Tweet Sentiment Analysis

Ali Elkilesly, Selim Sherif, Roy Turk

Abstract

This study evaluates the performance of sentiment analysis models for classifying tweets as positive or negative, addressing the linguistic and structural challenges of Twitter data. We compare a baseline shallow machine learning model, the Transformer-based BERT, and BERTweet—a domain-specific model pre-trained on Twitter data. A preprocessing pipeline was implemented to standardize input data. Experimental results reveal that BERTweet significantly outperforms the baseline and BERT in classification accuracy, underscoring the importance of domain-specific adaptations for social media sentiment analysis.

1 Introduction

Sentiment analysis on Twitter presents unique challenges due to the platform's informal language, use of emojis, hashtags, and abbreviations. This report investigates approaches to classify tweets as positive or negative, starting with a baseline machine learning model and advancing to transformer-based models like BERT and BERTweet.

The report details key steps, including data preprocessing, model implementation, and hyper-parameter tuning, to optimize performance. It concludes with a comparison of model results, an ethical risk assessment, and reflections on the implications of these models for real-world sentiment analysis.

2 Dataset Description

This study employs two labeled datasets of tweets to evaluate sentiment analysis models. The primary dataset comprises 2.5 million tweets, equally distributed between positive and negative sentiments. From this dataset, a subset of 200,000 tweets, evenly divided into positive and negative samples, is selected for initial analysis. Additionally, a test dataset containing 10,000 unlabeled tweets is provided for evaluating the models. The datasets reflect the informal and concise communication style characteristic of Twitter, making them ideal for sentiment analysis tasks in social media contexts.

It is important to note that the data is already tokenized, ensuring that words are separated by spaces. Furthermore, user mentions and URLs were replaced with placeholders to ensure the dataset is ethical and conforms to privacy standards.

3 Baseline Model

3.1 Overview

As a baseline for evaluating the performance of more advanced Transformer-based architectures, a simple classification model is initially developed using traditional machine learning techniques, including **Support Vector Machines** (SVM) and **Logistic Regression**. This baseline provides a clear performance benchmark, facilitating a direct comparison of the computational complexity and training time associated with Transformer models against the incremental improvements they may offer in predictive accuracy.

3.2 Data Preprocessing

3.2.1 Data Cleaning

Prior to training, the raw tweets are subjected to a rigorous preprocessing pipeline to ensure data quality and consistency. Given the inherently noisy nature of social media text, these steps are critical:

1. **Deduplication and Removal of Empty Tweets:** Duplicate and empty entries are removed to ensure the model is trained exclusively on unique, informative samples. This step reduces potential biases and improves computational efficiency.

2. **Character-Level Cleaning:** To reduce noise and enhance the quality of the text data, the following steps are applied:

   - **Removal of Special Tokens:** All occurrences of `<user>` and `<url>` placeholders are removed, as they do not contribute meaningful sentiment information.
   - **Numerals and Ordinals:** All numeric values, including ordinals, are discarded since they generally do not convey sentiment information.
   - **Hashtags:** Hashtags are removed to avoid introducing irrelevant or highly context-dependent tokens.
   - **Retweet Indicators:** Tokens such as *rt* (retweet) are eliminated, as they do not provide additional sentiment context.
   - **Whitespace Normalization:** Multiple consecutive spaces are replaced with a single space to ensure standardized formatting.

3. **Contraction Expansion:** Common contractions (e.g., *don't* → *do not*) are expanded to their full forms to simplify morphological complexity and improve tokenization consistency. This is accomplished using the *contractions* library [1].

4. **Tokenization:** The cleaned text is tokenized into individual words, ensuring precise segmentation and facilitating subsequent modeling steps.

5. **Spell Checking:** A spell-checking procedure using SymSpell is applied to efficiently correct typographical errors, thereby enhancing data quality and model robustness [2].

6. **Lemmatization:** Lemmatization is performed using the NLTK library's WordNetLemmatizer [3] to reduce words to their canonical forms, thus mitigating morphological variability and further standardizing the textual data.

3.2.2 Building the Data Vocabulary

After preprocessing, the next step involves constructing a vocabulary dictionary. Two key considerations are addressed:

1. **Inclusion of Stopwords:** Empirical results indicate that excluding stop-words offers at best negligible improvements and occasionally a slight decrease in accuracy (approximately 0.01–0.02). This outcome is intuitive, given that certain common words (e.g., *not*) carry critical sentiment information [4].

2. **Vocabulary Frequency Threshold:** To mitigate noise introduced by rare words, various minimum frequency thresholds were tested. As shown in Figure 1, very low thresholds lead to high variability in accuracy due to the inclusion of rare, noisy tokens. Conversely, very high thresholds exclude too many words, resulting in a significant reduction in accuracy. A balanced threshold of approximately 300 occurrences was identified as optimal, effectively filtering out excessively rare terms while retaining sufficient lexical richness for robust model performance:

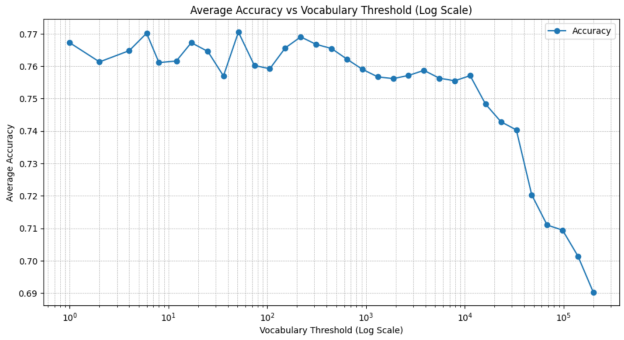

Figure 1: Average Accuracy vs. Vocabulary Threshold (log scale)

3.2.3 Vector Embeddings

Once the vocabulary is established, each token is represented as a numeric vector suitable for input into machine learning models. Pre-trained GloVe Twitter embeddings are chosen due to their relevance to this specific machine learning task and their availability in multiple dimensional configurations (25D, 50D, 100D, and 200D). For tokens not present in the GloVe dictionary, a random bounded vector is assigned to ensure comprehensive vocabulary coverage [5].

Preliminary experiments evaluate the impact of embedding dimensionality on model performance. The results indicate that higher-dimensional embeddings consistently enhance predictive accuracy without introducing excessive noise. Among the tested configurations, the highest-dimensional embeddings (200D) deliver the best accuracy, justifying their selection for subsequent modeling stages:

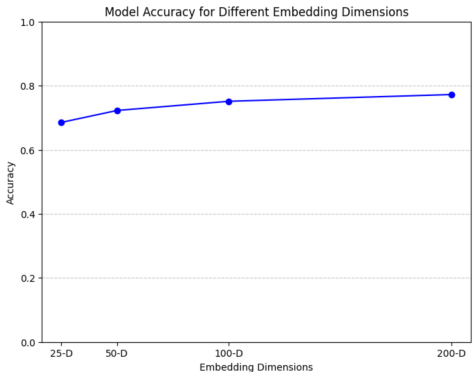

Figure 2: Average Accuracy vs. Token Embedding Size. The 200D embeddings provide the highest accuracy improvement, demonstrating the effectiveness of increased dimensionality.

3.3 Hyper-parameter Tuning

A comprehensive grid search is conducted on a subset of 200,000 tweets to optimize the hyper-parameters for both logistic regression and SVM models. The following hyper-parameter grid is evaluated:

- **Regularization Strength (`alpha`):** `[1e-4, 1e-3, 1e-2, 1e-1]` — penalizes large weights to mitigate overfitting.
- **Loss Function (`loss`):** `["log_loss", "hinge"]` — defines the objective to minimize during training.
- **Penalty (`penalty`):** `["l2", "l1", None]` — regulates model complexity to improve generalization.
- **Learning Rate (`eta0`):** `[0.0001, 0.001, 0.01, 0.1]` — controls the step size for weight updates during optimization.
- **Maximum Iterations (`max_iter`):** `[1000, 3000, 5000, 10000]` — specifies the upper limit on optimization iterations.
- **Convergence Tolerance (`tol`):** `[1e-4, 1e-3, 1e-2]` — sets the stopping threshold for optimization convergence.

The grid search yields the following optimal configuration for the model: `hinge` (loss), `l2` (penalty), `0.001` (`alpha`), `0.01` (`eta0`), `3000` (`max_iter`), and `1e-4` (`tol`).

The hyper-parameter tuning process reveals that the SVM model, using the `hinge` loss function, generally performs better than logistic regression. The optimal regularization strength (`alpha`) lies between `0.01` and `0.001`, with the `l2`-norm penalty proving most effective. Interestingly, variations in the learning rate (`eta0`) show little impact on the model's performance.

The hyper-parameter tuning and model training are performed using `scikit-learn` [6], which facilitates efficient experimentation and validation. These optimized parameters were then further used for the final test as they balance computational efficiency and predictive performance, achieving a final accuracy of approximately 0.77 during validation. This configuration is subsequently adopted for the final model training and testing phases.

4 Advanced model: BERT

4.1 Overview

After establishing a baseline model using SVM, we move to a more advanced approach with **BERT** (Bidirectional Encoder Representations from Transformers) [7]. BERT is a pre-trained model that understands the meaning of words in context by looking at both the words before and after them. This bidirectional approach allows BERT to capture complex relationships between words, making it highly effective for sentiment analysis tasks.

BERT's powerful performance stems from its robust architecture and training strategy. The BERT-base model consists of 12 transformer *encoder layers*, 768 *hidden dimensions*, and 110 million *parameters*. It was pre-trained on large, general-purpose datasets, including Google's BooksCorpus (800 million words) and English Wikipedia (2.5 billion words), using Masked Language Modeling (MLM) and Next Sentence Prediction (NSP) tasks. BERT uses a WordPiece tokenizer with a vocabulary size of **30,522 tokens**, which splits words into subwords to efficiently handle rare and out-of-vocabulary words. The model can process input sequences up to a maximum length of **512 tokens**, making it capable of understanding both short and relatively long texts. These properties enable BERT to capture rich contextual information, making it highly effective for tasks like sentiment analysis.

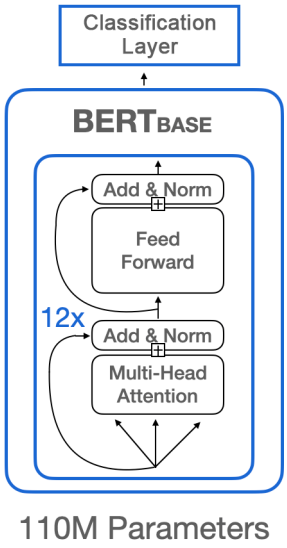

Figure 3: BERT model architecture [8]

4.2 Model implementation steps

4.2.1 Data preprocessing

For the advanced models, preprocessing focuses on the essential steps of duplicates removal and character cleaning, adapted from the baseline models. Duplicate entries are eliminated to ensure dataset integrity, while irregular characters are cleaned to standardize the text and facilitate better tokenization. The advanced models, such as BERT and BERTweet, do not require additional preprocessing steps like contraction expansion, spell checking, and lemmatization because of their pre-trained nature and robust tokenization mechanisms. These streamlined steps ensure the dataset is optimized for advanced modeling while maintaining efficiency.

4.2.2 Train-test split

The dataset is divided into training, validation, and test sets to ensure proper model training, fine-tuning, and evaluation. First, 70% of the data is allocated to the training set, which is used to teach the model patterns and relationships within the data. The remaining 30% of the data is initially set aside for further splitting.

To obtain a validation set and a final test set, the reserved 30% of the data is split again. One-third of this portion (10% of the total data) is used as the validation set, which helps tune hyper-parameters and monitor the model's performance during training to

avoid overfitting. The remaining two-thirds (20% of the total data) is retained as the test set, which provides an unbiased measure of the model's generalization performance on unseen data.

4.2.3 Tokenization of the data

The pre-trained tokenizer for the "bert-base-uncased" model is loaded using Hugging Face's Transformer library [9]. The tokenizer converts raw text into tokens and their corresponding input IDs, ensuring compatibility with the BERT model. This prepares the input data for efficient processing during training and inference.

The result of tokenization produces the following components, which are essential for ensuring that the input is correctly formatted for the BERT model:

- **Input IDs:** These are the numerical representations of the tokens generated from the input text. Each token is mapped to its corresponding ID in the model's vocabulary. Example: *Hello world* → `[101, 7592, 2088, 102]`, where `[101]` and `[102]` represent the special tokens `[CLS]` and `[SEP]`.
- **Token Type IDs:** Also known as *segment IDs*, these are used to distinguish between sentences when there are two inputs (e.g., in sentence-pair tasks).
  - Sentence 1 tokens → `0`
  - Sentence 2 tokens → `1`

  For single-sentence inputs, all values are `0`.
- **Attention Masks:** Attention masks indicate which tokens the model should attend to (value `1`) and which tokens should be ignored (value `0`), such as padding tokens. Example: `[101, 7592, 2088, 102, 0, 0]` → `[1, 1, 1, 1, 0, 0]`.

Together, these components ensure that the input text is properly structured for the BERT model.

4.2.4 Classification layer

To adapt the pre-trained BERT model for sentiment analysis, a classification layer is added on top of the model. This layer converts the `[CLS]` token's embedding into outputs for each sentiment class. The `[CLS]` token, representing the entire input sequence, is processed by the final transformer layer, and its output is passed through the classification layer to produce predictions. During fine-tuning, both the BERT layers and the classification layer are updated to learn task-specific features while retaining pre-trained knowledge.

5 Training and Hyper-parameter Tuning

To train the model, we configured key hyper-parameters from the small dataset of 200,000 tweets that influence training performance, including:

- **Learning Rate (`learning_rates`):** `[2e-5, 3e-5, 5e-5]` — controls the size of updates to the model's weights during optimization.
- **Batch Size (`batch_sizes`):** `[32, 64]` — determines the number of samples processed at once during training and evaluation.
- **Number of Epochs (`num_epochs`):** `[2, 3]` — defines the number of times the model sees the entire dataset during training.
- **Weight Decay (`weight_decays`):** `[0.01, 0.1]` — applies regularization to prevent overfitting by penalizing large weights.

Interestingly, modifying the learning rate, batch size, and number of epochs did not significantly influence the results, as the model consistently achieved similar performance across runs.

We observe that increasing the number of epochs beyond two often led to overfitting, with the validation loss being nearly twice the training loss in some cases, indicating poor generalization. A validation loss of `1.2` to `1.5` times the training loss is considered acceptable, indicating a reasonable balance between performance on the training and validation datasets. To mitigate overfitting, the training process is limited to two epochs.

The grid search yields the following optimal configuration for the Transformer-based model: `2e-5` (learning rate), `64` (batch size), `2` (epochs), and `0.01` (weight decay).

6 Advanced model: BERTweet

6.1 Overview

BERTweet [10] is a pre-trained language model specifically designed to process text from Twitter. Unlike the standard BERT model, BERTweet is trained on a massive dataset of 850 million English tweets. This enables BERTweet to better handle the informal and noisy nature of Twitter data, including slang, emojis, hashtags, and unconventional abbreviations.

In the context of sentiment analysis on tweets, BERTweet might be better suited than standard BERT because it has been exposed to the linguistic patterns and stylistic features unique to Twitter. This allows the model to capture the nuances of informal text more accurately, leading to improved performance on tasks like classifying sentiments in tweets.

6.2 Model implementation steps

The implementation of BERTweet for sentiment analysis closely mirrors that of standard BERT, with no changes to the training process or fine-tuning steps. However, there are two notable differences. First, in the preprocessing stage, the retweet token, hashtags, and emojis are retained to leverage BERTweet's ability to handle Twitter-specific content effectively. This ensures the model captures the unique linguistic patterns and sentiment cues present in tweets. Second, after running the same hyper-parameter grid the weight decay hyper-parameter is adjusted. For BERT, the weight decay is set to `0.01`, while for BERTweet, it is increased to `0.1`.

7 Final results and Conclusion

Model	Accuracy	F1 score
SVM	0.772	0.785
BERT	0.894	0.894
BERTweet	0.912	0.913

Table 1: Final results for all models*

*The best score obtained on AI Crowd differs slightly from the table for the BERTweet model since we did not impose a seed for the train-test split at first.

The Transformer models demonstrate a significant improvement over the baseline SVM model, underscoring their capacity to address the complexities inherent in sentiment analysis and text classification tasks. The BERTweet model, specifically pre-trained on Twitter data, outperforms the general-purpose BERT model, emphasizing the critical role of domain-specific pre-training in achieving superior performance. Nevertheless, fine-tuning these models necessitates substantial computational resources and time, posing challenges in terms of efficiency and scalability. This highlights the importance of critically evaluating the cost-benefit trade-off when considering the adoption of such models, particularly in applications where high accuracy and robust predictions must be weighed against resource constraints.

8 Ethical Risk Assessment

In the context of this sentiment analysis project, potential ethical risks are systematically evaluated using the *Digital Ethics Canvas* framework. This section outlines the identified risks, the measures taken to evaluate and mitigate them, and a reflection on the broader implications of ethical considerations in machine learning.

8.1 Identified Ethical Risks

Privacy Concerns: Although the dataset consists of publicly available tweets, there is a potential to infer sensitive or private information about individuals from their content, raising significant privacy concerns, particularly when analyzing personal sentiments or opinions.

Impact on Stakeholders:

- **Primary Stakeholders:** Users whose tweets are included in the dataset may face unintended consequences if their sentiments are misinterpreted or if the model is used inappropriately, potentially resulting in reputation or emotional harm.

- **Secondary Stakeholders:** Organizations or entities utilizing the sentiment analysis model may inadvertently perpetuate biases present in the dataset, leading to unfair or discriminatory outcomes that impact end-users or broader societal groups.

8.2 Evaluation of the Risks

To evaluate and address these risks, the following steps are undertaken:

- The dataset is obtained from a legitimate, publicly accessible source, specifically from EPFL course materials, ensuring its legal and ethical use.
- Sentiment categories in the dataset are balanced (positive and negative) to minimize skewed training outcomes that could amplify biases in predictions.
- Dataset is reviewed to ensure the removal of sensitive or personally identifiable information, such as usernames, URLs, and mentions.

8.3 Mitigation Measures

To minimize potential risks, the following measures are implemented:

- Dataset biases are addressed through stratified sampling and regularization techniques, ensuring that no specific sentiment or language pattern is over-represented during training.
- Documentation of the model's ethical limitations and potential biases is provided in this report to guide stakeholders toward responsible usage.

While these mitigation measures reduce certain risks, some limitations remain. For instance, addressing deep-seated linguistic and demographic biases requires comprehensive dataset re-balancing and advanced context-aware techniques, which are beyond the scope of this project.

8.4 Reflection

This project highlights the critical role of ethical considerations in the development and deployment of machine learning models. By identifying and addressing potential risks, this work ensures the responsible use of the sentiment analysis model. However, the limitations of these efforts are acknowledged, and further research is encouraged to explore methods to eliminate bias, enhance fairness, and safeguard the privacy of individuals in sentiment analysis tasks. Responsible AI development remains a cornerstone of machine learning practices.

References

[1] P. van Kooten, *Contractions: Fixes contractions such as 'you're' to 'you are'*, Version 0.1.73, 2016–2022. [Online]. Available: `https://pypi.org/project/contractions/` (cit. on p. 1).

[2] W. Garbe, *Symspell: 1 million times faster spelling correction fuzzy search through symmetric delete spelling correction algorithm*, GitHub Repository, 2015–2023. [Online]. Available: `https://github.com/wolfgarbe/SymSpell` (cit. on p. 1).

[3] N. Project, *Wordnetlemmatizer*, 2023. [Online]. Available: `https://www.nltk.org/api/nltk.stem.WordNetLemmatizer.html` (cit. on p. 1).

[4] PythonSpot, *Nltk stop words*, Accessed: 2023-12-18, 2023. [Online]. Available: `https://pythonspot.com/nltk-stop-words/` (cit. on p. 1).

[5] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Association for Computational Linguistics, 2014, pp. 1532–1543. [Online]. Available: `https://nlp.stanford.edu/projects/glove/` (cit. on p. 2).

[6] S.-L. Group, *Scikit-learn: Machine learning in python*, 2011. [Online]. Available: `https://scikit-learn.org/stable/` (cit. on p. 2).

[7] K. L. Jacob Devlin Ming-Wei Chang and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv*, 2018. DOI: `10.48550/arXiv.1810.04805` (cit. on p. 2).

[8] H. Face, *Bert 101 state of the art nlp model explained*. [Online]. Available: `https://huggingface.co/blog/bert-101` (cit. on p. 2).

[9] H. Face, *Transformers*. [Online]. Available: `https://huggingface.co/docs/transformers/en/index` (cit. on p. 3).

[10] T. V. Dat Quoc Nguyen and A. T. Nguyena, "Bertweet: A pre-trained language model for english tweets," *arXiv*, 2020. DOI: `10.48550/arXiv.2005.10200` (cit. on p. 3).