Amazon Reviews Sentiment Analysis

Gaetan Rieben, Iñaki Iribarren, Martim Teixeira, Zeyu Chen
grieben@ucsd.edu, iiribarrenpenuelas@ucsd.edu, mateixeira@ucsd.edu, zec013@ucsd.edu
University of California – San Diego, Rady School of Management

Abstract

The objective of this project, part of our Analyzing Unstructured Data course, is to design a sentiment analysis system for Amazon reviews. Leveraging the principles and methodologies explored in class, our aim is to develop a system that accurately predicts the sentiment of reviews based on their textual content. This focus on Amazon reviews is driven by a personal interest in understanding consumer feedback and sentiment analysis. This decision led us to seek out a dataset that was rich in user reviews and sentiments, a key first step in the project. Selecting the right dataset was essential, as it would lay the groundwork for all subsequent analysis. Our aim in this exploratory phase is to delve into the nuances of user feedback on products, examining elements like sentiment trends, word usage patterns, and engagement levels. These insights will be crucial in crafting an effective sentiment analysis system, reflecting my chosen area of focus.

The heart of the project is the construction of the sentiment analysis system itself. This involves applying natural language processing (NLP) and machine learning techniques to predict the sentiment conveyed in review texts. Additionally, the system is designed to classify reviews into categories such as positive, negative, or neutral, enhancing the analysis process and allowing for more nuanced and accurate sentiment predictions. The model's success will be gauged by its accuracy in reflecting users' sentiments as expressed in their reviews.

Throughout the report, we will detail our approach to data preprocessing, feature extraction, and model development. We will also benchmark the performance of our model against standard metrics to ensure its effectiveness. Additionally, we plan to review relevant literature, comparing our approach and findings with existing models and techniques in the field.

In the following sections, we will present a comprehensive narrative of my process in creating a sentiment analysis system for Amazon reviews, highlighting both the challenges encountered and the insights gained.

Dataset

Structure

For this analysis, we leveraged the Amazon Reviews dataset, meticulously compiled from customer feedback on Amazon products, with the specific aim of facilitating text classification. This dataset is an extensive collection of customer reviews, intentionally curated for the field of natural language processing and sentiment analysis due to its rich content and structured format.

The dataset categorizes reviews into three sentiment classes, negative, neutral, and positive reviews. These categories form the basis for a multiclass classification problem, aiming to accurately classify reviews based on their sentiment. To this classification task we will use the sentiment and cleaned review to predict the sentiment

Content and Use of Files

The dataset is consolidated into a single CSV file, streamlined to facilitate ease of access and analysis. This file is organized with a clear header row that outlines its comprehensive structure, embodying a unified resource for sentiment analysis and text classification tasks.

The Amazon Reviews dataset is structured to prioritize privacy and data consistency, not using IDs for the review. The file amalgamates all necessary data into four pivotal columns, each serving a specific purpose in the sentiment analysis framework:

- **Sentiments**: This column categorizes each review into one of three sentiment classes: Negative, Neutral, and Positive.
- **Cleaned Review**: It contains the text of each review. This process removes noise and irrelevant details, rendering the text ideal for detailed sentiment analysis. The clean text is crucial for models to accurately discern and predict sentiments.
- **Cleaned Review Length**: This column records the length of the cleaned review text. The length is indicative of the review's detail and comprehensiveness, potentially correlating with the strength or clarity of sentiment expressed.
- **Review Score**: Reflecting the original star rating provided by the reviewer on a scale from 1 to 5, this column offers a direct measure of customer satisfaction and sentiment.

This singular CSV file structure simplifies some processes, allowing for straightforward implementation of machine learning and deep learning algorithms aimed at sentiment analysis. By integrating all relevant data into one file, the dataset effectively supports the challenge of classifying Amazon product reviews into distinct sentiment categories, addressing the nuances of consumer feedback and the complexities introduced by class imbalances.

Interesting Findings

An initial exploration of the dataset reveals several interesting trends and patterns. If we look to the distribution of review scores at the Figure 1, it is evident from the chart that the dataset contains a higher number of ratings towards the upper end of the scale, with 5 being the most common rating, followed by 1, and having the scores 4, 2, and 3 as the scores with less instances in the dataset. This suggests a tendency among users to rate movies either very positively or very negatively.

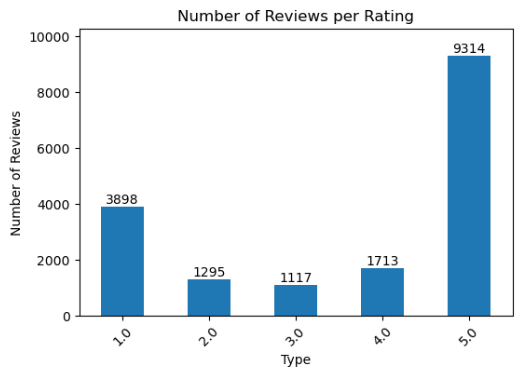

Figure 1 - Distribution of Review Scores

Observing now the distribution of review sentiments at Figure 2, we can observe how the dataset has a higher number of positive reviews, followed by neutral reviews, and having negative reviews as the sentiment category with the least number of observations in the dataset.

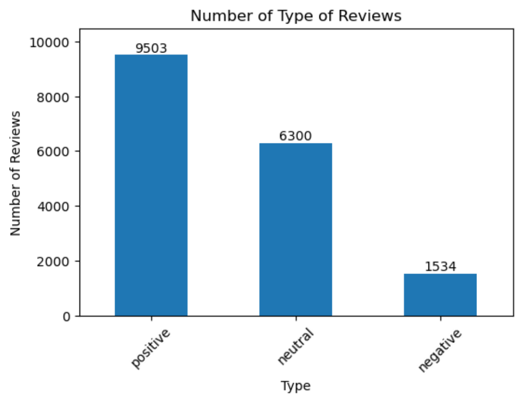

Figure 2 - Distribution of Review Sentiments

To find out what the average review score would be for each of the sentiment categories, we created a bar plot shown in Figure 3, in which we can see the mean review score for each of the three sentiment categories. Since our dataset was unbalanced and there were more positive sentiment instances (9503), than neutral (6303), and negative (1534), we calculated the weighted mean of the whole dataset to get a more representative result. We added the weighted mean to the plot as well, represented by a red dotted line. From the plot, we infer that the positive category is the only one that has an average review score that is above the weighted mean of the three categories. However, both neutral and negative categories' means are close to the weighted mean, which is an interesting finding, as we would expect that particularly the negative category word have a significantly lower review score mean than the other two categories.

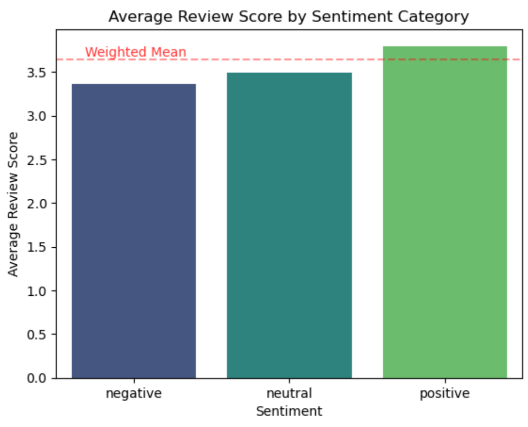

Figure 3 – Average Review Score by Sentiment

After this, we continued our exploratory analysis with a violin plot shown in Figure 4 for the review score distribution for each sentiment category. This violin plot presents the distribution of review scores across three sentiment categories: positive, neutral, and negative. Each "violin" shape encapsulates the density of the scores, indicating more common scores by the width at different score levels. The thickest section of each violin shows the most frequent score range for that sentiment, while the thin points indicate fewer common scores.

In Figure 4, the positive sentiment category displays a concentration towards the higher end of the score spectrum, suggesting that reviews in this category are predominantly favorable. The median score, indicated by the white dot, lies close to the top of the scale, which underscores the generally high scores within the positive sentiment.

For the neutral sentiment, the distribution is somewhat uniform but still shows a slight skew towards higher scores. The median score here is lower than that of the positive sentiment, yet it suggests a trend where reviewers tend to rate experiences as somewhat satisfactory, avoiding positive or negative extremes.

Conversely, the negative sentiment violin has a notable density near the lower score range, affirming the tendency for reviewers to give lower scores when expressing negative sentiment. Interestingly, there is also a thinning in the lower score range, which may indicate that extremely poor reviews are less frequent than negatives.

What stands out across all categories is the presence of a broader score distribution for both positive and negative sentiments, indicating varied degrees within these experiences. This contrast with the narrower shape for the neutral category implies a more consistent scoring within that middle range. From these observations, while reviewers clearly differentiate between positive and negative experiences, the scores within the negative sentiment are not as intensely clustered at the bottom as one might expect. This could suggest that customers are more nuanced in their feedback when dissatisfied. Additionally, the positive sentiment's dense upper range reflects a strong association between positive sentiment and high scores in review behavior.

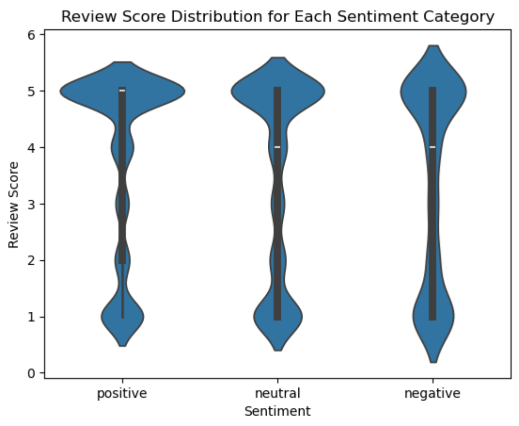

Figure 4 - Review Score Distribution by Sentiment

In our investigation, we sought to understand how the length of reviews varies across different sentiment categories. To visualize this, we constructed a bar plot as depicted in Figure 5, which displays the average review length for negative, neutral, and positive sentiments. By examining the bar plot, we can notice that reviews with negative sentiment tend to be longer on average compared to those with neutral and positive sentiments. This might suggest that customers who had negative experiences are more inclined to provide detailed feedback. In contrast, the average lengths of reviews with neutral and positive sentiments are relatively shorter, which could indicate that customers with satisfactory or pleasing experiences may be less verbose in their feedback.

The insight that emerges from this analysis is particularly intriguing. One might hypothesize that extremely positive or negative experiences would prompt more elaborate commentary, nonetheless, our findings shown in this plot (Figure 5) highlight that it is primarily the negative reviews that are more extensive. Further research could delve into the reasons behind this pattern, perhaps exploring whether individuals are more motivated to elaborate when they wish to warn others or express dissatisfaction, as opposed to when sharing positive recommendation or contentment with an Amazon product.

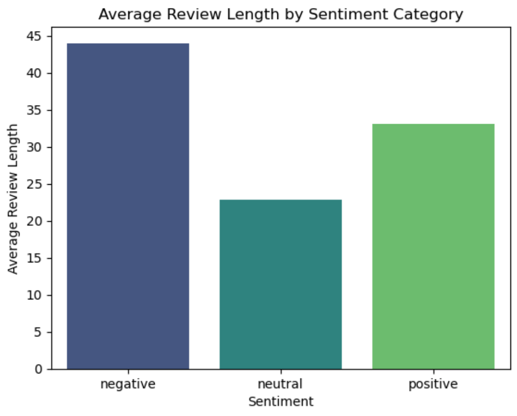

Figure 5 - Average Review Length by Sentiment

After this analysis we wanted to delve deeper into the nuances of customer feedback through the Average Review Length Distribution by Review Score Category, shown in Figure 6. Spanning scores from 1 (indicative of dissatisfaction) to 5 (symbolizing high satisfaction), the bar chart provides a comparative view of the verbosity of customers' reviews. At first glance, the chart depicts a striking uniformity in the average length of reviews across all review scores, a pattern that challenges preconceived notions about the correlation between review length and sentiment.

The uniform lengths suggest that customers write a comparable amount of text to their feedback, regardless of whether their experience was positive or negative. This consistency raises compelling points about the nature of online reviews. It might indicate that customers are equally willing to invest time in detailing their experiences or perhaps that review platforms guide users towards a similar verbosity, regardless of sentiment. One might expect the most dissatisfied customers (scoring 1) to provide lengthy, detailed accounts of their grievances, or conversely, for the most satisfied patrons (scoring 5) to share extensive praise and recommendations. Yet, the data does not show a significant increase in review length at the extreme ends of satisfaction. This could suggest that customers feel a sense of duty or responsibility to provide constructive feedback, regardless of sentiment, which translates into equally comprehensive reviews across the board.

Another aspect to consider is the potential influence of external factors on review length. For instance, the review platform's user interface or word limit constraints might affect review lengths. Moreover, certain product categories or services might naturally elicit more detailed feedback due to their complexity or the stakes involved in the purchase.

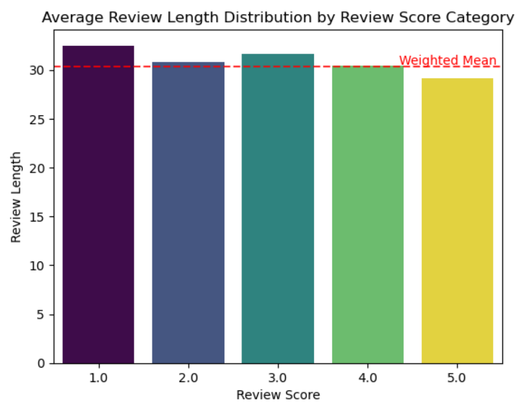

Figure 6 - Average Review Length Distribution by Review Score

Lastly, we wanted to analyze the text of the reviews, so we created a function to get the most common unigrams and bigrams. Figure 7 illustrates the frequency of unigrams, within a collection of textual data, with a particular emphasis on their prevalence in the corpus. The bar chart was generated by tokenizing the corpus into unigrams, counting each unique word's occurrence, and ranking them by frequency. At the apex of this chart is the unigram 'work', standing out as the most frequently mentioned term in the data set.

The prominence of 'work' suggests it is a focal point in the discourse of the corpus, potentially signifying the functional aspects or the performance of the products or services being reviewed. Following 'work', words like 'use' and 'great' also feature notably, likely indicating common topics or sentiments expressed in the reviews. The visualization of these unigrams serves as a foundational analysis for understanding the subjects and opinions most expressed by the users.

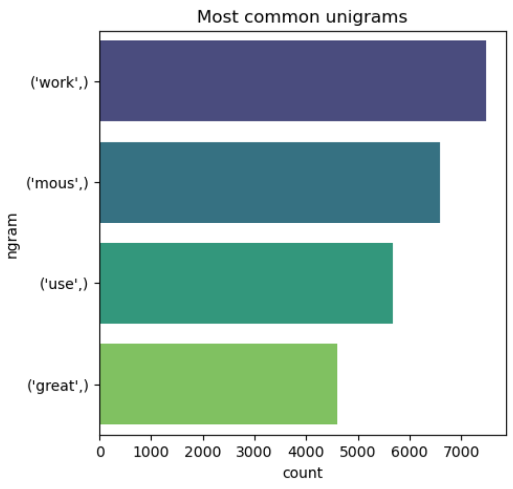

Figure 7 - Most Common Unigrams

Likewise, we also wanted to check for bigrams, which we can observe in Figure 8. Figure 5 offers a clear depiction of the most frequent bigrams, which are pairs of adjacent words, from a corpus of text data. Each bar signifies the count of occurrences for each bigram, with 'stop, work' being the most prevalent combination. The dominance of this bigram might suggest discussions around the functionality of products or services, possibly indicating issues or concerns where items have ceased working.

'Work, great' as a bigram, occupying the third place in terms of frequency, could reflect positive feedback about product performance or customer satisfaction. Meanwhile, 'sound, quality' and 'battery, life' highlight specific product attributes that reviewers frequently discuss, suggesting these are important factors in their evaluation. The chart illustrates these paired terms' prominence, offering valuable insights into the collective focus areas and sentiment of the reviews.

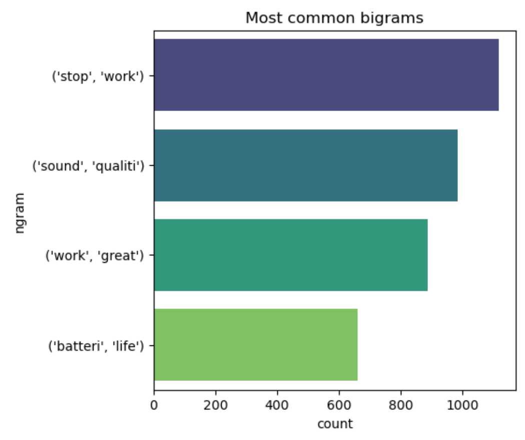

Figure 8 - Most Common Bigrams

Predictive Task

Objective

The aim of this project is to develop a text classification that suggests whether a comment on Amazon is Negative, Neutral or Positive. This objective is realized by transforming the comments into vectors. The methodology combines text pre-processing and text mining techniques that allow to pass the representations into classifier algorithm.

Methodology

Initial Approach

The first stage in the sentiment analysis involves pre-processing the comments that will format the text to improve the later steps. This approach tokenizes the comments and removes the stop words. For a better analysis, we reduced inflectional forms of a word to a common base, using NLTK library.

Text Representation

After preprocessing the entire corpus, the tokens must be transformed to make the text data machine actionable. The tokens have to be represented with numerical values. To do so, we used different text representation techniques:

- *Bag-of-words:* We create a vector space model where each term defines one dimension where each term as a weight. We decided to use a binary vector, a term frequency vector and a term frequency-Inverse Document Frequency (TF-IDF)
- *Word embedding:* Word embedding techniques, such as Word2Vec, offer a sophisticated alternative to traditional bag-of-words models. Instead of representing text using discrete, sparse vectors where each dimension corresponds to a separate term, word embeddings represent words in a continuous vector space. This method enables the model to capture more nuanced semantic relationships between words, as the embeddings are learned from the actual usage patterns of words in the text corpus.

Sentiment analysis

The sentiment analysis process in this project focuses on supervised machine learning, as we use the labels provided for each comment. Our focus is on logistic regression, using each one of the text representations we have created. Our purpose is to make sense of what representation techniques work best for this objective.

Finally, the model with the highest accuracy, although we do use macro and micro F1, as well as the area under the ROC.

Model Preparation

In our study, we employed a comprehensive text preprocessing pipeline alongside TF-IDF, binary vectorization, Word2Vec, and N-gram models to prepare our dataset for a machine learning task. This preprocessing pipeline is crucial for reducing noise in the data and for distilling the text to its most informative essence.

The initial step in preprocessing involves lowercasing the text and removing punctuation, which helps standardize the dataset by eliminating case sensitivity and the potential bias that punctuation can bring. We also collapse multiple spaces into single spaces to maintain consistency in word separation.

Tokenization is then applied to split the text into individual words, which allows for further refinement through the removal of stop words—common words in the English language that are generally considered to carry little meaningful information about the content of the text.

Stemming, executed using the Porter Stemming algorithm, reduces words to their word stem, or base form. This process aids in grouping together the different inflected forms of a word so they can be analyzed as a single item, increasing the robustness of statistics derived from the text.

Furthermore, lemmatization is applied, which involves morphological analysis of the words aiming to remove inflectional endings only and to return the base or dictionary form of a word, known as the lemma. Unlike stemming, lemmatization relies on correctly identifying the intended part of speech and meaning of a word in a sentence, as well as within the larger context surrounding that sentence, such as a paragraph or a document.

Logistic Regression Description

For all the models, we choose Logistic Regression due to its efficiency and effectiveness in binary and multiclass classification problems. The `max_iter` parameter was set to 10000 to ensure convergence, the `solver` was set to 'lbfgs' which is an optimization algorithm suitable for large datasets, and `C`, the inverse regularization strength, was set to 1, providing a balance between overfitting and underfitting. The model also utilized L2 regularization (penalty='l2') to penalize large coefficients and `class_weight='balanced'` to address imbalances in the dataset, which we knew were present from our exploratory data analysis.

Model Description

Word2Vec

This approach used word-embedding method that converts sentences into vectors using the word2vec package from gensim. This method first requires building a vocabulary by using the existing documents. To achieve this, we build the vocabulary using the training data. After building the vocabulary, we defined a function to calculate the word embeddings.

Our goal is to turn sentences into a 100-dimension vector. We tested different dimension sizes, such as 150 and 200, and the size of 100 seems to fit the complexity of the dataset well. From the word embedding retrieval function, the method used is skip gram which predicts the context based on the given target words. We choose skip gram over CBOW because the data set is relatively large and based on the context of the data set on product review, less frequent words that are related to the feature of the product may have a significant meaning. As a result, skip gram is a better choice for our word embedding model. There are two other hyperparameters introduced in the function, which are negative sampling and minimum word count. We set the negative sampling as 7, which means for each positive word, seven negative words will be sampled out. We believe that this size of negative sampling can improve commuting efficiency and potentially improve test results. The minimum word count is set to be 2, which means words appearing less than 2 times will be ignored. The parameter is set to be small, due to the size of the review data. They are generally short, and if the min word count is set too large, significant information may potentially be omitted.

This model displayed a moderate performance with an accuracy of 0.738350, an F1 Score (Micro) of 0.738350, an F1 Score (Macro) of 0.537976, and an AUROC of 0.860225. Its lower performance across these metrics may indicate a less effective capture of contextual information compared to other methods in this specific dataset.

Binary vector

This approach is the simplest instantiation to transform the text into vectors. Using the CountVectorizer from scikit library, it allows us to convert a collection of text documents into a vector of terms.

This method first tokenizes the text into individual words. It splits the text into words based on certain criteria, typically spaces and punctuation. It then creates a vocabulary of all the unique tokens (i.e., unique words) from the entire set of documents. For each document, it transforms the text into a vector (an array of numbers). The length of this vector is equal to the size of the vocabulary. By passing the argument Binary=True, each element in the vector will be 1 if the corresponding vocabulary word is present in the document at least once, and 0 otherwise. With binary encoding, the count is not important; only the presence or absence of the word in the document is noted.

These vectors are then used to train a logistic model on the train dataset. Once trained, the model is used to estimate the accuracy on the validation set to finally roll it out on the test set to see how well it performs.

This technique scored an accuracy of 0.830450, an F1 Score (Micro) of 0.830450, an F1 Score (Macro) of 0.748473, and an AUROC of 0.908270. These results suggest that while it effectively distinguishes classes, it may not capture the semantic richness of the text as well as n-grams.

Term Frequency

Like the previous method, we employ the CountVectorizer from Scikit-learn library. Unlike the binary mode, where each element in the vector is either 0 or 1 indicating the absence or presence of a word, here each element represents the count of times a particular word appears in the document. If a word in the vocabulary is found in the document, the corresponding value in the vector is incremented by the count of that word. If the word is not found, the corresponding value is 0. This results in a vector where each element counts how many times a specific vocabulary word appears in the document.

The TF achieved an accuracy of 0.837659, an F1 Score (Micro) of 0.837659, an F1 Score (Macro) of 0.755311, and an AUROC of 0.907577. The metrics indicate that it is quite adept at predicting outcomes, though it doesn't quite reach the performance peak of n-grams.

N-Grams

For this approach, we are implementing a text classification pipeline that utilizes both machine learning techniques for feature extraction and model training to categorize text data into predefined classes.

Initially, we employ the CountVectorizer with an n-gram range of (1, 2), allowing the extraction of unigrams and bigrams as features from the training, validation, and test datasets. This step transforms the textual data into a numerical format suitable for machine learning models, specifically a sparse matrix representation where each row corresponds to a document and each column to a specific unigram or bigram feature.

After this, we use logistic regression as explained before using the training set to train the model, and then we use the validation set to validate the results, and at the end the test set to see how well our classifier performs.

The results demonstrate the highest performance across the board with an accuracy of 0.848904, an F1 Score (Micro) of 0.848904, an F1 Score (Macro) of 0.761588, and an AUROC of 0.927819, n-grams excels in capturing both the syntactic and semantic structure of the language.

TF-IDF

Once the text is preprocessed, we convert our collection of text documents into a matrix of TF-IDF features. TF-IDF, short for term frequency-inverse document frequency, assigns a weight to each word in a document based on its term frequency and its inverse document frequency—the latter of which diminishes the weight of terms that occur very frequently across the document set and increases the weight of terms that occur rarely. This is particularly important in text classification tasks as it helps highlight the importance of words that are more relevant to the context. The TF-IDF matrix is then used to train a Logistic Regression model presented before.

As the last model, TFIDF showed good overall performance with an accuracy of 0.788351, an F1 Score (Micro) of 0.788351, an F1 Score (Macro) of 0.719327, and an AUROC of 0.919074. Although it factors in word importance, it still falls short compared to n-grams in capturing contextual dependencies.

Literature

On the Kaggle page where we sourced the Amazon Reviews Dataset, a variety of sentiment analysis methodologies are showcased, including advanced techniques such as BERT and simple models like the Dummy classifier. These diverse approaches illustrate the breadth of strategies to extract meaningful insights from customer feedback. While these methodologies offer valuable perspectives, we opted to explore different approaches in our analysis. Specifically, we focused on CountVectorizer, Word2Vec, and TF-IDF as our techniques for feature extraction and text representation. This decision was driven by a curiosity to understand the performance and insights that these alternative methods could provide when applied to sentiment analysis. By diversifying our approach, we aimed to uncover unique insights and potentially uncover strengths and weaknesses of each method when dealing with the complexities of natural language data. This exploration allowed us to compare the effectiveness of traditional text representation techniques against more modern, deep learning-based approaches, thereby enriching our understanding of sentiment analysis on the Amazon Reviews Dataset.

Key Studies and Findings

Text-classification-distilbert-features

In this approach, they applied Dummy classifier as the baseline. This classifier achieved an accuracy of approximately 54.93% on the test set. This performance metric indicates that simply guessing the most frequent sentiment in the training data matches the test data sentiment nearly 55% of the time, highlighting the imbalance or predominant sentiment within the dataset.

Following this, a Logistic Regression model, a more sophisticated algorithm, is trained on the same dataset, achieving a significantly higher accuracy of around 80.22%.

NLPClassification-DeepLearning

For this other approach, they used BERT and LSTM. The LSTM model is particularly suited for this task due to its ability to capture long-term dependencies in text data, making it a powerful tool for understanding the underlying sentiment expressed in reviews. This model is defined with an embedding layer, a spatial dropout layer to prevent overfitting, and an LSTM layer followed by a dense layer for classification.

The result for the BERT is 0.87 and for the LSTM 0.88.

Results comparison

Upon comparing our approach with the first one, where a Dummy classifier and logistic regression were utilized, it becomes evident that we were able to achieve a superior score. This indicates that our method offers more predictive accuracy in this context. However, when our results are juxtaposed with those obtained using advanced models such as BERT and LSTM, our score falls short. This disparity underscores the effectiveness of BERT and LSTM networks in dealing with complex datasets. These models excel in capturing the nuanced relationships between words and the overall sentiment, showcasing their superiority for more sophisticated analytical tasks.

Results and Conclusions

Upon examining the performance metrics for various text feature extraction techniques, it becomes evident that the n-grams representation outshines its counterparts. Let's delve into each metric to understand why.

Accuracy is a straightforward indicator of the overall correctness of the model's predictions. The n-grams method leads this metric with an accuracy of 0.848904, which suggests that when considering the context within text data by capturing sequences of 'n' words together, the model's predictive capabilities are enhanced. This might be due to the fact that n-grams preserve the local order of words, thus holding syntactic and semantic information that single words or binary counts lack.

Moving on to the F1 Score, which considers both precision and recall, it is noteworthy that n-grams once again demonstrate the highest values for both Micro and Macro averaging techniques. The Micro F1 Score of 0.848904 signifies a commendable overall performance across the entire dataset, indicating consistent effectiveness in making individual predictions. On the other hand, the Macro F1 Score of 0.761588, although generally lower due to its focus on achieving balance among classes, highlights the robust performance of n-grams across various classes, not solely the dominant ones.

The exceptional AUROC value of 0.927819 achieved by n-grams solidifies its outstanding performance. AUROC offers a comprehensive evaluation of performance at different classification thresholds, with a score closer to 1 indicating superior class distinction by the model. The impressive AUROC of n-grams signifies its proficiency in managing diverse levels of class differentiation, showcasing its adeptness in handling intricate text patterns.

Despite its ability to capture semantic relationships between words, Word2Vec seems to fall short in this comparison. Its vector representations may not capture critical syntactic nuances that n-grams can. While Countvector Bin and TF methods perform reasonably well, they still do not match up to n-grams. TFIDF, a strong competitor, offers an improved perspective by considering word frequencies adjusted for their inverse document frequency. However, it still does not fully achieve the local context retention that n-grams provide.

To summarize, n-grams excel in all metrics, highlighting the significance of context in text analysis. By treating a sequence of words as a cohesive entity, the n-grams technique can capture intricate patterns within the language, making it especially advantageous in tasks such as sentiment analysis, topic modeling, or when confronted with texts containing abundant jargon or slang. Due to these factors, n-grams emerge as the superior method for extracting text features in this comparative study and are likely to deliver optimal performance in similar textual analysis endeavors.

References

Ihenacho, D. (n.d.). Amazon Reviews Dataset. Kaggle. Available at: https://www.kaggle.com/datasets/danielihenacho/amazon-reviews-dataset/data