Sentiment Analysis on Customer Feedback

Progress Report

Group 23

Group Members

21903488 Emrehan Hoşver, 21901798 Gökberk Altıparmak, 21802480 Tuğberk Dikmen, 21901418 Efser Efe Kantik

1. Introduction

In the rising era of e-commerce, understanding customer sentiments has become paramount for enhancing customer experiences and driving business decisions. Sentiment analysis, a domain of natural language processing (NLP), offers a powerful tool to gauge public opinion, emotional nuances, and customer satisfaction by analyzing unstructured text data from customer feedback and reviews.

Our project, focused on sentiment analysis of customer feedback, aims to leverage machine learning techniques to decode the intricate layers of customer sentiment expressed in e-commerce platforms. This report outlines our progress in developing a model capable of classifying customer feedback into distinct sentiment categories: positive, negative, and neutral.

2. Background Information

The initial phase of our project involved an extensive analysis of various datasets, primarily sourced from Kaggle. These datasets include customer feedback from Brazilian e-commerce platforms and the Jigsaw Toxic Comment Classification Challenge dataset, which, though primarily for toxic comment classification, offers valuable insights for sentiment analysis.

Our approach has multiple stages, starting with data preprocessing, which involves cleaning, stop-word removal, and label mapping. The primary goal is to transform unstructured text data into a structured format suitable for machine learning algorithms.

We have explored different machine learning models:The first model we implemented was based on Support Vector Machines (SVM), known for their effectiveness in text classification tasks due to their ability to handle high-dimensional data and their robustness in various scenarios, including imbalanced datasets.

In addition to the SVM' s, we also developed a Naive Bayes classifier. The choice of Naive Bayes is motivated by its simplicity, efficiency, and particularly its suitability for large datasets commonly encountered in sentiment analysis tasks. Our implementation of Naive Bayes includes preprocessing steps such as text vectorization using CountVectorizer and handling imbalanced data using techniques like resampling and minority over-sampling.

3. Dataset Analysis

The objective outlines a comprehensive process for preparing a sentiment analysis dataset for machine learning. The initial step involves preprocessing each line of the dataset using a custom function in the code. This function arranges whitespace, removes extraneous quotation marks, and corrects any instances of double quotation marks. Following this, the datasets, stored in CSV files, are read and each line is processed accordingly.

Only the texts, sentiments and confidence values are loaded from the CSV file. Confidence values are between 1-5. Values between [1,2] are labeled as negative, 3 is labeled as neutral and [4,5] are labeled as positive. There are also different unnecessary pieces of information that do not have any effect on training or testing in the dataset. Once the data is loaded, it is structured into a Pandas DataFrame for ease of manipulation. In this phase, the column names are cleaned to remove any whitespace, and the first row, containing these column names, is discarded from the DataFrame. The dataset is further refined by excluding rows where the text column is null, ensuring data quality and consistency.

The next critical step involves feature extraction, where the text column of the dataset is transformed into a matrix of token counts using Scikit-Learn's `CountVectorizer`. This process includes the removal of English stop words, a common practice in natural language processing to reduce noise in the data. Concurrently, the sentiment column is processed to create three categories: 0 for negative, 1 for neutral, and 2 for positive sentiments. This

trinary classification system is pivotal for nuanced sentiment analysis, allowing for a more comprehensive understanding of the data's sentiment spectrum.

For the purpose of model training and evaluation, the dataset is split into training and testing sets. A 80-20 split is used, allocating 80% of the data for training and 20% for testing, using the `train_test_split` function with a fixed random state for reproducibility. Also, we created a validation set from the train set by splitting 25% of the train set, as seen in Figure 1. Research shows that this distribution is considered a standard practice in machine learning, balancing a substantial portion of data for model training while retaining enough for an effective evaluation of the model's performance.

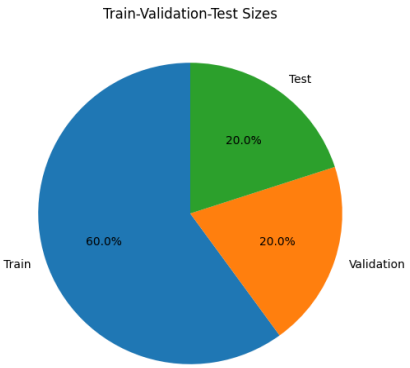

Figure 1: The train-validation-test dataset splitting

In conclusion, this dataset, tailored for a more detailed sentiment analysis, comprises text entries each associated with a sentiment rating that spans negative, neutral, and positive categories. The preprocessing steps, crucial in transforming raw text into a machine-readable format, are designed to accommodate this trinary sentiment classification, thereby equipping the machine learning model to perform a more layered and insightful sentiment analysis.

4. Coding Environment

The development of our project progressed on Python because of the support for machine learning and natural language processing. Python's ecosystem, with its

comprehensive set of libraries and tools, has been critical in the development of our sentiment analysis project. Below, we list the key Python libraries used in our project so far:

4.1. Pandas: A cornerstone in our data handling processes, Pandas provides high-level data structures and functions designed for easy manipulation and analysis of structured data. It has been used primarily in data cleaning, transformation, and the exploration of datasets.

4.2. Scikit-learn: This library forms the backbone of our machine learning operations. Scikit-learn offers a wide array of tools for data preprocessing, model building, and evaluation. We have specifically leveraged its capabilities for model training (using both SVM and Naive Bayes classifiers), data splitting, and feature extraction. The library's user-friendly interface and comprehensive documentation have greatly accelerated our development process.

4.3. NLTK (Natural Language Toolkit): Essential for NLP tasks, NLTK provides a suite of text processing libraries for classification, tokenization, stemming, tagging, and parsing. We have used NLTK for preprocessing the textual data, including removing stopwords, which are words that do not contribute significantly to sentiment analysis.

4.4. Imbalanced-learn: To address the challenge of imbalanced datasets, we have employed Imbalanced-learn, a library offering various resampling techniques. This includes SMOTE, which we have used to synthetically oversample the minority classes in our dataset, ensuring a more balanced and representative training process.

4.5. CountVectorizer from Scikit-learn: An integral part of our text preprocessing pipeline, CountVectorizer converts text data into a matrix of token counts, effectively transforming raw text into a format that can be easily processed by machine learning algorithms.

4.6. TfidfVectorizer from Scikit-learn: Similar to CountVectorizer, TfidfVectorizer was used in our SVM model to convert text documents into a matrix of TF-IDF features. This method emphasizes the importance of frequent terms in a document while accounting for

their frequency across the entire dataset, providing a more nuanced feature set for model training.

5. Trained Models

In the first phase of the project we decided to train the dataset by using a Multinomial Naive Bayes Classifier and Support Vector Machine.

5.1 Multinomial Naive Bayes

Main goal of the Multinomial Naive Bayes Classifier is to find the class which maximizes the posterior probability. MNB uses Bayes' theorem to maximize the posterior probability. The Bayes theorem states that

$$P(Y|X) = \frac{P(X|Y) \cdot P(Y)}{P(X)}$$

In this basic form of the formula, Y represents the classes and X represents the future. If we assume that X is a vector that holds multiple features

$$P(Y|\mathbf{X}) = \frac{P(X_1, X_2, \ldots, X_n|Y) \cdot P(Y)}{P(X_1, X_2, \ldots, X_n)}$$

The equation comes into this form. Naive assumption says that we can think each future is independent from each other.

$$P(X_1, X_2, \ldots, X_n|Y) = P(X_1|Y) \cdot P(X_2|Y) \cdot \ldots \cdot P(X_n|Y)$$

By doing this assumption we can reformulate our formula as

$$P(Y|X_1, X_2, \ldots, X_n) = \frac{P(Y) \cdot \prod_{i=1}^{n} P(X_i|Y)}{\prod_{i=1}^{n} P(X_i)}$$

Since the denominator (normalization) is constant we can reach the following formula.

$$P(Y|X_1, X_2, \ldots, X_n) \propto P(Y) \cdot \prod_{i=1}^{n} P(X_i|Y)$$

The Sklearn library that we are using in this part of the project uses the same formulation for the Naive Bayes Classifier [4]. To estimate the Likelihoods in Multinomial

Naive Bayes classifier $P(X_i|Y)$ The Sklearn library uses a smoothed version of the Maximum Likelihood Estimation [4].

$$\hat{\theta}_{yi} = \frac{N_{yi} + \alpha}{N_Y + \alpha n}$$

After doing these calculations in the Multinomial Naive Bayes Classifier we set the predicted class by choosing y that maximizes the probability of the Multinomial Naive Bayes Classifier.

$$\hat{y} = \arg\max_y P(Y = y) \cdot \prod_{i=1}^{n} P(X_i|Y = y)$$

5.2 Support Vector Machine Classifier

Main purpose of the Support Vector Machine classifiers is to find hyperlanes that separate data to classify the data. Support Vectors position these hyperplanes. In the Sklearn one can decide the type of function to classify the data such as linear, polynomial, radial basis, and precomputed [5]. In our initial experiment we used the default function which is the radial basis function. There are two different classifiers One-vs-One and One-vs-Rest. One-vs-One compares the classes as pairs. One-vs-Rest directly compares the class with all other classes combined. The Sklearn library lets us choose the classifier we want. However, when dealing with multiclass classification, One-vs-One will be used internally and One-vs-Rest matrix will be constructed using One-vs-One matrix, default option for this parameter is One-to-Rest [6]. The goal of Support Vector machines are formulated as

$$\min_{w,\,b,\,\xi} \frac{1}{2} w^T w + C \sum_{i=1}^{N} \xi_i$$

subject to:

$$y_i(w^T \phi(x_i) + b) \geq 1 - \xi_i$$
$$\xi_i \geq 0$$

(w, b) are the parameters of the hyperplane

φ(x) is the feature vector transformed using the RBF kernel

C is a regularization parameter

ξ_i are slack variables

N is the number of data points

this [5].

C regularization parameter is a parameter to decide how strictly we are going to classify the data. We also used the default parameter here which is 1 in our test model[6].

6. Training Results

Using the Amazon Musical Instruments dataset, we implemented the first machine learning model of Multinomial Naive Bayes. This classifier provided strong performance in classifying the positive sentiments. Nevertheless, we faced challenges in classifying the negative and neutral classes.

The imbalanced structure of this dataset impacts the negative and neutral results, with lower precision, recall, and F1-scores as seen from Figure 2. Still, the overall validation and test accuracies for the original dataset are almost 86% and 88%, respectively. These relatively high results were driven by the dominant positive class.

```
Validation Set Accuracy: 0.8611111111111112
Validation Set Classification Report:
              precision    recall  f1-score   support

    negative       0.40      0.02      0.04       105
     neutral       0.08      0.01      0.01       173
    positive       0.87      0.99      0.93      1774

    accuracy                           0.86      2052
   macro avg       0.45      0.34      0.32      2052
weighted avg       0.78      0.86      0.80      2052

Test Set Accuracy: 0.8830979055041402
Test Set Classification Report:
              precision    recall  f1-score   support

    negative       0.71      0.06      0.11        83
     neutral       0.12      0.01      0.02       150
    positive       0.89      0.99      0.94      1820

    accuracy                           0.88      2053
   macro avg       0.57      0.36      0.36      2053
weighted avg       0.83      0.88      0.84      2053
```

Figure 2: Accuracy and Performance Results of Initial Amazon Dataset

This sensitivity to the imbalanced nature of this dataset is overcomed using oversampling techniques. The substantial number of positive sentiments compared to other classes was addressed using the Minority Oversampling method. This method provides the incrementation of the number of instances in each minority class to accomplish a better balanced distribution of the sentiments.

The class distribution within this original dataset was characterized by a remarkable imbalance between majority and minority classes. Employing a resampling strategy, the instances in the minority classes were matched to equal the number of instances in the majority class, i.e. the positive sentiment. This meticulous process resulted in the acquisition of a balanced dataset. Subsequent to text vectorization and model training mirroring the approach of the initial model, the Multinomial Naive Bayes, was evaluated on both the validation and test sets to appraise its performance post-oversampling. This oversampling technique successfully balanced the class distribution, as evidenced by the pie charts provided in Figure 3.

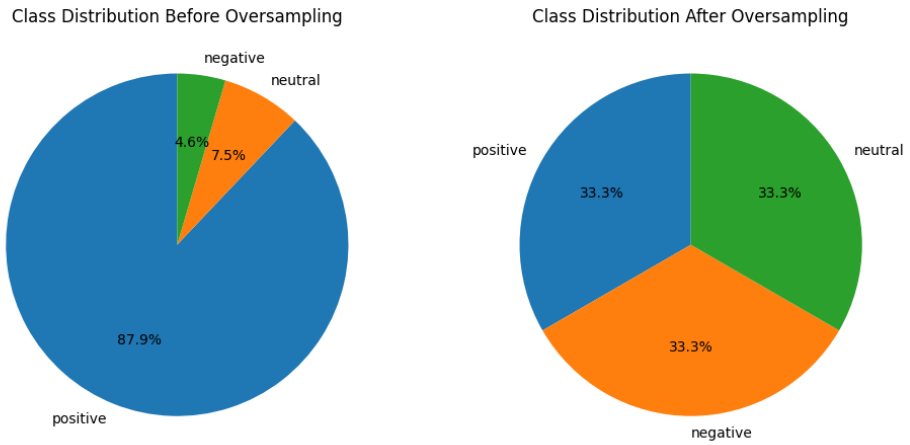

Figure 3: Class Distributions Before & After Oversampling of the Minority Classes

The application of oversampling to cope with the class imbalance had a remarkable impact on the Multinomial Naive Bayes model's performance. Comparing before and after results provides valuable insights to visualize effectiveness of this method.

The classifier reached accuracy around 88% and 89% for validation and test datasets, respectively, thus indicating enhancement in overall performance. The most significant improvement was observed in the e precision, recall, and F1-scores metrics of the minority classes, i.e. the negative and neutral classes. The weighted average F1-score, which takes the

imbalance in class sizes into consideration, saw an advancement from 84% to 89% in the test set, as seen in the below Figure 4.

```
Validation Set Accuracy: 0.8789950120081286
Validation Set Classification Report:
              precision    recall  f1-score   support

    negative       0.91      0.92      0.92      1757
     neutral       0.83      0.89      0.86      1824
    positive       0.90      0.83      0.86      1832

    accuracy                           0.88      5413
   macro avg       0.88      0.88      0.88      5413
weighted avg       0.88      0.88      0.88      5413


Test Set Accuracy: 0.8876985592907277
Test Set Classification Report:
              precision    recall  f1-score   support

    negative       0.93      0.92      0.93      1827
     neutral       0.84      0.89      0.87      1799
    positive       0.89      0.85      0.87      1788

    accuracy                           0.89      5414
   macro avg       0.89      0.89      0.89      5414
weighted avg       0.89      0.89      0.89      5414
```

Figure 4: Accuracy and Performance Results of Initial Amazon Dataset with Oversampling Technique

The Support Vector Machine classifier is applied on the Brazilian E-Commerce dataset.

7. Work Distribution

21903488 Emrehan Hoşver: Building and training Support Vector Machine Classifier, dataset research, data preprocessing.

21901798 Gökberk Altıparmak: Explanation of used datasets, dataset split and preprocessing stage, analyzing and printing output of a sample dataset referenced at [1].

21802480 Tuğberk Dikmen: Introduction, Background Information, Coding Environment explanation, Naive Bayes implementation.

21901418 Efser Efe Kantik: Multinomial Naive Bayes implementation, dataset selection, preprocessing and balancing, evaluation of results.

References

[1] V.Salodkar, "Customer Feedback Dataset, " Kaggle. [Online]. Available: https://www.kaggle.com/datasets/vishweshsalodkar/customer-feedback-dataset [Accessed: Oct 19, 2023].

[2] Olist, "Brazilian E-Commerce Public Dataset by Olist, " Kaggle. [Online]. Available: https://www.kaggle.com/datasets/olistbr/brazilian-ecommerce [Accessed: Oct 19, 2023].

[3]J. Peller, "jigsaw-toxic-comment-classification-challenge, " Kaggle. [Online]. Available:https://www.kaggle.com/datasets/julian3833/jigsaw-toxic-comment-classification-challenge/code [Accessed: Oct 19, 2023].

[4]"1.9.NaiveBayes,"*scikit-learn.org.*[Online]. Available:https://scikit-learn.org/stable/modules/naive_bayes.html [Accessed: Nov 22, 2023].

[5]"1.4. Support Vector Machines," *scikit-learn.org.* [Online]. Available: https://scikit-learn.org/stable/modules/svm.html# [Accessed: Nov 23, 2023].

[6]"sklearn.svm.SVC,"*scikit-learn.org.*[Online]. Available:https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html#sklearn.svm.SVC [Accessed: Nov 23, 2023]