Classifying Sentiments on the Amazon Fine Food Reviews Dataset

Tales Ivalque Taveira de Freitas, NLP, INSPER

***Abstract*—The abstract goes here.**

***Index Terms*—IEEE, IEEEtran, journal, LaTeX, paper, template.**

I. Dataset

THIS document presents the classification of sentiments on the Amazon Fine Food Reviews dataset. The dataset contains thousands of reviews of various food products, each consisting of a note of 1 to 5, a plain review text, and some other features. This dataset was used in the paper [1], which looks into how the previous experiences affects the grades given by each user, and how that can be used in a classification model.This dataset span a period of more than 10 years, including all 500,000 reviews up to October 2012.

II. Classification Pipeline

The classification model chosen for this paper was the Bernoulli Naive Bayes algorithm, which is appropriate for binary features such as word presence in text.The Bernoulli Naive Bayes algorithm is a variant of the Naive Bayes algorithm, particularly well-suited for binary/boolean data. It's commonly used when your features represent binary events — i.e., whether a word occurs in a document or not. Unlike other versions (like Multinomial Naive Bayes which counts occurrences of features), Bernoulli Naive Bayes uses binary features (1 for presence, 0 for absence of a word in a document). Still, as a Naive Bayes variation, this classifier is a probabilistic machine learning model based on Bayes' Theorem, and assumes that all features are independent of each other, which is seen as a "naive" assumption.

For the classification to be made possible, there are some necessary steps required for a better model accuracy. The first of which is the selection of features and target. For this specific dataset and assignment, the chosen feature was the Text, which is a plain text review of a food product, and the Score was chosen to be the Target for our classifier. Before passing the dataset to the classifier, there is the need to do some adaptations to both feature and target. For the target feature, there was only one change, changing the numeric order for a categorical one, and the rule used for this was that if the numerical score was less than or equal to 3, it would be changed to "Negative", and to "Positive" otherwise.

For the Feature, on the other hand, there was a need of a series of transformations to ensure that the naive assumption worked as well as possible. This is made using 4 steps:

1) Cleaning text: This step is required to ensure that only the words will be relevant, and signifies the removal of all non-alphanumerical symbols, meaning every punctuation, signals and whatnot will be removed, leaving only numbers and words.
2) Stopword removal: In every text, there are many words that don't carry a weight in the class given to that text, these words are called stopwords and are represented by the most common words in the dictionary, such as "this, that, a, ...". The removal of such words are necessary to insure that only meaningful words will be used in the model training and prediction.
3) Stemming: In natural language processing, words can appear in different forms (e.g., "running," "ran," "runner"). These variations often carry the same or similar meaning, and retaining all of them can lead to redundancy in the dataset. Stemming is the process of reducing words to their root or base form, known as the stem.This reduction is important to ensure that variations of the same word are treated uniformly during model training and prediction, helping to simplify the vocabulary and improve model performance.
4) Text Augmentation: In natural language processing, a key challenge is the availability of labeled data. Text augmentation is a technique used to artificially increase the size of the dataset by introducing variability in the text without changing its meaning. This can be done by substituting words with their synonyms, rephrasing sentences, or applying other transformations. For instance, replacing "happy" with "joyful" retains the sentiment but introduces new examples for the model to learn from. Augmenting text helps to improve model robustness, making it less sensitive to overfitting and better at generalizing to unseen data.

After this step of Pre-processing, a Classifying Pipeline was built. The pipeline consisted of:

- **CountVectorizer**: This step converts the raw text into a bag-of-words representation where each review is represented as a vector of word counts.
- **TfidfTransformer**: This step transforms the word counts into Term Frequency-Inverse Document Frequency (TF-IDF) values to account for how often words appear across the dataset.
- **Bernoulli Naive Bayes Classifier**: A Naive Bayes classifier that assumes binary features and calculates the probability of the review being positive or negative.

Figure 1 shows a diagram of the pipeline used in this project.

III. Evaluation

In this step, we evaluate the classifier by splitting the dataset into training and test sets multiple times, ensuring that each split is shuffled to avoid biases. Since the dataset is not perfectly balanced, we use the balanced accuracy score [2] as the primary evaluation metric, which accounts for imbalanced class distributions.

Figure 2 shows the uneven distribution of the Target Feature

As the data was not given with prior separation, in the Training step, it was necessary to split the test and train sets from the full dataset. This separation was made by shuffling the dataset and then splitting the dataset 80/20 for train and test, respectively.

Then, using the pipeline shown on the last section, with no change to the default parameters we trained the model, then we evaluated its performance on the test set using the balanced accuracy score to account for class imbalance. The classifier achieved a balanced accuracy of 0.72, indicating reasonable performance across both positive and negative sentiment classes.

The classification report, including precision, recall, and F1-score, is summarized in Table 1. The model performed better in predicting positive sentiments, as indicated by higher precision, recall, and F1-score for the positive class.

TABLE I
Classification Report for Sentiment Analysis

Class	Precision	Recall	F1-score	Support
Negative	0.63	0.53	0.57	24,930
Positive	0.87	0.91	0.89	88,756
Accuracy	0.83 (on 113,686 samples)			
Macro avg	0.75	0.72	0.73	113,686
Weighted avg	0.82	0.83	0.82	113,686

In addition, we analyzed the most important words contributing to the classification decisions. Table 2 lists the top 10 most important words, ranked by their importance scores. These words were identified using the log-probability differences between the two classes (positive and negative).

TABLE II
Top 10 Most Important Words for Classification

Word	Importance Score
nonmoney	-6.3757
chiou	-4.9074
mistakesbr	-4.7784
parse	-4.7663
abattoir	-4.7038
tobacman	-4.6710
refundable	-4.6020
productslower	-4.5656
carcinogenicbr	-4.5656
billswhen	-4.5656

IV. Dataset size

We evaluated the effect of dataset size on model performance by downsampling the dataset from 10% to 100% and measuring the classifier's performance at each step. For each fraction, the data was shuffled, split into training and test sets, and evaluated over 10 iterations using the balanced accuracy score.

Analyzing the results, it is shown that it will be very difficult to improve our model solely by increasing the dataset size. This is primarily due to the significant imbalance in the distribution of target values. Therefore, it would be more beneficial to focus on gathering additional negative reviews to improve our model's performance.

V. Topic analysis

The conclusion goes here.

References

[1] J. McAuley and J. Leskovec., "From amateurs to connoisseurs: Modeling the evolution of user expertise through online reviews," *WWW*, 2013.