Sandalphon@DravidianLangTech-EACL2024: Hate and Offensive Language Detection in Telugu Code-mixed Text using Transliteration-Augmentation

Nafisa Tabassum, Mosabbir Hossain Khan, Shawly Ahsan
Jawad Hossain, and Mohammed Moshiul Hoque
Department of Computer Science and Engineering
Chittagong University of Engineering and Technology
{u1804066,u1704085,u1704057,u1704039}@student.cuet.ac.bd
moshiul_240@cuet.ac.bd

Abstract

Hate and offensive language in online platforms pose significant challenges, necessitating automatic detection methods. Particularly in the case of codemixed text, which is very common in social media, the complexity of this problem increases due to the cultural nuances of different languages. DravidianLangTech-EACL2024 organized a shared task on detecting hate and offensive language for Telugu. To complete this task, this study investigates the effectiveness of transliteration-augmented datasets for Telugu code-mixed text. In this work, we compare the performance of various machine learning (ML), deep learning (DL), and transformer-based models on both original and augmented datasets. Experimental findings demonstrate the superiority of transformer models, particularly Telugu-BERT, achieving the highest f_1-score of 0.77 on the augmented dataset, ranking the 1^{st} position in the leaderboard. The study highlights the potential of transliteration-augmented datasets in improving model performance and suggests further exploration of diverse transliteration options to address real-world scenarios.

1 Introduction

In recent years, the growing problem of offensive language in user-generated content on online platforms and its harmful impacts has become a primary concern. The vast amount of daily user-generated content poses a significant challenge in the fight against offensive language. Consequently, automated methods are necessary for handling this task. Understanding code-mixed data is difficult due to several factors. First, it requires a deep understanding of the different linguistic levels involved. Second, the complex structure of code-mixed language makes it challenging to analyze. Finally, there needs to be more training data available for code-mixed languages, which hinders the development of effective classification systems. These challenges can lead to inaccurate classifications, mainly when using systems trained only on monolingual data. When a system trained on a single language encounters code-mixed data, it may be unable to accurately identify the different languages or understand the complex grammar rules involved. These challenges were addressed in the studies of Priyadharshini et al. (2023), which introduced a shared task for detecting hate and offensive language in Telugu. This paper contributes to the ongoing research efforts explored at the DravidianLangTech-EACL2024 (B et al., 2024).

The primary contributions of this work are illustrated in the following:

- Developed a transliteration-based augmentation scheme to help transformer models detect code-mixed Telugu offensive texts with high fidelity.
- Investigated various Ml, DL, and transformer-based techniques for the task and analyzed their performance in augmented and non-augmented datasets.

2 Related Work

Social media can often spread negativity and hurtful content. It is important to recognize such content because it can offend individuals and groups based on race, gender, or religion (Das et al., 2022). To maintain the integrity of the social media ecosystem, researchers and stakeholders must focus on creating computational models capable of swiftly detecting and categorizing offensive content (Sharif et al., 2021). Hence, to address this growing concern on social media platforms, earlier research utilized diverse machine learning algorithms such as Linear Regression, Support Vector Machine, and Naive Bayes for the automated identification of hate speech (Abro et al., 2020). The performance of these models falls short as they need help to capture the semantic and contextual information present in textual data. Utilizing a publicly available dataset of tweets, Wei et al. (2021) suggests a methodology for automating tweets into three classes: Hate, Offensive, and Neither. Their approach utilizes BiLSTM models with blank embedding and pre-trained Glove embeddings to identify Offensive Language and Hate Speech. In recent years, pre-trained models have demonstrated remarkable accuracy in classifying code-mixed texts across various languages (Hande et al., 2020). The current focus of research in text analysis is within the transliteration domain, which holds the potential for achieving superior results (Shekhar et al., 2023).

3 Task and Dataset Descriptions

This task aimed to develop a model that successfully detects code-mixed offensive texts. To implement such a model, we utilized the Telugu-English code-mixed corpus that the organizers[1] provided to the participants. The dataset contains labels of two classes: *hate* and *non-hate*.

Label	Train	Augmented	Test	W_T
hate	1939	5816	250	18094
non-hate	2061	6181	250	21378
Total	4000	11997	500	39492

Table 1: Distribution of datasets, where W_T denotes total words in the train dataset

Table 1 shows the distribution of different classes across the train and test datasets. Both the train and test datasets are well-balanced. The texts in both the train and test dataset were preprocessed and cleaned to remove any unwanted symbols, emojis and punctuation marks before the development of the system.

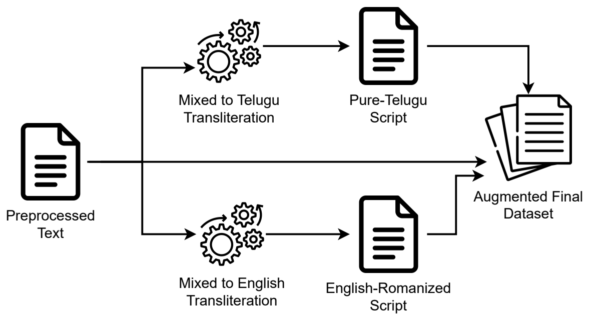

Figure 1: Augmentation process using transliteration

3.1 Data Augmentation with Transliteration

After cleaning, the dataset was augmented through transliteration. Transliteration handles code-mixed data since the dataset includes words in pure Telugu and English-romanized scripts. The proposed approach involves converting words between these scripts and translating code-mixed text to pure Telugu and English-romanized forms. Expanding vocabulary and providing context helps the model better understand the meaning of code-mixed text. We used AI4Bharat/IndicXlit[2] transformer model (Madhani et al., 2022) to first transliterate every single text from the cleaned dataset, to pure-Telugu text, as shown in Figure 1. Afterward, the cleaned texts were again transliterated, but this time to English-romanized text. These two new sets of texts are added to the original training dataset. In this way, every word in the original train dataset is guaranteed to appear at least twice in the augmented dataset: once in pure Telugu form and once in English-romanized form. This makes many common words reappear in both pure Telugu and English-romanized forms in the dataset, which ensures our model can learn from both forms of the exact text. The reason behind using this transliteration tool is that it has been trained on the Aksharantar (Madhani et al., 2023) dataset, which is, at the time of writing, the most extensive publicly available corpus on Indic languages. From the analysis of the authors of this tool, its performance in the Telugu language shows good potential. Hence, we chose this to generate synthetic transliterated texts.

4 Methodology

Figure 2 illustrates the overall process, including all employed models for the task. Various techniques are used to extract textual features for training the

[1] https://codalab.lisn.upsaclay.fr/competitions/16095

[2] https://pypi.org/project/ai4bharat-transliteration/

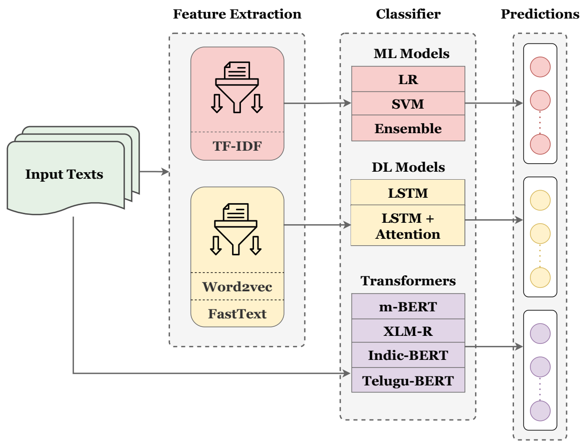

Figure 2: Abstract process of hate and offensive language detection in Telegu

respective ML and DL models. This work also utilized transformer-based techniques, such as m-BERT, XLM-R, Indic-BERT, and Telugu-BERT.

4.1 Feature Extraction

Following the approach of Tokunaga and Iwayama (1994), we utilized the TF-IDF (Term Frequency-Inverse Document Frequency) technique to extract unigram features. TF-IDF assigns word weights based on frequency within a document and across the entire corpus. This helps capture the importance of each word for distinguishing documents. We leveraged two widely used word embedding techniques for DL models: Word2Vec (Mikolov et al., 2013) and FastText (Grave et al., 2018). We implemented Word2Vec using the Keras embedding layer with an embedding dimension of 100 for both original and augmented datasets. Also, we utilized pre-trained FastText embedding matrices for each dataset. This pre-trained information incorporates subword information, potentially capturing more nuanced semantic relationships than Word2Vec.

4.2 ML Models

The 'lbgfs' optimizer was chosen for LR models, with C values 1.0 for both datasets. SVM models are configured with the linear kernel and $C = 1.0$ for both datasets.

4.3 DL Models

This research utilized a Bidirectional LSTM (BiLSTM) architecture with 100 cells per direction. To ensure robust generalization and prevent overfitting, we leveraged a dropout technique with a rate of 0.2. This technique randomly discards a small percentage of neurons during training. This promotes the network to learn relevant features across diverse data samples. Finally, the output of the BiLSTM layer is fed into a softmax layer for the final prediction. Inspired by the work of Vaswani et al. (2023), the DL model incorporated an attention mechanism to highlight impactful words in the input text. This attention layer, consisting of 20 neurons, was added to a BiLSTM layer. The BiLSTM output was then combined with the attention vector and fed into a softmax layer for the final prediction. We employed identical architectures for both datasets and utilized the 'sparse categorical cross-entropy' loss function. The model was trained with the 'Adam' optimizer (learning rate = $1e^3$, batch size = 32) for 15 epochs.

Transformer models often achieve the best results on various NLP benchmarks. Recognizing their strength, we employed four pre-trained transformer models: m-BERT (Devlin et al., 2019), XLM-R (Conneau et al., 2020), Indic-BERT (Kakwani et al., 2020) and Telugu-BERT (Joshi, 2023). All the transformer models were obtained from Huggingface[3] and fine-tuned with our original and augmented test set using the ktrain library (Maiya, 2020). In both datasets, we used a learning rate of $2e^{-5}$ in all the models and applied the method `fit_onecycle()` from the training library. The original dataset was trained with batch size 12, whereas the augmented dataset was trained with batch size 16 for all the models. Both datasets were used to fine-tune the transformer models for six epochs.

5 Results and Analysis

This section analyzes the effectiveness of different models for detecting hate speech and offensive language in Telugu code-mixed texts. The performance of the models is evaluated based on macro-averaged f_1-score. The evaluation result report is shown in Table 2, where the performance is presented in four cases: the original dataset (Non-Aug.), the original dataset augmented with Roman-transliterated texts (Roman-Aug.), the original dataset augmented with Telugu-transliterated texts (Telugu-Aug.), and finally, the intended, fully augmented dataset (Full-Aug.), which contains the combination of original, Roman-transliterated, and

[3]https://huggingface.co/docs/transformers/index

Classifiers	**Non-Aug.**			**Roman-Aug.**			**Telugu-Aug.**			**Full-Aug.**		
	P	**R**	**F**	**P**	**R**	**F**	**P**	**R**	**F**	**P**	**R**	**F**
LR	0.65	0.65	0.65	0.65	0.65	0.65	0.69	0.68	0.68	0.71	0.71	0.71
SVM	0.66	0.66	0.66	0.66	0.66	0.66	0.71	0.71	0.71	0.70	0.70	0.70
LSTM (Word2vec)	0.65	0.65	0.65	0.66	0.67	0.66	0.71	0.72	0.71	0.69	0.69	0.69
LSTM (FastText)	0.48	0.47	0.43	0.48	0.48	0.45	0.47	0.46	0.44	0.60	0.65	0.56
LSTM + Attention	0.62	0.62	0.62	0.62	0.62	0.62	0.64	0.64	0.64	0.67	0.67	0.67
m-BERT	0.67	0.67	0.67	0.65	0.65	0.65	0.71	0.71	0.71	0.72	0.72	0.72
XLM-R	0.73	0.72	0.72	0.53	0.52	0.51	0.72	0.72	0.72	0.76	0.75	0.75
Indic-BERT	0.67	0.66	0.66	0.67	0.66	0.66	0.69	0.68	0.68	0.67	0.66	0.65
Telugu-BERT	**0.74**	**0.73**	**0.73**	**0.73**	**0.73**	**0.73**	**0.77**	**0.76**	**0.76**	**0.78**	**0.77**	**0.77**

Table 2: Evaluation results obtained from models on four cases: original, only Roman-transliterated, only Telugu-transliterated, and fully-augmented test sets. P, R, and F denote macro-averaged precision, recall, and f_1 score. Here, "Aug." is used as shorthand for the word augmented.

Telugu-transliterated texts.

5.1 Discussion

The results indicated that ML models performed better than all DL models when using a fully augmented dataset. Word embeddings may not be able to capture the cultural subtleties and context of hate speech in code-mixed text as they are not trained on multilingual data. This might be a possible reason for such poor performance in DL classifiers. Still, the best result from ML and the best result from DL when applied to the Telugu-transliterated dataset was similar (0.71).

The transformer models' performance is outstanding compared to ML and DL models. All the transformer models except Indic-BERT for the original training dataset exceed the f_1 score previously obtained from ML and DL techniques. Among all the transformer models, Telugu-BERT shows the best result, with the highest f_1 score (0.73) achieved on the original train set and f_1 score of 0.77 in the fully augmented dataset. Although Indic-BERT was trained in Indic languages, it came last in performance, possibly due to many reasons. Alternatively, Telugu-English code mixed data might contain words in English rather than in the Telugu language written in a Roman script. Indic-BERT might need help finding word relationships better than XLM-Roberta or m-bert. Another reason might be because of data quality and context. XLM-Roberta and m-BERT, since they are trained on huge datasets, might capture hate-speech-related contexts better than Indic-BERT.

The results revealed the effect of various transliteration approaches. When only English-Romanized transliteration was applied, there needed to be more improvement in the performance. Compared to that, almost all classifiers performed better when Telugu transliteration was applied. Finally, in the fully augmented dataset, where both English-Romanized transliteration and Telugu transliteration were combined, we can see most of these classifiers show a slight performance improvement, which can be because this dataset is more extensive than all of the datasets above. Hence, relating different words between Roman script and Telugu script is easier. Also, the fully augmented dataset might give better context to the classifier due to many similar words appearing in both languages, and the two-way transliteration makes all forms of the similar words known to the classifiers. On the other hand, the IndicXlit transliteration tool is not perfect, and it might not always produce the best transliteration of every single word, so it hurts the performance.

The results showed that the proposed approach produces better f_1 scores when trained under the augmented dataset. The Telugu-BERT model trained on the augmented train set was chosen as the best overall performer.

5.2 Error Analysis

We conducted a comprehensive error analysis of quantitative and qualitative ways to gain an in-depth understanding of the best-performing model (Telugu-BERT).

Quantitative Analysis: It is shown that the best-performing model (Telugu-BERT) produced a maximum f_1 score of 0.77 (Table 2). Figure 3 represents the confusion matrix of this model. Among

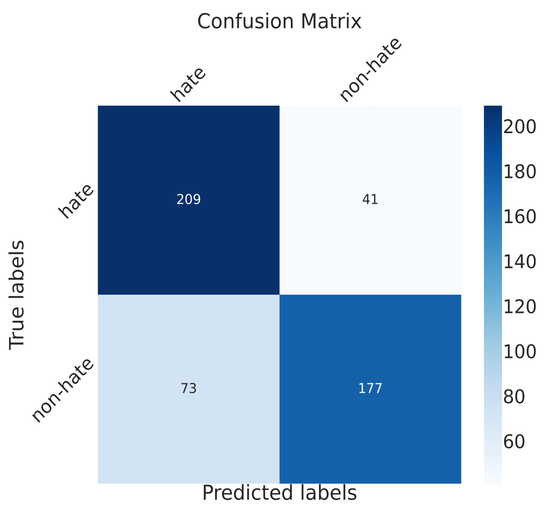

Figure 3: Confusion matrix of the best model (Telugu-BERT) on fully augmented dataset

Text	Actual	Predicted
ఎన్ని సార్లు అయిన వినాలని ఉంది చిట్టి తల్లి సూపర్ మా,, (How many times do you want to hear Chitti Mata Super Maa?)	non-hate	non-hate
ఇది బెండపూడి గవ్నమెంట్ స్టూడెంట్స్ కి మాత్రమే సాధ్యం. (This is only possible for Bendapudi Government students.)	hate	non-hate
dennichusthe chi daridram (If you look at this, you are poor)	hate	hate
నిజంగానే బుద్ధిలేకుండా గెలిపించుకున్నాము worst CM (We really elected the worst CM without any intelligence.)	non-hate	hate
ఇప్పుడు YCP ki antha seen ledhu బులుగు batch dhammunte yedhi prnchakunda వుంటారా (The YCP doesn't have much of a chance now. Will they not be scared if the Bulugu batch attacks?)	hate	hate

Figure 4: Predicted outputs for some sample texts using the proposed (Telugu-BERT) model on fully augmented dataset

the two classes, the hate class has the highest True-Positive Rate (TPR) of 83.6%, while the non-hate class has a lower TPR of 70.8%. This might indicate a slight bias towards detecting hate speech since a higher percentage of non-hate text (29.2%) was predicted to be from the hate speech class. Compared to that, 16.4% of actual offensive texts were predicted to be not offensive.

Qualitative Analysis: Figure 4 represents some predictions of the best-performing model compared to the actual labels. We generated the translations shown in the figure with the help of Google Translate[4]. The first sample was predicted accurately, entirely made up of pure Telugu script. Although the second sample had only Telugu script, it was not predicted accurately since it was a hate or offensive text, but Telegu-BERT labeled it *non-hate*. The third script was comprised of Telugu text in an English-Roman script, and Telegu-BERT was able to label it accurately. The fourth sample was of a code-mixed text, which mostly has text from Telugu script, with a minimal amount of English script. It was not predicted accurately. The final sample also consists of code-mixed text, with an English-script majority, but our model was able to predict its label accurately.

Limitations

The presented approach relies heavily on transliteration accuracy. Errors in transliteration can introduce misleading information, which can impact model performance. Furthermore, the diverse ways to transliterate a word in real-world scenarios highlight the importance of exploring methods to incorporate multiple transliteration options.

[4]https://translate.google.com/

6 Conclusion

This paper explored various ML, DL, and transformer-based approaches for hate and offensive language detection in Telugu code-mixed text and demonstrated the effectiveness of transliteration-augmented datasets. In most cases, transformer models outperformed ML and DL methods. Telugu-BERT, trained explicitly on Telugu text, achieved an impressive f_1 score of 0.77. Investigating further augmentation methods and their combinations presents an exciting future direction to enhance the dataset and improve performance.

References

Sindhu Abro, Sarang Shaikh, Zahid Hussain Khand, Ali Zafar, Sajid Khan, and Ghulam Mujtaba. 2020. Automatic hate speech detection using machine learning: A comparative study. *International Journal of Advanced Computer Science and Applications*, 11(8).

Premjth B, Bharathi Raja, Prasanna Kumar Kumaresan, Saranya Rajiakodi, Sai Prashanth Karnati, Sai Rishith Reddy Mangamuru, and Janakiram Chandu. 2024. Findings of the shared task on hate and offensive language detection in telugu codemixed text (hold-telugu). In *Proc. of the Fourth Workshop on Speech, Vision, and Language Technologies for Dravidian Languages*, Malta. European Chapter of the Association for Computational Linguistics.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale.

Mithun Das, Somnath Banerjee, Punyajoy Saha, and Animesh Mukherjee. 2022. Hate speech and offensive language detection in bengali. *arXiv preprint arXiv:2210.03479*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding.

Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov. 2018. Learning word vectors for 157 languages. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Adeep Hande, Ruba Priyadharshini, and Bharathi Raja Chakravarthi. 2020. KanCMD: Kannada CodeMixed dataset for sentiment analysis and offensive language detection. In *Proceedings of the Third Workshop on Computational Modeling of People's Opinions, Personality, and Emotion's in Social Media*, pages 54–63, Barcelona, Spain (Online). Association for Computational Linguistics.

Raviraj Joshi. 2023. L3cube-hindbert and devbert: Pre-trained bert transformer models for devanagari based hindi and marathi languages.

Divyanshu Kakwani, Anoop Kunchukuttan, Satish Golla, Gokul N.C., Avik Bhattacharyya, Mitesh M. Khapra, and Pratyush Kumar. 2020. IndicNLPSuite: Monolingual Corpora, Evaluation Benchmarks and Pre-trained Multilingual Language Models for Indian Languages. In *Findings of EMNLP*.

Yash Madhani, Sushane Parthan, Priyanka Bedekar, Gokul NC, Ruchi Khapra, Anoop Kunchukuttan, Pratyush Kumar, and Mitesh M. Khapra. 2023. Aksharantar: Open indic-language transliteration datasets and models for the next billion users.

Yash Madhani, Sushane Parthan, Priyanka A. Bedekar, Ruchi Khapra, Vivek Seshadri, Anoop Kunchukuttan, Pratyush Kumar, and Mitesh M. Khapra. 2022. Aksharantar: Towards building open transliteration tools for the next billion users. *ArXiv*, abs/2205.03018.

Arun S. Maiya. 2020. ktrain: A low-code library for augmented machine learning. *arXiv preprint arXiv:2004.10703*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality.

Ruba Priyadharshini, Bharathi Raja Chakravarthi, Malliga S, Subalalitha CN, Kogilavani S V, Premjith B, Abirami Murugappan, and Prasanna Kumar Kumaresan. 2023. Overview of shared-task on abusive comment detection in tamil and telugu. In *Proceedings of the Third Workshop on Speech and Language Technologies for Dravidian Languages*, Varna, Bulgaria. Recent Advances in Natural Language Processing.

Omar Sharif, Eftekhar Hossain, and Mohammed Moshiul Hoque. 2021. Nlp-cuet@ dravidianlangtech-eacl2021: Offensive language detection from multilingual code-mixed text using transformers. *arXiv preprint arXiv:2103.00455*.

Shashi Shekhar, Hitendra Garg, Rohit Agrawal, Shivendra Shivani, and Bhisham Sharma. 2023. Hatred and trolling detection transliteration framework using hierarchical lstm in code-mixed social media text. *Complex & Intelligent Systems*, 9(3):2813–2826.

Takenobu Tokunaga and Makoto Iwayama. 1994. Text categorization based on weighted inverse document frequency.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2023. Attention is all you need.

Bencheng Wei, Jason Li, Ajay Gupta, Hafiza Umair, Atsu Vovor, and Natalie Durzynski. 2021. Offensive language and hate speech detection with deep learning and transfer learning. *arXiv preprint arXiv:2108.03305*.