Amazon reviews sentiment classification by machine learning algorithms

Final project for DATS 6501 Data Science Capstone

Yina Bao

Introduction

In recent years, electronic commerce more and more dominates the market, and there has been a huge increase of interest from brands, companies and data scientists in sentiment analysis and the application to find the business intelligence insight. Nowadays, widely available text data from social media and product reviews can help your business in different sides, such as brand monitoring, customer service, product quality, market research and strategy. Recent study from Zendesk mentioned that 45% of people share negative customer service experience and 30% share positive customer service experience via social media, which shows a high demand for mining the information and extracting the opinion and meaning for further analysis. Sentiment analysis has become one of the trendiest topics of scientific and market research in the field of Natural Language Processing and Machine Learning. During this project, I will explore both deep learning algorithms and traditional machine learning algorithms to classify and make the sentiment analysis of the customer reviews from Amazon based on the text descriptive reviews. Those machine learning methods of classification and sentiment analysis not only helped to match the rating and reviews, but also to detect opinions in different fields, such social media platform, news reports and etc.

In this project, I will classify the reviews' sentiment based on the text input. I used multi-layer perceptron, convolutional neural network, support vector machine, random forest and naive Bayes methods to find out how these algorithms compare to each other and how can we improve the project.

Datasets

The dataset I chose is an Amazon product reviews dataset, which I found from the Kaggle website. The dataset provided on Kaggle is a pre-processed data, which already modified labels to two different classes, positive and negative, and saved in a specific format for fastText training. In order to make a more comprehensive understanding of my project, I choose to use the original data (the link provided on Kaggle website) from Xiang Zhang's Google Drive. The dataset is pre-split into a testing set and training set. The training set contains 3000000 observations, and the testing data contains 650000 observations. The dataset consists of the star ratings, the headline and the descriptive customer reviews.

- Link to Kaggle data page: https://www.kaggle.com/bittlingmayer/amazonreviews
- Link to dataset: https://drive.google.com/drive/folders/0Bz8a_Dbh9Qhbfll6bVpmNUtUcFdjYmF2SEpmZUZUcVNiMUw1TWN6RDV3a0JHT3kxLVhVR2M

Experimental Setup

In this project, I choose the text reviews as my input, and the star rating as the target labels. Since the original data is really large, running each of the models takes a long time. I selected 200,000 data from my training set and 50,000 data from my testing data to create a new smaller sample dataset. In this weighted dataset, each star has 50000 data. In my dataset, there are two descriptive text columns, one is headline and the other one is review which is relatively longer. For the following process, I decide to test each model on both the headline column and the review column, in order to see whether the headline contain the key part of the shorter sentence to predict the classes with less computation process.

Furthermore, I also tried several different models for two versions of labels. The first version is dividing 5 different star ratings to two sentiments, positive at 1 and negative at 0. Positive consists of 4 and 5 star, and negative consists 1 and 2 star. The second version is the original star rating from 1 to 5 as five different classes. The second version is much more complicated, which need more computation and complex model to handle it compared with the sentiment test. I will use the second version to help understanding the project. Improving the model in the second version is considered as further work.

- Data pre-processing

  In order to make the text part as simple as possible, I take the cleaning process first. I created a function named 'clean_text' to remove the website address, social account after '@', digit number, punctuation and other non-alphabetic characters. Lastly, I converted each word to the lower case. Then I saved this cleaned text as a csv file in order to save cleaning time before each model training. After the cleaning part, I removed the stop-words and stemmed the words with the same root by using Snowball stemmer in the nltk package. Additionally, I convert the words to vector of each sentence. For the traditional machine learning models, I used the TfidfVectorizer in sklearn to extract the feature of the text. This method converts a collection of raw documents to a matrix of tf-idf features, which is equivalent to CountVectorizer followed by TfidfTransformer. TfidfVectorizer is a bag of words approach, which means that there is only the word's frequency, and the order of the words is not considered. For deep learning model, I used "texts_to_sequences" after fitting the tokenizer, which builds up words dictionary and convert the words to a vector by the word's index. For convience, I used the "pad_sequences" to make each vector has the same length as input for the neural network model. Finally, using "to_categorical" converts the labels to the one-hot encoding format.

- Model setup

  Firstly, since I have two columns contains text, one is headline and another is text reviews, I need to compare which one is more powerful to make the classification. Therefore, I choose three simple classification methods, SVM, Random Forest and Naïve Bayes, to do the binary (positive and negative) classification by using headline and text reviews separately. From the results, SVM model can reach 0.86 10-fold cross validation f1-score and Naïve Bayes can reach 0.84 by using text reviews. Both of them get o.78 f1-score by using headline. Random forest model gets 0.77 f1-score by using headline and 0.83 f1-score for text reviews. Thus, I will use text reviews for the following model comparison.

F1-score	SVM	Naïve Bayes	Random Forest
Headline	0.7872	0.7893	0.7733
Reviews	0.8645	0.8406	0.8340

Table 1. Classification result of headline and reviews

  In order to make the model comparison, I selected five different models. The first one is Support Vector Machine (you can try gridsearchCV method to get the parameters for the best performance). The second model is a multinomial Naïve Bayes model which suitable for classification of discrete feature. The third is Random Forest with 100 trees. The fourth model is a simple Multi-layer perceptron, with two set of dense layer with ReLU activation function and 20 percent dropout layer, at the end is a dense layer with class number and softmax function. I chose the ‘binary_crossentropy’ loss function and ‘adam’ optimizer. The last model is a CNN model. I defined this CNN model by an embedding layer, a 1D convolutional layer and the max-pooling layer. Followed by two sets of dense and dropout layer and the output layer with softmax function.

Results and discussion

Based on the models I illustrated above, I will compare the performance of those models. Firstly, the multi-layer perceptron model gets 0.79 accuracy, with 15 epochs training, and the result is not changing after I add more epochs. The convolutional neural network model can reach 0.83 testing accuracy after 10 epochs training, and the training accuracy is very close to 1. The support vector machine classifier gets the highest accuracy among those models, which is 0.86 accuracy. Also, the training time is relatively shorter opposed to other models. Multinomial Naïve Bayes model can reach 0.84 accuracy and the random forest model only gets o.83 accuracy averagely. The training time of random forest model is the longest, so it is not that efficient based on the performance. From the plot of the best performance model, SVM, we can find that the ROC curve is really close to the left and upper border, which means that the model is relatively accurate.

	MLP	CNN	SVM	Naïve Bayes	Random Forest
F1-score	0.79	0.83	0.86	0.84	0.83

Table 2. Model comparison

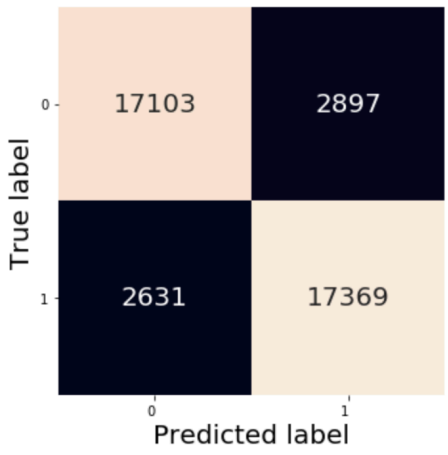

Figure 1. Confusion matrix of SVM model

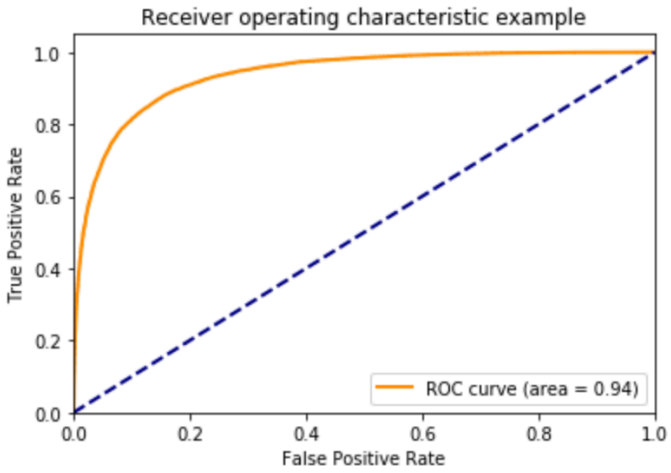

Figure 2. ROC curve of SVM model

Besides, for further exploring, I used the SVM model to take the classification of all five classes. From the confusion matrix plot, we can find that star 1 and star 5 have the highest accuracy, since those two classes are more extreme and distinguishable especially with sentiment. However, the true positive rate of 2, 3, 4 stars is not really high. Based on the plot, the highest misclassification labels of star 2 and star 4 are star 1 and star 5, which means star 2 is relatively similar with star 1 and star 4 is similar with star 5. Star 3 is more ambiguous and neutral based on the confusion matrix, so including star 3 cannot help the model to classify the positive and negative sentiment. If we include star 3 in each positive class or negative class, or random split equal amount into each class, the result will decrease to 0.77. Therefore, for sentiment classification, exclude the neutral will lead to the better results.

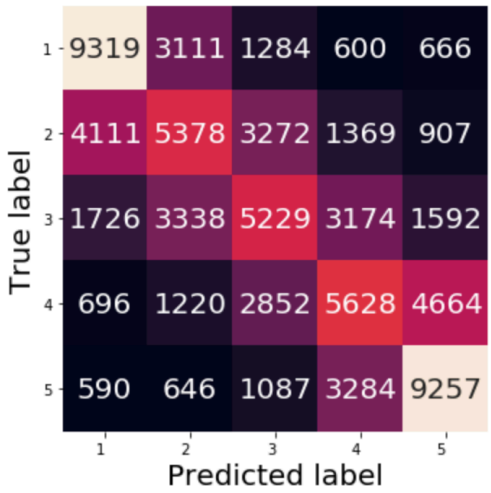

Figure 3. Confusion matrix of SVM model in 5 classes classification

Summary

In conclusion, the SVM model is acceptable for sentiment analysis (f1-score = 0.86), but there is some space for improving the model. Comparing more different hyper-paraments can optimize the model's strength. Moreover, although the MLP and CNN models do not get the best performance in this experiment, there are lots of combinations of different layers which I should

take more trials to optimize the model. Besides, the preprocessing of the data is also really important in natural language processing problems. In this project, actually I found that the default stop-words in nltk package will remove some strong sentiment words, such as ‘not’. Thus, customized stop-words might improve the results. Besides, although I already remove all non-alphabetic characters, there are some reviews were written in alphabetic characters but in other languages, such as Spanish, which is meaningless in English environment. Furthermore, there are so many different kinds of misspelling words, thus a more detailed and powerful cleaning process will help. In addition, based on the observation of the word cloud plot of headline and text reviews, combine those two text parts might led some changes to the performance since the headline contains so many key words of sentiment, like ‘good’, ‘bad’, ‘best’ and etc.

Figure 4. Word cloud of headline

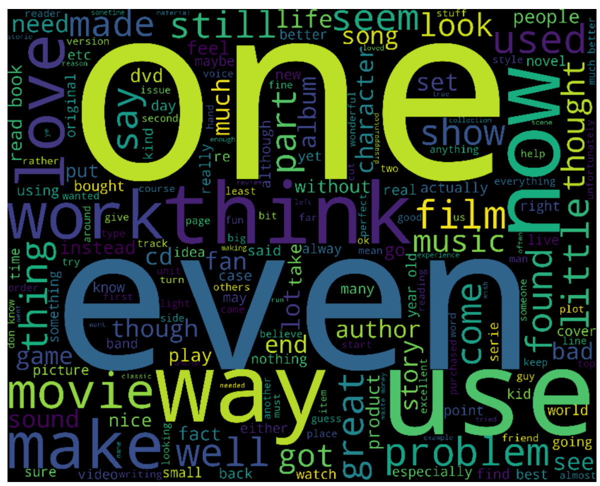

Figure 5. World cloud of reviews

A couple of points to be pursued further research. Based on the online research, using some pre-trained word embedding such as Glove, FastText as transfer learning will improve the data preprocessing and performance compared to the bag of words method.

Reference:

Bansal, Shivam, and Natural Language Processing and Machine Learning. “A Comprehensive Guide to Understand and Implement Text Classification in Python.” *Analytics Vidhya*, 4 Dec. 2018, www.analyticsvidhya.com/blog/2018/04/a-comprehensive-guide-to-understand-and-implement-text-classification-in-python/.

Sentiment analysis. Retrieved September 6, 2018, from https://en.wikipedia.org/wiki/Sentiment_analysis

Hashimoto, S. (n.d.). 7 Benefits of Sentiment Analysis You Can't Overlook. Retrieved from https://blog.insightsatlas.com/7-benefits-of-sentiment-analysis-you-cant-overlook

Hemmah, C. (2013, April 08). What is the Impact of Customer Service on Lifetime Customer Value? Retrieved from https://www.zendesk.com/resources/customer-service-and-lifetime-customer-value/

Maheshwari, Akshat. “Report on Text Classification Using CNN, RNN & HAN – Jatana – Medium.” *Medium.com*, Medium, 17 July 2018, medium.com/jatana/report-on-text-classification-using-cnn-rnn-han-f0e887214d5f.