Distinguishing Helpful User Reviews

An NLP classification analysis of book reviews by Andrew Dettor

Introduction

Books can take a long time to read, so finding the right book is important. People often look towards reviews to see if the book is right for them and worth their time. On many websites such as Goodreads.com, people can upvote or downvote people's reviews so others are more likely to see them. There are often thousands of reviews, but only a select few receive any upvotes at all. Why is this? What makes these reviews stand out?.

According to [10], higher quality reviews make potential customers more likely to buy the product and more likely to visit the website again. Maybe high quality book reviews contain useful information or thoughts on the book that aren't present in the book's description, giving a potential reader a better idea of whether they would enjoy reading it or not. Authors and publishers can potentially put this information in their descriptions to attract more readers. People who don't like the book based on the more informative description won't read it, and then won't leave a negative review. It's a win-win.

Related Work

Traditional Methods

Most related work on the topic of review helpfulness focuses on machine learning with feature engineering rather than deep learning. Features are extracted from the reviews themselves, but also the meta data of the review or product. According to [17], features from the reviews, "represent structural, lexical, syntactic and semantic information of the textual content." These include lemmatization, stemming, length of review, word segmentation, part of speech tagging, tokenization, some subjectivity or sentiment features, linguistic style, emotional qualities, informativeness, readability, time of review, quality of the reviewer, unigrams, bigrams, tf-idf (Term Frequency — Inverse Document Frequency) scores, the list goes on [4, 15, 2, 9, 1]. These papers use a lexicon, which is a "dictionary of sentiment words and phrases with their polarities and strengths" [15]. Sometimes they're premade from experts, but sometimes LDA (Latent Dirichlet Allocation) is used to create a lexicon without supervision [6, 15]. As you can tell, preprocessing data for this problem can be, "time-consuming, labor-intensive, and inflexible," says [8].

Using Deep Learning

To make preprocessing require less expert knowledge, and make it quicker and easier, many have proposed using deep learning. The allure of deep learning is that it handles feature engineering and selection all on its own [8]. Deep learning models are also robust to noise and they can deal with extremely large datasets [8]. They often use embedding matrices as the vocabulary, which are much more succinct and easier to deal with than a lexicon or “bag of words” [18, 16]. However, performance is mixed on many different classification problems. Papers like [3, 2], say they perform as well as traditional NLP methods, [7, 11, 14], say they perform better than traditional NLP methods, but [15] says their performance is just okay. Most models used are GRU’s, LSTM’s, or Convolutional Neural Networks [7, 2, 14, 11]. However, a deep learning model’s decisions are notoriously difficult to understand and interpret. Oftentimes an inordinate amount of data and compute resources are required to train it [11]. This makes it hard for someone to learn from the model; there are no “most important” features that determine if a review is helpful or not. A human can’t learn tips to write better reviews or write a better description on their product’s page using a deep learning model, but they can see the model’s prediction on their work.

What makes a review helpful

Speaking of most important features, the literature has pointed to several for this type of problem. Looking at [9, 17, 5, 13, 12], reviewer expertise, timeliness of the review, and linguistic features like subjectivity and writing style are the most important features. This is an issue for deep learning models, because if only the text is their input, then they’re missing out on key information. Nevertheless, deep learning models like in [14] still seem to have better or equivalent performance to traditional methods with only the text.

How helpfulness is defined

Another issue is how “helpfulness” is defined. It’s subjective, but many papers have proposed a way of assigning helpfulness labels to their data. On many websites, reviews have a way to react to a review with a like or something else. Amazon even tells you the proportion of people who liked a review. A proportion above 0.5 can be “helpful”, and below 0.5 can be labelled as “unhelpful” [14]. In [14] they looked at the distribution of the proportion of likes on reviews and saw that 0.5 was a natural cutoff point. Paper [15] suggests turning this into a regression problem on the proportion or number of likes. Paper [5] put the classification cutoff at 0.6, and also tried regression and said performance was comparable. The 0.6 cutoff was found by comparing the error rate of several different cutoff values. Paper [3] used a cutoff of 0 likes, while others removed reviews with too few likes [15].

Methodology

Data Collection

Book reviews were collected from Goodreads.com. On Goodreads.com, there are many user created and user maintained lists of books, such as “Best for Book Clubs” or “Best Books of 20th Century” [20]. I focused on trying to get a wide range of average ratings of books, so I could get many negative and positive reviews, books with few ratings and many ratings, books whose reviews had few or no likes to books with reviews with thousands of likes. These were the exact lists I got reviews from: Books With a Goodreads Average Rating of 4.5 and above and With At Least 100 Ratings, Books With a Goodreads Average Rating of 4.3 and Above, Books With a Goodreads Average Rating of 4.2 and Above, Books With a Goodreads Average Rating of 4.0 and above and With At Least 30,000 Ratings, Favorite Poorly Rated Books, and Worst Rated Books on Goodreads.

Each book list can have upwards of 70 pages, each with 100 books. For each book you can access about 10 pages of reviews with 30 reviews per page. What I did was as follows. I took the books on the first 5 pages of each list and got the reviews of the first page of reviews. All in all, this is about 6 lists * 5 pages * 100 books = 3000 books and then 3000 books * 30 reviews = 90000 reviews total (assuming every book has 30 or more reviews).

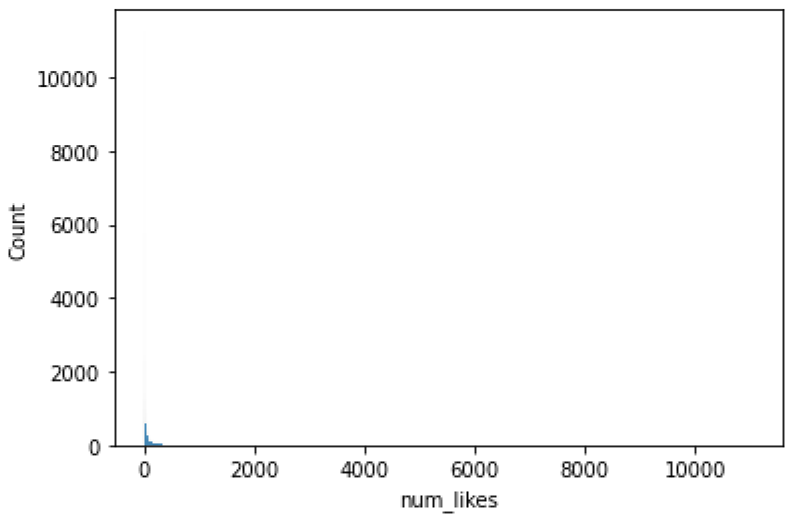
Web scraping was performed by getting the html of each list, getting the html of each book on each list, then getting the html of the reviews on all the books. Pages were accessed with the requests module in Python 3.8 and html information was parsed with BeautifulSoup4.

Data Preprocessing

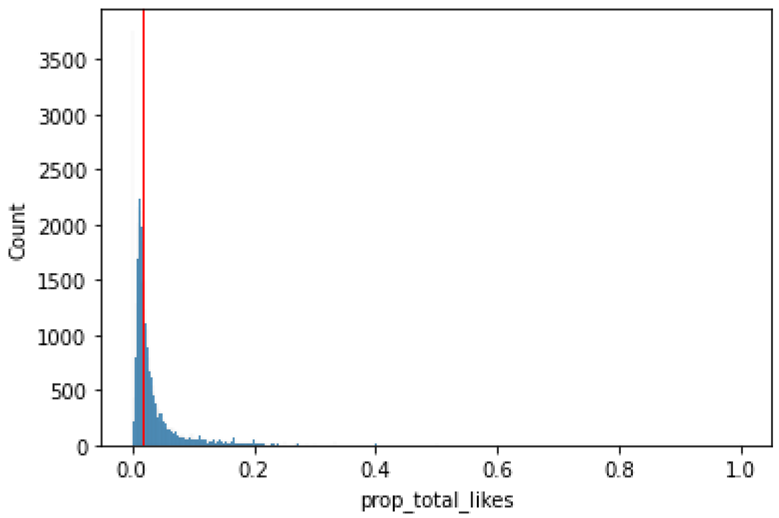
For some reviews, their review text was N/A so I just had to delete those entries. Then while looking at a few reviews I noticed many were not in English. There is no language filter on Goodreads.com so this isn't surprising. My goal was to have more coherence in the dataset so I used the Python library spacy_langdetect to guess the language of each review. After seeing the results I decided to remove all non-english entries, since spacy was quite accurate.

Preprocessing of the text was done with the Python library fastai, which made things quite simple. fastai tokenizes and numericalizes the text and puts it all into an embedding matrix of numbers. I used the default settings, which removed html, removed useless spaces, set all to lowercase, replaced repeated words or letters with a specific token, replaced capital letters with a token, and tokenized words. The embedding matrix had a vocabulary of size 60000 with a minimum frequency of 3. So if a word didn't show up 3 or more times or there were >60000 more frequent words it was replaced with an "unknown" token. This is probably where all the words in other languages would have ended up if I kept them in the dataset.

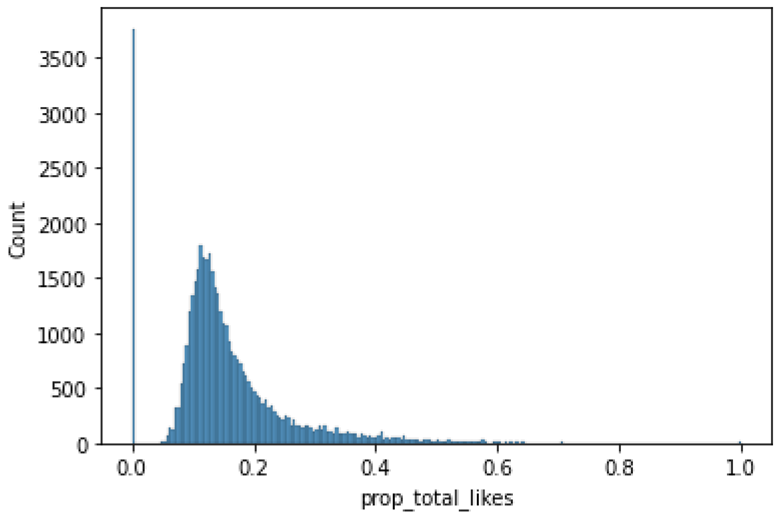
One problem with Goodreads is they don't show you the proportion of people who liked a review, they just show the number of likes. Trying out a strategy like [2] was impossible, then. So I tried several of my own methods of deciding what was helpful and compared their performance. First, I looked at the distribution of likes for all reviews, which was highly skewed.



This was hard to use, so I found the relative proportion of likes each review got based on the total number of likes on reviews of the first page of every book. This would put each book's reviews on a more level playing field and control for more popular books. I used the median as an initial cutoff point because it could maintain a balance of classes (median is the red line).



This performed poorly so I looked at other ways of discretizing this distribution. I started off with transformations. I couldn't use boxcox or log because there were 0 values in the dataset, so I tried a square root transformation.



This way reviews with no likes were separated further from the others. I still needed to discretize this distribution.

In DSC440 fashion, I used equal width histogram binning with 10 brins, Kmeans clustering with 2 clusters (found through the Silhouette metric), splitting on the median, and splitting into 4 quartiles. The idea behind the multi class discretizations is maybe it would give a better description of the distribution, with a higher bin number meaning higher helpfulness.

Note: at this point regression was probably a better approach, but I wanted to see the effect of different definitions of helpfulness to see if any were informative.

Modelling

I used deep learning and transfer learning through the fastai Python library. The main purpose was to find a method to make NLP classification simpler to perform. The text portion of the fastai library was based on paper [7]. A pre-trained language model can be used as a basis for transfer learning. This model can be trained on a large corpus of text. It can then be used to adapt the language model to the language of our domain (in this case, book reviews). Then the new language model can be modified to perform classification. All of this is provided in fastai, which uses AWD_LSTM as a base model. It was trained on wikitext-103, which is a dataset of all wikipedia pages. This method of Universal Language Model Fine-tuning purportedly reduces error by 18-24% on most datasets and requires one-hundred times less data than training by scratch.

Performance/Results

Standard classification performance metrics can be used, such as accuracy, precision, and recall. These metrics can be calculated for the multiclass cases, too.

Experiment

Data Collection

82637 reviews were collected from 2746 books. Maybe for some books or reviews there were issues with the web scraper, or maybe some pages weren't filled all the way with books or reviews. Either way, 82637 should be plenty. One problem with

Data Preprocessing

After taking out N/A reviews there were 78932 left, and after removing non english entries there were 42747 reviews left. I removed all non-english entries because the non-english entries I inspected either were clearly not english or they were just a few english words long. There weren't many false positives, either. I then used the fastai library to transform the reviews into an embedding matrix. I used a batch size of 128 where each example was up to 72 words long. 20% of the data was held out as a validation set.

Modelling

To start off, I had to change the AWD_LSTM language model into a book review language model. Language models are trained to try to generate text to predict the next word(s) in a sentence, so they have to understand the language quite deeply. Since the base model already knows English from wikipedia, it will be able to learn the English of book reviews. The final layer of AWD_LSTM was removed and replaced with a randomly initialized linear layer, since we want it to predict for a new problem. This was trained for one epoch on the book review data, while keeping all other layers "frozen", meaning their weights could not be changed. Then the model was unfrozen completely and trained for 5 more epochs, until the validation loss and training loss started to diverge. This resulted in about 31% accuracy, meaning it was able to guess the next word in a book review almost ⅓ of the time.

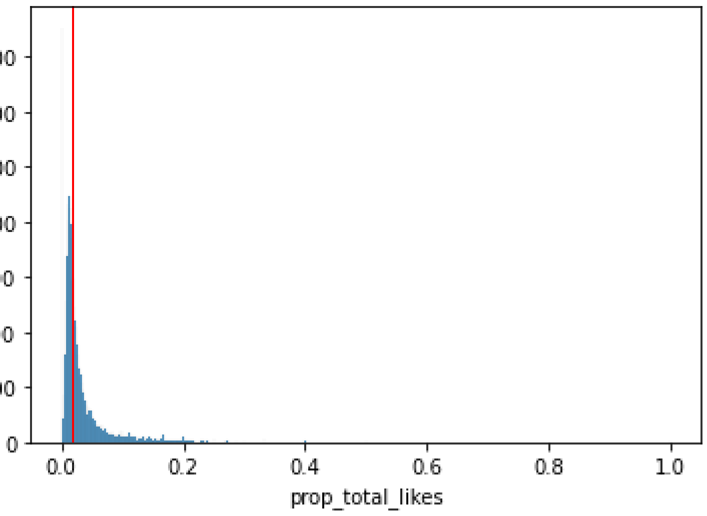
Using the domain-trained language model as a base, I made 5 classifiers corresponding to the different definitions of helpfulness I wanted to test. Classification is done by adding a layer to the end of the model . Each classifier was run for about 5 epochs, since that was when validation loss and training loss started to diverge. I'm surprised it started overfitting so quickly considering there was 50% dropout. I followed an example from the fastai documentation for unfreezing; I unfroze the top two layers after the first epoch, the last 3 layers after the second epoch, then unfroze the whole model for the remaining 3 epochs. I also used their example schedules for learning rates after the first epoch; they increase learning rate, decrease it, then increase it again during each epoch [21].

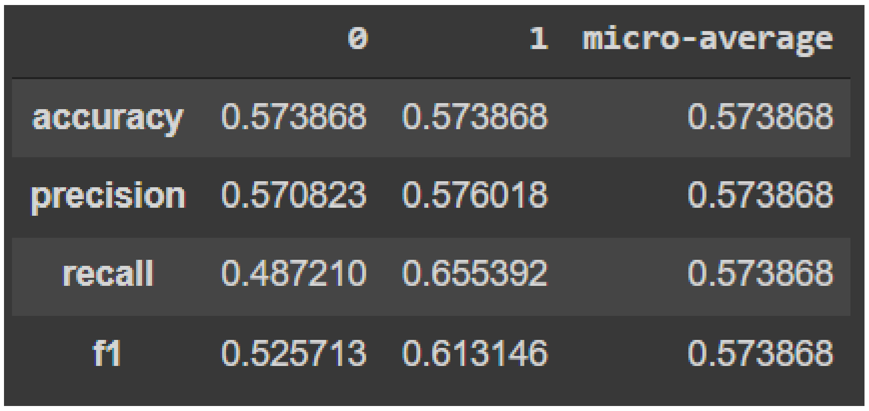
Training these models was quite a hassle. Even with a V100 GPU, each epoch can take upwards of an hour and a half. What's odd is the fastai documentation has a sentiment analysis example with a similar number of training examples, and those epochs were about 12 minutes. Take into account bug fixing and the odd way fastai handles performance metrics for different types of problems, it took way longer than expected. The classification epochs were much faster, at around 2 minutes each, but having to build the DataBlock and dataloaders for each classification model every time I started working on the project took a long time, too. The idea was to see if deep learning made NLP easier, but I guess every method has its own issues.

Performance/Results

Below are the graphs with class boundaries and performance for those classes for each definition of “helpfulness” I proposed.

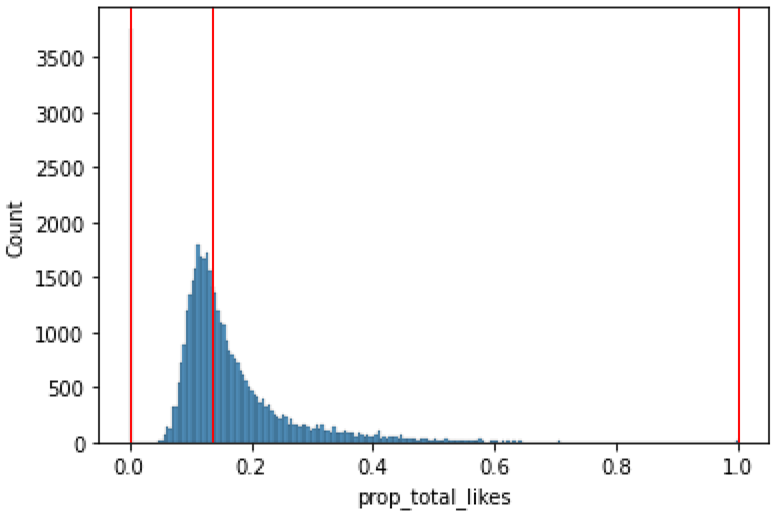
Definition 1: Median of prop_total_likes

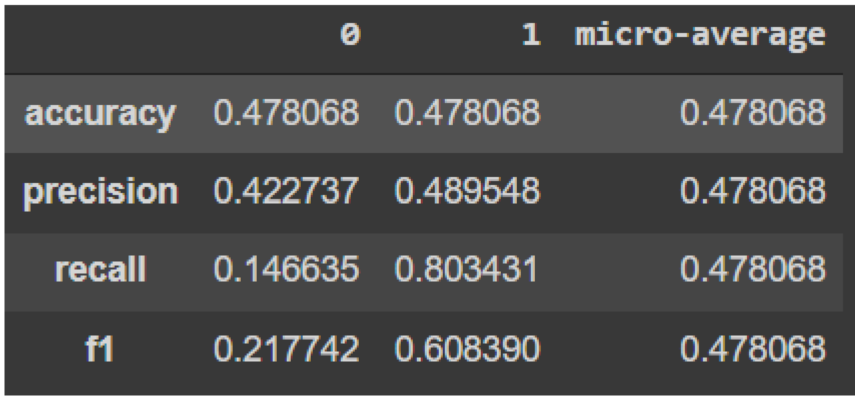




	0	1	micro-average
accuracy	0.573868	0.573868	0.573868
precision	0.570823	0.576018	0.573868
recall	0.487210	0.655392	0.573868
f1	0.525713	0.613146	0.573868

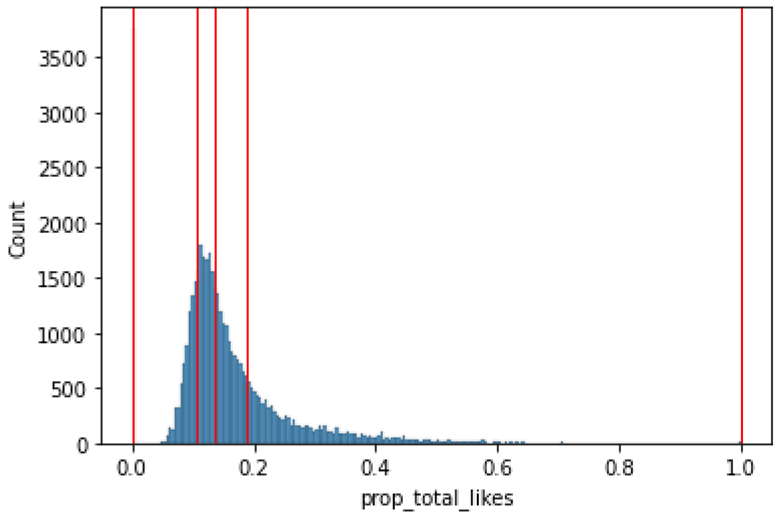
Definition 2: Median of sqrt(prop_total_likes)

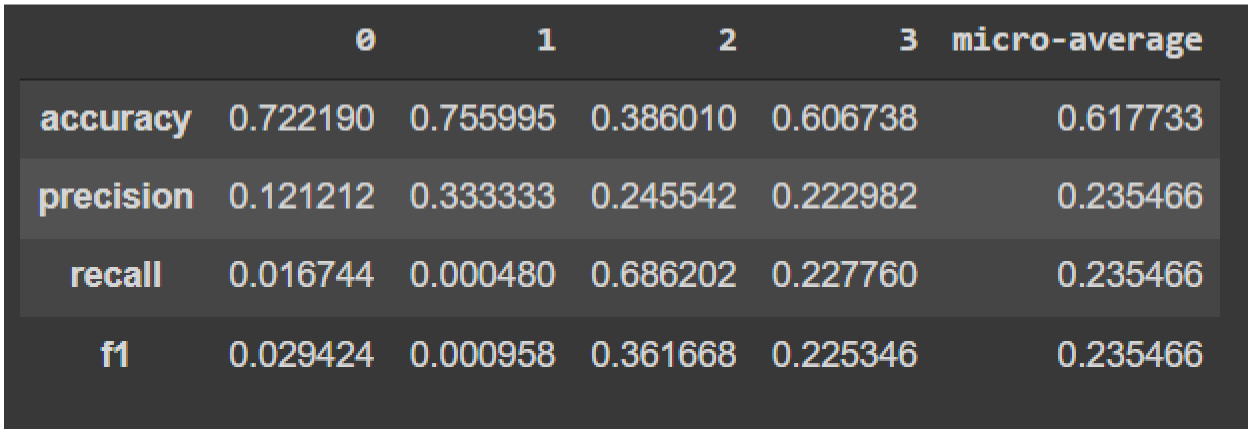




	0	1	micro-average
accuracy	0.478068	0.478068	0.478068
precision	0.422737	0.489548	0.478068
recall	0.146635	0.803431	0.478068
f1	0.217742	0.608390	0.478068

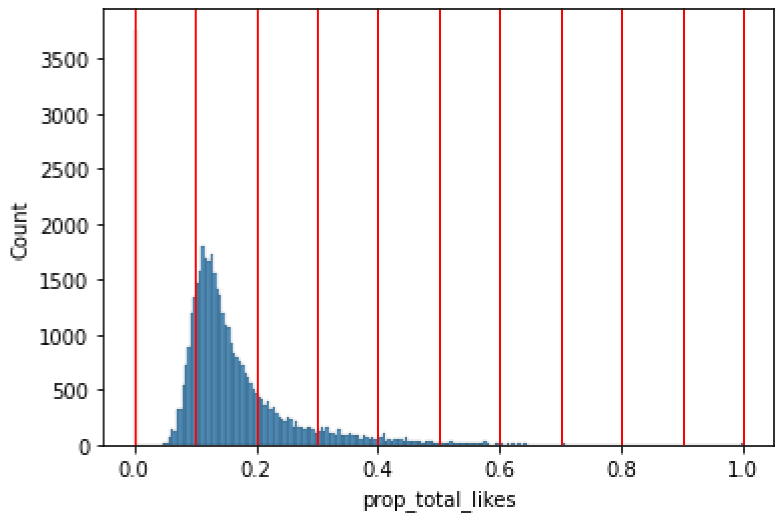
Definition 3: 4 quartiles of sqrt(prop_total_likes)

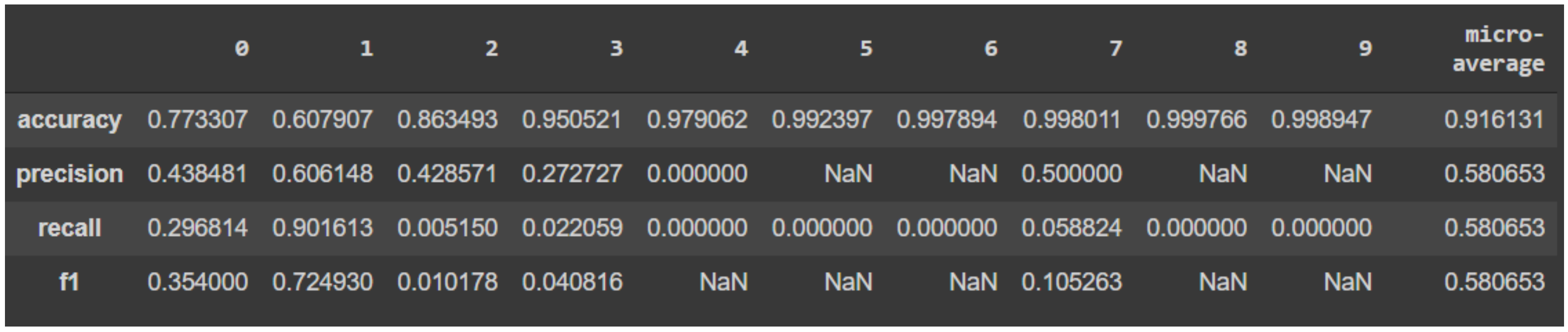




	0	1	2	3	micro-average
accuracy	0.722190	0.755995	0.386010	0.606738	0.617733
precision	0.121212	0.333333	0.245542	0.222982	0.235466
recall	0.016744	0.000480	0.686202	0.227760	0.235466
f1	0.029424	0.000958	0.361668	0.225346	0.235466

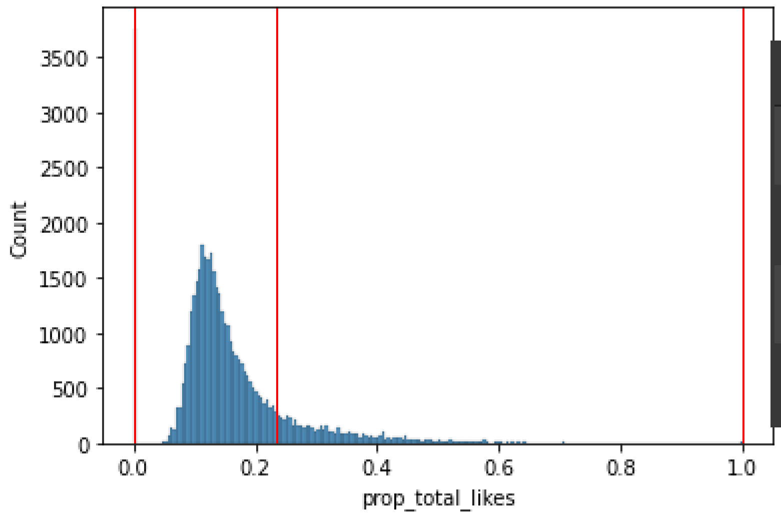
Definition 4: 10 histogram bins of sqrt(prop_total_likes)

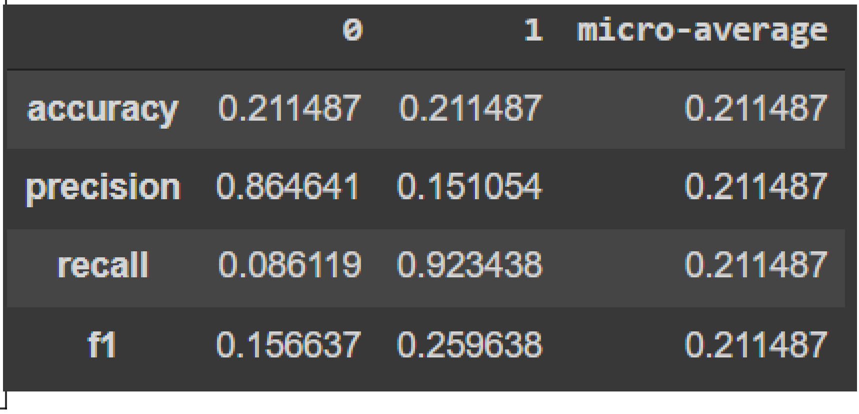




	0	1	2	3	4	5	6	7	8	9	micro-average
accuracy	0.773307	0.607907	0.863493	0.950521	0.979062	0.992397	0.997894	0.998011	0.999766	0.998947	0.916131
precision	0.438481	0.606148	0.428571	0.272727	0.000000	NaN	NaN	0.500000	NaN	NaN	0.580653
recall	0.296814	0.901613	0.005150	0.022059	0.000000	0.000000	0.000000	0.058824	0.000000	0.000000	0.580653
f1	0.354000	0.724930	0.010178	0.040816	NaN	NaN	NaN	0.105263	NaN	NaN	0.580653

Definition 5: 2 bins from KMeans clustering of sqrt(prop_total_likes)





	0	1	micro-average
accuracy	0.211487	0.211487	0.211487
precision	0.864641	0.151054	0.211487
recall	0.086119	0.923438	0.211487
f1	0.156637	0.259638	0.211487

Discussion of Results

On the whole, horrendous performance. Definition 1 was what I started with, which had 7% better accuracy than random. Definition 2 had high recall for the more helpful class but that's about it. Definition 3 had decent accuracy for the two least helpful classes, but miserable precision and recall. Definition 4 had seemingly good accuracy for everything until you realise the classes were extremely imbalanced. Definition 5 had high precision for the less helpful class and high recall for the more helpful class.

Conclusion

Key Findings and Ways to improve

Clearly, information other than just the text is needed for this problem. Looking deeper at paper [2], their feature space was much more comprehensive, with semantic, sentiment, content-based, and metadata related features. They also used convolutions combined with a GRU layer (similar to LSTM) instead of just an LSTM, and they didn't start with a language model. That was probably the better approach for this dataset. Maybe using a CNN would have worked well, like in [14].

The language model seemed to perform well, correctly predicting the next word in a sentence 31% of the time. This may be due to the predictability of English, though. The classification models had trouble figuring out helpfulness from the text alone, when to me it's quite obvious when a review is helpful. I wouldn't think the timeliness of the review or reviewer expertise would make the review text itself read any differently, but clearly those signals are needed for a machine to perform well on this problem. Speaking of signals, I clearly had trouble deciding on the definition of helpfulness, too. None of them seemed to be particularly useful, except for the 4 quartile one due to higher accuracy. This project unfortunately has a null result.

Bibliography (ACM)

[1] Kalpana Algotar and Ajay Bansal. 2018. Detecting Truthful and Useful Consumer Reviews for Products using Opinion Mining. EMSASW@ ESWC (June 2018). Retrieved from http://ceur-ws.org/Vol-2111/paper8.pdf

[2] Mohammad Ehsan Basiri and Shirin Habibi. 2020. Review Helpfulness Prediction Using Convolutional Neural Networks and Gated Recurrent Units. In 2020 6th International Conference on Web Research (ICWR), 191–196. DOI:https://doi.org/10.1109/ICWR49608.2020.9122297

[3] Enrique Bigne, Carla Ruiz, Antonio Cuenca, Carmen Perez, and Aitor Garcia. 2021. What drives the helpfulness of online reviews? A deep learning study of sentiment analysis, pictorial

content and reviewer expertise for mature destinations. Journal of Destination Marketing & Management 20, (June 2021), 100570. DOI:https://doi.org/10.1016/j.jdmm.2021.100570

[4]Yung-Chun Chang, Chih-Hao Ku, and Chien-Hung Chen. 2020. Using deep learning and visual analytics to explore hotel reviews and responses. Tourism Management 80, (October 2020), 104129. DOI:https://doi.org/10.1016/j.tourman.2020.104129

[5]Anindya Ghose and Panagiotis G. Ipeirotis. 2011. Estimating the Helpfulness and Economic Impact of Product Reviews: Mining Text and Reviewer Characteristics. IEEE Transactions on Knowledge and Data Engineering 23, 10 (October 2011), 1498–1512. DOI:https://doi.org/10.1109/TKDE.2010.188

[6] Leah M. Hamilton and Jacob Lahne. 2020. Fast and automated sensory analysis: Using natural language processing for descriptive lexicon development. Food Quality and Preference 83, (July 2020), 103926. DOI:https://doi.org/10.1016/j.foodqual.2020.103926

[7] Jeremy Howard and Sebastian Ruder. 2018. Universal Language Model Fine-tuning for Text Classification. arXiv:1801.06146 [cs, stat] (May 2018). Retrieved December 11, 2021 from http://arxiv.org/abs/1801.06146

[8] Christian Janiesch, Patrick Zschech, and Kai Heinrich. 2021. Machine learning and deep learning. Electron Markets 31, 3 (September 2021), 685–695. DOI:https://doi.org/10.1007/s12525-021-00475-2

[9] Soo-Min Kim, Patrick Pantel, Tim Chklovski, and Marco Pennacchiotti. 2006. Automatically assessing review helpfulness. Proceedings of the 2006 Conference on empirical methods in natural language processing (July 2006). Retrieved from https://aclanthology.org/W06-1650.pdf

[10] Eun-Ju Lee and Soo Yun Shin. 2014. When do consumers buy online product reviews? Effects of review quality, product type, and reviewer's photo. Computers in Human Behavior 31, (February 2014), 356–366. DOI:https://doi.org/10.1016/j.chb.2013.10.050

[11] Hang Li. 2017. Deep learning for natural language processing: advantages and challenges. National Science Review (2017).

[12] Yang Liu, Xiangji Huang, Aijun An, and Xiaohui Yu. 2008. HelpMeter: A Nonlinear Model for Predicting the Helpfulness of Online Reviews. In 2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology, 793–796. DOI:https://doi.org/10.1109/WIIAT.2008.299

[13] Yang Liu, Xiangji Huang, Aijun An, and Xiaohui Yu. 2008. Modeling and Predicting the Helpfulness of Online Reviews. In 2008 Eighth IEEE International Conference on Data Mining, 443–452. DOI:https://doi.org/10.1109/ICDM.2008.94

[14] Xianshan Qu, Xiaopeng Li, and John R. Rose. 2018. Review Helpfulness Assessment based on Convolutional Neural Network. arXiv:1808.09016 [cs] (August 2018). Retrieved December 11, 2021 from http://arxiv.org/abs/1808.09016

[15] Shiliang Sun, Chen Luo, and Junyu Chen. 2017. A review of natural language processing techniques for opinion mining systems. Information Fusion 36, (July 2017), 10–25. DOI:https://doi.org/10.1016/j.inffus.2016.10.004

[16] Antoine J.-P. Tixier. 2018. Notes on deep learning for nlp. arXiv preprint arXiv:1808.09772 (2018).

[17] Wenting Xiong and Diane Litman. 2011. Understanding Differences in Perceived Peer-Review Helpfulness using Natural Language Processing. Proceedings of the Sixth Workshop on Innovative Use of NLP for Building Educational Applications (June 2011). Retrieved from https://aclanthology.org/W11-1402.pdf

[18] Shirong Xu, Ben Dai, and Junhui Wang. 2021. Sentiment analysis with covariate-assisted word embeddings. Electronic Journal of Statistics 15, 1 (January 2021), 3015–3039. DOI:https://doi.org/10.1214/21-EJS1854

[19] View of RevRank: A Fully Unsupervised Algorithm for Selecting the Most Helpful Book Reviews. Retrieved December 11, 2021 from https://ojs.aaai.org/index.php/ICWSM/article/view/13945/13794

[20] Goodreads. Goodreads. Retrieved December 11, 2021 from https://www.goodreads.com/

[21] Transfer learning in text | fastai. Retrieved December 11, 2021 from https://docs.fast.ai/tutorial.text.html