www.editada.org

International Journal of Combinatorial Optimization Problems and Informatics, 17(1), Jan-April 2026, 76-84. ISSN: 2007-1558. https://doi.org/10.61467/2007.1558.2026.v17i1.1240

Toward Robust Offensive Language Detection in Urdu YouTube Discourse

Fiaz Ahmad [1], Nisar Hussain [2], Amna Qasim [2], Muhammad Usman [2], Muhammad Zain [2], Momina Hafeez [2], Fatima Hafeez[2], Grigori Sidorov [2]*

[1] The University of Central Punjab (UCP), Punjab, Pakistan,
[2] Instituto Politécnico Nacional (IPN), Centro de Investigación en Computación (CIC), Mexico,
fiaz.ahmad6251@gmail.com, nhussain2022@cic.ipn.mx, amnaq2023@cic.ipn.mx, usmancic21@gmail.com, muhammad23@cic.ipn.mx, mhafeez2025@cic.ipn.mx, sidorov@cic.ipn.mx

Corresponding author.

Abstract. This study addresses the critical task of detecting offensive language in Urdu text on social media, where harmful and offensive comments can cause considerable emotional distress. Using a dataset of Urdu comments collected from YouTube news channels, we employed both deep learning (DL) and machine learning (ML) models for offensive language detection. Features were extracted using Term Frequency-Inverse Document Frequency (TF-IDF) and Count Vectorizer to capture Unigrams, Bigrams, and Trigrams. Four ML models Random Forest (RF), Logistic Regression (LR), Support Vector Machine (SVM), and Gaussian Naive Bayes (NB) were tested, with SVM demonstrating superior performance among the ML models. For the deep learning approach, we implemented Bidirectional Long Short- Term Memory (Bi-LSTM) and Convolutional Neural Network (CNN) using custom Word2Vec and fast Text embeddings. Our results indicated that CNN outperformed all other models, underscoring its effectiveness in detecting offensive language in Urdu text.

Keywords: Machine Learning, Deep Learning, Logistic Regression, Support Vector Machine, Long Short-Term Memory

Article Info
Received August 11, 2025
Accepted Oct 22, 2025

1 Introduction

The rapid expansion of the Internet and the widespread adoption of social media platforms have transformed the way people communicate, providing unprecedented opportunities for individuals to express themselves and connect with others worldwide. Platforms such as YouTube, Facebook, and Twitter allow users to share multimedia content, including photos, videos, and text, creating vibrant online communities. However, this freedom of expression has also led to misuse, with some users exploiting these platforms to post harmful and offensive comments. Such language can have serious repercussions, causing emotional distress, fostering hostility, and negatively impacting individuals' mental health. Addressing this issue is especially important in digital spaces where offensive language can spread quickly and affect large audiences (Aklouche, Bazine, & Ghalia-Bououchma, 2024; Mehra & Hasanuzzaman, 2020).

Detecting offensive language automatically is crucial for maintaining healthier online communities, but this task is challenging, particularly in languages like Urdu. Offensive language detection in Urdu involves unique complexities due to the language's rich morphology, diverse dialects, and intricate script, which present significant challenges for automated processing. Unlike English, Urdu's structure and contextual dependencies make offensive content harder to identify, especially when users intentionally obscure abusive terms. Additionally, many publicly available resources and models for language processing are predominantly focused on English (Abro et al., 2020), leading to a shortage of effective tools and datasets tailored to Urdu. This gap underscores the need for research and development of language-specific approaches that can accurately detect offensive content in Urdu.

To address these challenges, our study applies both machine learning (ML) and deep learning (DL) techniques to detect offensive language in Urdu. We created a dataset of Urdu comments from various YouTube news channels, representing a range of content and community interactions. The study employs traditional ML models, Logistic Regression (LR), Gaussian Naive Bayes (NB), Support Vector Machine (SVM), and Random Forest (RF), alongside deep learning models, specifically Convolutional Neural Network (CNN) and Bidirectional Long Short-Term Memory (Bi-LSTM). Features were extracted using Count Vectorizer and Term Frequency- Inverse Document Frequency (TF-IDF) to capture n-grams, specifically Unigrams, Bigrams, and Trigrams, which allow the models to recognize patterns associated with offensive language (Daouadi, Boualleg & Guehairia, 2024). Our findings reveal that among the ML models, SVM demonstrated the highest accuracy, precision, and recall for detecting offensive content in Urdu.

In contrast, within the deep learning models, CNN outperformed Bi-LSTM, highlighting CNN's effectiveness in capturing key offensive language patterns. By demonstrating the relative strengths of these models in handling Urdu-specific challenges, this study contributes valuable insights into the application of both traditional and advanced computational approaches for offensive language detection in law-resource languages. These results underscore the potential for deploying such models in real-world applications to foster safer, more inclusive digital environments (Hussain et al., 2025).

2 Literature Review

Offensive language detection in online platforms has become an important area of research due to the growing presence of harmful content that impacts social media users' well-being. Various studies have investigated machine learning (ML) and deep learning (DL) approaches to detect offensive language, with a majority focusing on widely used languages like English. For instance, (Hussain & Aslam, 2024) developed a dataset of tweets for hate speech and offensive language, applying ML classifiers to differentiate between hate speech, offensive, and neutral content. Their research demonstrated the effectiveness of ML models in detecting offensive language but also highlighted limitations in handling linguistic nuances. Similarly, Zhang and Luo (Hussain & Aslam, 2024) examined how context-sensitive embedding models, combined with neural network techniques, improve the detection of abusive content in English. Despite the progress made in English and other languages, research on offensive language detection in Urdu remains limited, largely due to the scarcity of publicly available datasets and language-specific models.

Rahman-Laskar, Gupta, Badhani, and Pinto-Avendaño (2024) addressed this gap by collecting Urdu text from various online forums and developing a dataset focused on abusive language, thus setting a foundation for further study in this area. Hussain et al. (2025) and Kaur & Saini (2024) extended this by applying ML and DL techniques, including Logistic Regression and Support Vector Machines, to identify hate speech in Urdu. Their findings suggest that while traditional ML models perform well, DL approaches particularly those involving embeddings—are more effective at capturing con- textual nuances.

In recent years, researchers have explored advanced DL models like Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks for offensive language (Mnassri, Rajapaksha, Farahbakhsh & Crespi, 2022) experimented with CNN and LSTM models for hate speech detection in Hindi and English, achieving high accuracy with the CNN model. Building on this work, Mozafari, Mnassri, Farahbakhsh & Crespi (2024) applied CNNs and LSTMs to Urdu text, finding that the CNN model outperformed others in detecting offensive content. These studies underscore the advantage of CNN in extracting relevant n-grams and capturing linguistic patterns that characterize offensive language [14].

Further, combining ML and DL models with specific feature extraction techniques, such as Term Frequency-Inverse Document Frequency (TF-IDF) and Word2Vec, has proven to enhance performance in multilingual offensive language detection tasks. Mozafari, Mnassri, Farahbakhsh & Crespi (2024) used TF-IDF to capture term relevance in multilingual datasets, successfully identifying patterns associated with offensive language across languages. Meque, Hussain, Sidorov & Gelbukh (2023b) applied custom Word2Vec embeddings in Urdu offensive language detection, which allowed their DL models to better recognize complex and context-specific terms in abusive content. Another study by Mukherjee & Das (2023) integrated CNN and Bi-LSTM models with fastText embeddings for hate speech detection in Bengali, demonstrating the versatility of embedding techniques in low-resource languages.

Our study builds on this body of work by leveraging both ML and DL models to detect offensive language in Urdu text, collected from YouTube news channels. Given the effectiveness of CNNs and Bi-LSTM models in other languages, our approach applies to these models with customized embeddings to improve accuracy. Through these experiments, we aim to contribute to the growing research on multilingual offensive language detection, addressing a critical need in under-resourced languages like Urdu.

3 Methodology

As shown in the methodology flowchart in Figure 1, it combines a structured process to build a machine learning and deep learning system for offensive language detection through the preparation of text data from YouTube comments in Urdu. This collection phase is the first stage in creating Raw Urdu X Data by gathering comments from YouTube over specific time periods on various channels to obtain a wide variety of linguistic samples. Post-collection, a labeling process is performed on the collected data to categorize each comment as either offensive or non-offensive, resulting in ground-truth data. The labeled data is then processed, which will take care of cleaning the textual data (this may include removing noise, handling special characters, tokenization, and normalization). After preprocessing is done, we can choose to move forward in two parallel paths for Traditional ML Models or DL Models. In the Feature Extraction for ML approach, feature extraction is done using a Count Vectorizer (TF-IDF), transforming text data into numerical vectors and, in turn, capturing word frequencies and importance. Several traditional ML models are then trained using these vectors and evaluated through validation and testing for performance.

In the case of the DL approach, word embedding vectors are generated through FastText and Word2Vec, which retain se- mantic relationships in a dense vector representation of words. Then these embeddings are fed into deep learning models to train the model to identify offensive language patterns in the data. The trained DL models are finally assessed in a similar validation and testing phase, allowing the comparison of ML and DL. These two approaches enable us to investigate the effectiveness of the models for offensive language detection in YouTube comments in Urdu.

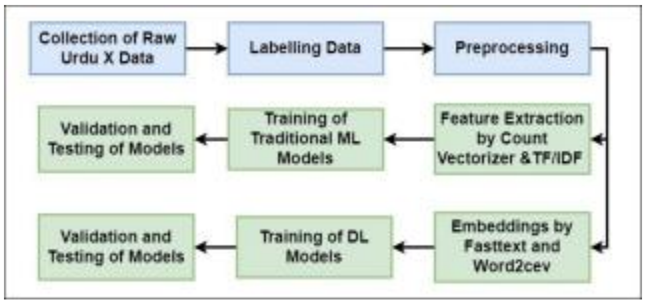
Collection of Raw Urdu X Data
Labelling Data
Preprocessing
Validation and Testing of Models
Training of Traditional ML Models
Feature Extraction by Count Vectorizer &TF/IDF
Validation and Testing of Models
Training of DL Models
Embeddings by Fasttext and Word2cev

Fig. 1. Methodology

3.1 Data collection

As an initial part of this research, only user-based YouTube comments were considered for collecting data on offensive language in Urdu. The dataset was developed by collecting comments across a diverse range of YouTube channels, including those focused on politics, news, and general interest. The dataset is collected from multiple content categories and hence represents a holistic nature of modern Urdu language usage on YouTube, specifically with respect to offensive and abusive language, thus enabling a more focused analysis of the offensive language trends over this medium.

3.2 Preprocessing

In preprocessing the Urdu dataset, many important processes were carried out to make the data clean, consistent, and ready for analysis, especially tasks related to OLID (offensive language identification).

1) *Data Cleaning:* The Urdu text dataset for linguistic study underwent several data cleansing processes to improve it. To reduce noise and make the dataset clearer, unwanted characters such as special symbols, emojis, punctuation, un- necessary whitespace, hashtags or URLs, and other non-standard features were removed. To ensure only genuine Urdu content remained, non-Urdu content (English words, numbers, and Roman Urdu) was filtered out. This filtration process was necessary for retaining the focus on Urdu in the dataset, which is helpful for accurate model training and further analyses. In addition, diacritics were removed as they often introduce inconsistencies and add unnecessary complexity to the text without a change in meaning. Our cleaning procedures resulted in a processed corpus optimized for fast linguistic and computational analysis.

2) *Normalization:* During normalization a few techniques were applied to make the Urdu dataset more consistent and error-free. Since certain characters in Urdu such as" ye" and" Yey" had multiple representations, step one involved character standardization. To resolve those discrepancies, all instances were brought to the same format, ensuring that the dataset was homogeneous. This process eliminated variation in the spelling, creating a consistent dataset. Another way to emphasize this is in casual Urdu, where one writes repeated characters. Repeated letters were converted to normal words while keeping the meaning and reducing redundancy.

3) Tokenization: The third step of data preprocessing was word tokenization. Here the Urdu text was divided into words, phrases, or meaningful sub-units. The tokenization problem in Urdu is unique because of the scripting and complex morphology of the language. In addition, compound words are very common in Urdu and are composed of numerous words to indicate a single meaning; therefore, they need detection as well, so careful tokenization was performed on them to keep semantics intact. This technique helped maintain the contextual meaning of a token and its more effective analysis over the Pakistani Urdu dataset.

4) Stop Words Removal: The fourth step in data preprocessing is defining and handling stop words. Urdu stopwords were customized to improve the existing dataset. Stop words such as" hey,"" or," and" ko" are basic terms with low information value in the whole context. Removing those terms helps reduce the size of your dataset to increase computing efficiency while maintaining core meaning. By using this selective retention, critical contextual information could be preserved, allowing for far better analysis in cases where stop words are important for meaning or finding specific patterns within the data.

5) Stemming: Stemming reduces words to their most basic or root form. Morphological richness makes stemming more challenging for Urdu as compared to many other languages. Stemming in Urdu is used to unify word forms and convert different variations of a word (tenses, plural) into its stem form. Lemmatization is a better normalization because it converts words from different forms to the dictionary form. This is even more useful for languages like Urdu, which have a complex grammar system. Lemmatization helps to identify the root form of a word while still keeping its meaning intact.

6) Handling Roman Urdu: Many users type Urdu phonetically with the Latin alphabet, known as Roman Urdu. The truth is, that Roman Urdu leads to ambiguity and therefore it is necessary to identify and remove it. Roman Urdu is transliterated into Urdu script for inclusion in the research. This requires a proper conversion lexicon or translation model.

7) Text Encoding: Unicode (UTF-8) is used to encode Urdu characters so that the symbols are interoperable across systems and applications. This ensures the data can be processed without encoding complications (a major pitfall, especially in machine learning). Characters that aren't of the regular type or are inconsistent are removed or replaced so that all of the data fits into a uniform style of encoding. After going through the above-mentioned preparation steps, it converts the Urdu dataset into a clean format, which allows normalization and analysis to be performed on this data, making it favorable for downstream linguistic analysis and machine learning tasks of learning offensive language use.

3.3 Feature Extraction

After the preprocessing stage, feature extraction was done using 2 most familiar methods: N-grams with TF-IDF and N- grams with count vectorization. In both of these approaches, we try to transform the text into vector models (numeric) so that the machine learning algorithms can learn what features make a comment abusive vs. neutral and thereby classify them accordingly.

1) N-gram with TF-IDF: TF-IDF computes how important a word (or sequence of words) is to the dataset. N-grams refer to contiguous sequences of words, with unigrams being single words, bigrams being two-word phrases, and trigrams comprising sets of three words. Using n-grams allows the model to learn the context between words in a phrase, which is very important while detecting offense because a sequence of words may have a specific meaning.

In the feature extraction step, Term Frequency (TF) and Inverse Document Frequency (IDF) are two essential measures used to calculate the TF-IDF weights, which indicate how relevant terms are within particular remarks. TF (word frequency) is the number of times any word/phrase is present within a particular comment; more frequently, words contribute more weight, indicating that the term is important in the context. IDF, or Inverse Document Frequency, measures how important a word is in the whole dataset of comments. IDF assigns a lower weight to common terms (like Urdu stop words) and a higher weight to less frequent terms. This is important to identify unique language that might be associated with offensive content, as such language could be very scarce in the data. Each word (or n-gram) is assigned a TF-IDF weight, calculated as the product of its TF and IDF; The resulting value indicates how relevant that word is to a comment in local (comment-centric) context compared to global (over the dataset) context. This weight gives prominence to words commonly used in a given comment but rare across the entire dataset, meaning they are extremely relevant for classification. However, N-grams in TF-IDF add another dimension as it takes into account the order of words to some extent, hence providing more context that may increase semantics. The phrase" batmeez admi" (rude person) together as a bigram could provide more context-level information about abusive language than looking at the words" batmeez" (rude) or" admi" (person) individually. A feature representation based on TF, IDF, and n-grams is quite powerful because it allows the system to identify offensive or neutral content more precisely.

2) N-gram with Count Vectorization: Matrix generation for the frequency of words or n-grams is called Count Vectorization. Unlike TF-IDF, Count Vectorization ignores word scarcity across the corpus and focuses solely on their count in individual comments. The process involves extracting features, where Count Vectorization creates a matrix of vocabulary by taking all unique words or n-grams from the dataset. In this matrix, each unique word or n-gram is represented as a feature or column. We used the Count Vectorizer to log the counts of individual n-grams per comment so that our model could make note of common trends that might indicate Offensive language. Counting n-grams helps to identify repeated patterns, which can be important in finding sequences related to harmful material.

Count vectorization with n-grams has the benefit of prioritizing the number of times a given sequence of words appears, which can uniquely detect repeated offending phrases that might be missed from TF-IDF as they appear in many comments. For instance, certain types of offensive terms or phrases may be repeated multiple times without needing to rely on term rarity in order to signal abusive content quickly. Low-count vectorization further improves the model's ability to recognize offensive language and provides strong evidence of the frequent appearance of abusive vocabulary.

3.4 Data splitting

In this step, we split the dataset into 80% training data and 20% test data. All models trained using the training set to learn, modify parameters, and detect patterns among the datasets. Models built using neural networks go through an iterative training process, where many epochs expose the models to the entire training data multiple times. Step by step this iterative approach makes the model understandable; each iteration allows us to isolate more complex and specific patterns in the data, which leads to better performance.

3.5 Algorithms for Machine Learning and Deep Learning

This work used machine learning and deep learning techniques to detect offensive language from the Urdu dataset.

1) Machine Learning Techniques:

Logistic Regression (LR) is a simple yet effective classification algorithm that models the probability of the binary outcome. The Logistic Regression takes into account text features like word presence or frequency and checks how likely a comment is to be offensive and how likely it is to be neutral. The logistic function converts input values into a value within the range of (0, 1), which then guides classification decisions depending on the model. LR is simple, efficient, and interpretable, which makes it a good baseline for offensive language detection missions.

Naive Bayes (NB) The Naive Bayes classifier uses the concept of Bayes' theorem, which describes conditional independence among features. Although this method assumes that the features are not dependent on each other (a" naive" assumption), it works well in text classification, especially for tasks such as spam or offensive language detection. Naive Bayes then builds a model that learns how likely each word or n-gram is to be seen in either an offensive comment or a neutral comment and classifies new comments based on that learned likelihood. Specifically, Naive Bayes is computationally efficient and performs well with text datasets, which are typically sparse and high-dimensional data.

Support Vector Machine (SVM) Support Vector Machine (SVM) is a robust classifier that tries to find the best boundary (hyperplane) between classes in the feature space. In offensive language detection, SVM patterns word vectors or embeddings to classify comments be- tween offensive and non-offensive. SVM reduces the error of misclassification by maximizing the margin be- tween classes, especially useful when there is a small amount of offensive content. SVM is well suited for high dimensions text data, and it performs very well in identifying complex patterns of language. The final decision from the model is obtained by aggregating thousands of decisions made from different individual trees, helping to reduce bias and variance and making it more robust overall. RF can work with complex interactions in the data and is capable of showing what features are most influential in predicting offensive language.

2) Deep Learning Techniques:

Convolutional Neural Network (CNN) being primarily an image processing architecture, CNNs have been shown to perform well on various text classification problems (offensive language detection among them). In CNN, convolutional layers recognize patterns in the local word sequences that converge into feature maps representing individual words and n-grams. CNNs can capture different features of offensive language using various filters (words, insults, phrases) and applying them at the same time. The spatial size of these features is then reduced by pooling layers to help the model generalize between various contexts. The periodic repetitive expressions that are offensive in nature and require identification is very well accomplished by CNN due to their surface-level feature-capturing abilities.

Bi-directional Long Short-Term Memory (Bi-LSTM) Long-Short Term Memory Networks (LSTMs) are a special kind of RNN suitable for sequence learning, which means that they work well on language problems. Com- pared to a standard LSTM which examines text in just a forward direction, a Bi-LSTM processes it both forwards and backwards so the model can have consideration for words surrounding each word. This is especially helpful for offensive language detection since the context can change a phrase from being non-offensive to offensive. Bi-LSTM gives this simple and robust ability to learn the contextual nuance, as detecting offensive language in Urdu is highly context-dependent.

Here, each of the machine learning and deep learning techniques provides its unique strength towards accurately predicting offensive content in an Urdu dataset, which together translates into the high-level framework.

3.6 Evaluation Measures

Among them, an average precision, recall, as well as F1- score are the most widely employed evaluation measures for offensive datasets.

Precision: We can define precision (P), to be the ratio between all positive cases (TP and FP) containing true positive cases:
$P= TP/(TP+FP)$.

Recall: Recall (R) is the ratio of positive cases that are correctly predicted:
$R= TP/(TP+FN)$.

F1–measure: The F1 measure is defined as the harmonic mean of two other measures that are precision (P) and recall (R) as follows: In our study, we have made use of the weighted average (F1-score):
$F1= (2\times P\times R)/(P+R)$.

4 Results and Discussion

Based on the results, each performs differently from the others, so it is a significant dependency on the type of n-grams you choose, such as Uni, Bi, Tri, or even a combination (Uni + Bag + Tri). These results are discussed in some detail here:

A. Unigram Features

For most of the models, we saw good accuracy using unigram features, especially for SVM which gave us maximum accuracy (95.48%), precision, recall, and F1-Score. Logistic Regression and Random Forest also topped with an accuracy of 94.56% which is still pretty close to SVM. Naive Bayes, on the other hand, achieved the lowest accuracy among all models with an accuracy of 71.55%. The SVM model outperformed in discriminating between classes through unigrams; implying that single words could detect offensive language, and thus, being particularly representative of class areas for this task.

B. Bigrams Features

Performance decreased on nearly all models. The best performance obtained for the bigrams was 86.12% accuracy using Logistic Regression. Naive Bayes also saw a significant bump from its unigram performance, achieving an accuracy of 81.21%, which illustrates that pairing words in some cases better captures the offensive language patterns than single words do. Using bigrams, SVM fell to 85.82% and Random Forest dropped to 76.77%. The results indicate that bigrams do include some contextual information than unigrams, but do not supplement the model with a lot of value for detecting offensive language in Urdu.

C. Trigram Features

Only performed poorly in aggregate across the board. Logistic Regression gave us 69.25% accuracy, Naive Bayes and SVM did almost the same with 75.39% and 75.23% accuracy respectively Trigrams enabled the best-performing model — Random Forest was the least accurate with a 60.20% score. The lower scores across the models indicate that utilizing three-word phrases may introduce some extra complexity or noise which in general makes it harder for the models to ascertain and rightfully predict offensive content within Urdu text. This is likely because there are offensive words that do not require a three-word combination necessarily to detect.

D. Combined Uni + Bi + Tri Features

The combination of the three features (uni, bi & tri) per- formed better than bigram or trigram-only models but lower than top-unigram results. Results in this experiments section This combined feature set produced the best from those models previously, with an SVM achieving the highest performance at 93.40%. The second best was shown by Logistic Regression at 92.26%, while Naive Bayes and Random Forest were only able to produce lower accuracy values of 83.98% and 89.42%, respectively. The n-grams combined gave further context, so they most likely helped the models to learn not just individual offensive words but also over single phrases but did not contribute too much more than unigrams alone.

E. Conclusion

In the case of feature-wise comparison, these results highlight the usefulness of unigram features to detect the presence of offensiveness in Urdu language data, while SVM emerges as a consistently best model over all types of features (shown in Table 2). For SVM, the use of bigrams and trigrams adds some value but has a much less significant impact than the inclusion of unigrams alone. The conclusion from this is that unigrams, or single word-based methods, are able to capture the pattern of Offensive language well in Urdu text, and since we used SVM which also works well on sparse matrices with a high number of dimensions.

Table 1: measure values of various word-level n-grams across different ML and DL models

Features	**Model**	**Precision**	**Recall**	**F1-Score**	**Accuracy**
Uni	Logistic Regressio	94.62	94.56	94.56	94.56
	Naïve Bayes	73.16	71.55	70.86	71.55
	SVM	95.50	95.48	95.48	95.48
	Random Forest	94.60	94.56	94.56	94.56
Bi	Logistic Regressio	86.19	86.12	86.12	86.12
	Naïve Bayes	81.34	81.21	81.21	81.21
	SVM	85.85	85.82	85.82	85.82
	Random Forest	78.06	76.77	76.39	76.77
Tri	Logistic Regressio	73.48	69.25	67.42	69.25
	Naïve Bayes	78.24	75.39	74.82	75.39
	SVM	77.68	75.23	74.75	75.23
	Random Forest	70.40	60.20	53.31	60.20
Uni +Bi+Tri	Logistic Regressio	92.26	92.26	92.26	92.26
	Naïve Bayes	84.17	83.98	83.93	83.98
	SVM	93.32	93.65	93.25	93.40
	Random Forest	89.26	89.57	89.45	89.42

Figure 2 shows the bar chart of F-measure values of different word-level n-grams (uni, bi, and tri-gram alone or their combined uni + bi + tri) based on the machine learning (ML) and deep learning (DL) models applied for offensive language detection on the Urdu dataset. F-measure (or F1- score) is a metric of performance combining precision and recall that captures a model's performance in terms of correctly identifying offensive language, false positives, and false negatives.
YouTube comments in Urdu.

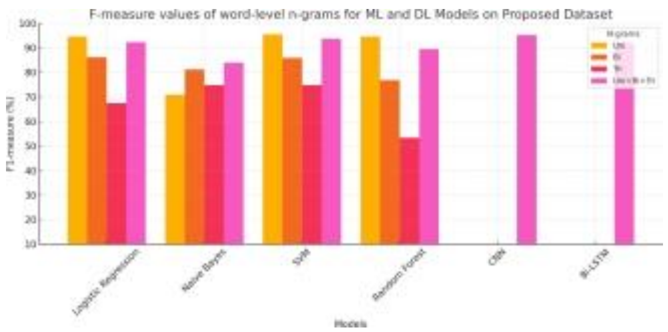

Fig. 2. F-measure values of various word-level n-grams across different ML and DL models

ML Models: Overall, F-measure scores for general n- gram-based configurations reveal that the combined Uni+ Bi + Tri-gram setting provides the highest F-measure values, with Logistic Regression, SVM, and Random Forest consistently outperforming other models. Naive Bayes do relatively poorly compared to the other ML models, particularly for trigrams.

DL Models: Despite being simple word-level representations, CNN and Bi-LSTM models obtain high F-measure values and are relatively stable across various n-grams, that indicate these two models are able to generalize towards the complexity of language (note the small difference between the max-micro and min-micro for each model). For DL models also the combined Uni + Bi + Tri-gram configuration performs slightly better.

This comparison shows that, although the n-gram combined setting improves the performance of all models to some extent, the deep learning (CNN and Bi-LSTM models) outperforms all traditional ML models with solid performance against specific combined settings.

A. DL Model Performance on Proposed Dataset

Table 2 summarizes the results of Deep Learning (DL) models, in which the Convolutional Neural Networks (CNN) and Bidirectional Long Short-Term Memory (Bi-LSTM) models applied over our proposed dataset to identify offensive language in Urdu text. This shows four key performance metrics: Precision, Recall, F1-Score, and Accuracy; each measured in percentage form. These metrics provide an overall measure of how well the model is able to detect offensive language while minimizing false positives and negatives.

Table 2: Results of DL Models on Proposed Dataset

Model	**Precision**	**Recall**	**F1-Score**	**Accuracy**
CNN	97.22	94.23	95.15	95
Bi-LSTM	92.12	91.56	92.22	92

Table 2 shows the performance of the CNN model is better than Bi-LSTM on all indexes. The CNN has a Precision of 97.22%, which means that it is capable of accurately predicting offensive content while misclassifying non-offensive content as offensive. It has a Recall of 94.23%, which means it is correctly dodging most of the real offensive instances in the dataset. Measuring the F1-Score at 95.15%, CNN displays fair precision and recall combining well to propagate its efficiency at offensive language detection. Moreover, CNN attains an Accuracy of 95%, proving its generalizability through all samples of the dataset.

However, the Bi-LSTM model gives comparable performance too, achieving a Precision of 92.12%, Recall of 91.56% and an F1-Score of 92.22%. However, these metrics are lower than the ones for CNN but still show decent performance overall. Accuracy-wise, the Bi-LSTM scores 92%, which is a good score but still sits slightly behind CNN. This performance difference could be explained by the higher efficacy of CNN in finding spatial characteristics in text data that is generally short, as is the case for YouTube comments (Abro et al., 2020).

Overall, this shows that the CNN model performs better, and it is most suitable option in detecting material/offensive language for given Urdu text classification task. The CNN, with greater precision, recall, F1-score and accuracy indicate that the model is better able to learn the complex patterns in this data classifying it more reliably. This discovery demonstrates the resilience of CNN and its appropriateness for text classification, especially where data are rich or diverse as with YouTube remarks.

5 Conclusion and Future Work

This study demonstrates combined Machine learning (ML) and deep learning (DL) models were effective for Urdu offensive language detection in YouTube comments, with ML models such as SVM performing better using simple word- level features, while DL methods outperformed the baseline on harder to detect patterns with CNN performing best of all. CNN is thus a potentially great tool for automating low- resource languages content moderation and that should be helpful in reducing the effect of the UGC. In future work, if we collect more data not only from YouTube but also other popular platforms, such as Twitter and Facebook, our model will be even more generalizable.

Furthermore, incorporating modern pre-trained language models like BERT and multilingual transformers could provide boost performance by capturing even deeper contextual information in Urdu. To promote safer online spaces, this can be extended via the use of data augmentation methods to mitigate class imbalance and by developing hybrid models which integrate ML and DL techniques for offensive language detection (particularly in low-resource and multilingual settings).

Acknowlegments
This work was partially supported by the Mexican Government through the grant A1-S-47854 of CONACYT, Mexico, and the grants 20254236, 20253468, and 20254341 provided by the Secretar´a de Investigacio´n y Posgrado of the Instituto Polite´cnico Nacional, Mexico. We gratefully acknowledge the computing resources made available through the Plataforma de Aprendizaje Profundo para Tecnolog´ıas del Lenguaje of the Laboratorio de Superco´mputo at the INAOE, Mexico, funded by CONACYT. Additionally, we express our sincere gratitude to Microsoft for their support through the Microsoft Latin America PhD Award, which has significantly contributed to the success of this work.

References

Aklouche, B., Bazine, Y., & Ghalia-Bououchma, Z. (2024). *Offensive language and hate speech detection using transformers and ensemble learning approaches*. Computación y Sistemas, 28(3), 1031–1039.

Mehra, S., & Hasanuzzaman, M. (2020). *Detection of offensive language in social media posts* (Doctoral dissertation).

Sigurbergsson, G. I., & Derczynski, L. (2019). *Offensive language and hate speech detection for Danish*. arXiv. https://arxiv.org/abs/1908.04531

Abro, S., Shaikh, S., Khand, Z. H., Zafar, A., Khan, S., & Mujtaba. (2020). *Automatic hate speech detection using machine learning: A comparative study*. International Journal of Advanced Computer Science and Applications, 11(8).

Daouadi, K. E., Boualleg, Y., & Guehairia, O. (2024). *Comparing pre-trained language model for Arabic hate speech detection*. Computación y Sistemas, 28(2), 681–693.

Hussain, N., Qasim, A., Mehak, G., Kolesnikova, O., Gelbukh, A., & Sidorov, G. (2025). *Hybrid machine learning and deep learning approaches for insult detection in Roman Urdu text*. AI, 6(2), 33.

Hussain, A., & Aslam, A. (2024). *Hate speech against women and immigrants: A comparative analysis of machine learning and text embedding techniques*. Journal of Applied Research and Technology, 22(4), 548–559.

Rahman-Laskar, S., Gupta, G., Badhani, R., & Pinto-Avendaño, D. E. (2024). *Cyberbullying detection in a multi-classification codemixed dataset*. Computación y Sistemas, 28(3), 1091–1113.

Hussain, N., Qasim, A., Mehak, G., Kolesnikova, O., Gelbukh, A., & Sidorov, G. (2025). *ORUD-Detect: A comprehensive approach to offensive language detection in Roman Urdu using hybrid machine learning–deep learning models with embedding techniques*. Information, 16(2), 139.

Kaur, M., & Saini, M. (2024). *Artificial intelligence inspired method for cross-lingual cyberhate detection from low resource languages*. ACM Transactions on Asian and Low-Resource Language Information Processing, 23(9), 1–23.

Mnassri, K., Rajapaksha, P., Farahbakhsh, R., & Crespi, N. (2022, December). *BERT-based ensemble approaches for hate speech detection*. In GLOBECOM 2022: IEEE Global Communications Conference (pp. 4649–4654). IEEE.

Mozafari, M., Mnassri, K., Farahbakhsh, R., & Crespi, N. (2024). *Offensive language detection in low resource languages: A use case of Persian language*. PLOS ONE, 19(6), e0304166.

Meque, A. G. M., Hussain, N., Sidorov, G., & Gelbukh, A. (2023). *Guilt detection in text: A step towards understanding complex emotions*. arXiv. https://arxiv.org/abs/2303.03510

Meque, A. G. M., Hussain, N., Sidorov, G., & Gelbukh, A. (2023). *Machine learning-based guilt detection in text*. Scientific Reports, 13(1), 11441.

Mukherjee, S., & Das, S. (2023). *Application of transformer-based language models to detect hate speech in social media*. Journal of Computational and Cognitive Engineering, 2(4), 278–286.