Helping Find Fine Foods

Elizabeth Tian
Princeton University
etian@princeton.edu

Victor Zhou
Princeton University
vzhou@princeton.edu

Devina Singh
Princeton University
devinas@princeton.edu

Abstract

Consumers everywhere today are now inundated with seemingly unlimited product offerings. All these products have not only thousands of numerical ratings, bu-talso a myriad of reviews on sites like Yelp and Amazon. With the rapid increase of technology and internet usage, the amount of information available to consumers has exploded. Ratings and customer reviews from many different sources, including Google, Yelp, and Amazon, are all potential sources of data for consumer decision making. How does one parse through all of this information to make the most informed decisions? What sort of information is most relevant and helpful to consumers when making these decisions? In this project, we use Amazon's Fine Food Reviews data set from Kaggle to address this issue by predicting the helpfulness of a review. This would help Amazon, as well as other B2C companies, provide better and more concise information for customers looking to purchase an item.

1 Introduction

Amazon is increasingly establishing itself as the largest and most extensive e-commerce company in the world. Its seemingly unlimited product offerings and millions of users result in hundreds or even thousands of reviews on popular products. The helpfulness rating of a review can add more dimensionality to the review and help Amazon better identify which sort of reviews that customers are more interested in reading. However, because this rating is completely manual, there is a significant lack of helpfulness data on all of the reviews. Thus, being able to identify which reviews might be helpful to consumers is an interesting problem that could be tackled through semantic analysis, natural language processing, or other machine learning methods.

To approach this issue, we utilize the Amazon Fine Foods dataset from Kaggle, which spans a period of more than 10 years and includes over 500,000 reviews on around 75,000 products up until October 2012. The variables that were taken from the reviews include product and user information, ratings, and a plain text review. In this report, we focus on utilizing and processing the text reviews in order to correctly label the helpful reviews. We use several different supervised learning models with the already-labeled helpful reviews (around 100,000) in order to classify each review into helpful, neutral, or unhelpful. These classification methods are: recurrent neural networks (RNN) including long short-term memory (LSTM) and bi-directional long short-term memory (BiLSTM), Random Forests (RF), and Support Vector Machine (SVM). For the last two, we use a bag-of-words representation for the reviews. We evaluate all the classifiers on their accuracy in categorizing their helpfulness.

2 Related Work

There exists a large body of previous work pertaining to the helpfulness of amazon reviews - Kim, Pantel et al. applied an SVM regression on MP3 player and digital camera reviews to predict helpfulness [7]. They found that the most useful predictive features included length of the review, product review and review unigrams, further suggesting that comparative words such as "more" or "better than" played a large role in these predictions. Through this we can infer that examining

textual features such as length and the types of words used in the reviews within the context of Amazon fine foods may yield similarly accurate predictions.

Furthermore, Korfiatis et al. relied on using a Random Forests model along with the readability features of a review i.e. reviews without grammatical or spelling errors [8]. They found a high correlation between readability and review helpfulness and determined that readability had a greater influence on helpfulness than review length. Moreover, Ghose and Ipeirotis similarly rely on a Random Forests approach along with readability, subjectivity of review and identity measures of the reviewer to demonstrate that highly subjective reviews tend to receive a lower helpfulness rating [6]. Therefore, we can infer from this that utilizing a Random Forests model along with a semantic analysis of the data could significantly improve the quality of our predictions.

3 Methods

3.1 Data processing

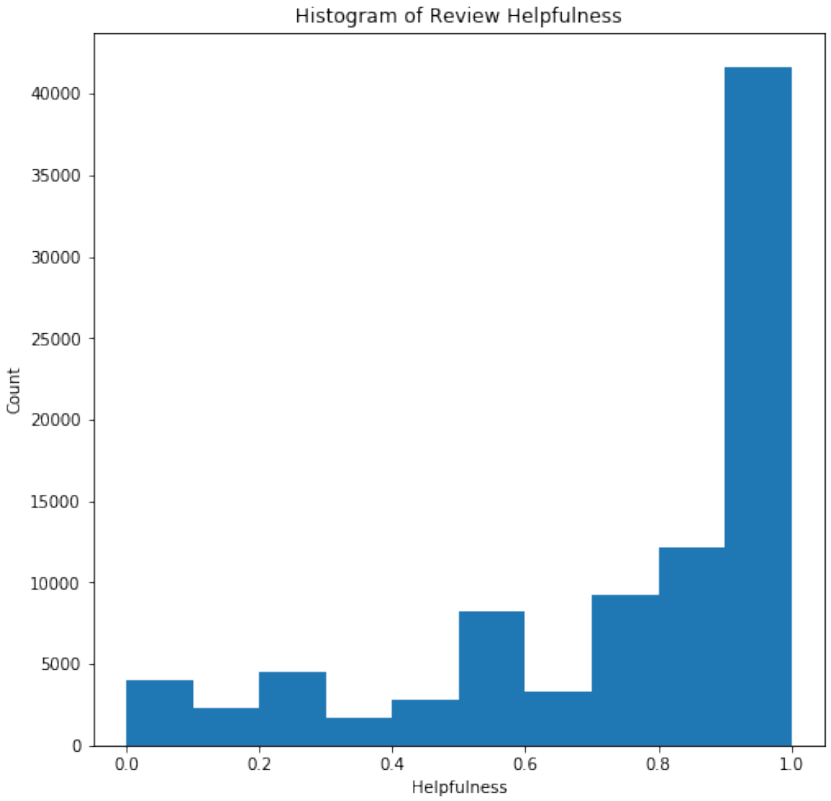

Figure 1: Histogram of review helpfulness.

As mentioned previously, we used the set of around 500,000 Amazon reviews of fine foods over 19 years [2]. These reviews were written by around 256,059 users about 74,258 products and the data has been processed and cleaned by the Stanford Network Analysis Project. Each review sample consists of a user ID, product ID, product score, time stamp, review summary, and text review. In order to determine the review labels, we used the "Helpfulness Denominator" (HD), the number of users who indicated whether they found the review helpful or not, and the "Helpfulness Numerator" (HN), the number of users who found the product helpful. We used the equation below to determine a helpfulness rating R:

$$R = \frac{HN}{HD}$$

Then, we used the following function as our review label, with 0 corresponding to not helpful, 1 to

neutral, and 2 to helpful:

$$Y = \begin{cases} 0 & R < 0.33 \\ 1 & 0.33 \leq R < 0.67 \\ 2 & R \geq 0.67 \end{cases}$$

When pre-processing the data, we chose to drop the reviews that did not have enough helpfulness information (those that had a helpfulness denominator of less than 4). This left us with around 18% of the data, or around $90,000$ reviews. We then split this data into a training set and a testing set, with a test size of 20%. We created a bag of words representation of the text reviews and the summary reviews using the `CountVectorizer` class from scikit-learn [10].

Figure 1 includes a histogram of the helpfulness of reviews we used in our dataset. Although there's clearly a skew towards helpful reviews (most of which have 100% helpfulness), there's a good spread of review helpfulness throughout the rest of the graph.

3.2 Classification models

We used 3 different models in order to classify review helpfulness. These models are as follows:

1. **Recurrent Neural Networks** (RNN): RNN is a type of neural network that utilizes an internal memory to allow persistent information to produce predictive results in sequential data. We use a Long Short-Term Memory RNN. In addition, as an extension after viewing initial LSTM results, we made a second version of the LSTM model.
   (a) **Long Short-Term Memory** (LSTM): LSTM networks extend the memory of RNNs by enabling RNNs to remember their inputs over a long period of time.
   (b) **Long Short-Term Memory Version 2** (LSTM2): LSTM2 is a stacked 2-layer LSTM that adds more complexity to the model.

   We implemented both of these LSTM models using Keras [4] with a Tensorflow [3] back-end. They used an embedding layer initialized with GloVe [11] 300-dim word embeddings.
2. **Random Forests** (RF): RFs combine many decision trees and train each one on a slightly different observation set. Each decision tree learns rules from this data to predict categories.
3. **Support Vector Machine with RBF Kernel** (SVM): SVM utilizes a hyperplane that splits the samples into categories in the most optimal manner using a kernel. We chose to use the default Radial Basis Function, or RBF, kernel.

In addition, we decided to run these models on different feature sets provided and selected features below:

- Full text review
- Review summary
- Text length

Review summary and text length were used as feature selection. The summary was selected because it provided a more concise and information-dense version of the full text. Text length was chosen because it is a useful feature commonly used in natural language processing applications, and it can provide a baseline with which the model is evaluated.

3.3 Evaluation

We utilize supervised learning in order to classify each of the reviews. Thus, we can evaluate review performance using metrics of accuracy, precision, recall, and the F_1 score. The accuracy is the fraction of predictions that match the calculated helpfulness label over the total number of samples in the test set. Precision and recall are the following, where TP is true positives, FN is false negatives, and FP is false positives:

$$\text{precision} = \frac{TP}{TP + FP}$$

$$\text{recall} = \text{true positive rate } (TPR) = \frac{TP}{TP + FN}$$

A higher accuracy, precision, and recall relates to a model that performs better on the helpfulness classification task.

The F_1 score calculates the weighted harmonic mean of precision and recall. A higher F_1 score also relates to better performance.

4 Spotlight Method: Long Short Term Memory (LSTM)

LSTMs work to mitigate the vanishing gradient problem where by the gradient i.e. values that are used to update neural network weights, shrink over time, causing the network to place less weight or even forget information learnt in the past [9]. LSTMs have mechanisms known as gates which recognize and preserve important information over a sequence of events, passing down this relevant information,resulting in better predictions [9].

In order to understand how an LSTM preserves this important information, we need to consider its architecture. A typical LSTM network is comprised of different memory blocks called cells with a cell state and a hidden state being transferred between these cells. Gates within these cells are involved in manipulating the memory of these cells. There are three types of gates: 1) forget gates, 2) input gates and 3) output gates [5].

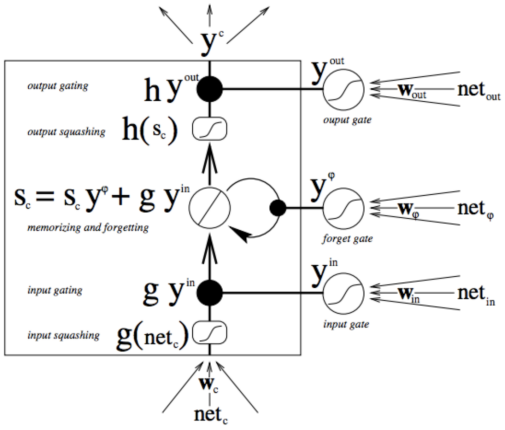

Figure 2: Flow of Information Through an LSTM cell with Forget, Input and Output Gates [1]

Forget gates remove information that is no longer required or important via the multiplication of a filter [9]. More specifically, a sigmoid function is used with the input h_{t-1} which is the hidden state of the previous cell and x_t which is the input at a particular time step to output a vector with values ranging from 0 to 1 [9]. Here a '0' represents information that should be forgotten and a '1' is information that should be definitely retained. This vector output v is multiplied to the cell state and is initially calculated as:

$$v_t = (W_v \cdot [h_{t-1}, x_t] + b_v) \tag{1}$$

where W_t is the weight matrix and b_t is some bias that is added [5].

Input gates are used to add important information to the cell state. This is done by a three step process where by a) a sigmoid function is used with h_{t-1} and x_t to create a vector of 0 and 1s that depict how important each value is, b) a tan function is used with h_{t-1} and x_t to create a vector of all possible values that can be added (where each value in the vector ranges from -1 to +1) and c) the sigmoid vector **s** (regulatory filter) is multiplied with the tanh vector **tv** and this result is added

to the cell state **C** [9]. We calculate these steps as [5]:

$$s_t = (W_s \cdot [h_{t-1}, x_t] + b_s) \tag{2}$$
$$tv_t = tan(W_{tv} \cdot [h_{t-1}, x_t] + b_{tv}) \tag{3}$$
$$C_t = v_t \cdot C_{t-1} + s_t \cdot tv_t \tag{4}$$

Lastly, the **output gates** work to select and output useful information using a similar three step process a) using a tanhn function on the cell state to scale the values to the range -1 to +1, b) using a sigmoid function to create a regulatory filter for the output values using h_{t-1} and x_t and c) multiplying this regulatory filter vector with the tanh vector and setting this result to be both the output signal of the cell and as the hidden state provided to the next cell [9]. This is similarly calculated as [5]:

$$o_t = (W_t \cdot [h_{t-1}, x_t] + b_t) \tag{5}$$
$$h_t = o_t \cdot tan(C_t) \tag{6}$$

As a result of LSTM's ability to maintain and propogate important information over time, it has many applications to time series related problems such as speech recognition. This is especially relevant to our project since many of the fine foods textual reviews are extremely long. Nevertheless, there still remain some limitations with using LTSM such as the exploding gradient problem where large error gradients may accumulate and cause large updates to neural network model weights, resulting in the model becoming unstable [1].

5 Results

5.1 RNN Analysis Results

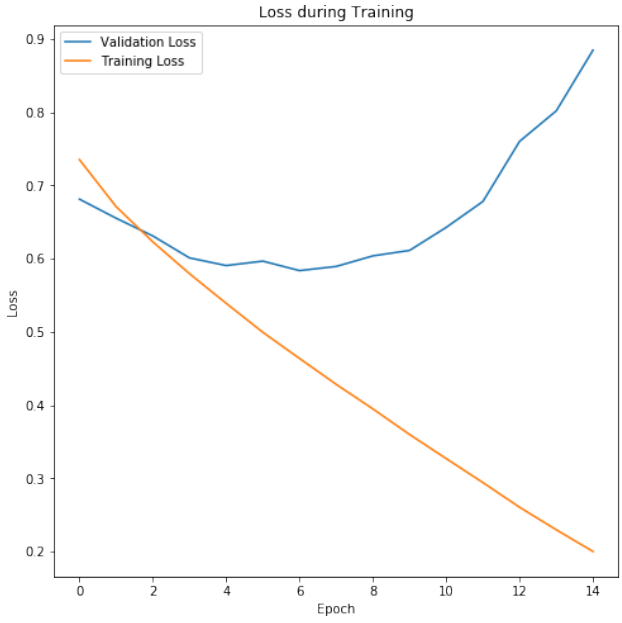

Figure 3: Graph of LSTM loss during different training epochs for the first version.

We decided to use a more complex model, a long short-term memory recurrent neural network, in order to attempt to capture the meaning within the review text. Thus, we used GloVe vectors that encapsulate the text meanings to represent the review data instead of a simpler BOW representation. The motivation behind this complex model was that the complex meanings within the reviews are what determine what consumers find to be helpful. However, when we used a long short-term memory recurrent neural network as a model to predict helpfulness of Amazon reviews, we did not

achieve the accuracy we were hoping for. Looking at Figure 3, we can see that the training loss decreases as the epoch increases, but the validation loss first decreases and then increases, a clear indication of overfitting.

In Figure 4, we can also see that the the training accuracy increases almost linearly with epoch, while the rate at which the validation accuracy improves is decreasing. The validation accuracy never reaches above the 0.8 mark, which was achieved by the random forest model (see Section 5.2).

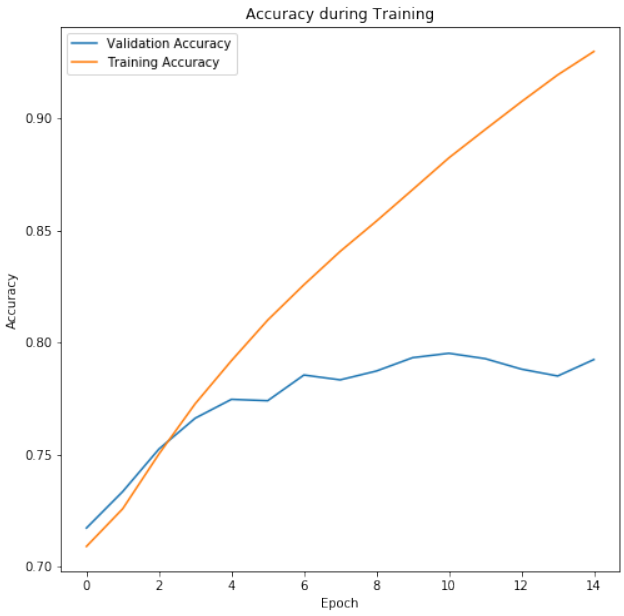

Figure 4: Graph of LSTM accuracy during different training epochs for the first version.

5.2 Random Forest Results

We used a random forest model on the full text review, the review summary, the full text and summary, and the text review length.

Feature Set	Accuracy
Full text	0.83
Summary	0.71
Text and summary	0.83

Table 1: Accuracy from the random forest classifier on text review data with different feature sets.

Feature Set	Precision (0)	Precision (1)	Precision (2)
Full text	0.67	0.71	0.77
Summary	0.95	0.97	0.81
Text and summary	0.96	0.97	0.81

Table 2: Precision from the random forest classifier on text review data with different feature sets. Scores are reported for each of the classes (0, 1, 2).

In general, the full text and the full text and summary were the feature sets that had the best performance for the random forest model, which makes sense because they had the most features. However, it is important to note that the text length as a singular feature still resulted in 71% accuracy, which is very close to the 83% that these feature sets had, indicating that perhaps the content of a review is not as important as it may seem when predicting helpfulness.

Feature Set	Recall (0)	Recall (1)	Recall (2)
Full text	0.28	0.27	0.97
Summary	0.42	0.44	0.99
Text and summary	0.42	0.41	0.99

Table 3: Recall from the random forest classifier on text review data with different feature sets. Scores are reported for each of the classes (0, 1, 2).

Feature Set	F_1 (0)	F_1 (1)	F_1 (2)
Full text	0.39	0.39	0.86
Summary	0.59	0.58	0.89
Text and summary	0.58	0.57	0.89

Table 4: F_1 scoree from the random forest classifier on text review data with different feature sets. Scores are reported for each of the classes (0, 1, 2).

From our results, we can see that the random forest model on the full text and on the text and summary out-performed any other model that we used, including LSTM version 1, LSTM version 2 (see Extension), and the SVM with RBF kernel. Thus, a simpler model with a bag of words representation may be a better way to predict helpfulness of Amazon reviews than a more complex model.

Class	Most Important Review Words
Overall	great, best, br, love, like, good
Unhelpful	worst, awful, return, threw, disgusting, msg, charged
Neutral	stream, horrid, swears
Helpful	great, best, br, love, like, good

Table 5: Most important words from the RF model on different classes and overall using the full text feature set.

Class	Most Important Review Words
Overall	great, best, product, excellent, delicious, horrible
Unhelpful	yuck, awful, worst, nasty, terrible, disgusting
Neutral	horrible, weak, hated, bland, okay, disappointment
Helpful	great, best, product, excellent, delicious, good

Table 6: Most important words from the RF model on different classes and overall using the summary feature set.

The important words for the helpful and unhelpful classes were generally very positive (great, best, love) and negative (awful, worst, disgusting), respectively. Furthermore, the neutral category of helpfulness also contained negative words (horrid, hated, horrible). This suggests that negative reviews are either not helpful or neutral, whereas positive words tend to be much more helpful for customers. This makes sense, because negative reviews could cause consumers to look for product alternatives (thus spending more time on their decision), whereas a positive review would sway a consumer to buy the product immediately. Most of the important words overall matched the important words for the helpful class, supporting the previous conclusion that positive reviews tend to be more helpful than negative ones. It is interesting to note that the line break ("br") for the full text feature set was the third most important feature overall, indicating that formatting and paragraph breaks within a review may make the text more readable and accessible to consumers, thus raising its helpfulness score.

5.3 SVM Results

When we ran SVM with an RBF Kernel on the review text data, we got a similar accuracy to if we had simply predicted that every single review was helpful (due to 70.8% of reviews being helpful in the dataset). This is probably largely due to the fact that we cut off SVM fitting after 100,000 iterations for the sake of time - because the Bag of Words representation has so many features (most of which aren't very useful), an SVM would take a very long time to find a decent fit. We concluded that we probably would not reasonably be able to train a good SVM model for this problem, so we stopped trying to do so.

Class	Precision	Recall	F_1
Unhelpful (0)	1.0	0.01	0.03
Neutral (1)	0.89	0.02	0.04
Helpful (2)	0.71	1.0	0.83

Table 7: Results from the support vector machine classifier on text review data with the entire review text.

	Accuracy
Train	0.715
Test	0.714

Table 8: Accuracies of the SVM classifier on review data.

5.4 Extension: LSTM Version 2

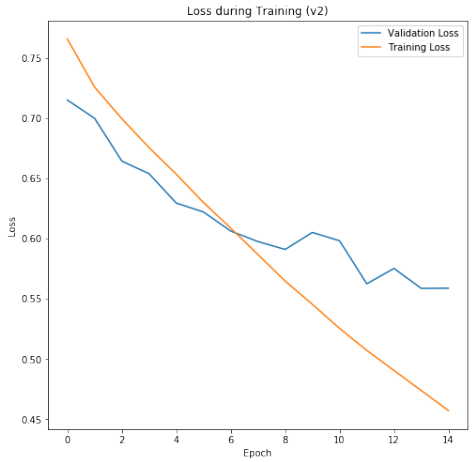

Figure 5: Graph of LSTM accuracy during different training epochs for the first version.

Because our simpler models tended to perform better than the complex neural networks, we wanted to improve upon the first LSTM model in order to see if we could achieve better results. Thus, we decided to run a 2-layer LSTM in order to increase the complexity of the model for our second version, LSTM2.

LSTM2 did not seem to have the same overfitting problem as LSTM as shown in Figure 5. However, the accuracy of LSTM2 was similar to that of LSTM, as seen in Figure 6. Thus, unfortunately, the performance of the model did not improve significantly, and it still performed worse than our Random Forest model.

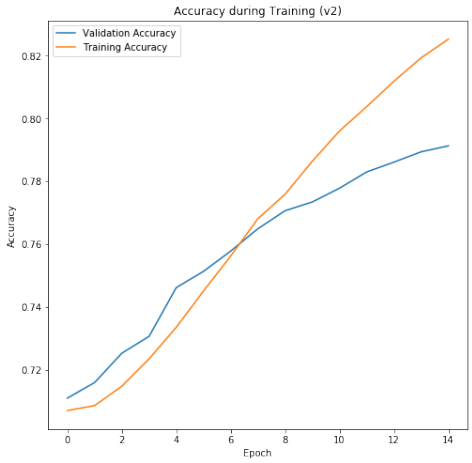

Figure 6: Graph of LSTM accuracy during different training epochs for the second version.

6 Next Steps

There are many ways we could continue the exploration we started in this project, such as:

- Building models with hand-picked features. The Random Forests models helped us determine many of the most useful Bag of Words features, and we could use a select group to build a model that could potentially improve the final accuracy.
- Performing unsupervised analysis to gain further insights into the data.
- Utilize time as a feature. The dataset was collected over **19** years, which is quite a long time (especially in the world of technology). It's very likely that reviews significantly changed over the course of those 19 years, so there is a lot more we could do here (e.g. looking at accuracy over different time periods, using year as a feature in a model, etc).
- Exploring the readability of a review as a feature. We think that humans reading and rating these reviews would appreciate more readable reviews and thus tend to rate them as more helpful. This could be an interesting avenue of exploration, especially given that our dataset includes the line break characters ("br") used in the original review HTML.

7 Conclusion

We found that our more complex LSTM-based models actually did not manage to achieve the same accuracies in predicting review elpfulness that our Random Forests models did, despite the fact that LSTMs had more parameters, and took much longer to train. This was not altogether surprising - it's possible that review helpfulness is much more correlated to the actual specific words used in the reviews and not the sentiments behind those words. That's the key difference between an LSTM-based model and a BOW-based model (which we used for the Random Forests) - the former represents words via GloVe vectors that encapsulates their meanings, while the latter represents words for what they themselves are. All in all, we felt like this was a good demonstration of the importance of using the right kinds of models for the problem and not just relying on brute strength.

References

[1] "A beginner's guide to lstms and recurrent neural networks, a.i. wiki."

[2] "Amazon fine foods dataset," *Stanford Network Analysis Project (Published on Kaggle)*, Jan 2016.

[3] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia,

R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: http://tensorflow.org/

[4] F. Chollet *et al.*, "Keras," https://keras.io, 2015.

[5] Colah, "Understanding lstm networks," 2015. [Online]. Available: https://colah.github.io/posts/2015-08-Understanding-LSTMs/

[6] A. Ghose and P. G. Ipeirotis, "Estimating the helpfulness and economic impact of product reviews: Mining text and reviewer characteristics. knowledge and data engineering, ieee transactions on, 23(10), 1498-1512." 2011.

[7] S. M. Kim, "Automatically assessing review helpfulness. in proceedings of the 2006 conference on empirical methods in natural language processing (pp. 423-430), association for computational linguistics." July, 2006.

[8] E. Korfiatis, N. anf García-Bariocanal and S. SánchezAlonso, "Evaluating content quality and helpfulness of online product reviews: The interplay of review helpfulness vs. review content. electronic commerce research and applications, 11(3), 205- 217," 2012.

[9] M. Nguyen, "Illustrated guide to lstm's and gru's: A step by step explanation," 2018. [Online]. Available: https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21

[10] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[11] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1532–1543. [Online]. Available: http://www.aclweb.org/anthology/D14-1162