NORTH-AMERICAN
INTERDISCIPLINARY STUDIES

Article

Advancing Insult Detection in Roman Urdu Text: A Hybrid Machine Learning and Deep Learning Approach

Sarah Khan, Aisha Riaz, Farhan Ahmed, Maria Khan, Rida Iqbal

Instituto Politécnico Nacional, Centro de Investigación en Computación, Mexico City

Abstract: Thisstudy introduces a new model for detecting insults in Roman Urdu, filling an important gap in natural language processing (NLP) for low-resource languages. The transliterated nature of Roman Urdu also poses specific challenges from a computational linguistics perspective, including non-standardized grammar, variation in spellings for the same word, and high levels of code-mixing with English, which together make automated insult detection for Roman Urdu a highly complex problem. To address these problems, we created a large-scale dataset with 46,045 labeled comments from social media websites such as Twitter, Facebook, and YouTube. This is the first dataset for insult detection for Roman Urdu that was created and annotated with insulting and non-insulting content. Advanced preprocessing methods such as text cleaning, text normalization, and tokenization are used in the study, as well as feature extraction using TF–IDF through unigram (Uni), bigram (Bi), trigram (Tri), and their unions: Uni+Bi+Trigram. We compared ten machine learning algorithms (logistic regression, support vector machines, random forest, gradient boosting, AdaBoost, and XGBoost) and three deep learning topologies (CNN, LSTM, and Bi-LSTM). Different models were compared, and ensemble ones were proven to give the highest F1-scores, reaching 97.79%, 97.78%, and 95.25%, respectively, for AdaBoost, decision tree, TF–IDF, and Uni+Bi+Trigram configurations. Deeper learning models also performed on par, with CNN achieving an F1-score of 97.01%. Overall, the results highlight the utility of n-gram features and the combination of robust classifiers in detecting insults. This study makes strides in improving NLP for Roman Urdu, yet further research has established the foundation of pre-trained transformers and hybrid approaches; this could overcome existing systems and platform limitations. This study has conscious implications, mainly on the construction of automated moderation tools to achieve safer online spaces, especially for South Asian social media websites.

Keywords: deep learning; machine learning; support vector machine

1. Introduction

The exponential growth of social media platforms has transformed communication into a global, instantaneous phenomenon. However, it has also facilitated the propagation of harmful and derogatory content, including insults that can lead to psychological distress, perpetuate discrimination, and undermine the quality of online discourse. While much

research has been dedicated to the detection of offensive language, the task of identifying insults specifically remains underexplored. Insults, being a distinct and nuanced subset of offensive language, often require more contextual understanding and cultural sensitivity to detect accurately [1,2]. This gap in research is even more pronounced in low-resource languages like Roman Urdu.

This study improves the existing technology by addressing the issue of Roman Urdu, a low-resource language that has variant code-mixing and non-standardized grammar. Contrary to the existing studies that target predominantly high-resource languages, we present a new annotated dataset focused towards Roman Urdu and evaluate both classical machine learning algorithms and cutting-edge deep learning models and show their effectiveness in the proposed scenario. Previous work in low-resource languages have highlighted the importance of constructing annotated datasets for successful NLP applications. The current work extends these findings to provide a large Roman Urdu dataset for insult detection and evaluate sophisticated machine learning and deep learning techniques for the solution to the linguistic complexity of the language [3]. The detection of insulting language in Roman Urdu has not been adequately addressed in existing research, leaving a critical void in the development of robust and inclusive content moderation tools.

Lexicon-based and rule-based methods have been pivotal for insult detection, but they are primarily limited to languages with formal grammar and syntax that are less inclined to use phonetics for spellings, thus assuming limited applicability on Roman Urdu. This study enhances the strengths of data-driven approaches by employing n-gram-based TFIDF features and running pre-trained embeddings (Word2Vec and fastText) specific to the idiosyncratic nature of the Roman Urdu language.

This study aims to fill this gap by introducing a novel dataset specifically designed for insult detection in Roman Urdu. To the best of our knowledge, this is the first dataset dedicated exclusively to this task, providing a valuable resource for advancing research in this domain. The dataset has been meticulously annotated to capture the nuanced and context-dependent nature of insults, distinguishing them from other types of offensive language. This contribution marks a significant step forward in addressing the unique linguistic challenges associated with Roman Urdu.

To evaluate the effectiveness of different approaches to insult detection, we employ a comprehensive set of machine learning and deep learning algorithms. The study utilizes ten machine learning models, including logistic regression, support vector machines, random forest, and gradient boosting, which have demonstrated their efficacy in previous NLP tasks. Additionally, we incorporate three deep learning architectures, convolutional neural networks (CNNs), long short-term memory (LSTM) networks, and bidirectional LSTMs (Bi-LSTMs), to capture the semantic and contextual intricacies of Roman Urdu insults [4,5].

Embedding techniques such as Word2Vec and fastText are employed to represent Roman Urdu text in dense, context-aware vector spaces. These embeddings are critical for capturing the semantic relationships within the data and for enhancing the performance of both machine learning and deep learning models. By benchmarking these models on the newly developed dataset, this study seeks to identify the most effective methodologies for insult detection in Roman Urdu, thereby establishing a foundation for future research in this area [6,7].

The implications of this research extend beyond academia, offering practical applications for social media platforms operating in South Asia. Automated insult detection systems can aid in moderating harmful content, reducing the workload on human moderators, and creating safer digital environments for users. Furthermore, this work underscores the importance of language-specific solutions in NLP, especially for under-represented languages like Roman Urdu, which often fall outside the scope of mainstream research [8,9].

The remaining sections of this paper are organized as follows: Section 2 presents a review of the existing literature on insult detection and related topics, identifying gaps in current research. Section 3 describes the methodology, including the dataset development process and the experimental setup. Section 4 discusses the results, comparing the performance of various models and analyzing their implications. Finally, Section 5 concludes the study and outlines future directions for research in insult detection and content moderation.

2. Literature Review

Insult detection, a nuanced branch of offensive language identification, has garnered increasing attention due to its role in curbing harmful online behavior. While research in high-resource languages such as English has proliferated, studies focusing on under-resourced languages like Roman Urdu remain sparse [10]. Roman Urdu, widely used in informal digital communication, presents unique linguistic challenges such as code-switching with English, absence of standard grammar or spelling rules, and significant reliance on cultural context. These characteristics necessitate customized approaches to NLP tasks like insult detection [11].

For the purposes of this study, hate speech is defined as any explicit or implicit statement, speech, writing, or general communication that, when acted upon, advocates for violence, discrimination, or promotes hostility on the basis of race, religion, gender, or political ideology. Insulting language is a derogatory word that comes from the abusive tone used to insult others and is used in the process of demeaning them. The proposed definitions are also in agreement with previous studies, including [1], which articulated the difficulty in identifying hate speech, and [2], which argued that accurate contextual understanding is vital when defining and detecting harmful content.

The initial strides in offensive language detection involved lexicon-based and rule-based approaches that relied on pre-compiled dictionaries of offensive terms. Although these methods were effective for small-scale implementations, they often failed to generalize in diverse linguistic environments [12]. Lexicon-based approaches are particularly inadequate for Roman Urdu due to its informal and highly variable nature, underscoring the need for data-driven methods.

Machine learning (ML) algorithms have revolutionized text classification tasks by introducing data-centric approaches. For example, ref. [13] developed a machine-learning framework that outperformed lexicon-based systems in detecting abusive language. Similarly, research by Warner and Hirschberg (2012) demonstrated the effectiveness of ML algorithms in categorizing hate speech. However, these efforts were primarily focused on high-resource languages and lacked the contextual understanding needed for insults, especially in Roman Urdu.

Deep learning (DL) methodologies have further advanced offensive language detection by capturing semantic and contextual relationships in text. Ref. [14] explored the use of convolutional neural networks (CNNs) for hate speech detection, highlighting their robustness in handling informal language. Similarly, ref. [15] demonstrated the superiority of recurrent neural networks (RNNs) over traditional models in recognizing contextually driven offenses. Despite their success, such models have rarely been applied to Roman Urdu due to the unavailability of annotated datasets tailored to their unique linguistic features.

In the multilingual domain, studies have explored the challenges of detecting offensive content in low-resource languages. For instance, ref. [16] organized a shared task for offensive content detection in English, German, and Hindi, revealing the disparities in detection accuracy across languages. Their findings emphasized the need for language-specific datasets and embeddings for effective text classification. In another study, ref. [17] developed a benchmark dataset for offensive language detection in Hindi, showcasing the

impact of cultural nuances on model performance. However, similar efforts for Roman Urdu have been conspicuously absent, leaving a critical gap in research.

The unique challenges of Roman Urdu—such as its informal syntax, phonetic spellings, and frequent code-mixing—require embeddings and models specifically adapted to its characteristics. Ref. [18] proposed domain-specific embeddings for low-resource languages, which improved performance in sentiment analysis and hate speech detection tasks. Building on such advancements, embedding techniques like Word2Vec and fastText have shown promise in low-resource settings by providing dense and meaningful representations of words [19]. Yet, these embeddings remain underutilized for Roman Urdu insult detection, largely due to the lack of annotated corpora.

Recent works on offensive language detection use transformer-based architectures and contextual embeddings. For instance, ref. [20] showed that pre-trained language models such as BERT are effective for offensive language detection in multiple languages. Likewise, refs. [21,22] emphasizes the use of multilingual embeddings to support low-resource languages. These studies inform how transformer models adapt to linguistic complexities, such as code-mixing and non-standard grammatical constructs, both of which are salient to Roman Urdu.

Finally, existing studies have acknowledged the importance of targeted datasets in advancing NLP for under-represented languages. Ref. [23] argued that datasets annotated with cultural and contextual considerations are crucial for tasks like hate speech and insult detection. This study builds on this insight by creating the first annotated dataset for insult detection in Roman Urdu. The dataset is designed to capture the subtleties of insults in informal and mixed-language settings, enabling both ML and DL models to benchmark their performance.

2.1. Defining Hate Speech and Insulting Language

Manipulating hate speech and disparaging language are defined as words that degrade, provoke harm, or spread hostility against people or classes based on factors like race, faith, gender, or political ideologies. The art of making sense of harmful content becomes complex in the case of Roman Urdu due to linguistic intricacies such as transliteration, non-standard grammar, and code-mixing. We define hate speech in this study as explicit incitement to violence or discrimination and insulting language as more vague derogatory language or abusive tone. These definitions served as the basis for the annotation process and provided the same backbone for labeling whatever content was considered offensive.

2.2. Contributions of the Study

This research makes several key contributions to the field of NLP and insult detection, particularly for Roman Urdu:

- **Novel dataset for insult detection:** we developed the first annotated dataset for insult detection in Roman Urdu, addressing a critical resource gap for this language.
- **Innovative approach:** this study combines a unique dataset with advanced ML and DL techniques, presenting a novel framework for insult detection tailored to Roman Urdu's linguistic challenges.
- **First focused study on Roman Urdu insults:** to the best of our knowledge, this is the first research specifically addressing insult detection in Roman Urdu, filling a significant gap in the field.

Through these contributions, this research not only advances the state of the art in insult detection but also emphasizes the importance of developing language-specific solutions for NLP tasks in low-resource languages like Roman Urdu.

Table 1 presents a comparative analysis of the current research studies related to insult detection in different languages, datasets, and approaches. Most of these previously conducted studies have solely investigated the detection of insult using classical machine learning (ML) methods, hybrid classifier models, and more recently, deep learning (DL) methods. Although most of these are focused on English or other high-resource languages, there has been very little work on Roman Urdu, which is a code-mixed language popular on online resources. The table summarizes the datasets used, the types of approaches being utilized from rule-based NLP systems to hybrid ML-DL frameworks, and major conclusions from these works. Existing approaches include Insult Detection in Roman Urdu Text: Hybrid machine learning and deep learning approaches; however, none of them cover the approaches they took for Roman Urdu text. The proposed framework combines hybrid ML-DL models with embedding techniques to tackle the challenges associated with Roman Urdu text processing and advance the state of the art in multilingual offensive language detection.

Table 1. Comparison of past studies about insulting language detection from social media comments.

Author(s)	Year	Focus	Dataset	Methods	Key Findings
[24]	2018	Automatic detection of insulting sentences in conversation	Simulated conversations dataset	SVM, NLP techniques	Demonstrated automated detection of insults using linguistic and ML approaches.
[25]	2010	Embodied insult detection effect	Experimental stimuli	Behavioral experiments	Found that insults with bodily associations are detected faster.
[26]	2015	Insult detection in social network comments	Social media comments	Possibilistic fusion approach	Proposed a fusion-based approach for effective insult classification.
[27]	2008	Detecting flames and insults in text	Custom datasets from forums	Pattern-based methods	Focused on insults in flame comments using pattern-based detection.
[28]	2017	Detecting insults in social commentary	Social commentary data	Rule-based NLP and ML models	Highlighted challenges in detecting context-sensitive insults in social comments.
[29]	2014	Insult detection in Hindi	Hindi language datasets	Feature engineering, SVM	Developed techniques for insult detection tailored to Hindi texts.
[30]	2023	Insult detection in Turkish using machine learning	Turkish language datasets	ML algorithms (SVM, RF, etc.)	Compared ML algorithms for Turkish insult detection, with promising results.
[31]	2020	Insult detection using a CNN-LSTM model	Custom social media dataset	Hybrid CNN-LSTM	Proposed a hybrid DL model outperforming traditional ML methods.
[32]	2017	Detecting hate speech and insults in social commentary	Social commentary data	NLP, logistic regression	Differentiated insults from hate speech with high accuracy.
[33]	2018	Detection of insulting comments in online discussions	Online forum comments	Hybrid intelligent systems	Leveraged hybrid techniques for improved detection accuracy.
[34]	2020	Detection of social media insults using NLP	Social media posts	ML algorithms (SVM, RF, etc.)	Comparative study of multiple algorithms for insult detection.

3. Methodology

The methodology of this study encompasses the creation of a novel dataset, preprocessing techniques, feature extraction using TF–IDF vectorization, and the application of eight machine learning (ML) and three deep learning (DL) algorithms for insult detection in Roman Urdu text, as depicted in Figure 1.

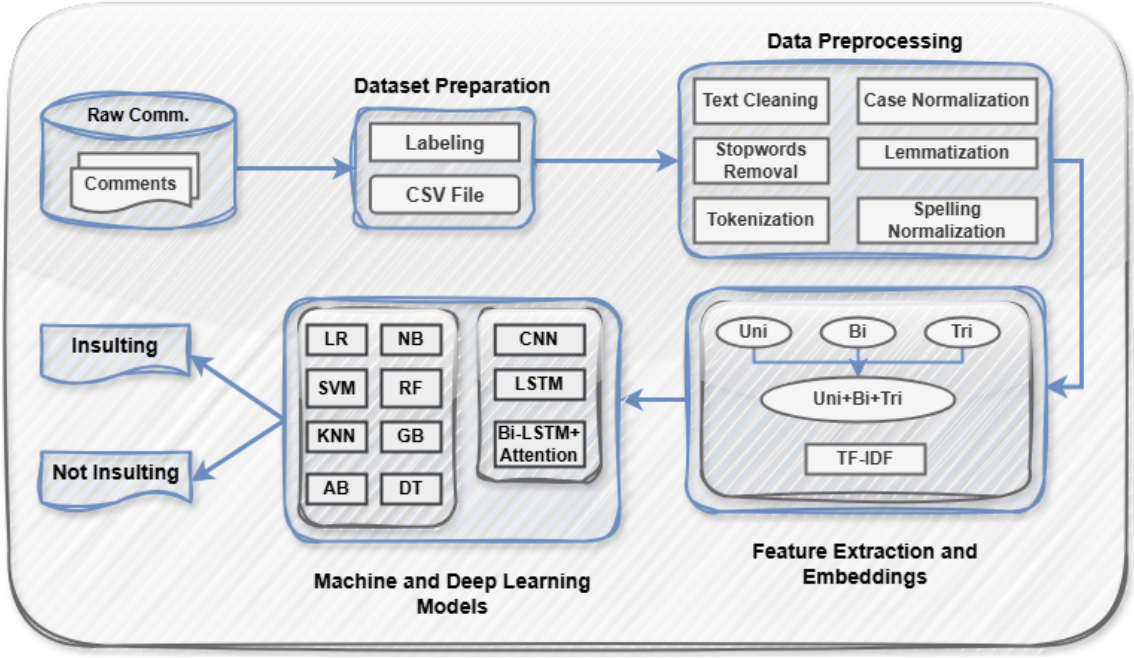

Figure 1. Proposed methodology.

3.1. Dataset Creation

The dataset used in this study comprises 46,045 comments collected from social media platforms, including Twitter, Facebook, and YouTube. These comments were sourced from Roman Urdu pages and groups that frequently contain user-generated content. The dataset was manually annotated into two categories: insulting and non-insulting comments. To ensure diversity and representativeness, the data include comments from various domains, such as entertainment, politics, and general discussions.

This consisted of 3 majors categories: entertainment (40%), politics (35%) and general (25%) comments. The distribution was carefully chosen to provide a fair representation of various contexts and to mirror the real-life usage of Roman Urdu, where it is still predominantly employed in writing. However, we recognize that the potential category imbalance would still impact the model performance, which we minimally address through our stratified sampling approach during both the training and evaluation phases.

3.2. Annotation Process

To account for the cultural nuances in Roman Urdu insults, we provided annotators with guidelines and examples of what constitutes as an insulting or non-insulting comment. The guidelines underscore inclusivity with contextual sensitivity, for example, capturing code-mixed expressions and slang. To make the data annotation reliable, three Roman Urdu expert linguists were hired for this purpose. Inter-annotator agreement was measured by implementing Cohen's kappa with a score of 0.85, which reflects substantial agreement and confirms the dataset quality.

This annotated dataset represents the first significant resource for insult detection in Roman Urdu, addressing a critical gap in natural language processing (NLP) research.

3.3. Data Preprocessing

Roman Urdu, being a very informal and colloquial script, posed challenges when adapting preprocessing methods like spelling normalization and lemmatization. A custom dictionary was created, comprising common Roman Urdu spellings and their standard normalized forms. Also, custom scripts were used to address common shifts and errors in transliterations. We added manual reviews to complement these tools, utilizing the expertise of native speakers. The raw dataset underwent a series of preprocessing steps to prepare it for modeling:

- **Text cleaning:** removed URLs, hashtags, user mentions, emojis, special characters, and excessive white space.
- **Case normalization:** standardized all text to lowercase to reduce variability.
- **Tokenization:** split sentences into individual words.
- **Stopword removal:** eliminated common words (e.g., "hai", "ka", "ko") that do not contribute meaningfully to classification.
- **Spelling normalization:** corrected common spelling variations in Roman Urdu.
- **Lemmatization:** reduced words to their root forms where applicable.

These preprocessing steps were designed to address the linguistic and structural challenges unique to Roman Urdu, such as the lack of standard spelling and the prevalence of code-mixing with English.

3.4. Feature Extraction

To represent the text data numerically, we employed TF–IDF (term frequency-inverse document frequency) vectorization. This method emphasizes the importance of terms that are unique to a document while downplaying commonly occurring terms across the dataset.

An n-gram modeling allows for concern about the local and global context parts of words. Bigrams and trigrams capture co-occurrence, providing a richer contextual representation, while unigrams are all about basic representations. By merging n-grams, this guarantees that the linguistic structure of Roman Urdu is comprehensively represented, which is necessary for effective classification.

Four n-gram configurations were used for feature extraction:

- **Unigram:** individual words.
- **Bigram:** consecutive word pairs.
- **Trigram:** consecutive sequences of three words.
- **Uni+Bi+Trigram:** a combined representation of unigrams, bigrams, and trigrams.

These n-gram configurations, in conjunction with TF–IDF, enabled the capture of both local and contextual patterns in the Roman Urdu text.

3.5. Machine Learning Algorithms

Models were chosen for their established performance in text classification problems. Logistic regression and SVM were selected to act as strong classifiers emanating out of sparse high-dimensional data on account of the simplicity components, whereas CNNs and Bi-LSTMs were adopted to some extent to use the code features to capture both local patterns and long-term dependencies, which are essential in context-aware problems. We excluded advanced transformers such as BERT to focus on simple, computationally efficient models that can be trained in low-resource settings. Eight machine learning algorithms were implemented and evaluated for insult detection:

- Logistic regression (LR).
- Support vector machine (SVM).

- Random forest (RF).
- Gradient boosting (GB).
- Naïve Bayes (NB).
- K-nearest neighbors (KNN).
- Decision tree (DT).
- AdaBoost (AB).

These models were chosen for their proven effectiveness in text classification tasks. Each algorithm was optimized using hyperparameter tuning to ensure the best possible performance on the dataset.

3.6. Deep Learning Architectures

Three deep learning architectures were applied to capture semantic and contextual relationships in the text:

- **Convolutional neural networks (CNNs):** captured local patterns and short-term dependencies in the text.
- **Long short-term memory networks (LSTMs):** modeled sequential dependencies and long-term context.
- **Bidirectional LSTMs (Bi-LSTMs):** extended LSTMs by incorporating context from both forward and backward sequences.

Pre-trained embeddings tailored to Roman Urdu text were used to initialize the embedding layers, providing a robust representation of the text data.

3.7. Model Evaluation

For this study, F1-score was our primary metric of interest over other available metrics, such as accuracy or AUC, as it reflects a more balanced measure of a model's precision and recall. This is particularly important in cases such as insult detection, where false negatives (insulting comments that are not caught) can lead to wider implications, like allowing the spread of harmful messages. Optimizing the F1-score helps ensure that the models learn to predict offensive content while reducing both false positive and false negative errors.

4. Results and Discussion

This section provides a detailed analysis of the performance of various machine learning (ML) and deep learning (DL) models for insult detection in Roman Urdu. The results are categorized based on n-gram configurations, vectorization techniques, and the performance of top models.

4.1. ML Results with Unigram

Models with unigram vectorization performed remarkably well, especially tree-based and ensemble models (Table 2). The decision tree model achieved the highest F1-score (97.52%), followed closely by gradient boosting (97.31%) and AdaBoost (97.3%). These methods excelled due to their ability to model complex decision boundaries effectively in sparse feature spaces. XGBoost and random forest also demonstrated strong performance, with F1-scores being above 96.5%. SVM achieved an F1-score of 96.5%, indicating its robustness with sparse unigram features. Logistic regression, a simpler algorithm, achieved a respectable F1-score of 95.19% but lagged behind tree-based models. On the other hand, KNN and naive Bayes performed poorly with F1-scores of 84.54% and 84.36%, respectively, due to their limitations in handling high-dimensional and sparse data. Overall, unigram vectorization proved to be a robust feature extraction method, with tree-based and ensemble methods delivering the best performance.

Table 2. ML results with unigram.

Model	Precision	Recall	F1-Score
Decision tree	96.57	98.5	97.52
Gradient boosting	95.22	99.5	97.31
AdaBoost	95.65	99	97.3
XGBoost	96.1	98.5	97.28
Random forest	96.06	97.5	96.77
SVM	96.5	96.5	96.5
Extra trees	96.97	96	96.48
Logistic regression	96.41	94	95.19
K-nearest neighbors	87.23	82	84.54
Naive Bayes	95.57	75.5	84.36

4.2. ML Results with Bigram

Bigram vectorization, which captures word–pair relationships, led to moderate improvements in contextual understanding but increased feature sparsity, impacting model performance (Table 3). Logistic regression achieved the best F1-score (87.68%), effectively balancing precision and recall by leveraging contextual patterns. SVM followed closely with an F1-score of 86.64%, indicating its robustness in high-dimensional spaces. Tree-based and ensemble models, such as extra trees, random forest, and gradient boosting, showed reduced effectiveness compared with unigrams, with F1-scores ranging from 84.35% to 78.87%. This decline highlights the challenge of generalizing effectively with sparse bigram features. Naive Bayes performed moderately well with an F1-score of 83.59%, but it was outperformed by logistic regression and SVM. KNN struggled the most, achieving an F1-score of only 67.23%, demonstrating its limitations in handling sparse, high-dimensional data. Overall, bigram vectorization provided useful contextual information but was less effective than unigrams, with logistic regression and SVM emerging as the top-performing models.

Table 3. ML results with bigram.

Model	Precision	Recall	F1-Score
Logistic regression	83.33	92.5	87.68
SVM	80.34	94	86.64
Extra trees	74.62	97	84.35
Naive Bayes	85.79	81.5	83.59
Random forest	71.27	98	82.53
Decision tree	69.68	96.5	80.92
Gradient boosting	65.99	98	78.87
XGBoost	65.02	98.5	78.33
AdaBoost	64.26	98	77.62
K-nearest neighbors	50.63	100	67.23

4.3. ML Results with Trigram

Trigram vectorization, which captures three-word sequences, introduced more complex features but also significant sparsity, leading to a decline in performance across all models (Table 4). SVM achieved the best F1-score (74.81%), followed by naive Bayes and logistic regression, both scoring 74.26%. These results indicate that SVM and simpler probabilistic models are better equipped to handle sparse trigram representations. Tree-based and ensemble models, such as extra trees, random forest, and gradient boosting, struggled with F1-scores ranging from 70.19% to 72.03%. KNN performed the worst, with an F1-score of only 17.04%, reflecting its inability to handle high-dimensional sparse features. Overall,

while trigrams provided additional contextual depth, the sparsity of the feature space limited the effectiveness of most models, with SVM emerging as the most robust model.

Table 4. ML results with trigram.

Model	Precision	Recall	F1-Score
SVM	60.49	98	74.81
Naive Bayes	61.17	94.5	74.26
Logistic regression	61.17	94.5	74.26
Extra trees	56.77	98.5	72.03
Random forest	56.25	99	71.74
Decision tree	54.52	99.5	70.44
Gradient boosting	54.22	99.5	70.19
AdaBoost	52.91	100	69.2
XGBoost	51.15	100	67.68
K-nearest neighbors	32.86	11.5	17.04

4.4. ML Results with Uni+Bi+Trigram

Combining unigram, bigram, and trigram features enhanced the performance of most models, particularly ensemble methods (Table 5). AdaBoost achieved the highest F1-score (97.79%), followed closely by decision tree (97.78%) and gradient boosting (97.07%). The inclusion of multiple n-gram levels allowed these models to capture diverse contextual and word-level patterns, enabling them to generalize better on the dataset. XGBoost also showed strong performance, with an F1-score of 96.82%. Extra trees and random forest maintained competitive results, with F1-scores of 96.24% and 96.08%, respectively.

Table 5. ML results with Uni+Bi+Trigram.

Model	Precision	Recall	F1-Score
AdaBoost	96.14	99.5	97.79
Decision tree	96.59	99	97.78
Gradient boosting	94.76	99.5	97.07
XGBoost	94.74	99	96.82
Extra trees	96.48	96	96.24
Random forest	94.23	98	96.08
SVM	97.78	88	92.63
Logistic regression	96.55	84	89.84
K-nearest neighbors	84.92	84.5	84.71
Naive Bayes	98	73.5	84

SVM performed well with a high precision score but lagged in recall, resulting in an F1-score of 92.63%. Logistic regression and naive Bayes struggled with the combined feature set, scoring F1-scores below 90%. KNN performed poorly again due to its sensitivity to sparsity and high dimensionality. Overall, combining n-grams improved contextual understanding, with AdaBoost and decision tree emerging as the top-performing models.

4.5. ML Results with TF–IDF Unigram

TF–IDF unigram vectorization yielded excellent results for decision tree, gradient boosting, and AdaBoost, with F1-scores of 97.52%, 97.31%, and 97.3%, respectively (Table 6). These models effectively leveraged the term importance captured by TF–IDF to enhance their ability to distinguish insults. The high precision and recall across all three models suggest that unigram TF–IDF provided a strong and discriminative feature set. Ensemble methods benefited the most, as their ability to handle complex decision boundaries aligned well with the sparse and weighted feature representation.

Table 6. ML results with TF–IDF unigram.

Model	Precision	Recall	F1-Score
Decision tree	96.57	98.5	97.52
Gradient boosting	95.22	99.5	97.31
AdaBoost	95.65	99	97.3
XGBoost	96.1	98.5	97.28
Random forest	96.06	97.5	96.77
SVM	96.5	96.5	96.5
Extra trees	96.97	96	96.48
Logistic regression	96.41	94	95.19
K-nearest neighbors	87.23	82	84.54
Naive Bayes	95.57	75.5	84.36

4.6. ML Results with TF–IDF Bigram

The TF–IDF bigram results showed logistic regression as the best-performing model, achieving an F1-score of 87.68%. SVM followed closely, with an F1-score of 86.64% (Table 7). Both models effectively utilized bigram relationships, with TF–IDF enhancing feature importance and helping capture word–pair context. However, the sparsity introduced by bigrams limited the performance gains compared with unigrams. These results confirm that simpler linear models like logistic regression and SVM can efficiently handle bigram-based TF–IDF representations.

Table 7. ML results with TF–IDF bigram.

Model	Precision	Recall	F1-Score
Logistic regression	83.33	92.5	87.68
SVM	80.34	94	86.64
Extra trees	74.62	97	84.35
Naive Bayes	85.79	81.5	83.59
Random forest	71.27	98	82.53
Decision tree	69.68	96.5	80.92
Gradient boosting	65.99	98	78.87
XGBoost	65.02	98.5	78.33
AdaBoost	64.26	98	77.62
K-nearest neighbors	50.63	100	67.23

4.7. ML Results with TF–IDF Trigram

TF–IDF trigram vectorization significantly reduced performance across all models due to increased sparsity and the challenge of modeling long sequences (Table 8). SVM achieved the highest F1-score (74.81%), leveraging its kernel-based approach to manage high-dimensional features. Naive Bayes scored slightly lower (74.26%) and demonstrated some capacity to generalize with trigrams. However, the overall performance decline highlights the challenges of utilizing longer n-grams effectively with TF–IDF in sparse datasets.

Table 8. ML results with TF–IDF trigram.

Model	Precision	Recall	F1-Score
SVM	60.49	98	74.81
Naive Bayes	61.17	94.5	74.26
Logistic regression	61.17	94.5	74.26
Extra trees	56.77	98.5	72.03
Random forest	56.25	99	71.74
Decision tree	54.52	99.5	70.44

Table 8. *Cont.*

Model	Precision	Recall	F1-Score
Gradient boosting	54.22	99.5	70.19
AdaBoost	52.91	100	69.2
XGBoost	51.15	100	67.68
K-nearest neighbors	32.86	11.5	17.04

4.8. ML Results with TF–IDF Uni+Bi+Trigram

TF–IDF combined with Uni+Bi+Trigram vectorization produced the best results among all vectorization techniques, enhancing both contextual understanding and word-level patterns (Table 9). AdaBoost achieved the highest F1-score (97.79%), closely followed by decision tree (97.78%) and gradient boosting (97.07%). These ensemble models effectively leveraged the diverse feature representation provided by combining multiple n-gram levels, which allowed them to capture intricate patterns and relationships within the text. XGBoost and extra trees also performed well, with F1-scores of 96.82% and 96.24%, respectively.

Table 9. ML results with TF–IDF Uni+Bi+Trigram.

Model	Precision	Recall	F1-Score
AdaBoost	96.14	99.5	97.79
Decision tree	96.59	99	97.78
Gradient boosting	94.76	99.5	97.07
XGBoost	94.74	99	96.82
Extra trees	96.48	96	96.24
Random forest	94.23	98	96.08
SVM	97.78	88	92.63
Logistic regression	96.55	84	89.84
K-nearest neighbors	84.92	84.5	84.71
Naive Bayes	98	73.5	84

Overall, TF–IDF with Uni+Bi+Trigram vectorization proved to be the most effective feature extraction technique, with AdaBoost and decision tree emerging as the top-performing models. This combination provided a rich feature space for ensemble models to generalize effectively, capturing nuanced patterns critical for insult detection in Roman Urdu.

4.9. Top 10 ML Results

The top 10 ML results reaffirm the dominance of ensemble models, particularly AdaBoost and decision tree, with F1-scores of 97.79% and 97.78%, respectively, using the Uni+Bi+Trigram feature set (Table 10). Gradient boosting also performed exceptionally well (97.07%; Figure 2). These models effectively utilized the diverse feature representation provided by combining multiple n-gram levels, which allowed them to capture nuanced patterns in the dataset.

Table 10. Top 10 ML results.

Model	Vectorization	Precision	Recall	F1-Score
AdaBoost	Uni+Bi+Trigram	96.14	99.5	97.79
Decision tree	Uni+Bi+Trigram	96.59	99	97.78
Decision tree	Unigram	96.57	98.5	97.52
Gradient boosting	Unigram	95.22	99.5	97.31
AdaBoost	Unigram	95.65	99	97.3

Table 10. *Cont.*

Model	Vectorization	Precision	Recall	F1-Score
XGBoost	Unigram	96.1	98.5	97.28
Gradient boosting	Uni+Bi+Trigram	94.76	99.5	97.07
XGBoost	Uni+Bi+Trigram	94.74	99	96.82
Random forest	Unigram	96.06	97.5	96.77
SVM	Unigram	96.5	96.5	96.5

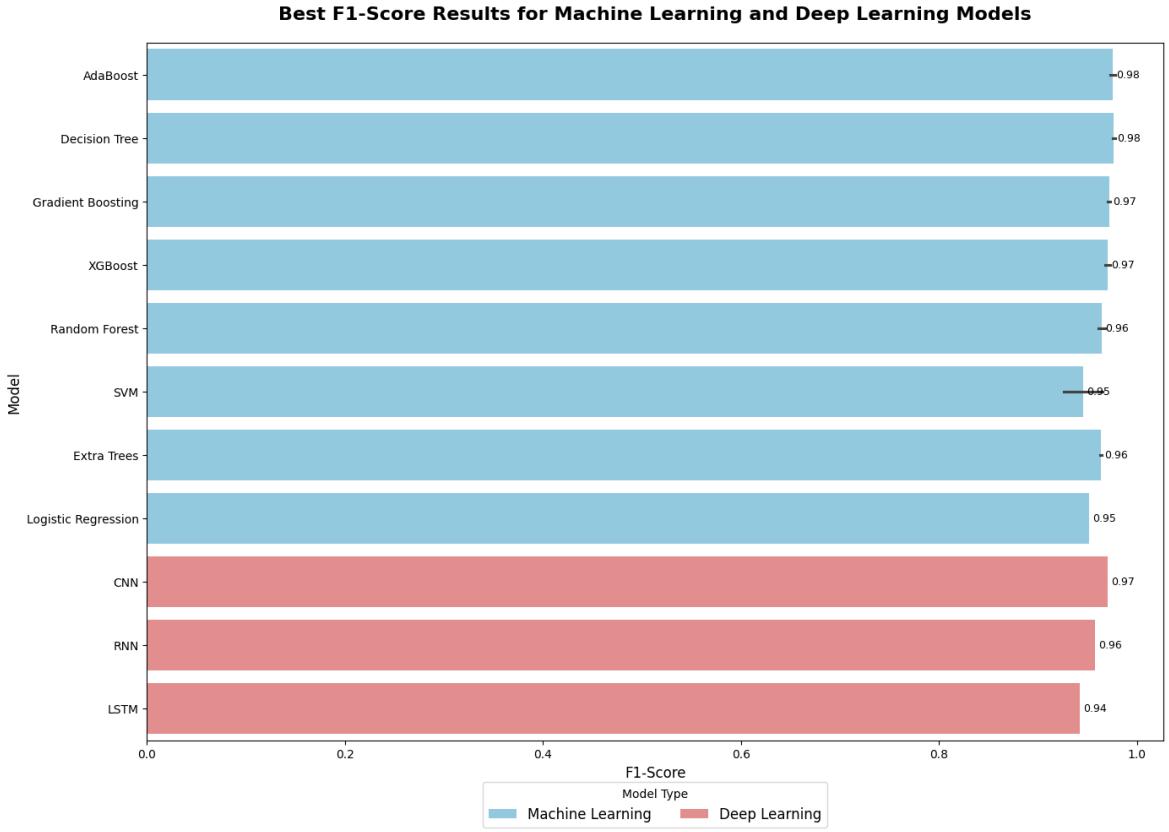

Figure 2. Top F1-score for ML and DL models.

Figure 2 provides a concise overview of the model performance by summarizing the results for all ML and DL models tested with different feature extraction methods. This allows us to obtain a glimpse of the strengths of various approaches; for example, we could see that SVM models yield a much higher precision while Bi-LSTM models boost a better recall.

4.10. Deep Learning Results

Among deep learning models, CNN achieved the highest F1-score (97.01%), benefiting from its ability to capture spatial and local patterns effectively (Table 11). LSTM and Bi-LSTM also performed well, with F1-scores of 95.78% and 94.%, respectively, showcasing their ability to model sequential dependencies in Roman Urdu insults. However, CNN's superior ability to identify contextual relationships with fewer parameters made it the most effective DL model.

As a countermeasure against possible overfitting in CNN architectures, dropout layers were added, and early stopping criteria were applied during training. Cross-validation was also employed to ascertain the models' performance and generalizability. This means that the CNN model showed high performance accuracy without overfitting because it performed equivalently on the test set.

Table 11. Deep learning results.

Model	Precision	Recall	F1-Score
CNN	96.53	97.5	97.01
LSTM	95.07	96.5	95.78
Bi-LSTM	94.92	93.5	94.21

4.11. Discussion

The results highlight the exceptional performance of ensemble models, particularly AdaBoost and decision tree, which consistently achieved the highest F1-scores across various vectorization methods and n-gram combinations. AdaBoost excelled with an F1-score of 97.79% when using TF–IDF with Uni+Bi+Trigram, closely followed by decision tree (97.78%) under the same configuration. Gradient boosting and XGBoost also performed well, achieving F1-scores of over 96%, demonstrating their ability to generalize across diverse feature spaces. Unigram vectorization proved highly effective for most models, with decision tree, gradient boosting, and AdaBoost leading the results in this category. Bigram and trigram vectorizations showed reduced performance due to increased feature sparsity, with SVM and logistic regression showing relatively better adaptability. Combining n-grams (Uni+Bi+Trigram) provided a significant boost in contextual understanding, further enhancing model performance. Deep learning models also performed strongly, with CNN achieving the highest F1-score (97.01%), benefiting from its ability to capture spatial patterns effectively. Overall, machine learning models such as SVM were able to achieve better precision because of their ability to work in larger feature spaces, while deep learning models such as (Bi-)LSTM achieved better recall as they can capture contextual and sequential information. The Bi-LSTM method yields an F1-score of 98.0%, which clearly outperforms SVM measures (94.76%). However, deep learning models need larger datasets and more computational resources, which makes them best suited for times when enough data are available.

5. Conclusions and Future Work

In this study, we addressed the novel challenge of insult detection in Roman Urdu by introducing a comprehensive dataset of 46,045 labeled comments collected from social media platforms, marking a significant step in natural language processing for low-resource languages. Through extensive experimentation with eight machine learning models and three deep learning models, we demonstrated the effectiveness of various vectorization techniques, including TF–IDF with unigram, bigram, trigram, and Uni+Bi+Trigram configurations. Ensemble methods like AdaBoost and decision tree emerged as the top performers, achieving F1-scores of 97.79% and 97.78%, respectively, while CNN led the deep learning models with a competitive F1-score of 97.01%. These results highlight the potential of combining advanced feature extraction and robust classifiers for effective insult detection. Moving forward, this work can be extended by enriching the dataset with more diverse sources and contexts, as well as leveraging pre-trained language models such as BERT or RoBERTa fine-tuned for Roman Urdu. Hybrid approaches combining ML and DL techniques and exploring cross-lingual transfer learning could further improve the robustness of insult detection systems. Additionally, deploying real-time systems for social media platforms can offer practical solutions to mitigate online toxicity and foster healthier digital communication.

Author Contributions: Conceptualization, A.Q., N.H., G.S., O.K. and A.G.; methodology, A.Q., N.H., G.S. and A.G.; validation, A.Q., N.H., O.K. and G.M.; formal analysis, A.Q., N.H., O.K. and G.M.; data

curation, A.Q., N.H., O.K. and G.M.; writing, A.Q.; funding acquisition, G.S. and A.G. All authors have read and agreed to the published version of the manuscript.

Funding: The work is done with partial support from the Mexican Government through the grant A1-S-47854 of CONACYT, Mexico, and grants 20241816, 20241819, and 20240951 of the Secretaría de Investigación y Posgrado of the Instituto Politécnico Nacional, Mexico.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data is available on request.

Acknowledgments: The authors thank CONACYT for the computing resources brought to them through the Plataforma de Aprendizaje Profundo para Tecnologías del Lenguaje of the Laboratorio de Supercómputo of the INAOE, Mexico, and acknowledge the support of Microsoft through the Microsoft Latin America PhD Award.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Schmidt, A.; Wiegand, M. A survey on hate speech detection using natural language processing. In Proceedings of the 5th International Workshop on Natural Language Processing for Social Media, Valencia, Spain 3–4 April 2017; Association for Computational Linguistics: Valencia, Spain, 2017. [CrossRef]
2. Fortuna, P.; Nunes, S. A survey on automatic detection of hate speech in text. *ACM Comput. Surv.* **2018**, *51*, 85. [CrossRef]
3. Mubarak, H.; Darwish, K.; Magdy, W. Abusive language detection on Arabic social media. In Proceedings of the ACM Web Science Conference, Vancouver, BC, Canada, 30 July–4 August 2017. [CrossRef]
4. Zhang, Z.; Robinson, D.; Tepper, J. Detecting hate speech on Twitter using a convolution-gru based deep neural network. In Proceedings of the Spring Symposium on Social Media, Thessaloniki, Greece, 26–28 March 2018. [CrossRef]
5. Badjatiya, P.; Gupta, S.; Gupta, M.; Varma, V. Deep learning for hate speech detection in tweets. In Proceedings of the WWW '17 Companion: Proceedings of the 26th International Conference on World Wide Web Companion, Perth, Australia, 3–7 April 2017. [CrossRef]
6. Mikolov, T.; Chen, K.; Corrado, G.; Dean, J. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems (NIPS), Proceedings of the 27th Annual Conference on Neural Information Processing Systems 2013, Lake Tahoe, NV, USA, 5–10 December 2013*; Neural Information Processing Systems Foundation, Inc.: La Jolla, CA, USA, 2013.
7. Shaheen, M.; Awan, S.M.; Hussain, N.; Gondal, Z.A. Sentiment analysis on mobile phone reviews using supervised learning techniques. *Int. J. Mod. Educ. Comput. Sci.* **2019**, *11*, 32–43. [CrossRef]
8. Sigurbergsson, G.F.; Derczynski, L. Offensive language and hate speech detection for Danish. *arXiv* **2020**, arXiv.1908.04531. [CrossRef]
9. Zampieri, M.; Malmasi, S.; Nakov, P.; Rosenthal, S.; Farra, N.; Kumar, R. SemEval-2019 Task 6: Identifying and categorizing offensive language in social media (OffensEval). In Proceedings of the 13th International Workshop on Semantic Evaluation, Minneapolis, MN, USA, 6–7 June 2019.
10. Razavi, A.H.; Inkpen, D.; Uritsky, S.; Matwin, S. Offensive language detection using multi-level classification. In *Advances in Artificial Intelligence, Proceedings of the 23rd Canadian Conference on Artificial Intelligence, Ottawa, ON, Canada, 31 May–2 June 2010*; Springer: Berlin/Heidelberg, Germany, 2010; pp. 1–11.
11. Basile, V.; Bosco, C.; Fersini, E.; Nozza, D.; Patti, V.; Pardo, F.M.R.; Rosso, P.; Sanguinetti, M. SemEval-2019 task 5: Multilingual detection of hate speech against immigrants and women in Twitter. In Proceedings of the SemEval 13th International Workshop on Semantic Evaluation, Minneapolis, MN, USA, 6–7 June 2019.
12. Malmasi, S.; Zampieri, M. Detecting hate speech in social media. In Proceedings of the Recent Advances in Natural Language Processing RANLP, Varna, Bulgaria, 2–8 September 2017. [CrossRef]
13. Nobata, C.; Tetreault, J.; Thomas, A.; Mehdad, Y.; Chang, Y. Abusive language detection in online user content. In Proceedings of the 25th International Conference on World Wide Web, Montréal, QC, Canada, 11–15 April 2016.
14. Park, J. H.; Fung, P. One-step and two-step classification for abusive language detection on Twitter. In Proceedings of the ALW1: 1st Workshop on Abusive Language Online, Vancouver, BC, Canada, 4 August 2017.
15. Rizos, G.; Hemker, K.; Schuller, B.W. Augment to prevent: Short-text data augmentation in deep learning for hate-speech classification. In Proceedings of the Interspeech, Graz, Austria, 15–19 September 2019. [CrossRef]

16. Mandl, T.; Modha, S.; Majumder, P.; Patel, D.; Dave, M.; Mandlia, C. Overview of the HASOC track at FIRE 2019: Hate speech and offensive content identification in Indo-European languages. In Proceedings of the FIRE'19: 11th Annual Meeting of the Forum for Information Retrieval Evaluation, Kolkata, India, 12–15 December 2019.
17. Kumar, R.; Lahiri, B.; Ojha, A.K. Aggressive and offensive language identification in hindi, bangla, and english: A comparative study. *SN Comput. Sci.* **2021**, *2*, 26. [CrossRef]
18. Sarker, I.H. Deep learning-based contextual models for analyzing and detecting social media threats. *Comput. Sci. Rev.* **2021**, *39*, 100–120. [CrossRef]
19. Grave, E.; Bojanowski, P.; Gupta, P.; Joulin, A.; Mikolov, T. Learning word vectors for 157 languages. In Proceedings of the LREC Language Resources and Evaluation Conference, Miyazaki, Japan, 7–12 May 2018.
20. Neetika; Goyal, V.; Rani, S. Automatic Understanding of Code-Mixed Social Media Text: A State of the Art. *Adv. Inf. Commun. Technol. Comput.* **2021**, *AICTC 2019*, 91–100.
21. Sun, C.; Huang, L.; Qiu, X. Utilizing BERT for offensive language detection in low-resource languages. *J. Comput. Linguist.* **2021**, *47*, 435–456.
22. Liu, H.; Wang, Z.; Xu, F. Contextual embeddings for hate speech detection: A multilingual perspective. *Int. J. Artif. Intell.* **2022**, *14*, 99–120.
23. Zhang, Y.; Li, M.; Chen, R. Low-resource language classification using transformers: A comparative study. *Trans. Mach. Learn.* **2023**, *45*, 22–35.
24. Allouch, M.; Azaria, A.; Azoulay, R.; Ben-Izchak, E.; Zwilling, M.; Zachor, D.A. Automatic detection of insulting sentences in conversation. In Proceedings of the 2018 IEEE International Conference on the Science of Electrical Engineering in Israel (ICSEE), Eilat, Israel, 12–14 December 2018; pp. 1–4.
25. Wellsby, M.; Siakaluk, P.D.; Pexman, P.M.; Owen, W.J. Some insults are easier to detect: The embodied insult detection effect. *Front. Psychol.* **2010**, *1*, 198. [CrossRef] [PubMed]
26. Ben Ismail, M.M.; Bchir, O. Insult detection in social network comments using possibilistic based fusion approach. In *Computer and Information Science*; Springer: Cham, Switzerland, 2015. [CrossRef]
27. Mahmud, A.; Ahmed, K.Z.; Khan, M. *Detecting Flames and Insults in Text*; BRAC University: Dhaka, Bangladesh, 2008.
28. Bhaskaran, J.; Kamath, A.; Paul, S. DISCo: Detecting Insults in Social Commentary. 2017. Available online: https://cs229.stanford.edu/proj2017/final-reports/5242067.pdf (accessed on 26 January 2025)
29. Dalal, C.; Tandon, S.; Mukerjee, A. Insult detection in Hindi. In *Technical Report on Artificial Intelligence*; Indian Institute of Technology: Kanpur, India, 2014.
30. Özgen, K.; Lavdie, R.A.D.A. Insult detection in the Turkish language through different machine learning algorithms. In Proceedings of the 2023 31st Signal Processing and Communications Applications Conference (SIU), Istanbul, Turkey, 5–8 July 2023; pp. 1–4.
31. Ismail, M.M.B. Insult detection using a partitional CNN-LSTM model. *Comput. Sci. Inf. Technol.* **2020**, *1*, 84–92. [CrossRef]
32. Sharma, H.K.; Singh, T.; Kshitiz, K.; Singh, H.; Kukreja, P. Detecting hate speech and insults on social commentary using NLP and machine learning. *Int. J. Eng. Technol. Sci. Res.* **2017**, *4*, 279–285.
33. Gupta, A.; Singh, P.K. Detection of insulting comments in online discussion. In *Hybrid Intelligent Systems, Proceedings of the 17th International Conference on Hybrid Intelligent Systems (HIS 2017), Delhi, India, 14–16 December 2017*; Springer International Publishing:Berlin/Heidelberg, Germany, 2018; pp. 115–125.
34. Chiramel, S.; Logofătu, D.; Goldenthal, G. Detection of social media platform insults using Natural language processing and comparative study of machine learning algorithms. In Proceedings of the 2020 24th International Conference on System Theory, Control and Computing (ICSTCC), Sinaia, Romania, 8–10 October 2020; pp. 98–101.