SENTIMENT ANALYSIS OF AMAZON PRODUCT REVIEWS

Abstract

The sentiment analysis of Amazon product reviews is discussed in this report, including dataset acquisition, dataset preprocessing, model choices, experimental results and final thoughts. The dataset was collected with the help of Apify, which allows scraping a website without Amazon’s restrictions on firewalls, and reviews were preprocessed to solve the class imbalance and use consistent language. Logistic Regression, Support Vector Machines (SVM), and Long Short-Term Memory (LSTM) with both their own and pretrained embeddings (Word2Vec, GloVe) have been tested. Models trained from reviews, including the stop words, were better than those without, showing the impact of these contextual words like negations. However, recall and F1-scores of negative sentiment improved, but class imbalance limited performance. Future work will focus on balancing the dataset and experimenting with alternative embeddings.

1. Introduction

Sentiment analysis is an increasingly important component of consumer opinion formation and business decision-making [1]. Online product reviews are a reliable source of information about product quality and customer satisfaction and are therefore well-suited to sentiment classification tasks [2]. This research aims to analyse Amazon product reviews and classify sentiment as positive or negative, using both traditional machine learning approaches and modern deep learning methods.

The primary objectives of this work are:

1. To create an appropriate preprocessing pipeline for Amazon review data
2. To contrast classic machine learning techniques with deep learning methods.
3. To determine the impact of different text pre-processing methods.
4. To examine the impact of class imbalance on model performance.

2. Dataset Description

2.1 Data Collection

To collect the data necessary for this project, we used APIFY — an online web scraping platform that is acclaimed for hosting robust tools designed to scrape data from websites like Amazon [3]. Amazon has aggressive firewall protections and anti-scraping technology that typically encompass authentication, and therefore, scraping the data directly was not just technically difficult but also potentially unsafe [4]. Using APIFY allowed us to work around these restrictions safely without exposing user credentials or violating Amazon's terms of use.

The data received was in the form of five CSV files of Amazon product reviews. Each review contained significant features such as:
- **reviewTitle:** brief title or summary provided by the user.
- **reviewDescription:** the full text of the review, typically with detailed opinions and experiences
- **ratingScore:** the rating number provided by the reviewer, which ranges from 1 to 5 stars.

2.2 Data Characteristics and Class Imbalance

Initial inspection revealed a broad class imbalance in the data, which was identified during the inspection. Most products on Amazon had average ratings of 4.5 and 5.0 stars, with fewer reviews in the ranges of 1.0, 2.0, 2.5, 3.0, and 3.5 stars. This is because negative ratings tend to discourage potential buyers, resulting in fewer ratings for poorly rated items.

Following a more detailed examination of 3-star reviews, we noted that the majority of them expressed negative sentiment despite the moderate rating. This informed our decision to cluster the ratings into two categories based on sentiment: 1-, 2-, and 3-star ratings were clustered into negative sentiment (0), and 4- and 5-star ratings were clustered into positive sentiment (1).

3. Methodology

3.1 Data Preprocessing

3.1.2 Language Detection and Translation

Since Amazon is catering to a global market, the reviews were written in different languages. To deal with this, each review was first checked for its language through the langdetect library [5]. If a review was not English, it was translated into English automatically using the googletrans library, which is a Python library for the Google Translate API [6]. This converted all reviews to a common language to be further analysed.

3.1.3 Text Cleaning

A variety of cleaning steps were carried out before the text was analysed, in a bid to simplify the reviews for the models to understand. These included lowercasing all the text, removing punctuation, stripping out non-alphabetic characters with regular expressions, and stripping out extra spaces.

3.1.4 Stopword Processing

To better understand the effect of stopwords (like "and," "the," or "is") on sentiment analysis, two instances of the dataset were created—one including stopwords and one excluding stopwords. Stopwords were removed with the NLTK library's English stopword list [7]. That left the door open for comparing models trained with and without stopwords.

3.1.5 Data Storage Strategy

After preprocessing, the clean reviews were saved into separate text files based on whether stopwords were removed or not. Labels were also saved in separate files. Although taking more storage space, this approach gave more flexibility in the event of future testing, retraining the model, or experimentation.

3.2 Traditional Machine Learning Models

3.2.1 Feature Extraction

To turn the text data into something readable by machine learning, TF-IDF (Term Frequency-Inverse Document Frequency) vectorisation was used [8]. TF-IDF indicates important words in a review by balancing how often they appear in a single review with how often they appear in all reviews. Various TF-IDF configurations were tried out to find the best configuration, including:

- Setting the max number of features (words to keep) between 1,000 and 8,000
- Experimenting with different n-gram ranges—from simple single words (unigrams) up to five-word phrases (5-grams)
- Comparing the results with stopwords present versus removed

3.2.2 Model Selection and Training

Three common machine learning models were selected for the sentiment classification task:

1. **Logistic Regression:** Owing to its simplicity, performance, and stability in text classification tasks [9], this model was chosen. The model was trained for at most 1,000 iterations, and a fixed random seed (42) was set so that the output is the same every time it is executed.

2. **Support Vector Machines (SVM):** As a classifier with good performance handling high-dimensional data, this model was an appropriate choice for TF-IDF features. Its parameters were optimised using a grid search:
   - C (regularisation): Trained values of 0.1, 1, and 10
   - Kernel types: Linear and radial basis function (RBF)
   - Gamma: Had 'scale' and 'auto' set to observe their effect on performance

3.3 Deep Learning Models

3.3.1 Text Tokenization and Sequence Preparation

To prepare the text data for deep learning, Keras's Tokenizer from TensorFlow was used to convert the review text to numerical sequences. The tokenizer was configured to use a maximum vocabulary size of 5,000 words, and any word outside this vocabulary was replaced with a special placeholder token (<OOV> for "out-of-vocabulary"). Each review was then translated into a sequence of up to 100 tokens. To have a uniform input length in all reviews, sequences were padded at the end (post-padding), and longer reviews were truncated from the end (post-truncation).

3.3.2 Embedding Strategies

To represent words in meaningful numerical form, four word embedding techniques were used and compared:

1. **Trainable Keras Embedding:** The method used a pre-trained intrinsic Keras embedding layer, which was trained with the model. It allowed the model to learn word representations specific to the dataset for the Amazon review dataset.
2. **Custom Word2Vec Embedding:** A Word2Vec model was trained on the review text using the Gensim library [10]. The model created 100-dimensional word vectors with a skip-gram model (sg=1), window size of 5, and minimum word frequency of 5.
3. **Pre-trained Word2Vec Embedding:** Instead of beginning from scratch, the model utilised Google's pre-trained Word2Vec vectors trained on the Google News corpus. These were then downsized to 100 dimensions to match the rest of the embeddings and ensure consistency across models.
4. **Pre-trained GloVe Embedding:** GloVe vectors, trained on 6 billion tokens in a large corpus, were also employed. The 100-dimensional GloVe embeddings were beneficial to encode global co-occurrence patterns of words [11].

3.3.3 LSTM Model Architecture

It used a Long Short-Term Memory (LSTM) neural network due to its capacity to handle sequential data like text [12]. The architectural design was as follows:

- An input layer accepting sequences of 100 tokens
- An embedding layer (trainable or with pre-trained embeddings)
- An LSTM layer with 64 units and return_sequences=True to maintain temporal information
- A Global Average Pooling layer for reducing dimensions
- A Dense (fully connected) layer with 32 units and ReLU activation
- An output layer with sigmoid activation to perform binary sentiment classification

Training was enhanced with both the Adam and Nadam optimizers having learning rates ranging from 0.00008 to 0.001. Early stopping was exercised with patience between 3 to 10 epochs, and a learning rate reduction policy kicked in when performance on validation plateaus. Binary cross-entropy loss function was utilised, suitable for the two-class sentiment classification task.

4. Experimental Results

4.1 Traditional Machine Learning Models

4.1.1 The Role of Stopwords in Model Performance

Comparative tests on models that were trained both with and without stopwords exhibited tiny differences in accuracy. Adding stopwords surprisingly led to slightly better results. This defies the usual intuition that stopwords add noise. Closer examination revealed that many stopwords in this corpus, such as “not”, “don't”, and “cannot”, were encoding vital sentiment information. Removing them removed vital context, especially in negative reviews. It turned out that stopwords weren't just fillers in such instances—they often held the emotional content of a sentence.

4.1.2 Logistic Regression Performance

Logistic Regression, with TF-IDF features, yielded solid and interpretable performance. Examining the words most impacting the results gave insight into what the model was making decisions based on. Good sentiment was linked with “great”, “good”, “perfect”, and “love” terms, and bad sentiment was driven by “not”, “waste”, “disappointed”, and “refund” terms. These top features reflected real patterns in the data, again indicating that the model wasn't making random guesses—it was detecting useful differences.

4.1.3 SVM Classifier Optimization

Support Vector Machines require proper tuning of hyperparameters. Grid search over different values for C, kernel, and gamma settings showed that linear kernels typically performed best, especially with high-dimensional TF-IDF features. The RBF kernel was sometimes better, but at the expense of longer training time and less stable performance. Linear SVM typically traded off efficiency against accuracy in most cases.

4.2 Deep Learning Results

4.2.1 Comparison of Embedding Strategies

Different word embeddings produced different results, visually and class-wise. t-SNE plots indicated that the in-house-trained Word2Vec model produced tight semantic clusters, which were intuitive about what words should be grouped by. But this did not always translate to better class performance. Pre-trained representations like GloVe and Google's Word2Vec displayed more uniform performance, likely due to the larger training corpus that they were derived from. The Keras embedding, while simple, also performed fairly well, especially when properly tuned.

4.2.2 Handling Imbalanced Classes

One of the main challenges in the deep learning tests was the class imbalance between positive and negative reviews. The model continued to lean towards the majority class, losing many of the subtle patterns in negative instances. LSTM models, with their increased capacity to discover sequence patterns, still had low recall on the minority class. There were some improvements achieved by adjusting the learning rate and using class weighting, yet the performance difference was still evident.

4.2.3 Observations During LSTM Model Training

Training behaviour was conditionally determined from the optimiser, embedding selection, and learning rate. Custom Word2Vec embeddings sped up convergence but had to be very carefully tuned to avoid overfitting. Models trained for excessively long would learn training patterns rather than generalise well, especially at low learning rates. Early stopping and learning rate decay helped avoid this. Otherwise, while the LSTM-based model had some potential, it needed closer monitoring during training than the base models.

5. Discussion

5.1 Key Findings

- **Stopword Retention Improved Accuracy:** Keeping stopwords in the data—i.e., "not," "don't," and "very"—led to slightly improved model performance. Such words often have useful emotional or contextual meaning in sentiment analysis, particularly in the identification of negative sentiment.

- **Traditional Models Performed Competitively:** Traditional models like Logistic Regression and SVM, when paired with good feature engineering like TF-IDF, achieved results comparable to deep learning models. They were less computation-heavy and easier to train and interpret.

- **Class Imbalance Affected All Models:** The dataset was skewed towards positive reviews, and it was harder for models to predict negative reviews correctly. Deep learning models were particularly vulnerable to fitting the majority class.

- **Embeddings Didn’t Always Equal Better Performance:** Pre-trained and custom embeddings varied in capturing word relationships (e.g., as reflected in t-SNE plots), but they did not always correspond to better classification performance. Good semantic clustering did not necessarily equate to better predictive accuracy.

5.2 Limitations

- **Extreme Class Imbalance:** The dominance of positive reviews in the data led to training bias and made models difficult to learn patterns for minority classes like negative sentiment.

- **Quality of Translations:** The non-English reviews were translated with the help of machine tools, which might have introduced errors or altered the original meaning, potentially affecting the model's interpretation.

- **Limited Deep Learning Architectures:** Simple LSTM models were used because of resource constraints. More sophisticated architectures such as Bi-LSTM, GRU, or transformer models were not explored.

- **Narrow Dataset Scope:** The models were trained and evaluated only on Amazon product reviews. Therefore, their performance on other websites or domains is not known and may not be generalised.

6. Conclusion

Throughout this project, we learned that traditional machine learning methods like Logistic Regression and SVMs are still strong when working with sentiment analysis. The most surprising aspect was how much of a difference stopwords made. Instead of removing them as is usually recommended, keeping words like "not," "don't," or "can't" actually enabled the models to better understand the emotional tone of a review, especially in negative sentiment cases. These small words ended up carrying a lot of meaning that helped the classifiers make better decisions.

Deep learning methods, LSTMs in particular, were promising with particular effectiveness at processing sequences and understanding context over time. However, the skewed sentiment review distribution (with many more positive reviews than negative ones) presented problems, especially in learning the model to predict negative reviews accurately. Without enough varied examples on both sides, the model depended too heavily on the majority class. While deep learning performance was not poor, it did not notably outperform traditional methods in this setup, and at considerably higher computational expense.

This experience highlighted that even the less complex models, when supported with the right data and preprocessing methods, can be as good as more complex ones if not more helpful in certain scenarios.

6.1 Recommendations

1. **Improve Data Balance**

One of the biggest problems we faced was the lack of balance between the number of positive and negative reviews. For future work, having or generating more samples of negative reviews would enable the models to view a complete picture of both ends of the sentiment spectrum.

2. **Try Modern Language Models Like BERT**

We were limited to simpler architectures due to time and computational constraints, but it would be a fascinating next step to attempt more powerful models like BERT or RoBERTa in the future. These models are designed to handle the complexity of language better and might be able to give stronger results, especially with more balanced data.

3. **Ensemble Methods**

Instead of choosing between traditional and deep learning models, future work could attempt to fuse them. Some hybrid of the two would perhaps be capable of picking up on more subtle trends and enhancing overall accuracy and generalizability.

4. **Test Across Multiple Domains**

All our data were Amazon product reviews. It would be fascinating to test the models on reviews from other places, like Yelp or TripAdvisor, to see how well they generalise to other kinds of language and contexts.

5. **Build for Real-World Use**

A useful step forward would be to make this a working system—a system capable of categorising reviews in real time. This would be especially useful for businesses looking to stay on top of customer reviews and respond promptly.

Group Contributions

Names	Contribution
Emmanuel Obolo	Data extraction, cleaning, preprocessing, and logistic regression model implementation
Jordan Nguepi	Hyperparameter optimisation for machine learning models and evaluation metric analysis
Jamillah Ssozi	Support Vector Machine (SVM) model development and report writing
Justice Chukwuonye	Long Short-Term Memory (LSTM) model implementation

References

[1]
Y. Mao, Q. Liu, and Y. Zhang, “Sentiment analysis methods, applications, and challenges: A systematic literature review,” *Journal of King Saud University. Computer and information sciences/Mağalaẗ ğam’aẗ al-malīk Saud : ùlm al-ḥasib wa al-ma’lumat*, vol. 36, no. 4, pp. 102048–102048, Apr. 2024, doi: https://doi.org/10.1016/j.jksuci.2024.102048.

[2]
L. Bharadwaj, “Sentiment Analysis in Online Product Reviews: Mining Customer Opinions for Sentiment Classification,” *ResearchGate*, Sep. 03, 2023.
https://www.researchgate.net/publication/373718284_Sentiment_Analysis_in_Online_Product_Reviews_Mining_Customer_Opinions_for_Sentiment_Classification

[3]
“Web Scraping, Data Extraction and Automation · Apify,” *Apify*, 2019. https://apify.com/

[4]
Tekla Kobakhidze, “How to Scrape Amazon Reviews Without Getting Banned in 2025,” *Multilogin*, Oct. 03, 2024. https://multilogin.com/blog/how-to-scrape-amazon-reviews/ (accessed Jun. 07, 2025).

[5]
M. Danilk, “langdetect,” *PyPI*, Oct. 03, 2016. https://pypi.org/project/langdetect/

[6]
S. Han, “googletrans: Free Google Translate API for Python. Translates totally free of charge.,” *PyPI*. https://pypi.org/project/googletrans/

[7]
“NLTK :: Search,” *www.nltk.org*. https://www.nltk.org/search.html?q=stopwords

[8]
N. Ninja, “TF-IDF: Weighing Importance in Text,” *Let’s Data Science*, Jun. 30, 2023.
https://letsdatascience.com/tf-idf/

[9]
Mochammad Rizky Ramadhani, R. D. Putra, Muhammad Dzikri, and Haikal Mufid Mubarok, “Implementation of Plagiarism Detection Using TF-IDF Vectorizer and Machine Learning Based on Flask,” *ResearchGate*, Dec. 21, 2024.
https://www.researchgate.net/publication/387275649_Implementation_of_Plagiarism_Detection_Using_TF-IDF_Vectorizer_and_Machine_Learning_Based_on_Flask

[10]
H. Zhang, “Research on Text Classification Based on LSTM-CNN,” pp. 277–282, Oct. 2024, doi: https://doi.org/10.1145/3708036.3708084.

[11]

J. Pennington, R. Socher, and C. D. Manning, “GloVe: Global Vectors for Word Representation,” *Stanford.edu*, 2014. https://nlp.stanford.edu/projects/glove/

[12]
R. Rehurek, “gensim: Python framework for fast Vector Space Modelling,” *PyPI*. https://pypi.org/project/gensim/

Appendices

Appendix A: Different model configurations

https://docs.google.com/spreadsheets/d/1YLojFlst-nGRykdGIQNkC7_fWPJjrQI9AVLKZh4nTvY/edit?usp=sharing

Appendix B: GitHub Repository

https://github.com/eobolo/Group_14_Sentiment_Analysis_Project.git