Amazon Fine Food Review to Rating Prediction

Anupam Kumar
MT20084
anupam20084@iiitd.ac.in

Divyang Patel
MT20114
divyang20114@iiitd.ac.in

Shivam Joshi
MT20130
shivam20130@iiitd.ac.in

Abstract

The Internet has provided a platform for people to express their views, emotions and sentiments towards products, people and life in general. Thus, the Internet is now a vast resource of opinion rich textual data.
Sentiment Analysis focuses on identifying whether a given piece of text is subjective or objective and if it is subjective, then whether it is negative or positive.
In our Work we have made two classifier in which one classifies between negative and positive while other classifies into negative, positive and neutral. So we have converted the text using TF-IDF and on the converted vector we have applied several ML models to achieve the accuracy which is 92% in case of first classifier and 82% in case of second classifier.

1 Introduction

1.1 Problem Statement

Here, We are doing sentiment analysis of customers reviews about Amazon fine food which is available on the Kaggle like whether it is positive, negative or neutral.
Performing the sentiment analysis on the customer reviews can help identify what is lacking and therefore guide you to improvement.
The goal of Sentiment Analysis is to harness this data in order to obtain important information regarding public opinion, that would help make smarter business decisions, political campaigns and better product consumption.

1.2 Motivation

Sentiment analysis is the process of reading tons of product reviews automatically and extract useful and meaningful information to discover if the customers are really satisfied with your product or not.
A person's feedback is more subjective rather than factual. Feedbacks can be negative, positive, or neutral.
Sentiment analysis applies Natural Language Processing (NLP) and Text Analysis techniques to highlight the subjective information from the text. You can review customer feedback and responses, and thus identify the negative comments and reasons why the customers have issues with your product or service.
Sentiment Analysis (SA) is one of the most widely studied applications of Natural Language Processing (NLP) and Machine Learning (ML). This field has grown tremendously with the advent of the Web 2.0.

1.3 Online Deployment Link

https://github.com/anupam121997/Amazon-Fine-Food-Review-to-Rating-Prediction

1.4 Project Pipeline Summary

On the Given dataset we have first preprocessed the dataset and removed null reviews, duplicate reviews, and un-useful reviews. Then we balanced the data based on Positive, Negative, and neutral class. After this we have applied TF-IDF and BoW vector on which several models like Random Forest, SVM, Multinomial Naive Bayes, Logistic Regression, and XGBoost are applied.
We have made a user interface in which user has to input a review and select whether he wants to find Positive, Negative Sentiment or Positive, Negative or Neutral Sentiment. and in the back-end we will pre-process the data and will predict the sentiment using our best saved model.

2 Related work

In this (Zhang and Wallace) the researchers have performed experiments to measure sensitivity analysis of the CNN model for sentence classification. They have made a baseline CNN model with 3 convolution layers and a classification layer then

they test the performance of the model on different architecture parameters like word embeddings, filter size, activation function, pulling strategy, and regularization. word embeddings using word2vec and glove Models perform almost the same. The filter region size can have a large effect on performance and should be tuned. Maxpooling outperforms other polling strategies by large magin. Tanh and RELU activatiobn functions performed almost same.

learning to store information over extended timestamps is not possible for recurrent neural networks. In this (Sundermeyer et al., 2012) introduces the efficient gradient-based method called 'Long short term memory ' (LSTM). This model has an internal mechanism called Gates that can regulate the flow of information these Gates can learn which data in a sequence is important to keep or not important this gets past relevant information down the chain to make predictions. The core concept of LSTM's is the cell state and its various gates. The cell state act as a transport highway that transfers relative information all the way down the sequence chain.

Likewise in this (Li et al., 2018), the author had used this approach to solve the classification problem of largescale text and In order to better extract the characteristics of the text. The Bi-LSTM-CNN model utilizes the loop structure to obtain the context information, and constructs the left and right contexts of each word through the Convolutional Neural Network (CNN) to construct the textual expression of the word, which is more accurately expressed the semantics of the text. Also, in this paper the result of Bi-LSTM and CNN combined had outperformed traditional ML models and CNN or Bi-LSTM implemented alone.

In this paper (Qi, 2020) the author has implemented the text classification using TF-IDF and ML models. The TF-IDF model is used to extract text features, and redefining each text data based on the extracted feature vector and inputting them into the trained model, respectively using XGBoost algorithm, KNN algorithm, the Naive Bayes algorithm, the SVM algorithm, GBDT algorithm to train and test data, and using the Precision (accuracy), Recall (recall rate), F1 - score (F1) to compare classification accuracy of each algorithm. Under the standard of high data quality, the combined algorithm based on TF-IDF and XGBoost is an effective text classification algorithm.

In this paper (Rhanoui et al., 2019) the author has stated the problem that arises due to larger texts or inter related text classification. The Author has applied several DL models like CNN, LSTM, Bi-LSTM, CNN-LSTM, and CNN-BiLSTM out of which CNN-LSTM and CNN-BiLSTM are the top two models. The combined CNN-BiLSTM model gives good results over the long text, since it benefits from the CNN's ability to extract features and the BiLSTM's characteristic to learn long-term bidirectional dependencies of the text.

3 Methodology

This dataset consists of reviews of fine foods from amazon.
The data span a period of more than 10 years, including all 500,000 reviews up to October 2012. Out of the given 10 features we have used the "Text: Review text" and "Score: Rating between 1 and 5".
On the Given dataset, firstly we have dropped all the rows which contain Null review Text. after this, we have removed all the duplicate reviews based on review text and score that means same review with same Score have been eliminated from the dataset. Also the dataset contains two features named "HelpfulnessNumerator : Number of users who found the review helpful" and "HelpfulnessDenominator : Number of users who indicated whether they found the review helpful or not" and the rows whose HelpfulnessNumerator value is greater than HelpfulnessDenominator are removed.
By the above Preprocessing we have left with 69% of data.
After this We have classified the Score of 1 and 2 into negative, 3 as neutral and 4 and 5 as positive sentiment and plotted the count for each sentiment shown in Figure 1.

From the Figure 1, we can clearly see that sentiments are in-balanced. so, we have taken a sample of data which contains 45000 negative sentiments, 30000 neutral sentiments and 45000 positive sentiments as shown in Figure 2

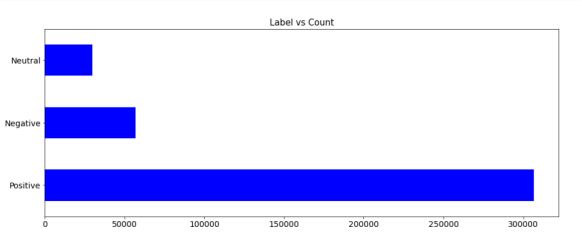

Figure 1: Sentiment class

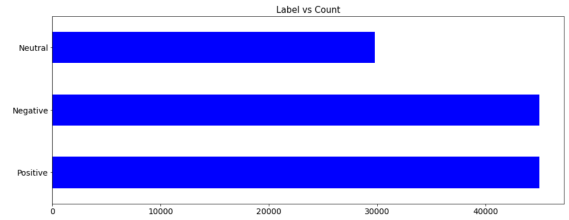

Figure 2: Sentiment class

After this we get our final dataset on which we have processed different text processing. It includes following steps:

- Removal of HTML Tags.
- Removal of punctuation.
- Removal of digits or numeric data.
- Removal of multiple White Spaces.
- Tokenization of data.
- lemmatization.

In our 2 class sentiment, we have applied TF-IDF on text and SVM model which resulted in 88% accuracy and for 3 class we have applied the same approach and get an accuracy of 72.05%.
In TF-IDF, TF denotes term frequency, where we first calculate the frequency of a term in a particular sentence. Then we calculate IDF, IDF denotes inverse of document frequency which is basically inverse of no of sentences which contain a particular word to the total no of sentences in a particular document. So, when we multiply TF and IDF together we get TF-IDF.
It preserves the semantic differences between two documents unlike Bag of Words or other techniques.
In our Approach We have first converted the processed text into TF-IDF using feature extraction of sklearn. Then on the converted training data, we have applied several ML models as listed below:

Logistic Regression

Logistic regression is a statistical model that in its basic form uses a logistic function to model a binary dependent variable.
It is used to model the probability of a certain class or event existing such as pass/fail, win/lose, alive/dead or healthy/sick or positive/negative.
On the training TF-IDF vectors, we have applied the model and tested on testing TF-IDF vectors and got a training accuracy of 90.20% and testing accuracy of 84.13%.

MultiNomial Naive Bayes

Being a supervised algorithm, it assumes that each feature is independent of the other in the sample space. It is based on the Bayes theorem.Given some prior information, Naive Bias computes the probability of occurrence of data in a specific category. In our project, we trained the Naive-bias model on 70% of the total corpus. The accuracy we got is Training Accuracy: 89.00%, Validation Accuracy: 85.35%.

Random Forest

Random forests is a supervised learning algorithm. It is utilized for classification. Firstly we have arbitrary data samples which are picked out. After that Random Forest generates a decision tree on these arbitrary data samples. The conjecture of each tree is obtained and the best and optimized solution is chosen. This is done by means of voting.In our project, we trained the Random Forest model on 70% of the total corpus. The training accuracy and validation accuracy we got is 99.99% and 84.56% respectively.

XGBoost

XGBoost is a highly efficient boosting algorithm. It is an execution of gradient boosting machines. It provides a collateral tree boosting that solves most data science and machine learning quickly.In our project, we trained the XGBoost model on 70% of the total corpus. The training accuracy and validation accuracy we got is 89.95% and 84.93% respectively.

SVM

A support vector machine (SVM) is a classification model that discriminates classes using a hyperplane defined using the formula w.x + b =0 where w represents the vectors, x represents the set of points

in ‘n’ dimensions and b is a constant the given Multi-class problem. In our project, we trained the SVM model on 70% of the total corpus. The training accuracy and validation accuracy we got is 92.54% and 92.67% respectively.

4 Experiments

After preprocessing the review text other than the TF-IDF and ML models, we have applied several approach to find the sentiment like CNN, Bi-LSTM, Bi-LSTM and CNN combined, and BoW with ML models. All of these Approaches are shown below:

CNN

We have made the CNN model in which we have add four convo1d layers with window size 2,3,4,5 and all of them has 40 filters. Then we apply max pooling on all of them and merge their result into one vector then we apply a multi-layer perception model to classify the given input as shown in Figure 3. we trained the SVM model on 70% of the total corpus. The training accuracy and validation accuracy we got is 99.97% and 88.199% respectively.

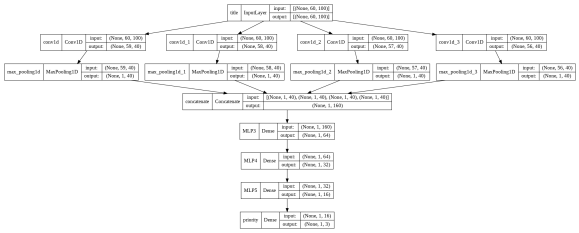

Figure 3: CNN Model

Bi-LSTM

Long short-term memory is the ARNN model. It has a feedback connection and it can process a complete sequence of data. In our project, we trained the Bi-LSTM neural network model on 70% of the data.
In our Approach we have used Glove Vectors for the weights and added two Bidirectional layers with a dense layer as shown in Figure 4.
Then we have fitted the model on training data and predicted on testing data and got an accuracy of 64.65%.

Bi-LSTM and CNN combined

We have also made an LSTM-CNN model whose initial layers are LSTM and the following layers

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, 50, 50)            5119850
_________________________________________________________________
bidirectional (Bidirectional (None, 50, 256)           183296
_________________________________________________________________
bidirectional_1 (Bidirection (None, 128)               164352
_________________________________________________________________
dense (Dense)                (None, 64)                8256
_________________________________________________________________
dropout (Dropout)            (None, 64)                0
_________________________________________________________________
dense_1 (Dense)              (None, 3)                 195
=================================================================
Total params: 5,475,949
Trainable params: 356,099
Non-trainable params: 5,119,850
_________________________________________________________________
```

Figure 4: LSTM Model

are CNN, at last, there are dense layers for classification. The LSTM layers take word embedding as input and generate new embedding which is dependent on previous words also now CNN layers extract important features from this embedding and the dense layer classifies the sentence using extracted features. The model summary is shown in Figure 5. Then we have fitted the model on training data and predicted on testing data and got an accuracy of 71.97%. For this we have taken Reference from (BiL).

```
Layer (type)                 Output Shape              Param #   Connected to
==================================================================================================
input_1 (InputLayer)         [(None, 100)]             0         []

embedding (Embedding)        (None, 100, 200)          20542600  ['input_1[0][0]']

spatial_dropout1d (SpatialDrop  (None, 100, 200)       0         ['embedding[0][0]']
out1D)

spatial_dropout1d_1 (SpatialDr  (None, 100, 200)       0         ['spatial_dropout1d[0][0]']
opout1D)

bidirectional (Bidirectional)  (None, 100, 256)       337920    ['spatial_dropout1d_1[0][0]']

spatial_dropout1d_2 (SpatialDr  (None, 100, 256)       0         ['bidirectional[0][0]']
opout1D)

conv1d_1 (Conv1D)            (None, 98, 32)            24608     ['spatial_dropout1d_2[0][0]']

global_average_pooling1d (Glob  (None, 256)            0         ['bidirectional[0][0]']
alAveragePooling1D)

global_max_pooling1d (GlobalMa  (None, 256)            0         ['bidirectional[0][0]']
xPooling1D)

global_average_pooling1d_1 (Gl  (None, 32)             0         ['conv1d_1[0][0]']
obalAveragePooling1D)

global_max_pooling1d_1 (Global  (None, 32)             0         ['conv1d_1[0][0]']
MaxPooling1D)

concatenate (Concatenate)      (None, 576)             0         ['global_average_pooling1d[0][0]'
                                                                  , 'global_max_pooling1d[0][0]',
                                                                  'global_average_pooling1d_1[0][0
                                                                  ]',
                                                                  'global_max_pooling1d_1[0][0]']

batch_normalization_2 (BatchNo  (None, 576)            2304      ['concatenate[0][0]']
rmalization)

dense (Dense)                (None, 128)               73856     ['batch_normalization_2[0][0]']

dropout (Dropout)            (None, 128)               0         ['dense[0][0]']

batch_normalization_3 (BatchNo  (None, 128)            512       ['dropout[0][0]']
rmalization)

dense_1 (Dense)              (None, 100)               12900     ['batch_normalization_3[0][0]']

dropout_1 (Dropout)          (None, 100)               0         ['dense_1[0][0]']

dense_2 (Dense)              (None, 3)                 303       ['dropout_1[0][0]']

==================================================================================================
```

Figure 5: Bi-LSTM + CNN Model

BoW with ML models

Bag of Words model is used to preprocess the text by converting it into a bag of words, which keeps a count of the total occurrences of most frequently

used words. It generally forms a matrix in which column is of total unique words and rows represents the sentence or training instances. It is basically a count Vectorizer in which if any particular word is present in any instance then at particular cell it will display 1 else 0.
After converting the text using Bag of Words, we have applied several Machine Learning Algorithms like we have applied on TF-IDF and recorded the accuracies which are less in comparison to TF-IDF.

5 Results and Analysis

The Training and testing Accuracy for several models that we have applied are shown in Figure 6 and Figure 7.

Models	Training Accuracy	Testing Accuracy
TF-IDF + SVM	92.54	92.67
TF-IDF + RF	99.99	84.56
TF-IDF + MNB	89.00	85.35
TF-IDF + XGB	89.95	84.93
CNN	96.03	81.79
Bi-LSTM	85.05	78.85
Bi-LSTM + CNN	88.75	83.46
BoW	93.87	86.55

Figure 6: 2 Class Results

Models	Training Accuracy	Testing Accuracy
TF-IDF + SVM	89.54	76.42
TF-IDF + RF	99.99	65.55
TF-IDF + MNB	68.67	64.85
TF-IDF + XGB	79.08	68.88
CNN	86.85	65.35
Bi-LSTM	85.05	64.65
Bi-LSTM + CNN	70.85	71.97
BoW	87.45	70.63

Figure 7: 3 Class Results

From this we can conclude that when we try to classify the reviews in positive, negative, neutral classes, the traditional machine learning models like naive bays, logistic regression, random forest, and XGboost on a bag of words and TF-IDF vector give accuracy in the range 68 to 70. When we apply SVM on the TF-IDF vector then we get an accuracy of 76. When we apply deep learning models like CNN, LSTM-CNN, Bilstm we get accuracy in the range of 65 to 70. A large number of neutral reviews are classified wrongly so accuracy has been dropped for the above models. Then we have tried to classify reviews in only two classes like positive and negative and we get accuracy in the range of 85 to 90 for traditional ML models and 80 to 83 for deep learning models.

References

Bi-LSTM and CNN model-TOP 10%. https://kaggle.com/parth05rohilla/bi-lstm-and-cnn-model-top-10.

Chenbin Li, Guohua Zhan, and Zhihua Li. 2018. News Text Classification Based on Improved Bi-LSTM-CNN. In *2018 9th International Conference on Information Technology in Medicine and Education (ITME)*, pages 890–893.

Zhang Qi. 2020. The Text Classification of Theft Crime Based on TF-IDF and XGBoost Model. In *2020 IEEE International Conference on Artificial Intelligence and Computer Applications (ICAICA)*, pages 1241–1246.

Maryem Rhanoui, Mounia Mikram, Siham Yousfi, and Soukaina Barzali. 2019. A CNN-BiLSTM Model for Document-Level Sentiment Analysis. *Machine Learning and Knowledge Extraction*, 1(3):832–847.

Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. 2012. LSTM neural networks for language modeling. In *Interspeech 2012*, pages 194–197. ISCA.

Ye Zhang and Byron Wallace. A Sensitivity Analysis of (and Practitioners Guide to) Convolutional Neural Networks for Sentence Classification. page 11.