Fine-grained Sentiment Analysis of Amazon Product Reviews

Helena Huang

yhuang77@stanford.edu

November 20, 2020

Abstract

space for abstract

I. Introduction

Amazon product rating is useful for both customers and sellers. Customers use it to compare products and decide which product to purchase, while sellers use it to improve the quality of their products. However, not every product review on the internet has an associated rating. For example, people might discuss a product on Twitter or Instagram, where they won't leave an explicit rating. Or people might be selling things on YouTube, but the comments section doesn't provide a rating system. It would be convenient if we could have an algorithm to rate every comment on a 1 to 5 scale based on its positivity.

Motivated by the above observation, this project uses Amazon review texts for sports and outdoors products to predict the rating of a costumer on a product. It falls into the category of fine-grained sentiment analysis, where each review text is classified as very positive, positive, neutral, negative, or very negative, according to the 5-star rating. More concretely, the input to our algorithm is a product review in English text. We then output an integer from 1 to 5 corresponding to the predicted 5-star rating.

II. Related Work

Methods for sentiment analysis can be categorized into lexical approaches and machine learning approaches. SentiStrength [2] is a lexical approach specialized in analyzing short texts. It assigns scores for both positivity and negativity for each word in the lexicon, utilizing the psychological finding that humans can simultaneously possess both positive and negative emotions. SentiStrength assigns to each sentence a positive score according to the most positive word in the sentence, and a negative score according to the most negative word in the sentence. Building on SentiStrength, Guzman and Maalej created a tool to analyze app reviews [3]. Instead of predicting an overall rating for each review like our project does, they first classify each sentence in the review texts according to the app feature described in the sentence, then generate ratings for each feature. The benefit of identifying different app features and generating ratings for specific features is that developers can have a clearer picture on how they can improve their apps. Zirn and his research group also used lexical approach in performing sentiment analysis on product reviews [4]. Besides considering positive and negative emotions simultaneously like the previous approaches, they also included sentence structures in their algorithm to capture the logical and hierarchical components within a sentence.

On the other hand, people have also developed many machine learning algorithms to perform the task of sentiment analysis. Ye, Ziqiong and Rob compared three different machine learning approaches (Naive Bayes, SVM, and character-based N-gram model) to the classification of online reviews [5]. They found that SVM and N-gram model outperformed the Naive Bayes model on classifying the sentiment of travel blogs. Their project is different from ours in that they only have two coerse categories for sentiment analysis: positive and negative. Our project performs a fine-grained sentiment analysis where each review can be classified into 5 classes: very positive, positive, neutral, negative, or very negative. Their SVM approach is also slightly different from ours as they use the frequency of each word in the review as their features, whereas we use a binary number to indicate if a word is present in the review text.

III. Data Set

The dataset we are using is from Amazon Review Data (https://nijianmo.github.io/amazon/) [1]. Specifically, we are using a dense subset of the Amazon Review Data in the sports and outdoors category, with about 3 million reviews in total. We divided our dataset according to the review time:

- Training set (80%): from 2000-09-14 17:00:00 to 2017-03-27 17:00:00

- Validation set (10%): from 2017-03-27 17:00:00 to 2017-10-17 17:00:00
- Test set (10%): from 2017-10-17 17:00:00 to 2018-10-03 17:00:00

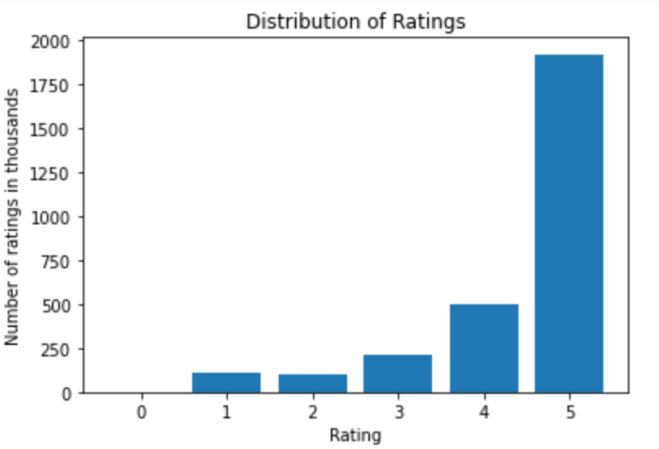

Figure 1: ***Distribution of Ratings.***

The dataset is skewed, as most of the reviews have rating 5 stars. The distribution of the ratings in our training set is shown in Figure 1.

The original dataset contains many features including the overall rating, review time, reviewer ID, etc. We are only using the review text and overall rating for our project. We preprocessed the review text by

- replacing all images with the word "image"
- replacing all dollar signs with the word "dollar"
- replacing all numbers with the word "number"
- removing all non-alphabetical characters
- converting all characters to lower case

IV. Methods

i. Linear Regression

A linear regression model minimizes the loss function

$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}\left(\theta^T x^{(i)} - y^{(i)}\right)^2 + \lambda\sum_{j=1}^{n}\theta_j^2$$

where $x^{(i)}$ is the feature vector for each training example, $y^{(i)}$ is the corresponding rating, and λ is the regularization constant.

In generating predictions for our linear model, we rounded the output of our model to the nearest integer, classified predictions below 1 as a 1 star rating, and predictions above 5 as a 5 star rating.

ii. Word Vectors

Word vectors provide more information about the meaning of each word, so it might be a better set of features for our project. A word vector is a vector of numbers representing the meaning of the word. Words with similar meanings like "woman" and "girl" will be close in the feature space in terms of euclidean distance. We could also interpret the addition of word vectors as the addition of meanings. For example, the word vector of "woman" plus the word vector of "king" is close to the word vector of "queen".

For this project, we used the word vectors from Stanford GloVe. The word vectors are pre-trained on Wikipedia 2014 and Gigaword 5. To generate a feature vector for a review example, we map each word in the review text to a word vector, and sum up the word vectors.

iii. Support Vector Machine

Support vector machine is a binary classification algorithm that minimizes the loss function

$$J(\theta) = C\sum_{i=1}^{m}\left[y^{(i)}cost_1(\theta^T x^{(i)}) + (1-y^{(i)})cost_0(\theta^T x^{(i)})\right] + \frac{1}{2}\sum_{j=1}^{n}\theta_j^2$$

where $x^{(i)}$ is the feature vector for each training example, $y^{(i)}$ is a binary label indicating if the example belong to a class, C is the regularization constant, $cost_0$ and $cost_1$ are squared hinge losses shown in Figure 2.

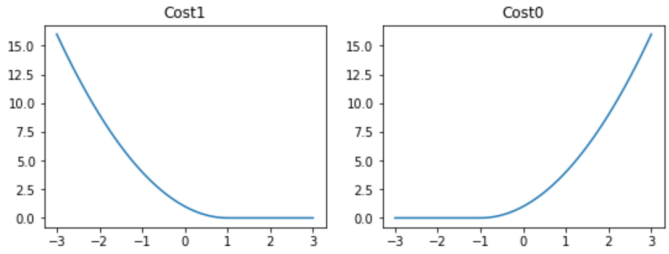

Figure 2: ***Squared Hinge Loss.***

Although the loss function above creates a binary classification problem, we could used the one-versus-all method to generate SVM to multi-class classification. We first treat rating 1 as one class and ratings 2, 3, 4, 5 as another class, and train the model to predict the probability that a review is rating 1. Then, we treat rating 2 as one class, and ratings 1, 3, 4, 5 as another class, and train the model to predict the probability that a review is rating 2. We repeat the process for all 5 ratings. In the end, our model classifies each review as the rating that produces the largest probability.

V. Results

We evaluated our models based on accuracy and average F1 score for the 5 ratings. The accuracy is the number of examples correctly labeled by our algorithm divided by

Truth | Pred	1	2	3	4	5
1	19	43	149	191	12
2	12	23	87	176	16
3	3	30	112	451	78
4	2	9	105	763	649
5	2	10	80	1840	5138

Table 1: ***Confusion Matrix of the Linear Model.***

the total number of examples. For each rating, we also compute

$$\text{precision} = \frac{\text{true positive}}{\text{true positive} + \text{false positive}}$$

$$\text{recall} = \frac{\text{true positive}}{\text{true positive} + \text{false negative}}$$

$$\text{F1 score} = \frac{2 * \text{precision} * \text{recall}}{\text{precision} + \text{recall}}$$

i. Linear Regression

We built a linear regression model with bag of words features. Due to memory constraints, we selected words that appear more than 500 times in the training set as our features. As a result, our feature size is 7400.

Limited by the capacity of our laptop, we only used 5% of our training data for training (100,000 examples), 3% of our validation data for cross validation (10,000 examples), and 3% of our test data for evaluation (10,000 examples).

We plotted the learning curve for our linear model in Figure 3, evaluated with both accuracy and average F1 score for the 5 ratings. It seems that we do have enough data to fit our linear model, but the model is suffering from high bias.

The confusion matrix in Table 1 shows that our model tends to classify 5 star ratings to 4 star. It might be because people have positive and highly positive reviews for 5 star ratings. In order to force highly positive reviews to a rating of 5, we might be supressing the positivity of certain words.

ii. Word Vectors

With the same linear model but pre-trained GloVe word vectors as features, we obtained the learning curve in Figure 4.

Using word vector produces worse performance than the baseline. The model also has very high bias.

iii. Support Vector Machine

The more complex and non-linear SVM with squared hinge loss seems to have reduced our bias. We now treat the problem as a classification problem instead of a regression problem.

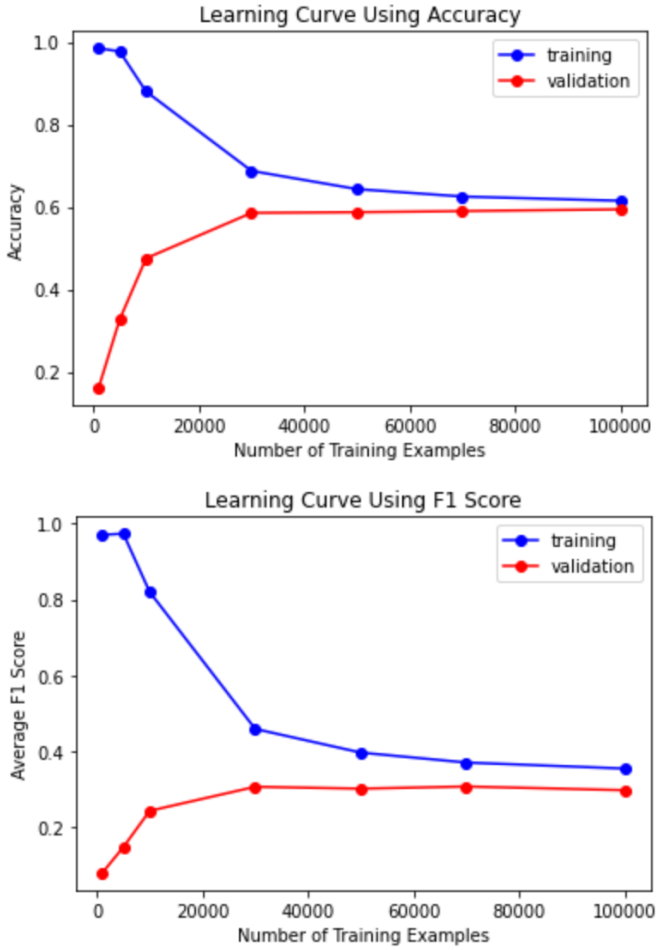

Figure 3: ***Learning Curve of the Linear Model with Bag of Words Features.***

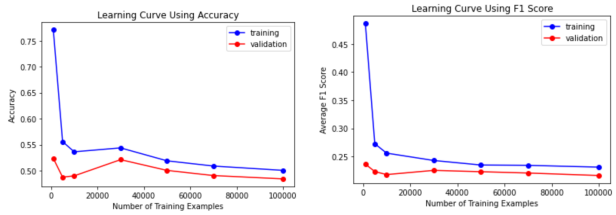

Figure 4: ***Learning Curves of the Linear Model with Word Vector Features.***

The learning curve in Figure 5 shows that our SVM model has high variance. We tried increasing regularization, but it didn't help with performance. We concluded that we need more training data to properly fit the model.

The confusion matrix in Table 2 shows that the SVM model tends to confuse 4 star ratings for 5 star ratings. This could be due to our training set being imbalanced, since more than half of our training examples have 5 star ratings.

We also tried using SVM on the word vector features, it achieved similar accuracy, but the average f1 score is much lower than using bag of words features.

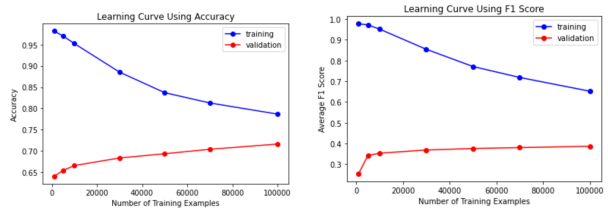

Figure 5: ***Learning Curves of SVM with Word Vector Features.***

Truth | Pred	1	2	3	4	5
1	154	47	30	18	165
2	59	36	38	29	152
3	44	31	107	126	366
4	16	24	98	280	1110
5	33	25	68	260	6684

Table 2: ***Confusion Matrix of the SVM Model.***

iv. Word Stemming

We tried adding word stemming to our preprocessing of the review texts, but it didn't seem to improve the performance of our model.

v. Balanced Training Set

To create a more balanced training set, we randomly selected 80000 examples of each rating from the training set and combined them into a new training set with 400,000 examples. We shuffled the data and used the first 100,000 examples for training. We obtained the learning curve in Figure 6 and confusion matrix in Table 3.

We obtained a higher average F1 score comparing to the SVM trained on a skewed data set, but lower accuracy. The learning curve on F1 score shows that the algorithm is suffering from high variance. The confusion matrix shows that less reviews are misclassified as rating 5.

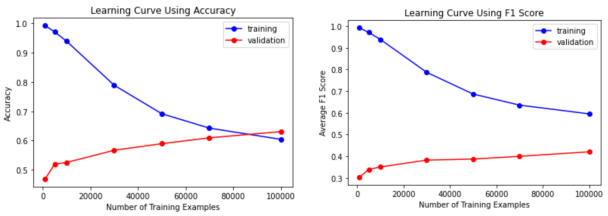

Figure 6: ***Learning Curves of SVM Trained on a Balanced Data Set.***

VI. Conclusion

According to our experiments, the best model for sentiment analysis over Amazon reviews is the SVM with bag of words features without stemming. The performance

Truth | Pred	1	2	3	4	5
1	273	64	31	21	25
2	123	83	59	23	26
3	106	122	228	126	92
4	58	92	271	533	574
5	226	199	343	1052	5250

Table 3: ***Confusion Matrix of the SVM Model Trained on a Balanced Data Set.***

Eval	LR b	LR v	SVM b	SVM v
Acc	0.61	0.50	0.73	0.71
F1	0.30	0.21	0.38	0.22

Eval	stem	balanced
Acc	0.72	0.64
F1	0.32	0.41

Table 4: ***Performance of Models.*** *"LR b" stands for linear regression with bag of words features. "LR v" stands for linear regression with word vectors. "SVM b" stands for SVM with bag of words features. "SVM v" stands for SVM with word vectors. Acc stands for accuracy. F1 stands for average F1 score across all five classes.*

of all models tried in our project is listed in Table 4, as measured on the test set.

People's sentiment across the 5 ratings might not be equally spaced. In addition, people who are satisfied, highly satisfied, or extremely satisfied could all be giving 5 star ratings to a product. Therefore, it is reasonable that SVM classification works better than linear regression.

An Amazon product review often consists of multiple sentences with a mixture of positive and negative comments. The addition of word vectors might not be able to capture the complexity of each review. Therefore, the higher dimensioned bag-of-words approach resulted in better performance.

Due to limitation of computational resources, this project only utilized 5% of all training data and our SVM model is suffering from high variance. If we could obtain access to more computational resources in the future, we would train our model on more data.

VII. Appendix

Link to poster and video:
poster-video

References

[1] Jianmo Ni, Jiacheng Li, Julian McAuley, Justifying recommendations using distantly-labeled reviews and fined-grained aspects, Empirical Methods in Natural Language Processing (EMNLP), 2019

[2] Thelwall, Mike. "The Heart and soul of the web? Sentiment strength detection in the social web with SentiStrength." Cyberemotions. Springer, Cham, 2017. 119-134.

[3] E. Guzman and W. Maalej, "How Do Users Like This Feature? A Fine Grained Sentiment Analysis of App Reviews," 2014 IEEE 22nd International Requirements Engineering Conference (RE), Karlskrona, 2014, pp. 153-162, doi: 10.1109/RE.2014.6912257.

[4] Zirn, Cäcilia, et al. "Fine-grained sentiment analysis with structural features." Proceedings of 5th International Joint Conference on Natural Language Processing. 2011.

[5] Ye, Qiang, Ziqiong Zhang, and Rob Law. "Sentiment classification of online reviews to travel destinations by supervised machine learning approaches." Expert systems with applications 36.3 (2009): 6527-6535.