# Sentiment Analysis Using Feedforward Neural Network and Word2Vec Embeddings

**1. Introduction**

Background:

Sentiment analysis is becoming more and more necessary as user-generated material on social media and e-commerce platforms grows. The objective of this project is to use machine learning to categorize the sentiment (positive or negative) of the text data. In order to accomplish classification tasks, we created a Feedforward Neural Network (FNN) that uses Word2Vec embeddings for text representation.

Project Overview: In this research, the strengths of a Feedforward Neural Network (FFNN) and Word2Vec embeddings are combined to create a machine learning-based sentiment analysis system. The main objective is to categorize user evaluations as either positive or negative by utilizing an effective neural network architecture and word semantic meaning.

In this method, raw text data is preprocessed to eliminate noise before being converted into useful vector representations using Word2Vec. After that, these word embeddings are fed into an FFNN, which is made to identify patterns in the data and forecast sentiment accurately. Numerous metrics, such as F1-score, recall, accuracy, precision, and a confusion matrix, are used to assess the model once it has been trained on a dataset of labeled reviews.

In order to illustrate the efficacy of combining Word2Vec embeddings with an FFNN for natural language processing tasks, this research will examine the benefits and drawbacks of this approach. Investigating more sophisticated embedding strategies, including transformer-based models like BERT, could increase performance in the future.

## Team Members and Tasks:

Dinesh devisetti : Data preprocessing , Word2Vec embedding generation.

Pattem Jaya Sankar : FNN architecture design , Training and evaluation .

Rohith : Training and evaluation , Report writing and results .

## Objectives:

* Preprocess raw text data to remove noise and extract meaningful patterns.
* Represent textual data using Word2Vec embeddings.
* Train an FNN classifier with at least three layers to identify sentiment.
* Evaluate the model’s performance utilizing parameters such as the confusion matrix, the F1 score, recall, accuracy, and precision.
* Discuss the effectiveness of Word2Vec embeddings in combination with FNN for sentiment analysis.

**2. Dataset and Preprocessing**

Dataset:

* + User reviews with the associated sentiment labels (positive or negative) make up the dataset used in this study. For the sentiment classification model to be trained and assessed, this dataset is essential.
  + • Qualities:
  + o Review (Textual Content): Every entry includes a text review that conveys the user's viewpoint regarding a good or service..

Sentiment (Target Label): The sentiment label associated with each review, either favorable or unfavorable, reflecting the review's sentiment.

* Size:
* **Total reviews are enormous in this dataset. This large dataset will givethe needed diversity during the training of the sentiment analysis model.**

**CLASS DISTRIBUTION:  
o The dataset is balanced between the two sentiment classes. The distribution is as follows:  
•  
o There is a balance between the two sentiment classes in this data. The distribution is as follows:Positive Reviews:** [Include the number, e.g., 12,500 reviews]

**Negative Reviews:** [Include the number, e.g., 12,500 reviews]

* + The even distribution ensures that the model is not biased towards one class, making the training more effective and the evaluation metrics more reliable.

**Preprocessing Steps**

Preprocessing is necessary as a preparation step for text data to be fedinto the Word2Vec model and Feedforward Neural Network. The following steps represent the text-cleaning methodology that wascarried out on the dataset.

1. Noise Removal:  
Objective: Remove extra characters and symbols which are not usefulfor sentiment analysis.  
• Preprocessing Method: Reviews were subjected to are removal of special characters, punctuation, URLs, and numbers using regularexpressions.  
• Example:

* + Input: "This product is great! 😃 Visit [https://example.com](https://example.com/)"
  + Output: "this product is great"

Lowercasing:

* Objective: Standardize the text by converting all characters to lowercase to ensure uniformity and avoid treating the same word differently based on capitalization.
* Method: All reviews were converted to lowercase.
* Example:
  + Input: "GREAT Product!"
  + Output: "great product"

Tokenization:

* objective: Tokenize the cleaned text into separate words so thatWord2Vec and FNN model may process it easily.  
  • Procedure: The text was tokenized by splitting the clean string at white spaces or punctuation, thereby forming a list of words.
* Example:
  + Input: "great product"
  + Output: ["great", "product"]

Stopword Removal:

* Objective: Remove common stopwords (words that carry little meaning, such as "the", "is", "and") that can negatively affect the model’s performance.
* Method: Used NLTK’s pre-defined stopword list to filter out stopwords from the tokenized text.
* Example:
  + Input: ["the", "great", "product", "is", "amazing"]
  + Output: ["great", "product", "amazing"]

Lemmatization:

* Objective: Reduce words to their base form or lemma to ensure that different forms of the same word (e.g., "running" vs. "run") are treated as the same word.
* Method: Used the WordNetLemmatizer from NLTK to lemmatize the words.
* Example:
  + Input: ["running", "better"]
  + Output: ["run", "good"]

**A screen shot of a computer program

Description automatically generated**

**3. Methodology**

**Word2Vec Embeddings:**

1. Word embeddings, such as Word2Vec, are one way to project words into a dense vector representation in a continuous vector space. The goal of embeddings in NLP is to capture semantic and syntactic wordrelations in a numerical way so that machine learning algorithms can use them effectively.  
   Unlike classic one-hot encoding, where every word would be represented as a sparse vector and there was no relationship inherent between different words, embeddings ensure that words with similar meanings will have vectors that are close to each other in that vector space. This makes embeddings memory-efficient and capable of capturing meaningful word relationships, which becomes crucial for tasks like sentiment analysis, translation, and text classification.
2. **Pre-trained Word2Vec**:

**What it is:** Pre-trained embeddings are generated from large corpora (e.g., Google News) and contain rich semantic and syntactic information.

**Why used:** Saves computation time and provides a strong foundation of word representations, especially for general language tasks.

**Details:** In this project, we used Google’s Word2Vec model with 300-dimensional embeddings. These embeddings encode similarities between words, such as "king" and "queen" or "happy" and "joyful."

**Advantages:**

* + - Eliminates the need to train embeddings from scratch.
    - Encodes universal language patterns.
  + **Example:** Words like “good” and “great” might have vectors with high cosine similarity due to their semantic closeness.

1. **Training Word2Vec on the Dataset**:

**What it is:** Instead of using pre-trained embeddings, a Word2Vec model is trained on the project’s specific dataset to capture domain-specific vocabulary and context.

**How it works:** We trained a Word2Vec model using the Skip-Gram method, where the model predicts the context words given a target word.

**Parameters used:**

* Vector size: 100
* Window size: 5
* Minimum word count: 1 (to include infrequent words).

**Why used:** Captures nuances and specific meanings unique to the dataset.

**Example:** In a dataset of reviews, "excellent" and "outstanding" might develop stronger contextual relationships than in generic embeddings. Trained a Word2Vec model using Gensim with the following parameters:

1. **Embedding Representation of Reviews**:

**What it is:** Individual word embeddings in a review are combined to create a single vector representing the entire review.

**How it works:**

For a review, the embeddings of all its words are averaged (or sometimes summed) to produce a single fixed-length vector.

* + **Why used:** Converts variable-length text into a consistent input format for the Feedforward Neural Network.
  + **Example:**
    - Review: “great product”
    - Word embeddings:
      1. Vector for "great": [0.45, 0.32, …, 0.76]
      2. Vector for "product": [0.39, 0.48, …, 0.66]
    - Review embedding: Mean of vectors for "great" and "product."

**Feedforward Neural Network (FNN):**

**1. Model Architecture**

A the Multi-Layer (MLP), also referred to as a Feedforward Neural Network (FNN), was meticulously designed to classify text reviews into two sentiment categories: positive or negative. Below are the detailed components:

1. **Input Layer**:
   * Size: The input layer consists of 100 units, where each unit represents a single dimension of the review's Word2Vec embedding.
   * Purpose:

* Accepts the input data, which is the averaged Word2Vec embedding of a review.
* Serves as the starting point for feature extraction within the neural network.
* Input Shape: For a batch of nn reviews, the input shape is (n,100)(n,100), where 100 corresponds to the Word2Vec embedding size.

1. **Hidden Layers**:
   * First Hidden Layer:
     + Size: 128 units (neurons).
     + Activation Function: ReLU (Rectified Linear Unit).
     + Purpose:
       - Extracts meaningful features by applying a linear transformation followed by the ReLU activation.
       - Captures non-linear relationships in the data, such as interactions between words that influence sentiment.
   * Second Hidden Layer:
     + Size: 64 units.
     + Activation Function: ReLU.
     + Purpose:
       - Further refines the extracted features.
       - Reduces dimensionality while preserving essential information, enabling the model to focus on the most critical patterns for classification.
2. **Output Layer**:
   * Size: 2 units, corresponding to the two sentiment classes: positive and negative.
   * Activation Function: Softmax.
   * Purpose:
     + Converts the raw outputs (logits) from the previous layer into probabilities that sum to 1.
     + Helps determine the likelihood of each class (e.g., a review might have an 85% probability of being positive and 15% of being negative).
   * Output Shape: For a batch of nn reviews, the output shape is (n,2)(n,2).

**2. Activation Functions**

* Activation functions are crucial for introducing non-linearities into the model, enabling it to learn complex patterns. Here are the details of the activation functions used**:**

1. **ReLU (**Rectified Linear Unit):
   * Formula:f(x)=max⁡(0,x)f(x)=max(0,x)
   * Purpose:
     + Efficiently handles non-linear patterns in the data, ensuring the model can learn interactions between words and their contextual meanings.
     + Prevents the vanishing gradient problem, which occurs when gradients shrink to near-zero values in deep networks.
     + Allows only positive values to pass through while setting all negative values to zero, simplifying computation and improving convergence speed.
   * Why ReLU for Hidden Layers?
     + **It is computationally efficient.**
     + **Improves the model's ability to learn meaningful relationships from data.**
2. Softmax:

​

Purpose:

* + Converts the raw outputs (logits) from the model into probabilities for each class.
  + Ensures the probabilities of all output classes sum to 1, making it suitable for multi-class classification.

Why Softmax for Output Layer?

* + Provides interpretable probabilities, which are helpful for classification tasks and evaluation metrics.
  + Enables easy determination of the predicted class by selecting the class with the highest probability.

3. Regularization

To prevent the model from overfitting (memorizing noise in the training data), dropout layers were incorporated into the architecture.

1. Dropout Layers:

* What it is: A regularization technique where a random subset of neurons is "dropped" (temporarily set to zero) during training.
* Purpose:
  + Forces the model to rely on multiple neurons rather than a few dominant ones.
  + Encourages the network to generalize better to unseen data.
* Implementation: Applied after each hidden layer with a dropout rate (e.g., 0.2, meaning 20% of the neurons are dropped).

**4. Training Process**

1. **Data Splitting**:

1. Data Splitting :

The dataset was split up into the following subsets in order to efficiently train and assess the model:

80% of the dataset is the training set.

Its purpose is to train the model by letting it discover relationships and patterns in the data.

Details:

Contains labeled examples for both positive and negative sentiments.

Ensures that the model has sufficient data to generalize across a variety of scenarios.

Examples are shuffled to prevent the model from learning order-dependent biases.

Impact: By allocating 80% of the data for training, the model has access to a broad range of samples for robust learning.

Testing Set (20% of the dataset):

Purpose: Evaluates the model's performance on unseen data after training is complete.

Details:

Composed of balanced examples from both sentiment classes.

The test set is never seen by the model during training, ensuring an unbiased measure of generalization.

Metrics Evaluated: F1-score, confusion matrix, recall, accuracy, and precision.  
10% of the training data is optionally included in the validation set:

Purpose: Monitors the model's generalization performance during training.

Details:

A 10% subset of the training data is set aside before training starts.

Helps tune hyperparameters and determine the stopping point to avoid overfitting.

The validation loss is periodically evaluated to guide adjustments to the model's training process.

Impact: Provides a safety check to ensure the model isn’t overfitting to the training set.

2. Loss Function

The model's performance during training is optimized using Categorical Crossentropy as the loss function. Here’s why and how it works:

Why Categorical Crossentropy?

The output layer uses Softmax activation, producing probabilities for the binary classes (positive and negative sentiment).

Categorical Crossentropy measures the difference between the predicted probability distribution and the true class labels.

Ideal for classification tasks with mutually exclusive classes.

Impact:

Minimizing this loss function ensures that the model learns to assign high probabilities to the correct classes.

3. Optimization

The Adam optimizer was used for gradient descent optimization.

Why Adam?

combines the benefits of RMSProp and AdaGrad, two well-known optimizers.

enhances convergence speed and efficiency by adaptively modifying learning rates for every parameter.

Works well with sparse data and noisy gradients, common in text data.

Key Features of Adam:

Uses momentum to accelerate learning.

Employs adaptive learning rates for faster convergence.

Parameters:

Learning rate (αα): 0.001, providing a balance between fast convergence and stability.

Stop Criteria

To prevent overfitting and unnecessary computation, early stopping was implemented.

What is Early Stopping?

A training process is halted if the validation loss does not improve for a specified number of epochs (patience).

Prevents the model from overfitting to the training data once validation performance stagnates.

Implementation:

Monitored the validation loss after each epoch.

Training stopped if no improvement was observed in validation loss for a set number of consecutive epochs.

Impact:

Ensured efficient use of resources.

Prevented the model from memorizing noise in the training set.

**5. Hyperparameters**

The following hyperparameters were chosen to optimize training efficiency and performance:

**Batch Size:** 32

**Purpose:** Divides the dataset into smaller groups of 32 samples for gradient updates.

**Impact:** Provides a balance between computational efficiency and convergence speed.

**Learning Rate:** 0.001

**Purpose:** Determines the size of updates to the model's parameters during optimization.

**Impact:** A moderate learning rate allowed for stable convergence without overshooting the optimal solution.

**Epochs:** 20

**Purpose:** Defines the number of times the model sees the entire training dataset.

**Impact:** Provided sufficient training iterations for convergence while preventing overfitting (aided by early stopping).

The dataset was split into:

80% training set.

20% testing set.

Optional 10% validation set (from the training data).

**5. Results**

**1. Accuracy**

* **Definition:**  
  The percentage of accurate predictions—both positive and negative—to the total number of samples is known as accuracy.
*  **Explanation:**  
  Accuracy indicates the model's overall ability to classify sentiments correctly. However, it may not always reflect performance accurately in cases of class imbalance.
*  **Achieved Accuracy: 84.57**

A screenshot of a graph

Description automatically generated

**2. Precision, Recall, and F1-Score**

To provide a detailed evaluation, these metrics are calculated for each class (positive and negative sentiment):

1. **Precision (Positive Predictive Value):**
   * **Definition:**  
     The precision metric calculates the proportion of accurately anticipated positive attitudes to all positive predictions.

 **Achieved Precision: 83 %**

* + **Purpose:**  
    Precision evaluates how often the model’s positive predictions are correct.

1. **Recall (Sensitivity):**

**Definition:**  
Recall quantifies the proportion of actual positive sentiments correctly identified by the mode

* **Achieved Recall: 87 %**
* **Purpose:**  
  Recall emphasizes the model’s ability to capture as many positive samples as possible.

**F1-Score (Harmonic Mean of Precision and Recall):**

* **Definition:**
* **The F1-score is a balanced statistic that takes into account both false positives and false negatives by combining precision and recall into a single metric.**.
*  **Achieved F1-Score: 85%**
*  Purpose**:**  
  When precision and recall must be traded off, the F1-score is especially helpful in ensuring that neither measure takes center stage in the assessment.

A screenshot of a graph

Description automatically generated

**3. Confusion Matrix**

The model's predictions for both sentiment classes are broken down in depth in the confusion matrix.

| **Actual Sentiment** | **Predicted Positive** | **Predicted Negative** |
| --- | --- | --- |
| **Positive** | True Positives (TP) | False Negatives (FN) |
| **Negative** | False Positives (FP) | True Negatives (TN) |

* **Definitions:**
  + **True Positives (TP):** Correctly classified positive reviews.
  + **False Positives (FP):** Negative reviews incorrectly classified as positive.
  + **True Negatives (TN):** Correctly classified negative reviews.
  + **False Negatives (FN):** Positive reviews incorrectly classified as negative.

**Purpose:**

* Provides a comprehensive view of the model’s performance for each sentiment class.
* Helps identify areas where the model struggles (e.g., high false negatives or false positives).

A computer screen with white text

Description automatically generated

**Visualization: Confusion Matrix Heatmap**

* **Heatmap:**
  + The confusion matrix was visualized using a heatmap for better interpretability. Each cell represents the count of predictions for a specific category, with color intensity indicating higher frequencies.
  + Example Tools: Python libraries such as Matplotlib or Seaborn were used for visualization.
* **Insights from the Heatmap:**
  + Cells along the diagonal (True Positives and True Negatives) should have the highest intensity, indicating correct predictions.
  + Off-diagonal cells (False Positives and False Negatives) highlight areas where the model misclassifies reviews.

A blue squares with white text

Description automatically generated

A graph of a curve

Description automatically generated

1. **Discussions**
2. **Effectiveness of Word2Vec with FNN**:
   * Word2Vec helped capture semantic and syntactic relationships between words by mapping them into a dense vector space. This allowed the model to understand the context and meaning behind words in the reviews, rather than just treating them as individual tokens. The ability of Word2Vec to represent words in a continuous vector space is one of the key factors that contributed to the success of this model. For example, words like "good" and "great" would have similar vector representations, allowing the model to recognize these as similar sentiments even if they are not identical.
3. **Feedforward Neural Network (FNN)**: The FNN model, with its multiple layers and ReLU activation functions, successfully learned non-linear relationships in the data. The two hidden layers (128 units and 64 units) enabled the network to capture complex patterns in the text, such as sentiment-related word combinations, and combine these features effectively for classification. The output layer, which used Softmax activation, allowed the model to provide probabilistic predictions for sentiment labels, making it ideal for this binary classification task.

Overall, the combination of Word2Vec for feature extraction and FNN for classification led to strong model performance, attaining high F1-score, recall, accuracy, and precision. This illustrates  that the integration of pre-trained embeddings with a neural network can be a powerful approach for sentiment analysis tasks.

1. **Pros and Cons**:
   * **Pros**:
     + **Scalability and Efficiency:**
     + Word2Vec embeddings provide a compact and efficient representation of words, which makes the model scalable even with large datasets. Using pre-trained embeddings (such as Google News Word2Vec) significantly reduces the computational cost, as it eliminates the need to train embeddings from scratch.
     + **Handling Complex Patterns:**
     + The FNN architecture, with its multiple layers and non-linear activation functions (ReLU), can learn complex patterns in data. This is crucial for sentiment analysis, as sentiment often depends on the context of words and their relationships.
     + **Generalization:**
     + The use of dropout layers for regularization helped prevent overfitting, allowing the model to generalize well to unseen data. Early stopping further ensured that the model did not overtrain, thereby improving the robustness of the model.
     + **Balanced Performance Metrics:**
     + The model achieved strong performance across several criteria, including F1-score, recall, and precision,indicating that it was effective in predicting both positive and negative sentiments.
   * **Cons**:
   * Out-of-Vocabulary (OOV) Words:

Word2Vec embeddings may struggle with words that were not present in the training corpus or out-of-vocabulary (OOV) words. In such cases, the model may not be able to capture the correct meaning of the word, leading to suboptimal performance. For example, words in highly domain-specific contexts or slang may not be represented well by general pre-trained embeddings.

* + **Dependence on Pre-trained Embeddings:**

While pre-trained Word2Vec embeddings can significantly improve performance and reduce computation, they may not fully capture domain-specific meanings and nuances. For example, in product reviews, certain words may have different meanings depending on the context that may not be well-captured by general embeddings.

* + **Limited Contextual Understanding:**

Word2Vec uses a fixed window size for learning word contexts, which may miss out on long-range dependencies in the text. For instance, it might fail to capture the full meaning of sentences where the sentiment is determined by interactions between words that are far apart.

1. **Further Improvements**: 

**Advanced Embedding Techniques (e.g., BERT or GPT):**

One of the limitations of Word2Vec is that it generates static embeddings for words, meaning the same word will have the same embedding regardless of its context. To address this, future improvements could include using more advanced contextualized word embeddings like GPT (Generative Pre-trained Transformer) or BERT (Bidirectional Encoder Representations from Transformers). Better performance may result from these models' generation of embeddings depending on the sentence's overall context especially in cases where sentiment depends on word context.

1. **Model Depth and Complexity:**

Although the FNN achieved good results, increasing the depth (number of hidden layers) or experimenting with more complex architectures such as **Convolutional Neural Networks (CNNs)** or **Recurrent Neural Networks (RNNs)** might help the model capture more intricate patterns in the text. For example, RNNs, particularly **LSTMs** (Long Short-Term Memory networks), are well-suited for tasks where the sequential nature of text is important.

1. **Hyperparameter Optimization:**

The performance of the model could potentially be improved by fine-tuning hyperparameters such as the number of hidden units, dropout rate, or learning rate. Methods like **grid search** or **random search** can be applied to find the optimal set of hyperparameters that minimize validation loss and improve model accuracy.

1. **Data Augmentation:**

To address class imbalance or underrepresentation of certain sentiments, data augmentation techniques could be applied. Techniques such as paraphrasing, back-translation, or synthetic data generation could help the model learn from more diverse examples, improving generalization.

1. **Fine-tuning Word Embeddings on Domain-Specific Data:**

If working with a specific domain (e.g., product reviews, movie reviews), it may be beneficial to train a Word2Vec model on domain-specific data. This would help the model better understand the specialized vocabulary and context specific to that domain, potentially leading to better sentiment predictions.

* + Use advanced embeddings like BERT for better contextual understanding.
  + Add more layers or tune hyperparameters for improved performance.

1. **Conclusions**

**The combination of Word2Vec embeddings and a Feedforward Neural Network proved to be a powerful approach for sentiment analysis. The model demonstrated strong performance across various metrics, highlighting its potential for sentiment classification tasks. However, there are areas for improvement, particularly in handling out-of-vocabulary words and incorporating contextual word representations. Future work may explore more advanced embedding techniques, model architectures, and hyperparameter optimizations to further enhance the model's performance.**

**The success of this approach sets a strong foundation for future research and applications in sentiment analysis, providing valuable insights into the effectiveness of neural networks in understanding and classifying sentiment in text.**

1. **References**

* **Word2Vec:**
* **Mikolov, T., Sutskever, I., Chen, K., Corrado, G., & Dean, J. (2013). *Distributed Representations of Words and Phrases and their Compositionality*. Advances in Neural Information Processing Systems, 26.**
* **Available at:**[**https://arxiv.org/abs/1310.4546**](https://arxiv.org/abs/1310.4546)

**Feedforward Neural Networks (FNN):**

* **Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.**
* **ISBN: 978-0387310732**

**Deep Learning:**

* **Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press.**
* **ISBN: 978-0262035613**
* **Available at:**[**http://www.deeplearningbook.org/**](http://www.deeplearningbook.org/)

**Adam Optimizer:**

* **Kingma, D. P., & Ba, J. (2015). *Adam: A Method for Stochastic Optimization*. In Proceedings of the International Conference on Learning Representations (ICLR).**
* **Available at:**[**https://arxiv.org/abs/1412.6980**](https://arxiv.org/abs/1412.6980)

**Early Stopping and Regularization:**

* **Prechelt, L. (1998). *Early Stopping - But When?* In *Neural Networks: Tricks of the Trade* (pp. 55-69). Springer.**
* **ISBN: 978-3540647224**

**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**

**Confusion Matrix Visualization:**

* **Hunter, J. D. (2007). *Matplotlib: A 2D Graphics Environment*. Computing in Science & Engineering, 9(3), 90-95.**
* **DOI: 10.1109/MCSE.2007.55**
* **Available at:**[**https://matplotlib.org/**](https://matplotlib.org/)

**Contextual Embeddings (BERT):**

* **Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. In Proceedings of NAACL-HLT 2019.**
* **Available at:**[**https://arxiv.org/abs/1810.04805**](https://arxiv.org/abs/1810.04805)

**Neural Network Hyperparameter Tuning:**

* **Bergstra, J., & Bengio, Y. (2012). *Random Search for Hyper-Parameter Optimization*. Journal of Machine Learning Research, 13, 281-305.**
* **Available at: http://www.jmlr.org/papers/volume13/bergstra12a/bergstra12a.pdf**

**Sentiment Analysis and Review Classification:**

* **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**

**Appendices**

1. Python Code:
   * Key preprocessing steps.
   * FNN implementation.
2. Screenshots:
   * Model training and evaluation results.
3. Dataset Sample:
   * Example rows from the dataset.