IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940



International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 10, October 2026

DOI: 10.17148/IJARCCE.2026.151074

Comparative Analysis of Deep Learning Models for Offensive Comment Detection in Code-Mixed Dravidian Languages, with a Focus on Tamil-English

Maria Sarah J[1], Dr. Golda Dilip[2]

M.Tech (Computer Science and Engineering), Department of Computer Science and Engineering, SRM Institute of Science and Technology, Vadapalani Campus, Chennai, Tamil Nadu, India[1]

Professor, Department of Computer Science and Engineering, SRM Institute of Science and Technology, Vadapalani Campus, Chennai, Tamil Nadu, India[2]

Abstract: Offensive and hate speech detection on social media has become an important natural language processing problem because of the scale at which harmful user-generated content is produced and the practical impossibility of moderating it manually. The problem is considerably harder for code-mixed Tamil-English text, where Tamil words are frequently transliterated into Roman script, spelling is inconsistent, and sentence structure borrows freely from both languages, all of which limit the effectiveness of models built primarily for monolingual, formally written text. This paper presents a comparative review of fifteen system-description papers submitted to the DravidianLangTech-EACL2021 shared task on offensive language identification in Tamil, Kannada and Malayalam code-mixed text, evaluated on the Dravidian Offensive Language Dataset released for that task, with the base paper SJ_AJ@DravidianLangTech-EACL2021 used as the primary point of reference. The fifteen studies are compared on dataset, preprocessing, feature representation, model architecture, training strategy and reported evaluation metric, rather than evaluated in isolation. The review finds that classical machine learning with TF-IDF or character n-gram features, non-transformer deep learning architectures such as BiLSTM and CNN-BiLSTM hybrids, and pretrained multilingual transformers such as mBERT and XLM-RoBERTa, often strengthened with ensembling, task-adaptive pretraining, transliteration or class-imbalance handling, together form the dominant methodological landscape for this task. Reported weighted F1-scores vary considerably across the three languages and techniques, with a severe class imbalance in the underlying dataset (over 70% of Tamil-English comments labelled Not-offensive) repeatedly identified as a limiting factor. The review further finds that interpretability of model predictions is addressed only marginally, mostly through confusion-matrix error analysis, and that no reviewed study applies a dedicated explainability technique. This observation motivates the proposed direction of incorporating explainable deep learning methods into Tamil-English code-mixed hate speech detection, so that model decisions can be examined for correctness as well as for the reasoning behind them.

Keywords: Hate Speech Detection; Code-Mixed Text; Tamil-English; Offensive Language Identification; Transformer Models; Explainable AI; Comparative Study

1. INTRODUCTION

Automatic identification of hate speech and offensive language has become an important research problem in natural language processing, mainly because unmoderated offensive content on social media platforms can cause real psychological and social harm while manual moderation at the scale of modern platforms is practically impossible. Social media text adds further difficulty to this classification problem because it is short, informal, grammatically loose, and frequently contains spelling errors, abbreviations, emojis and slang that differ from the standard written form of a language [1].

This difficulty becomes considerably more severe when the text is code-mixed, as is the case with Tamil-English comments collected from platforms such as YouTube. In code-mixed Tamil-English text, words and phrases from Tamil and English are freely combined within the same sentence, and Tamil words are frequently written using the Roman script rather than native Tamil script, a practice commonly referred to as transliteration. Several of the reviewed papers explicitly note that the same Tamil word may appear in multiple spelling variants across comments, and that pretrained language models built mainly for monolingual or formally written text struggle to construct a reliable vocabulary and consistent word representation under such conditions [7], [13]. Combined with informal sentence construction, the

 This work is licensed under a Creative Commons Attribution 4.0 International License

DOI: 10.17148/IJARCCE.2026.151074

absence of a fixed grammar for code-mixed writing, and a label distribution weighted heavily toward the non-offensive class, this creates considerable ambiguity for any classification system attempting to separate offensive from non-offensive content, and it forms the underlying motivation for comparing the approaches that have already been applied to this task.

The fifteen papers reviewed in this study were all submitted to the DravidianLangTech-EACL2021 shared task on offensive language identification in Tamil, Kannada and Malayalam code-mixed text, built on the Dravidian Offensive Language Dataset [1], [17]. Existing work on this task spans classical machine learning pipelines, non-transformer deep learning architectures, and pretrained multilingual transformer models, frequently combined with auxiliary techniques such as ensembling, task-adaptive pretraining, transliteration and class-imbalance correction. Despite addressing the same underlying task and largely the same dataset, the fifteen studies differ enough in methodology, preprocessing and evaluation protocol that a structured comparison is necessary before any conclusion can be drawn about which type of approach is more suitable for Tamil-English code-mixed offensive-language detection. A reading of these papers together also shows that nearly all of them concentrate on improving aggregate classification performance, while very few examine why a model classifies a given comment the way it does.

This paper makes three contributions. First, it organizes and compares the fifteen DravidianLangTech-EACL2021 system papers by approach family, dataset handling and special technique rather than summarizing them individually. Second, it compares their reported results while making explicit the methodological differences, such as dataset size, class distribution and evaluation metric, that limit direct numerical comparison between them, and it grounds this comparison in the actual class distribution of the underlying Tamil-English dataset. Third, it identifies the extent to which explainability has been addressed in this body of work and uses that finding to motivate a proposed direction for Tamil-English code-mixed hate speech detection built around explainable deep learning. The remainder of the paper is organized as follows: Section 2 describes the dataset; Section 3 reviews related work by approach family; Section 4 presents the literature survey; Section 5 describes the comparison methodology; Section 6 presents the comparative analysis; Section 7 discusses the reported results; Section 8 presents the comparative tables; Section 9 evaluates the models implemented in this study; Section 10 discusses the findings; Section 11 develops the research gap; Section 12 outlines the proposed study; and Section 13 concludes the paper.

2. DATASET DESCRIPTION

2.1 Dataset Overview

All fifteen reviewed papers use data drawn from the Dravidian Offensive Language Dataset released for the Offensive Language Identification in Dravidian Languages shared task (Chakravarthi et al., EACL 2021 DravidianLangTech workshop) [1], [17]. The dataset consists of YouTube comments in three code-mixed language pairs, Tamil-English, Kannada-English and Malayalam-English, manually annotated into up to six classes: Not-offensive, Offensive-Untargeted, Offensive-Targeted-Insult-Individual, Offensive-Targeted-Insult-Group, Offensive-Targeted-Insult-Other, and a Not-in-intended-language class for comments not actually written in the target language.

The Tamil-English subset, which is the primary focus of the proposed study, contains 35,139 comments in the training split, 4,388 in the development split and 4,392 in the test split, a total of 43,919 comments. The comments were collected from public YouTube posts and comprise naturally occurring code-mixed Tamil-English text, including Tamil words written in Roman script, mixed-script sentences, and informal spelling, which is the same source of difficulty discussed for the fifteen reviewed papers in Section 1.

Table 3 places the Tamil-English subset alongside the Kannada-English and Malayalam-English offensive-language subsets, using the row counts of the released split files. Tamil-English is the largest of the three, with a training split about 5.7 times the size of the Kannada-English training split and about 2.2 times the size of the Malayalam-English one. The released Malayalam-English files use five labels rather than six, because they contain no Offensive-Targeted-Insult-Other comments, whereas the Tamil-English and Kannada-English files use all six.

2.2 Class Distribution

Table 2 gives the per-class distribution of the Tamil-English subset across the three splits, obtained directly from the released dataset files. The distribution is heavily skewed: 72.4% of the combined comments are labelled Not-offensive, while the four offensive sub-classes together account for only about 23.5% of the data (Offensive-Untargeted 8.3%, Offensive-Targeted-Insult-Group 7.1%, Offensive-Targeted-Insult-Individual 6.8%, and Offensive-Targeted-Insult-Other just 1.3%), and the remaining 4.1% are Not-Tamil comments. Offensive-Targeted-Insult-Other is the smallest class

 This work is licensed under a Creative Commons Attribution 4.0 International License

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 10, October 2026

DOI: 10.17148/IJARCCE.2026.151074

in absolute terms, with only 590 comments across all three splits. This imbalance is consistent with the class-imbalance limitation reported across several of the fifteen reviewed papers (Section 6.4) and is shown graphically in Fig. 1.

The class proportions are almost identical across the three splits (Not-offensive 72.4% in training, 72.8% in development and 72.6% in test; Offensive-Targeted-Insult-Other 1.3%, 1.5% and 1.6%), so the imbalance is a property of the corpus and is not introduced by the split. The largest class (31,808 comments) is about 54 times the size of the smallest (590 comments). Exact-string matching on the released Tamil-English files, carried out for this review without any normalisation, also shows that 306 comment strings occur more than once (706 rows in total), 20 of them with conflicting labels, and that 69 of the 4,392 test rows (1.6%) and 62 of the 4,388 development rows (1.4%) have a comment string identical to one in the training split. These overlaps are small relative to the size of the corpus, but they are worth controlling for in any evaluation on it.

2.3 Annotation Guidelines

Each comment in the dataset was manually annotated with exactly one label. A comment is marked Not-offensive when it contains no abusive, threatening or demeaning content directed at a person, group or entity. The four offensive labels distinguish the target of the offence: Offensive-Untargeted covers general profanity or abusive language not directed at any specific individual or group; Offensive-Targeted-Insult-Individual covers offence directed at a specific named or identifiable person; Offensive-Targeted-Insult-Group covers offence directed at a community, organisation or group of people; and Offensive-Targeted-Insult-Other covers targeted offence that does not fit the individual or group categories. A sixth label, Not-Tamil (or the equivalent Not-Kannada/Not-Malayalam label for the other two subsets), is applied to comments that are not actually written in the intended code-mixed language pair, for example comments written entirely in English or in an unrelated language. None of the fifteen reviewed papers reports its own independent annotation process; all fifteen use the labels as released with the shared-task dataset. Table 1 illustrates each of the six labels with two comments from the training split, reproduced as released, together with an English translation and the basis on which the label fits.

Table 1: Annotation Label Scheme of the Tamil-English Subset [17]

Label	Annotation criterion	Examples (verbatim comments with English translation)
Not_offensive	No abusive, threatening or demeaning content directed at a person, group or entity.	**Comment:** “Padam Hittuuuuu.... vera onum solrathukku ilaaaaaa” **Translation:** *The film is a hit... there is nothing else to say.* **Comment:** “சில படத்தோட டிரெய்லர் லயே படம் ஹிட்டா இல்லையானு சொல்லிறலாம் ... இந்த படம் sure shot ...” **Translation:** *With some films you can tell from the trailer itself whether they will be a hit or not... this film is a sure shot.*
Offensive_Untargeted*	General profanity or abusive language not directed at any specific individual or group.	**Comment:** “semma mokka idellam oru padam edukurawanauku adikanam seruppala” **Translation:** *Total rubbish; whoever makes a film like this should be hit with a slipper.* **Comment:** “அட அரைவேக்காட்டு பயலே :-D :-D” **Translation:** *Hey, you half-baked fool! :-D :-D*
Offensive_ Targeted_Insult_ Individual	Offence directed at a specific named or identifiable person.	**Comment:** “Vetrimaran unakku ratham, kolai, aruval ithu illamma padam edukka theriyatha ???” **Translation:** *Vetrimaran, can you not make a film without blood, murder and machetes???* **Comment:** “என்னடா மேக்கப் மேன் சிங்கத்துக்கு மேக்கப் போட சொன்னா முயல் குட்டிக்கு போட்டு வச்சு இருக்க!!!!!!!” **Translation:** *What, make-up man! We asked you to make up a lion and you have made up a baby rabbit!!!!!!!*

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940



International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 10, October 2026

DOI: 10.17148/IJARCCE.2026.151074

Label	Annotation criterion	Examples (verbatim comments with English translation)
Offensive_ Targeted_Insult_ Group	Offence directed at a community, organisation or group of people.	**Comment:** "Politicians just seruppadi kudukum this Movie" **Translation:** *This movie will just give the politicians a sandal-beating.* **Comment:** "ஜாதி இல்லனு சொல்லிட்டு தேவைப்படுற இடத்துல மட்டும் பயன்படுத்திற நாய்ங்கலாம் இங்க டிஸ்லைக் போட்ருக்கிங்க போல" **Translation:** *Those dogs who say there is no caste but use it only where it suits them seem to have put the dislikes here.*
Offensive_ Targeted_Insult_ Other	Targeted offence that fits neither the individual nor the group category.	**Comment:** "Yaarum keela scroll pannirathinga like kettu oru kootam suthuthu" **Translation:** *Nobody scroll down; a crowd is going around begging for likes.* **Comment:** "13k dislikes இந்தநாடு நாசமாப்போகவேன்டும் என்று நினைப்பவர்கலிடமிறுந்து" **Translation:** *13k dislikes, from people who want this country to be ruined.*
not-Tamil	Comment not written in the intended code-mixed language pair, e.g. entirely English or another language.	**Comment:** "Ye movie hindi me bhi release karo" **Translation:** *Release this movie in Hindi too.* **Comment:** "youngster political movie | இளைஞர்களின் அரசியல் படம் | യങ്സ്റർ പൊലിടിക്കല് പടം" **Translation:** *"Youngster political movie", repeated in English, Tamil and Malayalam.*

Two comments per label, reproduced verbatim from the training split of the released files [17] (spelling, casing, punctuation and script unaltered) and chosen to illustrate the criteria listed in the same table. The English translations and the "Annotation criterion" column are this paper's own reading of the comments against those criteria; they are not annotator notes, since the repository contains no annotation manual, and labels are as released. Offensive wording is reproduced as it appears in the data, for research purposes only. * Spelled Offensive_Untargetede in the released files.

Provenance of the guidelines. The repository supplied with the corpus [17] describes the data as YouTube comments annotated manually by volunteer annotators, reports high inter-annotator agreement measured with Krippendorff's alpha, and ships the labelled files, but it does not include a standalone annotation manual. The guideline summary in this section is therefore limited to what the released files and the repository description establish: the label set and its hierarchy (Tables 1 and 2), the one-label-per-comment structure, and the per-class counts. The class criteria are those given in the paragraph above, and no further annotator instructions are assumed.

Label hierarchy. Table 1 lists the six labels as they are encoded in the released files, with each label's criterion and examples, and Table 2 gives its count and share. Reading the label names, the scheme separates comments that are not in the intended language (not-Tamil) from in-language comments; in-language comments are either Not_offensive or offensive, and offensive comments are either untargeted or targeted insults aimed at an individual, a group or another target. The order in which annotators applied these distinctions is not recorded in the repository, so the tables present a label hierarchy rather than a decision procedure.

File conventions. Three conventions of the released Tamil-English files matter for anyone reusing the labels. First, every row of the train, development and test files carries exactly one label, with the comment in the first column and the label in the second; the combined file lists the label first and the comment second. Second, the untargeted offensive class is spelled Offensive_Untargetede in all Tamil-English files, a spelling that this paper normalises to Offensive_Untargeted. Third, the out-of-language label differs by subset (not-Tamil, not-Kannada and not-malayalam), as shown in Table 3.

2.4 Corpus Characteristics

Table 4 summarises the script and length of the Tamil-English comments, computed for this review from the released files. A comment is counted as Tamil-script if it contains any character from the Tamil Unicode block (U+0B80–U+0BFF) and as Roman-script if it contains any basic Latin letter. On this basis 80.8% of the 43,919 comments are written only in Roman script, 14.5% only in Tamil script and 4.5% mix the two; 68 comments (0.2%) contain neither, and 66 of these are labelled not-Tamil. The Roman-script group cannot be separated further by script alone, because it contains both English and transliterated Tamil, so its share is an upper bound on the amount of transliterated text. This is the kind of variation that the transliteration-based techniques reviewed in Section 3.4 are intended to address.

DOI: 10.17148/IJARCCE.2026.151074

The script mix differs by class. Of the Offensive-Untargeted comments, 95.2% are Roman-only (3,456 of 3,630), compared with 69.0% of the Offensive-Targeted-Insult-Group comments (2,167 of 3,140), whose Tamil-script-only share (23.4%) is the highest of any class. Comments are short: the mean length is 10.4 whitespace-separated words and the median is 8 (maximum 127), with Offensive-Targeted-Insult-Group comments the longest (mean 15.0, median 11). A total of 2,011 comments (4.6%) contain at least one emoji or pictograph character (Unicode ranges U+1F300–U+1FAFF and U+2600–U+27BF). These descriptive statistics are not tied to any reviewed paper's results; they are included so that the preprocessing choices compared in Section 6, such as the emoji handling of Hypers [16] and the transliteration of SJ_AJ [7] and OFFLangOne [13], can be read against the actual composition of the corpus.

3. RELATED WORK

All fifteen papers reviewed here were submitted to the DravidianLangTech-EACL2021 shared task described in Section 2 [1]. Rather than describing each paper in isolation, this section groups the fifteen studies by the type of approach they adopt.

3.1 Feature-Based and Traditional Machine Learning Approaches

A small number of the reviewed systems rely on sparse, hand-engineered text representations combined with classical classifiers. IRNLP_DAIICT [8] compares TF-IDF character n-gram vectors against pretrained MuRIL sentence embeddings as input to Logistic Regression, Support Vector Machine and Random Forest classifiers, treating this comparison as a way of testing whether a pretrained multilingual embedding offers an advantage over a purely statistical representation without any fine-tuning. NLP-CUET [4] similarly reports Logistic Regression and SVM baselines using n-gram style features before moving on to deep learning and transformer models, allowing a direct within-paper comparison between classical and neural approaches on the same data split.

3.2 Deep Learning Approaches Without Pretrained Transformers

Several papers apply neural architectures trained from scratch on the shared-task corpus rather than relying on large pretrained language models. Amrita_CEN_NLP [15] proposes a hybrid CNN-BiLSTM-Dense network together with standalone BiLSTM and BiRNN models, combined with cost-sensitive learning to address class imbalance, and reports this as a deliberately non-transformer alternative. NLP-CUET [4] also experiments with LSTM and LSTM with attention before introducing transformer-based models, and IIITT [6] includes a CNN-BiLSTM configuration among its six compared architectures. These papers generally treat non-transformer deep learning as a baseline against which transformer-based fine-tuning is subsequently measured.

3.3 Transformer-Based and Multilingual Pretrained Models

The majority of the fifteen papers fine-tune one or more pretrained multilingual transformer encoders. Multilingual BERT (mBERT) and XLM-RoBERTa (XLM-R) are the most frequently used encoders, appearing either as the sole classifier or as part of a larger pipeline in Hate-Alert [2], indicnlp@kgp [3], NLP-CUET [4], DLRG [5], IIITT [6], the base paper SJ_AJ [7], cs@DLT [10], ZYJ123 [11], Codewithzichao [12], OFFLangOne [13] and Hypers [16]. Several papers additionally use encoders pretrained specifically on Indian languages, namely IndicBERT (Hate-Alert [2], NLP-CUET [4], Bitions [14]) and MuRIL (Hate-Alert [2], IRNLP_DAIICT [8]), as well as DistilBERT or DistilmBERT for a lighter-weight alternative (Hate-Alert [2], IIITT [6], Bitions [14]). CUSATNLP [9] instead uses LaBSE, a language-agnostic sentence embedding model, paired with a softmax classifier, while IIITT [6] also reports an ALBERT configuration and an AWD-LSTM-based ULMFiT model, the latter also used by indicnlp@kgp [3] and Bitions [14] for transfer learning from an English offensive-language dataset.

3.4 Task-Adaptive Pretraining and Transliteration

A smaller set of papers modifies the pretraining or input representation stage rather than only fine-tuning an off-the-shelf encoder. The base paper, SJ_AJ [7], performs task-adaptive masked-language-model pretraining of mBERT and XLM-R on the unlabelled shared-task corpus before fine-tuning, and additionally experiments with transliteration and a character-level BiLSTM fusion branch. OFFLangOne [13] takes a related but distinct approach, back-transliterating non-English tokens into their native script before fine-tuning mBERT and combining this with a class-balanced loss. These two papers represent the only reviewed studies that alter the pretraining process or script representation specifically to address the code-mixed nature of the data.

3.5 Class-Imbalance Handling and Ensemble Strategies

Because the shared-task labels are heavily skewed toward the not-offensive class, as confirmed for Tamil-English in Table 2, a majority of the transformer-based papers introduce some form of imbalance correction or ensembling, or both. DLRG [5] applies SOUP-based oversampling and undersampling; Codewithzichao [12] combines inverse-frequency

 This work is licensed under a Creative Commons Attribution 4.0 International License



International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 10, October 2026
DOI: 10.17148/IJARCCE.2026.151074

class weighting, focal loss and FGM adversarial training; OFFLangOne [13] uses an effective-number class-balanced loss; Bitions [14] combines class-weighting, focal loss and pseudo-labelling of additional unlabelled data; and Amrita_CEN_NLP [15] applies cost-sensitive learning. On the ensembling side, Hate-Alert [2] combines six transformer and fusion models using a genetic-algorithm-optimised weighted vote; indicnlp@kgp [3] averages predictions from ULMFiT, mBERT and XLM-R; SJ_AJ [7] uses a majority vote across six models; ZYJ123 [11] integrates XLM-R with a DPCNN head and combines five-fold cross-validation with voting; Codewithzichao [12] ensembles XLM-R and mBERT variants; and Bitions [14] uses a soft-vote ensemble of DistilmBERT and ULMFiT. These two families of technique, class-imbalance correction and ensembling, are the most consistently applied enhancements across the reviewed literature, and are discussed further in Section 6.4.

4. LITERATURE SURVEY

Across the fifteen studies, pretrained multilingual encoders dominate, and the highest reported scores combine an encoder with ensembling, cross-corpus transfer or domain adaptation rather than a single model. Classical TF-IDF and character n-gram classifiers are competitive only on the larger, less skewed Malayalam data (0.94–0.95 in [2], [3], [8]) and fall behind on Tamil and Kannada. Neural networks trained from scratch [4], [6], [15] lag pretrained models because the shared-task corpus is small, whereas pretraining, even on synthetic code-mixed text, narrows the gap: ULMFiT improved on a plain LSTM in [3] and was the best Tamil and Malayalam system in [6]. Among transformers, XLM-R is equal to or slightly better than mBERT [2], [4], [12], while encoders pretrained on Indian languages (IndicBERT, MuRIL) did not show the expected advantage [2], [8], [14].

Strategies aimed at the code-mixed nature of the data give mixed evidence. Task-adaptive pretraining raised development F1 by about 0.2–0.8 points on Tamil and up to 2.2 points on Malayalam [7], whereas romanisation of the input was not consistently helpful [7] and back-transliteration to native script [13] failed for Kannada. Pooling the three corpora or adding English OLID data helped most on the smallest corpus, Kannada [3], [12]. Imbalance remedies (SOUP [5], class weights and focal loss [12], [14], class-balanced loss [13], cost-sensitive learning [15]) show little measurable benefit: in [13] they change macro-F1 by only 0.01–0.02, and minority classes still score near zero in [4], [5] and [9]. Ensemble gains of 0.01–0.02 [2], [12] are small compared with the 15–20% seed variance reported for XLM-R in [2].

The reported results also need careful reading. They come from different splits and settings: [7] reports only development scores, the split is unclear in [6] and [13], and macro-F1 is reported by only [5], [9] and [13], so the Tamil weighted F1 of 0.64–0.80 hides much lower macro scores (0.31–0.48). Most importantly for the present study, all fifteen papers evaluate predictive performance only. Interpretability goes no further than confusion matrices [4], [9] and class-wise scores [5], [9], and explanations offered for results, such as Wikipedia coverage [16] or the benefit of back-transliteration [13], are not tested. No dedicated explainability technique is applied in this body of work, which motivates the explainable approach proposed in Section 12.

5. REVIEW METHODOLOGY

The fifteen papers are compared using a fixed set of criteria rather than being read and summarized independently, so that similarities and differences between studies can be identified systematically. The criteria used are: the dataset and task addressed (including which of the three languages are covered); the preprocessing applied to the raw text; the feature representation or embedding used as input to the classifier; the model or architecture family; any special training strategy, such as task-adaptive pretraining, transliteration or class-imbalance handling; whether the system is a single model or an ensemble, and if so how the ensemble is combined; the evaluation metric reported, which for this shared task is predominantly weighted F1-score; the numerical result reported for each language; and the strengths and limitations stated or evident from each paper.

This comparison is not intended only to rank the fifteen systems by reported score. Because the papers differ in train-test split and, in some cases, the exact metric reported, their numerical results are not always directly interchangeable. The methodology adopted here treats the reported figures as indicative of relative trends within the shared-task setting rather than as a strictly controlled benchmark, and explicitly flags cases where comparison is affected by such differences. The comparison is also used to understand methodological choices, such as why a particular paper chose transliteration over oversampling, or an ensemble over a single fine-tuned model, rather than only to read off which system scored highest.

 This work is licensed under a Creative Commons Attribution 4.0 International License

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940



International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 10, October 2026

DOI: 10.17148/IJARCCE.2026.151074

6. COMPARATIVE ANALYSIS

6.1 Traditional Machine Learning Approaches

Within the fifteen reviewed papers, purely traditional machine learning is used only as a point of comparison rather than as the primary system. IRNLP_DAIICT [8] reports that TF-IDF character n-gram features combined with Logistic Regression, SVM or Random Forest achieve weighted F1-scores of 0.64 (Kannada), 0.71 (Tamil) and 0.95 (Malayalam), figures that are competitive with several transformer-based systems on Malayalam but noticeably behind the stronger transformer or ensemble systems on Tamil and Kannada. The paper also reports that substituting the TF-IDF features with pretrained MuRIL embeddings, while still using classical classifiers without fine-tuning, does not produce a consistent improvement across all three languages. NLP-CUET [4] reports a similar pattern, where its SVM and Logistic Regression baselines are outperformed by its own LSTM and transformer configurations on the same data. The main advantage of the traditional approaches in these papers is computational simplicity and the absence of any GPU-based training requirement; their main limitation is an inability to capture contextual or cross-lingual information, which both papers identify as a factor in their comparatively lower performance on Tamil and Kannada.

6.2 Deep Learning Approaches

Non-transformer deep learning architectures occupy a middle position in the reviewed literature. Amrita_CEN_NLP [15] reports weighted F1-scores of 0.85 (Malayalam), 0.62 (Tamil) and 0.58 (Kannada) using its CNN-BiLSTM-Dense hybrid with cost-sensitive learning, values that are lower than most transformer-based systems on Tamil and Kannada but not far behind several of them on Malayalam. The paper attributes part of this gap to the absence of any pretrained language representation, meaning the model must learn word and sub-word semantics entirely from the comparatively small shared-task corpus. IIITT [6] includes a CNN-BiLSTM configuration among its six compared models and reports it performing below its transformer-based configurations for the same language. NLP-CUET [4] reports that its LSTM and LSTM-with-attention models improve over its own classical baselines but remain behind its transformer results, and additionally notes that severe class imbalance causes near-zero recall on minority offensive classes regardless of whether a classical, recurrent or transformer-based classifier is used. These findings indicate that recurrent and convolutional architectures without pretraining can approach transformer-level performance on the larger, less imbalanced Malayalam dataset, but fall further behind on Tamil and particularly Kannada, where training data is comparatively scarce.

6.3 Transformer-Based Approaches

Transformer-based fine-tuning is the dominant approach among the fifteen papers and also the approach associated with the overall highest reported scores. Hate-Alert [2] reports weighted F1-scores of 0.78 (Tamil), 0.75 (Kannada) and 0.97 (Malayalam) using a genetic-algorithm-weighted ensemble of XLM-R, mBERT, IndicBERT, MuRIL, DistilBERT and a CNN-BERT fusion model, the highest reported Kannada score and among the highest Tamil and Malayalam scores among the fifteen papers, though the paper does not explain why IndicBERT and MuRIL individually underperform the other encoders despite being pretrained on Indian languages. indicnlp@kgp [3] reports comparable results (0.97 Malayalam, 0.77 Tamil, 0.72 Kannada) using an averaging ensemble of ULMFiT, mBERT and XLM-R combined with transfer learning from an English offensive-language dataset. The base paper, SJ_AJ [7], reports 0.9676 (Malayalam), 0.7694 (Tamil) and 0.7030 (Kannada) using task-adaptive pretraining of mBERT and XLM-R together with a character-BiLSTM fusion branch and a majority-vote ensemble, and states that task-adaptive pretraining yields a gain of roughly one to two percentage points of F1 over directly fine-tuning the same encoders without it. In contrast, DLRG [5], which fine-tunes a single mBERT model with SOUP oversampling and does not use an ensemble, reports a considerably lower weighted F1-score of 0.64 on Tamil, with zero F1 on two of the minority offensive categories. CUSATNLP [9], using LaBSE embeddings without any additional preprocessing, reports 0.95 (Malayalam) but only 0.62 (Kannada) and 0.69 (Tamil). Fig. 2 shows these fifteen Tamil scores side by side, grouped by approach category; the highest Tamil score (OFFLangOne, 0.80) and several of the next-highest scores belong to systems that combine a transformer with either transliteration, class-balanced loss, or ensembling, while the lowest scores belong to single-model or non-transformer systems.

6.4 Additional Techniques

Several techniques recur across the transformer-based papers, each intended to address a specific weakness confirmed by the dataset statistics in Table 2. Task-adaptive pretraining, used only by SJ_AJ [7], is intended to adapt the encoder's language representation to the code-mixed domain before fine-tuning on the labelled classification task. Transliteration and back-transliteration, used by SJ_AJ [7] and OFFLangOne [13] respectively, are intended to reduce the script and spelling variation that complicates sub-word tokenization in code-mixed text; OFFLangOne [13] reports that this helps Malayalam and Tamil but that its transliteration tool performs poorly for Kannada, which in turn lowers that language's score below a plain mBERT baseline. Class-imbalance techniques, namely SOUP oversampling (DLRG [5]), class weighting and focal loss (Codewithzichao [12], Bitions [14]), class-balanced loss (OFFLangOne [13]) and cost-sensitive

DOI: 10.17148/IJARCCE.2026.151074

learning (Amrita_CEN_NLP [15]), are all intended to improve recall on the minority offensive classes identified in Table 2, though none of the papers reports that the imbalance problem is fully resolved. Pseudo-labelling, used by Bitions [14] to expand the training set with additional unlabelled comments, is intended to compensate for the limited size of the labelled corpus. Ensembling, the most widely used additional technique, is intended to reduce the variance of any single encoder's predictions by combining several models, and is used in various forms by Hate-Alert [2], indicnlp@kgp [3], SJ_AJ [7], ZYJ123 [11], Codewithzichao [12] and Bitions [14].

6.5 Overall Comparison

Across all fifteen papers, two broad observations emerge. First, ensembling and task-adaptive pretraining are the techniques most consistently associated with comparatively higher reported Tamil and Kannada scores, while single fine-tuned transformer models without these enhancements, and non-transformer deep learning models, tend to report comparatively lower scores on the same two languages. Second, this pattern does not hold as strongly for Malayalam, where even single-model or non-transformer systems often report weighted F1-scores above 0.90, suggesting that the larger size and comparatively less skewed label distribution of the Malayalam dataset, rather than model sophistication alone, may account for part of the difference. No single model or technique is reported as superior across all three languages and all fifteen papers; the relative ranking of systems changes depending on which language is considered, and the studies do not share an identical experimental setup, so these observations describe a trend rather than a controlled comparison.

7. RESULTS AND PERFORMANCE COMPARISON

Table 5 (Section 8) summarizes the dataset, approach, model, key technique and reported result for each of the fifteen papers. All fifteen papers report their primary result as weighted F1-score, in line with the official ranking metric of the DravidianLangTech-EACL2021 shared task, although a small number also report accuracy, precision and recall as supporting figures; this paper reports only the weighted F1-score for consistency and does not substitute accuracy or precision values where F1 is the metric actually used by a paper.

A consistent language-wise pattern is visible across almost all fifteen papers and is summarized in Fig. 3: the highest mean weighted F1-score across the fifteen papers is reported for Malayalam (0.94), the lowest for Kannada (0.68), with Tamil in between (0.74). This pattern holds even for papers whose absolute scores differ considerably, which suggests it is linked to properties of each language's dataset, such as the comparatively small size of the Kannada training set (6,217 comments, versus 35,139 for Tamil-English) and the proportion of the majority not-offensive class, rather than to the architecture of any specific model. Within this pattern, the highest reported Tamil score (0.80, OFFLangOne [13]) and the highest reported Kannada score (0.75, Hate-Alert [2]) both belong to systems that combine a transformer encoder with either transliteration and class-balanced loss, or ensembling, while the lowest reported scores for both languages belong to single-model systems without such enhancements, specifically DLRG [5] on Tamil-only data (0.64) and CUSATNLP [9] on Kannada (0.62).

It is also notable that OFFLangOne's SVM-TF-IDF baseline, reported within the same paper [13], performs considerably worse than its own mBERT-based system, whereas IRNLP_DAIICT's classical classifiers [8] come within a few points of several transformer-based systems on Malayalam. This indicates that a simpler, classical approach can be competitive specifically on the larger and less imbalanced Malayalam dataset, even though it is not competitive on Tamil or Kannada, where data scarcity and class imbalance appear to favor pretrained, fine-tuned or ensembled transformer models. Because the fifteen papers do not use an identical train-test split, and in some cases evaluate on a different subset of the label set, these comparisons should be read as indicative of broad tendencies in the reviewed literature rather than as a strict leaderboard.

8. COMPARATIVE TABLES

This section presents the dataset tables (Tables 2 to 4; the annotation scheme and example comments of Table 1 appear in Section 2.3), the four comparative tables (Tables 5 to 8) referred to in the preceding sections, and the supporting figures (Figs. 1 to 3). The proposed architecture is shown as Fig. 4 in Section 12.

 This work is licensed under a Creative Commons Attribution 4.0 International License

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 10, October 2026

DOI: 10.17148/IJARCCE.2026.151074

Table 2: Class Distribution of the Tamil-English Offensive-Language Dataset [1], [17]

Class Label	Train	Dev	Test	Total	% of Total
Not_offensive	25,425	3,193	3,190	31,808	72.4%
Offensive_Untargeted	2,906	356	368	3,630	8.3%
Offensive_Targeted_Insult_Group	2,557	295	288	3,140	7.1%
Offensive_Targeted_Insult_Individual	2,343	307	315	2,965	6.8%
not-Tamil	1,454	172	160	1,786	4.1%
Offensive_Targeted_Insult_Other	454	65	71	590	1.3%
Total	35,139	4,388	4,392	43,919	100%

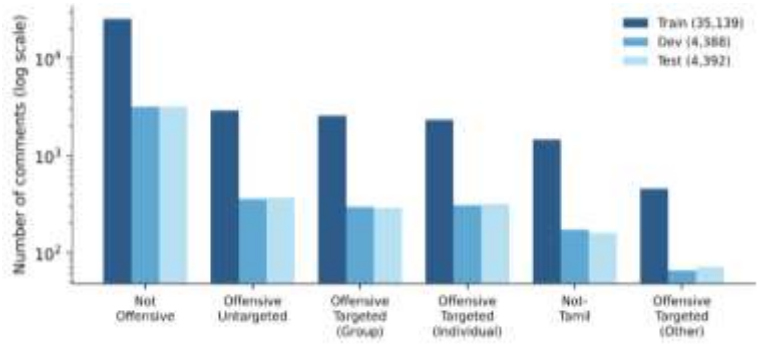


Fig. 1. Class distribution of the Tamil-English code-mixed offensive-language dataset across train, dev and test splits (log scale).

Table 3: Offensive-Language Subsets of the DravidianCodeMix Corpus (Row Counts of the Released Split Files) [17]

Language subset	Train	Dev	Test	Total	Labels used	Out-of-language label
Tamil-English	35,139	4,388	4,392	43,919	6	not-Tamil
Kannada-English	6,217	777	778	7,772	6	not-Kannada
Malayalam-English	16,010	1,999	2,001	20,010	5	not-malayalam

Table 4: Script Mix and Comment Length of the Tamil-English Subset by Class (Computed from the Released Files)

Class	Roman only	Tamil only	Tamil + Roman	Neither	Mean words	Median words
Not_offensive	25,110 (78.9%)	5,179 (16.3%)	1,517 (4.8%)	2	9.9	8
Offensive_Untargeted	3,456 (95.2%)	104 (2.9%)	70 (1.9%)	0	10.1	9
Offensive_Targeted_Insult_Individual	2,556 (86.2%)	282 (9.5%)	127 (4.3%)	0	11.0	9
Offensive_Targeted_Insult_Group	2,167 (69.0%)	734 (23.4%)	239 (7.6%)	0	15.0	11
Offensive_Targeted_Insult_Other	489 (82.9%)	72 (12.2%)	29 (4.9%)	0	11.6	9
not-Tamil	1,714 (96.0%)	2 (0.1%)	4 (0.2%)	66	10.3	9
All classes	**35,492 (80.8%)**	**6,373 (14.5%)**	**1,986 (4.5%)**	**68**	**10.4**	**8**

 This work is licensed under a Creative Commons Attribution 4.0 International License

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940



International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 10, October 2026

DOI: 10.17148/IJARCCE.2026.151074

Script is assigned by Unicode block as described in Section 2.4; percentages are shares of each class. Length is counted in whitespace-separated words.

Table 5: Comparative Overview of the 15 Reviewed Papers

#	Paper	Lang./Task	Dataset	Approach	Model	Key Technique	Result (Wtd. F1)
1	Hate-Alert	Ta,Ka,Ma / 6-class OLI	DravidianLangTech-2021 corpus	Multi-transformer fine-tuning	XLM-R, mBERT, IndicBERT, MuRIL, DistilBERT, CNN-BERT fusion	GA-weighted ensemble	Ta 0.78, Ka 0.75, Ma 0.97
2	indicnlp@kgp	Ta,Ka,Ma / 6-class OLI	Same shared-task corpus	Transfer learning + ensembling	AWD-LSTM (ULMFiT), mBERT, XLM-R	Averaging ensemble, OLID transfer learning	Ma 0.97, Ta 0.77, Ka 0.72
3	NLP-CUET	Ta,Ka,Ma / 6-class OLI	Same shared-task corpus	ML, DL and transformer comparison	LR, SVM, LSTM(+Attn), mBERT, IndicBERT, XLM-R	Confusion-matrix error analysis	Ta 0.76, Ma 0.93, Ka 0.71
4	DLRG	Tamil only / 6-class OLI	Tamil subset only	Single transformer fine-tuning	mBERT (bert-base-multilingual-cased)	SOUP oversampling/undersampling	Ta 0.64
5	IIITT	Ta,Ka,Ma / 6-class OLI	Same shared-task corpus	Transfer learning comparison (6 models)	CNN-BiLSTM, mBERT+BiLSTM, XLM-R, DistilmBERT, ALBERT, ULMFiT	Best-per-language model selection	Ma 0.9603, Ta 0.7895, Ka 0.7277
6	SJ_AJ (Base)	Ka,Ta,Ma / 6-class OLI	Same shared-task corpus	Task-adaptive pretraining + fusion	mBERT/XLM-R + char-BiLSTM fusion	Task-adaptive MLM pretraining, transliteration, majority vote	Ka 0.7030, Ta 0.7694, Ma 0.9676
7	IRNLP_DAIICT	Ka,Ma,Ta / 6-class OLI	Same shared-task corpus	Classical ML, feature comparison	TF-IDF char n-grams / MuRIL + LR/SVM/RF	Feature representation comparison	Ka 0.64, Ma 0.95, Ta 0.71
8	CUSATNLP	Ka,Ma,Ta / 6-class OLI	Same shared-task corpus	Language-agnostic embedding	LaBSE + softmax classifier	No additional preprocessing	Ma 0.95, Ka 0.62, Ta 0.69
9	cs@DLT	Ta,Ka,Ma / 6-class OLI	Same shared-task corpus	Transformer + CNN fusion	mBERT (last 3 layers) + TextCNN	Weighted layer-sum fusion	Ma 0.93, Ta 0.74, Ka 0.64
10	ZYJ123	Ka,Ma,Ta / 6-class OLI	Same shared-task corpus	Transformer + CNN fusion	XLM-R + DPCNN	5-fold CV + voting	Ka 0.69, Ma 0.92, Ta 0.76
11	Codewithzichao	Ta,Ma,Ka / 6-class OLI	Combined tri-lingual data	Imbalance handling + adversarial training	XLM-R / mBERT, avg-pooling	Class weights, focal loss, FGM, ensemble	Ta 0.75, Ma 0.94, Ka 0.72

 This work is licensed under a Creative Commons Attribution 4.0 International License

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 10, October 2026

DOI: 10.17148/IJARCCE.2026.151074

#	Paper	Lang./Task	Dataset	Approach	Model	Key Technique	Result (Wtd. F1)
12	OFFLangOne	Ma,Ta,Ka / 6-class OLI	Same shared-task corpus	Transliteration + loss reweighting	mBERT + back-transliteration	Class-Balanced Loss	Ma 0.96, Ta 0.80, Ka 0.65
13	Bitions	Ta,Ka,Ma / 6-class OLI	Same shared-task corpus + unlabelled data	Pseudo-labelling + ensembling	DistilmBERT, ULMFiT, IndicBERT	Class-weighting, focal loss, pseudo-labelling, soft-vote	Ma 0.97, Ta 0.75, Ka 0.70
14	Amrita_CEN_NLP	Ma,Ta,Ka / 6-class OLI	Same shared-task corpus	Non-transformer hybrid DL	CNN+BiLSTM+Dense, BiLSTM, BiRNN	Cost-sensitive learning	Ma 0.85, Ta 0.62, Ka 0.58
15	Hypers	Ma,Ka,Ta / 6-class OLI	Same shared-task corpus	Single transformer fine-tuning	Fine-tuned mBERT	Minimal preprocessing (emoji handling)	Ma 0.96, Ka 0.70, Ta 0.73

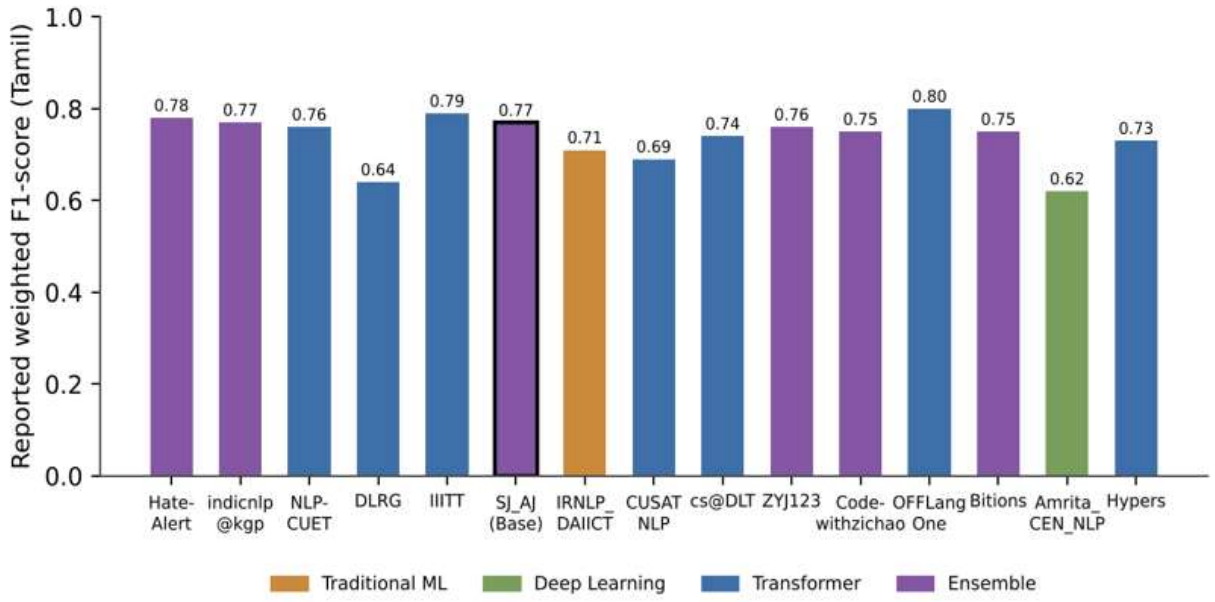


Fig. 2. Reported Tamil-English weighted F1-score across the 15 reviewed papers, grouped by approach category.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940



International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 10, October 2026

DOI: 10.17148/IJARCCE.2026.151074

Table 6: Comparison of Models and Approaches

Category	Papers	Representative Models	Observation
Traditional ML	IRNLP_DAIICT [8], NLP-CUET [4] (baseline)	LR, SVM, Random Forest with TF-IDF/char n-grams	Competitive on Malayalam; weaker on Tamil/Kannada; used mainly as baseline
Deep Learning (non-transformer)	Amrita_CEN_NLP [15], IIITT [6] (one config), NLP-CUET [4] (LSTM)	CNN-BiLSTM-Dense, BiLSTM, BiRNN, LSTM+Attention	Better than classical ML on some languages; behind transformer fine-tuning overall
Transformer-based	Hate-Alert [2], indicnlp@kgp [3], NLP-CUET [4], DLRG [5], IIITT [6], SJ_AJ [7], CUSATNLP [9], cs@DLT [10], ZYJ123 [11], Codewithzichao [12], OFFLangOne [13], Bitions [14], Hypers [16]	mBERT, XLM-R, IndicBERT, MuRIL, DistilBERT/DistilmBERT, LaBSE, ALBERT	Dominant approach; highest reported scores typically combine transformers with ensembling or pretraining adaptation
Ensemble Methods	Hate-Alert [2], indicnlp@kgp [3], SJ_AJ [7], ZYJ123 [11], Codewithzichao [12], Bitions [14]	GA-weighted, averaging, majority-vote, soft-vote, CV+voting	Associated with comparatively higher Tamil/Kannada scores versus single-model systems
Other / Hybrid	SJ_AJ [7] (fusion+task-adaptive pretraining), OFFLangOne [13] (transliteration), cs@DLT [10] / ZYJ123 [11] (CNN fusion heads)	Char-BiLSTM fusion, back-transliteration, TextCNN/DPCNN heads	Targets code-mixed/script issues specifically; used by a minority of papers

Table 7: Comparison of Techniques Used Across the Studies

Technique	Purpose	Papers Using It
TF-IDF / character n-grams	Sparse statistical text representation	IRNLP_DAIICT [8], NLP-CUET [4] (baseline)
Task-adaptive pretraining	Adapt encoder to code-mixed domain before fine-tuning	SJ_AJ (base paper) [7]
Transliteration / back-transliteration	Normalize script variation in code-mixed text	SJ_AJ [7], OFFLangOne [13]
Class weighting / focal loss	Improve minority-class recall under label skew	Codewithzichao [12], Bitions [14]
Class-balanced loss (effective number)	Reweight loss by per-class sample count	OFFLangOne [13]
Oversampling/undersampling (SOUP)	Rebalance training distribution	DLRG [5]
Cost-sensitive learning	Penalize minority-class errors more heavily	Amrita_CEN_NLP [15]
Pseudo-labelling	Expand training data using unlabelled comments	Bitions [14]
Ensembling (vote/GA-weighted/CV)	Reduce prediction variance across models	Hate-Alert [2], indicnlp@kgp [3], SJ_AJ [7], ZYJ123 [11], Codewithzichao [12], Bitions [14]

 This work is licensed under a Creative Commons Attribution 4.0 International License



International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 10, October 2026
DOI: 10.17148/IJARCCE.2026.151074

Technique	Purpose	Papers Using It
Confusion-matrix error analysis	Identify which classes are confused	NLP-CUET [4], CUSATNLP [9]
Dedicated explainability (LIME/SHAP/attention)	Explain individual predictions	Not reported in any of the 15 papers

Table 8: Strengths and Limitations of the Reviewed Studies

Paper	Strength	Limitation
Hate-Alert [2]	GA-optimised ensembling of five encoders yields highest Ka score and strong Ta/Ma scores	Does not explain weaker IndicBERT/MuRIL performance; GPU limits restricted XLM-R-large use
indicnlp@kgp [3]	Effective transfer learning from English OLID data to 3 Dravidian languages	Limited discussion of per-language error sources
NLP-CUET [4]	Systematic ML vs DL vs transformer comparison on identical data/split	Severe class imbalance causes near-zero recall on minority classes
DLRG [5]	Early Tamil-specific BERT application with explicit imbalance correction	Zero F1 on two minority classes; weak generalisation
IIITT [6]	Broad comparison of 6 transfer-learning architectures per language	Minority-class prediction remains weak across all 6 models
SJ_AJ (Base) [7]	Task-adaptive pretraining improves over vanilla fine-tuning by ~1-2% F1	No domain-adaptive pretraining; fusion branch helps inconsistently
IRNLP_DAIICT [8]	Direct comparison of sparse vs pretrained embeddings with classical ML	MuRIL embeddings not fine-tuned; underperform for Ka/Ta
CUSATNLP [9]	Simple, language-agnostic embedding pipeline without preprocessing	Weak on Kannada/Tamil minority classes
cs@DLT [10]	BERT-TextCNN fusion captures semantic and local features jointly	Below top-ranked systems, especially for Kannada
ZYJ123 [11]	DPCNN + stratified CV/voting adds robustness to XLM-R	Limited detail on error sources
Codewithzichao [12]	Combines data merging, imbalance handling and adversarial training in one pipeline	Language-specific causes of differences not analysed
OFFLangOne [13]	Back-transliteration + class-balanced loss improves Ma/Ta notably	Transliteration tool fails for Kannada, lowering its score
Bitions [14]	Pseudo-labelling + ensembling of DistilmBERT and ULMFiT	IndicBERT discarded for poor results; focal loss underperforms class-weighting
Amrita_CEN_NLP [15]	Demonstrates a non-transformer DL alternative with cost-sensitive learning	No pretrained transformer; weaker than BERT-based systems overall
Hypers [16]	Simple, minimal-preprocessing mBERT fine-tuning pipeline	No ensembling or fusion; relies solely on single-model capacity

 This work is licensed under a Creative Commons Attribution 4.0 International License

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 10, October 2026

DOI: 10.17148/IJARCCE.2026.151074

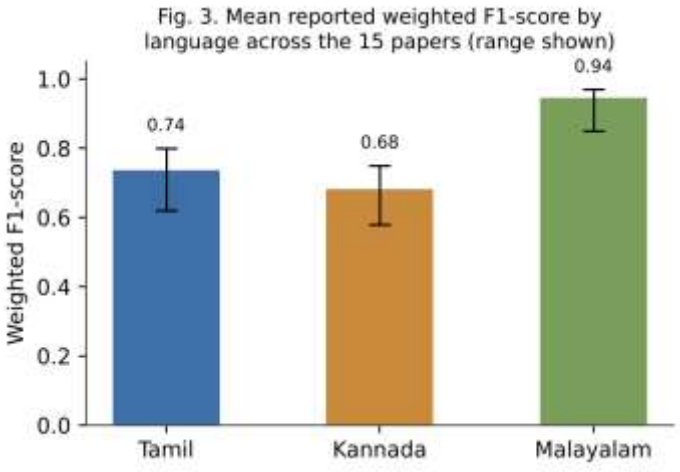


Fig. 3. Mean reported weighted F1-score by language across the 15 papers, with the minimum-to-maximum range shown as error bars.

9. COMPARATIVE EVALUATION OF MODEL PERFORMANCE

9.1 Evaluation Setup

This section evaluates only the models implemented and executed in this study, on the Tamil-English test split of 4,392 comments [17]. Seven individual models were trained on the 35,139-comment training split: three classical classifiers on TF-IDF character and word n-gram features (linear SVM, logistic regression and Complement Naive Bayes) and four pretrained transformers fine-tuned for four epochs (XLM-R base on raw and on romanised text, mBERT on romanised text, and MuRIL base on raw text). Class-balanced loss was used for the transformers and balanced class weights for the SVM and logistic regression. Three ensembles of the seven models were evaluated: equal weights, development-tuned weights, and the final ensemble, which adds a class bias to the tuned weights. Model and ensemble selection used only the 4,388-comment development split, and the test split was used only to report the final results. Each neural model is a single run (one seed).

Accuracy and weighted and macro precision, recall and F1-score were computed with the code in Listing 1. Weighted recall equals accuracy by definition. Macro averages give each of the six classes equal weight, so they expose minority-class performance that accuracy and weighted F1-score, which are dominated by the Not-offensive class (72.6% of the test comments), hide.

Listing 1: Evaluation code applied to the test-set predictions of every model

```
import pandas as pd
from sklearn.metrics import (accuracy_score, precision_recall_fscore_support,
                             classification_report, confusion_matrix)

LABELS = ["Not_offensive", "Offensive_Targeted_Insult_Group",
          "Offensive_Targeted_Insult_Individual", "Offensive_Targeted_Insult_Other",
          "Offensive_Untargeted", "not-Tamil"]

def evaluate(y_true, y_pred):
    row = {"Accuracy": accuracy_score(y_true, y_pred)}
    for avg in ("weighted", "macro"):
        p, r, f, _ = precision_recall_fscore_support(
            y_true, y_pred, labels=LABELS, average=avg, zero_division=0)
        row.update({f"Precision ({avg})": p, f"Recall ({avg})": r, f"F1 ({avg})": f})
    return row

# test_preds: {model name: predicted test labels}; y_test: gold test labels (4,392 comments)
results = pd.DataFrame({m: evaluate(y_test, y_hat) for m, y_hat in test_preds.items()}).T
```

 This work is licensed under a Creative Commons Attribution 4.0 International License

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 10, October 2026

DOI: 10.17148/IJARCCE.2026.151074

```
print(results.round(4))                    # -> Table 9

final = test_preds["Final ensemble"]
# class-wise report of the final ensemble -> Output 1
print(classification_report(y_test, final, labels=LABELS, digits=2, zero_division=0))

# share of each offensive class labelled Not_offensive -> Output 2
cm = confusion_matrix(y_test, final, labels=LABELS)
to_not_offensive = cm[:, 0] / cm.sum(axis=1)
for lab, share in zip(LABELS[1:5], to_not_offensive[1:5]):
    print(f"{lab:<40s}{share:.3f}")
```

9.2 Overall Performance Comparison

Table 9 gives the test-set scores of the ten models and Table 10 identifies the best- and lowest-performing model for each metric.

Table 9: Test-Set Performance of the Models Implemented in This Study (Tamil-English, 4,392 comments)

Model	Accuracy	Precision (weighted)	Recall (weighted)	Weighted F1	Precision (macro)	Recall (macro)	Macro F1
SVM (TF-IDF char/word n-grams)	0.7873	0.7502	0.7873	0.7552	**0.6098**	0.4290	0.4657
Logistic regression	0.7662	**0.7705**	0.7662	0.7668	0.5319	**0.5124**	**0.5087**
Complement Naive Bayes	0.7757	0.7294	0.7757	0.7384	0.5550	0.4020	0.4351
XLM-R base, raw text	0.7714	0.7429	0.7714	0.7559	0.4782	0.4509	0.4631
XLM-R base, romanised text	0.7714	0.7404	0.7714	0.7543	0.4763	0.4484	0.4599
mBERT, romanised text	0.7675	0.7470	0.7675	0.7557	0.5008	0.4511	0.4671
MuRIL base, raw text	0.7546	0.6665	0.7546	0.7055	0.3299	0.3624	0.3394
Ensemble, equal weights	**0.7926**	0.7524	**0.7926**	0.7665	0.5281	0.4470	0.4784
Ensemble, dev-tuned weights	0.7910	0.7661	0.7910	0.7749	0.5616	0.4775	0.4993
Final ensemble, dev-tuned weights + class bias	0.7885	0.7686	0.7885	**0.7755**	0.5383	0.4819	0.5050

Green and bold: best value in the column. Orange: lowest value in the column. Weighted recall is identical to accuracy by definition, so both columns share the same highlights. The final ensemble row is in bold type. Each neural model is a single run (one seed).

Table 10: Best- and Lowest-Performing Models for Each Metric (Test Set)

Metric	Best-performing model (score)	Lowest-performing model (score)	Spread
Accuracy / weighted recall	Ensemble, equal weights: 0.7926	MuRIL base, raw text: 0.7546	0.0380
Precision (weighted)	Logistic regression: 0.7705	MuRIL base, raw text: 0.6665	0.1040
Weighted F1-score	Final ensemble, dev-tuned weights + class bias: 0.7755	MuRIL base, raw text: 0.7055	0.0700
Precision (macro)	SVM (TF-IDF char/word n-grams): 0.6098	MuRIL base, raw text: 0.3299	0.2799
Recall (macro)	Logistic regression: 0.5124	MuRIL base, raw text: 0.3624	0.1500
Macro F1-score	Logistic regression: 0.5087	MuRIL base, raw text: 0.3394	0.1693

Spread is the best score minus the lowest score in the column of Table 10.

 This work is licensed under a Creative Commons Attribution 4.0 International License

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940



International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 10, October 2026

DOI: 10.17148/IJARCCE.2026.151074

No single model is best on every metric. The final ensemble has the highest weighted F1-score (0.7755) and the equal-weight ensemble the highest accuracy (0.7926). Logistic regression has the highest weighted precision (0.7705), macro recall (0.5124) and macro F1-score (0.5087), and the SVM the highest macro precision (0.6098). MuRIL base is the lowest-performing model on every metric, with a acro F1-score of 0.3394.

The final ensemble is the strongest overall model, by a small margin. Its weighted F1-score is 0.0087 above that of the best single model (logistic regression, 0.7668), while its macro F1-score is 0.0037 below it. Because each neural model is a single run, differences of this size are indicative rather than established. The proportion of Not-offensive comments in the test split is 0.7263 (3,190 of 4,392), so the accuracy of the final ensemble (0.7885) exceeds this proportion by 0.0622 and that of the weakest model (0.7546) by only 0.0283.

Effect of ensemble tuning. From the equal-weight to the development-tuned to the final ensemble, accuracy decreases (0.7926, 0.7910, 0.7885) while macro recall (0.4470, 0.4775, 0.4819), macro F1-score (0.4784, 0.4993, 0.5050) and weighted F1-score (0.7665, 0.7749, 0.7755) increase. The tuning therefore shifts predictions away from the majority class at a small cost in accuracy.

Model families. The three XLM-R and mBERT models (weighted F1-score 0.7543 to 0.7559) did not outperform the TF-IDF classifiers: logistic regression is higher (0.7668) and the SVM lies within the same range (0.7552). Romanisation changed XLM-R base by only 0.0016 in weighted F1-score and 0.0032 in macro F1-score. Rankings also depend on the metric: the SVM has a higher accuracy than logistic regression (0.7873 against 0.7662) but a lower weighted F1-score (0.7552 against 0.7668) and macro F1-score (0.4657 against 0.5087). The macro metrics separate the models more than accuracy does: across the ten models the spread is 0.0380 in accuracy, 0.0700 in weighted F1-score and 0.1693 in macro F1-score. Since each model was run once with fixed settings, these results do not establish a general ranking of the model families.

9.3 Class-Wise Performance of the Final Ensemble

Class-wise results are available for the final ensemble only. Output 1 shows its classification report on the test split, and Output 2 shows the share of each offensive class that it assigned to Not_offensive.

Output 1: Classification report of the final ensemble (test set)

	precision	recall	f1-score	support
Not_offensive	0.87	0.94	0.90	3190
Offensive_Targeted_Insult_Group	0.38	0.34	0.36	288
Offensive_Targeted_Insult_Individual	0.54	0.36	0.43	315
Offensive_Targeted_Insult_Other	0.08	0.08	0.08	71
Offensive_Untargeted	0.47	0.35	0.40	368
not-Tamil	0.88	0.82	0.85	160
accuracy			0.79	4392
macro avg	0.54	0.48	0.50	4392
weighted avg	0.77	0.79	0.78	4392

Output 2: Share of each offensive class labelled Not_offensive by the final ensemble (test set)

Offensive_Targeted_Insult_Group	0.399
Offensive_Targeted_Insult_Individual	0.406
Offensive_Targeted_Insult_Other	0.507
Offensive_Untargeted	0.394

The two non-offensive categories are detected well (Not_offensive F1-score 0.90, not-Tamil 0.85), whereas the four offensive classes obtain F1-scores of 0.08 to 0.43: Individual 0.43, Untargeted 0.40, Group 0.36 and Other 0.08. For the Group, Individual and Untargeted classes precision exceeds recall (0.38 against 0.34, 0.54 against 0.36 and 0.47 against 0.35), and about 40% of the comments of each of these classes are labelled Not-offensive (Output 2).

 This work is licensed under a Creative Commons Attribution 4.0 International License



International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 10, October 2026
DOI: 10.17148/IJARCCE.2026.151074

Offensive_Targeted_Insult_Other is almost undetected: its precision and recall are both 0.08, six of its 71 test comments are classified correctly, and 50.7% are labelled Not-offensive. The weighted F1-score (0.7755) exceeds the macro F1-score (0.5050) by 0.2705 for this reason: the macro F1-score is the mean of the six class F1-scores (0.9015, 0.3603, 0.4313, 0.0845, 0.4044 and 0.8479), and the weighted score is dominated by the Not-offensive class. Class-balanced loss (transformers) and balanced class weights (SVM and logistic regression) were already used in training, so reweighting alone did not resolve the minority-class errors on this dataset.

9.4 Summary and Limitations

The final ensemble is the strongest model on the primary metric (weighted F1-score 0.7755), logistic regression is the strongest on the macro metrics, and MuRIL base is the weakest on all metrics. The main weakness of every aggregate result is minority-class classification, in particular Offensive_Targeted_Insult_Other, which the final ensemble classifies correctly in six of 71 cases. The conclusions are limited by single-seed training of each neural model, the absence of confidence intervals or significance tests, and the availability of class-wise results for the final ensemble only. Differences of about 0.01 in weighted F1-score between the leading models should therefore not be treated as established.

10. DISCUSSION

Considering all fifteen papers together, pretrained multilingual transformer fine-tuning, most often mBERT or XLM-R, is by far the most frequently used approach, appearing in some form in twelve of the fifteen papers. Ensembling and class-imbalance correction are the next most frequent enhancements, each appearing in roughly a third of the papers. Task-adaptive pretraining and transliteration-based preprocessing are the least frequently used, appearing in only one and two papers respectively, despite being among the few techniques that directly target the code-mixed and transliterated nature of the text rather than only the classification stage.

Complex models do not uniformly outperform simpler ones in this literature. On the Malayalam dataset, where more labelled data is available and the class distribution is less skewed, classical machine learning (IRNLP_DAIICT [8]) and non-transformer deep learning (Amrita_CEN_NLP [15]) report weighted F1-scores within a reasonable range of the strongest transformer-based ensembles, even though they fall well behind on Tamil and Kannada. This suggests that dataset size and class balance play as large a role in determining reported performance as model architecture does, a point the dataset statistics in Table 2 support directly: the Tamil-English training set is nearly six times the size of the Kannada training set, and both are dominated by the not-offensive class.

Class imbalance is the limitation most frequently and explicitly acknowledged across the fifteen papers, usually in the form of near-zero recall on one or more minority offensive categories, and is addressed through a range of only partially successful techniques, including oversampling, class weighting, focal loss, class-balanced loss and cost-sensitive learning. Code-mixing and transliteration are acknowledged as sources of difficulty by several papers, including the base paper, but are directly addressed, through transliteration or back-transliteration of the text, by only two of the fifteen studies, and even then with mixed results, since OFFLangOne's transliteration tool is reported to degrade performance specifically for Kannada. The difference in reported performance across the three languages is therefore best explained by a combination of dataset size, label skew and script-handling reliability rather than any single factor.

A limitation that is repeatedly visible but rarely discussed directly is the lack of attention to model interpretability. Several papers report confusion matrices to show which classes are most often confused with one another, which is a useful first step, but none applies a dedicated explainability technique to examine why individual comments are classified as they are. Given that the same papers report persistent difficulty with code-mixed and transliterated text, an examination of model behaviour at the level of individual tokens or examples, rather than only aggregate metrics, remains comparatively unexplored in this body of work.

11. RESEARCH GAP

The comparison in Sections 6 to 8 and 10 shows that the fifteen reviewed papers concentrate almost entirely on improving aggregate classification performance, through model selection, ensembling, pretraining adaptation and class-imbalance correction, while giving comparatively less attention to explaining the predictions their models produce. Performance evaluation across all fifteen papers is carried out using weighted F1-score, and in a few cases supporting accuracy, precision and recall figures, with error analysis, where it is reported at all, limited to confusion matrices in papers such as NLP-CUET [4] and CUSATNLP [9]. A confusion matrix indicates which classes are confused with one another but does not indicate which words, code-mixed segments or transliterated tokens drove a specific prediction.

 This work is licensed under a Creative Commons Attribution 4.0 International License

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 10, October 2026

DOI: 10.17148/IJARCCE.2026.151074

None of the fifteen papers, including the base paper SJ_AJ [7], applies a post-hoc explanation technique such as LIME or SHAP, nor do any visualize attention weights from their transformer models to interpret individual predictions. This is not to claim that explainability has never been applied to hate speech detection in general, a claim this review does not have evidence to support, but within the specific set of fifteen Tamil-Kannada-Malayalam code-mixed offensive-language papers reviewed here, interpretability is not consistently addressed and is, at most, approached indirectly through aggregate error analysis. This gap is particularly relevant given that the same papers identify code-mixing, transliteration inconsistency and class imbalance, confirmed quantitatively in Table 2, as sources of misclassification, factors that a token-level or feature-level explanation method could help to examine directly rather than infer only from aggregate scores.

12. PROPOSED STUDY

Building on the research gap identified above, the proposed study, titled "Explainable Deep Learning for Hate Speech Detection in Code-Mixed Tamil-English Text: A Comparative Study," intends to extend the comparative perspective developed in this paper by incorporating explainability into a Tamil-English code-mixed hate speech detection pipeline built on the same family of pretrained transformer models reviewed here and trained on the Tamil-English subset described in Table 2. Rather than proposing an entirely new classification architecture, the direction motivated by this review is to apply explainable AI techniques, such as attention-weight visualization within a transformer classifier, or post-hoc methods such as LIME or SHAP applied to model predictions, to examine which tokens or code-mixed segments influence a given classification decision. This is intended to address the specific gap identified in Section 11, namely that existing Tamil-English and related Dravidian code-mixed offensive-language systems are evaluated almost exclusively through aggregate metrics without examining the basis for individual predictions.

Fig. 4 shows the architecture proposed for the comparative part of the study. Tamil-English code-mixed comments pass through data preprocessing (cleaning, noise removal, normalisation, tokenisation and code-mix handling), are split into training, validation and test sets, and are converted into text representations (token indices, embeddings and contextual embeddings). The same input is then given to five deep learning branches, namely CNN, LSTM, BiLSTM, CNN-LSTM and a Transformer branch using mBERT or XLM-R, each feeding a hate speech / non-hate speech classification stage. The branches are compared on accuracy, precision, recall and F1-score to identify the best-performing model and to summarise the strengths and limitations of each.

The figure connects to the review in three ways. The branches correspond to model families found in the reviewed papers: LSTM (NLP-CUET [4]), BiLSTM and CNN-BiLSTM-Dense (Amrita_CEN_NLP [15]), CNN-BiLSTM (IIITT [6]) and mBERT or XLM-R (Section 3.3). The evaluation stage lists accuracy, precision, recall and F1-score, of which the weighted F1-score is the primary metric reported by the reviewed papers (Section 7). Two differences also need to be stated. The figure draws no explainability stage; as described in this section, that component would operate on the predictions of the trained classifiers. And its output stage is a binary hate / non-hate decision, whereas the labels released with the dataset form six classes (Table 1), so how the two are reconciled is a design choice of the proposed study that this review does not fix.

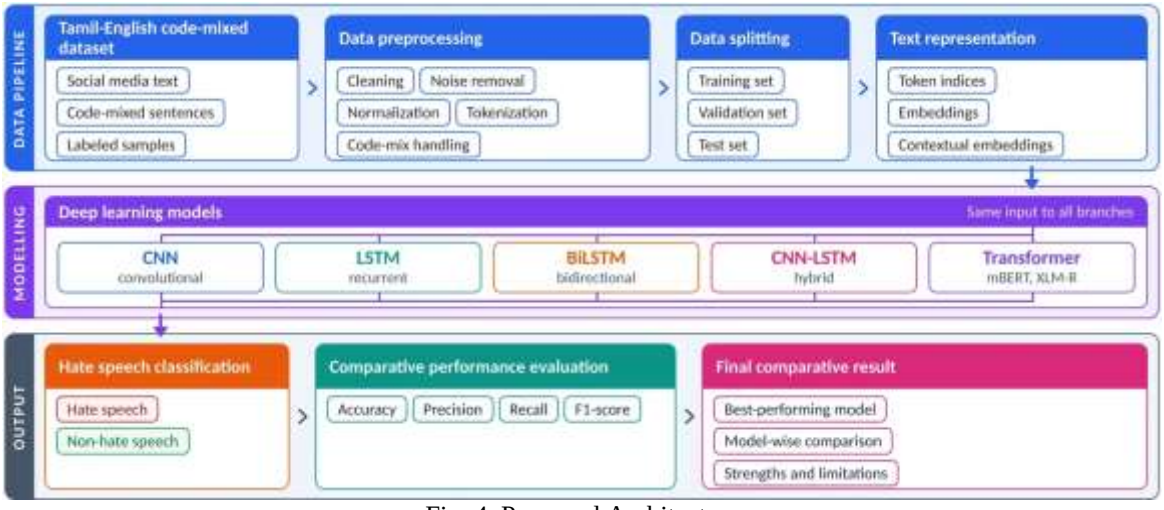


Fig. 4. Proposed Architecture

 This work is licensed under a Creative Commons Attribution 4.0 International License



International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 10, October 2026
DOI: 10.17148/IJARCCE.2026.151074

This direction is also intended to build on the comparative findings of Sections 6 and 7: since class imbalance, transliteration inconsistency and dataset scarcity are repeatedly identified as sources of error across the fifteen reviewed papers, an explainability layer applied to a Tamil-English classifier may help clarify whether misclassifications are driven by these factors at the level of individual examples, something the aggregate evaluation protocols used by the reviewed papers cannot show. Apart from the baseline evaluation in Section 9, no experimental results are claimed at this stage; the contribution of the present paper is the comparative groundwork, the baseline evaluation, the grounding of that comparison in the actual Tamil-English dataset statistics, and the identification of the gap that the proposed study is intended to address.

13. CONCLUSION AND FUTURE WORK

This paper reviewed and compared fifteen system-description papers submitted to the DravidianLangTech-EACL2021 shared task on offensive language identification in Tamil, Kannada and Malayalam code-mixed text, evaluated on the Dravidian Offensive Language Dataset, using the base paper SJ_AJ@DravidianLangTech-EACL2021 as the primary point of reference. Rather than summarizing the fifteen papers individually, the comparison organized them by approach family, covering classical machine learning, non-transformer deep learning and transformer-based methods, and by special technique, covering task-adaptive pretraining, transliteration, class-imbalance handling, pseudo-labelling and ensembling, and grounded this comparison in the actual class distribution of the Tamil-English subset of the dataset.

The comparison showed that pretrained multilingual transformer fine-tuning, most commonly using mBERT or XLM-RoBERTa, is the dominant approach across the fifteen papers, and that ensembling and task-adaptive pretraining tend to be associated with comparatively higher reported Tamil and Kannada scores, while a single fine-tuned model without such enhancements tends to report lower scores on the same two languages. This trend was less consistent for Malayalam, where simpler approaches were also competitive, most likely due to the larger size and less skewed class distribution of that language's dataset. Because the fifteen papers differ in dataset handling, train-test split and, occasionally, evaluation metric, these findings were presented as indicative trends rather than a strict ranking. Class imbalance, confirmed directly from the dataset statistics in Table 2, transliteration inconsistency for Kannada, and limited training data were identified as recurring limitations across the reviewed studies. The comparison further found that explainability is not consistently addressed in this literature; error analysis, where present, is limited to confusion matrices, and no reviewed paper applies a dedicated interpretability technique to its predictions.

This finding establishes the motivation for the proposed study, which aims to incorporate explainable deep learning methods into Tamil-English code-mixed hate speech detection. Realistic future work arising from this review includes evaluating attention-based or post-hoc explanation methods (such as LIME or SHAP) on a transformer-based Tamil-English classifier trained on the dataset splits described in Table 2; examining whether token-level explanations can help diagnose the class-imbalance and transliteration-related errors identified across the fifteen reviewed papers; and extending the comparison to more recent Tamil-English offensive-language datasets as they become available, so that findings are not limited to the single shared-task corpus examined in this review.

REFERENCES

[1] B. R. Chakravarthi, R. Priyadharshini, N. Jose, A. Kumar M, T. Mandl, P. K. Kumaresan, R. Ponnusamy, H. R. K. V, J. P. McCrae, and E. Sherly, "Findings of the shared task on Offensive Language Identification in Tamil, Malayalam, and Kannada," in Proc. First Workshop on Speech and Language Technologies for Dravidian Languages (DravidianLangTech-EACL 2021), 2021.

[2] P. Saha, S. Mathew, B. Garain, and A. Mukherjee, "Hate-Alert@DravidianLangTech-EACL2021: Multilingual transformer ensembles for offensive language identification," in Proc. DravidianLangTech-EACL 2021, 2021.

[3] K. Kedia and A. Nandy, "indicnlp@kgp@DravidianLangTech-EACL2021: Offensive language identification in Dravidian languages using transfer learning with transformer and LSTM models," in Proc. DravidianLangTech-EACL 2021, 2021.

[4] O. Sharif, E. Hossain, and M. M. Hoque, "NLP-CUET@DravidianLangTech-EACL2021: Offensive language detection from multilingual code-mixed text using machine learning, deep learning and transformer models," in Proc. DravidianLangTech-EACL 2021, 2021.

[5] R. Rajalakshmi, Y. Reddy, and S. Khare, "DLRG@DravidianLangTech-EACL2021: Transformer based approach for offensive language identification on code-mixed Tamil," in Proc. DravidianLangTech-EACL 2021, 2021.

[6] K. Yasaswini, K. Puranik, A. Hande, R. Priyadharshini, S. Thavareesan, and B. R. Chakravarthi, "IIITT@DravidianLangTech-EACL2021: Transfer learning for offensive language detection in Dravidian languages," in Proc. DravidianLangTech-EACL 2021, 2021.

 This work is licensed under a Creative Commons Attribution 4.0 International License



International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 10, October 2026

DOI: 10.17148/IJARCCE.2026.151074

[7] S. M. Jayanthi and A. Gupta, "SJ_AJ@DravidianLangTech-EACL2021: Task-adaptive pre-training of multilingual BERT models for offensive language identification," in Proc. DravidianLangTech-EACL 2021, 2021.

[8] B. Dave, S. Bhat, and P. Majumder, "IRNLP_DAIICT@DravidianLangTech-EACL2021: Offensive language identification in Dravidian languages using TF-IDF char n-grams and MuRIL," in Proc. DravidianLangTech-EACL 2021, 2021.

[9] S. Renjit and S. M. Idicula, "CUSATNLP@DravidianLangTech-EACL2021: Language agnostic classification of offensive content in tweets," in Proc. DravidianLangTech-EACL 2021, 2021.

[10] S. Chen and B. Kong, "cs@DravidianLangTech-EACL2021: Offensive language identification based on multilingual BERT model," in Proc. DravidianLangTech-EACL 2021, 2021.

[11] J. Zhao and H. Tao, "ZYJ123@DravidianLangTech-EACL2021: Offensive language identification based on XLM-RoBERTa with DPCNN," in Proc. DravidianLangTech-EACL 2021, 2021.

[12] Z. Li, "Codewithzichao@DravidianLangTech-EACL2021: Exploring multilingual transformers for offensive language identification in Dravidian languages," in Proc. DravidianLangTech-EACL 2021, 2021.

[13] S. Dowlagar and R. Mamidi, "OFFLangOne@DravidianLangTech-EACL2021: Transliteration and class-balanced loss for offensive language identification in Dravidian languages," in Proc. DravidianLangTech-EACL 2021, 2021.

[14] D. Tula, P. Potluri, S. Ms, S. Doddapaneni, P. Sahu, R. Sukumaran, and P. Patwa, "Bitions@DravidianLangTech-EACL2021: Ensemble of multilingual language models with pseudo labeling for offence detection in Dravidian languages," in Proc. DravidianLangTech-EACL 2021, 2021.

[15] S. Sreelakshmi, B. Premjith, and K. P. Soman, "Amrita_CEN_NLP@DravidianLangTech-EACL2021: Deep learning-based offensive language identification in Malayalam, Tamil and Kannada," in Proc. DravidianLangTech-EACL 2021, 2021.

[16] K. Vasantharajan and U. Thayasivam, "Hypers@DravidianLangTech-EACL2021: Offensive language identification in Dravidian code-mixed YouTube comments and posts," in Proc. DravidianLangTech-EACL 2021, 2021.

[17] B. R. Chakravarthi, R. Priyadharshini, V. Muralidaran, N. Jose, S. Suryawanshi, E. Sherly, and J. P. McCrae, "DravidianCodeMix: sentiment analysis and offensive language identification dataset for Dravidian languages in code-mixed text," Language Resources and Evaluation, Feb. 2022, doi: 10.1007/s10579-022-09583-7.

 This work is licensed under a Creative Commons Attribution 4.0 International License