Real-time Sentiment Feedback System: Binary Categorization and Context Understanding Based on Amazon Reviews

Arshpreet Buttar
Computer Science
University of Manitoba
Winnipeg, Canada
buttara@myumanitoba.ca

Jiawei Fan
Computer Science
University of Manitoba
Winnipeg, Canada
fanj2@myumanitoba.ca

Roba Geleta
Computer Science
University of Manitoba
Winnipeg, Canada
geletar@myumanitoba.ca

***Abstract*—In the digital age, user reviews are crucial for decision-making and product development. With the growing volume of Amazon product reviews, traditional analysis methods fall short, highlighting the need for smarter, quicker, and more scalable solutions. This study introduces a sentiment analysis framework tailored for Amazon reviews, utilizing n-grams for better context comprehension and enabling exploration of the machine learning model via real-time application. Initial findings show progress in model accuracy by concentrating on positive and negative reviews and fine-tuning hyperparameters. The paper addresses challenges like neutral review categorization, dataset equilibrium, and resource constraints, setting the stage for further enhancements.**

Keywords—Sentiment Analysis, Amazon Reviews, N-grams, Context Understanding, Machine Learning, Hyperparameter Optimization, Text Preprocessing, Dataset Balance.

I. Introduction

In today's digital landscape, user reviews on platforms like Amazon play a crucial role in guiding consumer decisions and influencing product development. The exponential increase in the volume of these reviews has rendered traditional sentiment analysis methods inadequate, highlighting the need for more intelligent, efficient, and scalable solutions [16]. This study introduces a specialized sentiment analysis framework for Amazon reviews, integrating n-grams to enhance context comprehension. The framework is designed to improve the accuracy of sentiment classification by focusing on positive and negative reviews and fine-tuning hyperparameters.

The research also explores the development of a real-time application for the machine learning model, addressing the practical aspects of sentiment analysis. This approach not only aids in understanding user sentiments more accurately but also in translating these insights into actionable strategies for product enhancement. The study confronts several challenges, including the classification of neutral reviews, maintaining dataset balance, and overcoming resource constraints [16]. These issues are critical in ensuring the model's effectiveness and reliability across various products.

This study focuses on a balanced model training approach and the integration of advanced techniques such as n-grams, aiming to develop a versatile and practical solution for digital product owners. Potential real-life applications include improving product recommendations, tailoring marketing strategies, enhancing customer service by identifying and addressing common concerns, and guiding product development based on consumer feedback trends. This approach positions sentiment analysis as a proactive tool for businesses to stay attuned to customer needs and preferences, ultimately driving better customer engagement and business growth.

In our research, we've made key contributions to sentiment analysis. Our primary achievement is the development of an efficient model that leverages n-grams to identify crucial word patterns in positive and negative reviews. Additionally, we've created a real-time application allowing users to test reviews against our model, showcasing its practicality and enhancing user interaction with sentiment analysis.

II. Background and Related Work

Sentiment analysis is vital in discerning consumer opinions in Amazon reviews, employing sophisticated techniques like TFIDFVectorizer and CountVectorizer, along with Logistic Regression and SVM models. This process is essential for interpreting large-scale text data to evaluate customer sentiments accurately.

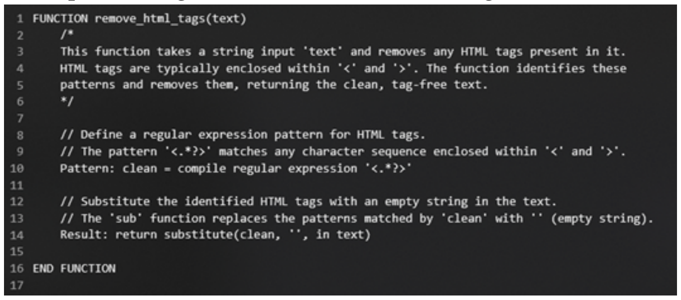
The paper reviews various studies that highlight different approaches in this field. M. Bibi et al. [11] and V. Vyas and V. Uma [14] focused on binary classification of tweets, employing binary-clustering frameworks with a reported accuracy of 79%. However, such low accuracy scores raise concerns about their reliability, particularly for diverse datasets like Amazon reviews. N. S. Gupta et al. [12] used representative terms for binary document classification, reflecting the nascent stages of sentiment analysis with an accuracy of 80%, indicating room for improvement.

M. Bouazizi and T. Ohtsuki [13] tackled the complexity of Amazon reviews with a multi-class sentiment analysis and a pattern-based approach, achieving 87% accuracy. Nevertheless, their methodology in data collection and classification strategies, involving manual processes, could introduce bias and errors. Y. A. Amrani et al. [15] combined Random Forest and SVM, achieving 83.4% accuracy, showing a promising direction for improved analysis accuracy. Interestingly, their experiments with single models, which achieved 82.4% accuracy, suggest that refining models like SVM might be sufficient without combining them with other models.

III. Methodology

A. Data Preprocessing and Textual Analysis

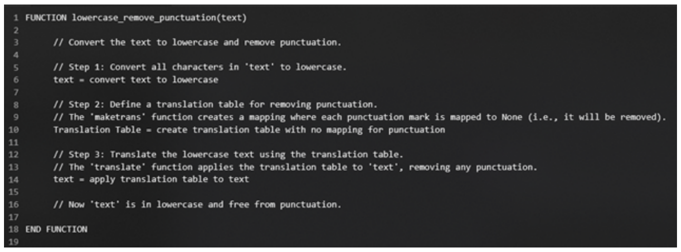
The review dataset from Kaggle included HTML tags that needed removal to avoid introducing irrelevant features and noise. These tags, lacking semantic value, could mislead models to focus on non-essential elements. Removing HTML tags was crucial for cleansing the text data of web-based markup, allowing models to focus on meaningful content.



```
FUNCTION remove_html_tags(text)
    /*
    This function takes a string input 'text' and removes any HTML tags present in it.
    HTML tags are typically enclosed within '<' and '>'. The function identifies these
    patterns and removes them, returning the clean, tag-free text.
    */

    // Define a regular expression pattern for HTML tags.
    // The pattern '<.*?>' matches any character sequence enclosed within '<' and '>'.
    Pattern: clean = compile regular expression '<.*?>'

    // Substitute the identified HTML tags with an empty string in the text.
    // The 'sub' function replaces the patterns matched by 'clean' with '' (empty string).
    Result: return substitute(clean, '', in text)

END FUNCTION
```

Figure 1. Pseudocode showing how HTML tags are removed from the review text

We also normalized all text to lowercase to address word representation inconsistencies, like between "Good" and "good". These inconsistencies could fragment the understanding of word frequencies and sentiments, affecting the models' learning efficiency. Furthermore, we removed punctuation to streamline the feature space as it typically adds zero significant semantic value for sentiment analysis, thus introducing unnecessary features and increasing the model complexity.



```
FUNCTION lowercase_remove_punctuation(text)

    // Convert the text to lowercase and remove punctuation.

    // Step 1: Convert all characters in 'text' to lowercase.
    text = convert text to lowercase

    // Step 2: Define a translation table for removing punctuation.
    // The 'maketrans' function creates a mapping where each punctuation mark is mapped to None (i.e., it will be removed).
    Translation Table = create translation table with no mapping for punctuation

    // Step 3: Translate the lowercase text using the translation table.
    // The 'translate' function applies the translation table to 'text', removing any punctuation.
    text = apply translation table to text

    // Now 'text' is in lowercase and free from punctuation.

END FUNCTION
```

Figure 2. Pseudocode showing how lowercase and punctuation removal is applied on the review text

These pre-processing steps are important because if they are not applied or applied improperly, they can result in the following:

Reduced Model Accuracy: Inadequate preprocessing might cause models to learn from irrelevant features (like HTML tags), leading to inaccurate sentiment interpretation.

Increased Model Complexity: Improper normalization enlarges and complicates the feature space, raising computational demands and reducing model efficiency.

Risk of Overfitting: Models trained on poorly preprocessed data risk overfitting, performing well on training data but poorly on unseen data due to learning from noise and irrelevant features [1].

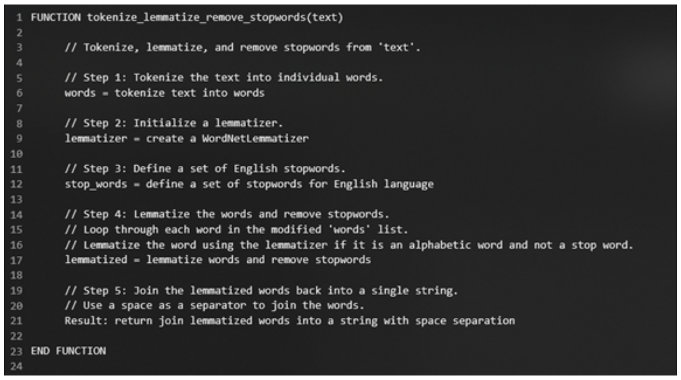
Misinterpretation of Data: Neglecting steps like lowercasing could lead to misinterpreting data, treating the same word with different cases as different features, and resulting in inaccurate sentiment analysis.

B. Tokenization, Lemmatization, and Stop Word Removal

Tokenization: An important process that involves breaking text into individual words or tokens. It's fundamental for machine learning models, allowing them to analyze each word as a separate feature, essential for understanding frequency and sentiment correlation [2]. We used Python NLTK.tokenize library's word_tokenize function to achieve this.

Lemmatization: A key step in text preprocessing, lemmatization reduces words to their base or dictionary forms, such as turning "running" into "run". Unlike stemming, lemmatization ensures the reduced form is a valid word while maintaining the semantic meaning of the text. By generalizing words to their base forms, lemmatization helps the model focus on core meanings rather than different morphological forms [3]. Without proper lemmatization, models might treat different forms of the same word as separate, scattering sentiment signals. We used Python NLTK.stem library's WordNetLemmatizer function to achieve this.

Stop Word Removal: This process involves removing common words that carry little to no sentiment value, such as "the", "is", and "in". These words are filtered out to reduce the dimensionality of the feature space, improving model efficiency. Focusing on sentiment-laden words enhances the model's accuracy in sentiment classification [4]. Removing stop words is essential because they can introduce increased computational complexity (as more features/words need processing), diluted sentiment signals (since stop words can overshadow meaningful words), and reduced model performance (due to the model processing many low-value words).



```
FUNCTION tokenize_lemmatize_remove_stopwords(text)

    // Tokenize, lemmatize, and remove stopwords from 'text'.

    // Step 1: Tokenize the text into individual words.
    words = tokenize text into words

    // Step 2: Initialize a lemmatizer.
    lemmatizer = create a WordNetLemmatizer

    // Step 3: Define a set of English stopwords.
    stop_words = define a set of stopwords for English language

    // Step 4: Lemmatize the words and remove stopwords.
    // Loop through each word in the modified 'words' list.
    // Lemmatize the word using the lemmatizer if it is an alphabetic word and not a stop word.
    lemmatized = lemmatize words and remove stopwords

    // Step 5: Join the lemmatized words back into a single string.
    // Use a space as a separator to join the words.
    Result: return join lemmatized words into a string with space separation

END FUNCTION
```

Figure 3. Pseudocode showing how tokenization, lemmatization and stop word removal is performed on review text

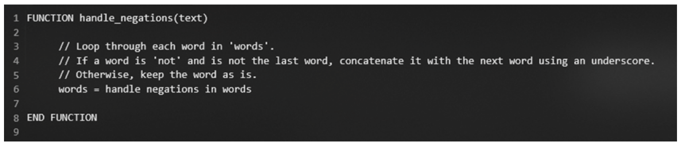
C. Handling Negations

We also made sure to account for the accurate handling of negations, like the word "not", as this step is crucial in sentiment analysis. Negations can significantly change a phrase's sentiment, as seen in the difference between "good" and "not good" which represent opposite sentiments.

Context Preservation: Handling negations ensures the preservation of context, essential in sentiment analysis. For example, transforming "not good" into "not_good" retains the contextual relationship between the negation and the negated word.

Avoiding Misinterpretation: Proper negation handling prevents significant misinterpretations. A model that doesn't recognize "not good" as a negation might incorrectly classify it as positive, failing to understand the inverse sentiment.

Nuanced Sentiment Understanding: Sentiment analysis involves grasping the subtleties of human language, not just keyword detection. Negation handling enables models to understand these nuances, distinguishing between fine shades of meaning [5].



```
FUNCTION handle_negations(text)

    // Loop through each word in 'words'.
    // If a word is 'not' and is not the last word, concatenate it with the next word using an underscore.
    // Otherwise, keep the word as is.
    words = handle negations in words

END FUNCTION
```

Figure 4. Pseudocode showing how negations are handled for review text

D. Sentiment Categorization

In our sentiment analysis model, we opted for a binary classification approach, categorizing sentiments based on star ratings. Specifically, we classified ratings of 4 and 5 as 'positive', and ratings of 1 and 2 as 'negative'. Notably, we made a conscious decision to exclude neutral 3-star ratings from our analysis. This decision was driven by several key considerations, aiming to enhance the model's effectiveness and relevance:

Focus on Extremes for Clearer Insights: By concentrating on the extreme sentiments (either highly positive or negative), our model taps into the most expressive and opinionated customer feedback. These reviews often contain specific and actionable insights, unlike neutral reviews which might offer vague or mixed sentiments.

Reducing Ambiguity and Complexity: Neutral ratings often encompass a wide range of sentiments, from slightly dissatisfied to moderately satisfied. By excluding these, our model avoids the complexity and potential inaccuracies associated with interpreting these nuanced sentiments. This approach acknowledges the inherent difficulty in accurately categorizing moderate opinions, which might otherwise compromise the model's precision.

E. Feature Extraction and Vectorization

TF-IDF Vectorizer: This method scores each word based on frequency in a document and rarity across all documents. It's effective in sentiment analysis, emphasizing distinctive words in a document but not common across all documents [6].

Count Vectorizer: This simpler method counts word occurrences, representing them as vectors. However, it treats all words equally and doesn't consider word importance or rarity, potentially allowing common, less informative words to dominate [6].

We chose the TF-IDF Vectorizer for its nuanced approach. It considers both word frequency and uniqueness, essential for identifying sentiment-driving words in reviews. It reduces the impact of common, less informative words, focusing on more sentiment-revealing words [6]. Moreover, the vectorizer was configured to extract unigrams and n-grams, capturing phrases and expressions with meanings beyond individual words. This enhances text understanding, allowing more accurate, context-aware sentiment analysis.

F. Model Selection and Training

Logistic Regression: Known for simplicity and interpretability, this algorithm is effective for linear relationships. However, its linear nature limits its ability in complex text analysis where feature-sentiment relationships might be non-linear [7].

Multinomial Naive Bayes: This algorithm is favored for text classification due to its simplicity and efficiency with discrete features. However, its assumption of feature independence can be unrealistic in language data, affecting nuanced sentiment analysis accuracy [7].

Support Vector Machine (SVM): SVM excels in complex classification tasks. It's effective in high-dimensional spaces and can handle non-linear feature-target relationships, making it suitable for natural language complexities [7].

SVM was selected for its accuracy in classifying sentiments, outperforming others during testing. Its ability to handle high-dimensional text data and manage non-linear relationships makes it adept for complex patterns in text data, crucial for nuanced sentiment analysis [7]. However, SVM's computational intensity is a consideration for training time and resources.

To enhance our prediction model, we employed Python Scikit-learn library's GridSearch and hyperparameter tuning. GridSearch methodically evaluated various parameter combinations to find the optimal settings, while hyperparameter tuning fine-tuned these parameters for greater accuracy and efficiency [8].

G. Model Evaluation and N-gram Analysis

We used 10-fold cross-validation for evaluation and examined n-grams to discern sentiment patterns in Amazon reviews.

10-Fold Cross-Validation: This method involves dividing the dataset into ten parts, using each part once as a test set and the others for training. It ensures every data point is used for both training and testing, assessing the model's generalization ability [8]. It's suitable for the diverse sentiments in Amazon reviews, ensuring the model is tested across varied data scenarios and reduce the risk of overfitting the model.

N-gram Analysis: N-grams, sequences of 'n' consecutive words, provide context missed by single words. The model calculates probabilities for these n-grams, analyzing their frequency in positive versus negative sentiments. The TF-IDF Vectorizer aids this process, weighing n-grams based on their importance [9]. This weighting considers not just the frequency of an n-gram in a single review but its significance across all reviews. The probabilities help identify which n-grams indicate positive or negative sentiments.

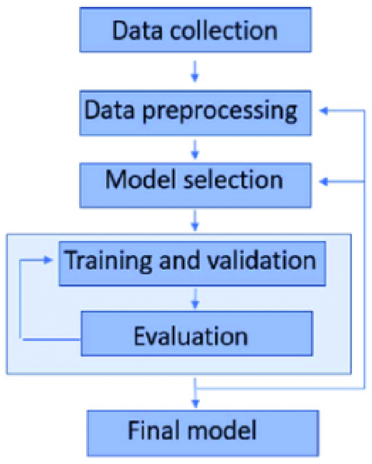


Figure 5. Flow chart showing the development cycle of the machine learning model [10].

H. Output and Application Integration

Our sentiment analysis project culminates in integrating the machine learning model into an application for real-time analysis of reviews. The application allows users to input their custom reviews and quickly get sentiment predictions (positive or negative). To achieve this, we exported the trained model using Python's joblib library, allowing for its reuse in different settings and integrations. For example, the model can be integrated into e-commerce sites for immediate analysis of customer feedback, or in social media tools for assessing public opinion on various topics.

I. Conclusion

Our sentiment analysis project's methodology section thoroughly outlines the development of our machine learning model. Preprocessing steps established a solid foundation, preparing clean, standardized data while preserving contextual nuances for accurate analysis. Selecting the SVM algorithm balanced precise sentiment prediction with managing natural language complexities within computational limits. The 10-fold cross-validation and n-gram analysis in model evaluation ensured a comprehensive assessment of the model's real-time analysis capability.

IV. Empirical Evaluation

A. Experimental Setup

The dataset utilized for this research consists of 101,879 user-generated reviews from Amazon US, specifically focusing on the Digital Software category. This category was deliberately selected due to its relevance and linguistic alignment with the types of products and services discussed on

ProductHunt.com. ProductHunt.com is a popular platform for sharing and discussing the latest in mobile apps, websites, hardware projects, and technological innovations. The language and phrases found in reviews for digital software products on Amazon are closely representative of the discourse on ProductHunt.com, making this dataset an ideal candidate for our sentiment analysis research. This close representation is crucial as it ensures the applicability of our findings to real-world scenarios where similar language patterns are used.

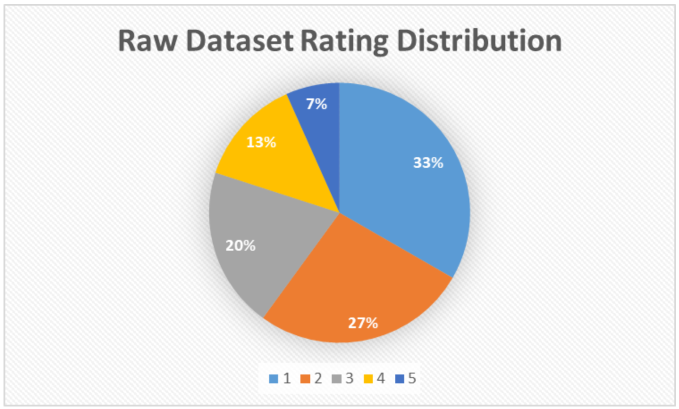


Figure 6. Raw Dataset Rating Distribution

Observed from Figure 6, the original dataset exhibited a significant imbalance in the distribution of review ratings, which is a common issue in real-world data and can lead to biased results in sentiment analysis. To address this, a data preprocessing step was undertaken to balance the dataset, which is critical for ensuring the accuracy and fairness of the sentiment analysis models. We reduced the dataset from its original size to a more manageable and balanced set of 30,000 reviews. The dataset was carefully balanced by sampling an equal number of reviews from each rating category, resulting in three distinct classes: positive, negative, and neutral. Positive reviews (4-5 stars) generally indicate customer satisfaction, emphasizing the software's strengths and advantages. Negative reviews (1-2 stars) often express dissatisfaction, highlighting flaws, bugs, or other drawbacks. Neutral reviews (3 stars), on the other hand, are meant to offer a balanced perspective, mentioning both positives and negatives without strongly leaning towards either sentiment. This approach ensures a comprehensive and varied representation of user feedback for the software.

This balanced approach allows for a more nuanced understanding of customer sentiment, as each class is adequately represented. The balanced dataset serves as a foundation for our initial experiment, enabling us to assess the strengths and weaknesses of our sentiment analysis methods across a diverse range of opinions.

a) *Initial Multi-Class Classification Experiment:*

The initial experiment involved segmenting a dataset into three classes based on star ratings for sentiment analysis: 11,250 positive reviews (4-5 stars), 11,250 negative reviews (1-2 stars), and 7,500 neutral reviews (3 stars), visualized in Figure 7. The methodology focused on applying text vectorization techniques like CountVectorizer and TfidfVectorizer and testing various machine learning models including Multinomial Naive Bayes, Logistic Regression, and SVM. This varied approach allowed us to assess the performance of each combination of vectorizer and model, thereby identifying the most effective method for sentiment classification in our dataset.

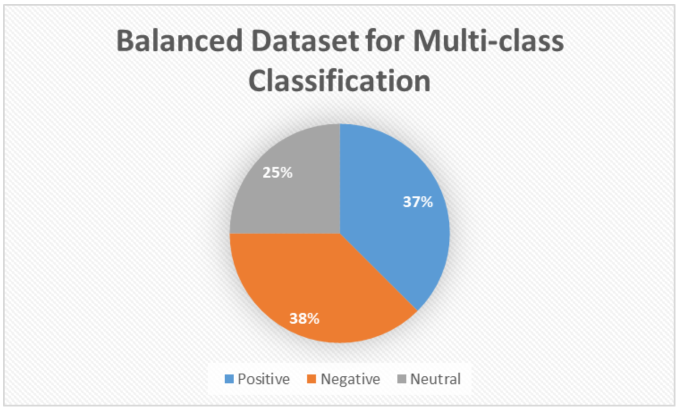


Figure 7. Balanced Dataset Rating Distribution for Multi-Class Classification

b) *Binary Classification Experiment:*

In the binary classification experiment, the focus shifted to exclude neutral reviews, thus concentrating on clear positive and negative sentiments. Shown in Figure 8, the dataset was adjusted to include 22,500 reviews, split evenly between 11,250 positive and 11,250 negative reviews, to enhance sentiment analysis by removing the ambiguity of neutral reviews. The methodology involved 10k-fold cross-validation, a rigorous statistical method that divides the data into segments, trains the model on all segments but one, and tests it on the remaining segment. This process is repeated to ensure thorough evaluation and fine-tuning of model parameters, especially the C parameter in the SVM model, aiming for high accuracy and good generalization to new data.

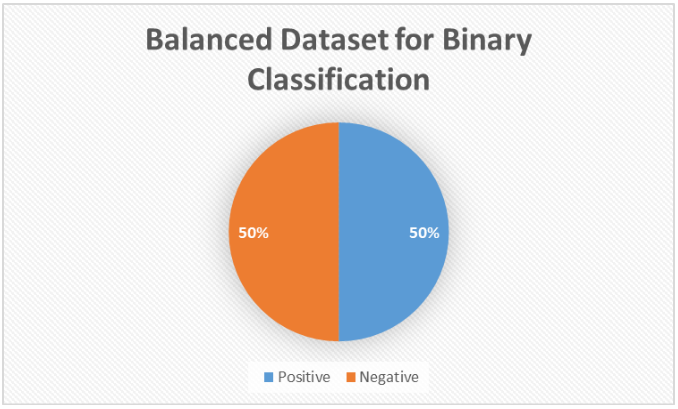


Figure 8. Balanced Dataset Rating Distribution for Binary Classification

B. Results and Analysis

a) Multi-Class Classification Results:

The most notable outcome from the multi-class classification experiment was the performance of the SVM model when combined with the TfidfVectorizer. This combination emerged as the best-performing pair, achieving an accuracy score of 0.6748, Figure 9. This metric is particularly significant as it reflects the model's overall ability to correctly classify reviews into positive, negative, and neutral categories. The TfidfVectorizer, which stands for Term Frequency-Inverse Document Frequency, proved effective in converting the text data into a format that the SVM could process efficiently. This vectorizer emphasizes words that are more unique to a document, thereby helping the SVM model discern subtle differences in sentiment. The success of this combination can be attributed to the SVM's robustness in handling high-dimensional data and its effectiveness in classification tasks, especially when paired with an adept vectorizer like TfidfVectorizer.

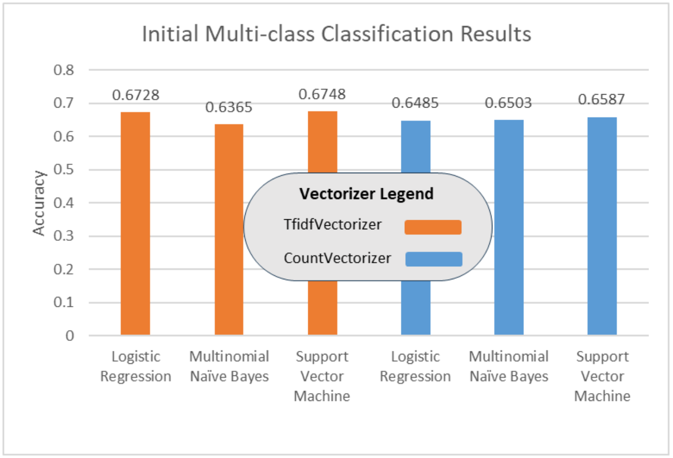


Figure 9. Accuracy Results for Initial Multi-Class Classification Experiment

Despite the success of the SVM with TfidfVectorizer combination, the experiment faced significant challenges, particularly in accurately classifying neutral reviews. This difficulty was evident in the relatively low precision and F1-scores for the neutral category, observed in the classification report in Figure 10. Precision measures the proportion of true positive predictions among all positive predictions [17], and a low precision score for neutral reviews indicates that the model often incorrectly identified reviews as neutral. Similarly, the F1-score, which is the harmonic mean of precision and recall [17], was also low for neutral reviews. This low F1-score suggests that the model struggled not only in correctly identifying neutral reviews (precision) but also in capturing all the relevant neutral reviews (recall) [17].

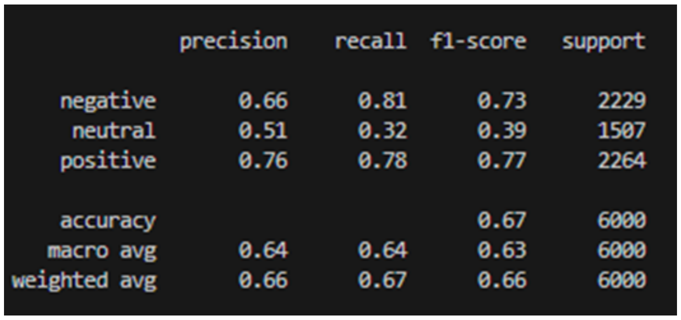

```
              precision    recall  f1-score   support

    negative       0.66      0.81      0.73      2229
     neutral       0.51      0.32      0.39      1507
    positive       0.76      0.78      0.77      2264

    accuracy                           0.67      6000
   macro avg       0.64      0.64      0.63      6000
weighted avg       0.66      0.67      0.66      6000
```

Figure 10. Classification Report for Initial Multi-Class Classification Experiment

While a 3-star rating is considered neutral for our experiment, it's essential to recognize that not every 3-star review necessarily reflects a neutral sentiment. The content of these reviews can vary, often leaning towards either positive or negative sentiments. This variability is critical for precise sentiment analysis, as assuming neutrality based on star ratings alone may lead to incorrect interpretations of consumer feedback. Additionally, given that the focus of this report is on binary classification, insights from the multi-class experiment have informed our decision to proceed by excluding neutral reviews. This step is pivotal in enhancing the clarity and effectiveness of our binary classification approach, allowing for a more direct comparison between positive and negative sentiments.

b) Binary Classification Results:

A significant finding from the binary classification experiment was the notable improvement in accuracy scores compared to the multi-class classification results, presented in Figure 11. SVM, when applied to the binary classification task, achieved a remarkable accuracy score of 0.864. Close behind, the Logistic Regression model also showed impressive performance with an accuracy score of 0.861. These results are particularly striking when contrasted with the multi-class classification outcomes, where the highest accuracy was 0.6748.

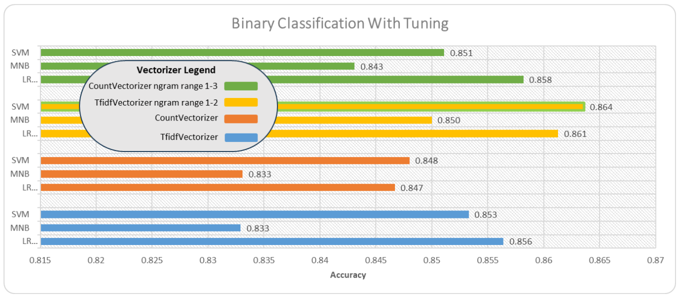


Figure 11. Accuracy Results for Binary Classification Experiment

This increase in accuracy underscores the complexity and challenges involved in multi-class classification, especially when dealing with neutral sentiments. By focusing solely on positive and negative sentiments, the models were able to distinguish between the two extremes more clearly, leading to a more accurate sentiment prediction.

Another critical aspect of the binary classification experiment was the tuning of the C parameter in the SVM with a linear kernel. The C parameter in SVM serves as a regularization parameter, controlling the trade-off between achieving a low training error and a low testing error, which is essentially a trade-off between the model's complexity and its generalization capability [18].

In our experiments, it was observed that the linear kernel SVM performed optimally with a lower C value, demonstrated in Figure 12. This finding indicates that a simpler model (with less regularization) was sufficient and even preferable for this dataset. A lower C value means the model is less penalized for misclassifying training points, leading to a simpler decision boundary that generalizes better to unseen data [18].

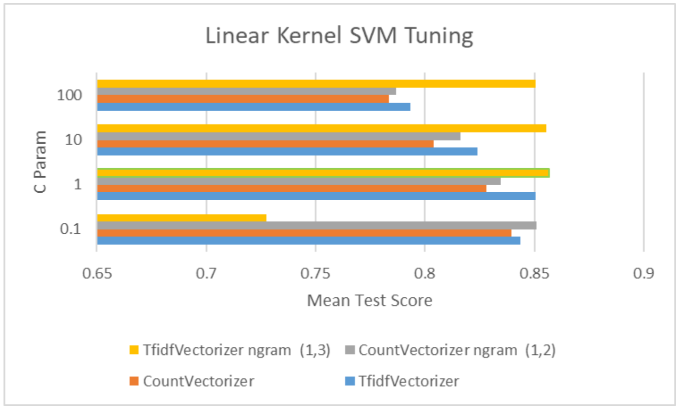


Figure 12. Tuning Results for Linear Kernel Support Vector Machine

The impact of tuning the C parameter effectively demonstrates the importance of hyperparameter optimization in machine learning. It highlights that the choice of hyperparameters can significantly influence the performance of a model and underscores the need for careful consideration and experimentation in choosing these parameters. This insight is particularly valuable for sentiment analysis tasks, where the optimal balance between model complexity and generalization capability is key to accurate and reliable predictions.

c) Impact of N-grams in Vectorizers**:**

As seen in Figure 11, our experiments highlighted the significant benefits of using extended n-gram ranges in TfidfVectorizer and CountVectorizer for sentiment analysis:

TfidfVectorizer (ngram_range=(1,3)): The performance of the TfidfVectorizer improved with an n-gram range of 1 to 3. This configuration enabled the capture of bi-grams and tri-grams, leading to a more nuanced text interpretation. This was particularly effective for phrases where sentiment is contextually dependent, enhancing the model's accuracy in sentiment prediction.

*CountVectorizer (ngram_range=(1,2))***:** Similarly, the CountVectorizer with a bi-gram range (1,2) outperformed its default setting. Including bi-grams allowed for a deeper understanding of word pairs and their contextual sentiment, crucial for accurate classification.

These results highlight the importance of word sequences in sentiment analysis. By extending the n-gram range, the models showed a marked improvement in detecting nuanced sentiments. This improvement underscores the critical role of context in achieving more accurate sentiment interpretation.

C. Comparative Analysis

In the domain of binary sentiment analysis, our system exhibits a noteworthy level of accuracy, aligning well with contemporary standards in the field. Specifically, it achieves an accuracy of 0.864 for SVM and 0.861 for Logistic Regression, positioning it alongside or even ahead of several established benchmarks.

Compared to the other studies, presented in Table 1, our finding of 86.4% accuracy for SVM shows the competitive nature of our system, especially given the reliability demanded in binary sentiment analysis. Even when compared to the findings of M. Bouazizi and T. Ohtsuki. (2016), ours stands as the more reliable approach. Their approach of manually classifying tweets into multiple classes before combining them into two categories introduces a higher risk of subjectivity and human error, potentially impacting the reliability of their 87% accuracy. In contrast, our method, which automatically classifies Amazon reviews based on star ratings, offers a more objective and consistent approach, leading to a robust and reliable dataset, even with a slightly lower accuracy of 86%.

Table 1. Other Findings for Binary Classification.

Source	**Accuracy**	**Disadvantage**
M. Bibi et al. (2020) [11]	79%-SVM	Lower accuracy
N. S. Gupta et al. (2012) [12]	80% - other models	Lower accuracy. Low dataset size.
M. Bouazizi and T. Ohtsuki. (2016) [13]	87%-SVM	Manually Classifying dataset into 7 classes, combining them to 2 classes. Prone to subjective classification and human error.
V. Vyas and V. Uma (2018) [14]	79%-SVM	Lower accuracy. Low dataset size.
Y. A. Amrani et al. (2018) [15]	82.4%-SVM 83.4%-SVMRF	Lower Accuracy. Computationally expensive.

The comparably high accuracy of our system, particularly with SVM and Logistic Regression models, demonstrates its effectiveness and potential for real-world applications in this area.

D. Resource Requirements

As seen in Table 2, The SVM model paired with TfidfVectorizer (n-gram 1,3) emerges as the most accurate but is also the most time-consuming, taking 9,232 seconds. On the other hand, Logistic Regression (LR) with the same vectorizer setting offers a notable balance between efficiency and performance, requiring significantly less time (578 seconds) for a slightly lower accuracy.

The extended n-gram range in the TfidfVectorizer notably increases training time for both SVM and LR, suggesting a trade-off between computational complexity and feature richness. Meanwhile, the CountVectorizer, despite its variations, underperforms compared to TfidfVectorizer.

Overall, while SVM with TfidfVectorizer (n-gram 1,3) is best for accuracy, LR presents a more practical alternative when considering time and resource constraints. This highlights the importance of balancing computational efficiency with model performance in practical applications.

Table 2. Duration of Model Cross Validation and Training on Best Parameters

Time to Cross Validate and Train Model on Best Parameters		
Vectorizer	Model	Time (Seconds)
TfidfVectorizer	LR	33
	MNB	1
	SVM	2,398
CountVectorizer	LR	67
	MNB	1
	SVM	4,026
TfidfVectorizer ngram (1,3)	LR	578
	MNB	4
	SVM	9,232
CountVectorizer ngram (1,2)	LR	299
	MNB	3
	SVM	4,448

V. Conclusion

This study focused on a sentiment analysis system tailored for Amazon reviews. It emphasized binary categorization, separating reviews into positive and negative sentiments, while specifically excluding neutral ones. The methodology included the use of n-gram models, chosen for their ability to better understand context within the constraints of natural language processing. We found that the Support Vector Machine (SVM) model had the highest accuracy out of the 3 models we tested. Finally, a real-time sentiment prediction application was developed to allow users to test the model on their custom reviews.

The conclusion of the paper brings to light certain limitations and areas for future exploration. One significant limitation is the extensive resources required for model training, including time, CPU power, and the numerous trials needed to achieve the final model. Looking ahead, we suggest several avenues for further research. These include determining the optimal dataset size while minimizing decrease in model accuracy (to reduce training data and time), diversifying dataset sources beyond just Amazon reviews to avoid platform bias, experimenting with higher n-gram ranges to improve the application interface, and exploring the combination of different models (such as SVM with logistic regression) to potentially enhance performance and accuracy in sentiment analysis.

References

[1] X. Ying, "An Overview of Overfitting and its Solutions," in Journal of Physics: Conference Series, vol. 1168, no. 2, 022022, 2019, doi: 10.1088/1742-6596/1168/2/022022.

[2] J. Webster and C. Kit, "Tokenization as the initial phase in NLP," in Proc. of the 14th Conference on Computational Linguistics - Volume 4, pp. 1106-1110, 1992. [Online]. Available: https://doi.org/10.3115/992424.992434

[3] I. Boban, A. Doko, and S. Gotovac, "Sentence retrieval using Stemming and Lemmatization with Different Length of the Queries," in Advances in Science, Technology and Engineering Systems Journal, vol. 5, no. 3, pp. 349-354, 2020, doi: 10.25046/aj050345.

[4] D. M. DiPietro, "Quantitative Stopword Generation for Sentiment Analysis via Recursive and Iterative Deletion," arXiv:2209.01519v1 [cs.CL], Sep. 4, 2022.

[5] P. K. Singh and S. Paul, "Deep Learning Approach for Negation Handling in Sentiment Analysis," in IEEE Access, vol. 9, pp. 102579-102592, 2021, doi: 10.1109/ACCESS.2021.3095412.

[6] G. M. Raza, Z. S. Butt, S. Latif and A. Wahid, "Sentiment Analysis on COVID Tweets: An Experimental Analysis on the Impact of Count Vectorizer and TF-IDF on Sentiment Predictions using Deep Learning Models," 2021 International Conference on Digital Futures and Transformative Technologies (ICoDT2), Islamabad, Pakistan, 2021, pp. 1-6, doi: 10.1109/ICoDT252288.2021.9441508.

[7] I. H. Sarker, "Machine Learning: Algorithms, Real-World Applications and Research Directions," in SN Computer Science, vol. 2, no. 160, 2021. [Online]. Available: https://doi.org/10.1007/s42979-021-00592-x.

[8] S. Raschka, "Model Evaluation, Model Selection, and Algorithm Selection in Machine Learning," University of Wisconsin–Madison, Department of Statistics, Nov. 2018, arXiv:1811.12808v3 [cs.LG], Nov. 11, 2020. [Online]. Available: https://arxiv.org/abs/1811.12808v3.

[9] O. Ojo, A. Gelbukh, H. Calvo, and O. Adebanji, "Performance Study of N-grams in the Analysis of Sentiments," in Journal of the Nigerian Society of Physical Sciences, vol. 3, pp. 477-483, 2021. [Online]. Available: https://doi.org/10.46481/jnsps.2021.201

[10] F. Ng, R. Jiang, and J. Chow, "Predicting radiation treatment planning evaluation parameter using artificial intelligence and machine learning," in IOP SciNotes, vol. 1, 014003, 2020. [Online]. Available: https://doi.org/10.1088/2633-1357/ab805d.

[11] M. Bibi, W. Aziz, M. Almaraashi, I. H. Khan, M. S. A. Nadeem and N. Habib, "A Cooperative Binary-Clustering Framework Based on Majority Voting for Twitter Sentiment Analysis," in IEEE Access, vol. 8, pp. 68580-68592, 2020, doi: 10.1109/ACCESS.2020.2983859.

[12] N. S. Gupta, B. Valarmathi, S. Joseph, “Sentiment Analysis Using Representative Terms A Grouping Approach for Binary Classification of Documents”, Journal of Theoretical and Applied Information Technology, pp. 161 – 165, Vol. 44. No. 2, 2012.

[13] M. Bouazizi and T. Ohtsuki, "Sentiment analysis: From binary to multi-class classification: A pattern-based approach for multi-class sentiment analysis in Twitter," 2016 IEEE International Conference on Communications (ICC), Kuala Lumpur, Malaysia, 2016, pp. 1-6, doi: 10.1109/ICC.2016.7511392.

[14] V. Vyas, V. Uma, "An Extensive study of Sentiment Analysis tools and Binary Classification of tweets using Rapid Miner", Procedia Computer Science, Volume 125, p. 329-335, 2018, https://doi.org/10.1016/j.procs.2017.12.044.

[15] Y. A. Amrani, M. Lazaar, K. E. El Kadiri, "Random Forest and Support Vector Machine based Hybrid Approach to Sentiment Analysis",Procedia Computer Science, Vol 127, 2018, P. 511-520, https://doi.org/10.1016/j.procs.2018.01.150.

[16] M. Wankhade, A. C. S. Rao, and C. Kulkarni, "A survey on sentiment analysis methods, applications, and challenges," in Artificial Intelligence Review, vol. 55, pp. 5731-5780, 2022. [Online]. Available: https://doi.org/10.1007/s10462-022-10144-1.

[17] S. Chowdhury and M. P. Schoen, "Research Paper Classification using Supervised Machine Learning Techniques," 2020 Intermountain Engineering, Technology and Computing (IETC), Orem, UT, USA, 2020, pp. 1-6, doi: 10.1109/IETC47856.2020.9249211.

[18] Chychkarov, Y., Serhiienko, A., Syrmamiikh, I., & Kargin, A. (2021). Handwritten Digits Recognition Using SVM, KNN, RF and Deep Learning Neural Networks. International Workshop on Computer Modeling and Intelligent Systems.