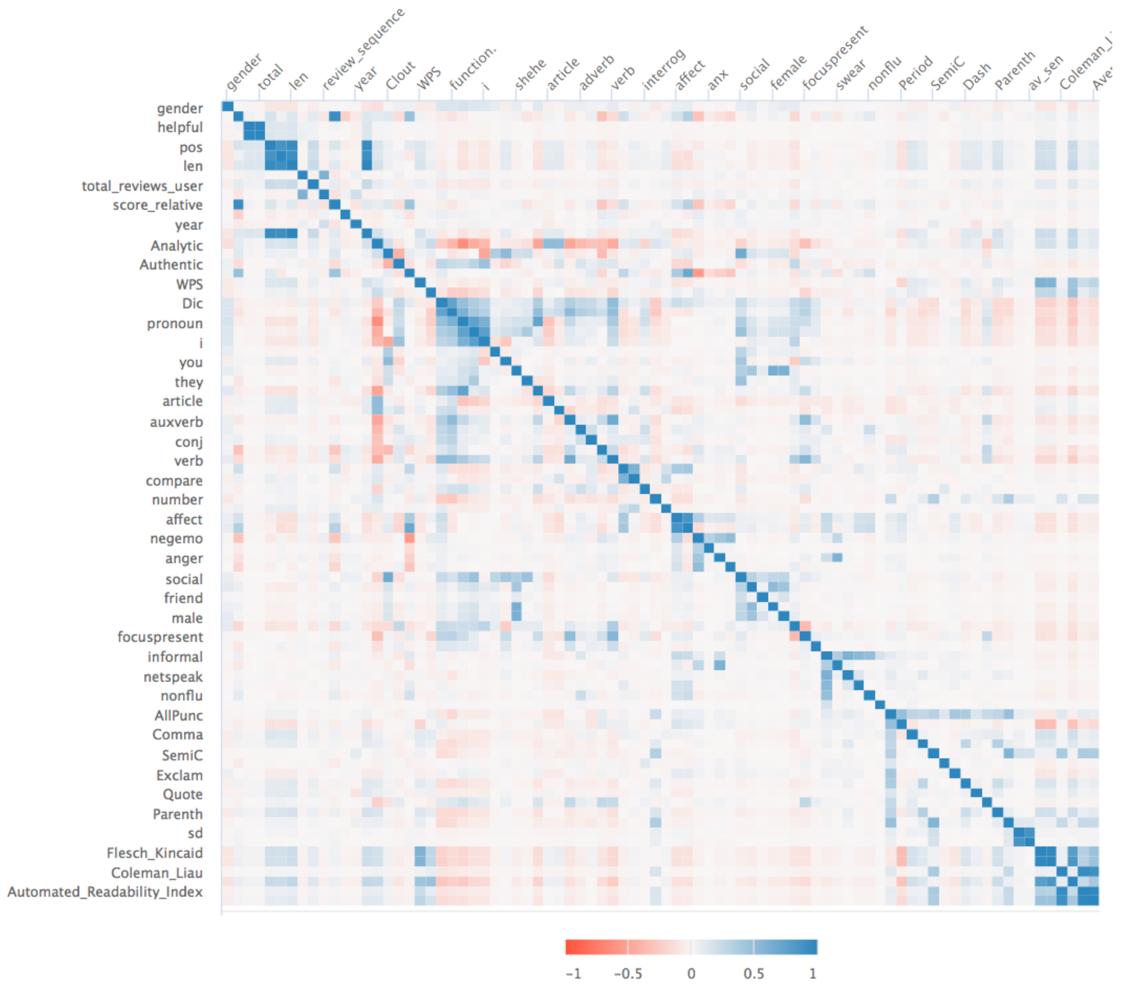


Figure 1: Correlation between features before PCA analysis

To solve this problem, we apply principal component analysis to our numerical features after re-scaling. It turns out we can capture 90 percentage of variance in the features by using only 46 of the factors that we get after analysis. The screen plots below presents the variance explained by the factors from principal component analysis.

3 Models

- AdaBoost

  In this setting, we set the helpfulness threshold to 0.8 to balance the ratio between positive

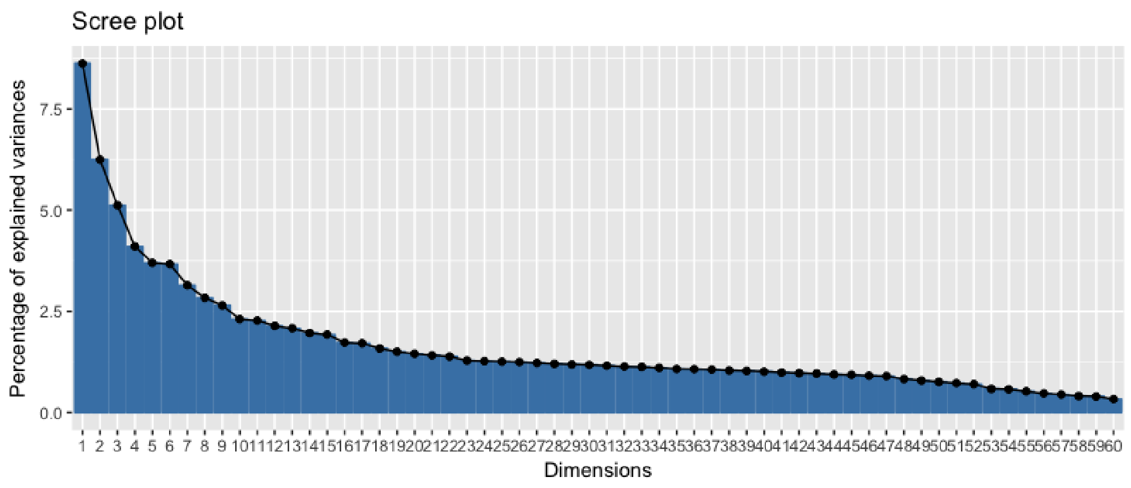


Figure 2: Screen Plot for PCA analysis

samples and negative samples. And we set the base classifier as decision tree classifier with maximal depth 4. Running epoch, or the number of base classifiers, is set as 100.

- Logistic Regression

  In this model, we use logistic regressions with l1 regulation. The model is fit with either the raw features or principal components.

- SVM

  In this model, we use SVM with l1, l2 or elastic net regulation. Same as above, the model is fit with either the raw features or principal components.

- Recurrent neural network (RNN)

  RNN language model has been widely used in many natural language processing (NLP) task, like text classification [10] and image captioning [9]. Evaluating the helpfulness of product review is very similar to text classification and sentiment analysis. What we need to do is to extract features that can well describe the sentence and do the binary classification.

  Long short-term memory (LSTM) [4] [1] has successfully solved the issue of long-term dependency and become very popular. We want to use LSTM module to help extract the features of sentence sequence. Inspired by [7], we feed the LSTM output into mean pooling before the fully connected network. We can also add a residual layer [3] between the mean pooling and fully connected network, or add batch normalization [5] layer, dropout [8] and parametric ReLU (pReLU) [2] into LSTM cell and/or fully connected layer. Figure 3 shows the basic architecture of our deep learning model.

  In our model setting, we set vocabulary size as $20,000$ (we introduce the numeric token – "$\langle CD \rangle$" – because we assume numbers may be important), excluding the "unknown" token, embedding and hidden size 50, dropout rate in both LSTM cell and fully connetecd layer 0.5. And because some of the reviews are very long, up to several hundred words, we filter those reviews whose length exceeds 100 words to avoid the gradient vanishing issue. We use Adam optimizer [6] and the loss function is negative log-likelihood (NLL).

4 Results

- AdaBoost

  In this model, we use 46 principal components as features, which can explain 80% of the original feature. Figure 4 shows the accuracy of training data, validation data and test data in the first 100 epochs. The accuracy is continuously increasing, the training accuracy reaches 0.72 while valiation and test accuracy 0.71.

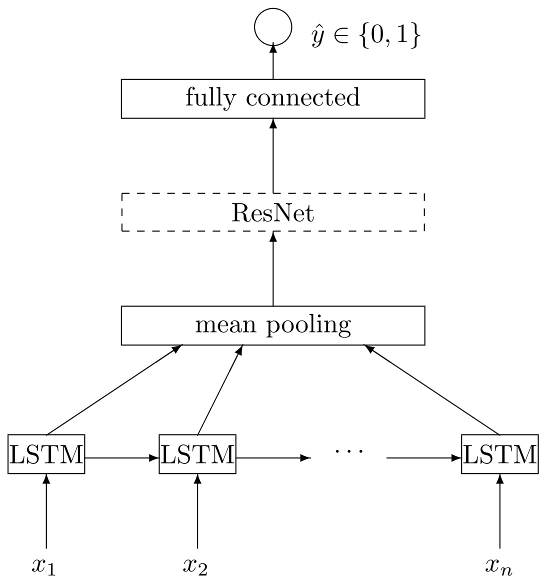


Figure 3: The basic architecture of the RNN model. x_i's are the input word embedding, they are fed into a single layer of LSTM cell. We put all the LSTM outputs into a mean pooling layer because simply taking the feature of the last word in each sequence does not get the correct result. We can use residual network later, or directly feed features into fully connected network.

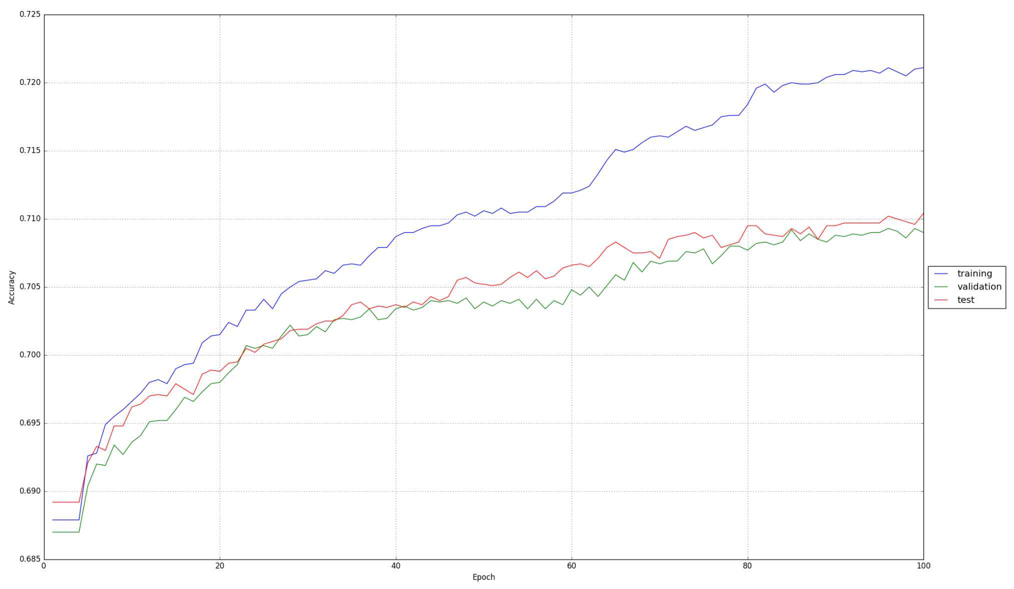


Figure 4: Accuracy of training, validation and test dataset in each epoch.

Figure 5 shows how much each principal component contributes to the final prediction. In the plot, we can see the first, second and tenth components influence the result most.

- Logistic Regression

  The best results for our logistic regression is reported in Figure 6, and the optimal regularization parameter is $1e-3$. The results using the raw features are included in the appendix.

- SVM

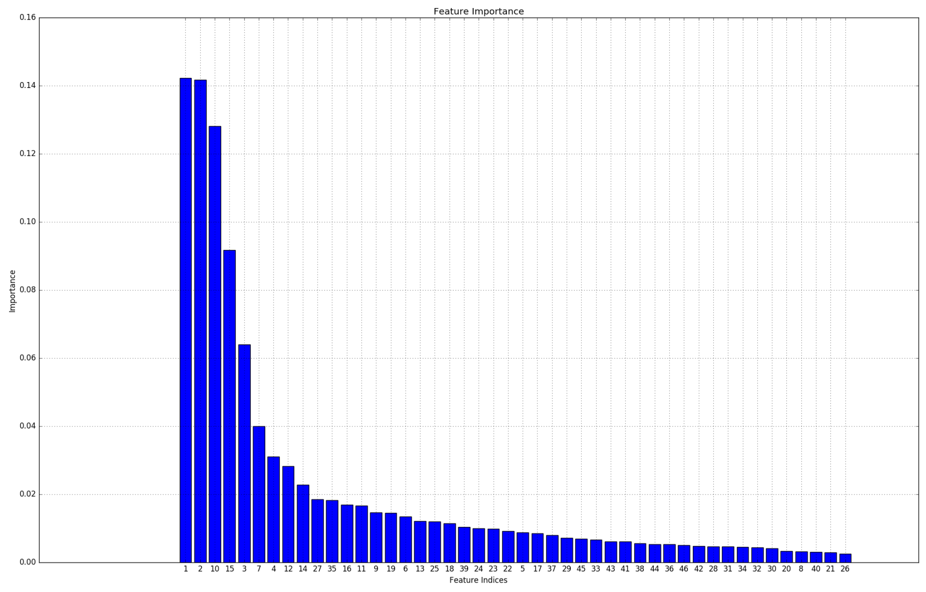


Figure 5: The importance of all the 46 principal components, ordered descendantly.

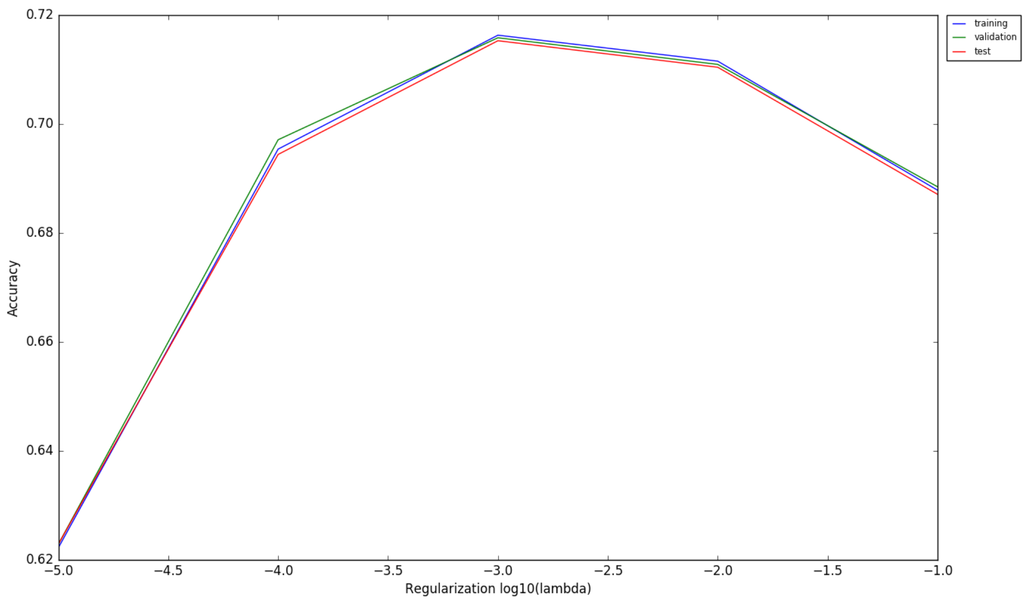
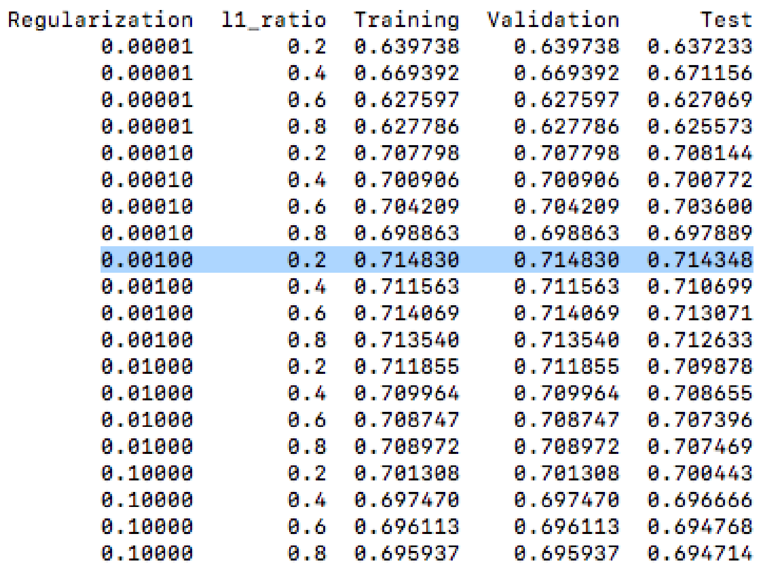


Figure 6: Logistic regressions with L1 regularization using PCA features

SVM is our best model so far. We have tried six versions of SVM: three kinds of regularization(l1, l2, elastic net) $*$ two sets of features (raw features, principal components). The best results are reported in Figure 7, and the optimal regularization parameter is $1e-3$, the optimal l1-ratio parameter is 0.2. With these parameters, we are able to achieve an accuracy of 71% in the test set. Other results are included in the appendix.

- Recurrent Neural Network (RNN)

  In the experiments, we have tried different versions of RNN model.



```
Regularization  l1_ratio  Training  Validation      Test
       0.00001       0.2  0.639738    0.639738  0.637233
       0.00001       0.4  0.669392    0.669392  0.671156
       0.00001       0.6  0.627597    0.627597  0.627069
       0.00001       0.8  0.627786    0.627786  0.625573
       0.00010       0.2  0.707798    0.707798  0.708144
       0.00010       0.4  0.700906    0.700906  0.700772
       0.00010       0.6  0.704209    0.704209  0.703600
       0.00010       0.8  0.698863    0.698863  0.697889
       0.00100       0.2  0.714830    0.714830  0.714348
       0.00100       0.4  0.711563    0.711563  0.710699
       0.00100       0.6  0.714069    0.714069  0.713071
       0.00100       0.8  0.713540    0.713540  0.712633
       0.01000       0.2  0.711855    0.711855  0.709878
       0.01000       0.4  0.709964    0.709964  0.708655
       0.01000       0.6  0.708747    0.708747  0.707396
       0.01000       0.8  0.708972    0.708972  0.707469
       0.10000       0.2  0.701308    0.701308  0.700443
       0.10000       0.4  0.697470    0.697470  0.696666
       0.10000       0.6  0.696113    0.696113  0.694768
       0.10000       0.8  0.695937    0.695937  0.694714
```

Figure 7: SVM models with elastic net regularization using PCA features

Table 2: Loss and accuracy of best model after 100 epochs of training. 100 and 500 are the two parameters that indicate the maximial sequence length.

Model	NLL loss			Accuracy		
	Train	Validation	Test	Train	Validation	Test
RNN V1 (100)	0.5607	0.5889	1.6775	0.7190	0.7025	0.6486
RNN V2 (100)	0.5823	0.5784	0.7041	0.7084	0.7041	0.6772
RNN V3 (100)	0.5890	0.5817	0.6324	0.7051	0.7007	0.6968
RNN V1 (500)	0.5095	0.5426	0.9837	0.7541	0.7303	0.7124
RNN V2 (500)	0.5594	0.5493	0.9837	0.7315	0.7269	0.7124
RNN V3 (500)	0.5778	0.5456	0.5805	0.6997	0.7287	0.7123

V1 This is our vanilla RNN model. We simply use word embedding, LSTM cell, mean pooling and one hidden-layer neural netwerk, without any activation function except *log_softmax* in the output layer, batch normalization or dropout. And we do not set l_2 regularization in the optimizer.

V2 Based on V1, we add pReLU activation function and batch normalization in the hidden layer, and dropout is added in the input and output layers of LSTM cell and in the fully connected network with different dropout rate.

V3 Based on V2, we add one residual layer between LSTM and fully connected network.

Besides, we also adjust other hyperparameters, like l_2 regularization term, numbers of hidden states and word embeddings, dropouts rates.

After 100 epochs of training, the accuracy results are shown in Table 4. From the table, we can see the third version of RNN model outperforms other two in the case of 100 sequence length, while there is little difference in the case of 500 sequence length. And we notice both NLL loss and accuracy improve if we choose a larger sequence length.

5 Discussion

By comparing the predicted labels with the true labels, we find some interesting patterns that could potentially be used as features in our future work.

- Brand names: some specific brands mentioned in the review. For example, 'The Foscam FI8919W is a great camera.' In this review, the brand 'Foscam' signals the high quality of the camera, and should be considered differently.

- N-gram: some phrases have totally different meanings when we consider them as phrases rather than separate words. For example, 'the isound i have try out is not that good' has different meanings if we look at 'good', 'that good' and 'not that good'.

We have implemented AdaBoost, logistic regression, SVM, and RNN model, and the result shows that the performances are quite close. In the traditional machine learning methods, we manually extract other features like POS tagging, readability index, etc, but these features may not capture the review text well. Besides, we convert this problem to binary classification, but the evaluation problem is not just black or white. If we set the threshold as 0.8, there may be a bias if one label is 0.75.

For the RNN model, explaining deep neural network is a difficult thing. Because our RNN model completely depends on the review text, we hope it outperforms the traditional methods. We analyze the raw text and the vocabulary we generate, and find the text data is noisy. An amount of sentences are followed by another without space. So when we tokenize the sequence, the output is erroneous because the last word of the first sentence, the period, and the first word of the second one are processed as one token, however, these should be three tokens. And some words are not capitalized at the beginning of the sentence. This will mislead the feature extraction and final result in both machine learning classifier and the RNN model.

The second issue is there are many modes of products, and the names of modes may become a word in vocabulary. This is very likely because customer may compare one product with another. This should be regraded as named entity, and we would better assign "unknown" token to them. If they are in our vocabulary, it is not very meaningful and reasonable.

To test whether it is the data quality issue, we also try to use IMDB dataset [3] [7], which is preprocessed and saved in a *pickle* file. We use our vanilla RNN model (Verison 1) and the result is shown in Table 5. The accuracy is much better than our dataset. So we believe with cleaner data, our RNN model can do better.

Table 3: Loss and accuracy of vanilla model, tested on IMDB dataset with different maximal sequence lengths.

max. length	NLL loss			Accuracy		
	Train	Validation	Test	Train	Validation	Test
200	0.3748	0.3161	0.5451	0.8592	0.8649	0.8614
300	0.3388	0.3650	0.8851	0.8688	0.8446	0.8520
500	0.3417	0.2897	0.8375	0.8657	0.8868	0.8685

6 Future Work and Conclusion

In this paper, we have described and implemented some models to predict whether a product review is helpful or not. Machine learning classifiers predict by using the information of product and reviewer as well as some features extracted from review text, while recurrent neural network uses word embedding and mean pooling to automatically extract hidden features. Due to the issue of cleaning data, the performance of our model is not good enough, but this does not mean our model cannot work, compared with the result of IMDB dataset.

In our further study, we would like to improve the performance. Except the data issue itself, one idea is to improve the RNN model, for example, by using bidirectional LSTM, multi-layer LSTM, and/or other pooling and attention methods. Another idea is to combine machine learning and deep learning because we cannot extract good features from text only in the traditional method while we may lose other indirect information if we use deep learning techniques only. We will further investigate a better model with these two ideas.

7 Appendix

[3]`http://www.iro.umontreal.ca/~lisa/deep/data/imdb.pkl`

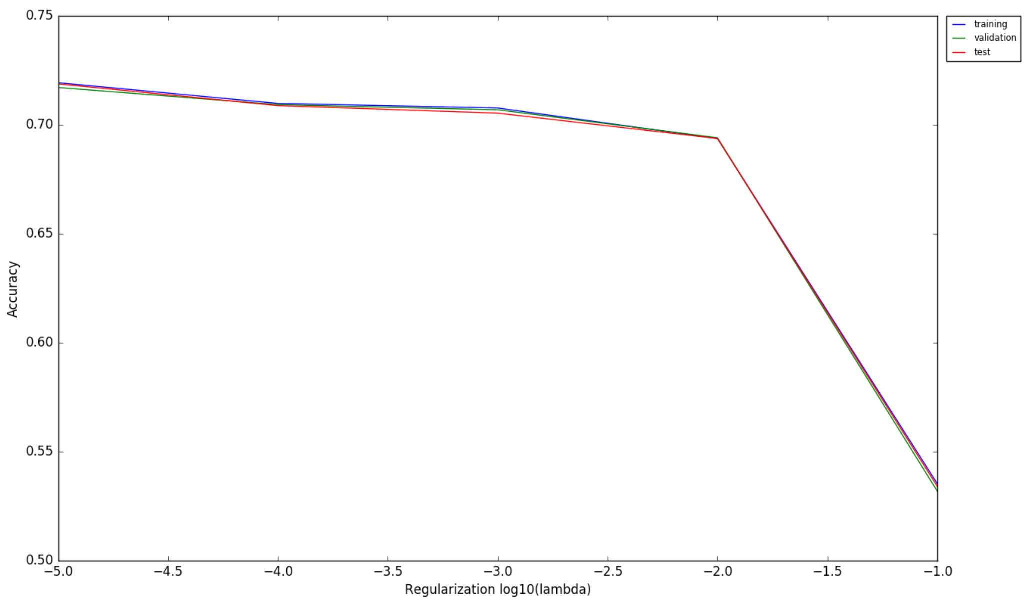


Figure 8: Logistic regressions with L1 regularization using raw features

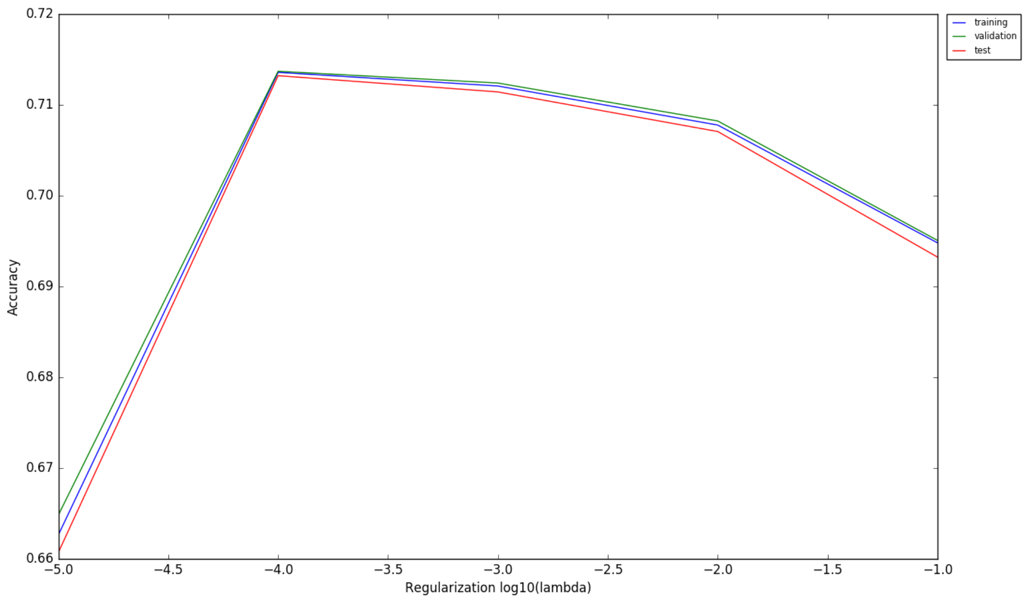


Figure 9: SVM models with L1 regularization using PCA features

References

[1] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with lstm. *Neural computation*, 12(10):2451–2471, 2000.

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for im-

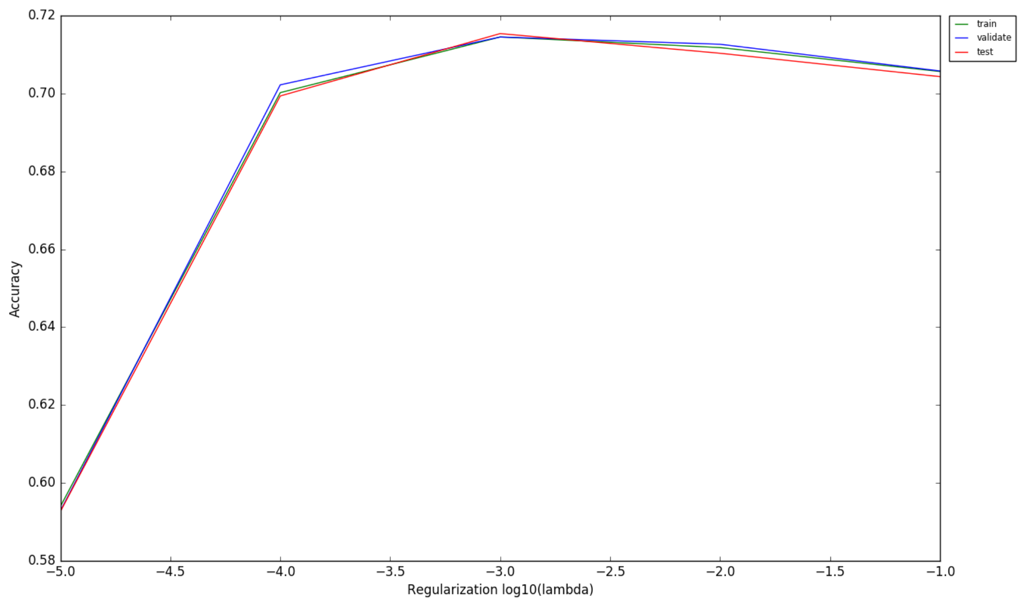
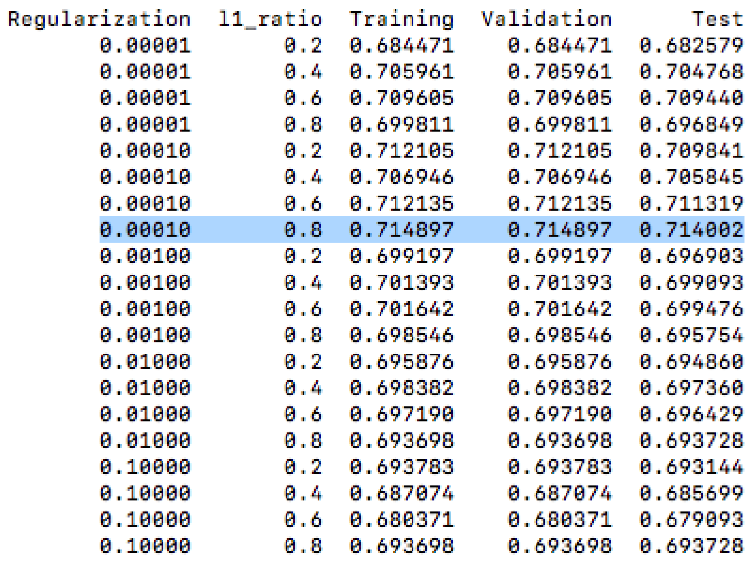


Figure 10: SVM models with L2 regularization using PCA features



Regularization	l1_ratio	Training	Validation	Test
0.00001	0.2	0.684471	0.684471	0.682579
0.00001	0.4	0.705961	0.705961	0.704768
0.00001	0.6	0.709605	0.709605	0.709440
0.00001	0.8	0.699811	0.699811	0.696849
0.00010	0.2	0.712105	0.712105	0.709841
0.00010	0.4	0.706946	0.706946	0.705845
0.00010	0.6	0.712135	0.712135	0.711319
0.00010	0.8	0.714897	0.714897	0.714002
0.00100	0.2	0.699197	0.699197	0.696903
0.00100	0.4	0.701393	0.701393	0.699093
0.00100	0.6	0.701642	0.701642	0.699476
0.00100	0.8	0.698546	0.698546	0.695754
0.01000	0.2	0.695876	0.695876	0.694860
0.01000	0.4	0.698382	0.698382	0.697360
0.01000	0.6	0.697190	0.697190	0.696429
0.01000	0.8	0.693698	0.693698	0.693728
0.10000	0.2	0.693783	0.693783	0.693144
0.10000	0.4	0.687074	0.687074	0.685699
0.10000	0.6	0.680371	0.680371	0.679093
0.10000	0.8	0.693698	0.693698	0.693728

Figure 11: SVM models with elastic net regularization using raw features

age recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

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

[5] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

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

[7] Kyunghyun Cho Pierre Luc Carrier. Lstm networks for sentiment analysis. `http://deeplearning.net/tutorial/lstm.html`, 2017.

[8] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[9] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International Conference on Machine Learning*, pages 2048–2057, 2015.

[10] Ye Zhang and Byron Wallace. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. *arXiv preprint arXiv:1510.03820*, 2015.