Sentiment Analysis on Amazon Fine Food Reviews

Alagesan, Aishwariya, alagesan.a@northeastern.edu
Gajula, Saiteja Reddy, gajula.sai@northeastern.edu
Hansda, Sheela, hansda.s@northeastern.edu

Abstract

This study explores sentiment analysis on Amazon Fine Food Reviews to classify customer sentiments as positive, neutral, or negative. Leveraging advanced natural language processing (NLP) techniques, the project aims to uncover trends, improve sentiment prediction accuracy, and provide insights into customer preferences. By utilizing topic modeling, sentiment classification, and metadata analysis, the findings contribute to understanding the dynamics of online product reviews.

1. Introduction

The rapid growth of e-commerce has necessitated better tools to analyze customer feedback for informed decision-making. Sentiment analysis serves as a cornerstone in this regard, offering insights into customer satisfaction and product quality. This project uses the Amazon Fine Food Reviews dataset to apply machine learning models for sentiment classification, supported by advanced NLP techniques to improve prediction accuracy and interpretability.

2. Background

Sentiment analysis has evolved with advancements in NLP and machine learning, transitioning from simple lexicon-based methods to sophisticated deep learning approaches. Previous studies highlight the effectiveness of techniques like LSTMs and transformers in capturing the nuances of sentiment in textual data. This project builds upon such work by integrating metadata like user preferences and product categories to refine sentiment prediction.

3. Data Preparation and Preprocessing

3.1 Data Acquisition

The dataset was sourced from the Stanford SNAP repository, containing Amazon food reviews. The first step involved downloading the dataset as a compressed TXT file. The TXT file was then converted into a CSV file to structure the data into columns like productId, userId, profileName, helpfulness, score, time, summary, and text. This conversion made it easier to analyze and manipulate the data using data analysis libraries such as pandas. The dataset used in this study contains over 500,000 reviews from the Amazon Fine Food Reviews dataset. Key fields include:

- **reviewText**: Customer reviews
- **score**: Ratings (1–5)
- **summary**: Review summary
- **metadata**: Product ID, user ID, and timestamps

3.2 Exploratory Data Analysis

In this section, we delve into the Amazon food reviews dataset to uncover key patterns, trends, and relationships that can aid in understanding consumer sentiment. The analysis aims to provide a comprehensive overview of the data and generate actionable insights to support sentiment classification. Below are the primary questions addressed during the exploratory analysis:

1. *What is the distribution of the target variable (sentiment/score)?*

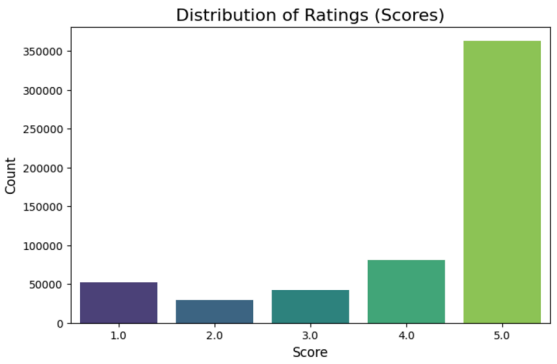

Fig 1.0

The dataset exhibits an imbalance, with a noticeable bias toward positive scores (5). To address this, we applied oversampling to the minority classes during preprocessing to achieve a balanced dataset.

2. *What is the distribution of reviews deemed "useful/useless"?*

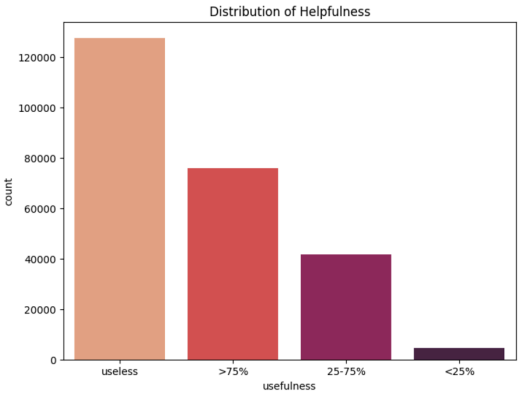

Fig 2.0

Not all ratings appear to be equally helpful to users. While people provide reviews, these are not always perceived as useful by others.

3. *What is the relationship between review usefulness and sentiment categories?*

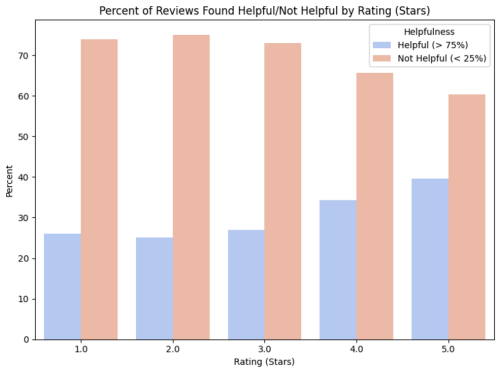

Fig 3.0

The graph shows that the majority of reviews, regardless of rating, are deemed "not helpful" (<25% helpfulness), though higher-rated reviews have a slightly greater proportion of "helpful" reviews . Low-rated reviews are the least likely to be considered helpful.

4. *Which words are most frequently used in positive reviews?*

Fig 4.0

The word cloud for positive sentiment prominently features words such as "good," "great," "love," and similar expressions.

5. *Which words are most frequently used in negative reviews?*

Fig 5.0

The word cloud for negative sentiment, similar to the positive one, prominently features words such as "bad," "disappointed," "waste," and other related expressions.

6. **Does the length of reviews (word count) influence their perceived helpfulness?**

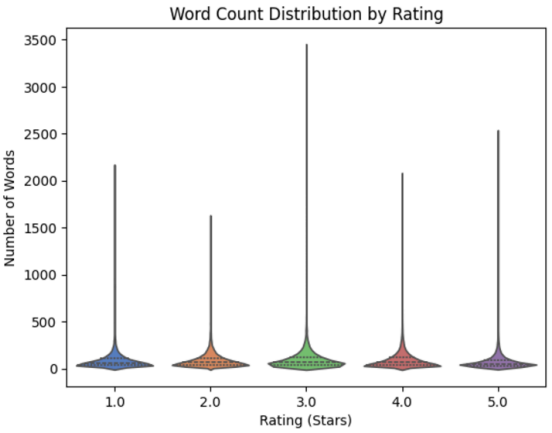

Fig 6.0

The number of words in a review does not appear to be significantly influenced by the star rating (1 to 5 stars), suggesting that review length is relatively consistent across different ratings.

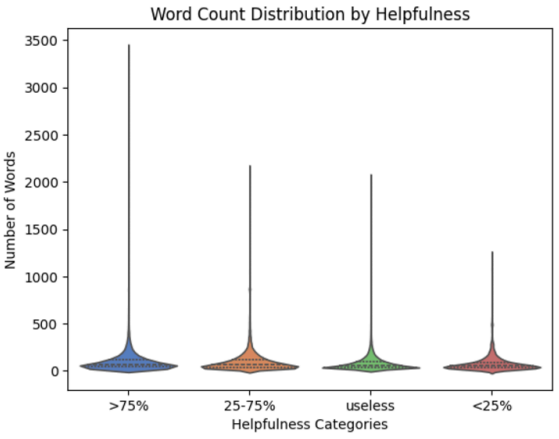

Fig 6.1

There appears to be minimal correlation between word count and rating distribution. However, a slight distinction is observed between reviews with >75% usefulness and those with <25% usefulness. Reviews in the >75% usefulness category tend to have longer word counts, indicating a higher likelihood of detailed content compared to reviews in the <25% usefulness category.

7. **Is there a temporal trend in product reviews?**

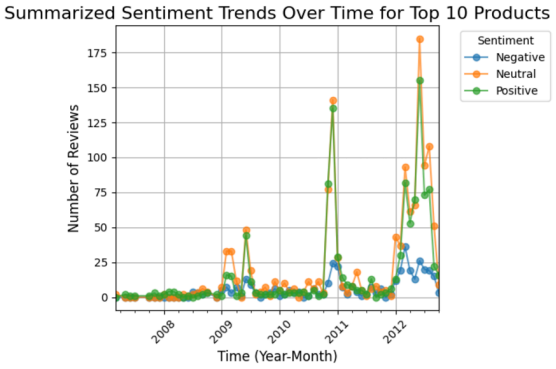

Fig 7.0

The number of reviews has steadily increased from 2008 to 2012, reflecting a growing trend of people sharing their opinions. Positive reviews consistently dominate, while neutral and negative reviews remain lower and relatively stable over time. Reviews prior to 2008 are negligible in volume.

8. **Are there identifiable patterns among frequent reviewers?**

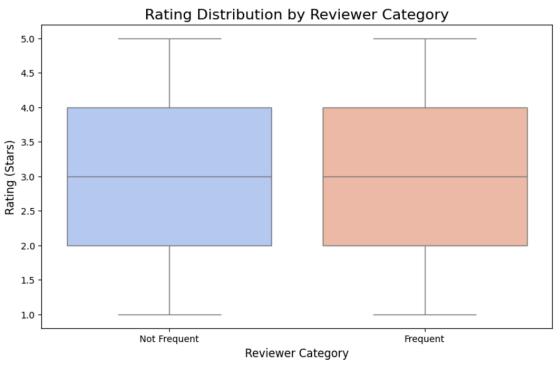

Fig 8.0

The similarity between the two distributions suggests that frequent and infrequent reviewers exhibit comparable rating behavior.

9. **Do frequent reviewers exhibit specific trends in review length?**

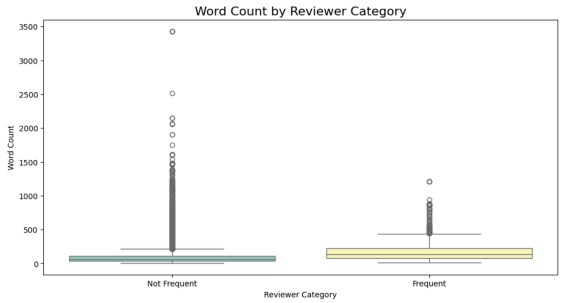

Fig 9.0

The word count distributions suggest that both frequent and infrequent reviewers tend to exhibit similar rating patterns.

3.3 Preprocessing Techniques

1. **Data Cleaning:**
   - The raw dataset contained inconsistencies and missing values that needed to be addressed. The time column, originally in Unix timestamp format, was converted into a readable date format to facilitate temporal analysis. Missing values in the profileName and summary columns were handled by replacing null entries with 'Unknown' and the most frequent value, respectively. Duplicates were removed to ensure data integrity. This step ensured that the dataset was clean, consistent, and ready for further analysis.
2. **Text Normalization:**
   - Converted text to lowercase for uniformity.
   - Removed stop words to retain meaningful content.
   - Applied stemming to reduce words to their base forms.
3. **Class Balancing with SMOTE:**
   - **Undersampling** was performed to reduce the number of reviews from the majority class (positive sentiment) to match the number of reviews in the minority classes (negative and neutral sentiments). The goal was to create a balanced dataset where each sentiment class (positive, neutral, and negative) had an equal number of reviews.
   - The undersampling process focused on randomly selecting a smaller subset of reviews from the positive sentiment class to match the number of reviews in the negative and neutral sentiment classes.
   - This helped eliminate the bias that would have otherwise occurred if the model was trained on a disproportionate number of positive reviews.

- For example, if the dataset had 362,936 positive reviews, 80,000 neutral reviews, and 50,000 negative reviews, undersampling was applied to the positive class to reduce the number of positive reviews to 50,000. This ensured that each class had 50,000 reviews, making the dataset more balanced and ensuring that the model learned equally from all sentiment categories.

4. **Parallel Processing:**
   - Utilized Dask to process large datasets in parallel by splitting the data into partitions.
5. **Feature Engineering:**
   - Created sentiment labels based on score:
     - 1-2: Negative
     - 3: Neutral
     - 4-5: Positive
   - Extracted features like word count and unique tokens.
   - Sentiment labels were assigned based on the score values. Reviews with scores of 4 or 5 were classified as Positive, those with a score of 3 as Neutral, and reviews with scores of 1 or 2 as Negative. This classification helped in segmenting the data into distinct sentiment categories, providing a clear understanding of the distribution of sentiments across the dataset.
6. **Text Preprocessing:** The text of each review underwent extensive preprocessing to prepare it for natural language processing tasks. HTML tags were removed using BeautifulSoup, and non-alphabetic characters were filtered out. The text was converted to lowercase, and stopwords (common words like "the" and "and") were removed to reduce noise. Finally, stemming was applied using the Snowball Stemmer to reduce words to their root forms, ensuring that variations of the same word were treated as identical. This preprocessing step was crucial for reducing the dimensionality of the text data and improving model performance.
7. **Word Count Analysis:** A new word_count column was created by counting the number of words in each review. This allowed for the analysis of how review length correlated with sentiment and helpfulness. Visualizations showed that the distribution of word counts did not vary significantly across different score categories, although reviews deemed highly helpful tended to be longer. This insight provided context on the role of review length in perceived helpfulness and sentiment.

3.3 Feature Extraction

After the initial data preprocessing steps, the next focus was on transforming the text data into a numerical format suitable for machine learning models. This transformation was achieved using techniques like Lemmatization, CountVectorizer, and TF-IDF (Term Frequency-Inverse Document Frequency) Vectorization, which are critical in converting raw text into structured data that the machine learning models can process. Below is a detailed explanation of each of these steps.

1. **Lemmatization:** Lemmatization is a text normalization process that reduces words to their base or root form, ensuring consistency in the representation of words. Unlike stemming, which simply trims word endings, lemmatization considers the context of the word to derive its correct base form. For example, "running" would be reduced to "run," and "better" would be transformed into "good." This process was applied to the reviews to ensure that variations of a word, such as "running," "ran," and "runs," were treated as the same word. This not only helped in reducing the dimensionality of the data but also ensured that the features represented a more meaningful and consistent form of the text. By applying lemmatization, the model was able to focus on the core meaning of words, thus improving its overall performance in sentiment classification.

2. **CountVectorizer:** CountVectorizer is a fundamental method for converting a collection of text documents into a matrix of token counts, a process often referred to as the "bag of words" model. This method treats each unique word in the entire dataset as a feature and counts its occurrences in each review. The result is a sparse matrix where each row represents a review, and each column represents a unique word from the dataset, with the value in each cell indicating the number of times that word appears in the corresponding review. While this technique provides a simple representation of the text data, it does not capture the importance of the words across different documents. It simply counts the frequency of words, making it useful for capturing basic word usage patterns but without accounting for their relative importance in the dataset.

3. **TF-IDF (Term Frequency-Inverse Document Frequency) Vectorization:** TF-IDF is a more sophisticated text vectorization method that improves upon CountVectorizer by assigning weights to words based on their frequency within a review and their uniqueness across the entire dataset. The Term

Frequency (TF) component measures how often a word appears in a specific review, while the Inverse Document Frequency (IDF) component adjusts this frequency based on how commonly the word appears across all reviews in the dataset. Words that appear frequently in many reviews, such as "product" or "review," are down-weighted, while terms that are more specific to a review (e.g., "delicious," "stale") are given higher importance. This allows the model to focus on more informative words while reducing the impact of common terms. The result is a weighted sparse matrix where each word's importance is better represented, making TF-IDF a powerful tool for text classification tasks like sentiment analysis.

4. **Combining Lemmatization with Vectorization:** In the preprocessing pipeline, lemmatization was applied to the text before using either CountVectorizer or TF-IDF Vectorizer. This ensured that words in different forms, such as "running" and "ran," were reduced to the same base word, allowing the models to work with more consistent features. Additional preprocessing steps, such as removing stopwords (common words like "the," "and," etc.), converting all words to lowercase, and removing punctuation and special characters, were also applied. These steps helped to further clean the text by eliminating unnecessary words and ensuring that the model focused only on the meaningful terms in the review. By using lemmatization and vectorization techniques together, the data was transformed into a structured and informative format, ready for machine learning model training.

These text preprocessing and feature extraction steps, including lemmatization, CountVectorizer, and TF-IDF, were crucial in converting raw text into a structured form that could be effectively used for machine learning tasks. By normalizing the text, reducing dimensionality, and emphasizing the most informative terms, these techniques ensured that the models could accurately classify sentiment based on the content of the reviews, providing valuable insights from Amazon food reviews.

3.4 Data Splitting

To evaluate the performance of the machine learning models, the dataset was divided into two distinct subsets: training and testing. The split ensures that the model is trained on a portion of the data and evaluated on a separate, unseen portion, which is crucial for understanding how well the model can generalize to new, real-world data.

- **Training Set:** Typically, 80% of the data was allocated for training. The model uses this set to learn patterns, relationships, and features from the data. During training, the model adjusts its parameters to minimize error and improve prediction accuracy.

- **Testing Set:** The remaining 20% of the data was reserved for testing. This subset is used to evaluate how well the model performs on data it has not encountered during training. The testing set provides an unbiased estimate of the model's performance, helping to assess its ability to generalize to new data.

4. Model Architecture and Training

4.1 Model Architecture

- **Baseline Models:** The sentiment analysis pipeline involves the use of two different models: **Recurrent Neural Networks (RNN)** and **Long Short-Term Memory (LSTM)**. Both models are well-suited for handling sequential data, such as text, and are capable of learning dependencies over time, which is crucial for sentiment analysis tasks.
- **Embedding Layer**: The input text is initially transformed into dense vector representations using an embedding layer. Word2Vec embeddings are utilized to map words into vectors that capture semantic relationships, allowing the model to understand word meanings in context.
- **Recurrent Neural Network (RNN)**: The RNN processes the sequence of words one at a time, maintaining a hidden state that is updated as it reads each word. While RNNs are effective for short-term dependencies, they can struggle with long-term dependencies due to the vanishing gradient problem, which limits their performance on tasks like sentiment analysis.
- **Long Short-Term Memory (LSTM)**: LSTM, an advanced version of RNN, overcomes the vanishing gradient problem and is better at capturing long-term dependencies in sequences. It uses special units (gates) to decide what information to remember and what to forget, which is crucial for sentiment analysis where the context of words can span long sequences.
- **Dense Output Layer**:
  Both models use a dense output layer with a **softmax** activation function to classify the sentiment of the input text into three categories: positive, neutral, and negative. The softmax function is appropriate here as it outputs a probability distribution over multiple classes.

4.2 Training Process

The models were trained with the following settings:

- **Optimizer**:
  The **Adam optimizer** was used for both RNN and LSTM models. Adam is well-suited for training deep learning models as it adapts the learning rate during training, making it more efficient.
- **Loss Function**:
  Categorical Cross-Entropy was chosen as the loss function, as it is suitable for multi-class classification tasks like sentiment analysis. This loss function measures how well the predicted probabilities align with the true sentiment labels.
- **Regularization**:
  Dropout was applied in both models to prevent overfitting. Dropout randomly disables a fraction of neurons during training, forcing the network to rely on multiple pathways and improving generalization. Additionally, **early stopping** was used to halt training when the validation performance stopped improving, thus preventing the model from overfitting to the training data.
- **Metrics**:
  The models were evaluated using **Accuracy**, **Precision**, **Recall**, and **F1-score**, to ensure that the model performed well across all sentiment classes, especially in terms of correctly identifying both positive and negative sentiments.

5. Results

1.Naive Bayes:

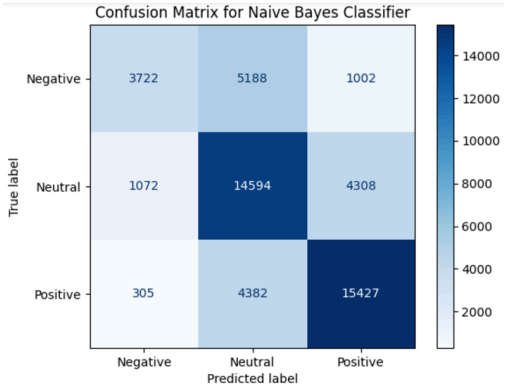

Fig 10.0

Correct Predictions:

- Negative: 3,722 correctly classified.
- Neutral: 14,594 correctly classified.
- Positive: 15,427 correctly classified.

Misclassifications:

- Negative misclassified as Neutral: 5,188.
- Negative misclassified as Positive: 1,002.
- Neutral misclassified as Negative: 1,072.
- Neutral misclassified as Positive: 4,308.
- Positive misclassified as Neutral: 4,382.
- Positive misclassified as Negative: 305.

Overall Insights:

- The model performs best for Neutral and Positive sentiments.
- Struggles to distinguish Negative from Neutral, with significant misclassifications.

Performance Metrics:

- **Negative Sentiment:**
  - *Precision: 73%*
  - *Recall: 38%*
  - *F1-Score: 50%*
- **Neutral Sentiment:**
  - *Precision: 60%*
  - *Recall: 73%*
  - *F1-Score: 66%*
- **Positive Sentiment:**
  - *Precision: 74%*
  - *Recall: 77%*
  - *F1-Score: 76%*

2. Logistic Regression :

Negative Sentiment:

- *Precision: 71%,*
- *Recall: 60%*
- *F1-Score: 65%*

Neutral Sentiment:

- *Precision: 67%.*
- *Recall: 71%*
- *F1-Score: 69%*

Positive Sentiment:

- *Precision: 78%*
- *Recall: 80%*
- *F1-Score: 79%*

Correct Predictions:

- Negative: 5,939 correctly classified.
- Neutral: 14,137 correctly classified.
- Positive: 16,048 correctly classified.

Misclassifications:

- Negative misclassified as Neutral: 3,238.
- Negative misclassified as Positive: 735.

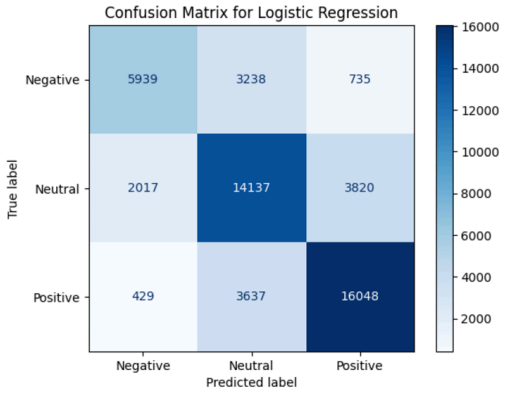

Fig 11.0

- Neutral misclassified as Negative: 2,017.
- Neutral misclassified as Positive: 3,820.
- Positive misclassified as Neutral: 3,637.
- Positive misclassified as Negative: 429.

Overall Insights:

- The model performs best for Neutral and Positive sentiments.
- Struggles to distinguish Negative from Neutral, with significant misclassifications.

Both the **RNN** and **LSTM** models were trained and evaluated, and their performance was compared based on test accuracy, precision, recall, and F1-score.

3.RNN Model:

- **Accuracy**: 82.26%
- **Precision**:
  - *Positive: 86%, Neutral: 80%, Negative: 80%*
- **Recall**:
  - *Positive: 83%, Neutral: 83%, Negative: 80%*
- **F1-Score**:
  - *Positive: 84%, Neutral: 81%, Negative: 80%*

4. LSTM Model:

- **Accuracy**: 78.25%
- **Precision**:
  - *Positive: 82%, Neutral: 76%, Negative: 74%*
- **Recall**:
  - *Positive: 78%, Neutral: 75%, Negative: 72%*
- **F1-Score**:
  - *Positive: 80%, Neutral: 75%, Negative: 73%*

The **RNN model** performed better in terms of overall accuracy and was more consistent across all sentiment classes. However, the **LSTM model** showed slightly improved performance in handling longer-term dependencies, which could be crucial for more complex textual data. While the **RNN** was faster to train, the **LSTM** model, due to its specialized architecture, could have potential advantages for more intricate sentiment analysis tasks.

Evaluation Metric Comparison among models

	Naive Bayes	Logistic Regression	LSTM	RNN
Accuracy	0.67	0.79136	0.7825	0.8226
Negative Precision	0.73	0.71	0.74	0.8
Negative Recall	0.38	0.6	0.72	0.8
Negative F1-Score	0.5	0.65	0.73	0.8
Neutral Precision	0.6	0.67	0.76	0.8
Neutral Recall	0.73	0.71	0.75	0.83
Neutral F1-Score	0.66	0.69	0.75	0.81
Positive Precision	0.74	0.78	0.82	0.86
Positive Recall	0.77	0.8	0.78	0.83
Positive F1-Score	0.76	0.79	0.8	0.84

Fig 12.0

References

1. **Original Dataset**: Amazon Fine Food Reviews Dataset
2. **Topic Modeling Techniques**: Latent Dirichlet Allocation (LDA) in NLP
3. **Sentiment Classification**: Applications in Python using scikit-learn and TensorFlow
4. **Scikit-learn Documentation** Pedregosa, F., Varoquaux, G., Gramfort, A., et al. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12, 2825-2830. URL: https://scikit-learn.org/
5. **TF-IDF and Count Vectorization** Salton, G., & Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. *Information Processing & Management*, 24(5), 513-523. DOI: 10.1016/0306-4573(88)90021-0
6. **Lemmatization and Stemming** Lancaster, F. W. (2003). *Information Retrieval: A Survey*. Wiley. URL: https://www.wiley.com/
7. **Undersampling and Class Imbalance** Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic Minority Over-sampling Technique. *Journal of Artificial Intelligence Research*, 16, 321-357. DOI: 10.1613/jair.953
8. **Text Preprocessing for Sentiment Analysis** Pang, B., & Lee, L. (2008). *Opinion Mining and Sentiment Analysis*. Foundations and Trends in Information Retrieval, 2(1–2), 1-135. DOI: 10.1561/1500000011
9. **Random Sampling and Stratified Sampling** Kohavi, R. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. In *International Joint Conference on Artificial Intelligence (IJCAI)* (pp. 1137-1143). URL: https://www.ijcai.org/
10. **Evaluation Metrics for Classification** Sokolova, M., & Lapalme, G. (2009). A systematic analysis of performance measures for classification tasks. *Information Processing & Management*, 45(4), 427-437. DOI: 10.1016/j.ipm.2009.03.002
11. **Sentiment Analysis:** 1.1. Zhang, L., Wang, S., & Liu, B. (2018). Deep learning for sentiment analysis: A survey. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery. 1.2. Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). BERT: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805. 1.3. Munikar, M., Shakya, S., & Shrestha, A. (2019). Fine-grained sentiment classification using BERT. 2019 Artificial Intelligence for Transforming Business and Society (AITB).
12. **Aspect-Based Sentiment Analysis:** 2.1. Ding, X., Liu, B., & Yu, P. S. (2008). A holistic lexicon-based approach to opinion mining. Proceedings of the 2008 International Conference on Web Search and Data Mining. 2.2. Xu, H., Liu, B., Shu, L., & Yu, P. S. (2019). BERT post-training for review reading comprehension and aspect-based sentiment analysis. arXiv preprint arXiv:1904.02232.
13. **Review Helpfulness Prediction:** 3.1. McAuley, J., & Leskovec, J. (2013). From amateurs to connoisseurs: modeling the evolution of user expertise through online reviews. Proceedings of the 22nd International Conference on World Wide Web.
14. **Cross-Domain Sentiment Analysis:** 4.1. Abdi, A., Shamsuddin, S. M., Hasan, S., & Piran, J. (2019). Deep learning-based sentiment analysis for Roman Urdu text. 2019 International Conference on Electrical, Electronics and Computer Engineering (UPCON).
15. **Multi-modal Analysis:** 5.1. Li, Y., Pan, Q., Yang, T., Wang, S., Tang, J., & Cambria, E. (2021). Learning binary codes for hypergraph-based multi-modal multi-view feature learning. IEEE Transactions on Multimedia. 5.2. Zeng, Z., Gu, W., Chen, Q., & Yu, Z. (2021). Multimodal sentiment analysis via reinforcement learning. Information Fusion.
16. **General Survey and State-of-the-Art:** 6.1. Yadav, A., & Vishwakarma, D. K. (2020). Sentiment analysis using deep learning architectures: a review. Artificial Intelligence Review.