CS230

Predicting Informativeness of Product Reviews: A Deep Learning Approach

Edvin Johansson
Department of Electrical Engineering
Stanford University
edvin@stanford.edu

Negar Matoorian
Department of Economics
Stanford University
negarm@stanford.edu

Artem Arzyn
Department of Linguistics
Stanford University
artem.arzyn@stanford.edu

Abstract

Many e-commerce websites collect user reviews about purchased products, often to display them on the website. This immediately creates the problem of deciding which reviews are the most *helpful* for new customers. We explore three different supervised approaches to predict helpfulness: a CNN-based, an LSTM-based, and a transformer-based, as well as an unsupervised method, testing all four on Amazon product reviews. We find that the LSTM-based approach works the best, followed closely by the transformer, which have average test errors of 0.169 and 0.173 respectively.

1 Introduction

Many websites that sell products or services allow buyers to post reviews of what they have purchased. Product reviews allow customers to make more informed purchasing decisions and offer vendors a means of communicating their product quality to customers.

Therefore, the ability of an e-commerce platform to rank reviews based on their *helpfulness* has direct effect on the amount of sales in the online market. The input to our algorithm is review text from an Amazon dataset of product reviews, specifically selected from the electronics category, and the goal of this project is to predict a score from 0 to 1 of how helpful the review is. We explore four different approaches of doing this. Three different deep neural networks: a CNN-, an LSTM- and a transformer based network, as well as an unsupervised approach based on keyword extraction and matching.

Note that this task differs from sentiment analysis, because both positive and negative reviews can be very helpful for buyers, and even reviews without strong feelings can still convey helpful product information.

2 Related work

Existing research on review helpfulness has involved a variety of data sets, approaches, and results. We looked at papers to influence our approach and parameters.

Previous studies ([8] and [14]) analyzed review helpfulness without NLP. Despite their limitations, they provided valuable insights. [8] limited their data set to only top reviewers on Amazon, focusing on analyzing reviewer ranking and word count. They focused specifically on quantitative review characteristics, using Tobit Regression to analyze the impact of word count, reviewer characteristics (based on number of reviews written), and product rating. Although they did not use NLP techniques, their results on the impact of word count, supported our choice to set a standard length for reviews. [14], also used regression models, except instead examined the relationship between informativeness and helpfulness, looking at several attributes not directly connected to the content of reviews (such as

CS230: Deep Learning, Spring 2023, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

information on the products, platform, and reviewers). [8]'s choice to focus on highly ranked reviewers and [14]'s choice to remove a substantial amount of data deemed repetitive or incomplete during preprocessing informed our choice to limit our analysis to relevant reviews that have enough votes in helpfulness analysis.

The remainder of the research implemented some form of deep learning to conduct an analysis. First, during the dataset selection and preprocessing, we saw variation. Based on the large user base on Amazon and its wealth of publicly available data, we chose to follow the path of [16] and [5]. From there, we saw different preprocessing across the three, with [10] discussing the removal of punctuation and stop words, as well as the use of stemming, whereas [16] set all reviews to a specific length (with either truncation or padding). Given the two instances of length selection, we decided to apply the same as we did preprocessing.

All three of these used neural networks and word embedding, albeit with different specifics. LSTM networks played roles in [5] PRH-Net model and [16] analysis, leading to our choice to use them for one of our supervised models. With regards to word embeddings, [10] specifically mentioned the use of Word2Vec to obtain document encodings while [16] drew from GLoVe embeddings. Based on the accessibility of GLoVe and its prevalence in the NLP space, we chose to implement it. For the final analysis to determine helpfulness, [16] most aligned with our approach, categorizing reviews as positive or negative and training on classification rather than regression how we chose. [10] used K-means clustering, but given our focus on determining the value of helpfulness, we did not follow this approach.

[7], [4], [3], [12], [9], [11], [6], [2] are libraries that were used in this project.

3 Dataset and Features

We are using a dataset consisting of product reviews on Amazon [1]. The dataset contains more than 130 million customer reviews from 1995-2022, from which we chose to use customer reviews in electronics.

A potential problem we saw with the Amazon data set in general is that many reviews have very few interactions with readers. I.e. a review may have only two votes, and perhaps one of the two readers indicated that the review was helpful. With such a low vote count, the labels are very noisy and do not necessarily reflect the true helpfulness of the review. As such, we decided to only train on reviews with ≥ 5 total votes. In the electronics data set there are about 300,000 such reviews.

The data set includes many useful fields and we are specifically using the *review text*, the amount of *helpful votes* (integer), and the amount of *total votes* (integer). Figure 5a, Figure 5c, and Figure 5e show example reviews from the dataset and their helpfulness labels (defined as $\frac{\text{helpful votes}}{\text{total votes}}$). We use a 90/5/5 % split of the dataset into train-/dev-/test- datasets, resulting in about 270,000 reviews for training and 15,000 reviews each for development and testing.

3.1 Preprocessing

We preprocessed the input slightly by changing the text to lowercase, adding spaces around punctuation (. , ! ?), and by splitting up contractions such as *don't* into *do not* or *would've* into *would have*. We also set a fix maximum word count for the input reviews at 200 words; reviews shorter than that were padded and longer reviews were cut off.

4 Methods

As previously mentioned, we explored three different architectures for supervised learning, and one approach of unsupervised learning. The supervised methods all used the mean absolute error as their loss function.

4.1 Word embedding

As commonly done for networks that use text as its input, we use an embedding layer as the first layer for all our models, which is the same for all models. We use the pre-trained word embedding model GloVe [13], developed by Stanford NLP researchers to convert a sequence of words into a numerical matrix based on statistical co-occurrence of words in text from Wikipedia, and assigns a 100 dimensional

embedded vector to each word in the text. This step also takes care of normalizing the input since each element in the embedding is in $[0, 1]$. We did not train the weight of the embedding layer.

4.2 Supervised methods

The architecture of the CNN is shown in Figure 1, together with the model's hyperparameters. After the embedding layer follows three repetitions of: a convolution, followed by a ReLu activation function, followed by maxpooling. After the third "conv→relu→maxpooling"-unit follows a flattening- and a dense layer with a sigmoid activation function to generate the output.

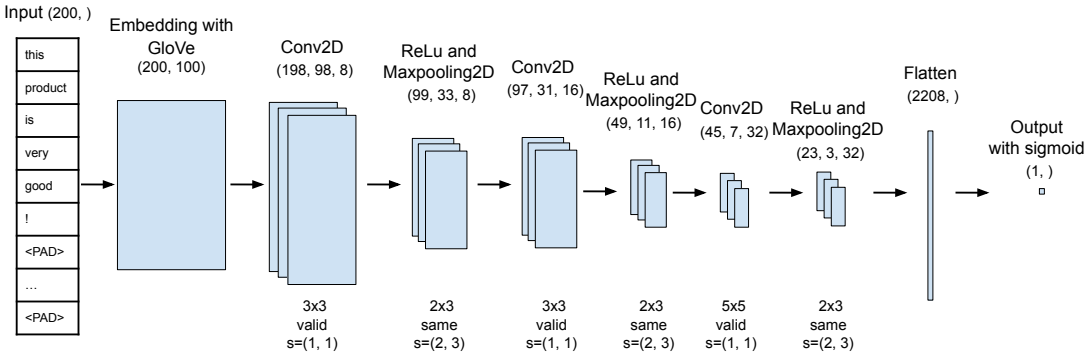


Figure 1: Architecture of the CNN.

The architecture of the LSTM is shown in Figure 2, together with the model's hyperparameters. After the embedding layer follows a bi-directional LSTM layer (which returns an entire sequence of the same shape as its input) and a dropout layer. It is then followed by another bi-directional LSTM layer (which only returns one output), followed by a dropout layer and dense layer (over the embedding dimension) with a sigmoid activation function to generate the output.

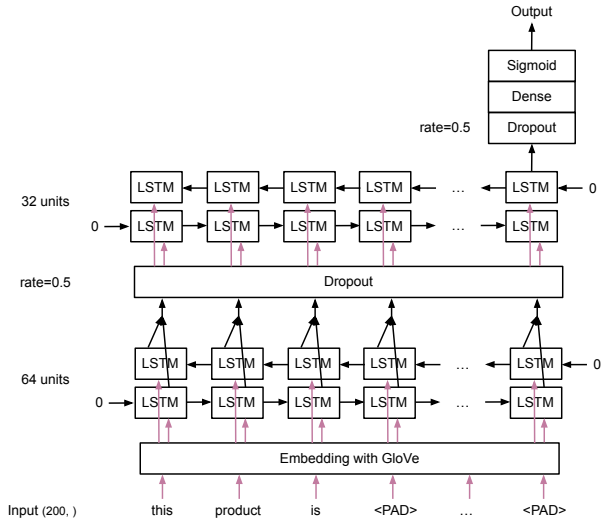


Figure 2: Architecture of the LSTM network.

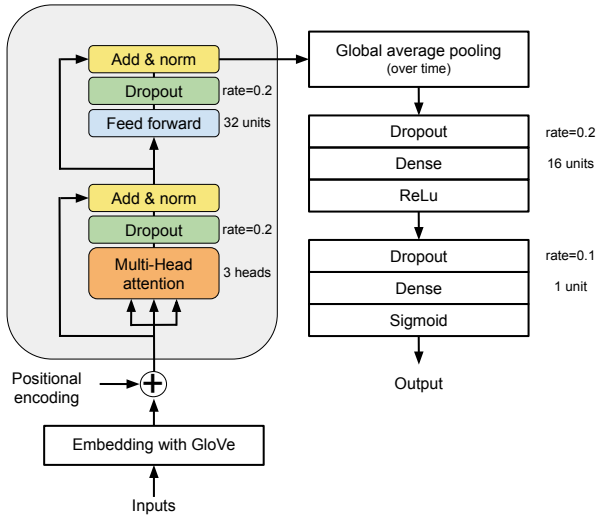
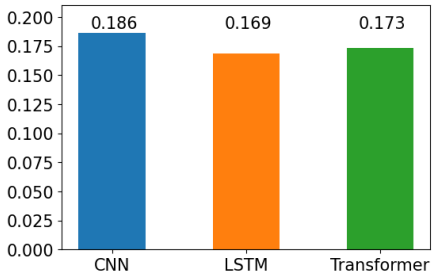
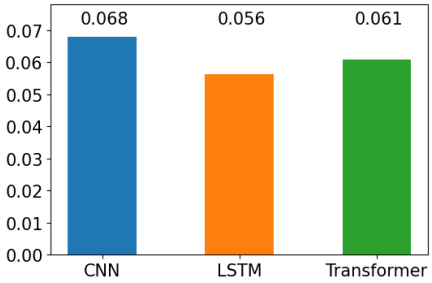


Figure 3: Architecture of the transformer based network.

The architecture of the transformer based network is shown in Figure 3, together with the model's hyperparameters. It uses a transformer block (gray in Figure 3) as proposed in [15], followed by a global average pooling (over time), followed by two "dropout→dense→activation" parts, where the last part uses a sigmoid activation function to generate the output.



(a) Mean absolute error of the four methods when evaluated on the test set.



(b) Mean squared error of the four methods when evaluated on the test set.

Figure 4: Mean squared error (a) and mean absolute error (b) for the four methods when evaluated on the test set.

4.3 Unsupervised method

Our analysis in this project as well as past studies on Amazon review data has shown that it can be difficult to make accurate predictions using supervised learning methods with helpfulness labels. Discussions mention bias in the helpfulness label as the reason behind this poor accuracy of supervised prediction methods.

The helpfulness votes data has several limitations due to inherent characteristics of the voting process. There are a number of reasons: 1) not all reviews are seen and voted on; 2) Number of helpfulness votes is correlated with period of the product being sold on the platform; 3) reviews receiving helpfulness votes would tend to gather more vote due to the snowball effect.

We therefore propose an unsupervised approach to predict helpfulness from the content of the reviews. The summary of our methods is as follows: we use the pre-trained BERT language model to extract keywords from all reviews for each product id and then find the cosine distance of words in the review with the keywords associated with the product's id. We use word embedding provided by BERT language model and take the average distance of all words.

This method predicts higher helpfulness for reviews that have a higher ratio of relevant words to the product associated keywords, normalized to the length of the review. Shorter, more information dense reviews then get higher helpfulness predictions.

5 Experiments/Results/Discussion

Figure 4 shows the mean absolute error and the mean squared error of the four methods when evaluated on the test set. The mean absolute error has been both the loss function when training the models, as well as the primary metric for comparison during development. We see that the LSTM performs the best, followed closely by the transformer. The unsupervised method is hard to compare to the supervised ones since it ranks reviews of a product among themselves, and not on a scale of 0 to 1.

Figure 5 shows examples from the test set and the models' corresponding predictions of helpfulness. The models might have been successful in the example in Figure 5a because the review is moderately long (86 words) and has high information density. They may have also been successful in the example in Figure 5c because the review is rather short, and therefore probably not that helpful. The example in Figure 5e is clearly helpful to a human but it might be hard for the models to deduce that because they might not be able to capture that this kind of information about an adapter can be very helpful.

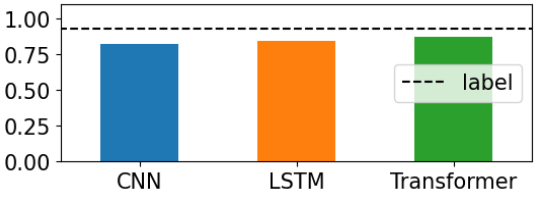
Since we tried multiple different architectures, there are many hyperparameters involved. All model-specific hyperparameters are shown in Figure 1, Figure 2, and Figure 3.

General hyperparameters and hyperparameters related to training are discussed briefly in Table 1.

When training both the LSTM, we noticed signs of overfitting early on. To combat this we first increased the dropout rates in the LSTM up to 0.5 in both layers but it did not seem to be enough. We did not want to increase it even further so we instead introduces L2 regularization in the LSTM layers as well, which mitigated the overfitting problem better.

pros its small , does not need batteries , holds lots of songs , sound is great , can make folders to organize player long battery life cons manual not to detail , had to call tech support to help get music on player because the manual does not tell u once that was done it was very easy to get music on my player , u may want better head phones so far am loving my new product very pleased and the price was great

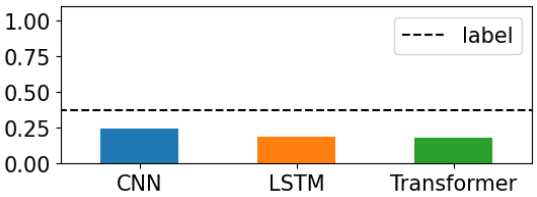
(a) A helpful example from the test dataset, which all supervised models were able to predict the helpfulness of fairly accurately.



(b) Predicted helpfulness of the review in Figure 5a by the four models.

not enough bass

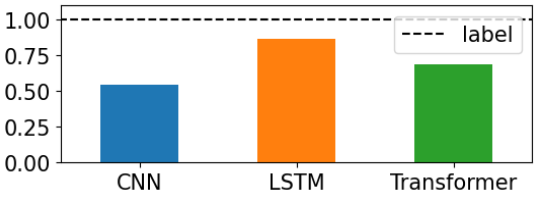
(c) An unhelpful example from the test dataset, which all supervised models were able to predict the helpfulness of fairly accurately.



(d) Predicted helpfulness of the review in Figure 5c by the four models.

this adapter does not have the proper plug to use with the acer iconia tab a100 do not waste your money

(e) A very helpful example from the test dataset, which no model was able to predict the helpfulness of very accurately.



(f) Predicted helpfulness of the review in Figure 5e by the four models.

Figure 5: Three examples from the test set (a), (c), (e), and the models' corresponding predictions of helpfulness (b), (d), (f) respectively.

Table 1: Hyperparameter values and a brief motivation behind the choices.

Hyperparameter	Value	Motivation
max review length	200	It was a balance between keeping the amount of model parameters small while not losing too much information from the review.
embedding dimension	100	Also a balance between keeping the amount of parameters small while still having a good enough representation of each word.
minimum total votes	5	See section 3
optimizer	Adam	What most people seem to use as their default choice.
learning rates	5e-2 / 1e-3 / 1e-3 for CNN / LSTM / Tr.	Started with the default learning rate of 1e-3 and tested a few values in [1e-4, 1e-2] (with logarithmic spacing) for each model and found that these values gave the best results. We did not experiment with learning rate scheduling.
number of epochs	30 / 60 / 50 for CNN / LSTM / Tr.	Stopped when there was no further noticeable improvement to dev. loss. Chosen after the learning rates were chosen.
batch size	32	Increasing it to 64 made each update step too slow and lowering it more seemed to give comparable performance, until a point where updates became too noisy.

6 Conclusion/Future Work

The problem of predicting helpfulness turned out to be a hard problem to solve. While the LSTM-based method did perform the best, whether an average error of 0.169 is truly a solution is debatable. For future work, we would estimate Bayes' error to better understand what error rate to realistically aim for. With more time or team members, we would have tried using the predictions from the unsupervised method as labels for the supervised ones and to further develop the transformer based method which is a promising technique that deserves more attention.

7 Contributions

Edvin designed the architectures- and wrote the code for the three supervised methods. He also trained them and performed (hyper)parameter selection.

Negar designed architecture of the neural network for the supervised models and coded the CNN model. She also designed and coded the unsupervised learning method.

Artem focused on related literature reviews, research on natural language processing methods for unsupervised learning and on existing algorithms, editing, analysis, and the final presentation.

The source code for the project can be found on this Github link.

References

[1] Amazon customer reviews dataset. `https://s3.amazonaws.com/amazon-reviews-pds/readme.html`. Accessed: 2023-06-09.

[2] Huggingface transformers. `https://huggingface.co/docs/transformers/index`. Accessed: 2023-06-09.

[3] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[4] François Chollet et al. Keras. `https://keras.io`, 2015.

[5] Miao Fan, Chao Feng, Lin Guo, Mingming Sun, and Ping Li. Product-aware helpfulness prediction of online reviews. In *The World Wide Web Conference*, WWW '19, page 2715–2721, New York, NY, USA, 2019. Association for Computing Machinery.

[6] Maarten Grootendorst. Keybert: Minimal keyword extraction with bert., 2020.

[7] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020.

[8] Albert H Huang, Kuanchin Chen, David C Yen, and Trang P Tran. A study of factors that contribute to online review helpfulness. *Computers in Human Behavior*, 48:17–27, 2015.

[9] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.

[10] María Olmedilla, M. Rocío Martínez-Torres, and Sergio Toral. Prediction and modelling online reviews helpfulness using 1d convolutional neural networks. *Expert Systems with Applications*, 198:116787, 2022.

[11] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[12] 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*, 12:2825–2830, 2011.

[13] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[14] Xinyu Sun, Maoxin Han, and Juan Feng. Helpfulness of online reviews: Examining review informativeness and classification thresholds by search products and experience products. *Decis. Support Syst.*, 124(C), sep 2019.

[15] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[16] Deliang Yang and Nan Du. Predict of helpfulness of amazon customer review using deep learning. 2019.