Twitter Entity Sentiment Analysis

Group 3 Chen Zhikun, Du Zhanke, Hao Bohan, Wong Yan Ho Matthew

1. Project Overview

As a microblogging platform, Twitter (now as X) offers a rich source of user-generated content where sentiments are often embedded in short, informal, and highly nuanced messages. This project explores the effectiveness of various machine learning models—ranging from traditional classifiers to deep learning approaches—combined with different feature engineering techniques, including Bag-of-Words, word embeddings, and contextual representations like BERT. By experimenting with these combinations, we aim not only to identify the most effective sentiment classification pipeline but also to deepen our understanding of how feature choices and model architectures impact performance in short-text sentiment analysis.

2. Dataset

2.1 Dataset Analysis

The dataset used in this study is sourced from the **Twitter Entity Sentiment Analysis Dataset** provided by Kaggle user *passionate-nlp* (Kaggle, 2021). It comprises two CSV files (*training.csv* and *validation.csv*) with a total of 149,362 tweet records. Each entry includes four key fields: tweet ID (id), entity/category (entity), sentiment label (sentiment), and tweet content (content) (Table 1).

Exploratory data analysis was done before processing the data. During the process, we observed that there is a significant imbalance in the distribution of sentiment categories in the dataset, with negative samples accounting for more than 30%, which may affect the model's training effectiveness and generalization ability (Fig 1.1). Further, the distribution of tweet lengths as illustrated in Fig 1.2 shows a highly right-skewed pattern. Additionally, we discovered that 1,372 tweets do not have any value in the content field, providing minimal semantic content and potentially adding noise.

2.2 Data Preprocessing Methods

Given the noisy nature of social media text, we implemented a complete text preprocessing pipeline, mainly including two core components: text standardization and emoji processing. In the text standardization process, we first convert the text to lowercase, then sequentially remove URL links, HTML tags, replace @ mentions with [USER] markers, remove RT (retweet) markers, and normalize whitespace characters. These steps ensure the consistency and comparability of the text.

For emoji processing, we developed a specialized module that can extract emojis from tweets and convert them into corresponding text descriptions. This method preserves the emotional information carried by emojis while making them effectively utilized by subsequent text processing modules.

2.3 Dataset Division and Balancing Strategy

To ensure the reliability of model training and evaluation, we used stratified sampling to divide the dataset into training set (70%, 103,164 records), validation set (15%, 22,107 records), and test set (15%, 22,107 records). The split followed a 70-15-15 ratio for training, validation, and test sets respectively, ensuring reliable evaluation and hyperparameter tuning while avoiding data leakage. This also ensures that the category distribution in each subset remains consistent with the original dataset.

To address the imbalance in the training set, we implemented a comprehensive balancing strategy. For overrepresented categories (such as Negative and Positive classes), we reduced their sample sizes from 31,194 and 28,800 to 27,000 through random downsampling. For underrepresented categories (such as Neutral and Irrelevant classes), we performed data augmentation using synonym replacement via WordNet to generate semantically similar tweets to increase their sample sizes from 25,225 and 17,945 to 27,000. This approach ensures that each sentiment class contributes equally during model training, improving the robustness and fairness of the resulting models.

2.4 Preprocessing Effect Analysis

The preprocessed dataset has shown significant improvements in both quality and quantity. Text length analysis shows that the average text length after preprocessing decreased by 1.83 characters (approximately 1.67%). As shown in Fig 1.3, the preprocessing pipeline has led to modest reductions in tweet length. This suggests that while noise was effectively removed, the semantic core of the tweets remained largely unaffected. Additionally, 1,984 tweets (1.33% of the dataset) that were empty or contained only a single word were removed, as they offered minimal semantic value and could introduce noise during model training.

In terms of category balancing, we successfully adjusted the training set to contain 108,000 balanced data points, with 27,000 samples for each sentiment category. This balanced data distribution provides an ideal foundation for subsequent model training. The dataset now contains multiple fields including original content, cleaned content, and tokenization results, supporting text analysis at different levels.

3. Approaches

3.1 Feature Engineering

To effectively capture the semantic and syntactic information from tweets, we engineered five distinct types of text features, each reflecting different representational strengths. These features were generated from the cleaned dataset and used as input to various machine learning models.

Bag-of-Words (BoW): We applied CountVectorizer with a max_features limit of 5,000 and min_df (minimum document frequency) of 5. This representation captures word occurrence frequency without considering context. It serves as a strong, interpretable baseline for traditional classifiers like Logistic Regression and SVM.

TF-IDF (Term Frequency-Inverse Document Frequency): We used TfidfVectorizer with the same parameters (max_features=5000, min_df=5) to encode the importance of words based on their rarity across documents. TF-IDF helps downweight common terms and emphasize more informative ones, often improving performance in sparse text classification tasks.

Word2Vec Embeddings: We used pre-trained Word2Vec vectors to create dense, continuous embeddings for each word. Sentence-level features were obtained by averaging word vectors. This captures semantic similarity between words and is especially useful for handling lexical variations.

GloVe Embeddings: We used pre-trained GloVe vectors to represent tweets as averaged word embeddings. GloVe complements Word2Vec by leveraging global co-occurrence statistics, offering a different perspective on word relationships that might be missed by local context methods.

BERT Embeddings: We extracted contextualized sentence embeddings using a pre-trained BERT model (bert-base-uncased). Unlike static embeddings, BERT representations capture word meaning in context, making them well-suited for tweets where sentiment can depend on subtle phrasing or nearby words.

3.2 Baseline Models

To establish strong baselines for comparison, we implemented three traditional machine learning models: Logistic Regression, Support Vector Machine (SVM), and Random Forest, each trained across five different feature sets. Each model underwent grid search with 5-fold cross-validation for optimal hyperparameters. Results were evaluated using accuracy and weighted F1-score, providing a reliable foundation to compare with more complex models.

Logistic Regression: the model is optimized using liblinear as the solver with a regularization strength C=100 and max_iter=2000 to ensure convergence. It offers interpretability and performs well on linearly separable data, making it a robust baseline.

SVM: the model is trained with a linear kernel (kernel='linear', C=1), which is suitable for high-dimensional sparse data like BoW and TF-IDF.

Random Forest: the model utilized 200 trees (n_estimators=200) with default depth and splitting criteria. It provides a non-linear alternative with built-in feature selection and robustness to overfitting.

3.3 Deep Learning Models

We explored two deep learning approaches for sentiment classification: a custom BiLSTM network and a fine-tuned BERT model. These models were trained on GPU-accelerated environments (Google Colab), allowing deeper architectures to learn complex patterns. The deep learning models were evaluated using the same metrics and splits, enabling a fair comparison with baseline models.

BiLSTM (Bidirectional LSTM): the model was built using PyTorch, with parameters tuned via Optuna. The optimized model, featurized with BERT embeddings and optimized via Optuna, consisted of two LSTM layers with a hidden dimension of 256, a dropout rate of 0.3, and a batch size of 64. It was trained using the Adam optimizer with a learning rate of 0.0003 for up to 10 epochs.

BERT Fine-tuning: The model leveraged the bert-base-uncased transformer with a classification head. Tweets were tokenized and passed through the model, with the [CLS] token representation used for classification. Fine-tuning allowed the model to adapt contextual embeddings to the sentiment task directly, offering superior performance on nuanced text.

4. Evaluation

4.1 Evaluation Metrics

To comprehensively assess model performance, we use the following metrics:

- **Accuracy**: Proportion of correct predictions over the total.
- **F1-Score** (macro): Harmonic mean of precision and recall across all classes, treating each class equally regardless of support.
- **Confusion Matrix**: Used to visualize misclassification patterns.
- **Classification Report**: A per-class breakdown of precision, recall, and F1-score.

All evaluations were done using `scikit-learn` functions such as `accuracy_score`, `f1_score`, `confusion_matrix`and `classification_report`.

4.2 Model Selection and Comparison

In Model Training part, we recorded the performance of each model using different feature engineering methods on both the training and validation sets in `model_comparison_results.csv` file (Fig 4.1). Here the top 3 performing models were selected to be evaluated (for each model with multiple feature engineering methods, only the best-performing one was selected). These were:

1. Fine-tuned Bert
2. Random Forest using BoW features
3. BiLSTM Using Bert features

4.3. Evaluation on Test Set

For every selected model, we evaluated it on the held-out test set. Here is the summarized result:

Model	Accuracy	F1 Score
Fine-tuned Bert	0.9602	0.9603
Random Forest using BoW features	0.9565	0.9565
BiLSTM Using Bert features	0.8443	0.8450

The classification report and confusion matrix for these 3 models are attached on the Appendix (Fig 4.2 - Fig 4.7).

4.4 Reproducibility

All evaluation scripts are written in Python and can be found in the submitted `04.1_Evaluation.ipynb` notebook, and results are saved to the `results/evaluation_results.pkl` file.

4.5 Interpretability

To gain insights into how different features contribute to sentiment predictions, we experimented interpretability analysis using SHAP (SHapley Additive exPlanations) values on all Logistic Regression models. Logistic Regression was selected for its simplicity and transparent decision boundaries, making it ideal for interpretability compared to complex models.

For models using dense embeddings, the resulting SHAP outputs were tied to pseudo-feature names (e.g., *GLOVE_DIM[1]*), lacking direct interpretability and thus offering limited insight. In contrast, the SHAP plots for BoW and TF-IDF models (Fig 4.8, 4.9) revealed some associations between input terms and model predictions. BoW emphasized high-frequency, sentiment-laden words, while TF-IDF highlighted rarer terms. This difference reflects their underlying mechanisms: BoW relies on raw term frequencies, while TF-IDF adjusts these frequencies by penalizing common terms and emphasizing those that are more informative across the corpus.

References

Kaggle. (2021, August 9). Twitter Sentiment Analysis Dataset. Kaggle. https://www.kaggle.com/datasets/jp797498e/twitter-entity-sentiment-analysis. Accessed 13 April 2025.

Jeffrey Pennington, Richard Socher, Christopher D. Manning. GloVe: Global Vectors for Word Representation. The Stanford Natural Language Processing Group. https://nlp.stanford.edu/projects/glove. Accessed 13 April 2025.

HuggingFace. google-bert/bert-face-uncased. https://huggingface.co/google-bert/bert-base-uncased. Accessed 13 April 2025.

Appendix

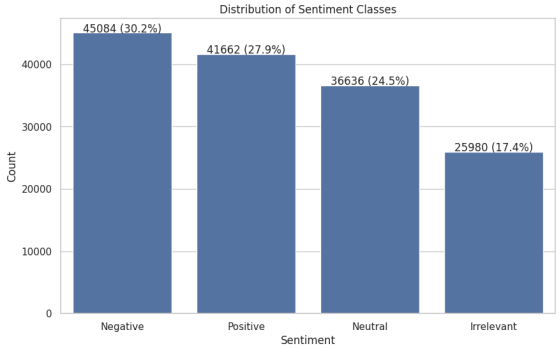

Fig 1.1 Distribution of Sentiment Classes

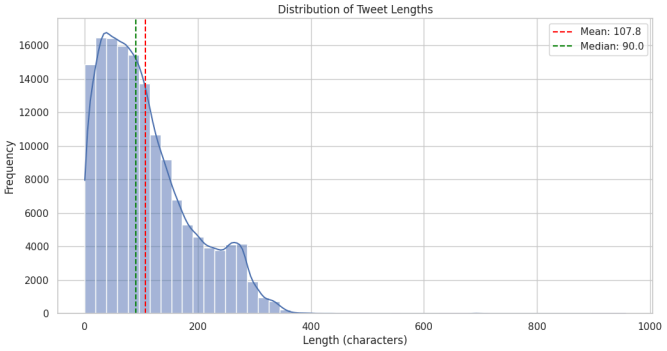

Fig 1.2 Distribution of Tweet Lengths

id	entity	sentiment	content
1	Borderland	Neutral	Rock-Hard La Varlope, RARE & POWERFUL, HANDSOME JACKPOT, Borderlands 3 (Xbox) dlvr.it/RMTrgF
2	Microsoft	Irrelevant	Very good of you to remind us. I thought it was GE but they sold!
3	CallOfDutyBlackopsColdWar	Negative	First round first nuclear... @ charlingINTEL. @ CfDutyDE
4	MaddenNFL	Positive	Well deserved .

Table 1: Twitter Entity Sentiment Analysis Dataset

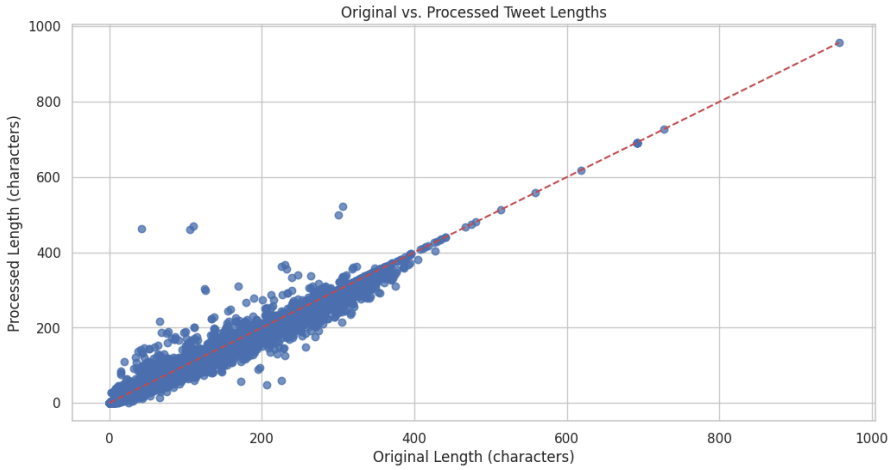

Fig 1.3 Original vs Processed Tweet Length

Model - Features	**Accuracy**	**F1-score**
Logistic Regression - BoW	0.76	0.76
Logistic Regression - TF-IDF	0.76	0.76
Logistic Regression - Word2Vec	0.41	0.41
Logistic Regression - GloVe	0.41	0.41
Logistic Regression - BERT Embeddings	0.58	0.84
SVM - BoW	0.78	0.78
SVM - TF-IDF	0.75	0.75
SVM - Word2Vec	0.36	0.36
SVM - GloVe	0.38	0.38
Random Forest - BoW	**0.95**	**0.95**
Random Forest - TF-IDF	**0.95**	**0.95**
Random Forest - Word2Vec	0.89	0.89
Random Forest - GloVe	0.87	0.87
Random Forest - BERT Embeddings	0.90	0.90
BiLSTM - BERT Embeddings	**0.90**	**0.90**
BERT Fine-tuned	**0.96**	**0.96**

Fig 4.1 Validation Accuracy

Classification	Precision	Recall	F1-score	Support
Irrelevant	0.98	0.94	0.96	3845
Negative	0.97	0.96	0.97	6684
Neutral	0.94	0.97	0.96	5406
Positive	0.95	0.96	0.96	6172
Accuracy			0.96	22107
Macro avg	0.96	0.96	0.96	22107
Weighted avg	0.96	0.96	0.96	22107

Fig 4.2 Fine-tuned Bert Classification Report

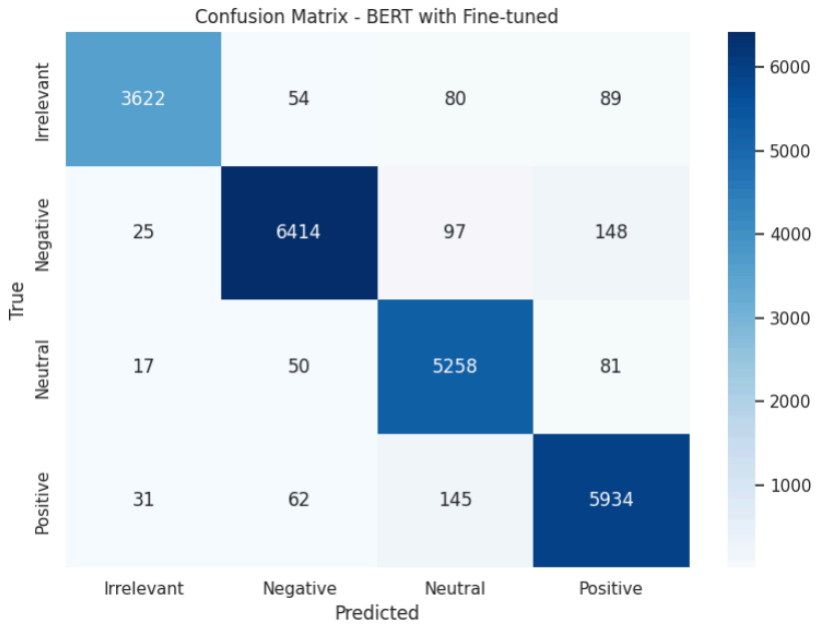

Fig 4.3 Fine-tuned Bert features Confusion Matrix

Classification	Precision	Recall	F1-score	Support
Irrelevant	0.97	0.94	0.96	3845
Negative	0.97	0.96	0.96	6684
Neutral	0.93	0.97	0.95	5406
Positive	0.96	0.95	0.96	6172
Accuracy			0.96	22107
Macro avg	0.96	0.96	0.96	22107
Weighted avg	0.96	0.96	0.96	22107

Fig 4.4 Random Forest using BoW Classification Report

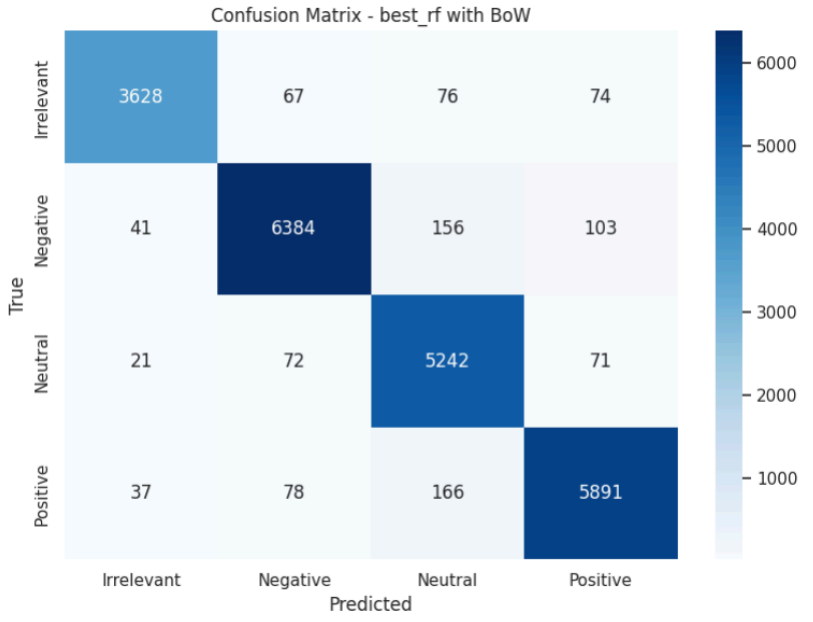

Fig 4.5 Random Forest using BoW Confusion Matrix

Classification	Precision	Recall	F1-score	Support
Irrelevant	0.77	0.88	0.82	3845
Negative	0.92	0.82	0.86	6684
Neutral	0.83	0.85	0.84	5406
Positive	0.84	0.85	0.84	6172
Accuracy			0.84	22107
Macro avg	0.84	0.85	0.84	22107
Weighted avg	0.85	0.84	0.84	22107

Fig 4.6 BiLSTM Using Bert features Classification Report

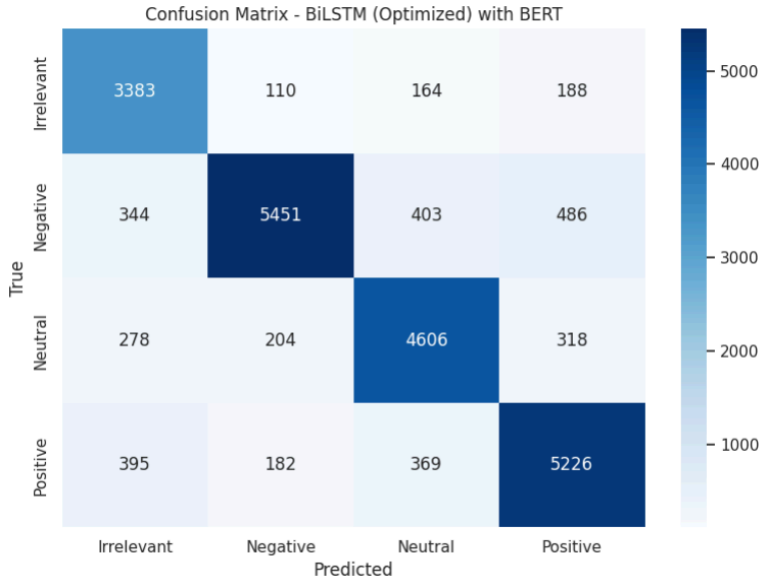

Fig 4.7 BiLSTM Using Bert features Confusion Matrix

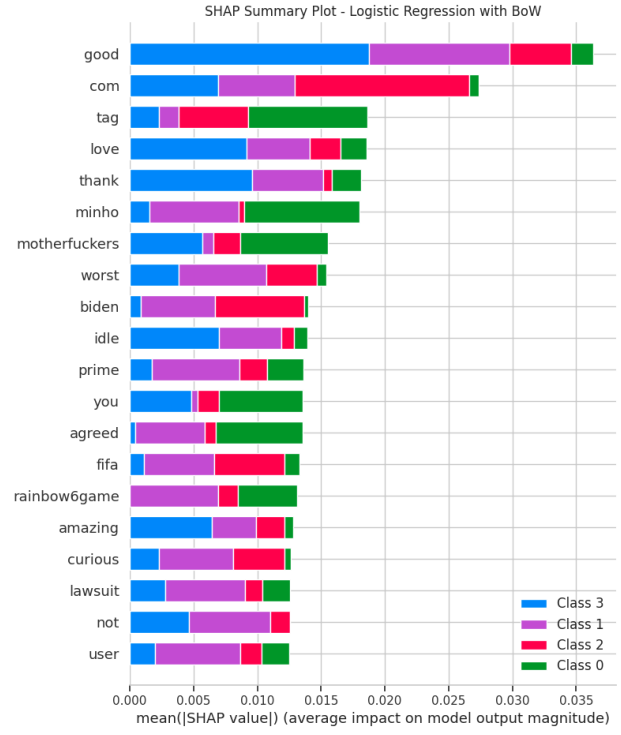

Fig 4.8 SHAP - Logistic Regression with BoW

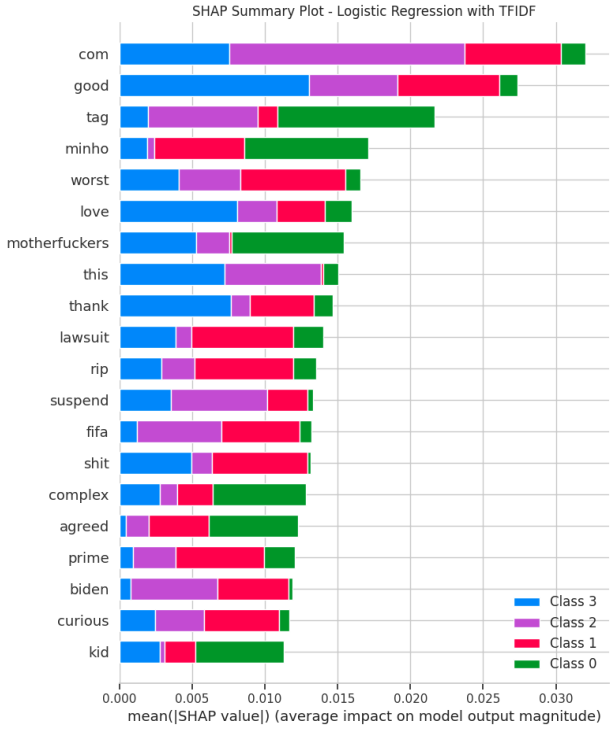

Fig 4.9 SHAP - Logistic Regression with TF-IDF