Sentiment Analysis of Product Reviews

Laxmi Pranaya Yennam
lyennam@memphis.edu
U00928465

Abstract

Sentiment analysis of product reviews is an application problem that has recently become very popular in text mining and computational linguistics research. Sentiment analysis is a useful tool for analyzing the emotions conveyed in written material. The objective of this study is to examine the relationship between customer ratings and Amazon product reviews in order to categorize customer reviews. Understanding customer feedback would be aided by this for the organization. We combine natural language processing (NLP) and machine learning techniques to allocate sentiment scores that are weighted to the subjects, categories, themes, and entities that make up a sentence or phrase.

Introduction

Every product on e-commerce websites, regardless of category, is rated on a 1-to-5 scale, making it difficult to assess the benefits and drawbacks of individual product components. In order to provide readers with deeper insight into the underlying attitude, we want to classify and display thoughts regarding many aspects of a product separately in this project.

Motivation

The motivation of this project is to categorize opinions about different parts of a product and present them independently to give readers more information about the underlying sentiment. Sentiment analysis looks for textual elements that are representative of the context in which it is found and builds systems to capitalize on these elements. Although it is not the only issue, the challenge of categorizing text as good, neutral, or negative provides a reasonable foundation for further development.

Technical Challenges

- Time Complexity (increases as the input size increases)
- High Dimensionality
- Understanding Algorithm
- Data Resampling
- Handling Imbalance Dataset
- Software Implementation

Solution/Methodology

Product Reviews are analyzed to help companies to get a complete idea on feedback's provided by customers and can take care on the particular fields.
The overall methodology involves four steps:
1. Data Collection
2. Pre-Processing
3. Sentiment Extraction
4. Classification of Sentiment

1. Data Collection

We have used the Amazon reviews original dataset to test our reviews classification methods. We have collected reviews of Musical Instruments from Amazon website. The datasets are available and have been collected from Kaggle.

Features of our dataset are:

reviewerID - ID of the reviewer, e.g. A2SUAM1J3GNN3B
productID - ID of the product, e.g. 0000013714
reviewerName - name of the reviewer
helpful - helpfulness rating of the review, e.g. 2/3
reviewText - text of the review
overall - rating of the product
summary - summary of the review
ReviewTime - time of the review

NOTE: The data is structured and comes in CSV format, comprising 10261 rows and 9 columns.

2. Pre-Processing

Data pre-processing is a significant step in the text mining process and plays an important part in a number of supervised learning techniques.

- Checking for Duplicates
- Checking for Null Values
- Making text lowercase
- Removing hyperlinks
- Removing punctuation marks
- Eliminating Stop Words
- Data Resampling

3. Sentiment Extraction

For the purpose of Sentiment Extraction we make use of TextBlob, which is a python library for Natural Language Processing (NLP).

- TextBlob actively used Natural Language ToolKit (NLTK) that supports complex analysis and operations on textual data. It returns Polarity and Subjectivity of a sentence.
- Polarity lies between [-1,1]: -1 defines a negative sentiment, 0 defines a neutral sentiment and 1 defines a positive sentiment.
- Subjectivity lies between [0,1]: Subjectivity quantifies the amount of personal opinion and factual information contained in the text. The higher subjectivity means that the text contains personal opinion rather than factual information.
- For example, "I do not like this concept at all, it is too boring" for this polarity = -1 and subjectivity = 1, which is fair.
- Also, "This was a helpful concept but I would prefer another one". It returns 0.0 for both subjectivity and polarity which is not the finest answer.

4. Classification of Sentiments

TF-IDF Vectorizer

TF-IDF is an abbreviation for **Term Frequency-Inverse Document Frequency.**

- This is a widely used algorithm that converts text into a meaningful numerical representation that may be fitted to a machine learning algorithm for prediction. Text is converted into feature vectors so that they can be fed into an estimator.
- It assigns a value to a term according to its importance in a document scaled by its importance across all documents in your corpus, which mathematically eliminates naturally occurring words in the English language, and select words that are more descriptive of text.

TFIDF

For a term i in document j:

$$w_{i,j} = tf_{i,j} \times \log\left(\frac{N}{df_i}\right)$$

tf_{ij} = number of occurrences of i in j
df_i = number of documents containing i
N = total number of documents

Figure 1: TF-IDF

Let consider an example of two sentences
Sentence 1: The car is driven on the road.
Sentence 2: The truck is driven on the highway.

Word	TF		IDF	TF*IDF	
	A	B		A	B
The	1/7	1/7	log(2/2) = 0	0	0
Car	1/7	0	log(2/1) = 0.3	0.043	0
Truck	0	1/7	log(2/1) = 0.3	0	0.043
Is	1/7	1/7	log(2/2) = 0	0	0
Driven	1/7	1/7	log(2/2) = 0	0	0
On	1/7	1/7	log(2/2) = 0	0	0
The	1/7	1/7	log(2/2) = 0	0	0
Road	1/7	0	log(2/1) = 0.3	0.043	0
Highway	0	1/7	log(2/1) = 0.3	0	0.043

Figure 2: TF-IDF Evaluation

Data Resampling

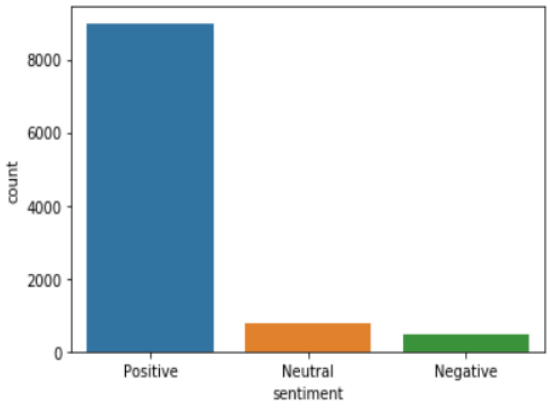

Figure 3: Data Resampling

In Figure 3, it shows that we have Positive, Neutral and Negative classes. Also, these three classes are imbalanced as Neutral and Negative classes have small amount of data while Positive class has more than 8000 reviews. Hence there is a need to resample the data so that our classification model is not biased to only a single class.

Handling Imbalanced Dataset

SMOTE (Synthetic Minority Oversampling Technique)

- It is an oversampling technique where the synthetic samples are generated for the minority class. This algorithm helps to overcome the overfitting problem posed by random oversampling. It increases the number of cases in your dataset in a balanced way.
- **SMOTE** works by selecting examples that are close in the feature space, drawing a line between the examples in the feature space, and drawing a new sample at a point along that line.

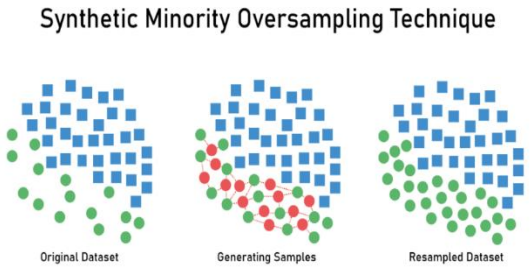

Figure 4: Synthetic Minority Oversampling Technique

Building Classification Model

Logistic Regression

- Logistic regression is a classification algorithm that uses the weighted combination of the input features and passes them through a sigmoid function.
- Here, we use One-vs-all is a strategy for multi-class classification that involves training N distinct binary classifiers, each designed to recognize a specific class.
- In this, we consider one class as 1 and rest all as 0, we train the model and get the requisite weights. We store the value of weights in a dictionary format for each classifiers.
- Then with the help of the Sigmoid Function we calculate the probability. The highest probability takes a presidency and we classify that data to corresponding classifier.

Decision Tree Classifier

- A decision tree is a powerful tool used in machine learning for both classification and regression tasks. It is a flowchart-like structure where each internal node represents a "decision" based on the value of a feature, each branch represents the outcome of that decision, and each leaf node represents the final classification or regression result.
- The strategy of the decision tree algorithm involves recursively partitioning the dataset based on the values of input features.
- Then selecting the best feature for splitting at each step to maximize information gain or minimize impurity, until a stopping criterion is met, resulting in a hierarchical tree structure where leaf nodes represent final classification or regression outcomes.

Support Vector Classifier

- Support Vector Classifier (SVC) is a supervised learning algorithm that aims to find the optimal hyperplane in a high-dimensional space that best separates the classes of the training data, maximizing the margin between the classes while minimizing classification errors, making it particularly effective for binary classification tasks.
- The strategy of the Support Vector Classifier (SVC) involves finding the hyperplane that best separates the classes in the feature space while maximizing the margin between the closest data points from each class, known as support vectors.
- This is achieved by solving a convex optimization problem, where the objective is to minimize classification errors while penalizing misclassifications, subject to the constraint that data points are correctly classified or lie within a certain margin from the decision boundary.

Implementation

Technologies Used

- Python
- NumPy
- Pandas
- Seaborn
- Scikit-Learn(sklearn)
- NLTK(Natural Language Toolkit)
- TextBlob
- Matplotlib

System Design

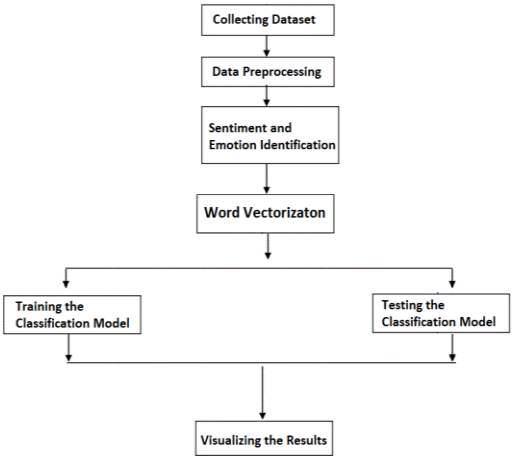

Figure 5: System Architecture of Sentiment Analysis

UML Diagrams

Unified Modeling Language (UML) is a standard language used for business modeling and other non-software systems as well as for defining, visualizing, building, and documenting the artifacts of software systems. Large and complex system modeling has shown to be successful when using a set of best engineering practices, which are represented by the UML. The software development process and the creation of objects-oriented software both heavily rely on the UML. Software project design is expressed mostly through graphical notations in the UML.

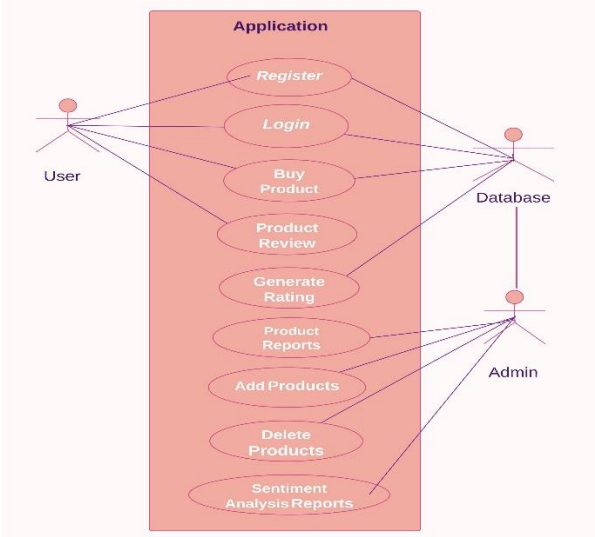

Figure 6: Use Case Diagram for Sentiment Analysis

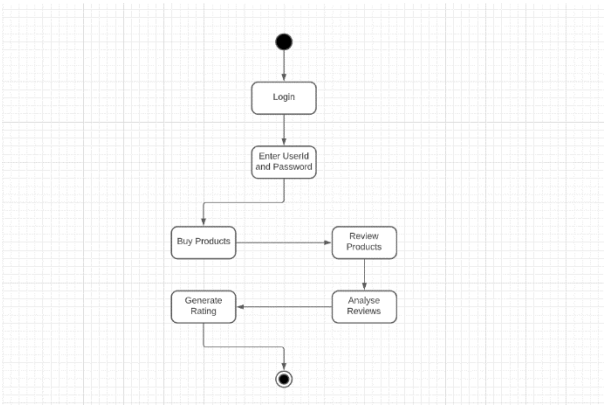

Figure 7: State Chart Diagram for Sentiment Analysis

Evaluation Results: Confusion Matrix, Classification Report-Accuracy, Precision, Recall, F1-Score, Support

Confusion Matrix Format-

Negative	TNg	FNt2	FP1
Neutral	FNg2	TNt	FP2
Positive	FNg1	FNt1	TP
Actual/ Predicted	**Negative**	**Neutral**	**Positive**

Logistic Regression-

Implemented Logistic Regression Algorithm

```
[17]: from sklearn.linear_model import LogisticRegression

      logreg = LogisticRegression(C=10000.0, random_state=0, max_iter=1000)
      logreg.fit(X_train, y_train)
      y_pred = logreg.predict(X_test)
      print('Accuracy of logistic regression classifier on test set: {:.2f}'.format(logreg.score(X_test, y_test)))
      print(f'\nConfusion Matrix for Logistic Regression:\n {confusion_matrix(y_test, y_pred)}')
      print(f'\nClassification Report for Logistic Regression:\n {classification_report(y_test, y_pred)}')

      Accuracy of logistic regression classifier on test set: 0.95

      Confusion Matrix for Logistic Regression:
       [[2326    0    0]
       [   0 2232    0]
       [ 173  169 1867]]

      Classification Report for Logistic Regression:
                    precision    recall  f1-score   support

                 0       0.93      1.00      0.96      2326
                 1       0.93      1.00      0.96      2232
                 2       1.00      0.85      0.92      2209

          accuracy                           0.95      6767
         macro avg       0.95      0.95      0.95      6767
      weighted avg       0.95      0.95      0.95      6767
```

Figure 8: Evaluation Results for Logistic Regression

Predicted the output using logistic regression

```
[19]: pred = logreg.predict(X_test[5741])
      print(pred)

      actual = y_test[5741]
      a = [0, 0 ,0]
      if(pred == 2):
          a[2] = 1
          a[0] = 0
          a[1] = 0
      elif(pred == 1):
          a[1] = 1
          a[0] = 0
          a[2] = 0
      else:
          a[0] = 1
          a[1] = 0
          a[2] = 0
      print(a)

      [1]
      [0, 1, 0]
```

Figure 9: Sample Input/Output Test Case

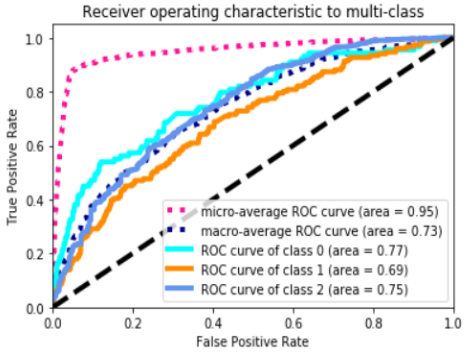

Figure 10: ROC Curve for Logistic Regression

Decision Tree Classifier-

Gini-

```
[20]: from sklearn.tree import DecisionTreeClassifier

      clf_gini = DecisionTreeClassifier(criterion="gini", random_state=100, max_depth=3, min_samples_leaf=5)
      clf_gini.fit(X_train, y_train)
      y_pred_clf_gini = clf_gini.predict(X_test)

[21]: print(f'Confusion Matrix for Decision Tree(gini):\n {confusion_matrix(y_test, y_pred_clf_gini)}')
      print(f'\nAccuracy for Decision Tree(gini): {accuracy_score(y_test, y_pred_clf_gini)*100}')
      print(f'\nClassification Report for Decision Tree(gini):\n {classification_report(y_test, y_pred_clf_gini)}')

      Confusion Matrix for Decision Tree(gini):
       [[ 725  766  835]
       [ 495  890  847]
       [ 129  592 1488]]

      Accuracy for Decision Tree(gini): 45.854883995862274

      Classification Report for Decision Tree(gini):
                    precision    recall  f1-score   support

                 0       0.54      0.31      0.39      2326
                 1       0.40      0.40      0.40      2232
                 2       0.47      0.67      0.55      2209

          accuracy                           0.46      6767
         macro avg       0.47      0.46      0.45      6767
      weighted avg       0.47      0.46      0.45      6767
```

Figure 11: Evaluation Results for Decision Tree Classifier(Gini)

Entropy-

```
[22]: clf_entropy = DecisionTreeClassifier(
              criterion="entropy", random_state=100,
              max_depth=3, min_samples_leaf=5)

      clf_entropy.fit(X_train, y_train)
      y_pred_clf_entropy = clf_entropy.predict(X_test)

[23]: print(f'Confusion Matrix for Decision Tree(entropy):\n {confusion_matrix(y_test, y_pred_clf_entropy)}')
      print(f'\nAccuracy for Decision Tree(entropy): {accuracy_score(y_test, y_pred_clf_entropy)*100}')
      print(f'\nClassification Report for Decision Tree(entropy):\n {classification_report(y_test, y_pred_clf_entro

      Confusion Matrix for Decision Tree(entropy):
       [[ 708  783  835]
       [ 449  936  847]
       [ 116  605 1488]]

      Accuracy for Decision Tree(entropy): 46.283434313580614

      Classification Report for Decision Tree(entropy):
                    precision    recall  f1-score   support

                 0       0.56      0.30      0.39      2326
                 1       0.40      0.42      0.41      2232
                 2       0.47      0.67      0.55      2209

          accuracy                           0.46      6767
         macro avg       0.48      0.47      0.45      6767
      weighted avg       0.48      0.46      0.45      6767
```

Figure 12: Evaluation Results for Decision Tree Classifier(Entropy)

Support Vector Classifier-

```
[24]: from sklearn.svm import SVC

      clf = SVC(kernel='linear')
      clf.fit(X_train, y_train)
      clf_predictions = clf.predict(X_test)

[25]: print(f'Confusion Matrix for Support Vector Classifier:\n {confusion_matrix(y_test, clf_predictions)}')
      print(f'\nAccuracy for Support Vector Classifier: {accuracy_score(y_test, clf_predictions)*100}')
      print(f'\nClassification Report for Support Vector Classifier:\n {classification_report(y_test, clf_predictio

      Confusion Matrix for Support Vector Classifier:
       [[2326    0    0]
       [   5 2218    9]
       [ 128  212 1869]]

      Accuracy for Support Vector Classifier: 94.76873060440373

      Classification Report for Support Vector Classifier:
                    precision    recall  f1-score   support

                 0       0.95      1.00      0.97      2326
                 1       0.91      0.99      0.95      2232
                 2       1.00      0.85      0.91      2209

          accuracy                           0.95      6767
         macro avg       0.95      0.95      0.95      6767
      weighted avg       0.95      0.95      0.95      6767
```

Figure 13: Evaluation Results for Support Vector Classifier

Conclusion

In conclusion, I tried traditional machine learning algorithms to analyze customer feedback and classify all the classes by splitting the sentiments (Positive, Neutral and Negative). Logistic Regression is best algorithm obtained among other machine learning algorithms with 95% of accuracy. These analysis help to get a complete idea on feedback's provided by customers and can take care on those particular fields.

Leveraging on machine learning and NLP, organizations can interact with their customers both rationally and emotionally, improve their customer experience, and provide tailored assistance.

References

1. Sangani, Chirag, and SundaramAnanthanarayanan. "Sentiment analysis of app store reviews." https://cs229.stanford.edu/proj2013/CS229-ProjectReport-ChiragSangani-SentimentAnalysisOfAppStoreReviews.pdf
2. Callen Rain,"Sentiment Analysis in Amazon Reviews Using Probabilistic Machine Learning" Swarthmore College, Department of Computer Science. https://www.sccs.swarthmore.edu/users/15/crain1/files/NLP_Final_Project.pdf
3. https://www.analyticsvidhya.com/blog/2018/04/a-comprehensive-guide-to-understand-and-implement-text-classification-in-python/
4. K.Yessenov and S. Misailovic "Sentiment analysis of movie review comments. Methodology". https://dl.acm.org/doi/10.1145/2173637.2173655
5. https://towardsdatascience.com/document-feature-extraction-and-classification-53f0e813d2d3
6. M.S.Elli and Y.-F. Wang."Amazon reviews, business analytics with sentiment analysis". https://www.irjmets.com/uploadedfiles/paper/issue_7_july_2022/28573/final/fin_irjmets1658457836.pdf
7. https://machinelearningmastery.com/random-oversampling-and-undersampling-for-imbalanced-classification/