A Real-Time Sentiment Feedback System: Binary Categorization and Context Understanding Based on Product Reviews

Arshpreet S. Buttar, Jiawei Fan, Olukoye O. Fatoki, Roba Geleta, Carson K. Leung ✉ [0000-0002-7541-9127]

University of Manitoba, Winnipeg, MB, Canada
✉ Carson.Leung@UManitoba.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. In this paper, we present a sentiment analysis system tailored for Amazon reviews, utilizing n-grams for better context comprehension and enabling exploration of the machine learning model via real-time application. Evaluation results show model accuracy by concentrating on positive and negative reviews and fine-tuning hyperparameters.

Keywords: social network analysis, social network mining, big data analytics, sentiment analysis, Amazon reviews, n-grams, context understanding, machine learning, hyperparameter optimization, text preprocessing, dataset balance.

1 Introduction

In the current era of big data, data are everywhere. Embedded in these data are implicit, previously unknown and useful knowledge that can be discovered by big data science—which makes good use of data mining [1-6], data management [7], machine learning [8], mathematics and statistics [9, 10], and visualization [11, 12]. Big data science has been applied to many real-world application areas like finance and meteorology [13], social media [14-16], as well as social networks [17-22]. This paper focuses on sentiment analysis on product reviews in social media and networks.

In today's digital landscape, user reviews on platforms like Amazon [23] 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 [24]. In this paper, we present a specialized sentiment analysis system for Amazon reviews, integrating n-grams to enhance context comprehension. This system 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 design and 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 [24], which include:

- classification of neutral reviews,
- maintenance of dataset balance, and
- overcoming of resource constraints.

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 a non-trivial 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.

Key contributions of this paper include our design and development of an efficient model that leverages n-grams to identify crucial word patterns in positive and negative reviews. Additionally, we create a real-time application allowing users to test reviews against our model, showcasing its practicality and enhancing user interaction with sentiment analysis.

The remainder of this paper is organized as follows. The next section provides background and related works. Section 3 describes our sentiment analysis system. Section 4 shows evaluation results. Finally, Section 5 draws conclusions.

2 Background and Related Works

2.1 Background of Text Mining and Sentiment Analysis

Text mining or *text data mining* is the extraction of useful information from written text. It involves the implementation of data science and data mining techniques to uncover information from the text that was not previously known [25]. *Sentiment analysis* specifically refers to specifically extracting information about the emotion behind a piece of text. We focus on two broad categories of emotions: positive and negative.

In the creation of a real-time sentiment analysis feedback system, and in the field of text mining, there are many techniques and algorithms to consider. For instance, Moreno et al. [26] discussed various text mining techniques and methods that we use in this project.

Support vector machines (SVM), multinomial naïve Bayes, and logistic regression are three distinct algorithms widely utilized in text mining applications, each with its

unique characteristics and approaches. In text mining, SVM is employed for tasks like document classification. SVM excels in handling high-dimensional data and is known for its effectiveness in scenarios with complex decision boundaries. It works by mapping text features into a multidimensional space, making it especially powerful in tasks where there is not a clear linear separation between classes.

On the other hand, multinomial naïve Bayes is a probabilistic classifier inspired by Bayes' Theorem. In text mining, multinomial naïve Bayes is commonly used for tasks like document classification and sentiment analysis. It calculates the probability of a document belonging to a particular class based on the occurrence of words in the document. Despite its simplistic assumptions, multinomial naïve Bayes often performs well in text classification, particularly when dealing with large datasets.

Logistic regression is a classification algorithm suitable for binary and multiclass classification tasks. In text mining, logistic regression is applied to tasks like sentiment analysis and document categorization. It models the probability of a document belonging to a particular class using a logistic function, and the model parameters are learned through optimization techniques. Logistic regression is interpretable and simple.

In terms of differences in results, SVM is particularly effective when dealing with complex decision boundaries, making it suitable for tasks where linear separation is insufficient. Multinomial naïve Bayes is known for its simplicity and computational efficiency, often performing well in text classification tasks, especially when the independence assumption is a reasonable approximation. Logistic regression, while also simple, is versatile and interpretable, making it valuable for understanding the impact of individual features on classification outcomes. These algorithms collectively extract information from text by transforming it into a numerical format, considering word frequencies, and learning patterns that discriminate between different classes in the training data. The choice of algorithm depends on the specific characteristics of the text mining task and the nature of the dataset at hand.

An element of sentiment analysis that is quite useful in the development of our system is n-gram coefficients. "This is just any combination of adjacent or n-tuples of words or characters that can be found in the text. The fundamental point of n-grams is that they capture the structure of the language from a statistical point of view. There are various n-gram sequences which include a 1-gram (or unigram), 2-gram (or bigram) and 3-gram (or trigram)" [27].

2.2 Related Works

Sentiment analysis is vital in discerning consumer opinions (e.g., in Amazon reviews), employing sophisticated techniques like logistic regression and SVM models, along with scikit-learn tools from open-source machine learning libraries for the Python programming languages (e.g., CountVectorizer, TFIDFVectorizer). This process helps interpret large-scale text data to evaluate customer sentiments accurately.

There are different approaches for sentiment analysis. For example, Bibi et al. [28], as well as Vyas and Uma [29], focused on binary classification of tweets, employing binary-clustering frameworks with a reported accuracy of 79%. However, such low

accuracy scores raise concerns about the classification reliability, particularly for diverse datasets like Amazon reviews. Gupta et al. [30] used representative terms for binary document classification, reflecting the nascent stages of sentiment analysis with an accuracy of 80%, indicating room for improvement.

Bouazizi and Ohtsuki [31] 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. This could be time-consuming, and may introduce bias and errors. Amrani et al. [32] combined random forest with 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.

Shrestha and Nasoz [33] utilized paragraph vectors and product embedding through a combination of gated recurrent unit (GRU) and SVM for classification. The approach demonstrates a notable improvement in accuracy (81.29% to 81.82%) when product embedding was included. As a preview, our sentimental analysis system involves sentiment analysis on similar datasets, as well as interacting extensively with the authors of text. Hence, their findings are incredibly useful in the development of our system.

3 Our Sentimental Analysis System

3.1 Data Preprocessing and Textual Analysis

We use a review dataset[1] from Kaggle as an illustrative example. It includes 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 logical for cleansing the text data of web-based markup, allowing models to focus on meaningful content.

We also normalize all text to lowercase to address word representation inconsistencies (e.g., between “Good” and “good”). These inconsistencies could fragment the understanding of word frequencies and sentiments, affecting the models’ learning efficiency. Furthermore, we remove punctuation to streamline the feature space because it typically adds zero significant semantic value for sentiment analysis, thus introducing unnecessary features and increasing the model complexity.

These pre-processing steps are important because, if they are applied improperly or not applied at all, they would lead to the following issues:

- *Reduced model accuracy*: Inadequate preprocessing might cause models to learn from irrelevant features (e.g., 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.

[1] https://www.kaggle.com/datasets/cynthiarempel/amazon-us-customer-reviews-dataset

Moreover, our system also aims to avoid the following issues:

- *Risk of overfitting*: Models trained on poorly preprocessed data risk overfitting [34], performing well on training data but poorly on unseen data due to learning from noise and irrelevant features.
- *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.

3.2 Tokenization, Lemmatization, and Stop Word Removal

In addition, we apply tokenization, lemmatization, and stop word removal. To elaborate, *tokenization* [35] is an important process that involves breaking text into individual words or tokens. It is fundamental for machine learning models, allowing them to analyze each word as a separate feature, essential for understanding frequency and sentiment correlation. In terms of implementation details, we use Python NLTK.tokenize library package's function called word_tokenize.

Lemmatization [36] is a key step in text preprocessing. It 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. Without proper lemmatization, models might treat different forms of the same word as separate, scattering sentiment signals. In terms of implementation details, we use function Python NLTK.stem library package's function called WordNetLemmatizer.

Stop word removal involves removing common words that carry little to no sentiment value (e.g., article "the", helping verb "is", preposition "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 [37]. 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).

3.3 Handling Negations

We also made sure to account for the accurate handling of negations (e.g., 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.

To elaborate, *context preservation* handles negations, which ensures the preservation of context. It is 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 by proper negation handling, which prevents significant misinterpretations. A model that does not recognize "not good" as a negation might incorrectly classify it as positive, failing to understand the inverse sentiment.

Nuanced sentiment understanding in 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 [38].

3.4 Sentiment Categorization

In our sentiment analysis system, we opt for a binary classification approach, categorizing sentiments based on star ratings. Specifically, we classify:

- ratings of 4 and 5 as "positive", and
- ratings of 1 and 2 as "negative".

Notably, we make a conscious decision to exclude neutral 3-star ratings from our analysis. This decision is driven by several key considerations, aiming to enhance the system's effectiveness and relevance:

We *focus on extremes for clearer insights* by concentrating on the extreme sentiments (either highly positive or negative). Our system 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).

We also reduce *ambiguity and complexity*. Neutral ratings often encompass a wide range of sentiments, from slightly dissatisfied to moderately satisfied. By excluding these, our system 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.

3.5 Feature Extraction and Vectorization

Our system applies *TF-IDF Vectorizer*, which scores each word based on frequency in a document and rarity across all documents. It is effective in sentiment analysis, emphasizing distinctive words in a document but not common across all documents [39].

We observed that, although *count vectorizer* is a simpler method counts and represents word occurrences as vectors, it treats all words equally and does not consider word importance or rarity, potentially allowing common, less informative words to dominate [39]. Hence, 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 [39]. 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.

3.6 Model Selection and Training

Among different models, we select SVM instead of alternative models (e.g., logistic regression, multinomial naïve Bayes). A reason is that *SVM* excels in complex classification tasks. It is effective in high-dimensional spaces and can handle non-linear feature-target relationships, making it suitable for natural language complexities [40]. In contrast, although logistic regression is known for simplicity, interpretability and effectiveness for linear relationships, its linear nature limits its ability in complex text analysis where feature-sentiment relationships might be non-linear [40]. Similarly, although multinomial naïve Bayes is favored for text classification due to its simplicity and efficiency with discrete features, its assumption of feature independence can be unrealistic in language data, affecting nuanced sentiment analysis accuracy [40].

In contrast, SVM is 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 [40]. However, SVM's computational intensity is a consideration for training time and resources.

To enhance our prediction system, we employ Python Scikit-learn library package's GridSearch and hyperparameter tuning. GridSearch methodically evaluates various parameter combinations to find the optimal settings, while hyperparameter tuning fine-tunes these parameters for greater accuracy and efficiency [41].

3.7 Model Evaluation and N-gram Analysis

We use 10-fold cross-validation for evaluation and examined n-grams to discern sentiment patterns in Amazon reviews. To elaborate, *10-fold cross-validation* 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 [41]. It is suitable for the diverse sentiments in Amazon reviews, ensuring the model is tested across varied data scenarios and reducing the risk of overfitting the model.

For n-gram analysis, n-grams (i.e., 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 [27]. 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

3.8 Output and Application Integration

Our sentiment analysis project culminates in non-trivially 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 export the trained model using Python library

package's Joblib, allowing for its reuse in different settings and integrations. For example, the system 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.

3.9 Summary

In our sentiment analysis system, preprocessing steps establishes a solid foundation in preparing clean and standardized data while preserving contextual nuances for accurate analysis. Selecting the SVM algorithm balances precise sentiment prediction with managing natural language complexities within computational limits. The 10-fold cross-validation and n-gram analysis in model evaluation ensure a comprehensive assessment of the model's real-time analysis capability.

4 Evaluation

4.1 Experimental Setup

The dataset[1] 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.

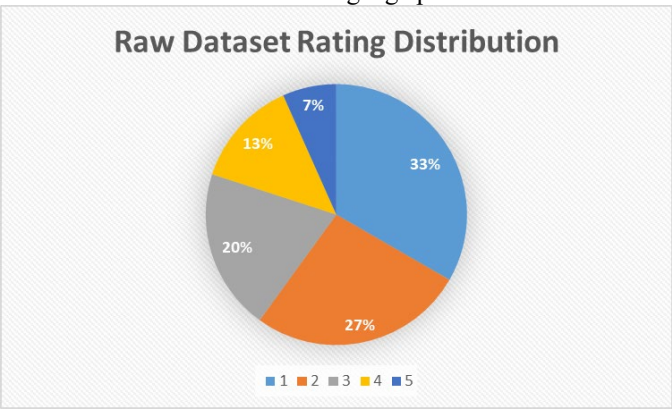

Fig. 1. Raw dataset rating distribution.

Observed from Fig. 1, 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.

4.2 Multi-Class Classification Experiment

The first 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 Fig. 2. The methodology focused on applying text vectorization techniques (e.g., CountVectorizer, TfidfVectorizer) and testing various machine learning models (e.g., multinomial naïve Bayes, logistic regression, 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.

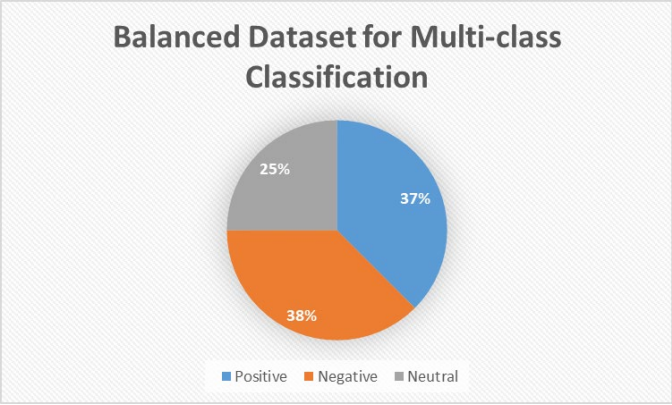

Fig. 2. Balanced dataset rating distribution for multi-class classification.

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,

Fig. 3. 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 converts a collection of raw documents to a matrix of term frequency-inverse document frequency (TF-IDF) features—demonstrated to be 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 Automatic Differentiation using Expression Templates (adept) vectorizer like TfidfVectorizer.

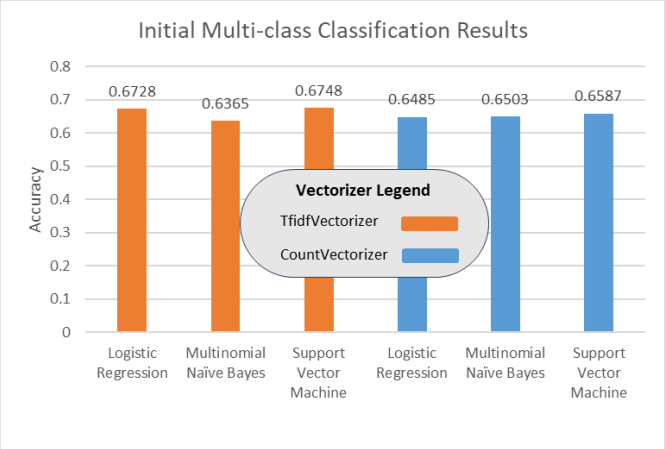

Fig. 3. Accuracy results for initial multi-class classification experiment.

Despite the success of the SVM with the 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 Table 1. *Precision* [42] measures the proportion of true positive predictions among all positive predictions, and a low precision score for neutral reviews indicates that the model often incorrectly identified reviews as neutral. Similarly, the *F1-score* [42]—which is the harmonic mean of precision and recall—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).

Table 1. Classification report for initial multi-class classification experiment.

	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 average	0.64	0.64	0.63	6000
Weight average	0.66	0.67	0.66	6000

While a 3-star rating is considered neutral for our experiment, it is 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.

4.3 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 Fig. 4, 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.

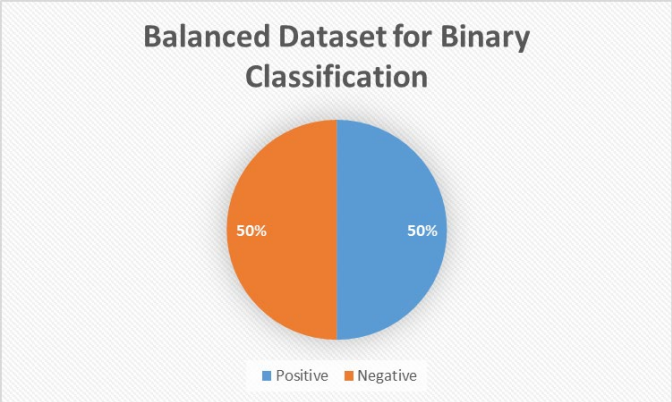

Fig. 4. Balanced dataset rating distribution for binary classification.

A significant finding from the binary classification experiment was the notable improvement in accuracy scores compared to the multi-class classification results, presented in Fig. 5. 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.

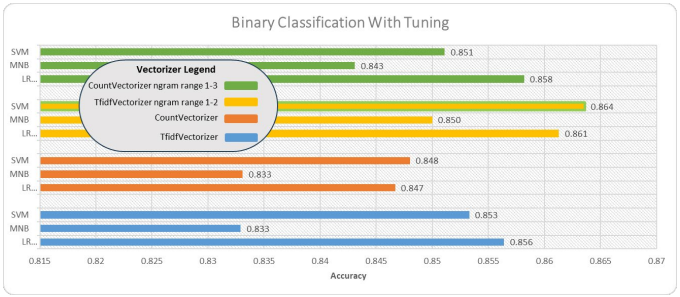

Fig. 5. Accuracy results for binary classification experiment.

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 [43].

In our experiments, the linear kernel SVM is observed to perform optimally with a lower C value, demonstrated in Fig. 6. 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 [43].

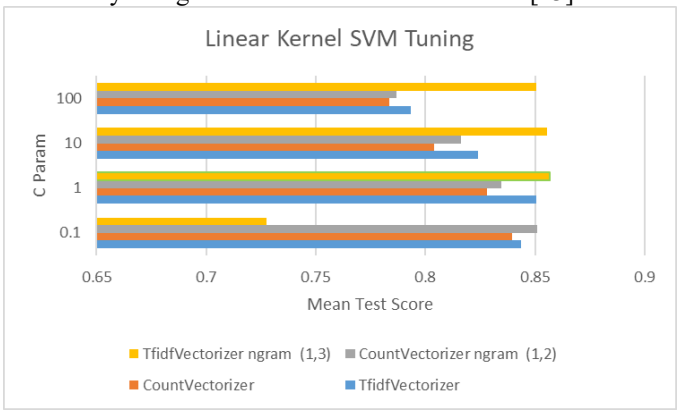

Fig. 6. Tuning results for linear kernel support vector machine.

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.

4.4 Impact of N-grams in Vectorizers

As observedseen in Fig. 5, 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 bigrams and trigrams, 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 bigram 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.

4.5 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 2, 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 Bouazizi and Ohtsuki [31], 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 system—which automatically classifies Amazon reviews based on star ratings—offers a more objective and consistent approach, leading to a robust and reliable dataset, with a compatible accuracy of 86.4%.

Table 2. Other findings for binary classification.

Source	**Model**	**Accuracy**	**Disadvantage**
Bibi et al. (2020) [28]	SVM	79%	Lower accuracy

Gupta et al. (2012) [30]	Other models	80%	Lower accuracy. Low dataset size.
Bouazizi and Ohtsuki (2016) [31]	SVM	87%	Manually classifying dataset into 7 classes, combining them to 2 classes. Prone to subjective classification and human error.
Vyas and Uma (2018) [29]	SVM	79%	Lower accuracy. Low dataset size.
Amrani et al. (2018) [32]	SVM SVMRF	82.4% 83.4%	Lower accuracy. Computationally expensive.

4.6 Resource Requirements

As seen in Table 3, 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 3. Duration of model cross validation and training on best parameters: time to cross validate and train model on best parameters.

Models	**Time (seconds)**		
	LR	**MNB**	**SVM**
TfidfVectorizer	33	1	2398
CountVectorizer	67	1	4026
TfidfVectorizer ngram (1,3)	578	4	9232
CountVectorizer ngram (1,2)	299	3	4448

5 Conclusions

In this paper, we focused on a sentiment analysis framework tailored for user 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 three models we tested. Finally, a real-time sentiment prediction application was developed to allow users to test the model on their custom reviews.

Acknowledgments. This work is partially supported by NSERC (Canada) and University of Manitoba.

References

1. Alam, M.T., et al.: Mining frequent patterns from hypergraph databases. In: PAKDD 2021, Part II. LNCS (LNAI), vol. **12713**, pp. 3-15. Springer, Cham (2021)
2. Choudhery, D., Leung, C.K.: Social media mining: prediction of box office revenue. In: IDEAS 2017, pp. 20-29. ACM (2017)
3. Hryhoruk, C.C.J., et al.: Social network mining and analytics for quantitative patterns. In: IEEE/ACM ASONAM 2023, pp. 717-724 (2023)
4. Hryhoruk, C.C.J., Leung, C.K.: Compressing and mining social network data. In: IEEE/ACM ASONAM 2021, pp. 545-552 (2021)
5. Roy, K.K., et al.: Mining sequential patterns in uncertain databases using hierarchical index structure. In: PAKDD 2021, Part II. LNCS (LNAI), vol. **12713**, pp. 29-41. Springer, Cham (2021)
6. Teng, Y., et al.: Text mining with information extraction for Chinese financial knowledge graph. In: IEEE/ACM ASONAM 2022, pp. 421-426 (2022)
7. Leung, C.K., et al.: Flexible compression of big data. In: IEEE/ACM ASONAM 2019, pp. 741-748 (2019)
8. De Guia, J., et al.: DeepGx: deep learning using gene expression for cancer classification. In: IEEE/ACM ASONAM 2019, pp. 913-920 (2019)
9. Bouguessa, M., Chouchane, A.: A statistical framework for handling network anomalies. In: IEEE/ACM ASONAM 2018, pp. 709-714 (2018)
10. Leung, C.K., Singh, S.P.: A mathematical model for friend discovery from dynamic social graphs. In: IEEE/ACM ASONAM 2021, pp. 569-576 (2021)
11. Barkwell, K.E., et al., Big data visualisation and visual analytics for music data mining. In: IV 2018, pp. 235-240. IEEE (2018)
12. Pham, D., Le, T.M.V.: Utilizing textual reviews for visualizing and understanding user preferences. In: IEEE/ACM ASONAM 2023, pp. 335-339 (2023)
13. Camara, R.C., et al.: Fuzzy logic-based data analytics on predicting the effect of hurricanes on the stock market. In: FUZZ-IEEE 2018, pp. 576-583 (2018)
14. Leung, C.K., et al.: Compression for very sparse big social data. In: IEEE/ACM ASONAM 2020, pp. 659-666 (2020)
15. Pazdor, A.G.M., et al.: Social network analysis of popular YouTube videos via vertical quantitative mining. In: IEEE/ACM ASONAM 2022, pp. 303-307 (2022)
16. Shultz, B.: An entity-aware approach to logical fallacy detection in kremlin social media content. In: IEEE/ACM ASONAM 2023, pp. 780-783 (2023)
17. Hryhoruk, C.C.J., Leung, C.K.: Social network analysis on interpretable compressed sparse networks. In: IEEE/ACM ASONAM 2022, pp. 324-331 (2022)
18. Jain, K., et al.: Modeling behavioral and epidemic dynamics in social contact networks. In: IEEE/ACM ASONAM 2023, pp. 356-363 (2023)
19. Leung, C.K., et al.: Efficient and flexible compression of very sparse networks of big data. In: Big Data and Social Media Analytics, pp. 167-195. Springer, Cham (2021)
20. Leung, C.K., Wen, Q.: Personalized privacy-preserving semi-centralized recommendation system in a social network. In: IEEE/ACM ASONAM 2023, pp. 727-736 (2023)
21. Otudi, H., et al.: Classifying severe weather events by utilizing social sensor data and social network analysis. In: IEEE/ACM ASONAM ASONAM 2023, pp. 64-71 (2023)

22. Singh, S.P., Leung, C.K.: A theoretical approach for discovery of friends from directed social graphs. In: IEEE/ACM ASONAM 2020, pp. 697-701 (2020)
23. Gope, J.C., et al.: Sentiment analysis of Amazon product reviews using machine learning and deep learning models. In: ICAEEE 2022, pp. 52-57 (2022)
24. Wankhade, M., et al.: A survey on sentiment analysis methods, applications, and challenges. Artificial Intelligence Review **55**(7), 5731-5780 (2022)
25. Tan, A.: Text mining: the state of the art and the challenges. In: PAKDD 1999 Workshop on Knowledge Discovery from Advanced Databases, pp. 65-70 (1999)
26. Moreno, A., Carlos, A.I.: Sentiment analysis for social media. Applied Sciences **9**(23), 5037:1-5037:4 (2019)
27. Ojo, O., et al.: Performance study of n-grams in the analysis of sentiments. Journal of the Nigerian Society of Physical Sciences **3**(4), 477-483 (2021)
28. Bibi, M., et al.: A cooperative binary-clustering framework based on majority voting for twitter sentiment analysis. IEEE Access **8**, 68580-68592 (2020)
29. Vyas, V., Uma, V.: An extensive study of sentiment analysis tools and binary classification of tweets using rapid miner. Procedia Computer Science **125**, 329-335 (2018)
30. Gupta, N.S., et al.: Sentiment analysis using representative terms a grouping approach for binary classification of documents. Journal of Theoretical and Applied Information Technology **44**(2), 161-165 (2012)
31. Bouazizi, M., Ohtsuki, T.: Sentiment analysis: from binary to multi-class classification: a pattern-based approach for multi-class sentiment analysis in Twitter. In: IEEE ICI 2016, pp. 4356-4361 (2016)
32. Amrani, Y.A., et al.: Random forest and support vector machine based hybrid approach to sentiment analysis. Procedia Computer Science **127**, 511-520 (2018)
33. Shrestha, N., Nasoz, F.: Deep learning sentiment analysis of Amazon.com reviews and ratings. International Journal on Soft Computing, Artificial Intelligence and Applications (IJSCAI) **8**(1) (2019)
34. Ying, X.: An overview of overfitting and its solutions. Journal of Physics: Conference Series **1168**(2), 022022:1-022022:7 (2019)
35. Webster, J., Kit, C.: Tokenization as the initial phase in NLP. In: COLING 1992, vol. **4**, pp. 1106-1110 (1992)
36. Boban, I., et al.: Sentence retrieval using stemming and lemmatization with different length of the queries. Advances in Science, Technology and Engineering Systems Journal **5**(3), 349-354 (2020)
37. DiPietro, D.M.: Quantitative stopword generation for sentiment analysis via recursive and iterative deletion. arXiv:2209.01519 (2022)
38. Singh, P.K., Paul, S.: Deep learning approach for negation handling in sentiment analysis. IEEE Access **9**, 102579-102592 (2021)
39. Raza, G.M., et al.: Sentiment analysis on COVID tweets: an experimental analysis on the impact of count vectorizer and TF-IDF on sentiment predictions using deep learning models. In: ICoDT2 2021, pp. 38-43 (2021)
40. Sarker, I.H.: Machine learning: algorithms, real-world applications and research directions. SN Computer Science **2**, 160:1-160:21 (2021)
41. Raschka, S.: Model evaluation, model selection, and algorithm selection in machine learning. arXiv:1811.12808 (2020)
42. Chowdhury, S., Schoen, M.P.: Research paper classification using supervised machine learning techniques. In: IETC 2020, pp. 7-12 (2020)
43. Chychkarov, Y., et al.: Handwritten digits recognition using SVM, KNN, RF and deep learning neural networks. In: CMIS 2021, pp. 496-509 (2021)