# From Traditional Machine Learning to Transformers: Sentiment Analysis on Social Media Data

**Patel Charmi D.[1], Mevada Jayesh M.[2], Patel Gaurang J.[3], Patel Amit A.[4]**

M. TechScholar, Computer Engineering,Sankalchand Patel College of Engineering,Visnagar,India[1]
Asst.Prof.,Computer Engineering, Sankalchand Patel College of Engineering, Visnagar, India[2]
Asst.Pro.,IT Engineering, SwamiSachchidan and Polytechnic College,Visnagar,India[3]
Asst.Pro.,IT Engineering, Swami Sachchidanand Polytechnic College,Visnagar,India[4]

Charmipatel1175@gmail.com[1], jmmevadace_sspce@spu.ac.in[2], gjpatel.sspc@spu.ac.in[3], amitpatel7245@gmail.com[4]

**Abstract**: It is notoriously hard to analyze sentiment in social media. It is written in a disorganized manner, using slang, and is extremely dependent on implicit context. Classical baselines, including TF-IDF with Linear Support Vector Machines (SVM), however, give a good starting point. Nevertheless, such approaches do not capture the underlying semantic meaning as they deal with words individually. This paper compares traditional machine learning models with RoBERTa, transformer architecture with the ability to capture the entire sentence context. Social media detain the real world are sloppy, with too fine emotion tags and grossly skewed classes. In order to solve these problems, we designed a semantics-aware label consolidation algorithm based on Sentence Transformers. We then balanced our training data by using SMOTE in the continuous embedding space. Experiments with a multi-platform dataset with diverse data show that RoBERTa is significantly better than all classical baselines in terms of accuracy, precision, recall, and F1-score. More to the point, the transformer is able to deal with sarcasm and negation, which bag-of-words models often fail to do. This paper points to the need of contemporary sentiment analysis to include context-sensitive embedding and specific preprocessing to be able to read online communication correctly

**Keywords**: Machine learning, RoBERTa, sentiment analysis, social media,TF-IDF.

## I. INTRODUCTION

Social media like Twitter, Instagram, and Facebook create a massive flow of opinion and emotion of the masses every day. The systematic determination of the sentiments of people in these posts is of invaluable importance, whether it is tracking brand reputation or assessing the policy of the people. However, the textual analysis of social media is infamously difficult. The grammar of these platforms is broken, slang is widespread, abbreviations are used, and sarcasm is constant. Therefore, it is much harder to process a post than a formal document, and the possibility of a universal solution seems, at best, unrealistic.

The primitive sentiment models were based on primitive lexicons, which simply gave words in a fixed repository a fixed positive or negative weight. Although these models were easily understandable, they completely failed to explain the fast development of internet language. The field later shifted to machine learning. Methods like Term Frequency-Inverse Document Frequency (TF-IDF) [9] with Support Vector Machines [9] or Passive Aggressive Classifiers were a significant improvement. However, these methods are still stained by a severe drawback: the bag-of-words assumption. Such models ignore the word order and semantic context by treating words in total isolation. Without a clear concept of inter-word relations, such models always find it difficult to identify sarcasm and negation, which are the most common phenomena on social media.

This has since been overcome by Transformer architectures [5] using multi-head self-attention. Transformers do not process words in a sequence, but instead process the entire sequence of tokens at once, encoding the relationship between tokens to build highly contextualized representations. RoBERTa [3], based on the original BERT [4] architecture, enhances performance by training on large corpora and refined goals, and is regularly state-of-the-art on a range of NLP tasks. However, there is still a gap in existing scholarship. Very little research has empirically tested these transformer models using messy and real-world data that cut across platforms, cover a wide geographic area, and use highly granular emotion labels. The need to fill this gap is the main reason why this study was conducted.

● **Semantic Label Consolidation**: In this model, we combine fine-grained emotional labels into three main sentiment categories in a systematic fashion based on Sentence Transformer [6] embedding that are based on cosine similarity.


http://doi.org/10.63766/spujstmr.26.000070

● **Embedding-Space Augmentation:** To resolve the extreme class imbalance, we apply SMOTE [2] to the continuous sentence embedding space, thus, creating a perfectly balanced data set of 500 instances per class.

● **Multi-Model Comparison:** We make a direct comparative analysis of four traditional TF-IDF classifiers (Logistic Regression, Support Vector Machine, Random Forest, and Passive-Aggressive) to a fine-tuned RoBERTa model.

● **Superior Performance:** We show that RoBERTa attains an accuracy of 95.33 per cent with a macro F1-score of 0.95, which is more than 13 percentage points higher than the best traditional baseline.

We make a direct analogy between the traditional algorithms and the modern transformers that use unstructured social media data. We use TF-IDF along with our classifiers, Linear Support Vector Machine and Passive-Aggressive baselines. Then, we continue the investigation with RoBERTa. The results clearly show that context-sensitive embedding significantly improved both intrinsic accuracy and overall reliability. The rest of this paper describes the methodology in detail. Section II is a literature review, Section III is a discussion of the dataset, Section IV is a description of the preprocessing pipeline and model training, Section V is the experimental results, and conclusion.

### 1.1 RelatedWork

Early sentiment analysis involved word lists where the words were assigned a positive or negative rating [7][8]. These were simple to comprehend yet not effective with the informal and dynamic language on social media. Subsequently, machine learning techniques that counted word frequencies and models such as Support Vector Machines [9] and other linear classifiers [11] were able to do better, although they still treated texts as a bag of words, disregarding the sequence of words and context.

The deep learning models such as convolutional neural networks (CNNs) for text classification [13] performed better than the previous methods. The greatest innovation was the Transformer architecture [5] and its pre-trained version BERT [4]. RoBERTa [3] was an improvement of BERT, as it was trained on more data and the settings were adjusted, making it the best at NLP tasks. In order to quantify the similarity of sentences, researchers developed Sentence-BERT [6], which generates fast sentence embeddings; this paper uses them to combine labels. Researchers employed SMOTE-based oversampling [2] to deal with the imbalanced number of various classes. Despite all these tools, few studies have examined the label consolidation, addition of new data in embedding space, and comparison of various models on diverse social media data. This paper fills that gap.

## II. DATASET

Our dataset was obtained directly on Kaggle, where we used the Social Media Sentiments Analysis Dataset published by KashishParmar [1]. Thislistincludes733 original, user-createdpoststhatweretakenoutofTwitter,Instagram,andFacebook.It provides a perfect empirical environment to study the specifics of short-form digital communication by covering three large social platforms. Besides the raw text and sentiment tags, the dataset is also filled with useful additional metadata.

Every post shows the date it was created, which platform it was made on, userIDwhich is anonymized, what hash tags were used, retweets, likes, and other stats from the post. The user's country is also collected along with the exact year, month, day, and hour of the post. The posts were collected between 2010 and 2023 which helps us understand digital communication over a long period of time with a diversesetofusersfromtheUnitedStates,Canada,UnitedKingdom,India,Australia, Germany, France, Brazil, and Japan. The data set value is increased by its faint sentiment analysis labeling system.

The authors opted for an emotion-specific taxonomy categorizing affective phenomena into emotions like Joy, Sadness, Fear, Anger, Disgust, Surprise, Love, Excitement, Gratitude, Nostalgia, Anxiety, Loneliness and Sadness and also Amusement, Admiration, Serenity, Determination, Hope, Contentment, and Empowerment. The level of detail provides a useful resource for the studies of affective computing because the Internet hosts complex and multi-layered emotions.

To remain consistent with existing standards in NLP we applied a single labeling system. Accordingly, hyper-specific emotion categories were collapsed into the broad Positive, Negative, and Neutral classification. After this process of harmonization, the textual data went through a strict preprocessing procedure. This step formed the final input for the learning algorithms that trained the model.

# III. METHODOLOGY

## 3.1 Basic Preprocessing

We started by cleaning up the data frame structure. First, we have dropped the unnecessary index columns. We also disregarded the Hashtags, Day, Hour fields, as they are not that useful in making the model predict sentiment. To prevent untidy duplicates as we explored the data, we removed any leading or trailing spaces of categorical variables such as Platform and Country. Lastly, we converted the Timestamp column to a valid date time format. That caused us to add a DayofWeek feature to keep track of time and map integer months to their actual names, which made our charts much easier to read.

Our preprocessing was done by means of heavy lifting within the text. We have developed a special cleaning routine which processes all posts in a rigid sequence of modifications. All text was made small to keep the tokens the same. After that, we stripped URLs, HTML tags, newlines and bracketed annotations with the help of regex. We also eliminated punctuations and any alpha numeric words containing digits to reduce noise. Lastly, we removed non-ASCII characters, which removed emojis and other special characters that always litter social-media posts. Once the raw text was clean, we tokenized it with the word tokenize function of NLTK [10]. We discarded English stopwords at once with the help of the corpus provided by NLTK [10]. This eliminates all the high-frequency filler words which contribute zero meaning. We then used the NLTK WordNetLemmatized [10] to process the rest of the tokens after eliminating the stop words. We have selected lemmatization in particular as opposed to simple stemming. Stemmers simply cut off the end of words without thinking and this usually results in non-dictionary words. A lemmatized does consider the morphological context in order to determine the actual root. For example, it maps "running" to "run" and "better" to "good" which is correct and expected. This extra step provided us with a clear and strong baseline. We saved the last tokens in a new column called CleanText. This column is the main input feature for all of our models.

## 3.2 Semantic Label Consolidation

Original datasets were detailed but disorganized. We needed to group them to train the models. Our main goal was to classify them into three categories: Positive, Negative, and Neutral. Rather than employing rigid, manual techniques, or employing low-level lexicons, a construction was geared toward semantical consciousness. The aim was to be realistic and linguistically purposeful, which is why the consolidation process was derived from dense vector models.

To do so, every distinct, finely granulated label, e.g. Euphoria, Despair, Amusement, Loneliness, etc., was subjected to a pre-trained Sentence Transformer[6]. This model represented every particular word by a high-dimensional embedding vector. By doing this, we successfully captured the deep semantic meaning of every emotion and plotted it directly into a continuous vector space. Formally, let the Sentence Transformer be defined as a function:

$$f: T \rightarrow R^{d} \quad (1)$$

where $\boldsymbol{T}$ denotes the input text (a label string) and $\boldsymbol{d}$ is the dimensionality of the embedding space. Each fine-grained label $\boldsymbol{l}$ is thus represented as a dense vector $\boldsymbol{e_i} = \mathbf{E}(\boldsymbol{l_i}) \in \mathbb{R}^{\boldsymbol{d}}$. Similarly, the three target class labels — **"positive"**, **"negative"**, and **"neutral"** — are encoded into their respective reference vectors $\boldsymbol{e^{+}}$, $\boldsymbol{e^{-}}$, and $\boldsymbol{e^{0}}$.

To determine the most semantically appropriate class for each label, the cosine similarity between the label embedding and each target class embedding is computed. Given two vectors **A** and **B**, cosine similarity is defined as:

$$(A,B) = (A.B) / (\|A\|.\|B\|) \quad (2)$$

where $\boldsymbol{A \cdot B}$ denotes the dot product of the two vectors, and $\|\boldsymbol{A}\|$ and $\|\boldsymbol{B}\|$ represent their respective L2 norms. Cosine similarity measures the angular proximity between two vectors in high-dimensional space, independent of their magnitude, and ranges from −1 (completely opposite) to +1 (perfectly aligned). It is widely used as a measure of semantic relatedness in NLP tasks. Each fine-grained label $\boldsymbol{l_i}$ is then assigned to the target class that maximizes the cosine similarity:

$$C^{*} = argma(C \epsilon \{positive, negative, neutral\}) cos(e_i, e_c) \quad (3)$$

For instance, the label **"Euphoria"** produces an embedding that is geometrically closest to **"positive"** in the semantic space, yielding the highest cosine similarity with $e^{+}$, and is thus mapped to the Positive class. Conversely, **"Despair"** aligns most closely with $e^{-}$, resulting in a Negative assignment, while **"Indifference"** gravitates toward $e^{0}$, mapping to Neutral.

This semantics-based, unsupervised method does not require manual annotation or hand-written mapping rules. Through the rich distributional semantics of pre-trained Sentence Transformer embedding [4][6], the label consolidation is based on the true emotional polarity of each fine-grained category, which guarantees a consistent and reproducible three-class label distribution across all 733 records of the dataset. It is distributed in Fig. 1. The data distribution across different platforms is shown in Fig. 2.

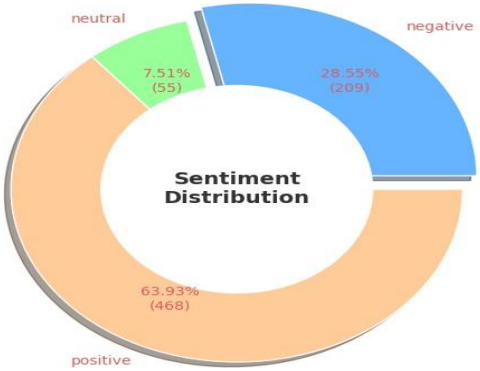


Fig.1.Data distribution after Sentiment Label Consolidation

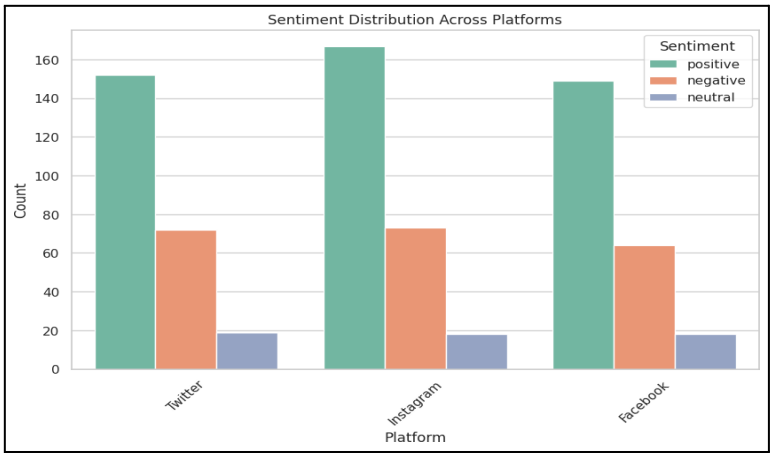


Fig.2.Sentiment distribution across the platforms

### 3.3 Data Augmentation

The resultant dataset is completely skewed with significant class imbalance, with the Positive class constituting 63.93% of samples (468), while Negative and Neutral accounted for only 28.55% (209) and 7.51% (55) respectively. Training on such skewed data risks biasing model predictions heavily toward the majority class., Thus, resulting in poor generalization on underrepresented categories. To address this concern, a text adapted variant of the Synthetic Minority Over-sampling Technique (SMOTE) [2] was employed. This operates in the continuous sentence embedding space rather than on raw text.

To achieve this, all post texts were first encoded into dense vector representations using the pre-trained Sentence Transformer all-MiniLM-L6-v2. The model maps each text into a high-dimensional semantic space where geometrically proximity between the vectors correspond to semantically similar content. A K-Nearest Neighbour (KNN) graph was constructed using cosine distance for each sentiment class specific embedding. SMOTE interpolation is applied to generate the synthetic embedding by applying the standard between each source sample and a randomly selected neighbour :

$$e_{synthetic} = e_i + \lambda * (e_j - e_i), \lambda \in [0,1] \quad (4)$$

where λ is sampled uniformly at random, placing the synthetic point on the line segment between two semantically related real samples within the same class.

A retrieval step must be added because the synthetic vector doesn't seem to be a real text. Rather than deterministically selecting the single closest text (argmax), probabilistic top-k sampling with temperature scaling was used by computing cosine similarities between the synthetic embedding and all existing texts.

Now, according to the softmax-weighted probability distribution, the top-k candidates are selected. This added a lexical diversity in our augmented data while keeping them semantically meaningful to their target class. This prevents the model from memorizing the repeated patterns, which prevents the model to generalize in real life scenarios. The augmentation was applied to all three classes until each reached a target count of 500 samples. This produced a fully balanced dataset of 1,500 records for subsequent model training.

### 3.4 Feature Extraction: TF-IDF

For our traditional machine learning baselines, we built a shared feature extraction pipeline using Term Frequency–Inverse Document Frequency (TF-IDF). First, we fed the clean text into a TfIdfVectorizer and capped the vocabulary at the top 5,000 most frequent features. We configured the n-gram setting to (1,3), which would allow the vectorizer to successfully collect unigrams, bigrams, and trigrams. This design choice enabled the capture of multi-word units so the models would learn phrases rather than have to learn patterns of individual words.

We made an 80/20 split of the dataset before training. In order to keep the evaluation fair and balanced for Positive, Negative, and Neutral sentiment classes, we performed a stratified split to ensure the same in the train and test data.

### 3.5 Baseline Models

We established our baselines for our TF-IDF features by training four traditional classifiers. The first one we chose was the Passive Aggressive Classifier (PAC). It only updates weights when there is a misclassification. This gave us a solid baseline of 75.33% accuracy and a 0.75 macro F1 score. Next, we selected Logistic Regression, which achieved 79.33% accuracy and a 0.79 macro F1 score, showing better performance. We also tried a Random Forest ensemble, but its performance was worse, at 76.33% accuracy and a 0.76 F1 score. This drop is expected because tree-based methods usually perform worse with sparse, high-dimensional text features. At last, the LinearSupport Vector Machine (SVM) was on par with Logistic Regression with 79.00% accuracy and 0.79 F1 score. Logistic Regression and SVM ended up being our best baselines and therefore provided a highly competitive ceiling for classical ML methods.

To further improve the performance of our baseline PAC, we created a dedicated pipeline which included SMOTE oversampling and systematic hyper parameter tuning. The pipeline was built on three sequential stages. Initially, it performs the TF-IDF vectorization [9]. Then, it uses SMOTE to balance the classes, but only on the training folds. The last step sends the balanced data to the PAC, employing a class-weighted loss function. As a result, data leakage to the test set was entirely sidestepped by the meticulous separation of the SMOTE augmentation within each cross-validation split.

The last step sends the balanced data to the PAC, employing a class-weighted loss function. As a result, data leakage to the test set was entirely sidestepped by the meticulous separation of the SMOTE augmentation within each cross-validation split. We conducted a grid search combined with cross-validation (GridSearchCV) for the sake of parameter tuning, and rather than simply optimizing for accuracy, we set our goals for the average (macro) F1-score. This engineering choice was significant because it incentivized the model to treat all three classes equally, thus mitigating the likelihood of class imbalance. We combed a search space of regularization strengths $C \in \{0.01, 0.1, 0.5, 1.0\}$, loss function choices (hinge vs. squared hinge), and fitting the intercept. The grid search located the optimal configuration at $C = 0.01$ with squared hinge loss and intercept fitting active. This configuration returned a macro F1 of 0.760 during cross-validation.

Our improved pipeline, using the withheld test data, achieved an impressive 82.00% accuracy in our final evaluation. Even more remarkable was the consistency, with precision, recall, and each class's F1-score all at 0.82. This improvement from our original baseline shows that proper target balancing and tuning can facilitate class-agnostic sentiment prediction.

### 3.6 Transformer-Based Model: RoBERTa

To move beyond the limits set by our usual baselines, we began fine-tuning RoBERTa (Robustly Optimized BERT Pre-training Approach) for three classification tasks. According to Liu et al. [3], RoBERTa is based on the original BERT architecture. The model is improved with methods like dynamic masking, large batch sizes, and longer training times. It examines the full bidirectional context for each token through multi-head self-attention. This method helps it capture the nuances of emotional expressions effectively and comprehend the context that static TF-IDF vectors often overlook.

We defined our three classes of sentiment as numerical values, using column names as labels. The values are 0 for Negative, 1 for Neutral, and 2 for Positive. We then established a stratified 80/20 train-test split with randomstate = 42. This ensured that the proportion of each class was preserved exactly the same in both the training and the testing instances of the dataset to increase the possibility that the model would generalize better in actual applications.

We accessed the RobertaTokenizer through the roberta-base checkpoint to implement tokenization. We opted to set the padding and truncation limit to 64 tokens for all sequences. Sixty-four tokens is optimal for example social media posts and balances computational efficiency with the likelihood that we will not cut off critical context.

Using scikit-learn's [11] computeclassweight with the 'balanced' option, we assigned higher weights to less frequent classes in the data to ensure that weights were balanced across classes. We then provided these weights to a custom WeightedTrainer class, which modifies the standard training loop to use a weighted version of Cross-Entropy loss:

$$\widehat{L} = -\sum_i w_i * y_i * log(y_i) \quad (5)$$

where $w_i$ is the class weight, $i$ is the true label, and $y_i$ is the predicted probability for class $i$. This loss function increases the penalty for misclassifying minority classes, which helps to achieve balanced generalization. Regardless of the fact that our dataset is quite balanced, this approach is still valid. The roberta-base model was initialized with a 3-label sequence classification head and fine-tuned with the following hyperparameters, as shown in Table 1.

TABLE I
MODEL PARAMETERS

| Model | Value |
|---|---|
| Epochs | 10 |
| Learning Rate | 1e-5 |
| TrainBatchSize | 16 |
| Eval Batch Size | 64 |
| WarmupSteps | 100 |
| WeightDecay | 0.01 |
| Evaluation Strategy | Per epoch |
| Best Model Selection | Maximum accuracy |

We set the learning rate to **$1e$-5** to allow the model to fine-tune on the three-class sentiment task without the risk of catastrophic forgetting. Warmup steps are used to gradually increase the learning rate at the beginning of training to help with optimization. For evaluation, we used the checkpoint with the best performance at the end of training. The model was trained with Hugging Face [12]

# IV. RESULT AND CONCLUSION

TABLE II
SUMMARY RESULTS OF ALL THE MODELS

| Model | Accuracy | MacroF1 |
|---|---|---|
| Passive Aggressive Classifier | 75.33% | 0.75 |
| Logistic Regression | 79.33% | 0.79 |
| Random Forest | 76.33% | 0.76 |
| SupportVectorMachine | 79.00% | 0.79 |
| OptimizedPAC(SMOTE+ Grid Search CV) | 82.00% | 0.82 |
| RoBERTa | 95.33% | 0.95 |

Experimental results across all models are summarized in Table 2. One thing was clearly evident: the transformer-based RoBERTa was far more powerful than classical machine learning algorithms for sentiment classification. For our raw TF-IDF baselines, Logistic Regression and the Support Vector Machine proved to be on the top spot. Both hit roughly 79% accuracy and a 0.79 macro F1-score. The Passive Aggressive Classifier and Random Forest did not perform as well as expected, scoring 75.33% and 76.33%. This outcome was not surprising. It matches findings in current text classification research [7][8]. Studies show that linear models typically outperform tree-based ensembles in sparse, high-dimensional feature spaces [9][11].

We then tested our improved PAC pipeline. By using SMOTE and running the GridSearchCV hyperparameter tuning, we increased the accuracy to 82.00% with a perfect 0.82 macro F1-score. As a result, despite being limited to a primitive bag-of-words model, significant substantive improvements can be achieved by carefully balancing the training data and carefully fine-tuning the model parameters to reduce the effects of class imbalance.

Other enhancements could not compete with the performance of the fine-tuned RoBERTa model. RoBERTa achieved 95.33% test accuracy and a 0.95 macro F1-score. This is a 13% increase over the best traditional baseline measure we achieved. This shows impressive consistency, and we saw F1-scores of 0.97 for Negative, 0.95 for Neutral, and 0.94 for Positive, which shows the model actually learned to generalize and did not just memorize the data to favour a particular class. This can be attributed to proper hyperparameters and data augmentation. The Negative class achieved a 0.98 recall and trumped the rest. Lexical methods struggled with negative comments as they contain high levels of sarcasm and indirect complaints, which RoBERTa was able to handle with ease. The RoBERTa confusion matrix in Fig. 3 shows a breakdown of these predictions. Fig. 4 compares the accuracy of the different models based on their performance.

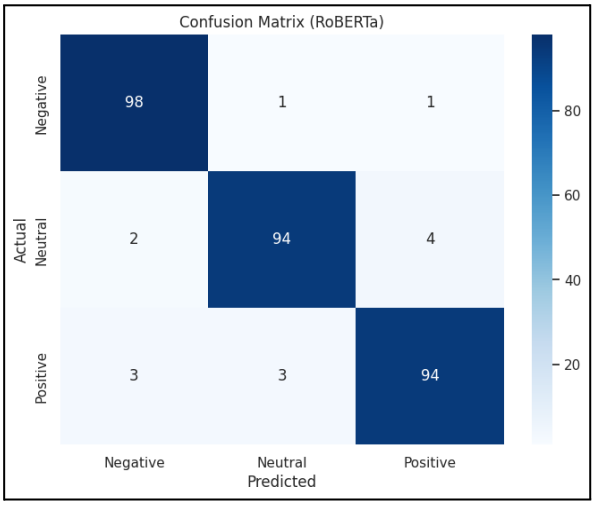


Fig.3. Confusion matrix for RoBERTa

The large difference in performance between the TF-IDF baselines and RoBERTa comes down to a key difference in their structure. RoBERTa not only captures context but also understands sarcasm, which traditional models often miss. TF-IDF considers each word in isolation. It ignores word order and surrounding context. However, social media does not operate that way. People frequently use irony, slang, and negation, so the actual sentiment relies on context. This is where transformers excel. RoBERTa employs self-attention, allowing each token to consider every other token in the sequence. It goes beyond a simple vocabulary check. This approach creates a detailed, context-aware representation of the entire post, capturing the true meaning behind the text.

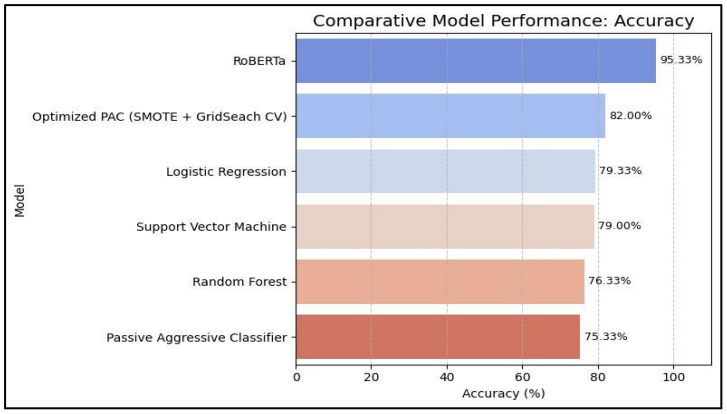


Fig.4. Comparative model performance based on accuracy

## REFERENCES

[1] K. Parmar, "Social Media Sentiments Analysis Dataset," *Kaggle, 2023.*

[2] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic Minority Over-sampling Technique," *Journal of Artificial Intelligence Research, vol. 16, pp. 321–357, 2002.*

[3] 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.*

[4] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," *in Proc. NAACL-HLT, Minneapolis, MN, USA, Jun. 2019, pp. 4171–4186.*

[5] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention Is All You Need," *in Proc. NeurIPS, Long Beach, CA, USA, Dec. 2017, pp. 5998–6008.*

[6] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks," *in Proc. EMNLP, Hong Kong, China, Nov. 2019, pp. 3982–3992.*

[7] B. Pang and L. Lee, "Opinion Mining and Sentiment Analysis," *Foundations and Trends in Information Retrieval, vol. 2, no. 1–2, pp. 1–135, 2008.*

[8] S. M. Mohammad, "Sentiment Analysis: Detecting Valence, Emotions, and Other Affectual States from Text," *in Emotion Measurement, L. Tinianov, Ed. Elsevier, 2016, pp. 201–237.*

[9] T. Joachims, "Text Categorization with Support Vector Machines: Learning with Many Relevant Features," *in Proc. ECML, Chemnitz, Germany, Apr. 1998, pp. 137–142.*

[10] S. Bird, E. Loper, and E. Klein, Natural Language Processing with Python. Sebastopol, CA, USA: O'Reilly Media, 2009.

[11] F. Pedregosa et al., "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research, vol. 12, pp. 2825–2830, 2011.*

[12] Y. Kim, "Convolutional Neural Networks for Sentence Classification," *in Proc. EMNLP, Oct. 2014, pp. 1746–1751.*

[13] T. Wolf et al., "HuggingFace's Transformers: State-of-the-Art Natural Language Processing," *in Proc. EMNLP: System Demonstrations, 2020, pp. 38–45.*