AI

MDPI

Article

Urdu Toxicity Detection: A Multi-Stage and Multi-Label Classification Approach

Ayesha Rashid [1], Sajid Mahmood [1,*], Usman Inayat [1] and Muhammad Fahad Zia [2]

[1] School of Systems and Technology, University of Management and Technology, Lahore 54770, Pakistan; f2019288010@umt.edu.pk (A.R.); usman.inayat@umt.edu.pk (U.I.)
[2] Department of Electrical and Computer Engineering, American University in Dubai, Dubai P.O. Box 28282, United Arab Emirates; mfz@ieee.org
* Correspondence: sajid.mahmood@umt.edu.pk

Abstract

Social media empowers freedom of expression but is often misused for abuse and hate. The detection of such content is crucial, especially in under-resourced languages like Urdu. To address this challenge, this paper designed a comprehensive multilabel dataset, the Urdu toxicity corpus (UTC). Second, the Urdu toxicity detection model is developed, which detects toxic content from an Urdu dataset presented in Nastaliq Font. The proposed framework initially processed the gathered data and then applied feature engineering using term frequency-inverse document frequency, bag-of-words, and N-gram techniques. Subsequently, the synthetic minority over-sampling technique is used to address the data imbalance problem, and manual data annotation is performed to ensure label accuracy. Four machine learning models, namely logistic regression, support vector machine, random forest, and gradient boosting, are applied to preprocessed data. The results indicate that the RF outperformed all evaluation metrics. Deep learning algorithms, including long short-term memory (LSTM), Bidirectional LSTM, and gated recurrent unit, have also been applied to UTC for classification purposes. Random forest outperforms the other models, achieving a precision, recall, F1-score, and accuracy of 0.97, 0.99, 0.98, and 0.99, respectively. The proposed model demonstrates a strong potential to detect rude, offensive, abusive, and hate speech content from user comments in Urdu Nastaliq.

Keywords: Urdu Nastaliq; toxicity; machine learning; deep learning; natural language processing; sentiment analysis

Academic Editors: Mohammed Abdulhakim Al-Absi, Ahmed A. Abdulhakim Al-Absi and Nadhem Ebrahim

Received: 24 June 2025
Revised: 13 August 2025
Accepted: 20 August 2025
Published: 21 August 2025

Citation: Rashid, A.; Mahmood, S.; Inayat, U.; Zia, M.F. Urdu Toxicity Detection: A Multi-Stage and Multi-Label Classification Approach. *AI* **2025**, *6*, 194. https://doi.org/10.3390/ai6080194

1. Introduction

Global social media platforms are the primary focal point for communication among individuals worldwide. Social media usage has also consistently increased. The ease of access and similarity of social media platforms make it simpler for irresponsible users to engage in unethical activities such as spreading hatred, using abusive language, and defamation. Geographical, religious, ethnic, and cultural differences, such as the partition of the Indian Subcontinent into India and Pakistan, can lead to individuals who are apart from one another resorting to derogatory communication [1,2]. This can culminate in online harassment and the dissemination of prejudice through hurtful and insulting language. It is crucial to create an automated system capable of identifying, preventing, or restricting the posting of toxic language before it is published on the Internet [3,4].

In this context, "toxic" comments involve derogatory language, including hateful or abusive words, threats, insults, or any communication that specifically targets an individual's gender, ethnicity, faith, or personal convictions. Such negative statements are also considered hate speech, cyberbullying, abusive language, and profanity. Consequently, toxicity has been on the rise in social media over the past few years [5,6]. The harm caused by toxicity to its victims is obvious, as they are frequently targeted in personal attacks on their physical appearances, social norms, religious values, and individual opinions [7]. Social media platforms are facing public criticism for addressing the issue of toxic content. Although these platforms have introduced multiple approaches, such as reporting the content, blocking the user, unsubscribing from the channel, and unfollowing unwanted pages, these measures have proven insufficient in managing toxic traffic [8]. Failures are caused by the time-consuming manual effort required to label content as harmful or toxic. Research on detecting toxic text has been significant in English and other resource-rich languages [9]; however, Urdu, with its extensive user base, has been largely overlooked. Pakistan's national language is Urdu, which is written in the Persian-Arabic script and spoken by more than 170 million people worldwide [10]. Urdu in the Nastaliq Font is a combination of Sanskrit, Turkish, Persian, and Arabic, which leads to a challenge in accurately identifying emotions because of its complicated grammatical and sentence structure. According to the ethnologue.com (https://www.ethnologue.com/language/urdu (accessed on 20 August 2025), Urdu is spoken by over 170 million people globally and is frequently utilized on social media in the right-to-left Nastaliq script. However, due to its complicated morphological structure, it becomes a challenge to develop a tool that prevents users from posting toxic comments on social media [11–15].

Urdu poses several difficulties for machine and deep learning-based techniques, including its flexible word order and absence of capitalization. The inclusion of numerous languages makes Urdu text processing difficult. In natural language processing (NLP), languages are divided into two categories: resource-rich and resource-poor languages. English, Arabic, Chinese, Spanish, Italian, German, French, and Japanese are well-resourced languages. Adequately well-developed datasets, tokenizers, stemmers, and WordNet tools are available for these languages [10,16]. In contrast, Urdu, Roman Urdu, Punjabi, African languages, and many others are known as resource-poor languages in computational linguistics [17]. Urdu, due to its complicated syntax and morphology, faces a lack of NLP resources. Urdu's lexicon is influenced by Arabic, Persian, and regional languages, such as Hindi and Punjabi. Moreover, the frequent mixing of Urdu with English (code-switching) in written and spoken forms adds complexity to NLP tasks [18,19]. There are limited annotated datasets available, which are further inadequate in terms of domains and topics, restricting the generalizability of the computational model. Moreover, the core resources, Urdu Wordnet, pre-trained language models, and Urdu word embeddings, remain underdeveloped [20,21].

Urdu has received less attention from researchers and fewer contributions in terms of resource development. One of the main reasons for this is that research funding for Urdu language development is limited compared to that for English, Arabic, and Chinese. This lack of investment also restricts the development of Urdu-language resources [22,23]. This is the primary reason for the limited availability of Urdu datasets [9]. The lack of a standard reference dataset hinders the creation of new methodologies. It obstructs a comprehensive assessment of sophisticated multi-label text classification techniques, which have been implemented for other languages [24,25]. As the Urdu content on social media increases daily, it is necessary to develop a sophisticated and precise multi-label classification system for detecting Urdu text toxicity [10]. This study contributed to the development of such a system by constructing a multi-label, multi-class Urdu toxicity dataset. The dataset

comprises four data platforms. The first step is to scrape user comments from Facebook, YouTube, Urdu News, and Twitter (X) along five categories: sports, showbiz, politics, and food. The scraped data are refined to make them useful for further analysis. Many unnecessary data fields are removed to ensure the uniformity of data across all forums. A single file of the used platforms is created to apply data preprocessing. Subsequently, data annotation is performed (details are described in the Section 3). A multilabel multiclass classification is developed. In addition to being concerned with multilabel classification, a negative comment may contain multiple labels. A negative comment may be classified as a multilabel instance with negative, offensive, abusive, or hate speech labels. A social media comment can be classified into five classes: "positive", "negative", "negative and offensive", "negative, offensive, and abusive", and "negative, offensive, and hate speech". Moreover, this study examines how ML classifiers, such as support vector machine (SVM), random forest (RF), gradient boosting (GB), and logistic regression (LR), address the structural features of the Urdu language. In deep learning (DL) algorithms, long short-term memory (LSTM), Bidirectional LSTM (BiLSTM) and gated recurrent unit (GRU) are used. Comprehensive experiments and analysis are conducted to assess whether standard information remains consistent across different languages, irrespective of whether the techniques used are based on ML or DL.

The main contributions of this study are summarized as follows:

- Develop a comprehensive multi-label UTC dataset comprising 18,000 user-generated comments in Urdu Nastaliq font collected from multiple social media platforms. The dataset consists of five distinct classes: "positive", "negative", "negative and rude", "negative and offensive and abusive", and "negative and offensive and hate speech". To the best of our knowledge, this is the first multilabel dataset addressing a significant gap in natural language resources for Urdu.
- Propose a state-of-the-art Urdu Toxicity Detection Model (UTDM) framework for Urdu toxicity detection from user comments across various social media platforms. The proposed model framework includes both ML models (SVM, RF, LR, and GB) and DL models (LSTM, BiLSTM, and GRU). The framework is evaluated using precision, recall, accuracy, and F1 score, demonstrating its effectiveness in a multilabel multiclass toxicity classification task for Urdu text in Nastaliq script.

The remainder of this article is organized as follows: Section 2 describes the related work; Section 3 outlines the methodology, explaining the construction methodology followed to build the needed corpus for coarse- and fine-grained toxicity detection. Section 4 provides a detailed overview of the experimental design and results. Section 5 describes the research study, its discussion, and its limitations. Section 6 concludes the paper.

2. Related Work

The use of NLP techniques to identify problematic or unwanted texts has been the subject of several studies. One of the main purposes is to find offensive, divisive, abusive, toxic, or hateful content. Languages such as English, Chinese, Spanish, and Arabic, with developed and rich linguistic resources, including large-scale datasets, taggers, tokenizers, stemmers, and other well-established NLP tools, are known as rich-resource languages. The availability of large-scale corpora and sophisticated resources has enabled the development of ML and DL models for sentiment analysis, text classification, and toxicity detection. Text-based data augmentation is proposed to enhance the performance and explainability of DL models [9]. A human-centered approach is adopted in [26] to investigate whether human toxicity detection is aligned with the perspective toxicity score. An NLP-based ML model is developed to control hate speech [27]. Hate speech detection is conducted on a corpus of over one million comments on YouTube videos using an ML model trained

and fine-tuned on a large set of manually annotated data [28]. In [29], the lack of online hate classifiers for social media is addressed by collecting 197,566 comments from YouTube, Reddit, Wikipedia, and Twitter (X), with 80% labeled as non-hateful and 20% as hateful. In [24], the authors studied how to detect cyberbullying and toxic language and proposed new methods to improve the identification of online harassment and offensive language. Toxicity features are first used in [30] to detect cyberbullying on social media. Small language models have been successfully utilized to detect toxicity in online interactions, achieving an impressive average precision of 0.95 [31]. A novel approach is proposed in [32] to detect toxic comments using deep neural networks. A multilabel multiclass method is developed in [33] to classify six toxicity levels from Wikipedia's talk page using the Kaggle English dataset. To address the difficulties of dataset imbalance and the scarcity of labeled data, the authors of [34] presented a multi-task joint learning strategy from several datasets to capture external emotional features. The potential of ChatGPT for this purpose is analyzed by comparing its performance with MTurker annotations across three key concepts related to toxic content: hateful, offensive, and toxic [35]. A DL-based model is developed in [36] to detect toxicity from given text data in six classes: toxic, severely toxic, obscene, threats, insults, and identity hate. Modern DL algorithms have been used for multilabel toxic comment classification [37]. A cyberbullying detection system is developed in [38] for the Moroccan dialect using a dataset collected from Instagram. Deep neural networks are presented in [39] that utilize FastText word embeddings and regularization methods for multi-class hate speech detection in the Norwegian language. As is the case with resource-poor languages, including Urdu, Roman Urdu, and many other regional languages of different countries, adequate NLP resources are not available, which leads to a lack of research. Many studies on sentiment analysis are available for Urdu and Roman Urdu, such as [18–20,22,35,40–42], but Urdu toxicity detection lacks research and development in various NLP tasks. For Urdu and Roman Urdu, many studies have focused on sentiment analysis, such as lexicon-based and ML approaches for polarity detection [7,17,43]. However, research on toxicity detection in these languages is limited. The following discussion concerns the Roman Urdu and Urdu languages in toxicity detection.

2.1. Nastaliq Urdu Toxicity Detection

Only a few studies have been conducted on toxicity detection in Urdu Nastaliq. Abusive words are identified in [1] using ML and DL models. The results showed that RF outperformed other traditional ML models on both proposed and existing datasets. The CNN used Word2Vec and fastText embeddings with attention layers. The Bi-LSTM + attention model with custom word2vec embeddings outperformed other models in identifying abusive language across datasets. ML- and DL-based models are presented in [2] to identify cyberbullying in social media forums. They created an annotated dataset in Urdu. The experimental results showed that FastTex outperformed other models in the detection of cyberbullying.

To overcome the lack of annotated datasets for low-resource languages, the authors of [3] investigated the effectiveness of several ML methods for sarcasm detection in Urdu. The authors selected the Urdu sarcastic tweets dataset and made it public to address this issue. The results showed that SVM outperformed with an accuracy of 0.85. An intelligent framework, Urdu slang and abusive words detection, is developed in [10] based on lexicons for the classification of abusive and slang words in Urdu tweets. This work is equally important in that it created a dataset of abusive, offensive, and slang words in Urdu Nastaliq, as no such standardized dataset existed previously. The analysis indicated that 72.6% of the tweets are accurately classified as abusive and non-abusive.

An ML method is developed in [12] to detect hate content in the Urdu language. The dataset is compiled from existing repositories, including Urdu Sentiment, Urdu tweets, and IMDB, wherein the tweets are Google translated to Urdu, as well as personal data. From the test conditions, it is noted that the SVM could detect hate speech in Urdu with an accuracy of 81.87%. In [13], guidelines for two-level Urdu dataset annotations are provided specifically to identify hate speech content. The authors created a large dataset of 21,759 tweets. Experiments are conducted to compare the performance of eight supervised ML and DL techniques for the automated identification of hateful content. Initially, experiments are conducted to detect hateful content as a binary classification task, followed by the classification of Interfaith, Sectarian, and Ethnic hatred detection as a multiclass classification task. Overall, the bidirectional encoder representation from transformers (BERT) emerged as the most effective technique for identifying hate content in Urdu tweets.

A comprehensive stop-words list and a stemming dictionary are introduced in [17] to preprocess Urdu Text. The adopted approach reduced the input size of Urdu sentences and eliminated inconsistent data. A deep sequential model based on LSTM units is applied to the efficiently pre-processed data. The proposed methodology achieved an accuracy of 82%. An Urdu content detection model is developed in [44] for detecting content that leads to the spread of propaganda on Urdu News forums. The authors created a labeled dataset with 11,574 Urdu news items. In addition, to analyze the psycho-linguistic features of Urdu texts, they created a dictionary for the linguistic inquiry word count system. This study evaluated the effectiveness of multiple classifiers using different N-gram modifications, news landscape (NELA) characteristics, word2vec embeddings, and BERT. The results illustrate that the best classification accuracy of 0.91 for Urdu text is achieved using a combination of NELA, word N-grams, and character N-grams. In [45], the primary objectives include developing an extensive annotated corpus comprising both Urdu and Roman Urdu comments, and evaluating multiple classification techniques for toxic comment detection. The study aims to establish a benchmark F1-score for Urdu in the toxicity classification task using transfer learning, DL, ensemble methods, and multilingual large language models (LLMs). In [46], the development of a voting-based part-of-speech tagger is presented for the Urdu language, which addresses its unique computational challenges and improves tagging accuracy through various ML and DL techniques.

2.2. Roman Urdu Toxicity Detection

More studies focused on toxicity detection in Roman Urdu instead of Urdu Nastaliq. A text corpus of Roman Urdu tweets is developed to identify cyberbullying using a combination of the GRU model with the word2vec word embedding technique [7]. The proposed model applied ML algorithms to compare with GRU. The results show that the GRU-based approach achieved 97% accuracy, and the F1-measure with lexical normalization yielded the best results. The authors in [8] aimed to identify hate speech, offensive speech, and various forms of cyberbullying in the Roman Urdu language. The targeted categories for examining hate speech are religion, racism, and national origin. Furthermore, these categories are further subdivided according to the expressed intensity of hate symbolization, insult, and attribution. The study generated a corpus of over 20 K tweets that are categorized by the kind and severity level of hate speech. The top F-scores for identifying hate speech based on religion, racism, and national origin are 80%, 81%, and 72%, respectively, whereas the highest macro-averaged F-score for identifying offensive speech is 86%.

Fine-grained multi-class target audience identification and hate speech detection in Nastaliq Urdu are performed using a recently developed hate speech and target community corpus sourced from Pakistani Facebook posts [11]. Using transfer learning, this study enhances two transformer models specifically designed for Urdu: Urdu-RoBERTa and

Urdu-DistilBERT. The optimized DistilBERT model outperformed 16 baselines with 86.58% accuracy and 86.52% F1-score for binary classification, and 84.17% accuracy and 83.91% F1-score for identifying target communities based on gender, politics, and religion. Benchmark techniques are presented, and a sizable corpus is used to identify harmful comments in Roman Urdu [23]. More than 72,000 comments gathered from well-known social media sites and meticulously categorized with high inter-annotator agreement make up the produced corpus, Roman Urdu toxic. They used this dataset to train new DL models based on word embeddings and several classification models with bag-of-words (BoW) representations. The first benchmark for classifying harmful comments in Roman Urdu is established using the authors' ensemble approach, which obtained the highest F1-score of 86.35%.

A context-aware Roman Urdu hate speech detection model is proposed in [47] using Bi-LSTM with an attention layer. The model used proprietary Word2Vec embeddings to investigate how the lexical normalization of Roman Urdu words affects performance. The experimental results show that Bi-LSTM with an attention layer outperforms the other DL and conventional ML techniques, with an accuracy of 0.875 and an F1-score of 0.885. A list of hateful words is developed in Roman Urdu (RU) [48]. Furthermore, 10,012 tweets in RU with coarse- and fine-grained categories of hate speech and abusive language generated the annotated dataset known as RUHSOLD. The authors suggested CNN-gram for the detection of hate speech and offensive language, and investigated the viability of transfer learning utilizing five current embedding models for RU. The performance of CNN-gram is compared with seven existing baseline methods. The study also contributed by constructing domain-specific embeddings publicly available after training them on more than 4.7 million tweets. A DL predictor is proposed in [14,43] for developing advanced language modeling strategies to extract discriminative and semantic patterns. These patterns are used to train an attention-based classifier for emotion classification. The challenge of detecting abusive or offensive language in Roman Urdu, an unstandardized transliteration of Urdu commonly used on digital platforms, is examined in [49]. The primary objective is to evaluate and compare the performance of traditional ML, DL, and transformer-based models, including fine-tuned large LLMs, for offensive language detection. This work is significant for the development of automated moderation tools for low-resource, code-mixed languages, thereby contributing to safer online environments and advancing NLP techniques for low-resourced languages. Table 1 provides a comparison of the available datasets in both Roman Urdu and Urdu Nastaliq. The types of classes and classification models are provided. The developed UTC dataset is more comprehensive, as it utilizes data from various social media platforms and is a multilabel multiclass dataset, features that are not present in other datasets.

From the aforementioned studies, it is also clear that there is a significant lack of studies focusing on cyberbullying and hate speech detection in Urdu. Further research and attention are required in this area. Although classical ML techniques have been widely applied, DL approaches for this task are limited. This gap presents an opportunity to evaluate DL methods for addressing the challenges in Urdu toxicity detection, and a comprehensive Urdu toxicity dataset is needed by the research community. Research in this direction will not only contribute to a greater understanding of toxicity dynamics in Urdu-speaking communities but also help develop robust detection methods. In the final analysis, these efforts will eventually lead to a safer digital environment for Urdu-speaking individuals.

Table 1. A summative comparison analysis of previous studies in Urdu toxicity detection.

Ref.	Toxicity	Classification	Labels	Data Size	Classifier
[1]	Abusive	Binary	Abusive/non-abusive	12,082 tweets	BiLSTM
[2]	Cyberbullying	Multiclass	insult, offensive, name-calling, profane, threat, curse, or none	12,000 tweets	Naïve Bayes (NB), LR, LSTM, FastText
[3]	sarcasm	binary	Sarcastic/non-sarcastic.	19,955 tweets	SVM, Muti-NB
[7]	Cyberbullying	Multiclass	Neutral, Offensive, Religious Hate, Sexism, and Racism	36,000 tweets	GRU
[8]	Hate speech	Binary	symbolization, insult, and attribution	20,000 tweets	SVM, LR, CNN, LSTM, BERT
[10]	Abusive word	Binary	Abusive/non-abusive	5000 tweets	SVM, NB
[11]	Hate Speech Detection	Binary	Hate speech/non-hate speech	20,000 Facebook posts	Urdu-RoBERTa Urdu-DistilBERT
[12]	Hate speech	Binary	Hateful/normal tweets	16,000 tweets	SVM, Multinomial NB
[13]	Hate speech	Binary	Hateful/non-hateful	21,759 tweets	BERT, NB
[17]	Threatening	Binary	Threatening/non-Threatening	3564 tweets	LSTM
[23]	Toxic	Binary	Toxic/non-toxic	72,000 Twitter (X) and Facebook posts	NB, SVM, LR, RF, LSTM, CNN
[43]	Abusive	binary	Abusive/non-Abusive	196,226 Facebook posts	LR, SVM, SGD, RF
[44]	Propaganda spotting	Binary	Propaganda/non-propaganda	11,574 Urdu news articles	BERT
[45]	Toxicity detection	Binary	Toxic/non-toxic	72,771 labeled comments	Transfer learning
[47]	Hate speech	Binary	Hate speech, Neutral speech	30,000 Twitter (X) and Facebook posts	Bi-LSTM
[48]	Hate speech	Binary	Hate/non-hate	14,731 tweets	SVM
[14]	Threatening	Binary	Threatening/non-threatening	3564 tweets	LR, SVM, FastText
[49]	Offensive language detection	Binary	Offensive/non-offensive	46,025 YouTube New channels comments	NB, SVM, LR, RF, LSTM, CNN
This Paper	Toxic	Multilabel Multiclass	Multiple labels	18,000 Facebook, Twitter (X), YouTube, Newsgroup	ML & DL

3. Methodology

This section describes the process of developing a toxicity detection model using Urdu-language datasets. It outlines the steps involved in preparing the data, selecting appropriate algorithms, and training the model to classify different types of toxic comments.

The workflow, illustrated in Figure 1, provides a clear overview of the methodology used to ensure accurate and efficient toxicity detection tailored to the Urdu language. It includes the first data processing, annotation, and resampling stages. The processed data are then used for feature extraction using term frequency-inverse document frequency (TF-IDF), BoW, and N-gram methods, and then classifiers are used to detect toxicity in the Urdu Nastaliq comments.

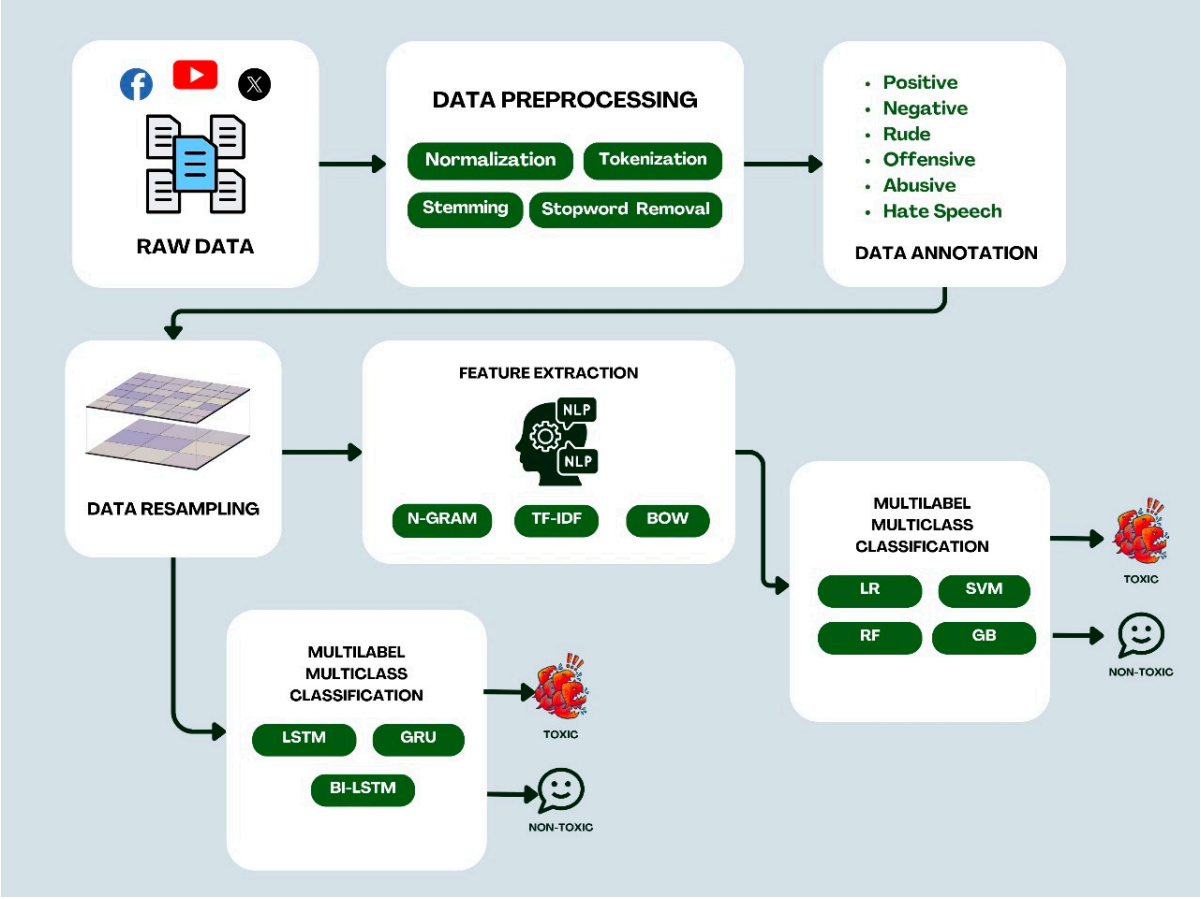

Figure 1. Workflow of the UTDM framework.

3.1. Data Collection

The first step is to scrape user reviews from four online social media forums: Facebook, Twitter (X), YouTube, and Urdu Newsgroups. ARY News, Geo News, and Urdu News forums are used to extract news-related data. The data consisted of five categories: Politics, Showbiz, Sports, News, and Food. The above-mentioned forums are highly used by the public, and people are used to commenting on posts to share their views on uploaded content. Most Urdu data instances are found on Twitter (X) and Facebook. Twitter (X) is a highly used platform by the public to tweet and retweet posts. Facebook is also a frequently used platform for user comments in Urdu. However, YouTube and Urdu News are less-used forums for Urdu comments. We found a smaller number of Urdu comments on YouTube than on Twitter (X). The complete statistics of the data are presented in Table 2. The total number of scraped user comments from all forums is 21,091.

Table 2. Statistics of Urdu toxic corpus data.

Social Media Forums	Category	No. of Comments
Facebook	Politics	1740
	Showbiz	1600
	Food	1506
	News	110
	Sports	216
Twitter (X)	Politics	4034
	Showbiz	2000
	Food	1506
	News	2501
	Sports	700
YouTube	Politics	1148
	Showbiz	16
	Food	149
	News	90
	Sports	1982
Urdu	Politics	249
	Showbiz	135
	Food	67
	News	560
	Sports	782
Total Comments		21,091

3.2. Data Preprocessing

Data preprocessing is crucial for developing a prediction model, especially for an Urdu dataset with varying formats. Data scraped from multiple forums is presented in different formats with different attributes. Data scraped from Facebook consists of "level-id", "parent-id", "object-id", "object-type", "query-time", "query-type", and "message". For further processing, only the "message" attribute is required. Similarly, data from other platforms also consists of many irrelevant data columns. Only relevant columns are extracted; for Twitter (X), only the "Text" attribute is retained; for YouTube, the "Comment" column; and for Facebook, the "Message" attribute. Data from Urdu news websites often lacked textual comments because most of the news platforms take reviews by using emojis or ratings. After extracting the relevant text columns, preprocessing involved tokenization, stemming or lemmatization, and converting the text into numerical representations, such as TF-IDF or word embeddings. This ensures data quality and consistency, thereby significantly enhancing the model performance. For this purpose, the data preprocessing steps are applied: (1) remove other language data; (2) remove numerical numbers; (3) remove punctuation marks; (4) remove repeating characters; (5) remove Urdu stop words; (6) tokenization; (7) stemming; and (8) data normalization.

Stop words carry less information in the dataset; therefore, the common practice is to remove these words while preprocessing the data for tasks such as text classification and clustering. In Urdu, stop words include frequently used words, such as pronouns,

prepositions, conjunctions, and some common verbs and adverbs. Here is a list of some common Urdu stop words.

كي، كى، كو، كا، هي، هں، هوں، تهى، تهي، تها، اىك، ہہ، وہ، مىں، تم ، ہم، آپ، اور، سي، پر، نہىں، كىا، جس، جو ، جسي، تاكہ، مىں، مىں ني، ہم ني، وہ، اس، ان، ان كى، اس كي، اس كي ساته، كيوں، كب، كىا، كسي، كون، كىا، كچھ، بہت، وغيرہ، كيونكہ، اگر، مگر، بہى، وہ، ان، ان كا، ان كي، اس كا، اسى، انہى، جب، تاكہ، جسي، جن، جنہىں، جنہوں، جو، جس، جن كي، جن كا، جس كا، تاكہ، لهذا، بعد، بعد مىں، پہلي، وہ، ہاں، وہاں، كس، كس كا، كس كي، كس كو، كيوں كہ، هي، كىا، پهر، تهي، رہىں، لي، والي، والى، وغيرہ، جبكہ، البتہ، جبكہ، باوجود، علاوہ، بغير، حالانكہ، حتى كہ، تاكہ، لي، تاہم، باوجود، البتہ

Emoji and UrduHack, along with NLTK, are used for implementing Urdu text preprocessing. NLTK's word_tokenize is suited for basic token separation as a basic rule-based tokenizer. For consistency issues related to the Nastaliq style of script normalization, urduhack.normalize() is used, which specifically takes care of Yeh (ىvs. ے), Heh (ہvs. ھ), and zero-width characters. Using the emoji library, non-Urdu emojis are removed via emoji library, and non-Urdu characters in \u0600–\u06FF (Urdu script unicode range) are filtered out using regex. Custom stopwords list for Urdu is applied alongside the removal of numbers, repetition of characters (e.g., "چلووووو" becoming "چلو"), and punctuation, which is removed through re.sub.

3.3. Remove Other Languages Data

The dataset required for this study is an Urdu dataset; therefore, it is necessary to clean the data by removing other languages. This helps improve the quality of preprocessing and model performance in tasks such as text classification, sentiment analysis, and language modeling. The types of non-Urdu content include English letters (e.g., A–Z, a–z), which might appear due to code-switching, and users are usually habitual to using mixed language in comments. Numerical digits (e.g., 0–9): Phone numbers, dates, or quantities that may not be relevant for NLP tasks focused on sentiment or semantics. Arabic words: The words are not required in the text; therefore, they must be removed. Arabic verses (such as Quranic references) may need to be removed depending on the task. Other scripts/symbols include emojis, special characters, or foreign scripts that do not belong to the Urdu language context. To clean this data efficiently, regular expressions (regex) in Python 3.11.7 can be used to identify and remove unwanted characters and words. This ensures that your dataset only contains relevant Urdu text, which in turn boosts the effectiveness of any downstream processing or ML tasks.

Mixed language comments are omitted during preprocessing and data annotation. The comments in the Urdu Nastaliq font are considered only. Comments in English, Roman Urdu, or Punjabi are not considered.

3.4. Remove Punctuation Marks

In text classification, dividing sections of written content is considered the least important task. Hence, it has become much more common and efficient to completely erase punctuation marks in any relevant datasets. The extracted dataset consisted of inverted commas, exclamation marks, and semicolons. In the case of text classification, these marks are non-essential and distracting. Removing punctuation marks cleans up the text, obscuring it with text manipulations due to marking, and makes it easier to work with in further data processes.

3.5. Remove Repetitive Words

Erasing repeating characters in Urdu texts follows an almost identical method to that in other languages [14,47]. Extra repeating characters can occur because of all sorts of reasons, including blunders in writing, adding stress, or casual ways of writing spellings. This step enhanced the data quality. This step is performed manually to remove repeating words by analyzing the context of the comment. In a sentence, repeated words also indicate the strength of the user's opinion. An example of repeated words is presented in Table 3. We implemented a preprocessing step that removes consecutive duplicate words using a regular expression or a custom Python script. For example, the phrase 'تم کیا بکواس کر رہی ہو' is normalized to 'تم کیا بکواس کر رہی ہو'. This reduces redundancy introduced by users intentionally repeating words for emphasis or aggression. Duplicate words are removed to reduce noise and sparsity in feature space. In informal Urdu social media text, users often repeat words like 'چپ چپ چپ' or 'نہیں نہیں نہیں', which may skew N-gram frequency statistics without adding semantic value. Tokenization alone does not merge or normalize repeated tokens unless explicitly programmed. These duplicates are removed during the preprocessing stage rather than models themselves as features to reduce data imbalance, especially in models sensitive to surface-level repetition like TF-IDF. This study did not measure the strength of user comments; therefore, removing such words is a better option for making the text smoother.

Table 3. Example of repetitive words.

Original Text	Original Text in English	After Removing Repeated Words
منافق منافق جھوٹا دفع ہو جاؤ دفع ہو جاؤ	You are the hypocrite liar, leave.	منافق جھوٹا دفع ہو جاؤ

3.6. Tokenization

The next logical step is to determine the use of words in a single sentence or phrase, for which tokenization is used. Any sentence, phrase, or even symbol of text can be divided into small and manageable units known as tokens. In the case of Urdu text analysis, owing to the multifaceted construction of the Urdu language, this stage is the most important in extracting meaningful parts of the language.

3.7. Data Annotation

Data annotation is performed manually, where each user comment is treated as an individual instance. The dataset consisted of 18,000 instances that had to be annotated with six toxicity labels: positive, negative, rude, offensive, abusive, and hate speech. A crowdsourcing approach to data annotation is employed, engaging three MSCS psychology students to manually annotate the data. All three students are native Urdu speakers and neutral towards user comments. Regular meetings are held to ensure a unified understanding of the context and semantics, following a predefined toxicity hierarchy. An Urdu (Nastaliq) abusive lexicon is utilized from a publicly available dataset on GitHub (https://github.com/pervezbcs/Urdu-Abusive-Dataset/blob/master/Urdu%20Abusive%20Dataset.zip), accessed on 20 June 2025. This lexicon is used to identify whether a given comment has abusive content or not. Table 4 describes each annotation label along with illustrative example. The guidelines for each label are provided in the following.

Table 4. Toxicity Labels with Descriptions and Examples.

Label	Urdu Example (Script)	Transliteration	English Translation	Notes on Nuance
Rude	تم ہمیشہ بیوقوف ہی رہنا	tum hamesha bewaqoof hi rehna	You always stay stupid	Indirect insult (disrespectful, sarcastic)
Offensive	پاکستان ترقی کا سب بڑا دشمن	Pakistan Tarraqi ka sb se bara dushman	The biggest enemy of Pakistan's progress	Targets a group with blame; stronger than rude
Abusive	جھوٹا بہتان باز نفرت پھیلانے والا	Jhoota buhtan baz nafrat phelane wala	Lier, Hawk, Hater	Vulgar + threatening language
Hate Speech	یہ منحوس جماعت نہ اہل ہے	Ye manhoos jamat na-ahal hai	This inauspicious group is unqualified	Targets a religious, political, social group, event or any news

Guidelines for Rude

a. Use of rude, dismissive, or sarcastic words in speech.
b. Lacks basic respect but it does not necessarily involve outright harm or hatred.
c. Tends to take place during arguments or when mocking someone subtly.

Guidelines for offensive

a. Includes words that tend to insult, provoke, or hurt feelings.
b. Seeks sensitive areas such as religion, gender, or looks.
c. More intense in tone and effect than "rude", yet not necessarily abusive.

Guidelines for negative, offensive and abusive

a. Harmful or threatening words meant to intimidate.
b. Has slurs, name-calling, or violent verbal attacks.
c. Incites psychological distress or fear in the recipient.

Guidelines for negative, offensive and hate speech

a. Encourages hatred or violence against individuals or groups based on their identity.
b. Is discriminatory in nature—attacks race, religion, ethnicity, gender, etc.
c. Legally and ethically grave; can lead to actual harm in real life.

Comments are annotated as either rude or offensive, with offensive comments classified as targeted or untargeted. Targeted offensive comments are labeled as either abusive (directed at individuals) or hate speech (targeting groups or events). The process is divided into three stages: Stage 1 involved binary classification to identify positive or negative content; Stage 2 performed multi-class classification for negative content (negative only, offensive, or rude); and Stage 3 used binary classification to distinguish between abusive and hate speech within offensive content. Three independent annotators performed the annotation according to the scenarios. To avoid bias, they performed independent annotations without influencing each other. Fleiss Kappa statistic is used to evaluate the agreement level with all annotators. Approaching +1 denotes stronger concordance on the given annotations. The last labeling choice is based on the greatest inter-annotator agreement. We designated the majority label as final label. For some instances with low Kappa values, we did re-labeling based on standard operating procedures.

The multilabel multiclass classification is shown in Figure 2. An instance may contain multiple labels if it is negative. As shown in Figure 2, a negative instance is further labeled as offensive, abusive, or hate speech. Multiclass means the five classes, and multilabel means a comment represents multiple labels if it is a negative comment. The data annotations and labels are presented in Table 5.

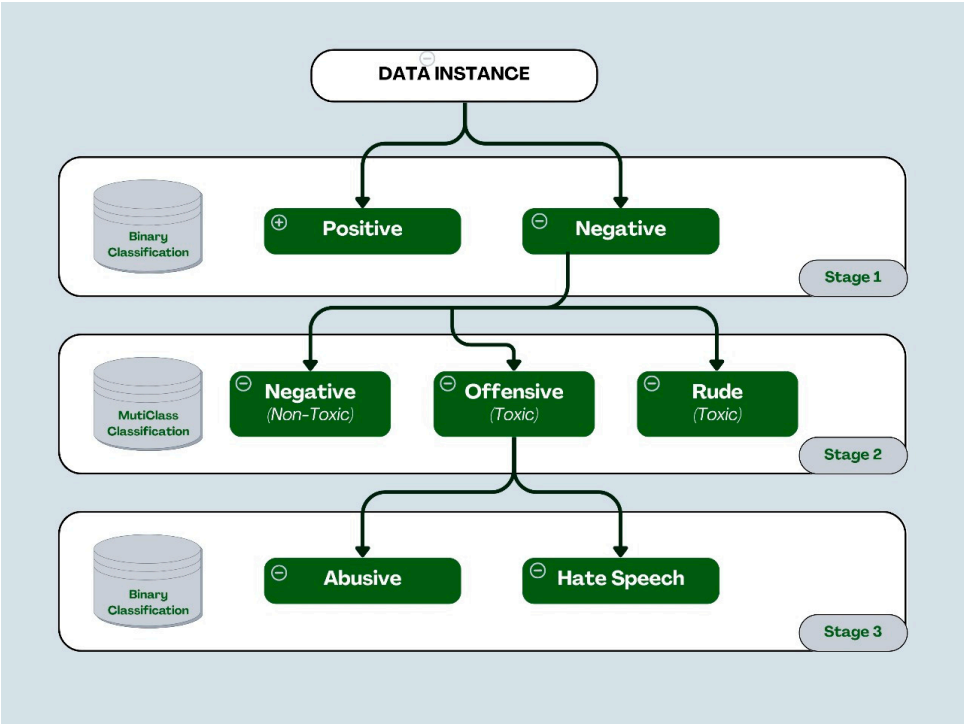

Figure 2. Taxonomy of multilabel multiclass classification in Urdu toxic corpus data.

Table 5. Examples of Data annotation for Urdu toxic corpus data.

Comment	Comments in English	Positive	Negative	Rude	Offensive	Abusive	Hate Speech
گھڑی چور	The Watch theif	0	1	1	0	0	0
فتنہ	Temptation	0	1	0	1	1	0
سب شعہ کافر	All Shias are infidels	0	1	0	1	0	1

To convert the hierarchical multi-label classification approach into a multi-class classification task, the problem must be restructured so that each instance is classified into one and only one of several possible categories. The different labels ("Rude", "Offensive", "Abusive", and "Hate Speech") are combined into five unique classes as follows: A description of all the labels defined in the five classes is provided in Table 6. The statistics of all classes in the dataset are also included in Table 6.

Table 6. Description of classes and their instances.

Classes	Labels	Explanation	Instances
Class 1	Positive only	Instances that are labeled as positive (non-toxic).	4898
Class 2	Negative only	Instances that are negative (toxic) but do not have specific toxic attributes (like rude or offensive).	1009
Class 3	Negative and Rude	Instances that are negative and contain rude language	3415
Class 4	Negative and Offensive and Abusive	Instances that are negative and contain offensive and abusive language.	5783
Class 5	Negative and Offensive and Hate Speech	Instances that are negative and contain offensive language and hate speech.	2894

3.8. Feature Extraction

Feature extraction methods play a vital role in transforming raw textual data into a structured format that ML algorithms can interpret and learn. In this study, the most widely used text representation techniques are applied to capture the essential characteristics and patterns of Urdu for effective classification. The primary feature extraction methods used are BoW, N-grams, and TF-IDF.

3.8.1. Bag-of-Words

This technique represents each document as a vector based on the frequency of words occurring in it without considering the order. Although simple, it effectively captures term presence and frequency.

3.8.2. N-Grams

N-grams (such as bigrams or trigrams) consider sequences of N consecutive words or characters in the text. This helps preserve contextual information and capture local patterns, such as common toxic phrases or expressions in Urdu.

3.8.3. Term Frequency-Inverse Document Frequency

TF-IDF improves the BoW model by assigning weights to terms based on their frequency in a specific document relative to their occurrence across the entire dataset. This highlights important or unique words while downplaying common words, enhancing the discriminative power of the features.

Together, these techniques reduce the dimensionality of the data, filter out irrelevant or redundant information, and ensure consistency and scalability across all the features. This transformation enables classifiers to better detect toxic content by focusing on meaningful textual patterns and structures.

In the identified feature set, both TF-IDIF and N-gram features are important for the detection of toxic content; therefore, they must be used in combination. TF-IDIF helps identify salient keywords using the text of a document, considering its importance relative to other words in the corpus. It also captures BoW information, which works for the retrieval of toxic keywords and slurs, abusive words, and phrases (curse words). N-grams, especially bigrams and trigrams, are also very useful in the retrieval of vertical and topical information and for understanding how toxicity is conveyed. N-grams provides "how" while TF-IDF provides "what". Thus, N-grams enable the model to learn sarcasm, imperative group targeting, and many other grammatical constructs that are common in toxic speech. For instance, the bigram تم لوگ ("you people") or "مار ڈالوں" ("I'll kill") metaphorically captures phrasing patterns associated with offensive or abusive intent. Hence, single words cannot reflect offensive constructs and therefore cannot reflect abusive, aggressive, or sarcastic constructs. Thus, the feature set created by combining TF-IDF weighting with N-gram structures effectively captures not only the morphologically rich Urdu language but also context-sensitive languages, addressing syntax, lexical salience, and toxic patterns.

3.9. Data Imbalance Handling

After completing the data annotation, we observed that the data are imbalanced in the context of class label frequency. We have six class labels: positive, negative, rude, offensive, hate speech, and abusive. The data instances with the label "rude" are fewer than those with the label "offensive", and there are very few "only negative" instances. In the context of toxic language detection for the Urdu dataset, class imbalance is a significant challenge, particularly when certain categories, such as "rude" and "only negative" com-

ments, are very few compared to more frequent labels, such as offensive. As discussed in the Section 3.7, five classes are constructed using class labels. It is evident from Table 6 that "only negative" is the least frequent class in the dataset because it occurs only 1009 times in 18,000 instances. However, the combination of negative with other labels is somehow more than the sum of its parts. The frequency of Class 3 is 3415, Class 4 occurs 5783 times, and Class 5 appears 2894 times in the dataset, which leads to an imbalance owing to Classes 2, 3, and 5.

This situation creates a skew that disproportionately favors the majority class. This reduces the ability to accurately identify minority class instances, which are often the most nuanced and context-dependent. To handle the class imbalance in our toxicity dataset, we utilized the synthetic minority over-sampling technique (SMOTE) method, which artificially creates new instances of the minority class through k-nearest neighbor interpolation. This enhances generalization of the model by promoting better underrepresented class representation. This approach artificially increases the number of instances in minority classes by replicating existing examples, ensuring that all classes contribute equally to the training process, and reducing the bias toward dominant classes. In particular, the dataset is initially split into smaller parts with class labels, focusing on the imbalanced rude and negative classes. The dual-class subset is balanced using the resample() function from the sklearn. The utils module randomly duplicates samples in minority subsets until their instance counts are equal to the most frequent, offensive class. The negative class, which contains non-rude and non-offensive comments, is particularly underrepresented. Therefore, it requires careful upsampling to maintain diversity and avoid overfitting. Subsequently, the subsets of all classes are integrated into a single dataset to ensure equal class distribution. This is particularly important in Urdu toxic language classification, where subtle, culturally contextual cues influence class boundaries and may lead to unfair or biased model performance, particularly when infrequent classes are misclassified as dominant ones.

4. Experimental Setup

This section discusses the experiments performed on the preprocessed dataset. Four ML classifiers are selected to evaluate the effectiveness of different supervised learning techniques for detecting multi-label multi-class toxicity in the Urdu social media dataset. LR, SVM, RF, and GB are selected due to their strong text classification performance and class imbalance robustness.

LR is a linear classification model that estimates the probability of binary outcomes using a sigmoid function. This study used L2 regularization to avoid overfitting. Class imbalance is handled by setting the parameter "class_weight = balance" and GridSearchCV over the regularization strength "C" with values {0.1, 1, 10} is used to conduct the hyperparameter search. The maximum number of iterations is set to 1000 to ensure convergence. The RF is configured with 50 decision trees and "class_weight = balanced" to mitigate class imbalance. The trees are trained in parallel to make the method efficient for large datasets. In addition, a linear SVM kernel is used because the dataset consists of high-dimensional text features. Class_weight is set to handle the imbalanced dataset, and the probability is set to true to set the probability estimates. SVM is less sensitive to outliers due to its margin-based nature. Another shallow learning classifier, GB, is applied to the dataset. Unlike RF, GB executes trees sequentially by correcting the errors of previous trees. The model is trained using 100 estimators with the default learning rates. This classifier is robust in capturing complex patterns and nonlinear relationships. However, this approach is computationally expensive.

To critically assess the performance of traditional ML models for multi-label toxicity classification, we used a stratified 5-fold cross-validation approach. The dataset, with engineered features and binary labels for five classes of toxicity, is divided into five folds while maintaining positive and negative sample distributions in each class. Four popular classifiers, namely LR, RF, SVM, and GB, are individually trained and tested on each fold employing a one-vs-all strategy for multi-label classification. Class weights are used, where supported by the classifier, to address issues of class imbalance. For LR, hyperparameter tuning is incorporated through a nested 3-fold grid search to find the optimal regularization parameter (C) with precision as the scoring function, so that the optimum model setting is chosen per fold. The remaining classifiers are trained using fixed parameters that are typically employed in toxicity detection tasks.

After being trained on the training folds, models are evaluated on the held-out fold, and assessment metrics—accuracy, precision, recall, and F1-score—are calculated for every class. These metrics are averaged over all five folds to give strong and unbiased performance estimates of the models. Precision, recall, and F1-scores visualization over all classes and classifiers further showed comparative weaknesses and strengths. This 5-fold cross-validation process gave a robust benchmark for traditional ML baselines by which the deep models are compared.

To evaluate the performance of DL models for multi-label toxicity classification on Urdu, two experiment strategies are used: train-test split and 5-fold stratified cross-validation. Both methods are utilized with three recurrent neural network models—LSTM, BiLSTM, and GRU—trained under a multilabel multiclass classification paradigm for five classes of toxicity. During the first phase, all the models are trained and tested with a static train-test split. The cleaned Urdu comment dataset is divided into training and testing in an 80:20 ratio while keeping stratified sampling to ensure that the distribution of positive and negative examples among classes remains the same. This enabled rapid testing and early hyperparameter tuning.

Text preprocessed included tokenization through Keras' Tokenizer, translating the preprocessed comments into integer sequences of word indices according to frequency. To make the input length uniform, the sequences are padded to the highest comment length found in the dataset. Individual binary classifiers are trained for each class of toxicity in a multiclass classification manner. All models have same structure: an embedding layer (embedding dimension set within [64, 128, 256] for BiLSTM and GRU, and fixed at 128 for LSTM), followed by either LSTM, BiLSTM, or GRU recurrent layer with units tunable as (32, 64, 128). A dropout layer (dropout rates 0.2–0.5) is used to avoid overfitting, and the last dense layer contained a single sigmoid neuron for binary classification. Hyperparameter tuning is conducted through random search with a small number of trials (usually five) based on validation splits sampled from the training set (ordinarily 10%). The Adam optimizer with binary cross-entropy loss is applied consistently. Performance of models on the unseen test set is measured in terms of accuracy, precision, recall, and F1-score.

This train-test splitting strategy facilitated fast hyperparameter tuning and baseline model architecture evaluation, as well as offering a baseline for future cross-validation. Each model is trained separately for its target class using the same training, validation, and testing strategies to ensure a fair comparison. The evaluation metrics for all models are precision, recall, accuracy, and F1 score.

5. Results and Discussion

Machine and DL classifiers learn from labeled text data and use this knowledge to predict the labels for new text. Two methods are used to measure the performance of these classifiers. The first method involves splitting the data into a training set (70%) and a test

set (30%). For DL models, we also created a validation set by taking 10% of the training data, leaving 60% for training and 30% for testing. It is important to ensure that both the training and test sets are evenly balanced so that easy and hard examples are spread equally. The second method, known as k-fold cross-validation, divides the data into five equal parts. In each iteration, one part is used for testing, whereas the remaining four parts are used for training. This process is repeated five times, and the average performance is calculated. To evaluate multi-label text classification, performance metrics such as precision, accuracy, recall, and F1-score are used because they are good metrics for information retrieval and multi-class classification tasks.

5.1. ML Methods

The results for precision, recall, F1-score, and accuracy across the three models (SVM, RF, and LR) highlighted significant differences in their performance, which are presented in Table 7. For SVM, the precision is relatively low for most classes, especially Class 2 (0.3395) and Class 3 (0.2750), indicating a high number of false positives. However, recall is very high, particularly for Classes 2 (0.9272) and 3 (0.9643), suggesting that the model captures most true instances but has poor precision. This imbalance results in moderate F1-scores, with the best performance for Class 5 (0.6919). In contrast, the RF demonstrates exceptional performance across all metrics. It achieves near-perfect precision, recall, and F1 scores for all classes, with most values exceeding 0.96. This indicates that the model is highly reliable and robust, capturing nearly all true instances while minimizing false positives. However, such high performance may also suggest potential overfitting, which requires further validation in future studies. LR has shown moderate performance compared to the other two models. It achieves high recall for Classes 2 (0.9379) and 3 (0.8452), ensuring that most true instances are identified. However, its precision is consistently low, with values below 0.57 for most classes, leading to a significant number of false positives. This imbalance between precision and recall results in lower F1 scores, particularly for Class 3 (0.4288), which struggles the most across all the metrics. Accuracy is moderate overall, with the best performance for Class 5 (0.8693). Although LR is more interpretable than SVM and RF, it falls short in terms of overall effectiveness, especially when precision is critical.

Table 7. Toxicity detection using a train-test split and using ML algorithms.

ML Method	Classes	TP	FP	FN	TN	Precision	Recall	F1-Score	Accuracy
Logistic Regression	Class 1	3472	3515	1426	9587	0.4969	0.7088	0.5842	0.7768
	Class 2	946	1584	63	15,407	0.3740	0.9379	0.5348	0.7865
	Class 3	2886	7164	529	7421	0.2872	0.8452	0.4288	0.6244
	Class 4	4569	5204	1214	7013	0.4675	0.7900	0.5874	0.6385
	Class 5	1882	1422	1012	13,684	0.5697	0.6504	0.6074	0.8693
Random forest	Class 1	4834	24	64	13,078	0.9951	0.9869	0.9910	0.9960
	Class 2	1009	0	0	16,991	1.0	1.0	1.0	1.0
	Class 3	3415	251	0	14,334	0.9315	1.0	0.9645	0.9877
	Class 4	5712	154	71	12,063	0.9738	0.9878	0.9807	0.9874
	Class 5	2894	40	0	15,066	0.9863	1.0	0.9931	0.9978
Support vector machine	Class 1	3169	2318	1729	10,784	0.5775	0.647	0.6105	0.817
	Class 2	936	1820	73	15,171	0.3395	0.9272	0.4974	0.7542
	Class 3	3293	8682	122	5903	0.2750	0.9643	0.4280	0.570
	Class 4	5254	5819	529	6398	0.4745	0.9086	0.6235	0.6425
	Class 5	1993	874	901	14,232	0.6952	0.6887	0.6919	0.9047

Table 7. *Cont.*

ML Method	Classes	TP	FP	FN	TN	Precision	Recall	F1-Score	Accuracy
Gradient boosting	Class 1	2596	357	2302	12,745	0.879	0.53	0.66	0.88
	Class 2	928	9	81	16,982	0.99	0.92	0.977	0.9849
	Class 3	1708	17	1708	14,568	0.99	0.50	0.66	0.92
	Class 4	2892	251	2892	11,966	0.92	0.50	0.65	0.82
	Class 5	1794	18	1100	15,088	0.99	0.62	0.76	0.94

LR is trained with class_weight = 'balanced' and max_iter = 1000 to handle class imbalance and ensure convergence. RF is set with 50 estimators and class weights balanced to efficiently deal with minority classes. SVM becomes computationally intensive when using a linear kernel, probability estimation, and class_weight = 'balanced.' However, it excels in performing confidence-based evaluations. GB is specified with 100 estimators and learning parameters as the default. All models are trained with the one-vs-all classification approach, wherein a binary classifier for every class is trained against all the others. The results indicated that RF is the model that performed best out of the others, with near-perfect ratings across all classes, followed by strong performance on the part of GB. LR and SVM are quite good on some classes, but overall are less accurate and stable than ensemble-based models. These results indicate the importance of using ensemble methods and cross-validation for stable performance in tasks involving multi-label toxicity detection. Table 8 presents the results of a 5-fold cross-validation over ML methods.

Table 8. 5-fold cross-validation results using ML methods.

ML Method	Classes	Precision	Recall	F1-Score	Accuracy
Logistic Regression	Class 1	0.4833	0.6859	0.5670	0.7672
	Class 2	0.3928	0.9383	0.5537	0.7931
	Class 3	0.2857	0.8405	0.4265	0.6158
	Class 4	0.4490	0.7726	0.5679	0.6259
	Class 5	0.5553	0.6587	0.6025	0.8671
Random forest	Class 1	0.9740	0.9948	0.9841	0.9928
	Class 2	1.0	1.0	1.0	1.0
	Class 3	0.9342	1.0	0.9660	0.9880
	Class 4	0.9715	0.9915	0.9814	0.9881
	Class 5	0.9874	1.0	0.9937	0.9981
Support vector machine	Class 1	0.5706	0.6357	0.6012	0.8126
	Class 2	0.3541	0.9377	0.5537	0.7931
	Class 3	0.2860	0.9735	0.4421	0.5824
	Class 4	0.4678	0.8948	0.6142	0.6427
	Class 5	0.6881	0.6937	0.6907	0.9050
Gradient boosting	Class 1	0.9020	0.5204	0.6598	0.8808
	Class 2	0.9978	0.9522	0.9744	0.9932
	Class 3	0.9811	0.5066	0.6681	0.9145
	Class 4	0.9174	0.5192	0.6630	0.8322
	Class 5	0.9927	0.6450	0.7817	0.9450

The difference between basic ML models based on a basic train-test split and those tested with 5-fold cross-validation shows substantial differences in performance stability and testing reliability, as presented in Table 8. With the basic train-test split (most commonly 70% training and 30% testing), models such as RF and GB achieved extremely high performance for all five classes. For instance, the RF performed very close to perfection, with F1-scores above 0.98 and accuracies close to 1.0 for all classes. Likewise, GB had high precision and recall values for Class 2 and Class 5. The results indicate a good performance. However, they can also indicate overfitting or poor data partitioning. A single split does not necessarily guarantee that the model will generalize effectively to unseen data, especially in imbalanced or heterogeneous datasets, such as those used for toxic comment classification.

The use of 5-fold cross-validation provides a more stable and generalized assessment of the models by splitting the dataset into five parts and iteratively training and validating the model on various proportions. This approach reduces the bias introduced by random splits and allows every instance an opportunity to be in the training and test sets. Consequently, the cross-validated performance indicators for models such as LR and SVM revealed significant declines in precision, recall, and F1-score, especially for rare classes such as Classes 3 and 4. Even top-performing models, such as RF, had slightly lower but more realistic results. This indicates that cross-validation reveals the actual generalization capacity of the models and avoids overly optimistic assessments. The critical review concludes that although plain validation provides a rough estimate of model performance, cross-validation is necessary for equitable and credible comparisons, particularly in scientific applications involving reproducibility and statistical integrity.

5.2. DL Methods

The results across the DL models—LSTM, GRU, and BiLSTM—highlight varying levels of performance in the multiclass classification task, revealing their strengths and limitations. The results of DL algorithms are presented in Table 9. Among the models, LSTM and GRU generally outperform BiLSTM, achieving higher F1-scores for most classes. However, their performance is far from uniform in all categories. To properly identify toxic content in Urdu text, three DL models are used with a multiclass classification approach for multilabel classification. Each model is implemented with an embedding layer with an initial dimension of 128, which converts tokenized words into dense vector representations. This is followed by the central recurrent unit: a typical LSTM layer consisting of 64 units in the LSTM model, a BiLSTM layer with 64 units in the second model, and a GRU layer with 64 units in the third model. A dropout layer of 0.5 rate is implemented after the recurrent layers to minimize overfitting. A final dense output layer with a softmax activation function is used for binary classification in every one-vs-all configuration. In the Urdu toxicity detection experiment, the issue can be defined in two ways, based on whether the labels are represented in a multi-label or single-label format. First, the problem is naturally a multi-label classification problem because one comment can simultaneously have multiple types of toxicity associated with it, for example, being rude, offensive, abusive, and hateful. In such multi-label environments, the sigmoid activation function is used in the final layer of DL models, which enables each label to be estimated independently and allows the model to assign multiple labels to a single instance. Alternatively, reformulating the problem by introducing mutually exclusive classes, for example, Class 1: Positive only, Class 2: Negative only, and Class 3: Negative + Rude, transforms it into a multi-class classification scenario in which each instance is assigned to only one class. In such cases, the softmax activation function is applied in the output layer to produce a probability distribution over all possible classes and selecting the best class for each instance. This difference determines the choice of activation and loss functions: sigmoid activation with binary cross-entropy

for multi-label tasks, and softmax activation with categorical cross-entropy for multi-class tasks.

Table 9. Toxicity detection in the Urdu dataset using DL algorithms using a train-test split.

DL Method	Classes	Precision	Recall	F1-Score	Accuracy
LSTM	Class 1	0.6528	0.5222	0.5802	0.8038
	Class 2	0.8333	0.9091	0.8696	0.9913
	Class 3	0.6667	0.3373	0.4480	0.9004
	Class 4	0.7237	0.8291	0.7728	0.7489
	Class 5	0.7917	0.3654	0.5000	0.9452
GRU	Class 1	0.6782	0.6556	0.6667	0.8297
	Class 2	0.8750	0.6364	0.7368	0.9856
	Class 3	0.6154	0.5783	0.5963	0.9062
	Class 4	0.7771	0.7619	0.7648	0.7694
	Class 5	0.6327	0.5962	0.6139	0.9437
BiLSTM	Class 1	0.741	0.5056	0.5778	0.8081
	Class 2	0.9524	0.9091	0.9302	0.9957
	Class 3	0.5443	0.5181	0.5309	0.8903
	Class 4	0.7584	0.7563	0.7574	0.7504
	Class 5	0.8500	0.3269	0.4722	0.9452

The models are compiled with the Adam optimizer with the default learning rate of 0.001, and the loss function is binary cross-entropy, suited for multilabel classification. The models are trained for 10 epochs with a batch size of 32, and 10% of the training set is utilized as a validation set. The input sequences are padded to an integer maximum sequence length based on the longest sequence in the dataset. For model performance evaluation, accuracy, precision, recall, and F1-score measures are calculated for each class. In order to address class imbalance in the data, SMOTE is used while preprocessing to ensure that minority classes are suitably represented while training. Random search is used for hyperparameter tuning, sampling sets of combinations of embedding dimension, recurrent units, dropout rate, batch size, and learning rate. Stratified sampling is also utilized while splitting the data to ensure that every class is represented proportionally in the training and testing datasets. This strong and uniform pipeline enabled equitable testing and comparison of all three DL models on the multilabel Urdu toxicity classification task.

The results show that the RF classifier outperformed the DL models, which can be attributed to several factors. It is widely known that RFs perform well for small-to-medium-sized datasets. In contrast, more complex models, such as LSTM and GRU, usually require a large dataset volume to learn the general patterns and thresholds. In this particular case, the new dataset contains 18,000 labeled instances, due to which the RF outperformed the DL models. Although attempts are made to balance the dataset and reduce annotation inaccuracies. In addition, RF is more robust in such scenarios. Compiling the set of features for extraction with TF-IDF, N-grams, and BoW involved traditional ML algorithms that exploit these techniques, which is optimal for RF and other traditional ML models. With TF-IDF, RF becomes more accurate in determining critical toxic keywords without requiring an in-depth explanation of the language used. These algorithms are highly effective because they are non-parametric models that focus less on optimizing hyperparameters, leading to a more relaxed error-accepting training phase than other models. Therefore,

strong outcomes can be achieved early in experimentation. In the context of the Urdu dataset, these difficulties provided an opportunity for the RF classifier to achieve the best performance. In the absence of reliable datasets that are extensively balanced and labeled, RF proves to be useful without losing efficiency. Its ability to generalize without overfitting makes it particularly suitable for low-resource language-processing tasks.

The 5-fold cross-validation of the DL algorithms is presented in Table 10. The performance of LSTM, GRU, and BiLSTM on the five toxicity classes shows the subtle strengths and weaknesses of each DL model. LSTM had a high accuracy for Class 2 (0.9841) and Class 5 (0.9339), showing good discriminative power in recognizing these categories. Nonetheless, its precision and recall for Classes 3 (0.5614 and 0.5711, respectively) and 5 (0.5681 and 0.5200) are fairly low, indicating an inability to accurately categorize samples from these classes. Meanwhile, the GRU has a more balanced performance across classes. It particularly improves the precision of Class 5 (0.6173) over LSTM and BiLSTM and shows the highest accuracy (0.9391) for that class. Although GRU has a slightly lower recall value for Class 3 than LSTM, its F1-score (0.5755) is still competitive and reflects higher consistency. BiLSTM achieves the highest precision for Class 2 (0.8109) and performs well on the other classes, though its performance decreases for Class 3 (F1-score: 0.5545). Overall, all three models performed poorly on Class 3, possibly because of overlapping classes or data sparsity. Although LSTM and BiLSTM yield excellent performance for particular classes, GRU shows a better generalization balance, especially for minority or difficult classes. This suggests that GRU is a stronger contender for toxicity detection tasks with imbalanced or subtle class distributions.

Table 10. 5-fold cross-validation using DL algorithms.

DL Method	Classes	Precision	Recall	F1-Score	Accuracy
LSTM	Class 1	0.6119	0.6527	0.6300	0.8019
	Class 2	0.8104	0.6779	0.7255	0.9841
	Class 3	0.5614	0.5711	0.5592	0.8885
	Class 4	0.7700	0.7611	0.7654	0.7597
	Class 5	0.5681	0.5200	0.5391	0.9339
GRU	Class 1	0.6035	0.6426	0.6210	0.7970
	Class 2	0.7679	0.6965	0.7237	0.9835
	Class 3	0.5889	0.5663	0.5755	0.9004
	Class 4	0.7616	0.7577	0.7596	0.7531
	Class 5	0.6173	0.4965	0.5463	0.9391
BiLSTM	Class 1	0.6061	0.6392	0.6212	0.7979
	Class 2	0.8109	0.6238	0.6942	0.9833
	Class 3	0.5369	0.5807	0.5545	0.8874
	Class 4	0.7643	0.7434	0.7533	0.7493
	Class 5	0.5855	0.5080	0.5420	0.9359

A comparative analysis of the 5-fold cross-validation performance of ML and DL models indicates significant performance differences between toxicity classes. Among the ML models, RF is consistently superior to the others, with nearly perfect precision, recall, F1-score, and accuracy in all five classes. This gives an F1-score of 0.9841 for Class 1, 1.0 for Class 2, and equally strong values for the rest of the classes, reflecting its strength and good generalization capability. GB also produces competitive results, particularly for Classes 2

and 5. On the other hand, DL models—LSTM, GRU, and BiLSTM— produce fair to good performance but do not reach the precision and reliability of ML equivalents. The best F1-score for LSTM is 0.7654 for Class 4, but it performed poorly on Class 3 and Class 5, yielding F1-scores less than 0.60. GRU performs marginally better, with better inference time than other DL models, but is still behind RF in classification accuracy. Despite using a bidirectional context, BiLSTM has no apparent gain over its unidirectional counterparts in this environment. Additionally, the ML models are significantly more computationally efficient. RF uses 31.91 s for training and 0.79 s for inference, whereas DL models have much longer training times, from 1270 s (LSTM) to more than 2500 s (GRU), and slower inference rates. This efficiency, coupled with high accuracy, underscores the real-world advantages of ML models for toxicity detection tasks, particularly in situations where computational resources or speed of deployment are limited.

In summary, although DL models have the theoretical benefit of learning complex sequential patterns, they are surpassed by traditional ML models, specifically RF, in both effectiveness and efficiency in this work. Considering the nature and size of the dataset, the findings suggest that well-optimized ML models are highly competitive and, in this case, more suitable for Urdu toxicity classification than DL models.

The related existing studies in the literature, which are discussed in Section 2, reported an accuracy ranging from 72.6% to 91% and the obtained results are not directly comparable due to significant methodological differences. Most of these studies focused on binary or multiclass classification with a limited number of labels, such as offensive content or cyberbullying. These studies are conducted on less diverse domains and smaller datasets, for instance, 12,428 tweets. In contrast, our proposed framework addresses a more complex challenge: multi-class, multi-label toxicity detection within a hierarchically labeled dataset of 18,000 Urdu social media posts covering diverse topics such as politics, religion, and sports. Our model adopts a more advanced and structured methodology, incorporating one-vs-all classification, label hierarchy, SMOTE to manage class imbalance, and both ML and DL models with optimized hyperparameters. The enhanced performance of our models, particularly the RF (F1-score ≈ 0.98), can likely be attributed to these conditions. However, we recognize that for a fair comparison, future research should assess models using standardized taxonomies, pre-processing pipelines, and evaluation protocols.

Another key observation from the obtained results is the robustness of DL models across all classes compared to conventional ML algorithms such as LR and SVM. Although LR and SVM perform well on certain classes, their class-wise performance is highly varying, often performing poorly on minority or complex class labels. However, DL models (e.g., LSTM, BiLSTM, GRU) and some ensemble ML algorithms (e.g., RF, GB) exhibit more balanced and stable performance across all class labels.

5.3. Computational Time Analysis

In addition to measuring the classification accuracy of ML and DL models, it is important to determine their computational efficiency for deployment in real-world applications. This includes both the training and inference (prediction) times for each model. Models that are highly accurate but computationally expensive may not be practical in situations where time or resources are constrained. Hence, the training and inference times of the RF, LSTM, BiLSTM, and GRU are measured and compared for all five classification problems. This comparison highlights the trade-off between accuracy and computational cost, offering insights into the scalability and usability of each model.

Figures 3 and 4 show a noticeable variation in the training and inference times between the ML and DL models. The RF model is found to be the fastest and most efficient, requiring less than 8 s for training and less than 0.22 s per class for inference, thus making

it applicable for real-time deployment. Conversely, DL models, particularly BiLSTM, experience significant challenges with extended training durations, as training a single class with BiLSTM can take up to 718 s. Although LSTM and GRU are significantly faster than BiLSTM in terms of training time, they still require considerably more time than ML-based models. However, the flexibility of DL models is advantageous because they can accommodate and capture complex patterns in sequential data, thereby justifying the additional computational burden in certain scenarios. The comparative performance in training and inference times among the RF, LSTM, BiLSTM, and GRU models illustrates the significant trade-offs between model complexity and computational efficiency. RF has the lowest training (31.91 s) and inference (0.79 s) times, making it suitable for time-sensitive or resource-limited environments. The LSTM model requires 1270.94 s for training and 15.68 s for inference, whereas BiLSTM requires even more time for training (2141.33 s) and slightly longer inference time (15.85 s). Interestingly, the GRU model, despite having the longest training time (2557.60 s), achieves the quickest inference time (9.89 s) among the DL models. This suggests that the GRU strikes an optimal balance between deployment efficiency and the ability to learn from context.

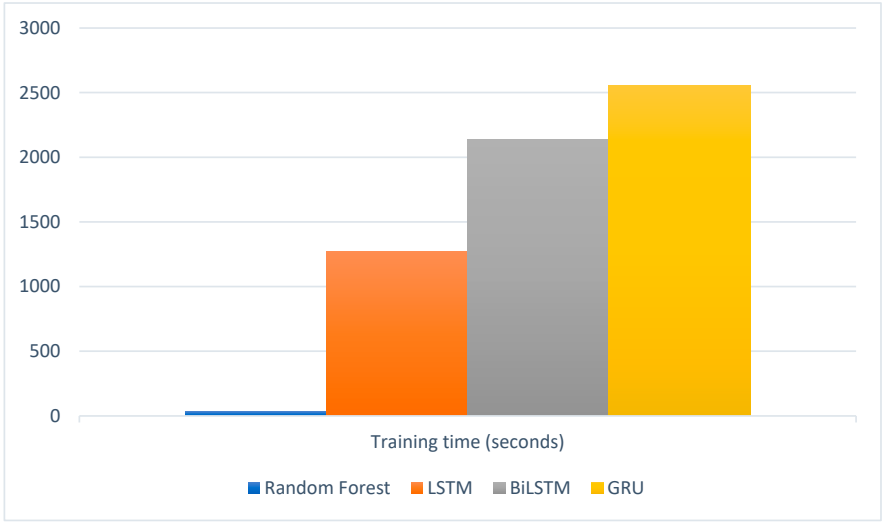

Figure 3. Training time analysis.

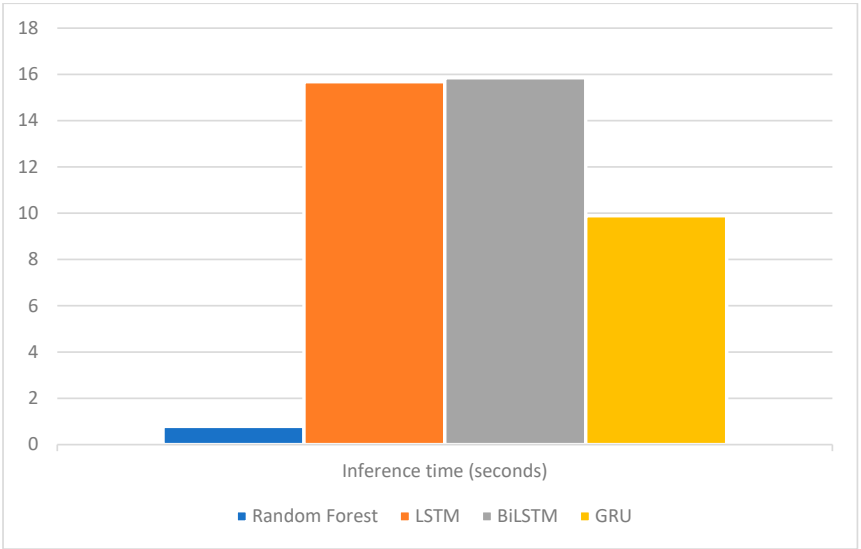

Figure 4. Inference time analysis.

5.4. Statistical Comparison of Model Performances

Paired *t*-tests are used to analyze the fold-wise F1-scores of RF with the DL models, GRU and BiLSTM, on all five classes. The results are shown in Table 11.

Table 11. Statistical Comparison of Model Performance.

Class	RF vs. GRU (t, *p*)	RF vs. BiLSTM (t, *p*)
Class 1	t = 39.161, $p < 0.001$	t = 61.594, $p < 0.001$
Class 2	t = 12.618, $p < 0.001$	t = 6.690, $p < 0.001$
Class 3	t = 25.023, $p < 0.001$	t = 33.556, $p < 0.001$
Class 4	t = 42.422, $p < 0.001$	t = 35.362, $p < 0.001$
Class 5	t = 19.121, $p < 0.001$	t = 19.104, $p < 0.001$

These significant findings ($p < 0.001$) in Table 11 indicate that the performance of the RF is also statistically better than that of GRU and BiLSTM for all classes. The high statistical significance shows the differences in performance are extremely unlikely to be caused by random chance. It reaffirms the conclusion that, in our dataset and task of Urdu toxicity classification, the RF model offers better and more robust classification performance than the benchmarked DL models.

5.5. Comparative Analysis

To evaluate the efficiency of our proposed framework, we performed a comparative analysis with [2] and [7], which focused on multiclass classification to identify cyberbullying on social media platforms from the Urdu dataset. Table 12 presents a comparative analysis of the evaluation metrics of the benchmark study and the proposed model. In [2], FastText outperformed their dataset in terms of precision, recall, accuracy, and F1 score of 82.4%. The dataset is created using 12,428 tweets. Another study [7] used a multiclass approach to detect cyberbullying from Roman Urdu tweets and identified abuser profiles from tweets. GRU is applied with Word2Vec to generate a dataset. They achieved 97% accuracy and recall rates. In contrast, the proposed work develops a dataset of 18 K instances from multiple social media forums, such as Facebook, Twitter (X), and Urdu News, and achieves significant results. Moreover, state-of-the-art ML and DL models are used for toxicity detection using a comprehensively generated dataset. The proposed framework outperforms existing frameworks by achieving 99% accuracy, precision, recall, and F1-score with RF. In addition, BiLSTM also shows good performance on the dataset and achieves 83% accuracy.

Table 12. Performance comparison analysis of the proposed UTDM with benchmark studies.

Ref.	Dataset Size	Data Sources	Methods	Objective	Language	Precision	Recall	F1-Score	Accuracy
[7]	10,060	Twitter (X)	LR, RF, SVM, GB, decision tree, NB, GRU	Cyberbullying detection	Roman Urdu	0.954	0.952	0.952	0.952
[2]	12,428	Twitter (X)	LR, SVM, LSTM, FastText	Cyberbullying detection	Urdu (Nastaliq)	0.83	0.83	0.83	0.83
This Paper	18,000	Facebook, Twitter (X), YouTube, Urdu News	RF, LR, SVM, GB, LSTM, BiLSTM, GRU	Toxicity Detection	Urdu (Nastaliq)	0.977	0.994	0.985	0.993

The benchmark dataset of [2], consisting of user comments in Urdu (Nastaliq) script, is used to test the performance of the proposed UTDM. As the original dataset contains

emojis and non-Urdu characters, a robust preprocessing pipeline is used to clean the data and maintain the linguistic integrity of the Urdu language. This preprocessing involves cleaning emojis, special characters, and non-Urdu characters to ensure that the textual data properly captures the structure and semantics of native Urdu content. With these noise factors addressed, the dataset is improved to better match the objectives of toxicity detection using the proposed model.

After preprocessing, the UTDM is applied to the benchmark dataset and results are presented in Table 13. Experimental evidence shows that the RF classifier performs better than other classifiers, with an accuracy of 98%. This represents a 15% improvement over that reported in the initial benchmark study. The performance metrics of the UTDM for this dataset are presented in Table 12. This remarkable improvement in performance indicates not only the efficiency of the preprocessing methods adopted but also the resilience and flexibility of the UTDM framework for identifying toxic content in Urdu (Nastaliq) texts.

Table 13. Performance analysis of UTDM on a benchmark dataset.

Model	Precision	Recall	F1-Score	Accuracy
LR	0.6382	0.7541	0.6912	0.75
RF	0.9891	0.9892	0.9891	0.9892
SVM	0.82	0.81	0.81	0.82
GB	0.75	0.73	0.74	0.73
LSTM	0.7867	0.8024	0.7878	0.802
BiLSTM	0.7721	0.7833	0.7691	0.7833
GRU	0.8609	0.8796	0.8596	0.8796

The results in Table 13 show that UTDM outperforms the baseline methods in toxicity detection for Urdu texts. Among the ML models, RF achieves the highest accuracy of 98.92%, which validates its ability to handle complex feature spaces effectively, reduce overfitting, and improve classification accuracy. Additionally, RF results in high precision and recall, which explains the reliable detection of toxic content. This significant improvement over previous studies highlights the effectiveness of the proposed UTDM. In DL models, GRU achieved an accuracy of 87.96%, outperforming LSTM and BiLSTM, demonstrating its effectiveness in capturing distant dependencies in text. The performance of the GRU can be attributed to its gated mechanism, which reduces the vanishing gradient problem and allows the model to retain relevant contextual information more effectively. Compared with LSTM and BiLSTM, GRU requires fewer parameters and computational resources, making it an optimal choice for large-scale text classification. Unlike prior research, which primarily focused on multiclass classification, this paper also incorporates multi-label classification, allowing for more granular toxicity detection. This enables the model to assign multiple toxicity labels to a single comment, thereby capturing the complexity of real-world toxic language. The combination of advanced preprocessing, feature selection, and model optimization enabled UTDM to perform better than baseline methods, and it provides a new benchmark for Urdu cyberbullying detection in both multiclass and multilabel settings.

5.6. Ethical and Deployment Challenges

Automated toxicity detection systems, particularly in politically or culturally sensitive areas such as Urdu-speaking regions, raise significant concerns regarding censorship, prejudice, and bias. These systems are prone to suppressing satire, political dissent, or culturally rich expressions due to their rigidity in flagging content—especially when it comes

to sarcastic remarks, religious references, or local language jokes. To counter such risks, future research should add human-in-the-loop moderation combined with explainable AI techniques, such as SHAP, attention maps, or other confidence-based flagging systems designed to avoid unjust moderation. Furthermore, real-world social media scenarios require a combination of speed and resource efficiency. Although models such as BiLSTM are powerful, they often require significant computational resources, rendering them impractical for large-scale or real-time applications. The authors could benefit from exploring lightweight or distilled architectures or hybrid pipelines, where a rapid content classifier precedes a more detailed analysis with a deeper model.

6. Conclusions

This paper makes important contributions to user comment toxicity detection in Urdu for social media. Because of the intricate morphological structure of Urdu (Nastaliq) and its poor NLP resources, there is no extensive multilabel multiclass dataset. In this paper, a large multilabel multiclass Urdu Toxic Corpus is created. This dataset is a valuable resource for developing and evaluating toxicity detection models. One of the key features of this paper is the dataset design and annotation. A three-stage taxonomy is suggested for manual data annotation that guarantees a strong framework for classifying data instances as toxic or non-toxic. This hierarchical taxonomy not only supports huge classification but also assures a scalable framework that can be implemented for other low-resource languages in the future. Another significant contribution of this research is proposing UTDM framework to detect and classify toxic content in comments of Urdu social media users. The model is trained and tested using both ML and DL methods. Among the ML algorithms used, RF performed extraordinarily well, with an accuracy, precision, recall, and F1-score of 0.993, 0.977, 0.994, and 0.985, respectively, becoming the standard for toxicity detection in Urdu.

DL models have also proven their capability to handle certain types of toxicity efficiently. Three DL architectures are implemented: LSTM, BiLSTM, and GRU. The LSTM model excelled in classes 1 and 2, which is a testament to its potential to learn contextual dependencies in these two classes. However, the GRU is superior in classes 3, 4, and 5 and showed better classification of more complicated toxicity cases. Nevertheless, this study has some limitations. Although the manual annotation procedure is extensive, it is time-consuming and susceptible to subjective biases. Moreover, the scope of the study is limited to the Urdu dataset. Future research can focus on expanding the dataset to include a larger and more diverse collection of Urdu texts, using semi-supervised or unsupervised training methods to reduce dependence on human labeling, and developing deeper pre-trained transformer models, such as BERT or GPT, specifically fine-tuned for the Urdu language.

In summary, this paper provided a robust foundation for advancing Urdu natural language processing, specifically in addressing the challenging task of identifying toxic content. The proposed data resource and framework not only improve toxicity detection in Urdu but also create new directions for research in multilingual and low-resource natural language processing, making the digital world more secure and inclusive.

Author Contributions: Conceptualization, A.R. and S.M.; methodology, A.R., S.M. and U.I.; software, A.R.; validation, U.I. and M.F.Z.; formal analysis, A.R. and S.M.; investigation, A.R. and U.I.; resources, A.R.; data curation, A.R., S.M. and M.F.Z.; writing—original draft preparation, A.R.; writing—review and editing, A.R., S.M., U.I. and M.F.Z.; visualization, A.R., U.I. and M.F.Z.; supervision, S.M. and M.F.Z.; project administration, S.M. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The data are available publicly.

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

References

1. Khan, A.; Ahmed, A.; Jan, S.; Bilal, M.; Zuhairi, M.F. Abusive language detection in Urdu text: Leveraging DL and attention mechanism. *IEEE Access* **2024**, *12*, 37418–37431. [CrossRef]
2. Adeeba, F.; Yousuf, M.I.; Anwer, I.; Tariq, S.U.; Ashfaq, A.; Naqeeb, M. Addressing cyberbullying in Urdu tweets: A comprehensive dataset and detection system. *PeerJ Comput. Sci.* **2024**, *10*, e1963. [CrossRef]
3. Khan, S.; Qasim, I.; Khan, W.; Khan, A.; Ali Khan, J.; Qahmash, A.; Ghadi, Y.Y. An automated approach to identify sarcasm in low-resource language. *PLoS ONE* **2024**, *19*, e0307186. [CrossRef]
4. Zoya; Latif, S.; Latif, R.; Majeed, H.; Jamail, N.S.M. Assessing Urdu language processing tools via statistical and outlier detection methods on Urdu tweets. ACM Trans. *Asian Low-Resour. Lang. Inf. Process.* **2023**, *22*, 1–31. [CrossRef]
5. Ashraf, M.R.; Jana, Y.; Umer, Q.; Jaffar, M.A.; Chung, S.; Ramay, W.Y. BERT-based sentiment analysis for low-resourced languages: A case study of Urdu language. *IEEE Access* **2023**, *11*, 110245–110259. [CrossRef]
6. Bonetti, A.; Martínez-Sober, M.; Torres, J.C.; Vega, J.M.; Pellerin, S.; Vila-Francés, J. Comparison between ML and DL approaches for the detection of toxic comments on social networks. *Appl. Sci.* **2023**, *13*, 6038. [CrossRef]
7. Atif, A.; Zafar, A.; Wasim, M.; Waheed, T.; Ali, A.; Ali, H.; Shah, Z. Cyberbullying detection and abuser profile identification on social media for Roman Urdu. *IEEE Access* **2024**, *12*, 123339–123351. [CrossRef]
8. Saeed, R.; Afzal, H.; Rauf, S.A.; Iltaf, N. Detection of offensive language and its severity for low-resource languages. *ACM Trans. Asian Low-Resour. Lang. Inf. Process.* **2023**, *22*, 1–27. [CrossRef]
9. Mehmood, F.; Shahzadi, R.; Ghafoor, H.; Asim, M.N.; Ghani, M.U.; Mahmood, W.; Dengel, A. EnML: Multi-label ensemble learning for Urdu text classification. *ACM Trans. Asian Low-Resour. Lang. Inf. Process.* **2023**, *22*, 1–31. [CrossRef]
10. Haq, N.U.; Ullah, M.; Khan, R.; Ahmad, A.; Almogren, A.; Hayat, B.; Shafi, B. USAD: An intelligent system for slang and abusive text detection in Perso-Arabic-scripted Urdu. *Complexity* **2020**, *2020*, 6684995. [CrossRef]
11. Malik, M.S.I.; Nawaz, A.; Jamjoom, M.M. Hate speech and target community detection in Nastaliq Urdu using transfer learning techniques. *IEEE Access* **2024**, *12*, 116875–116890. [CrossRef]
12. Ali, M.Z.; Ehsan-Ul-Haq; Rauf, S.; Javed, K.; Hussain, S. Improving hate speech detection of Urdu tweets using sentiment analysis. *IEEE Access* **2021**, *9*, 84296–84305. [CrossRef]
13. Akram, M.H.; Shahzad, K.; Bashir, M. ISE-Hate: A benchmark corpus for inter-faith, sectarian, and ethnic hatred detection on social media in Urdu. *Inf. Process. Manag.* **2023**, *60*, 103270. [CrossRef]
14. Ameer, I.; Sidorov, G.; Gomez-Adorno, H.; Nawab, R.M.A. Multi-label emotion classification on code-mixed text: Data and methods. *IEEE Access* **2022**, *10*, 8779–8789. [CrossRef]
15. Akhter, M.P.; Zhang, J.; Naqvi, I.R.; Abdelmajeed, M.; Sadiq, M.T. Automatic detection of offensive language for Urdu and Roman Urdu. *IEEE Access* **2020**, *8*, 91213–91226. [CrossRef]
16. Aziz, K.; Yusufu, A.; Zhou, J.; Ji, D.; Iqbal, M.S.; Wang, S.; Hadi, H.J.; Yuan, Z. UrduAspectNet: Fusing transformers and dual GCN for Urdu aspect-based sentiment detection. *ACM Trans. Asian Low-Resour. Lang. Inf. Process.* **2024**, *23*, 3663367. [CrossRef]
17. Ullah, A.; Khan, K.U.; Khan, A.; Bakhsh, S.T.; Rahman, A.U.; Akbar, S.; Saqia, B. Threatening language detection from Urdu data with a deep sequential model. *PLoS ONE* **2024**, *19*, e0290915. [CrossRef]
18. Nabeel, Z.; Mehmood, M.; Baqir, A.; Amjad, A. Classifying emotions in Roman Urdu posts using machine learning. In Proceedings of the 2021 Mohammad Ali Jinnah University International Conference on Computing (MAJICC), Karachi, Pakistan, 6–7 December 2021; pp. 1–7. [CrossRef]
19. Qureshi, M.A.; Asif, M.; Hassan, M.F.; Abid, A.; Kamal, A.; Safdar, S.; Akbar, R. Sentiment analysis of reviews in natural language: Roman Urdu as a case study. *IEEE Access* **2022**, *10*, 24945–24954. [CrossRef]
20. Liaqat, M.I.; Hassan, M.A.; Shoaib, M.; Khurshid, S.K.; Shamseldin, M.A. Sentiment analysis techniques, challenges, and opportunities: Urdu language-based analytical study. *PeerJ Comput. Sci.* **2022**, *8*, e1032. [CrossRef]
21. Sehar, U.; Kanwal, S.; Dashtipur, K.; Mir, U.; Abbasi, U.; Khan, F. Urdu sentiment analysis via multimodal data mining based on DL algorithms. *IEEE Access* **2021**, *9*, 153072–153082. [CrossRef]
22. Naqvi, U.; Majid, A.; Abbas, S.A. UTSA: Urdu text sentiment analysis using DL methods. *IEEE Access* **2021**, *9*, 114085–114094. [CrossRef]
23. Saeed, H.H.; Ashraf, M.H.; Kamiran, F.; Karim, A.; Calders, T. Roman Urdu toxic comment classification. *Lang. Resour. Eval.* **2021**, *55*, 971–996. [CrossRef]
24. Jadhav, R.; Agarwal, N.; Shevate, S.; Sawakare, C.; Parakh, P.; Khandare, S. Cyber bullying and toxicity detection using machine learning. In Proceedings of the 2023 3rd International Conference on Pervasive Computing and Social Networking (ICPCSN), Bengaluru, India, 17–18 February 2023; pp. 66–73. [CrossRef]

25. Ansari, G.; Kaur, P.; Saxena, C. Data augmentation for improving explainability of hate speech detection. *Arab. J. Sci. Eng.* **2024**, *49*, 3609–3621. [CrossRef]
26. Muralikumar, M.D.; Yang, Y.S.; McDonald, D.W. A human-centered evaluation of a toxicity detection API: Testing transferability and unpacking latent attributes. *ACM Trans. Soc. Comput.* **2023**, *6*, 1–38. [CrossRef]
27. Singh, R.K.; Sanjay, H.A.; SA, P.J.; Rishi, H.; Bhardwaj, S. NLP-based hate speech detection and moderation. In Proceedings of the 2023 7th International Conference on Computation System and Information Technology for Sustainable Solutions (CSITSS), Pune, India, 11–12 November 2023; pp. 1–5. [CrossRef]
28. Cinelli, M.; Pelicon, A.; Mozetič, I.; Quattrociocchi, W.; Novak, P.K.; Zollo, F. Dynamics of online hate and misinformation. *Sci. Rep.* **2021**, *11*, 22083. [CrossRef]
29. Salminen, J.; Hopf, M.; Chowdhury, S.A.; Jung, S.; Almerekhi, H.; Jansen, B.J. Developing an online hate classifier for multiple social media platforms. *Hum.-Centric Comput. Inf. Sci.* **2020**, *10*, 1. [CrossRef]
30. Teng, T.H.; Varathan, K.D. Cyberbullying detection in social networks: A comparison between ML and transfer learning approaches. *IEEE Access* **2023**, *11*, 55533–55560. [CrossRef]
31. Yang, Z.; Grenon-Godbout, N.; Rabbany, R. Game on, hate off: A study of toxicity in online multiplayer environments. *Games Res. Pract.* **2024**, *9*, 3675805. [CrossRef]
32. Kabakus, A.T. Towards the importance of the type of deep neural network and employment of pre-trained word vectors for toxicity detection: An experimental study. *J. Web Eng.* **2021**, *20*, 2243–2268. [CrossRef]
33. Carta, S.; Correra, A.; Mulas, R.; Recupero, D.; Saia, R. A supervised multi-class multi-label word embeddings approach for toxic comment classification. In Proceedings of the 11th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management, Leipzig, Germany, 6–8 September 2019; pp. 105–112. [CrossRef]
34. Mnassri, K.; Rajapaksha, P.; Farahbakhsh, R.; Crespi, N. Hate speech and offensive language detection using an emotion-aware shared encoder. In Proceedings of the ICC 2023—IEEE International Conference on Communications, Rome, Italy, 28 May–1 June 2023; pp. 2852–2857. [CrossRef]
35. Li, J.; Valdivia, K.P. Does media format matter? Investigating the toxicity, sentiment, and topic of audio versus text social media messages. In Proceedings of the 10th International Conference on Human-Agent Interaction, Christchurch, New Zealand, 5–8 December 2022.
36. Suresh, S.; Yadav, B.; Kumari, S.; Choudhary, A.; Krishika, R.; TR, M. Performance analysis of comment toxicity detection using machine learning. In Proceedings of the 2023 International Conference on Computer Science and Emerging Technologies (CSET), Kuala Lumpur, Malaysia, 5–6 September 2023; pp. 1–6. [CrossRef]
37. Abbasi, A.; Javed, A.R.; Iqbal, F.; Kryvinska, N.; Jalil, Z. DL for religious and continent-based toxic content detection and classification. *Sci. Rep.* **2022**, *12*, 17478. [CrossRef] [PubMed]
38. Rachidi, R.; Ouassil, M.A.; Errami, M.; Cherradi, B.; Hamida, S.; Silkan, H. Social media's toxic comments detection using artificial intelligence techniques. In Proceedings of the 2023 3rd International Conference on Innovative Research in Applied Science, Engineering and Technology (IRASET), Tangier, Morocco, 21–22 June 2023; pp. 1–6. [CrossRef]
39. Hashmi, E.; Yayilgan, S.Y.; Shaikh, S. Augmenting sentiment prediction capabilities for code-mixed tweets with multilingual transformers. *Soc. Netw. Anal. Min.* **2024**, *14*, 86. [CrossRef]
40. Mehmood, F.; Ghani, M.U.; Ibrahim, M.A.; Shahzadi, R.; Mahmood, W.; Asim, M.N. A precisely Xtreme-Multi Channel hybrid approach for Roman Urdu sentiment analysis. *IEEE Access* **2020**, *8*, 192740–192759. [CrossRef]
41. Mehmood, K.; Essam, D.; Shafi, K. Sentiment analysis system for Roman Urdu. In *Intelligent Computing*; Arai, K., Kapoor, S., Bhatia, R., Eds.; Springer: Cham, Switzerland, 2019; Volume 858, pp. 29–42. [CrossRef]
42. Ghulam, H.; Zeng, F.; Li, W.; Xiao, Y. Deep learning-based sentiment analysis for Roman Urdu text. *Procedia Comput. Sci.* **2019**, *147*, 131–135. [CrossRef]
43. Hussain, S.; Ali Shah, S.F.; Bostan, H. Analyzing hateful comments against journalists on X in Pakistan. *J. Stud.* **2025**, *26*, 1187–1207. [CrossRef]
44. Kausar, S.; Tahir, B.; Mehmood, M.A. ProSOUL: A framework to identify propaganda from online Urdu content. *IEEE Access* **2020**, *8*, 186039–186054. [CrossRef]
45. Saeed, H.H.; Khalil, T.; Kamiran, F. Urdu Toxic Comment Classification with PURUTT Corpus Development. *IEEE Access* **2025**, *13*, 21635–21651. [CrossRef]
46. Ahmed, U.; Raza, A.; Saleem, K.; Sarwar, A. Development of Voting-Based POS Tagger for the URDU Language. *J. Comput. Sci. Appl. (JCSA)* **2025**, *2*, 1–14. [CrossRef]
47. Bilal, M.; Khan, A.; Jan, S.; Musa, S. Context-aware DL model for detection of Roman Urdu hate speech on social media platform. *IEEE Access* **2022**, *10*, 121133–121151. [CrossRef]

48. Shrestha, A.; Kaati, L.; Akrami, N.; Linden, K.; Moshfegh, A. Harmful communication: Detection of toxic language and threats in Swedish. In Proceedings of the 2023 International Conference on Advances in Social Networks Analysis and Mining, Ottawa, ON, Canada, 27–30 August 2023; pp. 624–630. [CrossRef]
49. Zain, M.; Hussain, N.; Qasim, A.; Mehak, G.; Ahmad, F.; Sidorov, G.; Gelbukh, A. RU-OLD: A Comprehensive Analysis of Offensive Language Detection in Roman Urdu Using Hybrid Machine Learning, Deep Learning, and Transformer Models. *Algorithms* **2025**, *18*, 396. [CrossRef]