

# Amazon Fine Foods Review Sentimental Analysis

Li, Shuhan, [li.shuhan1@northeastern.edu](mailto:li.shuhan1@northeastern.edu)

He, Genshi, [he.ge@northeastern.edu](mailto:he.ge@northeastern.edu)

## Abstract

This study analyzes the Amazon Fine Foods Review dataset. Employing three Natural Language Processing (NLP) techniques—Simple Neural Network, Convolutional Neural Network (CNN), and Long Short-Term Memory (LSTM) network—we aim to classify reviews based on their content. Our findings reveal that the LSTM network demonstrates the highest accuracy in classifying user reviews.

## Introduction

The proliferation of online shopping has led to an exponential increase in user-generated content, such as product reviews, which significantly influence consumer behavior and business strategies. This study focuses on the Amazon Fine Foods Review dataset, a comprehensive collection of over 500,000 user reviews, which serves as a rich corpus for applying and evaluating various Natural Language Processing (NLP) techniques. Our project aims to delve into this dataset, exploring the relationship between textual reviews and user satisfaction levels by categorizing scores into two classes: unsatisfied, and satisfied.

The motivation for us in this project is that understanding user attitude through reviews can provide businesses with insights into consumer satisfaction and product quality, which will benefit decision-making. This project also offers a real-world application of text classification methods.

We applied four different NLP approaches to achieve our goal: a Simple Neural Network, a Convolutional Neural Network (CNN), Long Short-Term Memory (LSTM) network, and CNN-LSTM combined network. The choice of these methods is predicated on their proven effectiveness in text analysis and classification tasks. The Simple Neural Network serves as a baseline, the CNN exploits spatial hierarchy in data, and the LSTM is adept at handling sequential information, crucial for understanding the context of textual data.

For our experiments and evaluation, we utilize the Amazon Fine Foods Review dataset available at Kaggle ([Kaggle: Amazon Fine Food Reviews](#)). This dataset encompasses over 500,000 records, including user ratings (Score) and comments (Text), making it an ideal corpus for our project.

Through this research, we aim to elucidate the capabilities of each NLP method in the context of text categorization, with a particular focus on their applicability to real-world data. The ultimate goal is to identify the most effective approach for classifying user reviews into our defined sentiment categories, thereby contributing valuable insights to the field of NLP and sentiment analysis.

## Background

Currently, simple neural networks are mainly applied in NLP for some straightforward text classification tasks, such as sentiment analysis or spam detection.

Convolutional Neural Networks (CNNs) are a type of deep learning architecture originally designed for image recognition tasks, but later found to be highly effective for processing textual data as well. The main idea of CNNs in NLP is to capture local features in text through convolutional layers, which is particularly effective in handling linguistic data. This is because many significant features in text, such as phrases or common word combinations, have local characteristics.

The latest LSTM models typically employed attention mechanisms, residual connections, and multi-layer structures to enhance model performance and stability on unstructured small text and *already achieved comparable performances with less parameters on sentiment analysis tasks*<sup>1</sup>. Additionally, pre-trained language models like BERT-LSTM and GPT-LSTM have made significant progress in the NLP field. The latest LSTM models typically employ attention mechanisms, residual connections, and multi-layer structures to enhance model performance and stability.

Additionally, pre-trained language models like BERT-LSTM and GPT-LSTM have made significant progress in the NLP field.

Few-shot Learning has emerged as a significant advancement, enabling models to achieve remarkable performance with minimal training data. According to [2], *they proposed framework, combination of Siamese CNNs and few-shot learning, leads to better results in accuracies on twitter classifications and outperforms some popular traditional text classification methods and a few deep network approaches.*

## Approach

### 1. Data preprocess

Amazon Fine Foods Review dataset encompasses over 500,000 records, including user ratings (Score) and comments (Text). The scores for this dataset are 1-5, we plan to categorize all records into two classes: Unsatisfied(1-2), and Satisfied(4-5). However, in this dataset, the distribution of scores is very disparate, which may affect the training of the model. Of these, 9.1% had a score of 1, 5.2% had a score of 2, 14.1% had a score of 4, and 63.8% had a score of 5. In order to avoid too many records in the "Satisfied" category affecting the accuracy of the model, we randomly selected 10,000 records from each score to generate a new dataset, then we labeled records with scores of 1-2 as "Unsatisfied", and records with scores of 4-5 as "Satisfied" and used this new dataset to train the models.

```
import pandas as pd
df = pd.read_csv('Reviews.csv')
df_1 = df[df['Score'] == 1].sample(n=10000, random_state=1)
df_2 = df[df['Score'] == 2].sample(n=10000, random_state=1)
df_4 = df[df['Score'] == 4].sample(n=10000, random_state=1)
df_5 = df[df['Score'] == 5].sample(n=10000, random_state=1)
df_extract = pd.concat([df_1, df_2, df_4, df_5])
df_extract.to_csv('Reviews_10000.csv', index=False)
```

### 2. Simple Neural Network

First of all, we applied the simple neural network in the dataset. The network is structured as a linear stack of layers beginning with an Embedding layer that maps each word to a

100-dimensional vector and is designed to handle input text of fixed length. It includes a GlobalAveragePooling1D layer to reduce the dimensionality of the output from the Embedding layer by averaging over the sequence dimension. This is followed by a Dense layer with a sigmoid activation function that outputs the probability of the input belonging to one class. The network uses a Stochastic Gradient Descent (SGD) optimizer with a learning rate of 0.01 and a momentum of 0.9 to optimize the binary cross-entropy loss function.

```
[ ] # Construct simple NN network
model = tf.keras.Sequential()
model.add(tf.keras.layers.Embedding(3000, 100, input_length=max_length))
model.add(tf.keras.layers.GlobalAveragePooling1D())
model.add(tf.keras.layers.Dense(1, activation='sigmoid'))

# Creating an instance of the Stochastic Gradient Descent(SGD) optimiser
optimizer = tf.keras.optimizers.SGD(learning_rate=0.01, momentum=0.9)

# Compiling the model, using the SGD optimiser
model.compile(optimizer=optimizer,
              loss='binary_crossentropy',
              metrics=['accuracy'])
```

### 3. Convolutional Neural Network (CNN)

The model starts with an Embedding layer to convert text data into dense 100-dimensional vectors, followed by a Conv1D layer with 128 filters to extract features. A GlobalMaxPooling1D layer reduces the output to the most significant features. The model also includes a Dense layer with 10 neurons for additional learning and concludes with a Dense layer with a sigmoid activation for binary output. The model is compiled using a previously defined SGD optimizer, using binary cross-entropy as the loss function and tracking accuracy.

```
# Construct Convolutional Neural Network
model_cnn = tf.keras.Sequential()
model_cnn.add(tf.keras.layers.Embedding(input_dim=3000, output_dim=100, input_length=max_length))
model_cnn.add(tf.keras.layers.Conv1D(filters=128, kernel_size=5, activation='relu'))
model_cnn.add(tf.keras.layers.GlobalMaxPooling1D())
model_cnn.add(tf.keras.layers.Dense(10, activation='relu'))
model_cnn.add(tf.keras.layers.Dense(1, activation='sigmoid'))

# Compiling the model
model_cnn.compile(optimizer=optimizer, loss='binary_crossentropy', metrics=['accuracy'])
```

### 4. Long Short-Term Memory (LSTM) network

To enhance our comprehension of the sequential dynamics inherent in the textual data, we opted to employ an LSTM network as the foundational model for our analysis. LSTM networks are particularly adept at processing sequences and retaining information over extended periods, which makes them highly suitable for the nuanced task of text analysis.

In addition to the basic LSTM architecture, we integrated a bi-directional LSTM (Bi-LSTM) framework. This advanced configuration allows the network to analyze the text data from both

forward and backward directions, effectively capturing the complex relationships between words that may be missed when only a single direction is considered. The bidirectional approach is instrumental in understanding the context surrounding each word, thus providing a more holistic analysis of the textual content.

```
1 # model initialization
2 model = tf.keras.Sequential([
3     tf.keras.layers.Embedding(vocab_size, embedding_dim, input_length=max_length),
4     tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(64, return_sequences=True)),
5     Attention(return_sequences=False),
6     tf.keras.layers.Dropout(0.5),
7     tf.keras.layers.Dense(32, activation='relu'),
8     tf.keras.layers.Dense(1, activation='sigmoid')
9 ])
10 # compile model
11 model.compile(loss='binary_crossentropy',
12               optimizer=tf.keras.optimizers.SGD(learning_rate=0.03, momentum=0.8),
13               metrics=['accuracy'])
14 # model summary
15 model.summary()
```

To further enhance the model's capacity to handle complex patterns in the text data, we implemented a hybrid architecture combining Convolutional Neural Networks (CNN) with Long Short-Term Memory (LSTM) networks. This combination leverages the strengths of both CNNs and LSTMs to improve feature extraction and sequence modeling capabilities.

```
1 model_cb = tf.keras.Sequential([
2     tf.keras.layers.Embedding(vocab_size, embedding_dim, input_length=max_length),
3     tf.keras.layers.Conv1D(filters=128, kernel_size=8,
4                             strides=1,
5                             activation=activation,
6                             padding='causal'),
7     tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(64, return_sequences=True)),
8     Attention(return_sequences=False),
9     tf.keras.layers.Dense(32, activation=activation, kernel_regularizer = regularizer),
10    tf.keras.layers.Dense(1, activation='sigmoid')
11 ])
12 # compile model
13 model_cb.compile(loss='binary_crossentropy',
14                  optimizer=tf.keras.optimizers.Nadam(learning_rate=0.001),
15                  metrics=['accuracy'])
16 # model summary
17 model_cb.summary()
```










Following the embedding layer, we introduce a convolutional layer. This layer is designed with 128 filters and a kernel size of 8, which is intended to convolve across the text to capture relevant features such as key phrases or patterns that are crucial for understanding the context. The convolution operates with a stride of one and utilizes the LeakyReLU activation function with an alpha value of 0.01 to introduce non-linearity to the model, enabling it to learn more complex patterns. Furthermore, we employ 'causal' padding to ensure that the convolution output has the same length as the input, meaning that the layer does not look ahead of the position being predicted.

Directly succeeding the CNN layer is the bidirectional LSTM layer, with 64 units and an attention mechanism that does not return sequences, to focus the model on the most relevant parts of the input for making predictions. By setting 'return\_sequences' to True, we ensure that the subsequent attention mechanism has access to the full sequence output from the LSTM layer. The dense layers that follow serve as fully connected layers where the extracted and processed features are integrated and passed through. The first dense layer contains 32 neurons and utilizes the same LeakyReLU activation function with the addition of a kernel regularizer to reduce overfitting by applying L2 regularization.

This hybrid approach, combining CNN for feature extraction with LSTM for sequence processing, aims to increase the model's complexity and improve its ability to capture both the local and global textual features, thereby enhancing the accuracy of our review classification system.

## Result

Corpus to be used:

# Id	# ProductId	# UserId	# ProfileName	# HelpfulnessName...	# HelpfulnessDeno...	# Score	# Time	# Summary	# Text
Row Id	Unique Identifier for the product	Unique Identifier for the user	Profile name of the user	Number of users who found the review helpful	Number of users who indicated whether they found the review helpful or not	Rating between 1 and 5	Timestamp for the review	Brief summary of the review	Text of the review
									
1	B001E4K7G8	A35DXH7A9HUBW	deImartian	1	1	5	1383862488	Good Quality Dog Food	I have bought several of the Vitality canned dog food products and have found them all to be of good...
2	B00813GRG4	A1D87F6ZVE5NK	d11 pa	0	0	1	1346976988	Not as Advertised	Product arrived labeled as Jumbo Salted Peanuts...the peanuts were actually small sized unsalted. No...
3	B000L00CH8	ABXLRWJXXAIN	Natalia Corres "Natalia Corres"	1	1	4	1219017688	"Delight" says it all	This is a confection that has been around a few centuries. It is a light, pillowy citrus gelatin wi...
4	B000UABQIQ	A395B0RC6FOVVV	Karl	3	3	2	1387923288	Cough Medicine	If you are looking for the secret ingredient in Robitussin I believe I have found it. I got this in...

[Kaggle: Amazon Fine Food Reviews](#)

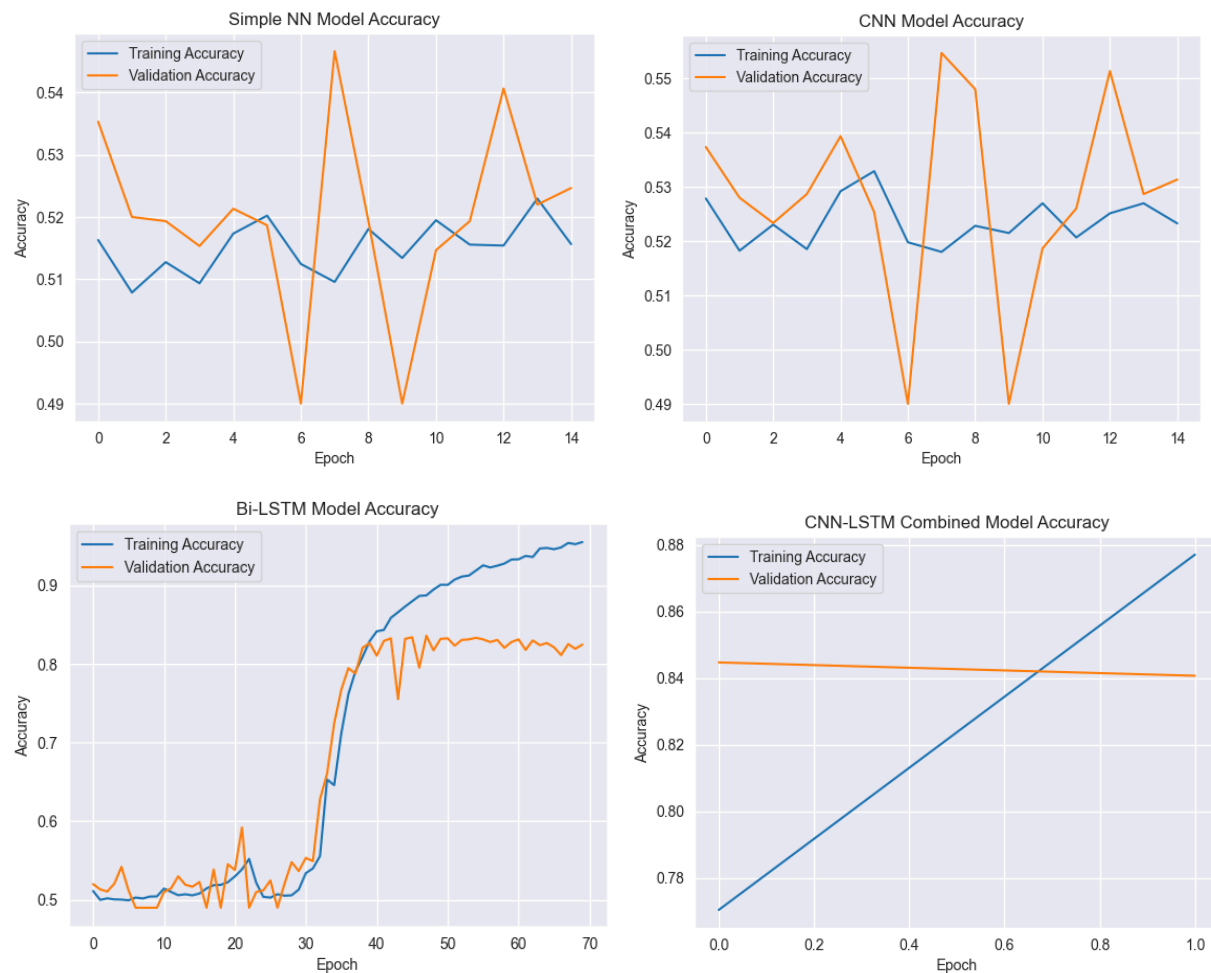
We used "Score" and "Text" in this corpus to train the models. We labeled records with scores of 1-2 as "Unsatisfied", records with scores of 3 as "Neutral", and records with scores of 4-5 as "Satisfied". Then we use Text as the training feature to classify customer experience as "Satisfied" and "Unsatisfied".

Experiments:

To test the models' efficiency and accuracy in the same binary classification task, we defined a plot function to evaluate the model's performances.

```
def plot_acc(his):
    plt.plot(his.history['accuracy'], label='Training Accuracy')
    plt.plot(his.history['val_accuracy'], label='Validation Accuracy')
    plt.title('Model Accuracy')
    plt.ylabel('Accuracy')
    plt.xlabel('Epoch')
    plt.legend()
    plt.show()
```

The performance results and accuracy are listed in the image below.



Model Name	Test Accuracy
Simple NN	51.22%
CNN	50.64%
Bi-LSTM	82.16%

CNN-LSTM	84.30%
----------	--------

The data depicted in the preceding charts offer a revealing look at the model's learning trajectory. In the simple NN and CNN model, we observe that both the training and validation losses hover around the 0.5 mark post the initial 15 epochs, indicating a lack of significant learning progress. Moreover, the fluctuations in these metrics suggest a certain instability in the model's learning process at this stage. The corresponding test accuracy also reflects a suboptimal performance, plateauing at approximately 50%.

In the case of the Bi-LSTM model, the early epochs up to the 30th mark show a rather gradual convergence, hinting at a slower learning rate during the initial phase. As the epochs advance from 30 to 50, the model's learning pace accelerates, as reflected in the convergence rate. Nevertheless, a performance plateau becomes evident post the 50-epoch threshold, where the validation accuracy stalls at around 82%, despite the training accuracy inching closer to 95%. It is worth noting that after surpassing 70 epochs, the Bi-LSTM model's accuracy achieves a stable around 82.64%.

In our final model that integrates CNN with LSTM, we recorded an impressive accuracy peak of 84.03%. Given the model's rapid convergence, we restricted the training to a brief span of two epochs. This decision was informed by the model's propensity to overfit post the 4-epoch mark—a trend manifested by a decline in validation accuracy in contrast to the increasing training accuracy.

#### Discussion:

The trends we've observed, such as early stagnation and overfitting, are not uncommon in the domain and prompt a reflection on the complexity of natural language and the subtleties involved in textual interpretation by machine learning models.

The performance metrics of our models highlight the delicate balancing act required between model complexity, training duration, and the risk of overfitting. They underscore the necessity for intricate architectures that can not only capture nuances in large-scale textual data but also maintain robustness when exposed to new, unseen data.

Further exploration of hyperparameter tuning through techniques like grid search or random search could lead to more optimal settings that strike a better balance between training and validation accuracies. Another aspect is the investigation of transfer learning and pre-trained models. Leveraging models that have been pre-trained on large, diverse corpora could provide a starting point that carries a more general understanding of language, potentially improving accuracy and convergence speed. However, to better understand the text context, it might be useful to use other more complex embedding methods and attention mechanisms.

#### Conclusion:

In this project, we set out to analyze the Amazon Fine Foods Review dataset to classify user reviews based on their content. Our approach was grounded in the use of three Natural Language Processing (NLP) techniques—Simple Neural Networks, Convolutional Neural Networks (CNN), and Long Short-Term Memory (LSTM) networks. The rationale behind selecting these methods lies in their proven

effectiveness in parsing and interpreting complex sequence data such as text. The LSTM network, particularly when enhanced with bi-directional capabilities and attention mechanisms, demonstrated the highest accuracy, reaching 82.64% post 70 epochs. However, the CNN-LSTM hybrid model outperformed this, achieving an accuracy of 84.03% within just two epochs, albeit with signs of overfitting after the fourth epoch. While more complex models like the CNN-LSTM can lead to rapid and significant improvements in accuracy, they also require careful tuning to avoid overfitting. This balance is crucial in the development of effective NLP systems for sentiment analysis.

Adding to our future trajectory, we envisage the adoption of more sophisticated embedding methodologies and attention mechanisms, with an eye toward the potential of transformer models. The transformer architecture, known for its exceptional ability to handle long-range dependencies within text, presents an exciting frontier for further enhancing the accuracy and depth of sentiment analysis. This evolution in model complexity will unlock new dimensions in NLP's application to voluminous and complex datasets.

### **References:**

1. Hassan, A., and A. Mahmood. "Deep Learning Approach for Sentiment Analysis of Short Texts." 2017 3rd International Conference on Control, Automation and Robotics (ICCAR), 2017, Nagoya, Japan, pp. 705-710. IEEE, doi:10.1109/ICCAR.2017.7942788.
2. Yan, L., Zheng, Y., & Cao, J. "Few-shot Learning for Short Text Classification." *Multimedia Tools and Applications*, vol. 77, no. 22, 2018, pp. 29799–29810. Springer, doi.org/10.1007/s11042-018-5772-4.

### **Contribution:**

Shuhan Li: write the Bi-LSTM and CNN-LSTM model, train the data, and write the part of the final report and slides

Genshi He: write the Simple NN and CNN model, trained the data, and write the part of the final report and slides