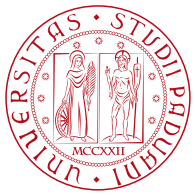

UNIVERSITÀ DEGLI STUDI DI PADOVA

DIPARTIMENTO MATEMATICA
Dipartimento di Matematica "Tullio Levi-Civita"

Natural Language Processing Project

Sentiment Analysis on Amazon Fine Food Dataset

Author: Eldar Eyvazov

20/01/2025

1 Abstract

In this project, we conducted sentiment analysis on the Amazon Fine Food Reviews dataset, aiming to explore and compare various machine learning and deep learning approaches. Our methodology began with classical machine learning algorithms, including Logistic Regression, to establish baseline performance metrics. We then advanced to more sophisticated deep learning models such as Convolutional Neural Networks (CNN), Long Short-Term Memory networks (LSTM), Bidirectional LSTMs, and a hybrid approach that combines LSTM with CNN.

To tackle the problem of class imbalance, we employed oversampling techniques and tested some of our models on a balanced dataset. The dataset presented a significant challenge due to its highly skewed distribution toward positive sentiment. Thus, our primary objective was to build a robust model capable of accurately classifying sentiments across the spectrum while maintaining computational efficiency.

For the word embeddings, we used pre-trained GloVe embeddings to represent the textual data in a dense, continuous vector space. This approach allowed the model to better capture semantic relationships between words, improving performance across various models.

Sentiment analysis, a key task in Natural Language Processing (NLP), involves determining the emotional tone behind a series of words. It helps systems to understand and classify text data into categories such as positive, negative, or neutral. In this study, we focused on accurately classifying sentiments within Amazon Fine Food reviews, contributing to the development of scalable and efficient sentiment analysis systems for large-scale, imbalanced datasets.

2 Methodology

Our sentiment analysis study involved implementing and evaluating a range of models to identify the most effective one, not only in terms of performance but also with respect to computational cost and real-world feasibility. A key focus was on finding a model capable of handling the significant class imbalance in our dataset, which is heavily skewed toward the positive class. Rather than relying on sampling techniques such as oversampling or undersampling, we prioritized models inherently robust to such limitations.

We implemented the following models, progressing from classical machine learning algorithms to more advanced deep learning architectures:

1. **Logistic Regression**
2. **Convolutional Neural Networks (CNN)**
3. **Long Short-Term Memory Networks (LSTM)**
4. **Bidirectional LSTM (BiLSTM)**
5. **Hybrid Model (LSTM + CNN)**

Extensive hyperparameter tuning was carried out on the validation set for each method. This included optimizing parameters such as learning rates, layer sizes, and dropout rates to enhance model performance. Additionally, we investigated the impact of using pre-trained word embeddings—*GloVe* and *Word2Vec*—to assess their influence on classification accuracy and computational efficiency. This comprehensive approach allowed us to evaluate both the predictive performance and the practical applicability of the models.

2.1 Dataset Collection

For our project, we are using the Amazon Fine Food Reviews dataset, introduced by Julian McAuley and Jure Leskovec in their study [6]. This dataset contains 568,427 reviews of fine food products sold on Amazon, and includes valuable information such as review text, ratings, helpfulness metrics, and timestamps. The reviews are divided into three sentiment classes: **Positive** (443,777), **Negative** (82,012), and **Neutral** (42,638). As shown in Figure 1, the distribution of sentiment classes clearly illustrates the significant imbalance, with the majority of reviews classified as Positive, while Negative and Neutral reviews are far less frequent.

2.2 Data Processing and Preparation

For our study, we performed a series of steps to process and prepare the data for sentiment analysis. The key steps are as follows:

- The sentiment labels were mapped from the review scores as follows:

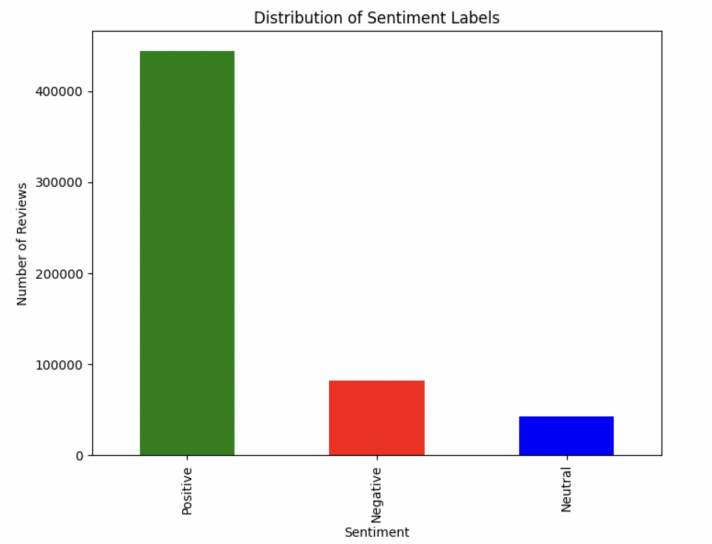

FIGURE 1: Distribution of sentiment classes in the Amazon Fine Food Reviews dataset, showing a significant imbalance between Positive, Negative, and Neutral reviews.

  - Scores ≥ 4 were mapped to **Positive**.
  - Scores ≤ 2 were mapped to **Negative**.
  - Scores between 3 and 4 were mapped to **Neutral**.
- The dataset contains 53 samples with missing values, which were replaced with an empty string. The missing values were distributed across two columns:
  - `ProfileName`: 26 missing values.
  - `Summary`: 27 missing values.
- For text preprocessing, we used the `nltk` and `string` libraries:
  - Convert text to lowercase, remove punctuation (`string.punctuation`), and tokenize using `nltk.word_tokenize`.
  - Remove stopwords using `nltk.corpus.stopwords.words('english')` and lemmatize words with `nltk.WordNetLemmatizer`.
  - Reconstruct the cleaned text by joining the lemmatized tokens.
- Use the `Keras Tokenizer` to convert the preprocessed text into integer sequences, limiting the vocabulary to the 10,000 most frequent words.
- Pad the integer sequences to a maximum length of 100 tokens.
- Split the data into training (64%), validation (16%), and test (20%) sets, ensuring stratification to maintain the sentiment class distribution.
- To address class imbalance, we created a resampled version of the dataset using the `imbalanced-learn` library [4]. The resampling process involved increasing the number of samples in the minority sentiment classes (**Negative** and **Neutral**) to match the number of samples in the majority sentiment class (**Positive**).
- After resampling, each sentiment class (**Positive**, **Negative**, and **Neutral**) contained 443,777 samples.
- The same preprocessing steps (e.g., tokenization, padding, and splitting) were applied to the resampled dataset to ensure consistency across experiments.
- The class distribution after resampling is visualized in Figure 2.

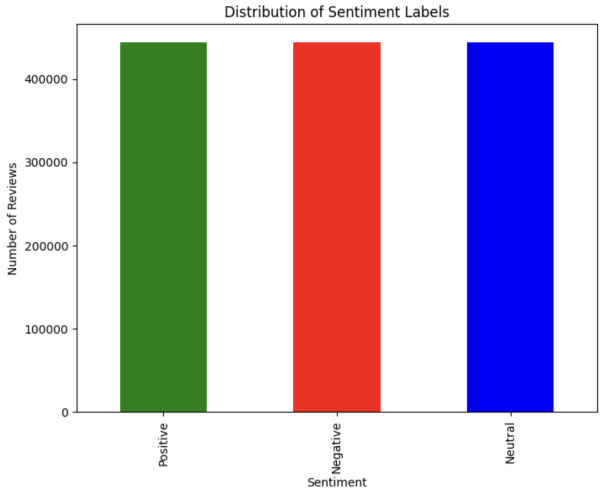

FIGURE 2: Class distribution of the dataset after oversampling. Each sentiment class (**Positive**, **Negative**, and **Neutral**) contains an equal number of samples (443,777).

2.3 Implementation Details

The implementation of our models involved several key choices. The *Adam* optimizer was used consistently across all deep learning models. To prevent overfitting, two essential callbacks, *EarlyStopping* and *ReduceLROnPlateau*, were employed. The *EarlyStopping* callback monitored validation loss, halting training when no improvement was observed for a specified number of epochs. Meanwhile, *ReduceLROnPlateau* adjusted the learning rate by a factor of 0.1 when validation loss plateaued, with a minimum learning rate set to 1×10^{-7}.

Various batch sizes, ranging from 64 to 128, and different learning rates were tested to optimize performance. The loss function, *SparseCategoricalCrossentropy*, effectively handled class labels. To address overfitting tendencies in certain models, *Dropout* layers with varying values were incorporated as needed.

For text representation, we relied on pre-trained *GloVe 100-dimensional 6B* word vectors [7], trained on a massive corpus of six billion tokens, including Wikipedia and Gigaword. These embeddings were selected for their balance between representation power and computational efficiency. Other versions of GloVe embeddings, such as 50, 200, and 300 dimensions, were considered, but the 100-dimensional vectors provided an optimal trade-off. Higher-dimensional embeddings, while capturing more semantic nuances, increase computational cost, whereas lower-dimensional embeddings are computationally efficient but may lack depth in representing semantic relationships. Using pre-trained GloVe embeddings instead of training embeddings from scratch further streamlined training and reduced computational complexity.

2.4 Metrics

2.5 Metrics

To evaluate the performance of our models, we used four key metrics: *Accuracy*, *Precision*, *Recall*, and *F1-Score*. These metrics collectively provide insights into the models' ability to handle imbalanced datasets. For the sake of brevity, we omit the mathematical formulations here.

- **Accuracy**: Measures the ratio of correctly predicted instances to the total number of instances. While commonly used, accuracy may not fully reflect the model's performance on imbalanced datasets.
- **Precision**: Represents the proportion of correctly predicted positive instances out of all instances predicted as positive.
- **Recall (Sensitivity)**: Evaluates the proportion of actual positive instances that were correctly predicted.
- **F1-Score**: The harmonic mean of precision and recall, providing a balanced measure of the model's performance, particularly useful in cases of class imbalance.

By using these metrics, we aim to capture both the overall accuracy and the model's ability to effectively identify minority class instances.

3 Experiments

3.1 Logistic Regression

For our sentiment analysis project, we started with Logistic Regression, a well-known machine learning algorithm commonly used for classification tasks. As a classical approach, it serves as a solid foundation for our study. For feature extraction, we used **TF-IDF** (Term Frequency-Inverse Document Frequency) with bi-grams, enabling the model to capture the relationships between pairs of consecutive words in the text. For example, bi-grams like "great product" or "fast delivery" help the model understand the context beyond individual words, providing a richer representation of the text and improving sentiment analysis.

The Logistic Regression model achieved an overall accuracy of **91%**, indicating strong performance on the dominant class (positive sentiment). However, it struggled with the minority classes, particularly the neutral class. Many neutral samples were misclassified as positive, revealing the model's bias toward the dominant class, a common problem in unbalanced data sets. Furthermore, misclassification occurred between the positive/negative and neutral classes, highlighting the model's difficulty in accurately classifying the neutral class.

The **confusion matrix** shown in Figure 4 illustrates these misclassifications. As observed, the neutral class is frequently misclassified as positive, highlighting the model's bias and the challenges faced when handling imbalanced datasets.

To address this issue, we experimented with training the model using class weights to counterbalance the imbalance of the dataset. However, this approach did not result in any significant improvement in overall performance. Instead, we observed a trade-off across different metrics, with minor gains in some areas offset by declines in others. Ultimately, the adjustments did not yield a meaningful or consistent improvement in the model's ability to handle the minority classes.

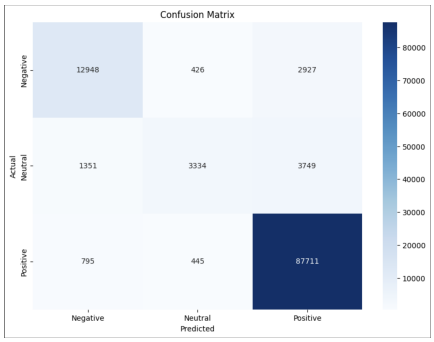

FIGURE 3: Confusion matrix of Logistic Regression

3.2 Convolutional Neural Networks (CNN)

Convolutional Neural Networks (CNNs) are designed to extract spatial features from text sequences, making them highly effective for sentiment analysis tasks. By utilizing convolutional layers, CNNs capture patterns and local features within the text, enabling the model to identify key elements for sentiment classification.

For our project, we implemented and trained several CNN models, inspired by the methodologies presented in *"Convolutional Neural Networks for Sentiment Analysis on Weibo Data: A Natural Language Processing Approach"* by Yufei Xie and Rodolfo C. Raga Jr. [8], and *"Sentiment Classification Using Convolutional Neural Networks"* by Hannah Kim and Young-Seob Jeong [3]. These papers provided valuable insights into CNN architecture design for sentiment classification.

We adapted these models by reducing their complexity to better suit our dataset, balancing performance and computational efficiency. Our model architecture is as follows:

- Three 1D convolutional layers with filters of 128, 64, and 32, capturing local text features.
- Batch normalization after each convolutional layer to stabilize learning.
- Dropout (rate = 0.4) applied after convolutional blocks to prevent overfitting.
- A global average pooling layer to reduce the feature map's dimensionality.
- A dense layer with ReLU activation to further refine features.
- The final output layer uses softmax activation for multi-class sentiment classification.

The model consists of a total of 1,053,443 parameters, with only 52,995 being trainable. This relatively lightweight architecture ensures efficient memory usage and faster training times, making it well-suited for environments with limited resources.

We observed that the CNN model excels at classifying the positive class, not only achieving high accuracy but also delivering impressive metrics across all evaluations for the positive class. Additionally, the CNN model is likely more efficient in terms of training cost. However, when evaluating

overall performance, the CNN model shows results comparable to the Logistic Regression model we initially trained. Despite experimenting with different CNN architectures, we have not reached a performance level that justifies the use of deeper models. As a result, we decided to employ the oversampling technique discussed earlier in this report, along with slight adjustments to the architecture to better align with the model's complexity. The changes made were as follows:

– Three 1D convolutional layers with filters of 128, 64, and 32, respectively, to capture local text features. The number of filters in each convolutional layer has been doubled.

- The number of units in the dense layer was increased from 32 to 64.

And we observed a substantial, or you could say significant, performance improvement.

Metrics (Macro)	CNN	CNN (Resampled)
Accuracy	90.64%	97.64%
Precision	79.51%	97.68%
Recall	72.33%	97.63%
F1 Score	74.79%	97.62%
Loss	0.2719	0.1032

TABLE 1: Comparison of CNN models on original and resampled dataset

3.3 Long Short-Term Memory Networks (LSTM)

Long Short-Term Memory Networks (LSTMs) were introduced to capture sequential dependencies and contextual information within text data. Their capability to remember long-term dependencies makes them particularly effective for processing and analyzing text sequences in sentiment analysis. After exploring existing approaches in sentiment analysis, particularly Convolutional Neural Networks (CNNs), we further analyzed the literature and implemented our model inspired by [5]. Our architecture is composed of:

- An embedding layer that uses pre-trained GloVe embeddings (non-trainable) to convert input text into dense vector representations. Which is same accross all models

- Two LSTM layers:

  - The first LSTM layer with 128 units, which outputs sequences to be passed to the next LSTM layer.

  - The second LSTM layer with 64 units, which outputs the final sequence representation for the model.

- A Global Average Pooling layer to aggregate the LSTM outputs into a fixed-size vector.

- One or more dense layers with ReLU activation and dropout (rate = 0.5) applied to prevent overfitting.

- The final output layer with a softmax activation function, which outputs probabilities for multi-class sentiment classification.

Similar to Convolutional Neural Networks, the LSTM model also performs well on the positive class but faces challenges with the minority classes, particularly the Neutral class. We observed significant classification errors with the Neutral class. We attempted to test these models on the balanced dataset to evaluate whether they could provide improvements over the CNN model. While we anticipated more stable performance with the balanced dataset compared to the original, we encountered a challenge due to the higher computational cost of LSTM models, including BiLSTM, which have more trainable parameters. This resource limitation prevented us from adapting the architecture to the new balanced dataset. As a result, it remains a future work of this research to explore and identify a suitable LSTM-based architecture that can be effectively applied to our dataset while keeping the computational cost manageable. The model has a total of 168,403 trainable parameters.

3.4 Bidirectional LSTM (BiLSTM)

Next, we explored the possibility and feasibility of improving performance by replacing the LSTM layers in the previous architecture with BiLSTM layers. The *Bidirectional LSTM (BiLSTM)* enhances the traditional LSTM model by processing text sequences in both forward and backward directions. This bidirectional approach allows the model to capture both past and future context, which can significantly improve its ability to understand and classify sentiment in long sequences. Previous studies have highlighted the advantages of BiLSTM over traditional LSTM models, particularly in text processing, due to its ability to leverage information from both directions of the sequence, resulting in more accurate representations [2].

- An embedding layer that uses pre-trained GloVe embeddings (non-trainable) to convert input text into dense vector representations. This remains the same across all models.

- Two BiLSTM layers:

  - The first BiLSTM layer with 128 units, which outputs sequences to be passed to the next BiLSTM layer.

  - The second BiLSTM layer with 64 units, which outputs the final sequence representation for the model.

- The remaining part of the network, including the Global Average Pooling layer, dense layers with ReLU activation, dropout (rate = 0.5) to prevent overfitting, and the final output layer with softmax activation for multi-class sentiment classification, remains unchanged.

However, when testing our model on the original dataset and comparing it to the previous LSTM architecture, we did not observe any significant improvement. Instead, we encountered an overfitting tendency with the BiLSTM-based architecture. To maintain a consistent comparison, we decided to keep the architecture unchanged, without adding or removing any layers. In this comparison between the LSTM

and BiLSTM models, as shown in Table 2, the performance metrics such as precision, recall, F1 score, and accuracy are nearly identical for both models. However, considering the computational complexity, the LSTM model can be favored due to its relatively simpler architecture, which can reduce training time and resource consumption without significantly sacrificing performance. This makes the LSTM model a more computationally efficient choice, especially for large-scale datasets.

Metric	LSTM	BiLSTM
Loss	0.2728	0.2725
Precision	77.76%	79.06%
Recall	71.77%	71.75%
F1 Score	74.38%	74.27%
Accuracy	90.28%	90.53%

TABLE 2: Comparison of LSTM and BiLSTM performance on the dataset.

3.5 Hybrid Model (LSTM + CNN)

As the final model of this project, we explored a "Hybrid Model" that combines both LSTM and Convolutional Neural Networks (CNNs), inspired particularly by the approach proposed by Colón-Ruiz, C., & Segura-Bedmar, I. (2021) in their paper *"Comparing Deep Learning Architectures for Sentiment Analysis on Drug Reviews"* (Journal of Biomedical Informatics) [1].

The Hybrid Model combines Bidirectional LSTM and Convolutional Neural Networks (CNN) for sentiment analysis, utilizing pre-trained GloVe embeddings. The architecture consists of the following components:

Input Layer: Accepts tokenized and padded sequences of length *max_length*.

Embedding Layer: Uses pre-trained GloVe embeddings, non-trainable, to convert input text into dense vector representations.

Dropout Layer (Embedding): Applied with a rate of 0.5 to prevent overfitting after the embedding layer.

Bidirectional LSTM Layer: A Bidirectional LSTM with 250 units processes the input text in both forward and backward directions, capturing both past and future context.

Convolutional Layers: Two 1D convolutional layers with ReLU activation:

One with 32 filters and a kernel size of 2.

One with 32 filters and a kernel size of 3.

MaxPooling Layers: Max-pooling layers (pool size = 2) applied to the outputs of the convolutional layers.

Flatten Layers: Flattens the pooled outputs to create 1D vectors.

Concatenate Layer: Concatenates the flattened outputs from the max-pooling layers.

Dropout Layer (Concatenated): Applied with a rate of 0.5 to prevent overfitting after concatenation.

Dense Layer: A fully connected dense layer with 128 units and ReLU activation.

Output Layer: A softmax output layer for multi-class sentiment classification.

Metric	Original Dataset	Balanced Dataset
Loss	0.217	0.2294
F1	77.12	91.90%
Accuracy	91.23	91.89%
Precision	81.2	91.99%
Recall	74.3	91.88%

TABLE 3: Performance of models on both datasets.

4 Conclusions

In this study, we initially applied classical machine learning algorithms, starting with Logistic Regression, and later explored deep learning models like CNN, LSTM, and BiLSTM. When testing on the imbalanced dataset, the models struggled with minority classes. To address this, we applied oversampling, which allowed us to test only two models—CNN and our Hybrid Model—on the balanced dataset due to increased computational costs.

Our experiments showed that the CNN model offered the best balance between performance and training efficiency. While the Hybrid Model showed strong performance, its higher computational cost made fine-tuning challenging. Therefore, the CNN model emerged as the most cost-effective choice, delivering solid results with manageable training time and resource consumption.

In this section, we present the confusion matrix after applying oversampling to balance the dataset. The matrix shows the number of true positives, false positives, true negatives, and false negatives for each class, which helps assess the model's ability to classify samples correctly.

From the confusion matrix, we observe that oversampling has reduced misclassification of the minority classes (Negative and Neutral), allowing the model to better distinguish between classes and achieve more balanced performance.

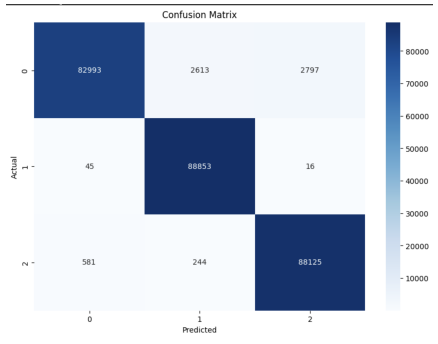

FIGURE 4: Confusion Matrix after Oversampling

5 Work Plan

The project began in November 2024, with the primary goal of developing a model that effectively addresses class imbal-

ance and delivers stable performance. We also aimed to compare the model's performance on a balanced dataset, particularly by evaluating the impact of different pre-trained embeddings. We experimented with both GloVe and Word2Vec embeddings, each with a dimension of 100. However, we observed that the models consistently performed similarly using both embeddings, leading us to proceed with GloVe and discard Word2Vec from our project.

Throughout the study, Google Colab's computational units were utilized for resources, which was a key constraint in testing different models and setups. The limitations of these resources influenced the scope of model testing and experimentation.

References

[1] C. Colón-Ruiz and I. Segura-Bedmar. Comparing deep learning architectures for sentiment analysis on drug reviews. Journal of Biomedical Informatics, 2021.

[2] A. Graves, A. R. Mohamed, and G. Hinton. Speech recognition with deep recurrent neural networks. IEEE Transactions on Audio, Speech, and Language Processing, 22(4):586–597, 2013.

[3] Hannah Kim and Young-Seob Jeong. Sentiment classification using convolutional neural networks. 2019.

[4] Guillaume Lemaître, Fernando Nogueira, and Christos K Aridas. Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning. Journal of Machine Learning Research, 18(17):1–5, 2017.

[5] U.B. Mahadevaswamy and P. Swathi. Sentiment analysis using bidirectional lstm network. Journal of Advanced Computing Research, 12(3):45–56, 2021.

[6] Julian McAuley and Jure Leskovec. From amateurs to connoisseurs: Modeling the evolution of user expertise through online reviews. In Proceedings of the 22nd International World Wide Web Conference (WWW), 2013.

[7] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1532–1543, 2014.

[8] Yufei Xie and Rodolfo C. Raga Jr. Convolutional neural networks for sentiment analysis on weibo data: A natural language processing approach. 2020.