Sentiment Classification on Twitter: A Comparative Study of Classical and Transformer-Based Models

Kinane Habra, Sofiya Malamud, Jack Pulcrano
CS-433, EPFL

***Abstract*—Sentiment analysis of tweets poses challenges due to the informal nature of social media text, including slang, sarcasm, and emojis. Robust models are essential for reliable sentiment classification in this context. This project addresses these challenges using a pipeline that integrates traditional machine learning models with TF-IDF features and a RoBERTa-based model combined with a lightweight Multi-Layer Perceptron (MLP). The best-performing model was a linear SVM classifier, which achieved an F1 score of 0.862 and an accuracy of 0.86, highlighting its capability to effectively capture subtle distinctions in sentiment.**

I. Introduction

Social media platforms, such as X (formerly known as Twitter), have become essential avenues for individuals to express their opinions and emotions on a wide range of topics. The vast amounts of textual data generated by these platforms present significant opportunities for sentiment analysis, a field focused on identifying the emotional tone embedded in text.

In this project, we address the task of sentiment classification for tweets, aiming to determine whether the sentiment of a given tweet is positive or negative based solely on its textual content. To achieve this, we develop a machine learning pipeline that integrates preprocessing techniques and model optimization strategies to handle the unique challenges posed by social media text.

II. Data Preprocessing

The dataset consists of labeled positive and negative tweets, with 1.25 million examples in each class. Preprocessing is a crucial step in preparing this data for natural language processing tasks, as it ensures that the text is cleaned, normalized, and formatted appropriately for analysis. Without effective preprocessing, the noisy and unstructured nature of social media text–such as the presence of slang, emoticons, hashtags, and URLs—may negatively impact the performance of NLP models.

In this project, we apply a series of preprocessing steps, including lowercasing, removing punctuation, normalizing repeated characters, lemmatizing words, and replacing special tokens (e.g., URLs and user mentions). These steps not only reduce noise but also standardize the textual data, making it more suitable for feature extraction and model training.

A. Text Normalization

Text normalization was the initial step in preparing raw tweets for analysis. To standardize the text, it was first converted to lowercase. Then, repeated characters, commonly found in informal social media text (e.g., ”gooood”), were normalized to a maximum of two repetitions (e.g., ”good”).

Furthermore, numerical values were replaced with the token `number`, and exclamation and question marks were substituted with descriptive tokens (`exclamation` and `question`, respectively). These transformations standardized symbolic expressions, allowing the extracted features to better capture the semantic content of the tweets.

Next, informal textual elements such as emoticons, emojis, and slang were addressed. Emoticons and emojis were replaced with descriptive textual representations to retain their conveyed sentiment (e.g., ”:)” was replaced with `happy`, and ”haha” with `laugh`).

Slang terms and contractions were then expanded into their full forms using a predefined dictionary (e.g., ”2mrow” became `tomorrow`, and ”can’t” was expanded to `cannot`).

Lastly, punctuation was removed to simplify the text further.

B. Tokenization and Lemmatization

Tokenization and lemmatization were essential steps in preparing the text for analysis. Tokenization divided each tweet into individual words or tokens, allowing word-level processing as a prerequisite for lemmatization.

Lemmatization then reduced each word to its base or dictionary form (e.g., ”running” was reduced to ”run”), ensuring that different forms of the same word were treated as a single entity.

Finally, the tokens were rejoined to reconstruct a cleaned version of the tweet, ready for further processing and analysis.

III. Models and Architectures Used

A. TF-IDF Features with Classification Algorithms

Term Frequency-Inverse Document Frequency (TF-IDF) is used in NLP for transforming raw text into numerical feature representations. It quantifies the importance of words in a document relative to a collection of documents, allowing models to focus on informative terms while downweighting common, less meaningful ones. In this project, TF-IDF was utilized to convert tweets into numerical vectors, enabling the application of classifiers on them.

The resulting TF-IDF features were used as input for three classification algorithms: Naive Bayes, Logistic Regression, and Support Vector Machines (SVM). Note

that Naive Bayes serves as a simple benchmark model, often humorously referred to as the "punching bag of classifiers" by Lewis (1998) [1], while Logistic Regression and SVM were selected as the primary competing classifiers. Hyperparameter tuning, performed through grid search, further optimized the performance of these models by selecting the best parameter configurations. Cross-validation with 5 folds was employed to evaluate the models.

Component	Hyperparameters	Values
Tf-IDF	`ngram_range`	{(1, 1), (1, 2), (1, 3)}
	`min_df`	{1, 5}
Classifiers	Naive Bayes	`alpha`: {0.1, 1, 10}
	Logistic Regression	`C`: {0.1, 1, 10}
	SVM	`C`: {0.1, 1, 10}

Table I
GRID SEARCH HYPERPARAMETERS.

Table I summarizes the hyperparameters optimized for both the TF-IDF vectorizer and the classifiers. For the TF-IDF vectorizer, the n-gram range (`ngram_range`) was adjusted to include unigrams, bigrams, and trigrams, while the minimum document frequency (`min_df`), which specifies the minimum number of documents a term must appear in to be included as a feature, was set to 1 or 5. For the classifiers, `alpha` was optimized for Naive Bayes to control smoothing, and `C` was tuned for Logistic Regression and SVM to adjust regularization strength. Table II presents the results of the grid search, highlighting the hyperparameter configurations that achieved the highest F1 score on the validation set. To ensure robustness, the grid search was performed using three different random seeds. The results indicate that the optimal hyperparameters remained consistent across all seeds for all classifiers and the vectorizer, with the exception of Naive Bayes' `alpha`, which produced two different values (0.1 and 1), both of which are reported in the table. Notably, this consistency was observed regardless of whether the full dataset (2.5 million tweets) or the reduced version (200,000 tweets) was used.

Classifier	n-grams	min_df	Opt. Hyperparam.
Naive Bayes	(1, 3)	1	`alpha: 0.1, 1`
Logistic Regression	(1, 3)	1	`C: 10`
SVM	(1, 3)	1	`C: 1`

Table II
OPTIMAL HYPERPARAMETERS FOR EACH CLASSIFIER BASED ON GRID SEARCH.

As an example to demonstrate the importance of performing a grid search in such a context, Figure 1 illustrates the significant impact of varying the parameter `C` on the F1 score, using data from the reduced dataset. In particular, the results obtained from the reduced dataset align exactly with those derived from the entire dataset, with values of 10 for Logistic Regression and 1 for SVM, respectively. The `C` parameter was evaluated using the values $[0.01, 0.1, 1, 10, 100]$ for both models.

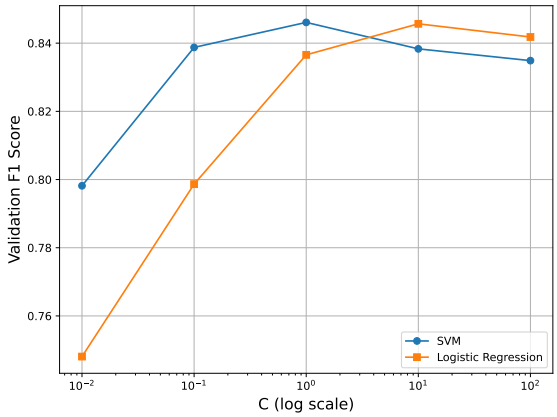

Figure 1. F1 Scores of SVM and Logistic Regression Models for Different C Values on the Validation Set.

B. Twitter RoBERTa, fine-tuned with an MLP

This approach combines a pre-trained transformer model (RoBERTa) [2], with a custom-built neural network for classification to perform binary text classification into negative and positive sentiment (the Twitter-specific RoBERTa model can be obtained from the Hugging Face Hub: https://huggingface.co/cardiffnlp/twitter-roberta-base-sentiment-latest.).

We decided to test this approach instead of simply using and fine-tuning a pre-trained model, in order to constitute a more innovative experiment. Furthermore, using a lightweight MLP instead of fine-tuning the entire transformer reduces the computational cost, which was a barrier we faced during the completion of this project.

RoBERTa is a pre-trained language model from the Hugging Face library. It generates contextual embeddings for input text, capturing semantic and syntactic relationships between words.

We prepare our data by initially tokenizing using RoBERTa's tokenizer to ensure compatibility with the model.

First, the input text data is shuffled and split into training and evaluation sets. We then tokenize each text using RoBERTa's tokenizer (e.g., `AutoTokenizer` from Hugging Face) with a maximum sequence length of 128 tokens. Any sequences exceeding this length are truncated. We assumed truncating the length wouldn't be an issue, as tweets are generally small text sequences.

The tokenizer outputs `input_ids` and `attention_mask`, ensuring that padded tokens are not attended to by the model. Additionally, the data are normalized from $-1/+1$ to $0/1$ to ensure compatibility with the binary classification framework.

A key feature of RoBERTa leveraged in this experiment is the `[CLS]` token, appended at the

beginning of each sentence and designed to capture the aggregate information from the entire sequence. This makes it suitable as a representation of the overall meaning of the sequence.

Leveraging the `[CLS]` token allows for a considerable increase in computational efficiency compared to having to extract all hidden layer features.

Another notable characteristic of the pre-trained model is the use of the attention mask, which helps it selectively pay attention only to tokens that carry meaningful information, ensuring that padding (or any non-relevant tokens) does not distort the contextual representations the model computes.

Thus, to construct our MLP, we only extract the `[CLS]` token from the last hidden layer of the pre-trained model, which we then feed into our MLP for further training.

Our custom MLP accepts vectors of dimension 768 and contains a hidden layer of dimension 128 to reduce complexity and introduce non-linearities via a ReLU activation.

Another crucial aspect of this experiment is that, during training, the transformer model operates in evaluation mode, meaning its parameters are frozen and not updated; only the MLP is trained.

Training relied on backpropagation to compute gradients of the loss function with respect to weights and biases, allowing the model to update parameters and minimize prediction errors [3].

We used the Adam optimizer (in PyTorch) with a learning rate of 1×10^{-4}, promoting faster and more efficient convergence.

Due to computational constraints, we were unable to perform hyper parameter tuning on the learning rate.

The binary cross-entropy loss function (`BCEWithLogitsLoss` in PyTorch) was employed, which is standard practice for binary classification tasks. Data are processed in batches of 800 during 4 epochs due to computational constraints.

Predictions were made using the Sigmoid activation function $\sigma(x) = \frac{1}{1+e^{-x}}$ to generate probabilities between 0 and 1. Probabilities below 0.5 were mapped to 0, and those above 0.5 to 1, which were then respectively mapped back to -1 and 1, effectively handling the binary classification task. The model architecture is illustrated in Figure 2.

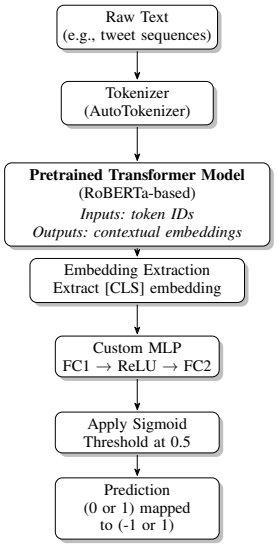

Figure 2. Model Architecture

IV. Results

A. TF-IDF Features with Classification Algorithms

To ensure robust evaluation and account for the impact of randomness, each classification algorithm was tested using three different random seeds. For each seed, the dataset was split into training and validation sets, and the models were trained and evaluated on the test set. Performance metrics, including accuracy and F1-score, were calculated for each seed, and the final results were averaged to provide a reliable estimate of each model's effectiveness.

Table III compares the performance of the three classifiers—Naive Bayes (NB), Logistic Regression (Log Reg), and Support Vector Machines (SVM)—on the test set. The table presents the average accuracy and F1-score calculated across three random seeds, comparing results from training on a reduced dataset (200,000 tweets) and the full dataset (2.5 million tweets).

Classifier	**Reduced Dataset**		**Full Dataset**	
	Accuracy	**F1-Score**	**Accuracy**	**F1-Score**
NB	0.797	0.811	0.824	0.837
Log Reg	0.830	0.833	0.857	0.858
SVM	0.832	0.835	0.859	0.860

Table III
PERFORMANCE METRICS FOR CLASSIFICATION ALGORITHMS USING TF-IDF FEATURES ON THE REDUCED AND FULL DATASETS (AVERAGED OVER THREE SEEDS).

These evaluations suggest that Naive Bayes consistently exhibited the lowest performance among the three classifiers. Logistic Regression and Support Vector Machines (SVM) showed relatively close performance, with SVM slightly outperforming Logistic Regression in terms of accuracy and F1-score. A notable improvement in performance was observed when using the full dataset compared to the reduced dataset; however, this came at the cost of significantly higher computational resources. Additionally, SVM required significantly more computational resources than Logistic Regression, as measured by runtime, despite providing only a marginal improvement in performance. Naive Bayes, on the other hand, had the lowest computational cost based on runtime, making it the most efficient algorithm, even though it consistently delivered the poorest performance.

B. Twitter RoBERTa fine-tuned with an MLP

The first point we would like to highlight is the difference between simply using the RoBERTa model to

perform sentiment classification, compared to feeding the outputs of the pre-trained RoBERTa model into an MLP, and then performing the classification.

The primary issue we aim to address is the limited expressiveness of a pre-trained language model's final representations when directly applied to sentiment classification in the Twitter domain. We consider the RoBERTa model as a robust baseline, but we start from the hypothesis that adding an MLP would allow for further optimization of its final layer, thus improving its classification capability.

By placing a lightweight Multi-Layer Perceptron on top of the RoBERTa embeddings, we allow the network to transform and refine these representations into a more sentiment-aware feature space.

In our experiments, running solely the RoBERTa model on the concatenated train sets results in an accuracy of 0.76 and an F1 Score of 0.754. Note that as RoBERTa is a pre-trained model, we tested it's performance of the fully concatenated data sets, with no separation into train and validation sets. On the other hand, training the model on the combination of RoBERTa/MLP, using the concatenated positive and negative datasets (with 80% of the dataset used for training and 20% for validation), we achieve an accuracy of 0.8672 and a F1 Score of 0.8667, on the validation set.

Model	Accuracy	F1 Score
RoBERTa Only	0.7600	0.7540
RoBERTa + MLP	0.8672	0.8667

Table IV
PERFORMANCE COMPARISON BETWEEN ROBERTA ALONE AND ROBERTA COMBINED WITH AN MLP CLASSIFIER.

This improvement of approximately 10 percentage points in accuracy underscores the added value of the MLP. By transforming the pre-trained embeddings into sentiment-focused representations, we enhance the model's ability to accurately classify tweets, thereby improving overall model performance and more effectively addressing the initial problem of capturing subtle sentiment distinctions.

Despite this, running Roberta/MLP achieved an accuracy of solely 0.788 on the test set, suggesting that our model architecture needs further hyper-parameter tuning and training, which was impossible to perform due to computational constraints.

V. DISCUSSION

A. TF-IDF Features with Classification Algorithms

One of the key limitations of this study was the scope of the grid search for hyperparameter optimization. While crucial for achieving the best model performance, performing an extensive grid search on the full dataset was computationally prohibitive due to the significant time and resource demands. For instance, running the full model across three seeds took us approximately 8 hours, making a more comprehensive search infeasible within the available resources. Consequently, the hyperparameter search space was constrained, leaving potential room for further performance improvements.

Another limitation was the selection of classifiers. Although the study focused on three models, there is an opportunity to explore a broader range of classifiers in future work. Methods such as ensemble models or neural networks could offer additional insights and potentially better results. However, time and computational constraints restricted this study to a smaller set of models.

A practical improvement for future research would involve conducting a more extended grid search on the reduced dataset, where computational requirements are more manageable. The hyperparameters identified through this process could then be applied to train and evaluate the models on the full dataset. This approach would strike a more effective balance between computational efficiency and model optimization, allowing for a broader exploration of hyperparameter values.

B. Twitter RoBERTa with an MLP on Top: Negative Versus Positive Sentiments

An interesting result we observed when training our model with the pre-trained RoBERTa model, is that our data preprocessing (tweet 'cleaning') improved model accuracy over the negative data set but resulted in lower accuracy for the positive data set [4]. Due to this trade off, we decided to exclude tweet cleaning from the RoBERTa/MLP experiment.

We stipulate this is due to pre-trained models like RoBERTa being trained on large, diverse datasets that often include noisy and unstructured data, such as misspellings, abbreviations, and informal language. This makes them inherently robust to raw input text. Cleaning tweets (e.g., removing special characters, URLs, or emojis) may inadvertently strip away meaningful contextual signals that the model could otherwise leverage.

We also note that the model has a significantly lower accuracy for negative tweet classification compared to positive tweet classification, which may stem from the fact that the negativity is formulated as sarcasm [5], which is extremely complex to teach to a machine.

Furthermore, if computational and time constraints weren't an issue, this model architecture could probably significantly benefit from hyper-parameter tuning and more extensive training, as well as additional computational layers, instead of the one simple hidden layer used in our architecture.

VI. SUMMARY

Training our data on a dataset of 2.5 million "cleaned" tweets using a linear SVM classifier with TF-IDF features, led to the best possible results, yielding an F1 score of 0.862 and an accuracy of 0.86. While these results are promising, further improvements could likely be achieved with more advanced algorithms, better hyperparameter tuning, or greater computational resources for experimentation.

Ethical risks

One key ethical risk identified in our project is the propagation of bias and unfairness when using machine learning models for sentiment analysis on social media data. The stakeholders most affected by this risk include minority communities and marginalized groups, who may already face discrimination in various facets of society. If the training data—comprising user-generated posts—reflects underlying societal prejudices, the sentiment analysis model could perpetuate these biases [5]. For instance, tweets associated with specific racial, gender, or cultural identities could be repeatedly misclassified as more negative, unfairly tarnishing the sentiments attributed to those groups. This leads to skewed outcomes and the reinforcement of harmful stereotypes.

The severity of this risk is considerable, given the pervasive nature of social media and the diverse groups represented online. Historical research and recent studies on language models, such as RoBERTa, show that bias is not a rare anomaly but a recurring issue. These models often inherit problematic patterns from their pretraining data [4]. Although quantifying the exact likelihood can be challenging, the presence of bias across numerous studies suggests a non-negligible probability of occurrence.

Evaluating this risk is particularly challenging because the RoBERTa model used in this study is pretrained on a large, fixed corpus, which we cannot modify or retrain due to computational constraints. This limitation makes it difficult to address biases inherent in the original training data directly. To assess and mitigate potential issues, it is essential to analyze how the pretrained model interacts with our specific dataset, identifying terms and phrases that may exhibit biased sentiment predictions for certain demographics. Utilizing fairness metrics, such as measuring disparate impact or evaluating sentiment scores across demographic subgroups, could provide a more concrete understanding of the model's performance in this context [6].

To address this, adjustments can be made at the dataset level, ensuring balanced representation across various identity groups and applying debiasing techniques during preprocessing or post-hoc model evaluation. Additionally, implementing fairness-oriented metrics and regular audits of the model's outputs can help identify and document biases transparently. While computational constraints prevent extensive retraining of the RoBERTa model, these steps aim to promote a more equitable and responsible application of sentiment analysis.

References

[1] D. D. Lewis, "Naive (bayes) at forty: The independence assumption in information retrieval," *Machine Learning: ECML-98*, pp. 4–15, 1998. [Online]. Available: https://doi.org/10.1007/BFb0026666

[2] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized BERT pretraining approach," *arXiv preprint arXiv:1907.11692*, 2019. [Online]. Available: https://arxiv.org/abs/1907.11692

[3] A. Saleh, M. Sheaves, D. Jerry, and M. Rahimi Azghadi, "Applications of deep learning in fish habitat monitoring: A tutorial and survey," *Expert Systems with Applications*, vol. 238, p. 121841, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417423023436

[4] H. Cai, Y. Wang, S. Li, and J. Zhang, "The impact of data cleaning on machine learning for natural language processing," *IEEE Access*, vol. 8, pp. 137 590–137 600, 2020.

[5] M. Khodak and D. Roth, "Implicit and explicit sarcasm detection: A new dataset and baseline," *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2371–2382, 2018.

[6] E. M. Bender and B. Friedman, "Data statements for nlp: Toward mitigating system bias and enabling better science," *Transactions of the Association for Computational Linguistics*, vol. 6, pp. 587–604, 2018.