# IE7500


## Jensen Ho, Arundhati Ubhad


# Sentiment Analysis on Amazon Fine Food Reviews Dataset

**Link to GitHub repository: https://github.com/jasonhops/IE7500**

---

## Objective:

The objective of this project is to develop a **sentiment analysis** model that classifies customer reviews from the Amazon Fine Food Reviews dataset as **positive** or **negative** based on their ratings. We are using the column 'Score' from the dataset to classify the reviews . Reviews with a score >=4 are considered positive, while reviews with score < 4 are categorized as negative. This project involves multiple Natural Language Processing (NLP) tasks, including:

- **Text Classification:** Training a model to categorize reviews into positive or negative sentiment.
- **Text Preprocessing:** Tokenization, stopword removal and vectorization (TF-IDF, Byte Pair Encoding).
- **Model Development:** Implementing and comparing three different approaches.
    - **Traditional Machine Learning Model:** Using Naïve Bayes with TF-IDF for baseline sentiment classification.
    - **Deep Learning Model:** Fine-tuning RoBERTa (Robustly Optimized BERT Pre Training Approach) for contextual understanding.
    - **Transformer Model:** Implementing LSTM (Long Short-Term Memory) with embedded representations for sequence modeling.

By leveraging TF-IDF for traditional frequency-based feature extraction, BPE for subword tokenization, and transformer-based architectures for contextual representation, this project aims to enhance sentiment classification accuracy and compare the effectiveness of different NLP techniques.

## 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. 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

## 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.

## 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.

**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.

**Observation:**

Accuracy: 0.8669

Classification report:

| | 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 |
| weighted avg | 0.89 | 0.87 | 0.87 | 12500 |

- 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. **Naive Bayes Classifier using NLTK :**

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.

➢ Model Evaluation

- The trained classifier was tested on the test set using accuracy, precision, recall, and F1-score.
- The 10 most informative features were identified to analyze the most influential words in sentiment classification.

**Observation:**

| | 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 |

NLTK's NB did not perform well. 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.

Accuracy: 50%

- 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.

### 3. LSTM

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`.
- Optimizer: `adam`.

**Observation:**

Classification report:

| | 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 |

Test accuracy: 0.7860000133514404

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.

### 4. RoBERTa

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.

**Pretrained Model & Tokenization**

- Pretrained Model: The “siebert/sentiment-roberta-large-english” model from Hugging Face’s transformers library was used for binary sentiment classification.
- Tokenizer: The RoBERTa tokenizer was used to convert review texts into tokenized inputs.
- Maximum Token Length: Text was tokenized with truncation and padding up to a maximum length of 512 tokens.

**Observation:**

Model Accuracy on Test Set: 95.15.

Classification report:

| | 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
```

Model Accuracy on Test Set: 95.15.

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. The catch is  that the model needs GPU hardware to train and inference. We used google colab pro for this model .