**II. Data Sources and Processing**

In this section, it is the way in which we obtain and process data. The dataset comprises two subsets: **aggressive data** and **non-aggressive data**, each containing 118,829 samples. The dataset is balanced in size, with columns labeled as "No." and "Message."

**A. Characteristics of the Dataset**

* **Aggressive Data**:
  + **40%**: Contains offensive language.
  + **30%**: Represents biased opinions on sensitive topics like race, religion, and politics.
  + The remaining samples include nonsensical or ambiguous phrases.
* **Non-Aggressive Data**:
  + Includes neutral or non-offensive comments.
  + Contains ambiguous words or phrases lacking clear intent.

**B. Dataset Partitioning**

The dataset was split into:

* **70% Training Set** for model development.
* **15% Testing Set** for performance evaluation.
* **15% Validation Set** for hyperparameter tuning.

**C. Data Preprocessing**

Effective preprocessing is crucial for reducing noise and ensuring meaningful features. We applied the following steps:

1. **Digit Removal**: Eliminated numbers to avoid numerical noise.
2. **Stop Word Filtering**: Removed high-frequency, low-information words like "is," "the," and "a."
3. **Punctuation Removal**: Stripped punctuation marks to ensure clean tokenization.
4. **Stemming**: Reduced words to their root forms to enhance semantic generalization (e.g., "running" → "run").
5. **Word Segmentation**: Tokenized sentences into individual words for feature extraction.

**III. Methodology**

**A. Traditional Approach: TF-IDF + Machine Learning Models**

Initially, we employed **TF-IDF (Term Frequency-Inverse Document Frequency)** to generate feature matrices. TF-IDF is defined as:

where:

* : Frequency of term in document .
* : is the total number of documents, is the number of documents containing .

After applying **PCA (Principal Component Analysis)** for dimensionality reduction and normalization, we trained models including:

* **Support Vector Machine (SVM)**
* **Random Forest (RF)**
* **Logistic Regression (LR)**

**B. Word2Vec + Multilayer Perceptron (MLP)**

While these models demonstrated reasonable performance, they lacked semantic awareness, as TF-IDF prioritizes frequency over context. To capture semantic relationships, we replaced TF-IDF with **Word2Vec embeddings**. Word2Vec uses a **Skip-Gram** model to predict a word’s context based on its neighbors:

where is the target word, is the context word, and represents the word vector.

**MLP Architecture**

The generated embeddings were fed into an MLP for classification:

1. **Input Layer**: Takes Word2Vec embeddings of sentences.
2. **Hidden Layers**:
   * First hidden layer: 128 neurons, **ReLU** activation.
   * **Dropout** (rate = 0.2).
   * Second hidden layer: 128 neurons, **ReLU** activation.
   * **Dropout** (rate = 0.5).
3. **Output Layer**: 2 neurons, **Softmax** activation.

Although this approach improved accuracy (from **79% to 86%**), equal weighting of words reduced robustness, especially in handling ambiguous or outlier words.

**C. Naive Bayes Weighted Average Embedding + Deep Feedforward Neural Networks**

The above two methods are simple averaging techniques where each word in the sentence has equal weight.

E.g.: Support a sentence contains N words , and word vectors are , then the traditional average word vector s of the sentence is:

To address the limitations of equal-weighting, we introduced **Naive Bayes Weighted Average (NBWA)** embeddings. This method calculates the frequency of each word in a particular category, then based on the ratio of those frequencies to measure the importance of that word in text classification.

Each word is given weight by calculating the log-count ratio for each word.

**Steps for NBWA**

1. **Generated word vector:**
   * Tool: Pre-trained word embedding models like Word2Vec. Skip-gram model learns semantic relationships based on context and can adjust dynamically based on surrounding words.
2. **Word Frequency Calculation:**
   * Compute word frequency in training data;
   * Calculate word frequency per class;
   * Compute the probability for each word given a class;
   * Apply Laplace Smoothing (Add 1 smoothing) to avoid zero probabilities;
3. **Log--Count Ratio**:
   * Calculate the log ratio between two classes for each word：

It measures the difference in the distribution of the word w in the two categories

1. **Weighted Average Vector**:
   * Compute the sentence embedding :

**Deep Feedforward Neural Network**

* **Layer 1**: Fully connected, 128 neurons, **ReLU** activation.
* **Layer 2**: Fully connected, 64 neurons, **ReLU** activation.
* **Layer 3**: Single neuron, **Sigmoid** activation.

**D. Adversarial Training and N-BEATS**

Despite high accuracy, our model was overly sensitive to extreme words (e.g., "fuck" or "stupid"), often misclassifying sentences like "You're so fucking beautiful" as aggressive due to the presence of the word "fuck."

**Adversarial Training**

To mitigate this, we performed adversarial testing by:

* 1. Manually creating positive or neutral sentences containing extreme words (e.g., "This is the shit I love").
  2. Incorporating these examples into the training set, forcing the model to focus on context and semantics rather than keywords.

**N-BEATS for Dynamic Word Embeddings**

We introduced **N-BEATS**, a time-series forecasting model, to dynamically adjust word embeddings based on sentence semantics. Unlike static embeddings, N-BEATS generates context-sensitive representations, improving the model’s ability to distinguish offensive from non-offensive text.

**E. Transformer with Multi-Head Attention**

Finally, we adopted a **Transformer model** with **Multi-Head Attention**, which computes:

where:

* , , are query, key, and value matrices.
* is the dimensionality of the key.

The attention mechanism allowed the model to:

* Focus on semantically important words.
* Capture long-term dependencies across sentences.

**IV. Experimental Results**

* 1. **Evaluation Results**

Our evaluation metrics included **accuracy**, **recall**, and **loss**. Table 1 compares different methods:

|  |  |  |
| --- | --- | --- |
| **Method** | **Accuracy** | **Comments** |
| TF-IDF + SVM/LR/RF | 79% | Limited by frequency-based representation. |
| Word2Vec + MLP | 86% | Improved semantic understanding but sensitive to word weight equality. |
| NBWA + Deep Feedforward Network | 99% | Robust, handles semantic context effectively. |
| NBWA + N-BEATS | ~99% | Handles adversarial cases, slight drop in overall accuracy. |
| Transformer with Attention | ~99% | Best semantic understanding, robust to adversarial samples. |

* 1. **Sensitive Word Analysis**
* Without adversarial training, 99% of sensitive-word-positive sentences were misclassified.
* Incorporating adversarial training and N-BEATS reduced this rate to **0.06%**.