Northeastern University

IE7500

Sentiment Analysis on Amazon Fine Food Reviews Dataset

Final Project Report

Jensen Ho, Arundhati Ubhad

2025-04-14

GitHub Submission

Link to GitHub repository: https://github.com/jasonhops/IE7500
Repository is set to public and can be viewed by anyone with access to the link.

Abstract

Sentiment analysis has become a critical task in Natural Language Processing (NLP), enabling machines to interpret human emotions from textual data. This study focuses on extracting and classifying sentiments from product reviews in the Amazon Fine Food Reviews dataset. We explore a comprehensive pipeline involving data preprocessing, feature engineering, and model implementation across traditional machine learning and deep learning approaches. Specifically, we evaluate and compare the performance of Naïve Bayes, Long Short-Term Memory (LSTM) networks, and a transformer-based RoBERTa model. This project highlights the effectiveness of modern deep learning models in sentiment analysis and their ability to generalize across diverse review texts.

Problem Statement

In today's digital age, customer opinions shared through online reviews greatly influence consumer behavior and business decisions. However, the sheer volume of textual feedback makes it impractical to manually interpret sentiments at scale. This project aims to classify the user reviews using Natural Language Processing techniques. The core idea is to build a robust model that can accurately distinguish between **positive** and **negative** sentiments expressed in reviews, despite variations in language, length, and writing style. Addressing this problem involves data cleaning, handling class imbalance, selecting suitable features, and evaluating various classification models to determine the most effective approach for real-world deployment.

Dataset Description

The dataset used in this project is the **Amazon Fine Food Reviews Dataset**, which contains over **500,000** customer reviews for food products.It has reviews from Oct 1999 - Oct 2012. The dataset consists of the following columns:

- **Text**: The full review text
- **Summary**: A short version of the review
- **Score**: Ratings from 1 to 5 (used to derive sentiment labels)
- **User and Product Information**: Metadata about users and products

Sentiment Labeling Approach

Instead of treating only 5 as positive, the sentiment labels were defined as:

- Positive: Reviews with Score ≥ 4
- Negative: Reviews with Score < 4

Data Preprocessing

Before training the models, the dataset underwent essential preprocessing steps. First, missing values were checked and handled appropriately to ensure data integrity.

Regular analysis was performed to understand the distribution of review scores and sentiment labels. The Text and Summary columns were cleaned by converting them to lowercase and removing non-alphabetic characters. Sentiment labels were assigned based on review scores, categorizing ratings ≥ 4 as positive and < 4 as negative. The dataset was then split into training (80%) and testing (20%) sets using stratified sampling. To address class imbalance, the training data was oversampled, ensuring equal representation of 25,000 positive and 25,000 negative reviews. These steps prepared the data for feature extraction and model training.

Project Goals

A trained sentiment classification model capable of categorizing reviews into positive and negative sentiments with a reasonable degree of accuracy. Three types of models will be investigated for suitability including traditional machine learning, deep learning, and transformer. Evaluate performance based on accuracy, precision, recall and confusion matrix. Finally, we will build the model using best practices for data cleaning, feature engineering, and model selection.

Methodology

This project focuses on analyzing sentiment in the **Amazon Fine Food Reviews dataset** using below models:

1. MultinomialNB from scikit-learn

The Multinomial Naïve Bayes (MultinomialNB) model from scikit-learn is a probabilistic classifier based on Bayes' theorem, commonly used for text classification tasks. Since it assumes that word occurrences follow a multinomial distribution, it works well with term frequency-based features. Let's look at the implementation step.

Model Implementation Steps:

- ➢ Feature Extraction using TF-IDF:
  - The TF-IDF Vectorizer (TfidfVectorizer from sklearn.feature_extraction.text) was applied to convert text data into numerical features suitable for Naïve Bayes classification.
- ➢ Training the MultinomialNB Model:
  - The model was trained using the MultinomialNB classifier from sklearn.naive_bayes.
- ➢ Model Evaluation:
  - The trained model was evaluated using accuracy, precision, recall, and F1-score to measure its performance on sentiment classification.

2. NLTK Naive Bayes model

This model assumes that word occurrences are independent. We have used unigram features to train the classifier for predicting sentiment in Amazon Fine Food Reviews.

Model Implementation Steps:

➢ Feature Extraction using Unigrams:
  - The training dataset was tokenized using NLTK’s word tokenizer to extract individual words.
  - Unigram features were generated by considering words that appeared at least 5 times (min_freq=5) to reduce noise.
  - The NLTK Sentiment Analyzer was used to extract unigram features, converting raw text into feature sets.

➢ Training the Naïve Bayes Model:
  - The NLTK Naïve Bayes classifier was trained using the extracted unigram-based features.
  - The model was trained on labeled sentiment data to distinguish between positive and negative reviews.

3. LSTM model

LSTMs can capture long-range dependencies in text, making them effective for understanding contextual sentiment.

Model Implementation Steps:

➢ Tokenization and Sequence Padding:
  - A Tokenizer (Tokenizer from tensorflow.keras.preprocessing.text) was used to convert text into sequences.
  - The vocabulary size was set to 25,000.
  - Sequences were padded/truncated to a fixed length of 500 tokens.

➢ Model Architecture - the LSTM model was built using the Keras Sequential API with the following layers:
  - Embedding Layer : Converts tokenized words into dense vector representations of size 128.
  - LSTM Layer: A 64-unit LSTM layer with dropout (0.2) and recurrent dropout (0.2) for regularization.
  - Dense Output Layer : A fully connected sigmoid activation layer that outputs a probability score for binary classification.
  - The model was compiled with loss function binary_crossentropy and optimizer adam.

4. RoBERTa model

The RoBERTa (Robustly Optimized BERT Pre Training Approach) model is a transformer-based model designed to improve text classification performance by handling complex language patterns more effectively than traditional models.

Model Implementation Steps:

- ➢ Tokenization and Encoding
  - The `Combined_Text` data from the test set was tokenized using the RoBERTa tokenizer with `truncation`, `padding`, and a maximum sequence length of 512 tokens.
    Encoded inputs included `input_ids` and `attention_mask`, which were converted into a PyTorch-compatible `TensorDataset` and wrapped in a `DataLoader` for efficient batch processing.
- ➢ Model Loading and Configuration
  - The `AutoModelForSequenceClassification` class from the Hugging Face Transformers library was used to load the RoBERTa model with 2 output labels for binary classification.
  - The model was loaded onto a GPU (if available) to accelerate inference.
- ➢ Inference
  - The test set was processed in batches (batch size = 32) to avoid memory overload.
  - For each batch, predictions were generated by passing the inputs through the model in evaluation mode (no gradient computation).
  - The logits returned by the model were converted into predicted sentiment classes using `argmax`.
- ➢ Postprocessing and Evaluation
  - Predicted numerical labels (0 for negative, 1 for positive) were mapped back to textual sentiment labels for interpretability.
  - Final performance was evaluated by comparing the predicted labels with the true sentiment labels from the test set

Results and Discussion

Evaluation metrics like ROUGE and BLEU are not appropriate for sentiment analysis. Sentiment analysis is a classification task and not a text generation task. BLEU evaluates **how similar two texts are**, not what **opinion or sentiment** they express. ROUGE is a set of metrics used to evaluate **text summarization and other natural language generation (NLG)** tasks.

We will discuss the overall model evaluation analysis and the insights we gained from this project. NLTK Naïve Bayes is second fastest at training but has the lowest accuracy, with only 0.50. Additionally, its training time was longer than that of Scikit-Learn's MultinomialNB (~30 minutes for NLTK NB vs. ~10 minutes for Scikit-Learn). Given this, we would recommend Scikit-Learn's MultinomialNB if we had to choose between the two Naïve Bayes implementations. Refer to the classification reports table for details. We have also evaluated LSTM and RoBERTa models. RoBERTa had the highest accuracy but also the longest training time of the four models. Finally, our team concluded that BLEU and ROUGE scores are used for evaluating generated text, not for text classification tasks like sentiment analysis.

So, let’s delve deeper into our model evaluation.

NLTK Naive Bayes
This model was fast to implement but yielded the lowest performance, with an accuracy of only **0.50**. Despite its simplicity, the training time was surprisingly longer (at ~30 minutes) than that of Scikit-Learn’s MultinomialNB. Given its poor performance metrics (e.g., F1-score of **0.35** for negative sentiment), we do not prefer this model.

Scikit-learn MultinomialNB
Among the Naive Bayes models, the MultinomialNB from Scikit-Learn performed significantly better. It achieved an accuracy of **0.87**, with strong results for both precision and recall on positive sentiment. It also had the fastest training time across all models (at ~10 minutes), making it a strong baseline model. This would be our preferred choice if limited to traditional models without deep learning or transformer-based methods.

LSTM
LSTM was slower than Naive Bayes algorithm at training with decent accuracy of 0.79. However, recall for negative sentiment was 0.00 with F1-score of 0.01. Therefore, we decided not to select LSTM for the final candidate of models. Also, the training time, at around 55 minutes, is longer than the two Naive Bayes implementations.

RoBERTa
RoBERTa was the top-performing model, achieving the highest accuracy of **0.95** and an impressive overall F1-score of **0.97** for positive sentiment. It also maintained strong performance across both classes. However, this performance came at the cost of longer training time and a need for GPU acceleration. We used a pre-trained RoBERTa model from Hugging Face, which allowed us to leverage transfer learning.

Overall, Sklearn MultinomialNB is suitable for devices where low resource usage is a priority. For example, IoT devices and mobile phones are computing resource limited. However, if computing resources is not an issue like in a datacenter or workstation, then RoBERTa is the best state-of-the-art model for training and prediction of sentiment analysis.

Classification Reports Summary Table

	Accuracy	0-Precision	0-Recall	0-F1 Score	1-Precision	1-Recall	1-F1 Score	Overall
Sklearn Multinomial NB	0.87	0.64	0.88	0.74	0.96	0.86	0.91	0.83
NLTK NB	0.50	0.24	0.67	0.35	0.82	0.47	0.59	0.52
LSTM	0.79	0.35	0.00	0.01	0.79	1.00	0.88	0.54
RoBERTa	0.95	0.90	0.87	0.89	0.96	0.97	0.97	0.90+

Model Resource Usage Observations

Model	Training Time	Prediction Time	Memory Usage	Compute Usage
SK-MultinomialNB	🟢 Very Fast	🟢 Very Fast	🟢 Low	🟢 Low
NLTK NB	🟢 Fast	🟢 Very Fast	🟢 Low	🟢 Low
LSTM	🔴 Slow	🟡 Moderate	🔴 High	🔴 High
RoBERTa	🔴 Very Slow	🟡 Moderate	🔴 Very High	🔴 Very High

We will discuss the performance metrics in this section.

Below performance metrics are used to evaluate the models for sentiment analysis:

- **Accuracy**: Measures the overall percentage of correct predictions.
- **Precision**: Out of predicted positive reviews, how many were actually positive?
- **Recall**: Out of all actual positive reviews, how many did the model identify correctly?
- **F1 Score**: A balance between precision and recall.

Classification Report is the detailed breakdown of precision, recall, and F1-score for each class.

1. **Scikit-Learn MultinomialNB**

```
          precision  recall    f1-score support
negative   0.64          0.88        0.74     2668
positive   0.96          0.86        0.91     9832

accuracy                             0.87    12500
macro avg 0.80           0.87        0.82    12500
```

Observations:

- The model demonstrates high precision (96%) for positive reviews, indicating that when it predicts a review as positive, it is correct most of the time.
- The recall for negative reviews (88%) is high, meaning the model correctly identifies most negative reviews.
- However, precision for negative reviews (64%) is lower, suggesting that some positive reviews are misclassified as negative.
- The F1-score for positive sentiment (0.91) is significantly higher than for negative sentiment (0.74), showing a performance imbalance favoring positive reviews.

Limitations:

- The lower precision for negative reviews suggests the model struggles with identifying negative sentiment accurately.
- Since Naïve Bayes assumes word independence, it lacks contextual understanding, which can impact classification performance on nuanced text.

2. **NLTK NB**

```
              precision    recall  f1-score   support

    negative       0.24      0.63      0.35      2668
    positive       0.82      0.47      0.59      9832

    accuracy                           0.50     12500
   macro avg       0.53      0.55      0.47     12500
weighted avg       0.70      0.50      0.54     12500
```

Observations:

- Precision for positive reviews: 82% (high precision, meaning most predicted positive reviews are correct).
- Precision for negative reviews: 24% (low precision, meaning many negative reviews were misclassified as positive).
- Recall for negative reviews: 63% (indicating the model correctly identifies most negative reviews).
- F1-score for positive sentiment: 0.59, while for negative sentiment it is 0.35, highlighting an imbalance.
- Macro F1-score: 0.47, showing that the model struggles to classify both sentiments equally.

Limitations:

NLTK's NB identified positive reviews but struggled to predict negative reviews. This model took a long time to train >30 minutes and the accuracy was poor. The lack of contextual understanding negatively impacts the usability of this model for sentiment analysis.

3. **LSTM**

```
               precision    recall  f1-score   support

           0       0.35      0.00      0.01      2668
           1       0.79      1.00      0.88      9832

    accuracy                           0.79     12500
   macro avg       0.57      0.50      0.44     12500
weighted avg       0.69      0.79      0.69     12500
```

Observations:

- Precision for positive reviews: 79% (high precision, meaning most predicted positive reviews are correct).
- Precision for negative reviews: 35% (low precision, meaning many negative reviews were misclassified as positive).
- Recall for negative reviews: 0% (indicating the model could not identify any negative reviews).
- F1-score for positive sentiment: 0.88, while for negative sentiment it is 0.01, highlighting an inability to identify negative sentiment.
- Macro F1-score: 0.44, showing that the model struggles to classify both sentiments equally.

Limitations:

The LSTM model with a F1 score of 0.88 is quite good for positive sentiment but is unable to pick out negative reviews well with F1 score of 0.01. The relatively small training set for LSTM and a lack of large pre-training corpus compared to Transformer models severely impacted suitability for sentiment analysis.

LSTM has high computing cost and tends to overfit with small training sets. Long range dependency is not captured well. Scikit-learn MultinomialNB is a better fit for sentiment analysis and the RoBERTa model is even better with its pre-training corpus.

4. **RoBERTa**

```
              precision    recall  f1-score   support

    Negative       0.90      0.87      0.89      2741
    Positive       0.96      0.97      0.97      9758

    accuracy                           0.95     12499
   macro avg       0.93      0.92      0.93     12499
weighted avg       0.95      0.95      0.95     12499
```

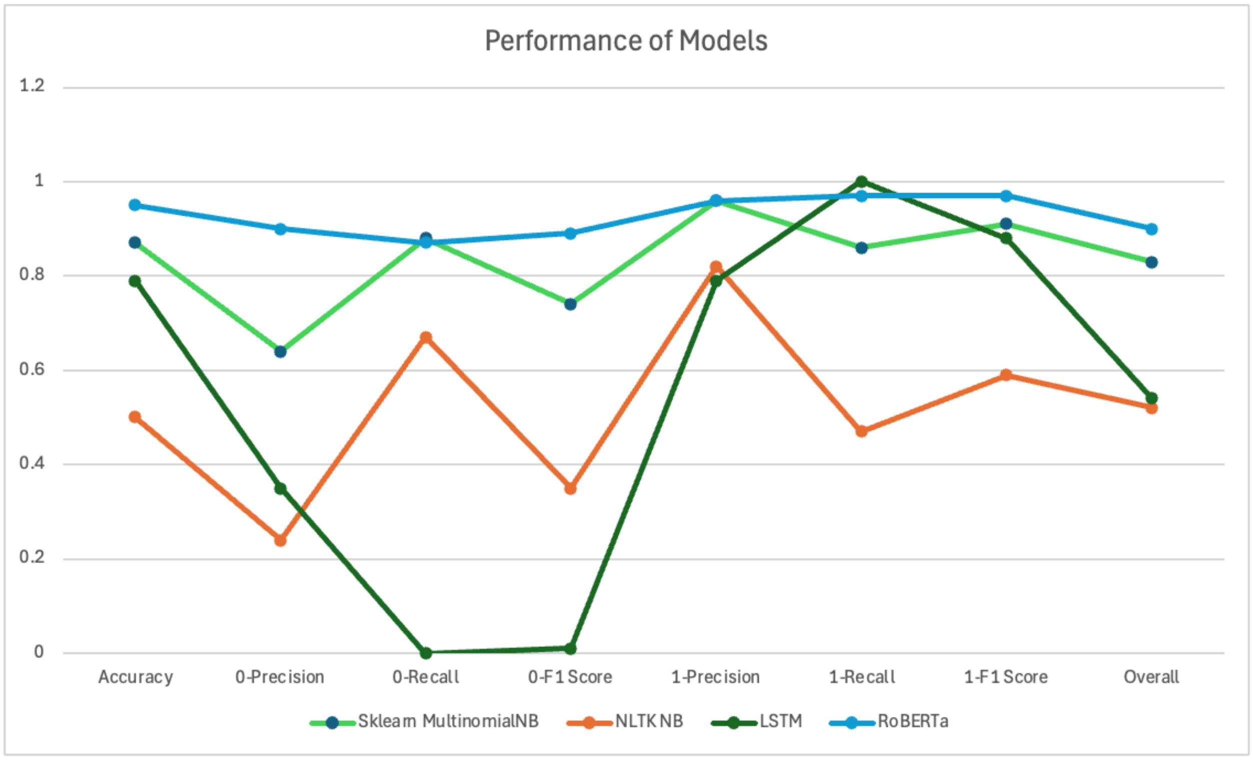
Observations:

- The model performs very well overall with 95% accuracy, with a strong ability to correctly classify both positive and negative sentiments.
- Precision: **0.96**, Recall: **0.97**, F1-score: **0.97** indicates that the model is highly effective at correctly identifying positive sentiment with minimal false positives or negatives.
- The impressive result of the pretrained model is a result of the large dataset that the model was trained on. Recurrent models can also capture long range dependencies that other models can't for context.

Limitations:

- RoBERTa requires powerful GPU hardware for both training and inference due to its large size and complex architecture. This makes it less feasible for deployment in low-resource or real-time environments.

Performance of Models Chart



Conclusion

This project explored sentiment classification on the Amazon Fine Food Reviews dataset using both traditional machine learning and transformer-based deep learning models .Among all the models evaluated, **RoBERTa** demonstrated superior performance, owing to its deep contextual understanding and pretraining on a large corpus. The methodology involved robust data preprocessing, feature extraction using TF-IDF and token embeddings, class balancing via oversampling, and rigorous model evaluation using metrics like accuracy, precision, recall, and F1-score. The results highlight the effectiveness of transformer models in capturing nuanced sentiment from text, especially in the presence of class imbalance and informal language. **Scikit-Learn's MultinomialNB** achieved 87% accuracy, with high precision (96%) for positive reviews and recall (88%) for negative reviews. However, its precision for negative reviews (64%) was lower, suggesting misclassification of some positive reviews, likely due to the Naive Bayes independence assumption limiting contextual understanding. **NLTK's NB** performed poorly, with 50% accuracy and a low F1-score for negative sentiment (0.35), struggling with negative review precision (24%) and lengthy training times (>30 minutes), further hampered by its lack of contextual awareness. The **LSTM** model, with 79% accuracy, excelled at positive sentiment (F1-score 0.88) but failed entirely to identify negative reviews (recall 0%), likely due to a small training set and lack of pre-training, alongside high computational costs and overfitting tendencies.

Thus, while traditional models like MultinomialNB offer efficiency, advanced transformer-based models like RoBERTa provide superior accuracy for sentiment analysis, albeit with greater computational demands, highlighting a trade-off between performance and practicality.

Future Work

We identified two areas that can be explored in the future. Expanding beyond binary classification to include neutral or mixed sentiments would allow for a more nuanced understanding of customer feedback. Finally, identifying sentiments related to specific aspects (e.g., taste, packaging, delivery) could provide more actionable insights to businesses.

Appendices

Citation for paper

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., & Stoyanov, V. (2019). *RoBERTa: A robustly optimized BERT pretraining approach*. arXiv. https://doi.org/10.48550/arXiv.1907.11692

Sak, H., Senior, A., & Beaufays, F. (2014). *Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition*. arXiv preprint arXiv:1402.1128. https://arxiv.org/abs/1402.1128

scikit-learn developers. (n.d.). *sklearn.naive_bayes.MultinomialNB*. scikit-learn. https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html

Bird, S., Klein, E., & Loper, E. (n.d.). *nltk.classify.naivebayes module*. Natural Language Toolkit. https://www.nltk.org/api/nltk.classify.naivebayes.html

Codes for preprocessing and oversampling

This is an overview of the parameters we used as input for the four models.

```
import pandas as pd

df= pd.read_csv("Reviews.csv")
df.head()
df.isna().sum()
df.dropna(subset=["Summary"], inplace=True)
df.dropna(subset=["ProfileName"], inplace=True)
df.info()
df.isna().sum()
duplicates = df.duplicated()
print(duplicates.sum())
duplicates

print(df.isnull().sum())
df[df.isnull().any(axis=1)]

invalid_score = df[(df["Score"] < 1) | (df["Score"] > 5)]
print(invalid_score)

invalid_date = df[(df["Time"] < 938736000) | (df["Time"] > 1351728000)]
print(invalid_date)

df["Text"] = df["Text"].str.lower().str.replace(r'[^a-z\s]', '', regex=True)
df["Summary"] = df["Summary"].str.lower().str.replace(r'[^a-z\s]', '',
regex=True)

from sklearn.model_selection import train_test_split
df['Sentiment_label'] = df['Score'].apply(lambda x: 'positive' if x >= 4 else
'negative')
```

```
test_size = 0.2  # 20% test split
train_size = 0.8  # 80% train split

# Split the test set first
train_df, test_df = train_test_split(df, test_size=test_size,
stratify=df["Score"], random_state=42)
test_df=test_df.sample(n=12500, random_state=42)
train_oversampled_df =
pd.concat([train_df[train_df["Sentiment_label"]=='positive'].sample(n=25000,
random_state=42),
train_df[train_df["Sentiment_label"]=='negative'].sample(n=25000, random_state=
42)], ignore_index=True)
```