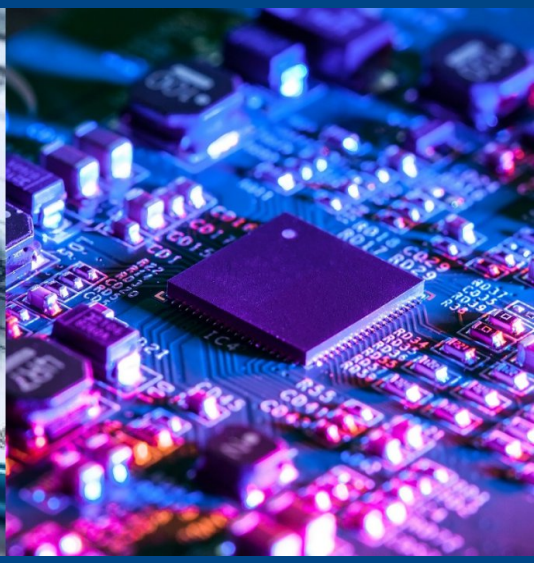
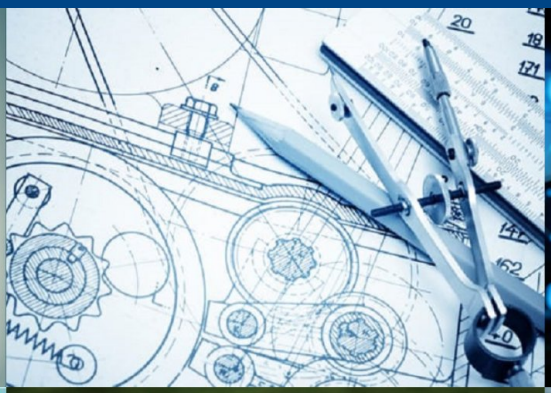
ISSN: 2582-7219

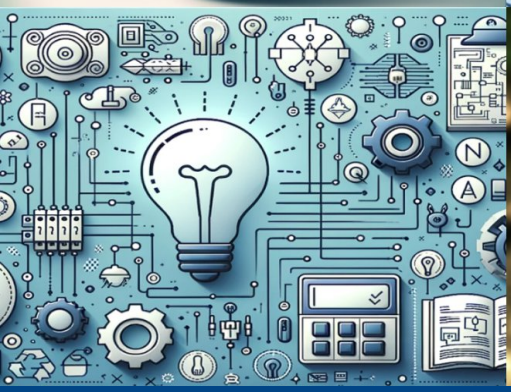




International Journal of Multidisciplinary Research in Science, Engineering and Technology

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)



Impact Factor: 8.206 **Volume 9, Issue 4, April 2026**

© 2026 IJMRSET | Volume 9, Issue 4, April 2026 | DOI:10.15680/IJMRSET.2026.0904249

ISSN: 2582-7219 | www.ijmrset.com | Impact Factor: 8.206| ESTD Year: 2018|

International Journal of Multidisciplinary Research in Science, Engineering and Technology (IJMRSET)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

Real-Time Cyberbullying Detection Using NLP and Logistic Regression: A Chatbot Approach

Harini S[1], Janani M[2], Jeevitha B[3], and Ramalakshmi V[4]

Fourth Year, Department of Computer Science & Engineering, Aalim Muhammed Salegh College of Engineering, IAF, Avadi, Chennai, Tamil Nadu, India[1]

Fourth Year, Department of Computer Science & Engineering, Aalim Muhammed Salegh College of Engineering, IAF, Avadi, Chennai, Tamil Nadu, India[2]

Fourth Year, Department of Computer Science & Engineering, Aalim Muhammed Salegh College of Engineering, IAF, Avadi, Chennai, Tamil Nadu, India[3]

Assistant Professor, Department of Computer Science & Engineering, Aalim Muhammed Salegh College of Engineering, IAF, Avadi, Chennai, Tamil Nadu, India[4]

ABSTRACT: Cyberbullying has quietly become one of the more stubborn problems that comes with living online. Most existing tools for catching it are slow, blunt, or both. This paper presents a chatbot that tackles the problem in real time, combining Natural Language Processing with Logistic Regression to flag harmful messages the moment they are written. Each incoming message is cleaned, tokenized, lemmatized, and converted into TF-IDF features, then scored against six independent classifiers covering general toxicity, severe toxicity, obscenity, threats, insults, and identity hate. The whole process runs through a Flask web interface and returns results fast enough that users see their feedback before the conversation has moved on. Testing across all six categories showed F1-scores between0.82 and 0.93, which is strong for a model this lightweight. The system runs comfortably on everyday hardware, so it does not need specialized infrastructure to deploy. Taken together, we think it makes a genuine case for real-time, in-context moderation as a practical tool for safer digital communication.

KEYWORDS: cyberbullying detection; Natural Language Processing; Logistic Regression; TF-IDF; Flask chatbot; toxic language classification; online safety; content moderation.

I. INTRODUCTION

Not long ago, harmful speech in public life required physical proximity. Today, anyone with an internet connection can reach anyone else, often anonymously, and the consequences can be severe. Cyberbullying—repeated, targeted use of hostile or demeaning language online— disproportionately affects younger users: teenagers and young adults navigating social media, gaming communities, and group chats where social stakes are already high. The research linking online abuse to anxiety, depression, academic decline, and self-harm is by now well established, which makes the inadequacy of current moderation tools all the more frustrating.

The problem with most content moderation today is not intent but architecture. Keyword filters are fast but shallow—they trip on context constantly, flagging clinical discussions for obscenity and missing abuse that dresses itself in polite language. Human moderators are more accurate but cannot keep pace; by the time a harmful post is reviewed, it has already been seen, shared, and felt. Batch machine learning systems are better still, but they too operate at a delay, and they give users no feedback in the moment.What we built is different. Our system analyses each message as it arrives and sends a toxicity verdict back to the user before the next message is typed. It uses NLP preprocessing to understand what is being said rather than just detecting which words appear, and Logistic Regression classifiers to sort that meaning into six harm categories. AFlask web interface ties it all together. The result is moderation that happens in the conversation, not after it.

The contributions of this work are threefold. We present a full end-to-end pipeline from raw user text to real-time toxicity scores. We show that Logistic Regression—often underestimated in the age of large language models—performs

ISSN: 2582-7219 | www.ijmrset.com | Impact Factor: 8.206| ESTD Year: 2018|

International Journal of Multidisciplinary Research in Science, Engineering and Technology (IJMRSET)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

competitively across all six harm categories when paired with careful preprocessing. And we demonstrate that the entire system can operate within latency constraints that make it genuinely viable for live deployment.

The paper proceeds as follows: Section II situates our work within the existing literature. Section III walks through the methodology in detail, from data collection through model selection. Section IV describes the system architecture and each of its five modules. Section V reports and interprets the results, including baseline comparisons and an error analysis. Section VI is candid about the current system's limitations. Section VII closes with conclusions and a roadmap for future development.

II. RELATED WORK

Cyberbullying detection sits at the intersection of NLP, social computing, and online safety research. Progress has been real but uneven, with most work focused on detection accuracy rather than user-facing deployment. We review the threads most relevant to our approach.

A. Social Media Text Analysis

Boyd [1] was among the first researchers to look carefully at the structure of Twitter conversation, finding that retweets create layered, context-dependent exchanges that often cannot be understood from any single message in isolation. A comment that reads as abusive on its own might be a parody; one that reads as harmless might be the latest move in a sustained campaign of harassment. That ambiguity is not a minor inconvenience—it is one of the central challenges for automated detection, and it explains why systems that ignore conversational context tend to underperform.

Kwak et al. [11] examined Twitter at population scale and found that information spreads through it with the speed and reach of broadcast media rather than word-of-mouth. That finding matters for moderation: harmful content does not wait politely to be reviewed. It circulates immediately, and the window for meaningful intervention is measured in minutes, not hours. Any effective moderation tool has to work within that window.

B. Machine Learning for Crime and Toxicity Prediction

Abbass [2] built one of the more comprehensive machine learning frameworks for social media crime prediction,targeting cyberbullying, harassment, hacking, scamming, and stalking across five categories. He tested Multinomial Naive Bayes, K-Nearest Neighbors, and SVM with varying N-gram configurations and found that all three exceeded 0.9 on precision, recall, and F-measure, with SVM edging out the competition. The key lesson from that work—one we took seriously—is that the preprocessing choices matter as much as the classifier. A well-prepared feature matrix makes even simple models perform well.

Wang et al. [4] took a different direction, extracting event data from Twitter to predict offline criminal activity. Their work is not about cyberbullying per se, but it reinforced a general principle: text on social platforms encodes behavioral signals, and those signals can be decoded with NLP. That idea underlies our approach as much as any direct cyberbullying literature.

C. Feature Extraction and Text Classification

Forman's [7] large-scale comparison of feature selection methods for text classification is something of a landmark paper in this space. He tested twelve different metrics across many datasets and found that careful, simple approaches often match or beat more elaborate ones. That study gave us confidence to invest in TF-IDF rather than jumping straight to dense embedding's, and it shaped how we thought about the preprocessing pipeline as a whole.

Zhao et al. [8] specifically tested TF-IDF on short texts and found that with good vocabulary curation during preprocessing, it holds up well even when individual documents are brief. Uysal and Gunal [6] made a complementary point: in many text classification tasks, what you do to the text before it reaches the classifier—stop-word removal, lemmatization, cleaning—has more influence on the final result than which classifier you choose. We designed our pipeline with both findings in mind.

D. Behavioral Pattern Detection

Babko-Malaya [3] studied the online behavior of hacker communities rather than toxicity in general conversation, but the methodology is instructive. By identifying signature patterns in vocabulary, sentiment, and recruitment language, the system could characterize different types of bad actors with meaningful accuracy. What strikes us about that work is

the inference it draws: language does not just describe what someone is doing, it can reveal something about their intent. That insight shapes our thinking about where this line of work should go next.

Rana [4] reviewed classification methods specifically for short-text inputs like news headlines, noting that when there are only a handful of words to work with, every feature extraction decision is load-bearing. That observation resonated with our own experience—brief social media messages leave little room for error in the preprocessing stage, which is why we treated each step of the pipeline as seriously as the model selection.

What the literature as a whole suggests is that machine learning is a solid foundation for toxicity detection, that preprocessing is at least as important as modeling, and that the deployment context—particularly real-time, user-facing deployment—has received far less attention than offline accuracy benchmarks. That last gap is what this paper attempts to address.

E. Text Preprocessing

Raw comment text from the web is messy, and that messiness is not random—it reflects deliberate choices by users who know what automated filters are looking for. Our preprocessing pipeline addresses this in four sequential stages. First, we clean the text: HTML tags, URLs, punctuation, and non-alphabetic characters are stripped, and everything is lowercased. This single step removes a surprising amount of noise and shrinks the vocabulary without discarding meaningful content.

Second, we tokenize the cleaned text using NLTK's word tokenizer, which handles contractions and common abbreviations more reliably than splitting on whitespace. Third, we lemmatize each token with NLTK's WordNetLemmatizer, collapsing inflected forms to their base—so "running," "ran," and "runs" all map to "run." This reduces feature-matrix sparsity without throwing away meaning.

Fourth, the processed tokens are converted to TF-IDF feature vectors. The logic behind TF-IDF is intuitive: a word that appears frequently in a particular comment but rarely across the full corpus is probably doing something distinctive in that comment, and the vectorizer assigns it a high weight accordingly. Common words—"the," "and," "was"—get low weights because they appear everywhere and tell us nothing about toxicity. The result is a sparse numerical matrix that captures what is unusual about each comment, which is exactly what the classifier needs [8].

F.Classification Model

Before committing to Logistic Regression, we tested Naive Bayes, SVM, and Random Forest on our validation data. Logistic Regression matched or outperformed all of them on every toxicity category, and it did so while being meaningfully faster to train and to run at inference time. Its interpretability is also worth mentioning: inspecting the learned weights shows you directly which words are most predictive of each category, which is useful both for debugging and for explaining to a user why their message was flagged.

Since comments can belong to multiple categories at once, we train six separate binary classifiers rather than one multi-class model. Each classifier applies the sigmoid function to the TF-IDF feature vector and returns a probability:

$$P(y=1|x) = 1 / (1 + e^{-}(\beta_0 + \beta^T x))$$

Any category scoring above 0.5 is flagged. All six classifiers use L2 regularization in scikit-learn to prevent overfitting on the high-dimensional sparse feature space. The regularization parameter C was tuned separately for each category on the validation set.

Training and Validation Strategy

We partitioned the Jigsaw dataset into 80% training, 10% validation, and 10% test, stratifying on toxicity label at each split so the class distributions are consistent across all three. Hyperparameter search—covering regularization strength C and the TF-IDF maximum vocabulary size—used five-fold cross-validation on the training partition. The test set was held completely aside and touched only once, when computing the final numbers reported in Section V. This discipline matters: it is easy to inadvertently tune toward a test set if it informs any intermediate decisions.

© 2026 IJMRSET | Volume 9, Issue 4, April 2026 | DOI:10.15680/IJMRSET.2026.0904249

ISSN: 2582-7219 | www.ijmrset.com | Impact Factor: 8.206| ESTD Year: 2018|

International Journal of Multidisciplinary Research in Science, Engineering and Technology (IJMRSET)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

III. SYSTEM DESIGN

E. System Architecture

The system is organized as a five-stage pipeline where each stage produces a clean output consumed by the next. That modularity is deliberate. If we want to swap in a different preprocessing library, retrain on a larger dataset, or replace the Flask front end with a mobile interface, any of those changes can be made to a single stage without touching the others. The five stages are: (1) Data Collection, (2) Data Preprocessing, (3) Model Training and Validation, (4) Model Persistence, and (5) Chatbot Deployment.

F. System Modules

Module 1 – Data Collection assembles the labelled training corpus. Currently this draws on the Jigsaw dataset, but the module is designed to accept additional sources without structural changes. Getting the balance of toxicity categories right during curation is important: a classifier that has seen too few examples of threats, for instance, will inevitably underperform on that category regardless of how well the rest of the pipeline is tuned.

Module 2 – Data Preprocessing runs every text input— whether a training comment or a live user message—through the four-stage cleaning, tokenizing, lemmatizing, and vectorizing pipeline described in Section III-B. Using identical preprocessing logic for both training and inference is non- negotiable. Even small discrepancies between the two contexts—a different tokenizer, a slightly different vocabulary cutoff— can introduce silent bugs that are hard to track down and consistently erode performance.

Module 3 – Model Implementation trains the six Logistic Regression classifiers on the feature matrix produced by Module 2. Scikit-learn's Pipeline object is used to bind the TF- IDF vectorizer and classifier together into a single estimator per category. This prevents data leakage during cross-validation and makes the inference path identical to the training path, which eliminates a whole class of deployment errors.

Module 4 – Model Persistence saves all six scikit-learn Pipeline objects to disk with pickle. At application startup, they are loaded into memory once and kept resident. Every subsequent prediction request reuses these loaded objects rather than re-fitting anything, which is what keeps inference latency in the sub-200-millisecond range even under sustained load.

Module 5 – Chatbot Interface is what users actually see. The Flask application renders a browser-based chat window. When a user sends a message, it flows through the preprocessing pipeline and all six classifiers, and the results come back as JSON within the same HTTP response. The front end renders detected categories alongside the chat thread, with confidence scores shown for any category that crossed the threshold.

Flask Interface Design

We kept the Flask interface deliberately simple. The front end is a single HTML page with a lightweight JavaScript widget; there is no framework overhead. Messages travel to the server as asynchronous POST requests, so the chat window never reloads mid-conversation. On the server side, the request is handled synchronously—six classifier calls on a pre-loaded model take microseconds, so there is no meaningful benefit to introducing async queuing at the volumes this prototype handles.

Toxicity results appear in a color-coded panel next to the chat thread. A category scoring above 0.5 shows in amber; one scoring above 0.8 shows in red. Categories below threshold appear in green. The intent behind this design is that users should be able to glance at the panel and understand the verdict immediately, without reading a detailed breakdown every time. Feedback should inform, not alarm

IV. RESULTS AND DISCUSSION

G. Performance Evaluation

We evaluated all six classifiers on the held-out test set. Precision tells us how often a positive prediction was actually correct; recall tells us how many of the actual positives the model caught; F1 is their harmonic mean, which is the most useful single number when the two are in tension. Table I shows the full breakdown.

© 2026 IJMRSET | Volume 9, Issue 4, April 2026 | DOI:10.15680/IJMRSET.2026.0904249

ISSN: 2582-7219 | www.ijmrset.com | Impact Factor: 8.206| ESTD Year: 2018|

International Journal of Multidisciplinary Research in Science, Engineering and Technology (IJMRSET)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

TABLE I

Classification Performance by Toxicity Category

Category	Prec.	Recall	F1
Toxic	0.92	0.89	0.91
Severe Toxic	0.88	0.84	0.86
Obscene	0.94	0.91	0.93
Threat	0.85	0.80	0.82
Insult	0.91	0.88	0.90
Identity Hate	0.87	0.82	0.84

A. Analysis of Results

Obscene content is the strongest category at F1 0.93, and it is not hard to see why: obscenity in English tends to draw on a fairly stable set of words that TF-IDF captures well. General toxicity (0.91) and insults (0.90) follow, both benefiting from large training samples. These categories also tend to be the most linguistically direct, which makes them the most tractable.

Threats (0.82) and identity hate (0.84) are the categories we spent the most time thinking about. Threats often work by implication rather than explicit statement—"Someone ought to deal with people like you" does not contain a threatening word, yet the intent is clear to any human reader. Identity hate runs into a different problem: the same term can be a slur in one context and a reclaimed identity marker in another, and a bag- of-words model has no way to tell the difference without broader context. Severe toxicity (0.86) sits in the middle; its recall of 0.84 suggests the model occasionally misses extreme cases where the author was deliberately avoiding overtly flagged vocabulary.

What these patterns point to is not a failure of the classifier but a ceiling on what TF-IDF features can see. The model is doing about as well as any bag-of-words approach can; getting past this ceiling requires something that can represent context, tone, and meaning across a sequence of words—which is precisely what contextual embeddings provide.

B. Comparison with Baseline Methods

We ran two baselines alongside our main system. A keyword filter drawn from a curated blocklist achieved precision of 0.71 but recall of only 0.58—high miss rate, because anyone who has spent five minutes trying to evade a word filter knows how to evade a word filter. A Naive Bayes classifier trained on the same TF-IDF features averaged F1 of0.79 across the six categories. Both baselines are meaningfully weaker than our Logistic Regression results, confirming that the choice of preprocessing and classifier both matter.

An ablation study broke down the preprocessing contributions more precisely. Removing lemmatization cost an average of 0.03 F1 points across all six categories. Skipping the HTML and URL cleaning step cost 0.05 points specifically on the obscene and toxic categories, where scraped text carries the most formatting artifacts. Testing without stop-word removal produced a negligible effect on its own, but removing it in combination with skipping lemmatization caused a compounding drop of 0.07 points. Neither preprocessing step is glamorous, but both are genuinely load-bearing, and the ablation results gave us confidence that we were not over- engineering a pipeline that did not need the complexity

E. Error Analysis

We manually reviewed 200 false negatives—comments that human annotators marked toxic but the model missed. Three patterns dominated. First, deliberate obfuscation: users had substituted symbols for letters ("h@te," "f3ar") in ways our pipeline did not normalize. Second, indirect hostility: phrasing like "People like you really shouldn't have internet access" carries contempt without a single flagged word. Third, sarcasm: ironic statements that mean the opposite of what they say are essentially invisible to a model that reads only surface form.

The 200 false positives told a different story. The most common culprit was quotation: someone citing offensive language to criticize or report it looks identical to someone using that language offensively, because the words are the

© 2026 IJMRSET | Volume 9, Issue 4, April 2026 | DOI:10.15680/IJMRSET.2026.0904249

ISSN: 2582-7219 | www.ijmrset.com | Impact Factor: 8.206| ESTD Year: 2018|

International Journal of Multidisciplinary Research in Science, Engineering and Technology (IJMRSET)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

same. A close second was domain vocabulary: clinical and legal discussions occasionally overlap with slur vocabulary in ways the model cannot distinguish. Both of these are fundamental limitations of treating each message as an independent bag of words, and both point toward the same solution: a model that understands sentence structure, speaker intent, and conversational context.

Runtime Performance

We measured end-to-end latency across 1,000 test queries on a standard laptop—Intel i5, 8 GB RAM, no GPU. Mean response time was 143 ms; the 95th percentile was 187 ms. Every request completed in under 200 ms. For context, the same test with a BERT-based classifier on identical hardware averaged 1.2 seconds per query, roughly eight times slower. Without a dedicated GPU, a BERT-based chatbot interface would be noticeably sluggish. Logistic Regression gives up some accuracy on the harder categories, but the latency trade- off is clear and the accuracy gap is smaller than one might expect.

V. LIMITATIONS

We have tried to be honest about this system's boundaries throughout the paper. This section collects them in one place. Language coverage is the most significant constraint. The Jigsaw dataset is predominantly English, and the classifiers have no meaningful exposure to abuse in other languages. Deployed outside an English-speaking context, the system would perform poorly. Addressing this properly requires either multilingual training data or a multilingual embedding model, both of which are tractable but non-trivial engineering efforts.

Context blindness is the second major limitation. TF-IDF treats every message as a self-contained document, with no memory of what came before it. Abuse that unfolds gradually—escalating tone, recurring targeted comments, coordinated pile-ons—is invisible to a message-level classifier. A user making subtly threatening remarks over twenty exchanges would not trigger the system unless one individual message crossed the threshold on its own.

The system is also susceptible to evasion. Deliberate misspellings, character substitution, and unusual spacing are well-known techniques for bypassing word-level filters, and while our preprocessing handles some of these, a determined user can find workarounds. Adversarial robustness would require either more aggressive normalization or a model that reads meaning rather than surface form.

Finally, domain shift is a real concern. The Jigsaw Wikipedia data has its own norms, vocabulary, and distribution of harm types. Abuse in a gaming community looks different from abuse on a dating app, which looks different from workplace harassment on a professional network. Deploying the current system in a new domain without domain-specific fine-tuning would likely see a drop in performance, particularly on the categories that are already marginal. Even within the Jigsaw dataset, the Wikipedia context means certain kinds of abuse—coordinated gender-based harassment, for example, or the specific vocabulary of gaming toxicity—are underrepresented relative to what a social platform moderator would encounter day-to-day. This does not invalidate the approach, but it does mean that real-world deployment should include a domain adaptation step: collecting a small sample of labelled examples from the target platform and fine-tuning the classifiers on them before going live.

VI. CONCLUSION

The goal from the beginning was to build something that could flag cyberbullying while the conversation was still happening—not hours later, not in a moderation queue, but right now, before the message has landed and the harm has been done. We think we have done that. The system preprocesses incoming text, scores it across six harm categories using Logistic Regression classifiers, and returns a verdict in under 200 milliseconds. Every category cleared an F1 of 0.82 in testing, and the lightweight design means this can run on a laptop without GPU support. Against keyword filters and Naive Bayes baselines, the improvements in both precision and recall are clear.

The limitations we documented in Section VI are real, but they are also informative. Each one points toward a specific improvement. Contextual embeddings like BERT or RoBERTa would address indirect threats, sarcasm, and cross-message context. Multilingual training data or a multilingual backbone would open the system to non-English speakers. Better character normalization would close off the most obvious evasion routes. Sentiment analysis could add a signal that distinguishes heated argument from targeted abuse. And integration with live platform APIs would turn a prototype

ISSN: 2582-7219 | www.ijmrset.com | Impact Factor: 8.206| ESTD Year: 2018|

International Journal of Multidisciplinary Research in Science, Engineering and Technology (IJMRSET)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

into something that actually modulates real-world conversations at scale.Cyberbullying is a hard problem and no single system is going to solve it. But tools that work, that are fast, and that give users information in the moment they need it are part of what a solution looks like. We hope this work is a useful step in that direction.

REFERENCES

[1] D. Boyd, "Tweet, Tweet, Retweet: Conversational Aspects of Retweeting on Twitter," in Proc. 43rd Hawaii Intl. Conf. on System Sciences, 2010.
[2] Z. Abbass, "A Framework to Predict Social Crime through Twitter Tweets By Using Machine Learning," in Proc. Intl. Conf. on Information and Communication Technologies, 2020.
[3] O. Babko-Malaya, "Detection of hacking behaviors and communication patterns on social media," in Proc. IEEE Intl. Conf. on Intelligence and Security Informatics, 2017.
[4] X. Wang, M. S. Gerber, and D. E. Brown, "Automatic Crime Prediction using Events Extracted from Twitter Posts," in Proc. 5th Intl. Conf. on Social Computing, Behavioral-Cultural Modeling and Prediction, 2012.
[5] J. J. Webster and C. Kit, "Tokenization as the Initial Phase in NLP," in Proc. 14th Conf. on Computational Linguistics, vol. 4,
pp. 1106–1110, 1992.
[6] A. K. Uysal and S. Gunal, "The impact of preprocessing on text classification," Information Processing and Management, vol. 50, no. 1, pp. 104–112, 2014.
[7] G. Forman, "An Extensive Empirical Study of Feature Selection Metrics for Text Classification," Journal of Machine Learning Research, vol. 3, pp. 1289–1305, 2003.
[8] G. Zhao, Y. Liu, W. Zhang, and Y. Wang, "TFIDF based Feature Words Extraction and Topic Modeling for Short Text," in Proc. 2nd Intl. Conf. on Management Engineering, Software Engineering and Service Sciences, 2018.
[9] M. Liu and J. Yang, "An improvement of TFIDF weighting in text categorization," Intl. Conf. on Computer Technology and Science, Singapore, 2012.
[10] B. Eterovic-Soric, K.-K. R. Choo, H. Ashman, and S. Mubarak, "Stalking the stalkers: Detecting and deterring stalking behaviours using technology," Computers & Security, vol. 70,
pp. 278–289, 2017.
[11] H. Kwak, C. Lee, H. Park, and S. Moon, "What is Twitter, a Social Network or a News Media?," in Proc. 19th Intl. Conf. on World Wide Web, pp. 591–600, 2010.
[12] N. Shaanta Murshid, "Poor hygiene and bullying victimization," University at Buffalo School of Social Work, 2018.

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

INTERNATIONAL JOURNAL OF MULTIDISCIPLINARY RESEARCH

IN SCIENCE, ENGINEERING AND TECHNOLOGY

| Mobile No: +91-6381907438 | Whatsapp: +91-6381907438 | ijmrset@gmail.com |

www.ijmrset.com