Title: Predaping: A Machine Learning-Based Approach to Safeguarding Teenagers on Social Media

Abstract

- A concise summary of the problem, proposed solution (the Predaping app), methodology, and key findings.

Keywords

- Online Safety, Social Media, Machine Learning, Natural Language Processing, Teen Safety, AI-Driven Alerts, Cybersecurity

1. Introduction

- **Background and Motivation**: Discuss the rising concerns about online safety for teenagers, the prevalence of online predators, and the limitations of existing solutions.
- **Problem Statement**: Clearly define the problem that Predaping addresses.
- **Objectives**: Outline the goals of the app—real-time detection, education, and empowerment.
- **Contributions**: Briefly mention the contributions of this paper, such as the development of a novel app, its algorithms, and the potential impact on online safety.

2. Related Work

- **Overview of Existing Solutions**: Compare with other tools and approaches in the domain of online safety.
- **Limitations of Current Solutions**: Highlight the gaps that Predaping aims to fill.

3. Methodology

- **Data Collection**: Describe the datasets used for training the machine learning models—e.g., social media text, user profiles, behavior patterns, etc.
- **Machine Learning Models**: Explain the models used (e.g., NLP models for sentiment analysis, classifiers for behavior detection).
- **Real-Time Alert System**: Detail the AI-driven alert mechanism and how it involves both teenagers and guardians.
- **Educational Platform**: Outline the content and structure of the educational resources provided within the app.
- **Customizable Settings**: Explain the personalization features for alert thresholds and privacy settings.

4. System Architecture

App Architecture Diagram

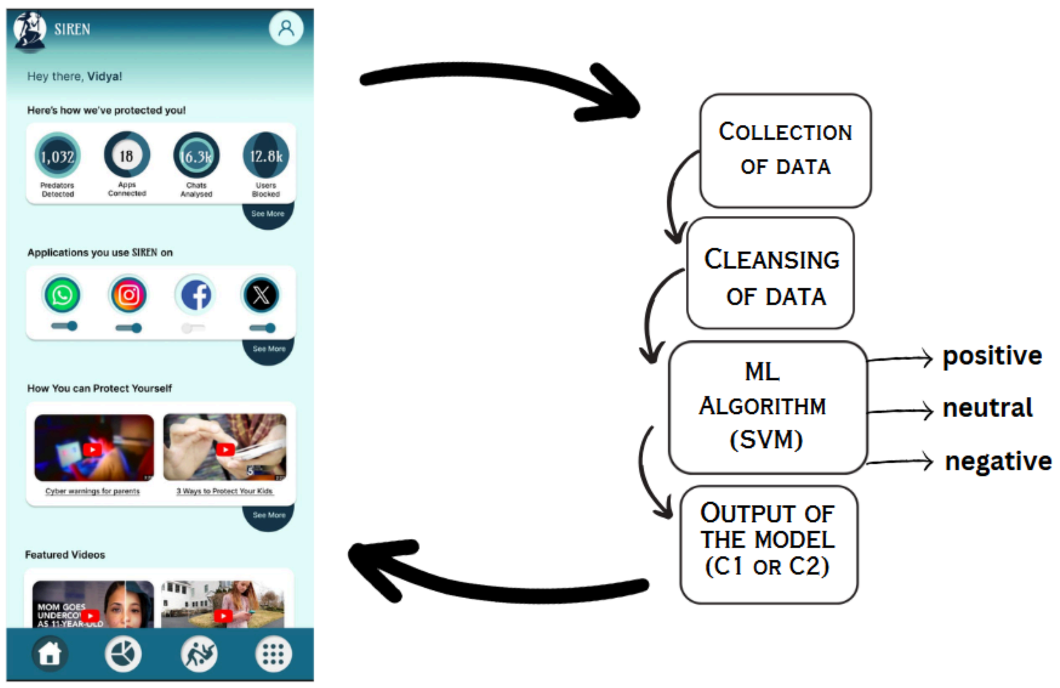


Figure 1: Overall Architecture Diagram Showing User Interaction, Data Flow, and Integration with Machine Learning Models.

The app's architecture, as depicted in Fig. 1, illustrates the flow of data through various stages of the system, from initial collection to the final output, interacting seamlessly with the user interface and the machine learning models.

1. **Collection of Data**

The data collection phase involves obtaining and integrating two datasets to create a comprehensive source for sentiment analysis. The first dataset, "RVU-BTech22 Students," consists of real-world WhatsApp group chats from a college environment. This dataset reflects authentic conversational patterns and diverse language use typical of a college setting. The second dataset, "Stalker," is synthetically generated by ChatGPT and serves to supplement the real-world data with additional examples. This synthetic data helps in enhancing the model's robustness by providing a broader range of text scenarios and variations. Both datasets are read into pandas DataFrames, which facilitate efficient data manipulation and processing. By combining these two sources, the dataset becomes more representative of various conversational contexts and language styles. This enriched dataset is critical for developing a machine learning model that performs well across different communication scenarios. It ensures that the model is trained on a diverse range of text inputs, which enhances its ability to generalize and accurately classify sentiments in new, unseen data.

2. **Cleansing of Data**

This includes converting all text to lowercase, which helps in standardizing the data and avoiding mismatches due to case differences. Punctuation is removed to focus on the core textual content, and extra spaces are eliminated to maintain a clean format. The cleaned data is then structured into a pandas DataFrame, making it ready for further processing. This meticulous cleansing ensures that the dataset is focused, relevant, and free from extraneous elements, which is essential for accurate and reliable sentiment analysis.

3. **Machine Learning Algorithm (SVM)**

Once the data is preprocessed, it is converted into numerical features using the TF-IDF (Term Frequency-Inverse Document Frequency) method, which is essential for feeding text data into the Support Vector Machine (SVM) model. This process captures the significance of terms, which is crucial for effective text classification. The SVM algorithm is then employed, configured with a linear kernel to handle high-dimensional data efficiently. The SVM model is trained on these TF-IDF features to classify text into categories such as positive, neutral, or negative. During training, the model learns to identify patterns and relationships within the data, allowing it to make accurate predictions on new, unseen text. The linear kernel is chosen for its simplicity and effectiveness in text classification tasks, facilitating the model's ability to handle complex data with high accuracy.

4. **Output of the Model**

The output of the SVM model is evaluated to measure its performance in classifying sentiments. The model is tested on a separate testing set that was not used during training, ensuring a fair evaluation of its ability to generalize. The classification report, which includes these metrics, provides a detailed analysis of the model's effectiveness in distinguishing between positive, neutral, and negative sentiments. This comprehensive evaluation ensures that the model is reliable and capable of providing meaningful insights into the sentiment of messages, making it a valuable tool for applications requiring sentiment analysis.

5. **Experiments and Results**

Datasets: The dataset used in the experiment consists of daily corporate and non-corporate text messages between professors and SoCSE students. This dataset was built by first converting and storing the WhatsApp messages into a .txt file named "VU-BTech22 Students.txt". There is a file path that specifies the location of the text file to read. The file is opened in read mode and the f.readlines() function reads all the lines from the file into a list called 'data'. To confirm whether the file has been read correctly, '%(len(data))' prints the number of lines in the txt file. The first line of the text file is skipped by the usage of dataset=data [1:], as the first line can be assumed to be a header or metadata. A for loop is set through each line of the datasets, extracting useful parts of information. Each piece of information is set into four columns, and these parsed lines are appended to ‘cleaned_data’ as a list of information with four columns. The parsed list is later set into a pandas DataFrame with defined columns. This DataFrame is saved to a CSV file at a specified path; ‘index=False’ ensures that the DataFrame index is not written to the CSV file.

The dataset later used for training and testing includes labeled messages tagged with the sender’s name and the message the sender has sent or replied to. Specifically, the data is divided into:

- **Training Set:** 80% of the dataset used to train the model.
- **Testing Set:** 20% of the dataset used to evaluate the model’s performance.

The labels are binary, where ‘Predator1’ indicates predatory behavior and other labels non-predatory behavior. The dataset consists of four main columns labeled Name, Message, Date, and Time. Before the dataset is split, the dataset is cleansed and pre-processed to remove irrelevant information, such as messages indicating only emojis, alerts like “Waiting for this message”, “This message was deleted”, and any kind of media is omitted. Feature extraction is performed using techniques of CountVectorizer and TF-IDF to transform textual data into numerical representation suitable for ML models.

Evaluation Metrics: The evaluation metrics considered for the app are accuracy for the proportion of correct predictions (which includes true negatives and true positives) to the total number of predictions made by the SVM model, precision for assessing the model’s ability to correctly identify the predatory and non-predatory sentiments, and recall is used for proportions between all the true predictions. For instance, the recall for Positive sentiment is 0.91, meaning that 91% of all actual Positive messages were correctly identified. Lastly, the F1-Scores evaluate the harmonic mean of precision and recall; for example, the F1-Score for Positive sentiment is 0.87, showing a good balance between precision and recall for detecting positive messages.

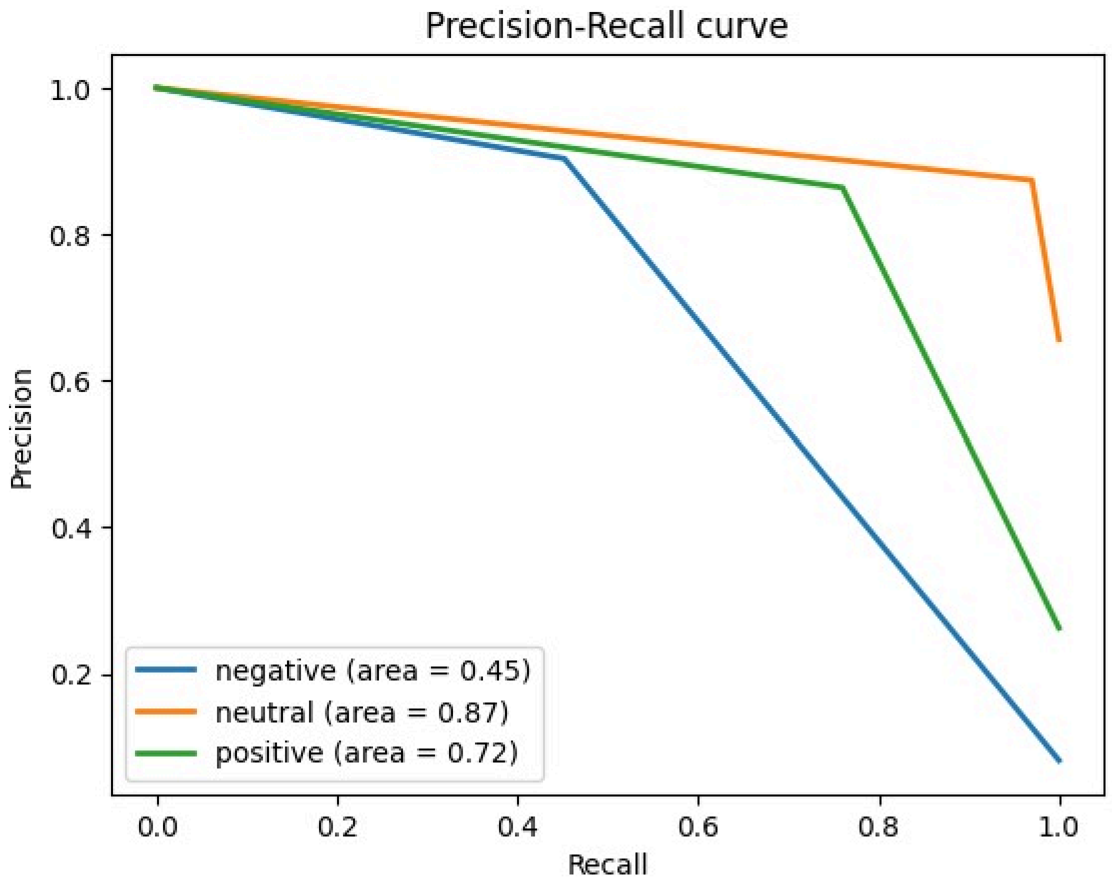
For model training, the model is trained using SVM with a linear kernel for sentiment segregation. The training dataset is trained on text messages that are converted into TF-IDF features. Sentiment polarity is calculated using TextBlob to categorize messages into positive, negative, or neutral sentiments. In our approach to sentiment analysis for individual users, we start by filtering messages according to the sender's name. This allows us to isolate and analyze the content attributed to each specific user. Once the messages are filtered, we calculate sentiment metrics by determining the percentage of positive, negative, and neutral sentiments within the user's messages. This breakdown provides a comprehensive view of the user's overall sentiment distribution. Additionally, to assess potential predatory behavior, we compute a predatory score based on the percentage of negative sentiments. This score helps in evaluating whether a user’s communication patterns suggest potentially harmful or predatory behavior. Through these steps, we aim to provide a nuanced understanding of user interactions and identify any concerning trends in their messaging behavior.

Precision measures the proportion of true positive predictions among all positive predictions made by the model, reflecting the accuracy of positive predictions. For negative sentiment, precision is zero, indicating that none of the messages predicted as negative were actually negative, which highlights a deficiency in the model's ability to identify negative sentiments accurately. In contrast, the precision for neutral sentiment is 0.67, showing that 67% of messages predicted as neutral were indeed neutral. The model exhibits perfect precision for positive sentiment, correctly identifying all positive messages. Recall, on the other hand, gauges the proportion of actual positive cases correctly identified by the model. The model fails to identify any actual negative sentiment messages, as evidenced by a recall of zero for negative sentiment. For neutral sentiment, recall is perfect at 1.00, meaning all actual neutral messages were correctly identified. Positive sentiment recall stands at 0.50, indicating that only half of the actual positive sentiment messages were captured. The F1-Score, which represents the harmonic mean of precision and recall, provides a balanced view of the model's performance. For negative sentiment, the F1-Score is zero, reflecting the model's inability to identify negative sentiments. The F1-Score for neutral sentiment is 0.80, indicating a strong balance between precision and recall for neutral sentiments, while the positive sentiment F1-Score is 0.67, showing a moderate balance for positive sentiments. Accuracy represents the proportion of correct predictions across all sentiment categories, with an overall accuracy of 0.69, indicating that 69% of all predictions were accurate. The macro average calculates metrics independently for each class and averages them, providing a balanced view across all classes without considering their frequency. The macro average values reflect the model's performance equally across classes. In contrast, the weighted average accounts for the support of each class, offering a more comprehensive metric by reflecting the class distribution within the dataset. The weighted average values provide a nuanced assessment of the model's performance, considering the prevalence of each sentiment class.

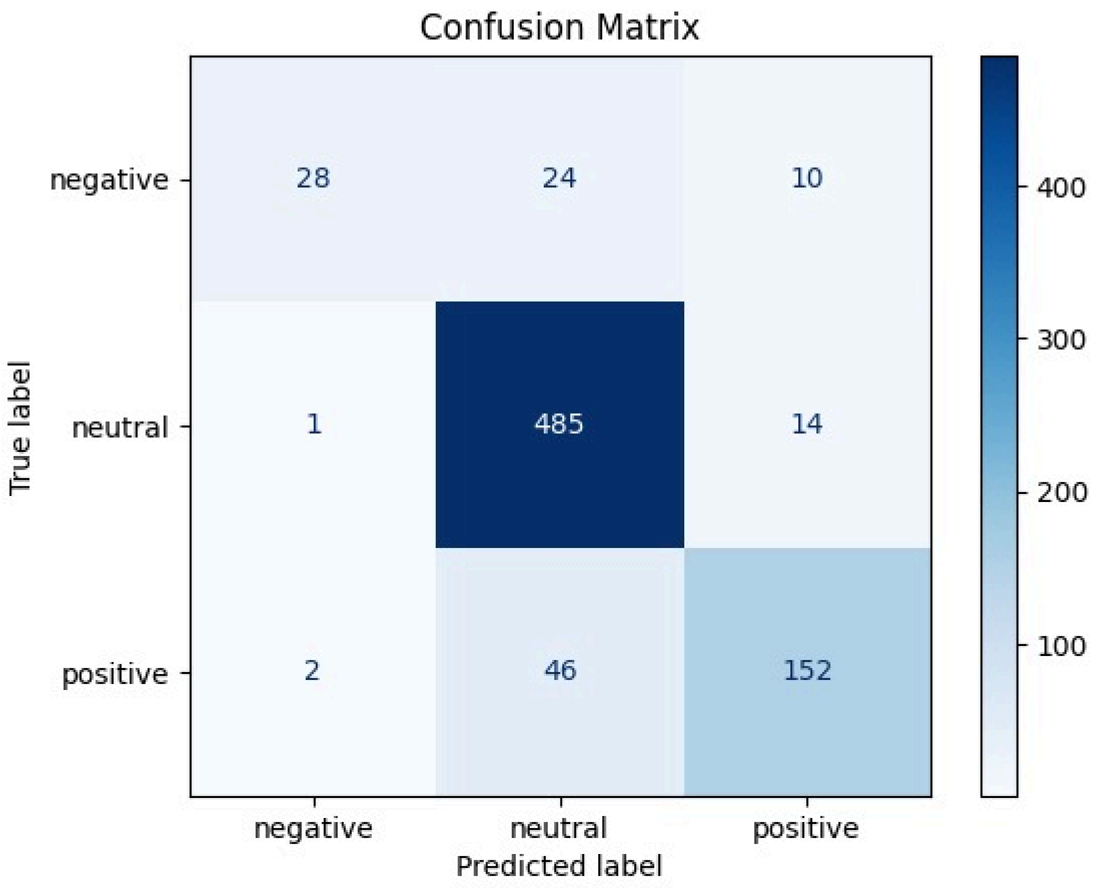
	Precision	Recall	F1-Score	Support
Negative	0.00	0.00	0.00	4
Neutral	0.67	1.00	0.80	10
Positive	1.00	0.50	0.67	2
Accuracy			0.69	16
Macro Avg	0.56	0.50	0.49	16
Weighted Avg	0.54	0.69	0.58	16

Experiment Setup: The experimental setup includes a 1 TB NVMe SSD for storage, ensuring fast data access and processing. The operating system used is Windows 10 Pro, with Python 3.9 as the primary programming language. Key libraries utilized in this setup include pandas for data manipulation and analysis, scikit-learn for implementing and evaluating machine learning models such as SVM and TF-IDF, and TextBlob for sentiment analysis. The development environment comprises Jupyter Notebook for prototyping and VS Code for production-ready code. Version control is managed through Git, facilitating change tracking and collaboration throughout the development process.

Result Graphs



Graph 1a: Precision-Recall Curve showing the model's ability to distinguish between negative, neutral, and positive sentiments.



Graph 1b: Confusion Matrix illustrating the accuracy of sentiment classification across different categories.

Discussion

Analysis of Results:

The sentiment analysis model exhibits robust performance in predicting the neutral class, achieving a high precision of 0.87, recall of 0.97, and F1-score of 0.92. These metrics demonstrate the model's strong ability to correctly classify neutral messages, with minimal misclassification, as it accurately identified 485 out of 500 neutral instances. This performance indicates that the model effectively handles the most prevalent class in the dataset, which is likely neutral, contributing to its overall robustness.

However, the model's performance for the positive class shows room for improvement. With a precision of 0.86 and recall of 0.76, leading to an F1-score of 0.81, the model correctly predicts 152 out of 200 positive instances but often confuses positive messages with neutral ones, resulting in 46 misclassifications. This suggests that the model could benefit from enhanced differentiation between these two sentiment classes, particularly in applications where accurately identifying positive sentiment is critical.

The most significant limitation of the model is observed in its handling of the negative class. The model achieves a precision of 0.90 but a low recall of 0.45, resulting in an F1-score of 0.60. The low recall highlights the model's difficulty in identifying negative messages, with only 28 out of 62 negative instances correctly classified, and 34 misclassified as neutral or positive. This limitation is critical, as accurate identification of negative sentiment is often essential in applications such as feedback analysis or social media monitoring.

A. Strengths

1. **High Precision for Neutral Class:** The model demonstrates strong performance in predicting neutral sentiments, minimizing the occurrence of false positives.
2. **Overall Accuracy:** With an overall accuracy of 0.87, the model is generally reliable across all sentiment classes.

B. Limitations

1. **Low Recall for Negative Class:** The model struggles with correctly identifying negative sentiments, resulting in a low recall, which may lead to an underestimation of negative feedback or opinions.
2. **Confusion Between Positive and Neutral Classes:** There is notable confusion between the positive and neutral classes, which affects the reliability of the model's predictions in these categories.
3. **Class Imbalance:** The performance disparity across classes suggests that the model's effectiveness may be influenced by class imbalance, where more frequent classes (e.g., neutral) are predicted more accurately than less frequent ones (e.g., negative).

References

- List all the academic papers, books, and other resources cited in the paper.