Computer Speech and Language 76 (2022) 101404

Contents lists available at ScienceDirect

Computer Speech & Language

journal homepage: www.elsevier.com/locate/csl

Offensive language detection in Tamil YouTube comments by adapters and cross-domain knowledge transfer

Malliga Subramanian [a,*], Rahul Ponnusamy [b], Sean Benhur [c], Kogilavani Shanmugavadivel [a], Adhithiya Ganesan [a], Deepti Ravi [a], Gowtham Krishnan Shanmugasundaram [a], Ruba Priyadharshini [d], Bharathi Raja Chakravarthi [e]

[a] Kongu Engineering College, Perundurai, Erode, Tamil Nadu, India
[b] Indian Institute Of Information Technology and Management-Kerala, Kerala, India
[c] PSG College of Arts and Science, Coimbatore, Tamil Nadu, India
[d] ULTRA Arts and Science College, Madurai, Tamil Nadu, India
[e] Insight SFI Research Centre for Data Analytics, Data Science Institute, National University of Ireland Galway, Galway, Ireland

ARTICLE INFO

Keywords:
Adapter
Cross-domain analysis
Finetuning
HASOC
Multilingual
Machine learning models
Offensive texts
Transformer models

ABSTRACT

Over the past few years, researchers have been focusing on the identification of offensive language on social networks. In places where English is not the primary language, social media users tend to post/comment using a code-mixed form of text. This poses various hitches in identifying offensive texts, and when combined with the limited resources available for languages such as Tamil, the task becomes considerably more challenging. This study undertakes multiple tests in order to detect potentially offensive texts in YouTube comments, made available through the HASOC-Offensive Language Identification track in Dravidian Code-Mix FIRE 2021.[1] To detect the offensive texts, models based on traditional machine learning techniques, namely Bernoulli Naïve Bayes, Support Vector Machine, Logistic Regression, and K-Nearest Neighbor, were created. In addition, pre-trained multilingual transformer-based natural language processing models such as mBERT, MuRIL (Base and Large), and XLM-RoBERTa (Base and Large) were also attempted. These models were used as fine-tuner and adapter transformers. In essence, adapters and fine-tuners accomplish the same goal, but adapters function by adding layers to the main pre-trained model and freezing their weights. This study shows that transformer-based models outperform machine learning approaches. Furthermore, in low-resource languages such as Tamil, adapter-based techniques surpass fine-tuned models in terms of both time and efficiency.

Of all the adapter-based approaches, XLM-RoBERTa (Large) was found to have the highest accuracy of 88.5%. The study also demonstrates that, compared to fine-tuning the models, the adapter models require training of a fewer parameters. In addition, the tests revealed that the proposed models performed notably well against a cross-domain data set.

* Corresponding author.
E-mail addresses: mallinishanth72@gmail.com (M. Subramanian), rahul.mi20@iiitmk.ac.in (R. Ponnusamy), seanbenhur@gmail.com (S. Benhur), kogilavani.sv@gmail.com (K. Shanmugavadivel), adhithiyaganesan@gmail.com (A. Ganesan), deeptiravi01@gmail.com (D. Ravi), gowthamkrish1145@outlook.com (G.K. Shanmugasundaram), rubapriyadharshini.a@gmail.com (R. Priyadharshini), bharathi.raja@insight-centre.org (B.R. Chakravarthi).

[1] https://competitions.codalab.org/competitions/31146.

https://doi.org/10.1016/j.csl.2022.101404
Received 26 January 2022; Received in revised form 7 May 2022; Accepted 10 May 2022
Available online 17 May 2022

1. Introduction

People all over the world are increasingly using social media to share information and communicate. However, the widespread use of social media and microblogging platforms may have a parallel negative impact on people's well being. Hateful and offensive comments proliferate on social media, often by toxic users hiding behind the anonymity features of these platforms (Blair, 2003; Lee and Kim, 2015). If such toxic behavior is not addressed in a timely manner, it can have a cascading effect, discouraging other people from becoming involved in the online community (Obadimu, 2020). It is likely that such a hostile setting will prevent people from expressing themselves freely because of the fear of being abused or harassed. Consequently, offensive language, hate speech, and other unpleasant content available on the internet constitute a threat to the general welfare of our society (De Smedt et al., 2018). Since offensive language has a significant impact on public opinion, platforms such as YouTube, Facebook, and Twitter have policies and procedures in place to moderate hate speech content and related objectionable behavior in order to mitigate the negative consequences on society (Alkiviadou, 2019).

Due to the development of social media, users from multilingual societies have been posting and commenting in a code-mixed format in recent years. Code-mixed format is one in which a sentence contains vocabulary and grammar derived from more than one language (Chakravarthi et al., 2020b). It is common for multilingual people to communicate through code-mixed text, since they lack the ability to express themselves in a single language (Suryawanshi et al., 2020). Due to the lack of regulation, a considerable amount of offensive content is frequently shared in the code-mixed format on various social media networks. Since it is impossible to identify such content manually among the massive quantity of data generated on social media, automated moderation of this content is imperative. Such a need has prompted a large number of researchers in the field of Natural Language Processing (NLP) to design computational systems capable of limiting the spread of objectionable content or removing them altogether by utilizing state-of-the-art NLP techniques. Many dedicated workshops and evaluation campaigns have been conducted (Schmidt and Wiegand, 2019; Suryawanshi and Chakravarthi, 2021; Liu et al., 2019a) to attract the active participation of the NLP community in detecting offensive contents. This has also led to the creation of a corpus for Dravidian languages such as Tamil, Malayalam, and Kannada in the context of sentiment analysis and offensive language detection tasks (Chakravarthi et al., 2022b).

There are a number of NLP systems that can automatically identify hateful and offensive texts. These systems can be categorized under machine learning models and deep learning-based multilingual transformer models. The current study aims to classify offensive language from a Tamil code-mixed data set of comments and posts collected from YouTube (Chakravarthi et al., 2021) using various machine learning and multilingual transformer models. The study's focus is limited to Tamil, a classical language originating in India and spoken in Tamil Nadu, India's southernmost state, as well as Sri Lanka, Malaysia, and Singapore (Thavareesan and Mahesan, 2021, 2020b,a, 2019; Sakuntharaj and Mahesan, 2016, 2017, 2021). There are small inscriptions in the city of Adichanallur that date back to 905 BC to 696 BC. Tamil has agglutinative grammar, which means that suffixes are used to show noun class, number, case, verb tense, and other grammatical types. Tamil's standard metalinguistic terminology and scholarly vocabulary is itself Tamil, as opposed to the Sanskrit used by most Aryan languages. Besides dialects, Tamil has a lot of different styles: a classical literary style that was written in the old language, a modern literary and formal style, and the current colloquial form (kotuntami). These styles blend into each other, making them look like they belong together. It is possible to write centami with words from the cankattami language, for example, or to speak kotuntami with words from one of the other varieties. Tamil words are made up of a lexical root and one or more affixes, which are added to them. Most of the affixes in Tamil are endings. They can be derivational suffixes, which change the word's part of speech or meaning, or inflectional suffixes, which change things like person, number, mood, tense, and so on. There is no limit to how long and wide agglutination can be. This could lead to big words with many suffixes that need many words or a whole sentence in English (Subalalitha, 2019; Subalalitha and Poovammal, 2018; Anita and Subalalitha, 2019b,a; Srinivasan and Subalalitha, 2019; Narasimhan et al., 2018).

In comparison to transformer models, machine learning models with feature selection approaches are simpler and easier to interpret. So, various machine learning algorithms such as Naïve Bayes (MNB), Support Vector Machine (SVM), Logistic Regression (LR), Random Forest (RF), Decision Tree (DT), K-Nearest Neighbor (KNN), and a lot more have been used for creating models to classify code-mixed data set. Current studies on recent developments in offensive language detection suggest an increased use of deep learning-based transformer models. The majority of deep learning techniques require a large amount of manually labeled data, which limits their applicability in many fields with a scarcity of annotated resources (Tsvetkov, 2017). In such cases, models that can extract linguistic information from unlabeled data becomes an alternative to manually annotating the data, which can be time-consuming and expensive. Such models can be fine-tuned to classify small amounts of labeled data. This is called transfer learning and is a strategy in which a model is first pre-trained on large unlabeled text corpora using self-supervised learning before being used on labeled text corpora.

In recent years, transformer-based models such as BERT (Devlin et al., 2018), DistilBERT (Sanh et al., 2019), RoBERTa (Peters et al., 2019), XLM-RoBERTa (Conneau et al., 2019), MuRIL (Dave et al., 2021), etc. have garnered attention for their ability to recognize and classify offensive texts via contextual and semantic learning. These models performed exceptionally well in classifying code-mixed texts of different languages with good accuracy (Chakravarthi et al., 2020b). These transformer models can be used in two ways: (i) by fine-tuning the transformer and (ii) by integrating and training an adapter on a transformer model. In the fine-tuning approach, the pre-trained models, such as BERT, RoBERTa, MuRIL, etc., are retrained in their entirety on a downstream task. Thus, this approach requires a huge number of parameters to be retrained. Adapters, on the other hand, are typically lightweight modules that are inserted between transformer layers (Mahabadi et al., 2021b; Semnani et al., 2019). Model tuning on a downstream task simply updates the parameters of adapters, while the weights of the original transformer layers remain unchanged. He et al. (2021) showed that adapter-based tuning outperforms fine-tuning on low-resource and cross-lingual tasks.

The present study uses a few machine learning models and transformer-based models to identify offensive language content consisting of code-mixed data of Tamil comments collected from YouTube, made available through the HASOC-Offensive Language Identification track in Dravidian Code-Mix FIRE 2021. Through this study, we address the following research questions:

RQ1. : What are the most predictive features that distinguish offensive contents in user-generated social media text in Dravidian languages?
Upon investigating several state-of-the-art systems on available benchmark data sets from the Dravidian Codemixed (HASOC 2021) shared tasks, we found that most of the submitted systems used traditional machine-learning approaches. Therefore, the present study aimed to explore more deeply the most predictive features that would help the classifiers identify offensive texts in Tamil.

RQ2. : How effective are pre-trained transformer models in classifying offensive language content in user-generated social media text?
To answer this question, we trained and fine-tuned multiple transformer and adapter pre-trained models. To further prove our hypothesis (Deep learning-based transformer models perform better than machine learning models), we compared the performance of the transformer-based models as well.

RQ3 . : Is the knowledge about offensive language learned from a given domain useful in predicting offensive language in another domain?
The problem of offensive language is not constrained to a particular domain. So, we also examined the cross-domain transfer ability of offensive text detection. To this end, we used other available data sets in Tamil and tested the ability of the developed models to classify texts as misogynous and gauged their performance.

In this study, we created machine learning-based models and transformer-based models. The main contributions of this study are as follows:

i. Extracted unigram-based Term Frequency-Inverse Document Frequency (TF-IDF) features and investigated the traditional machine learning models for offensive language detection.
ii. Fine-tuned pre-trained multilingual transformer models such as MBERT, MuRIL, and XLM Roberta.
iii. Implemented an efficient adapter module with pre-trained models and assessed the effectiveness of adapters in offensive language identification in Tamil.
iv. Tested the cross-domain ability of models by training them on offensive content identification data set and testing them on the misogynous text identification data set.

Hence, the current work investigates whether texts in a corpus are offensive or not using various conventional machine learning classifiers and deep learning transformer models. The novelty of this research study lies in the integration of adapters into pre-trained transformer models and adaption of the developed models to cross-domain tasks. The potential benefits of social media platforms are obscured by the widespread use of offensive and abusive language; hence, this attempt focuses on finding solutions to this problem.

The rest of the article is structured as follows: Section 2 presents a brief overview of machine learning and transformer-based research attempts at offensive language detection. Section 3 details the task description, a summary of the data set, and the feature selection approach used in the study. Section 4 provides an explanation of the proposed models. Section 5 discusses the training procedure for all experiments, including hyperparameter selection and performance metrics. Results and analysis are delineated in Section 6. An in-depth analysis of error/mis-classification of the texts is presented in Section 7. Finally, Section 8 presents the conclusion along with a summary and recommendations for future work.

2. Literature survey

Offensive content on social media may have adverse implications for its users, including mental health issues, which can even lead to suicide attempts (Del Vigna et al., 2017). With the vast number of people using social media, it is impossible for researchers to manually identify and remove objectionable content. To preserve the social media ecosystem, researchers and stakeholders should work to develop computational models capable of quickly identifying and classifying objectionable contents. Automated techniques for filtering offensive language on social media have exploded in the last several years, but the subjective and context-dependent nature of the text has made it difficult to classify nuances (Vandersmissen, 2012; Schmidt and Wiegand, 2019). Multilingual users post texts containing more than one language, referred to as code-mixed data, which further adds to the challenge of identifying offensive content. There has been a shortage of research aimed at low-resource languages since many earlier systems are characterized by the exclusion of data in languages other than English (Schmidt and Wiegand, 2019; Cieri et al., 2016), resulting in a dearth of study in this area for languages with limited resources. To direct the attention of researchers, many shared tasks on low-resource languages have been released, and researchers have attempted to develop models for these tasks (Sampath et al., 2022; Chakravarthi et al., 2022a; Ravikiran et al., 2022; Priyadharshini et al., 2022; Bharathi et al., 2022). Four such attempts are described below.

2.1. Shared tasks

The findings on offensive contents in Tamil, Malayalam, and Kannada have been reported by Chakravarthi et al. (2021) based on the results of the first shared task[2] on offensive content identification in Dravidian languages. The authors provided a summary of the data set used for this task and an overview of the methodology and results of the proposed systems for this task. The authors believe that this work and data set will pique interest in and promote additional study on low-resource languages. In an innovative approach, Suryawanshi and Chakravarthi (2021) incorporated a multimodal classification challenge in the shared task "Troll Meme Classification in Tamil", which contained an enhanced data set named TamilMeme that included Tamil texts from memes. The findings of numerous models built for this task were also summarized. While this study presented a multimodal classification challenge, it also explored the problems associated with the NLP of a language- and code-mixed data set with limited resources. Additionally, Chakravarthi et al. (2021) described a shared task of machine translation for Dravidian languages (Tamil) that was presented at the first workshop on Speech and Language Technologies for Dravidian Technologies,[3] where the best-performing systems achieved a high Bilingual Evaluation Understudy Score despite a lack of training data. Chakravarthi et al. (2020a) collected a code-mixed data set of comments/posts in Dravidian languages (Malayalam-English and Tamil-English) from social media. A summary and the findings of this shared task on detecting offensive texts in Dravidian languages were presented. Based on an annotated data set, a wide variety of systems were tested for two different tasks in two languages. In this effort, it was found that many models rely on transformers and pre-trained embedding systems. These shared tasks stress the need for further study in the detection of offensive texts in under-resourced languages. Recently, several machine learning and NLP-based research attempts have made a breakthrough in detecting offensive texts on social media platforms. A comprehensive review of the techniques and findings of such attempts is provided below.

2.2. Machine learning models

Traditional machine learning algorithms use the features extracted using word-level and character-level n-grams, among others. Davidson et al. (2017) created a multi-class classifier to classify tweets as hate speech, offensive, or neither hate nor offensive using NB, DT, and RF, with a five-fold cross-validation and claimed model performance metrics of 0.91, 0.90, and 0.90 pertaining to precision, recall, and F1 score respectively. Gaydhani et al. (2018) created a machine learning technique for detecting hate speech and offensive language on Twitter using n-gram features weighted by TF-IDF. Using various sets of feature values and model hyperparameters, a comparative analysis of LR, NB, and SVM were carried out in this attempt. The results indicate that LR performs better with an optimal n-gram range of 1 to 3 for L2 normalization, and a 95.6% accuracy rate was obtained while assessing the model against test data. De Gibert et al. (2018) constructed a data set of hate speech that was manually labeled from Stormfront, a white supremacist internet forum that contains both hateful and non-hateful sentences. SVM, Convolutional Neural Networks (CNN), and Long Short Term Memory (LSTM) were utilized to annotate the test data using hand-annotated training data, of which the LSTM-based classifier obtained better results. Ayo et al. (2020) collected hate speech benchmark data sets to evaluate machine learning models for offensive text classification. This attempt evaluated the pros and cons of single and hybrid machine learning algorithms for text classification. Additionally, this study presented a general metadata architecture for categorizing hate speech on Twitter in order to address the challenges associated with categorizing hate speech in Twitter data streams. The proposed generic metadata architecture outperformed comparable approaches on all metrics.

Apart from investigating algorithms and data sets for offensive speech recognition, MacAvaney et al. (2019) examined the difficulties associated with online automated offensive speech detection. This work developed a multi-view SVM strategy using data sets such as HatebaseTwitter, Stormfront, and TRAC, and it was found to outperform neural approaches while remaining simpler and more interpretable than neural methods. Silva and Roman tested four distinct models (SVM, MLP, LR, and NB) (Silva and Roman, 2020) with various configurations in order to test their performance in detecting hate speech in tweets written in Portuguese. The results revealed that these algorithms outperform the benchmark LSTM in terms of micro-averaged F1 score as well as other results in the related literature.

Research by Putri et al. (2020) sought to automatically detect hate speech in Indonesian posts on Twitter. The models used NB, Multi-Level Perceptron (MLP), AdaBoost Classifier, DT, and SVMs. The study assessed the model's performance with and without Synthetic Minority Oversampling Technique (SMOTE), a commonly used oversampling method to solve the imbalance problem. The Multinomial NB (MNB) algorithm delivered the best model with a 93.2% recall and a 71.2% accuracy for detecting hate speech. Dave et al. (2021) classified the YouTube comments in Tamil, Malayalam, and Kannada into five classes using LR and linear SVM models. The comments were converted into word embeddings using TF-IDF and pre-trained MuRIL and fed into the classifiers. The authors found that MuRIL worked better for Malayalam, while TF-IDF worked better for Tamil and Kannada. Machine learning and text mining feature extraction techniques were used in an attempt by Mohapatra et al. (2021) to present a hate speech detection model. The authors collected code-mixed data having English and Odia from a Facebook public page and manually classified them into three categories. Machine learning models such as SVM, NB, and RF were trained using the features extracted from word unigram, bigrams, trigram, n-grams, word2vec, and TF-IDF. SVM with word2vec was found to outperform NB and RF models.

Nayel and Shashirekha (2019) proposed three classification algorithms, namely Linear classifier, SVM, and MLP, to train the data sets provided by the Hate Speech and Offensive Content Identification in Indo-European Languages (HASOC) shared task. This

[2] https://competitions.codalab.org/competitions/27654.

[3] https://competitions.codalab.org/competitions/27650.

task was applied for three languages, namely English, German, and Hindi. The authors claimed that the proposed models achieved good results considering their simplicity. Another study by Saroj et al. (2019) used the same task and proposed text classification approaches to distinguish between hate speech, profane language, and offensive contents by employing XGBoost and SVM classifiers. The best result was attained by XGBoost with an accuracy of 81%. Zampieri et al. (2019) analyzed various models submitted for the "OffensEval" task for identifying and categorizing offensive language in the Offensive Language Identification (OLID) data set. The models employed in the task submissions ranged from classical machine learning techniques, such as SVM and LR, to deep learning techniques, such as CNN, RNN, and BiLSTM, which included an attention mechanism. In the shared task on Tamil, Malayalam, and Kannada (Chakravarthi et al., 2021), a few machine learning-based approaches have been attempted. One such attempt by Andrew (2021) performed language-specific pre-processing and applied machine learning algorithms for offensive text classification. This effort produced a precision of 0.54, a recall of 0.73, and an F1 score of 0.61 for the Tamil language. For the Malayalam language, a precision of 0.94, a recall of 0.94, and an F1 score of 0.93 were obtained. Correspondingly, the Kannada language achieved a precision of 0.66, a recall of 0.67, and an F1 score of 0.63. Bharathi et al. (2021) presented a method for automatically identifying offensive languages in Dravidian languages using a variety of machine learning methods. On the test set, they received F1 scores of 0.95 for Malayalam, 0.7 for Kannada, and 0.73 for Tamil. Chakravarthi et al. (2020a) also summarized the methodologies, including the machine learning models used for offensive language identification from code-mixed data with comments in Tamil, Malayalam, and Kannada. The results of these methodologies seem to be appreciable. Although numerous feature extraction methods have been utilized in machine learning-based approaches, they require a common, well-defined feature extraction strategy. To increase the performance of hate speech and offensive content detection models, neural network models now use text representation and deep learning methodologies such as CNNs, Bi-directional LSTMs, and BERT (Ayo et al., 2020).

2.3. Deep learning and transformer models

While a variety of feature extraction techniques have been used in machine learning-based approaches, they all require a well-defined feature extraction strategy. To improve the performance of models for detecting hate speech and objectionable content, neural network models increasingly incorporated text representation and deep learning techniques such as CNNs, BiLSTM and BERT (Dowlagar and Mamidi, 2021). Dowlagar and Mamidi (2021) also proposed BERT- and multilingual-BERT (mBERT)-based models to detect hate speech and offensive content in English, German, and Hindi. In addition, Dowlagar and Mamidi (2021) used ELMO (Embeddings from Language Models) to find contextual word embeddings that captured the meaning of the words in their context as proposed by Sarzynska-Wawer et al. (2021). These models were compared with SVM and exhibited better results during classification. Liu et al. (2019a) investigated the following classifiers: a linear model with features derived from word unigrams, word2vec, and Hate-base; a word-based LSTM; and a fine-tuned BERT. This study used the OLID (Zampieri et al., 2019) data set gathered via the Twitter API by searching a specific set of terms and identified that BERT outperforms the other models. On the basis of current pre-trained language bidirectional encoder representation models, a study by Mozafari et al. (2019) presented BERT as a transfer learning approach. New fine-tuning strategies based on transfer learning have been used to evaluate BERT's ability to capture hateful and offensive content in social media using two publicly available data sets annotated for racism, sexism, hate, or offensive content.

While BERT (Devlin et al., 2018) and other models demonstrate success in monolingual NLP tasks, researchers have also developed several monolingual models for NLP in a variety of languages in addition to English. Furthermore, the multilingual model is another alternate strategy that has received scarce attention. XLM (Lample and Conneau, 2019) is a cross-language pre-training model that successfully extends the Masked Language Model (MLM), the full-length and training approach described in BERT, to several languages. In comparison to BERT, RoBERTa (Lan et al., 2019), a variant of BERT, makes use of a greater number of model parameters, a larger batch size, and a greater amount of training data. It is based on the BERT language masking strategy, which adjusts important hyperparameters in BERT, including the deletion of BERT's next sentence prediction task, allowing RoBERTa to be more extensible to downstream tasks when compared to BERT.

A study by Vasantharajan and Thayasivam (2022) presented extensive experiments using multiple deep learning and transfer learning models to detect offensive content on YouTube and proposed fine-tuning and ensembling multilingual transformer networks such as BERT, DistilBERT, and XLM-RoBERTa; Universal Language Model Fine-tuning for Text Classification (ULMFit); mBERT-BiLSTM; and deep learning models such as CNN-BiLSTM. These models classified offensive language from a Tamil code-mixed data set of comments and posts collected from YouTube (Chakravarthi et al., 2021). The experimental results showed that ULMFiT and mBERT-BiLSTM performed well for this Tamil code-mix data set. Benhur and Sivanraju (2021), created and tested two models based on mBERT and MuRIL on the shared task for identifying the offensive texts released by Chakravarthi et al. (2021). These two models pooled the last layers of pre-trained transformers, and both gave comparatively better performance. Xu et al. (2020) focused on detecting multilingual hate speech written in English and German by fine-tuning the XLM-RoBERTa for sentence embedding and extracting the layer with the best performance for slicing and splicing. This study used the task shared by Chakravarthi et al. (2020a). The performance of these models was compared with that of other models such as SVM, LR, BiLSTM, and fine-tuned XLMRoBERTa was found to possess the highest accuracy.

Hande et al. (2021) used a few transformer-based models such as IndicBERT, DistilBERT, ULmFiT, MuRIL, and XLMRoBERTa to classify code-mixed social media comments/posts in the Dravidian languages of Tamil, Kannada, and Malayalam. They created a custom data set by transliterating code-mixed texts into the respective Dravidian language, either Kannada, Malayalam, or Tamil, and observed that fine-tuning ULMFiT on the custom data set yielded the best results on the code-mixed test sets of all three languages.

Recently, in addition to fine-tuning transformer models, adapter-based tuning has emerged as a viable alternative to fine-tuning. It works by incorporating lightweight adapter modules into a pre-trained language model and modifying the adapter modules' parameters only when learning on a downstream job. As such, it introduces only a few additional trainable parameters per new task, allowing for a high degree of parameter sharing. A brief overview of adapter-based models is presented here. Rücklé et al. (2020) evaluated the computation efficiency of adapters and removed adapters from lower transformer layers during training and inference. They demonstrated that this approach dynamically reduced the computational overhead associated with inference across multiple tasks while preserving task performance. Additionally, the adapters from AdapterFusion (Pfeiffer et al., 2020a) were pruned to increase inference efficiency while maintaining task performance. Various adapter models that have been shown to work well for machine translation were also presented in Pfeiffer et al. (2020a). Further, Artetxe et al. (2019) and Pfeiffer et al. (2020c) leveraged the modular architecture of adapters for parameter-efficient transfer to new languages or tasks. Artetxe et al. (2019) trained a transformer-based masked language model on one language and then transferred it to another using a new embedding matrix. This model was found to be comparable with mBERT. Pfeiffer et al. (2020c) introduced MAD-X, a framework based on adapters that provides great portability and parameter-efficient transfer to arbitrary tasks and languages through the acquisition of modular language and task representations. Additionally, this study proposed a novel invertible adapter architecture and a robust baseline technique for converting a multilingual model that has been previously trained to a new language. Artetxe et al. (2019) and Pfeiffer et al. (2020c) showed that a monolingual model trained in one language learns semantic concepts that are generalizable to other languages. He et al. (2021) identified that current adapter-based models focus on parameter efficiency but lack effectiveness. Since adapter-based fine-tuning yields representations with less deviation from those generated by the current adapter-based models, adapter-based tuning better mitigates forgetting issues than fine-tuning alone. So, adapter-based tuning outperforms fine-tuning on low-resource and cross-lingual tasks. This study proposed and ran two models based on RoBERTa using low- and high-resource tasks and demonstrated that adapter-based tuning gives better performance. A comparison between linear classifiers and neural networks yielded inconsistent results across data sets and architectures, with linear classifiers proving to be very competitive, if not superior. Systems based on pre-trained language models, on the other hand, have been proven to have the best performance in this area, achieving new state-of-the-art results. However, one limitation in the case of these pre-trained models is that they are suited only for general-purpose language comprehension tasks because of their training language variety. So, to support low-resource and cross-lingual tasks, adapter-based transformer models have also been introduced.

From this review of state-of-the-art research studies, it can be seen that some of the machine learning models also performed equally well by using hybrid approaches for feature selection/extraction from texts rather than single feature selection/extraction algorithms. In comparison to utilizing a transformer, the advantage of employing machine learning models with feature selection techniques is that the model remains simple for easier interpretation; hence, it forms the basis of the study we describe in this paper.

It was also observed that, in many attempts by researchers, transformer-based models excelled in classifying the offensive texts in monolingual and code-mixed data. Fine-tuning a transformer model for a new task, however, would consume a huge amount of time; hence, adapter modules were recommended. However, we could not find any study that used adapter models for offensive text classification in Tamil.

Based on the no free lunch theorem Ho and Pepyne (2002) and our review of current works dealing with data set disparity, we concluded that a method that works well for one data set may not be suitable for another. This means that there might not be a single classifier that performs well on all kinds of data sets. We found that a few pre-trained transformer models have been developed for under-resource languages such as Tamil. To the best of our knowledge, no study on adapter-based tuning of transformer models has been undertaken yet; hence, the present study aims to develop such models. In addition, we also attempt to adapt the developed models for cross-domain tasks to verify their effectiveness. The term "cross-domain" refers to the transfer of learning from the developed models to a different data set published by different studies but not necessarily in different hatred domains.

To this end, we demonstrated the performance of four different machine learning algorithms in the classification of Tamil texts as Offensive or Not Offensive. Furthermore, we developed five transformer models for classifying the data set. These five models were employed as fine-tuning and adapter-based transformer models. The details of the models are presented in Section 4.

3. Pre-processing

3.1. Task description

This study identified offensive texts based on the common definition for offensive language often expressed as 'flames', which refers to "offensive messages or remarks that in some circumstances are inappropriate, exhibit a lack of respect towards certain groups of people or are just rude in general" (Razavi et al., 2010). The data set used to identify offensive language content consisted of code-mixed data of Tamil comments collected from YouTube, made available through the HASOC-Offensive Language Identification track in Dravidian Code-Mix FIRE 2021.[4] The data set includes comments in Tamil, English, and Malayalam, but none of the comments is entirely in English or Malayalam; these languages are mixed with Tamil (Kumaresan et al., 2021). The comments in the data set contain more than one sentence, but the corpora's average sentence length is one (Chakravarthi et al., 2021). Each comment in the data set was annotated either as Offensive (OFF), Not Offensive (NOT), or Not Tamil. The data set contained 5880 Tamil texts for training and 654 Tamil texts for testing, with class labels as Offensive, Not Offensive, and Not Tamil. The training set had 1153 texts classified as Offensive and 4724 as Not Offensive texts. Three texts mixed with Tamil and English/Malayalam were omitted. The distribution of classes in the data set is shown in Fig. 1. Sample Offensive and Not Offensive texts from the data set are presented in Fig. 2.

[4] https://competitions.codalab.org/competitions/27654.

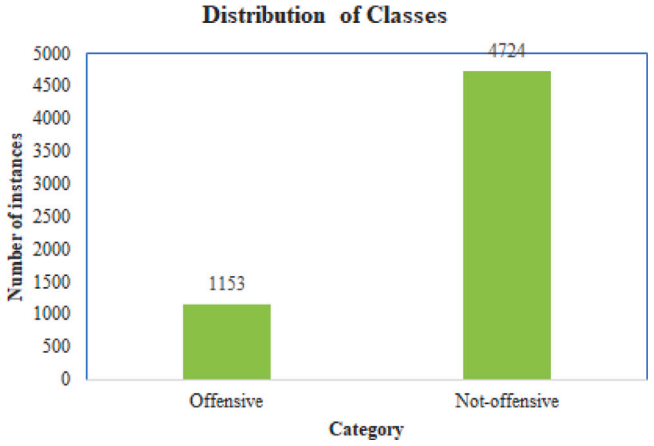

Fig. 1. Distribution of classes.

Documents	Texts	Label
tam1	திருமலை நாயக்கர் பேரவை சார்பாக படம் வெற்றி பெற வாழ்த்துக்கள் **(Congratulations for the success of the film on behalf of the Thirumalai Nayakar Council)**	NOT
tam108	படத்துல என்னமோ பெருசா இருக்குனு மட்டும் தெரிது **(It is only known that there is something big in the picture)**	NOT
tam1=352	இந்தப் படத்தைப் எல்லோரும் தியேட்டர்ல போய் பார்க்கணும் கண்டிப்பா **(Everyone must go to the theater to see this movie)**	NOT
Tam2	இந்த ட்ரெய்லர் கூட பார்க்கிற மாதிரி இல்லை.. இதை தியேட்டர் ல போய் பார்க்கனுமா.. . **(This trailer is not even like watching .. Can you go and see this in the theater .)**	OFF
tam1115	இதை விட கேவலமாக ஒரு டிரைலர் நான் பார்த்ததில்லை **I have never seen a trailer worse than this**	OFF

Fig. 2. Sample training texts from the data set.

3.2. Pre-processing

In real-world situations, data is frequently incomplete, inconsistent, and/or deficient in specific behaviors or trends, and it is likely to contain a significant number of errors. Pre-processing of such data constitutes a significant part of NLP. Clean data is essential for classifying any text with high accuracy. As a result, preparing data is the first step in NLP before analyzing or categorizing it. It is necessary to convert raw data into a format that the NLP models can understand. Since the corpus included emojis, punctuation characters, and words that were not in Tamil, these characters had to be deleted before the rest of the data could be processed properly. The following steps were performed for pre-processing.

1. **Removal of emojis:** Text messages can contain emojis, which are pictorial representations of an idea or emotion. It is possible to deal with them in one of two ways: Either a textual term can be used to substitute an emoji, or they can simply be removed altogether. In this study, we chose to remove the emojis and emoticons from the text because they generally do not convey any semantic value.
2. **Removal of punctuation characters, digits, and texts that are not in Tamil:** In addition to the removal of emojis, we also removed various punctuation characters, such as !, ?, etc., and digits. Even though these characters improve readability, they do not serve the classification goal.

3.3. Feature selection

In machine learning, feature selection is the process of selecting a subset of relevant features from a data set to use in the creation of a model. There are numerous features that must be considered while developing a machine learning model for text classification. Since features are derived from words, the corpus's context is broader, resulting in a higher dimensionality for features. The completion of these features require a significant amount of time and processing power. Word embeddings or word vectorization is a technique used in NLP to map words or phrases from vocabulary to a vector of real numbers. Word embeddings help in the following use cases:

- Computation of similar words
- Text classifications
- Document clustering/grouping
- Feature extraction for text classifications
- Natural language processing

Term Frequency — Inverse Document Frequency (TF-IDF) vectorization is a popular vectorization technique. TF-IDF is a statistical measure that evaluates how relevant a word is to a document in a collection of documents and is given as

$$TF - IDF = TermFrequency(TF) * InverseDocumentFrequency(IDF), \tag{1}$$

where TF is the measure of the frequency of a word in a document and is expressed as

$$TF = (Number\ of repetitions\ of\ a\ word\ in\ a\ document)/(number\ of\ words\ in\ a\ document) \tag{2}$$

IDF is the inverse of document frequency which measures the informativeness of a word in the document and is calculated as

$$IDF = log((Number\ of\ documents)/(Number\ of\ documents\ containing\ the\ word)) \tag{3}$$

After pre-processing, the texts in both the training and test data sets were tokenized into words and converted into feature vectors using TF-IDF vectorization. From these feature vectors, important features were chosen and fed into the classifiers for training.

4. Proposed classifiers

The pre-processed data set is sent to the feature engineering module to obtain feature vectors. While training vectors are used to train classifiers, development/test vectors are used to evaluate the models. According to the no free lunch theorem (Ho and Pepyne, 2002), there is no single classifier that can achieve best performance on all kinds of data sets. Therefore, it is recommended to apply several different classifiers on feature vectors to identify the one that exhibits better results. Hence, we selected two kinds of models, namely machine learning models and pre-trained multilingual transformer models. The scikit-learn libraries in Python were used to implement the proposed models. The rationale behind the use of these models is presented in the following sections.

4.1. Machine learning models

Machine learning has matured significantly during the last few years and has the potential to fundamentally alter how people view important applications such as image identification, data mining, expert systems, and NLP. Machine learning can provide solutions in all of these areas and is capable of generating the kind of innovation on which humankind will rely heavily in the future. Numerous strategies have been developed in many research attempts to address the challenge of detecting offensive texts. These solutions have been built using a variety of machine learning and deep learning methods as well as transformer models. In this section, we describe the four machine learning models that were used for text classification in the present study.

NB is an efficient method that is used to analyze text input and solve problems involving binary and multiple classes. BNB is a probabilistic model and a variant of the NB method that is most frequently employed in NLP (Singh et al., 2019). BNB determines the probability of each tag in a given sample and outputs the tag with the highest probability.

SVMs are supervised classification algorithms that use training data to generate an optimal hyperplane for classifying new data. The support vectors produce a hyperplane that splits the examples into two non-overlapping classes in binary classification. SVM determines the maximum separation margin between hyperplanes. Typically, if a data set is linearly separable, there are infinite separating hyperplanes. SVM classifiers perform admirably in text classification tasks (Kotsiantis et al., 2007) and can be used to address problems involving classification and regression. One interesting attribute of SVM is that its learning performance is not dependent on the dimension of the feature space (Joachims, 1998).

LR is a machine learning method that is used to solve classification problems. It is a predictive analytic approach that is based on the concept of probability. LR is frequently utilized in a variety of data mining and machine learning use cases, where it is used to define response variables using one or more predictor variables (Hosmer and Lemeshow, 2000; Ginting et al., 2019). LR is used to forecast a discrete result using discrete, continuous, or mixed data. Thus, LR is a frequently employed technique when the dependent variable comprises two or more distinct outcomes. The outcome could be in the form of Yes/No, 1/0, True/False, or High/Low, given a set of independent variables. In this attempt, we employed L2 regularization to deal with data that contained a

binary dependent variable, such as Offensive or Not-offensive. It calculates the probability of output by combining the independent or prediction variables in a linear fashion.

KNN is a straightforward text classification method that classifies new data by comparing it to all previously classified data using a measure of similarity. While KNN is a straightforward technique that can be used for classification (Kovács et al., 2021) and regression tasks and makes no assumptions about the data, it is memory- and time-intensive due as it requires that all the training data be retained. The class of a text document is decided based on the classes of the K documents that are closest to it (Gao and Huang, 2017). KNN is a classification algorithm based on statistics, and it performs lazy learning. The testing step of the KNN algorithm is resource-intensive in terms of memory and time (Abro et al., 2020).

4.2. Transformer models

Despite the fact that Recurrent Neural Networks (RNN) are meant to take a series of inputs with no predetermined size limit and remember information learned from earlier input while generating output, they suffer from long-term dependency and poor training. Furthermore, they take the input sequentially, one by one, which does not make good use of GPUs as they are built for parallel computation. To overcome these constraints, a transformer with an attention mechanism has been suggested. A transformer is a unique architecture developed by Vaswani et al. (2017) that tries to handle sequence-to-sequence tasks while handling long-range dependencies with simplicity. It is the most recent cutting-edge approach in the field of NLP. A transformer is a deep learning model that uses the attention mechanism, differently weighting the significance of each element of the input data, and is typically employed in language translation, summarization, and text classification. A basic transformer consists of an encoder that reads the text input and a decoder that predicts a task.

The transformer is built with encoder–decoder modules. They are made up of modules with feed-forward and attention layers. Multiple identical encoders and decoders are layered on top of each other to form the encoder and decoder blocks. The encoder and decoder stacks have the same number of units. The attention mechanism examines an input sequence and determines whether other portions of the sequence are essential at each stage. If the input data is a natural language sentence, for example, the transformer does not need to process the beginning of the sentence before the end. Rather, it determines the context that gives meaning to each word in the phrase. The encoder and the decoder stack work as follows: Initially, the word embeddings of the input sequence are provided to the first encoder. They are then converted and propagated to the next encoder, and finally, the output from the last encoder in the encoder-stack is passed to all the decoders in the decoder stack.

4.2.1. Why transformers?

Many transformer-based NLP models have been developed particularly for transfer learning (Devlin et al., 2018; Raffel et al., 2019). Transfer learning is a strategy in which a model is first pre-trained on large unlabeled text corpora using self-supervised learning before being used on labeled text corpora (Aßenmacher and Heumann, 2020). Once this is accomplished, it is only minimally modified during fine-tuning on a given NLP downstream task (Devlin et al., 2018). A significant advantage of using transformers is that an existing pre-trained transformer model can be fine-tuned on a given data set, obtaining better outcomes with less work. Labeled data sets for specific NLP tasks are often small in size. When a model is newly trained on such a tiny data set without prior training, the outcomes are lower than when the model is trained on a larger data set with prior training. But an existing pre-trained model can be used to fine-tune a variety of downstream NLP tasks, such as text classification, summarization, question answering, and many more. Transformers have recently been the model of choice for NLP challenges, replacing RNN models such as LSTM and Gated Recurrent Unit (GRU). Training on bigger data sets is made possible by the use of additional training parallelization, which reduces training time. One of the most significant milestones in the development of NLP in recent years has been the introduction of Google's BERT, which has been marked as the start of a new era in NLP. Consequently, pre-trained BERT models such as mBERT, Robustly Optimized BERT Pre-training Approach (RoBERTa), Multilingual Representations for Indian Languages (MuRIL), and many more have been developed. Three models that serve as the basis for the models proposed in this study are described below.

4.2.2. mBERT (BERT multilingual base model (cased))

mBERT is a self-supervised transformer model pre-trained on a large corpus of multilingual data (Devlin et al., 2018; Pires et al., 2019). It is trained entirely on raw texts, with no human labeling, and uses an automatic mechanism to generate inputs and labels from those texts. mBERT is a 12-layer transformer (Devlin et al., 2018), but instead of being trained on monolingual English data with an English-derived vocabulary, it is trained on Wikipedia articles from 104 languages with a shared word piece vocabulary. It makes no use of a marker to indicate the input language, and there is no explicit mechanism in place to encourage translation equivalent pairs to have similar representations.

4.2.3. XLM-RoBERTa

XLM-RoBERTa is a multilingual variant of RoBERTa (Liu et al., 2019b). XLM-RoBERTa is a self-supervised transformer model that has been pre-trained on 2.5TB of filtered CommonCrawl data from 100 languages. This model does not require lang tensors to understand which language is being used, and it should be able to determine the proper language based on the input IDs. It employs the same training approach as the RoBERTa model, which uses the MLM technique without the Next Sentence Prediction (NSP) technique. The training procedure entails sampling streams of text from various languages and masking some tokens so that the model can anticipate the missing tokens. Since no linguistic embeddings are used, the model can deal with code-switching more effectively. XLM-RoBERTa has demonstrated outstanding performance in a variety of multilingual NLP tasks.

4.2.4. MuRIL

The primary goal of MuRIL is to increase the efficiency with which certain downstream NLP activities are performed. MuRIL is the latest multilingual model to be introduced by Google, and it is intended to improve interoperability between languages. MuRIL surpasses mBERT on all tasks in the challenging cross-lingual XTREME test (Hu et al., 2020).

In our review of extant literature, we found two methods for utilizing pre-trained transformer models: fine-tuning a transformer model and integrating an adapter into a transformer model. A quick overview of these methods are provided in 4.2.5 and 4.2.6.

4.2.5. Fine-tuning a transformer model

Since training a transformer model such as BERT from scratch on a small data set would result in overfitting, it is preferable to employ a pre-trained model developed on a large data set. Then, the model can be refined by training it on a smaller data set; this is referred to as model fine-tuning. There are several advantages to fine-tuning, including rapid model building, task tuning on a considerably smaller data set, and attaining state-of-the-art results with minimal task-specific changes in a variety of tasks such as classification, summarization, etc. During fine-tuning, one or more dense layers of neurons and a classification layer are added on top of pre-trained models. Then, the new model is trained on a new downstream classification task. The transformer models mBERT, XLM-RoBERTa (Base and Large), and MuRIL (Base and Large) were fine-tuned in this attempt.

4.2.6. Adapters - parameter-efficient transfer learning

In the context of transformer-based NLP models, parameter inefficiency occurs when a completely new model must be trained for each downstream task, resulting in an overly large number of parameters. Additionally, these enormous amounts of weights must be stored for inference. To address these issues, adapter modules have been introduced.

Adopting adapters, as presented by Houlsby et al. has been shown to be a viable alternative to the full fine-tuning of most tasks (Houlsby et al., 2019; Peters et al., 2019). Recently, adapters have exhibited remarkable performance in multi-task and cross-lingual transfer learning (Pfeiffer et al., 2020a). Adapter modules result in a compact and extensible model since they are a tiny collection of newly initialized weights added to each layer of the transformer. These weights are subsequently trained during fine-tuning, while the pre-trained parameters are frozen. As the original network's parameters remain constant, the adapters result in a high degree of parameter sharing. Only a few trainable parameters are required for each task; these additional task-specific parameters are referred to as adapters. Training of a large number of task-specific and language-specific adapters enables effective parameter sharing across tasks. Transfer learning becomes extremely efficient when adapter modules are used. The main component, the pre-trained model, is shared by all downstream tasks.

Recently, adapter-based models have been used by Houlsby et al. (2019), Peters et al. (2019) Pfeiffer et al. (2020b), Kim et al. (2021a) to fine-tune pre-trained transformer models. The attempts by Houlsby et al. (2019) and Pfeiffer et al. (2020b) used a benchmark data set called General Language Understanding Evaluation (GLUE) containing English sentences for evaluating understanding levels such as question answering, sentiment analysis, textual entailment, etc. Peters et al. (2019) and Kim et al. (2021a) used a diverse set of target tasks such as sentiment analysis, natural language inference, sentence pair tasks, relationship classification, review sentiment, and many more, containing only English sentences. We found that the state-of-the-art adapter models have not been applied and evaluated against the performance for low-resource languages such as Tamil. In the present research study, we adapt the adapter model proposed by Houlsby et al. (2019) to classify offensive texts in Tamil. The architecture of the proposed adapter module and its integration with the transformer layer is depicted in Fig. 3.

The transformer's layers are divided into two basic sub-layers: an attention layer and a feedforward layer. The attention layer weights the significant part of the input text differently and gives context for every place in the input sequence. In NLP, this layer aids in the memory of long source texts. Based on the information from the attention layer, the feedforward layer generates a new representation. After the attention and the two feedforward levels, an adapter module is inserted twice into each transformer layer. The adapter is made up of a bottleneck layer with fewer parameters in comparison to the attention and feedforward layers in the original model. The blue layers are trained on the chosen data set task during adapter tuning, which includes the adapter and normalization layer parameters. As can be seen, an adapter module is a simple two-layer feedforward network with non-linearity. The network's hidden dimensionality is low, implying that the total number of parameters in the adapter is also less. This is what makes adapters so effective.

In the proposed study, the above architecture is integrated into pre-trained transformer models such as mBERT, XLM-RoBERTa (Base and Large), and MuRIL (Base and Large) for identifying offensive texts. The reason behind utilizing these models is that they use cross-lingual model transfer, which involves leveraging task-specific annotations in one language/domain to fine-tune the model for evaluation in another language/domain. The training and testing process for transformer-based models is depicted in Fig. 4.

Furthermore, this study attempts to transfer the knowledge learned by the developed classifiers to a cross-domain data set. The cross-domain data set is obtained from Shared Task on Abusive Comment Detection in Tamil from DravidianLangTech@ACL 2022.[5] Instances in this data set have been classified into seven types of abusive comments, namely Misogyny, Misandry, Homophobia, Transphobia, Xenophobia, Counter Speech, and Hope Speech, with an additional category named None (Chakravarthi, 2020; Chakravarthi and Muralidaran, 2021a; Priyadharshini et al., 2022). From these instances, we selected those under Misogyny and None to create a data set. Since there were more number of samples classified as Misogyny, we extracted those instances. The number of instances in the Misogyny class was 1642, while 149 instances were not offensive. We examined the generalization capability of the models for cross-domain data set, the results of which are presented in Section 6.

[5] https://competitions.codalab.org/competitions/36403.

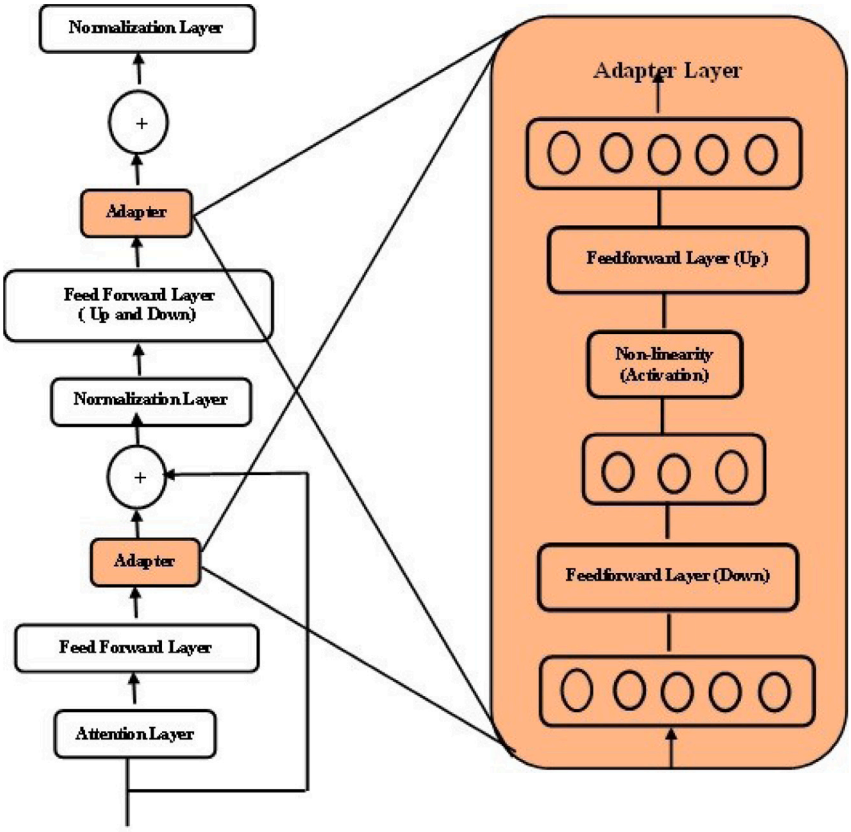

Fig. 3. Adapter Module integrated into Transformer.

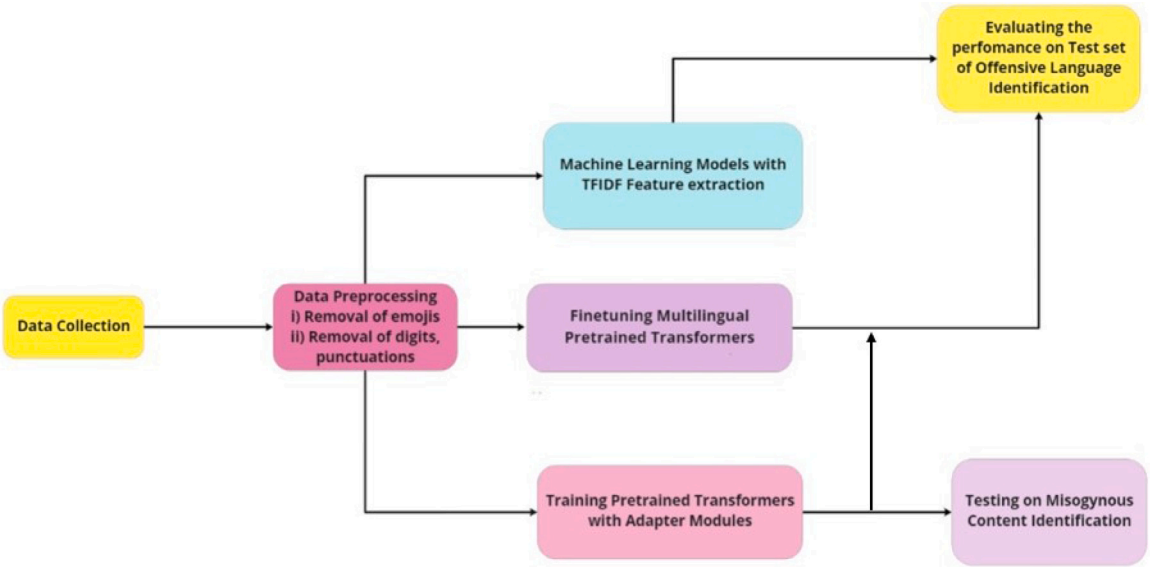

Fig. 4. Training and testing process for transformer models.

Table 1
Experimental platform.

Item name	Specifications
GPU	GPU DELL EMC 740
RAM	128 GB
GPU RAM	16 GB
OS	Ubuntu
Language	Python
IDE	Jupyter notebook

Table 2
Hyper-parameters and their search space.

ML models	Parameter	Search space
Bernoulli NB	Alpha	[0.5,0.6, 0.2, 0.3, 0.4, 0.7, **0.1**, 0.8, 0.9, 1.0]
	fit_prior	[True, **False**]
Logistic Regression	solvers	['lbfgs', **'liblinear'**]
	penalty	[**'l2'**]
	c_values	[**100**, 10, 1.0, 0.1, 0.01]
SVM	C	[0.05,0.1,0.2,0.3,0.25,0.4,0.5,0.6,0.7,0.8,0.9,**1**]
	gamma	[0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,**1.0**]
	kernel	[**'rbf'**,'linear']
KNN	n_neighbors	[15,20,25,30,35,**40**,45,50],
	weights	['uniform',**'distance'**],
	metric	['minkowski','euclidean',**'manhattan'**]

5. Experimental setup

This section describes the implementation details of the training process for all experiments, including the selection of hyperparameters, the hardware used, and the evaluation metrics used to compare the performance of each model. In order to analyze the performance of the ML models and adapter transformer models, a set of experiments were conducted. The details of the experiments are presented below.

5.1. Experimental platform

Python programming on Jupyter notebook and Colab notebook environment was used to train and test the proposed models. The Jupyter notebook environment does not require any setup. The required model architectures were imported from sklearn. We ran the proposed models on Graphical Processing Unit (GPU) because they may consume a lot of power due to parameter optimization and require high-performance hardware to operate well. Table 1 lists the hardware and software configurations used.

5.2. Tuning of hyper-parameters for ML models

Hyper-parameters are variables that define the network structure, such as the number of hidden units, dropout, activation function, and weight initialization, as well as how the network is trained, including the learning rate, momentum, batch size, and epochs. Hyper-parameter tuning is the process of determining optimum values for hyper-parameters used in a learning algorithm. The goal of hyper-parameter tuning is to find optimal values for hyper-parameters to minimize a loss function and provide better results. Manual search, random search, grid search, and Bayesian optimization are some of the optimization techniques. The present study used grid search optimization. In grid search, each combination of hyper-parameters is attempted. Grid search is used to find the optimal hyper-parameters of a model, which results in the most 'accurate' predictions. Even though it increases the time and computing power, it is the most effective, as the optimal solution is least likely to be missed. A different set of hyper-parameters has been used based on the models being developed. The hyper-parameters tuned in this study and their search spaces are summarized in Table 2.

Grid search optimization was run to find an appropriate set of hyper-parameters for each of the models. The accuracy of all the possible values of hyper-parameters was observed, and the set of hyper-parameters that led to the highest accuracy is highlighted in Table 2.

5.3. Training the transformer models: Fine-tuning and integrating adapters

Five different versions of transformers were considered in this study: mBERT, MuRIL - Base, MuRIL - Large, XLM-RoBERTa - Base, and XLM-RoBERTa - Large. These transformers were used in two modes: fine-tuning the entire weights and training only adapter modules. For fine-tuning, a classification layer was added on top of the pre-trained models. The entire pre-trained models were then retrained on training data set, and the output was fed to the softmax classification layer. While training the models, the

error was propagated backward through the entire architecture, and the model's pre-trained weights were adjusted in accordance with the training data set. The classification layer had two neurons as the data set contained two labels, namely Offensive and Not Offensive. In the second approach, adapters were trained in the same way as the model was fine-tuned to its maximum potential. The pre-trained weights of the transformer layers were fixed, with just the adapter weights being trained, in contrast to full fine-tuning. Since the pre-trained weights were fixed, adapter weights were encapsulated within transformer weights, requiring them to acquire compatible representations across tasks. Furthermore, a task-specific classification head was added on top of the transformer model's final layer to map the final hidden state to one of two classifications: Offensive or Not Offensive. The training procedure is detailed below:

1. Add a new adapter to each of the transformer models.
2. Add a binary classification head to the top of the last layer.
3. Configure the training process using appropriate training arguments.
4. Train the adapters as follows:

    i. Freeze all the weights of the pre-trained model so that only the adapter weights are updated during training.
    ii. Activate the adapter and the prediction head such that both are used in every forward pass.

The training arguments used for training the proposed transformer models were learning rate, number of training epochs, the batch size for training, the batch size for evaluation, and logging steps. The learning rate was set to 0.0001 for both the cases. The batch size for training and evaluation were 8 for the fine-tuning approach and 32 for adapter-based tuning. The number of training epochs was 3 for fine-tuning and 50 for adapter-based tuning. When compared to adapter-based tuning, the values of the training arguments for the fine-tuning strategy were set to be considerably small. This is because fine-tuning requires a significant amount of time to train as it retrains all layers of the model.

5.4. Performance metrics

The performance of the different models used for the classification was evaluated using the following metrics: accuracy, precision, recall, and F1 score ((Ayo et al., 2020). These metrics are commonly used for the evaluation of classifiers and are defined as follows. Accuracy is defined as the number of texts correctly classified as belonging to a specific class divided by the total number of texts in that class, as represented in Eq. (4).

$$Accuracy = \frac{(TP + TN)}{(TP + TN + FP + FN)} \tag{4}$$

where, TP is true positive (the number of correctly classified texts for each class), TN is true negative (the number of texts correctly classified in other classes except the correct class), FP is false positive (number of texts misclassified in other classes except the right class), and FN is false negative (the number of texts misclassified in the relevant class).

The number of texts correctly categorized as a certain class out of the total number of actual texts in that class is defined as recall (also known as sensitivity or true positive rate) and is computed using Eq. (5).

$$Recall = \frac{TP}{TP + FN} \tag{5}$$

Precision (positive predictive value) is defined as the number of texts accurately categorized as a specific class out of the total number of texts categorized as that class and is given by Eq. (6).

$$Precision = \frac{TP}{TP + FP} \tag{6}$$

F1-score is defined as the harmonic average of precision and recall; that is, it is the weighted average of precision and recall. It is calculated as in Eq. (7).

$$F1score = \frac{2 - Precision - Recall}{Precision + Recall} \tag{7}$$

Macro Average: The macro average is calculated as the unweighted average of the class-wise scores. Macro average gives equal weight to each of the two classes in the data set to find the final averaged metric. For instance, macro average precision is calculated as

$$Macro\ average\ precision = (Precision_{Not-offensive} + Precision_{Offensive})/2 \tag{8}$$

Weighted Average : The weighted average is obtained by taking the weighted average of class-wise scores, where the weights are proportional to the number of instances of each of the classes; that is, each class's contribution to the average is weighted by its size. It is calculated as follows:

$$\begin{aligned} Weighted\ average\ precision = (&Precision_{Not-offensive} * \ Proportion\ of\ Not-offensive\ samples) \\ &+ (Precision_{Offensive} * Proportion\ of\ offensive\ samples) \end{aligned} \tag{9}$$

Fig. 1 shows that the number of instances of the two classes is unequally distributed, indicating that the weighted average, which is biased in favor of frequent classes, may underestimate the error in less frequent classes. The macro-average score mitigates this disadvantage by treating each class equally, and thus it can provide a more accurate representation of the model's performance on infrequent cases.

Table 3
Performance of ML models for varied number of features.

ML models	Testing accuracy (%) for top 'N' features					
	900	1000	1050	1100	1200	1500
BNB	79.951	80.336	80.428	**80.429**	80.416	80.425
SVM	79.152	79.318	79.663	**79.666**	79.641	79.664
LR	80.864	80.928	81.345	81.341	**81.631**	81.041
KNN	80.418	79.921	**81.651**	81.172	81.521	81.712

Table 4
Precision, recall, and F score for ML models.

Classifiers	Class labels/ Macro or Weighted avg	Testing acc (%)	Precision (%)	Recall (%)	F1-Score (%)
BNB	Not-Offensive	80.428	85.664	91.418	88.448
	Offensive		43.902	30.508	36.000
	Macro Average	–	64.783	60.963	62.224
	Weighted Average	–	78.129	80.428	78.985
SVM	**Not-Offensive**	79.663	**89.587**	85.075	87.273
	Offensive		44.828	**55.085**	**49.430**
	Macro Average	–	67.208	70.080	68.351
	Weighted Average	–	81.511	79.664	80.445
KNN	**Not-Offensive**	**81.651**	83.016	**97.575**	**89.708**
	Offensive		45.833	09.322	15.493
	Macro Average	–	64.425	53.448	52.601
	Weighted Average	–	76.307	81.651	76.318
LR	Not-Offensive	81.345	88.993	88.333	88.662
	Offensive		**46.610**	48.246	47.414
	Macro Average	–	67.801	68.289	68.038
	Weighted Average	–	81.605	81.346	81.472

6. Experimental results

This section delineates the results of the models built for classifying texts in the data set obtained from Dravidian Code-Mix FIRE 2021 into Offensive and Not Offensive. We trained each classifier using the features extracted from the training set and tested the models using the test data set provided.

6.1. Results of ML models

The experiments to assess the performance of the proposed machine learning models were carried out using the tuned hyper-parameters highlighted in Table 2, and these values produced the best results during training. For training the models, the feature vectors obtained from TF-IDF vectorization were used. TF-IDF produces a vocabulary of words learned from the training data, and the top 1050 features were chosen. These top 1050 important features as calculated by TF-IDF were chosen through rigorous experiments. We ran the models by taking the top 900, 1000, 1050, 1100, 1200, and 1500 features. The testing accuracy produced by machine learning models for varying number of features is listed in Table 3. It can be observed from Table 3 that there is no significant improvement in accuracy beyond 1050 features. Table 4 shows the overall testing accuracy with precision, recall, and F1 score for both the classes, namely Offensive and Not Offensive for each of the models.

Based on indices such as TP, TN, FP, and FN, confusion matrices were obtained. A confusion matrix compares the actual labels with those predicted by a machine learning model that helps to perform an error analysis of the proposed models. The confusion matrices for the proposed machine learning models are shown in Fig. 5. The diagonal elements of the confusion matrix represent correct classifications. The predicted classes are represented by the y-axis, and the actual classes are represented by the x-axis. For instance, Fig. 5 shows that the BNB model classified 36 offensive texts correctly as offensive and 82 non-offensive texts incorrectly as offensive.

6.2. Results of fine-tuning transformer models

For fine-tuning, a classification layer was added on the top of the pre-trained models, and the entire pre-trained models were retrained on the training data set. The prediction results of the fine-tuned models on the test data set are presented in Table 5. The confusion matrices for the fine-tuned transformer models are shown in Fig. 6. Among all the models, MuRIL (large) gave the highest accuracy. Since we fine-tuned the models in entirety, the training of these models was a time-intensive process. The adapter transformer models proposed in this work were tested for their performance using the test data set. The values of the performance metrics for the testing data set of all the adapter transformer models are presented in Table 6. It is evident from Table 6 that XLM-RoBERTa (large) gave the highest accuracy when compared to all the other adapter models. The confusion matrices for the adapter transformer models are shown in Fig. 7.

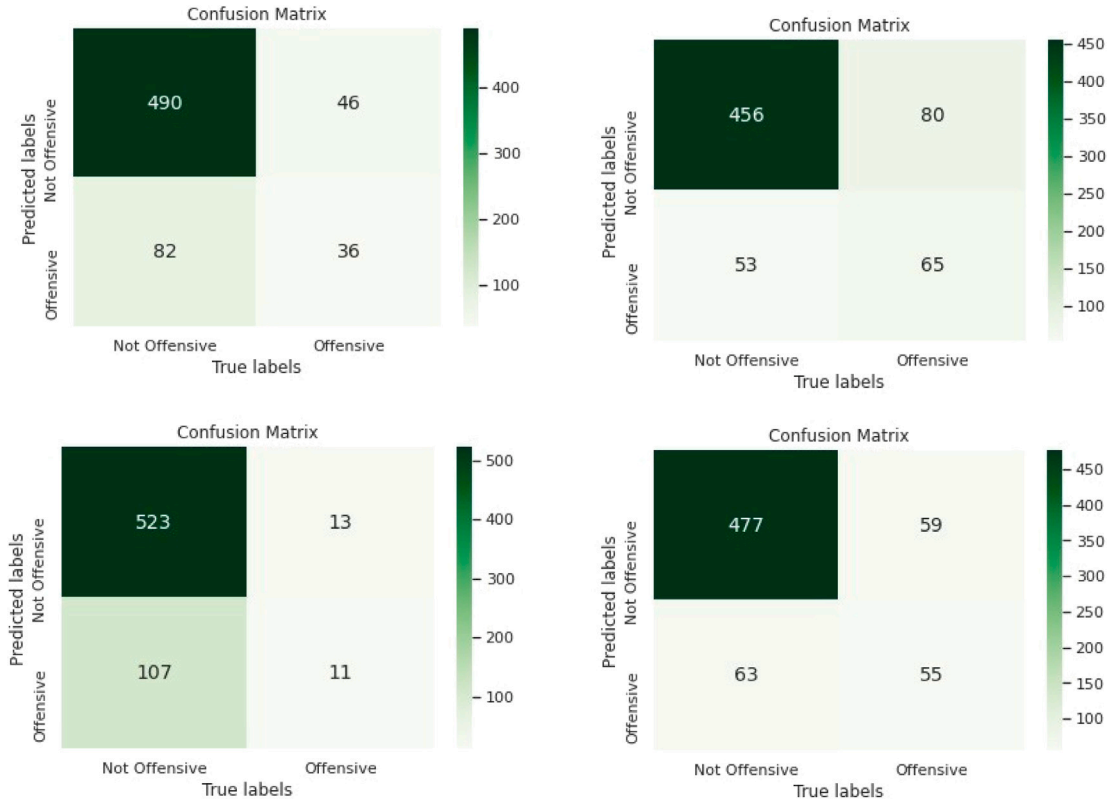

Fig. 5. Confusion matrix for Bernoulli NB, SVM, KNN, LR.

Table 5
Performance of fine-tuned transformer models.

Classifiers	Class labels	Testing accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
mBERT	Not Offensive	80.428	81.530	93.777	87.226
	Offensive		75.424	**47.340**	58.170
	Macro Average	–	78.477	70.559	72.698
	Weighted Average	–	79.775	80.428	78.873
XLM-RoBERTa - Base	Not Offensive	76.911	74.813	96.163	84.155
	Offensive		86.441	43.038	57.465
	Macro Average	–	80.627	69.601	70.810
	Weighted Average	–	79.027	76.911	74.483
XLM-RoBERTa - Large	Not Offensive	81.957	81.957	**100.00**	90.084
	Offensive		0.000	0.000	0.000
	Macro Average	–	40.979	50.000	45.042
	Weighted Average	–	67.170	81.957	73.830
MuRIL - Base	Not Offensive	68.960	63.806	97.436	77.114
	Offensive		**92.373**	35.974	51.781
	Macro Average	–	78.089	66.705	64.448
	Weighted Average	–	77.041	68.960	65.377
MuRIL - Large	Not Offensive	**88.226**	**94.030**	91.803	**92.903**
	Offensive		61.864	69.524	**65.471**
	Macro Average	–	77.947	80.664	79.187
	Weighted Average	–	88.866	88.226	88.499

6.2.1. Parameter efficiency of adapter models

Parameter inefficiency, in the context of transfer learning for NLP, arises when a model needs to be trained in entirety for every downstream task, and the number of parameters grows too large. Houlsby et al. (2019) proposed adapter modules that provide parameter efficiency by only adding a few trainable parameters per task and do not require that previous tasks be revisited as new tasks are added. The main idea of this attempt is to enable transfer learning for NLP on an incoming stream of tasks without training a new model for every new task. A standard fine-tuning model copies weights from a pre-trained network and tunes them on a downstream task, which requires a new set of weights for each task. In other words, fine-tuning involves adjusting

Table 6
Performance of adapter transformer models.

Classifiers	Class labels	Testing accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
mBERT	Not Offensive	85.015	**94.963**	87.759	91.291
	Offensive		39.831	63.514	48.958
	Macro Average	–	67.397	75.636	70.088
	Weighted Average	–	88.724	85.015	86.437
XLM-RoBERTa - Base	Not Offensive	84.862	91.064	90.092	90.842
	Offensive		54.237	58.716	56.388
	Macro Average	–	72.921	74.404	73.615
	Weighted Average	–	85.377	84.862	85.099
XLM-RoBERTa - Large	Not Offensive	**88.532**	93.097	**92.924**	**93.010**
	Offensive		**67.797**	**68.376**	**68.085**
	Macro Average	–	80.447	80.650	80.548
	Weighted Average	–	88.571	88.532	88.551
MuRIL - Base	Not Offensive	84.862	91.045	90.538	90.791
	Offensive		56.780	58.261	57.511
	Macro Average	–	73.912	74.399	74.151
	Weighted Average	–	85.020	84.862	84.939
MuRIL - Large	Not Offensive	87.920	92.910	92.393	92.651
	Offensive		65.254	66.957	66.094
	Macro Average	–	79.082	79.675	79.373
	Weighted Average	–	88.047	87.920	87.981

Table 7
Number of trainable parameters in fine tuning and adapter transformer model.

Classifiers	Number of trainable parameters	
	Fine-tuned transformer model	Adapter transformer model
M-BERT	177,854,978	4,223,490
MuRIL (Base)	237,557,762	1,486,658
MuRIL (Large)	505,909,250	1,486,658
XLM-RoBERTa (Base)	278,045,186	1,486,658
XLM-RoBERTa (Large)	559,892,482	1,486,658

the original parameters of each layer for each new task, thus limiting compactness. Fine-tuning is advantageous in that it could be more parameter-efficient if the lower layers of the network are shared between tasks. Fine-tuning large pre-trained models for transfer learning in NLP is effective but parameter-inefficient. For a total of N tasks, full fine-tuning requires N times the number of parameters of the pre-trained models. But in the proposed adapter models, we add new modules between layers of a pre-trained network called adapters. This means that parameters are copied over from pre-trained models (meaning that they remain fixed), and only a few additional task-specific parameters are added for each new task, all without affecting previous ones. Only the adapter blocks and layer normalization weights are modified when adapters are integrated into transformers. These layers have a small number of parameters. Surprisingly, the transition from fine-tuning the entire model to fine-tuning with orders of magnitude with fewer parameters has almost no negative effect on accuracy.

Table 7 shows the number of trainable parameters for the fine-tuned and adapter-based transformer models. Only 3(%) of the parameters or fewer are needed to be trained for adapter-based models, which almost matches or exceeds that of fully trained models. As a result, compared to fine-tuning, adapters are far more time- and storage-efficient.

In summary, adapter-based tuning demonstrates comparable performance to full fine-tuning while simultaneously maintaining high parameter efficiency. A comparison between machine learning and transformer models was performed, and the accuracy of all the models is presented in Table 8. In our experiments, we identified that XLM-RoBERTa (Large), a transfer learning-based adapter model, performed better for the data set under consideration. The experimental results demonstrate that typical machine learning models are unable to comprehend the context of the message and, as a result, may not be a good choice for sentiment analysis tasks in general.

6.3. Cross-domain transfer between data sets

Cross-domain transfer refers to the generalization of an offensive speech detection model trained on a given data set to other data sets with the same or similar class labels. In order to evaluate the cross-applicability between data sets, the pre-trained models were tested on a different data set, as already explained in Section 4.2.6. The results of adaptability of the pre-trained models for cross-domain data set are depicted in Table 9. The confusion matrices for the adapter transformer models on cross-domain data set are shown in Fig. 8. It is evident from Table 9 that mBERT achieved the highest accuracy among all the models. From the results presented above, it appears that the task of detecting abusive language on social media is a difficult one because the content is extremely unstructured and subjective. However, our models have still been able to provide appreciable results. In order to demonstrate the relevance of our research and set it apart from similar studies, we examined the performance of other models

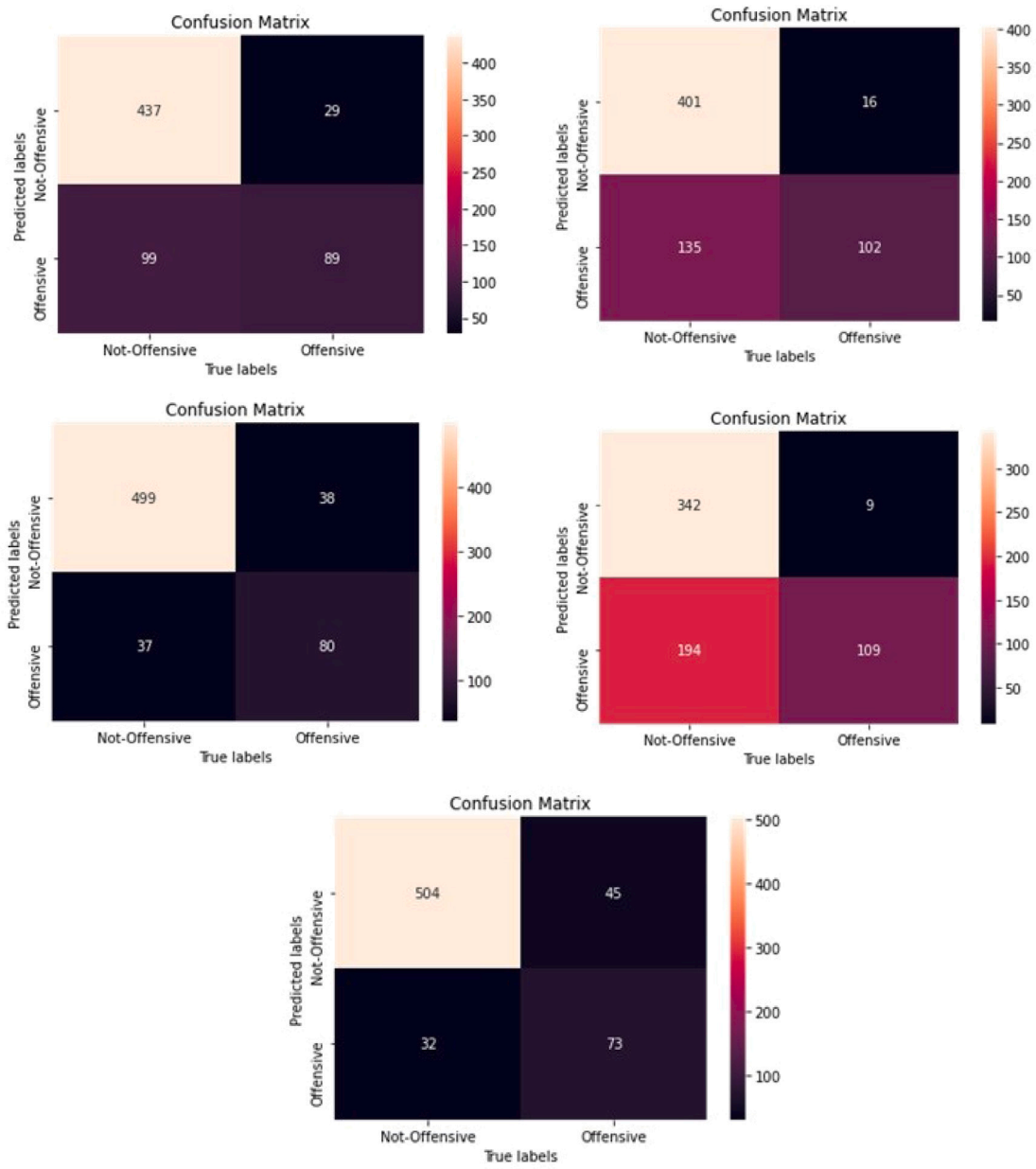

Fig. 6. Confusion matrices for fine-tuned M-BERT, XLMRoberta base, XLMRoberta large, MuRIL - base, and MuRIL - large.

that use the Dravidian Code-Mix FIRE 2021 data set to detect offensive texts. The study by Jada et al. (2021) gave an accuracy of 81.2%, a precision of 80.7%, a recall of 81%, and an accuracy of 80.7% for this data set. But all the proposed adapter-based models gave higher values for all the aforementioned metrics. An attempt by Kumaresan et al. (2021) summarized and ranked the results of all the methodologies submitted for the HASOC-Offensive Language Identification track in Dravidian Code-Mix FIRE 2021. Surprisingly, the weighted average of accuracy, recall, and F1 score for the XLM-RoBERTA (Large) and MuRIL (Large) models was found to be greater than that of the models summarized by Kumaresan et al. (2021). Detecting abusive content is difficult due to a scarcity of benchmark data sets. Researchers gather data sets from several social media platforms and create algorithms to classify these data sets, so there is no uniformity among the data sets.

6.4. Findings and discussion

Various machine learning models have been used to tackle the problem of offensive speech detection, as detailed in the literature review. Based on the research conducted, BNB, SVM, LR, and KNN models were implemented in the present study. Analysis of the

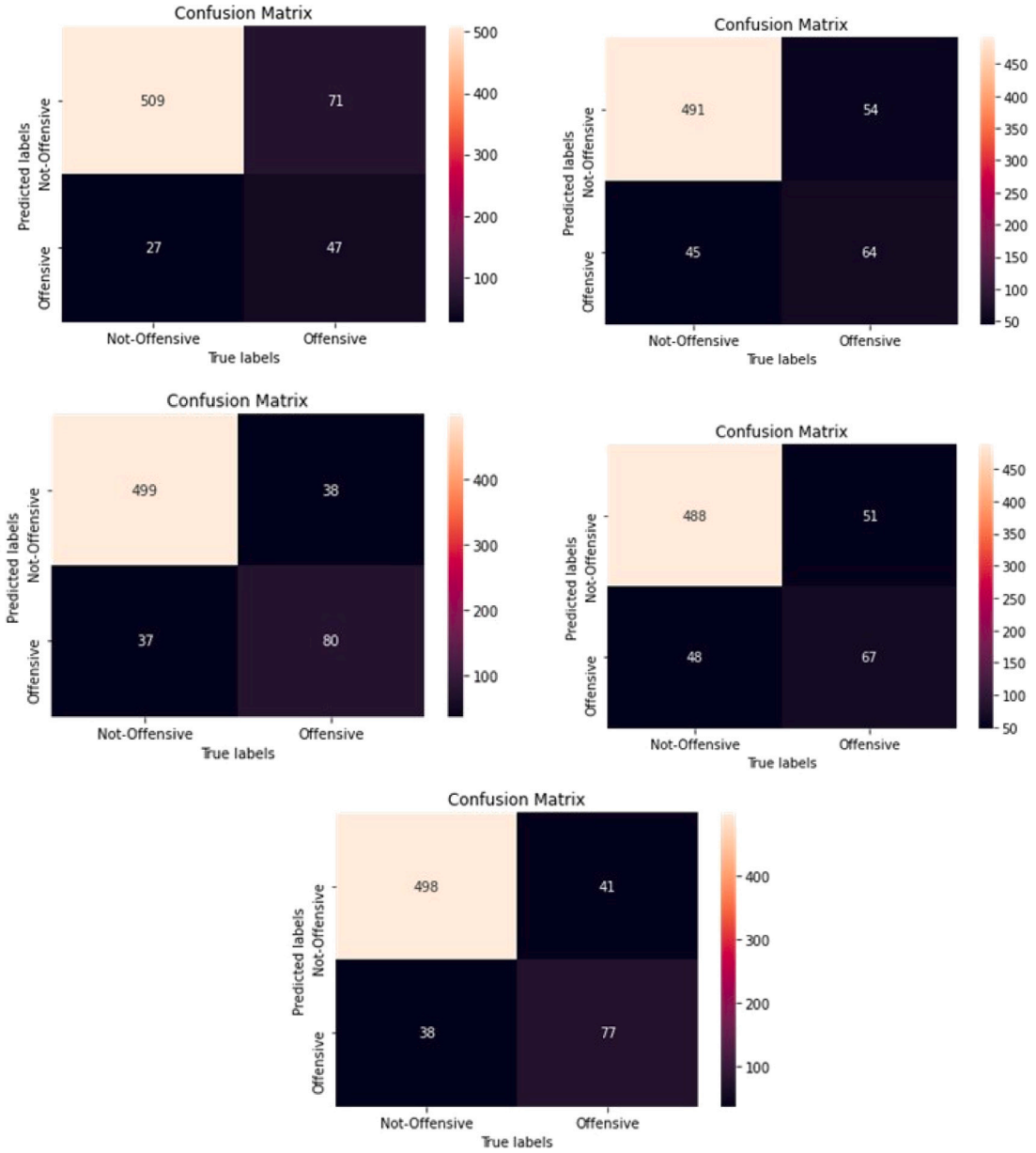

Fig. 7. Confusion matrix for adapter-based M-BERT, XLMRoberta base, XLMRoberta large, MuRIL - base, and MuRIL - large.

confusion matrices illustrates that classifiers have been successful in identifying Not Offensive comments due to the higher number of samples in this category in the given data set. Training accuracy was also calculated to get an idea about the scoring loss between training and testing data sets. This gives information about the overfitting of the models. As shown in Table 4, the training and testing accuracy of the proposed models is very close, which indicates that the models were not over-fit. Of all the models, SVM showed better accuracy for the training data set. Since SVM is designed to be more effective in high-dimensional spaces and works well when there is a clear margin between classes, it produced good results. Another reason for the good results is that the number of features extracted is greater than the number of samples. But with regard to testing data, all the models presented more or less similar accuracy, of which the accuracy of KNN was the highest. Since KNN performed better, it is clear that the test data set is not easily separable using hyper-planes. So, SVM does not perform equally well on test data set.

Since long-range dependencies play a significant role in sequence-to-sequence tasks, transformer models have been introduced in recent times. For offensive language detection in texts, finding the long-range dependencies is important. Hence, a few top-performing transformers models were also attempted in this study. Of all the transformer models proposed, XLM-RoBERTa (Large) showed an accuracy of 88.532%. As this model had been trained on more than 100 languages, including Tamil, it was able to give

Table 8
Transformer models vs machine learning models.

Classifiers	Testing accuracy (%)
Machine learning models	
Bernoulli NB	80.428
SVM	79.663
LR	81.345
KNN	**81.651**
Transformer models	
mBERT	80.428
XLM-RoBERTa - Base	76.911
XLM-RoBERTa - Large	81.957
MuRIL - Base	68.960
MuRIL - large	**88.226**
Adapter-based transformer models	
mBERT	85.015
XLM-RoBERTa - Base	84.862
XLM-RoBERTa - Large	**88.532**
MuRIL - Base	84.862
MuRIL - large	87.920

Table 9
Performance of adapter transformer models for cross-domain data set.

Classifiers	Class labels	Testing accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
mBERT	Not Offensive	81.575	84.896	94.444	89.416
	Offensive		44.966	21.270	28.879
	Macro Average	–	64.931	57.857	59.148
	Weighted Average	–	77.874	81.575	78.769
XLM-RoBERTa - Base	Not Offensive	78.671	81.303	94.681	87.484
	Offensive		49.664	19.423	27.925
	Macro Average	–	65.484	57.052	57.704
	Weighted Average	–	74.573	78.671	74.814
XLM-RoBERTa - Large	Not Offensive	72.864	73.873	95.512	83.310
	Offensive		61.745	17.658	27.463
	Macro Average	–	67.809	56.585	55.387
	Weighted Average	–	70.345	72.864	67.064
MuRIL - Base	Not Offensive	75.321	77.040	95.113	85.128
	Offensive		56.376	18.221	27.541
	Macro Average	–	66.708	56.667	56.334
	Weighted Average	–	71.721	75.321	70.305
MuRIL - Large	Not Offensive	72.920	74.056	95.373	83.373
	Offensive		60.403	17.442	27.068
	Macro Average	–	67.229	56.407	55.220
	Weighted Average	–	70.122	72.920	67.151

better accuracy for the test data set under consideration. Another reason is that this model was specially trained on 12.2 GB of monolingual Tamil in addition to 300.8 GB of English corpus (Conneau et al., 2019). This allowed effective cross-lingual transfer learning by the model. As the data set was in Tamil, the model was found to be effective. The XLM-RoBERTa model combines the advantages of XLM and RoBERTa. It is integrated with an adapter and uses multitask training to improve the performance of prediction. This enables XLM-RoBERTa to achieve good accuracy while classifying offensive texts. Further, an ablation study was performed on whether the performance of the fine-tuned models could be further improved by changing the architecture of these models. For this, we used the concept of adapters and added the adapter layers into the transformer models, as shown in **Figure** .

The innovation of this study is in the strategy that is used to design the adapter module to achieve parameter efficiency without compromising performance. The parameter efficiency of adapter transformer models are presented in Table 7. For instance, mBERT adapter transformer model is compared with a fully fine-tuned mBERT model on the chosen data set. The findings show that the number of task-specific parameters to be trained for fine-tuned models is 42 times that of the adapter-based transformer model. This is similar for other models too and proves the efficiency of the adapter models. A comparison of the accuracy of all the models developed in this study is presented in Table 8. It is evident from this table that adapter-based models perform better than transformer-based models. As an example, the adapter-based mBERT model outperforms the transformer-based one by 5.7%. Out of all the adapter-based models, XLM-Roberta (Base) provides the highest increase in performance (a gain of 23.06%). Owing to their ability to encode task-specific representations in intermediary layers, adapters outperform their pre-trained counterparts. An analysis of the adapter models using the cross-domain data set reveals that their performance has slightly degraded. mBERT's performance is reduced by around 4%, and XLM-Roberta Large recorded the highest reduction in performance of around 17%. This

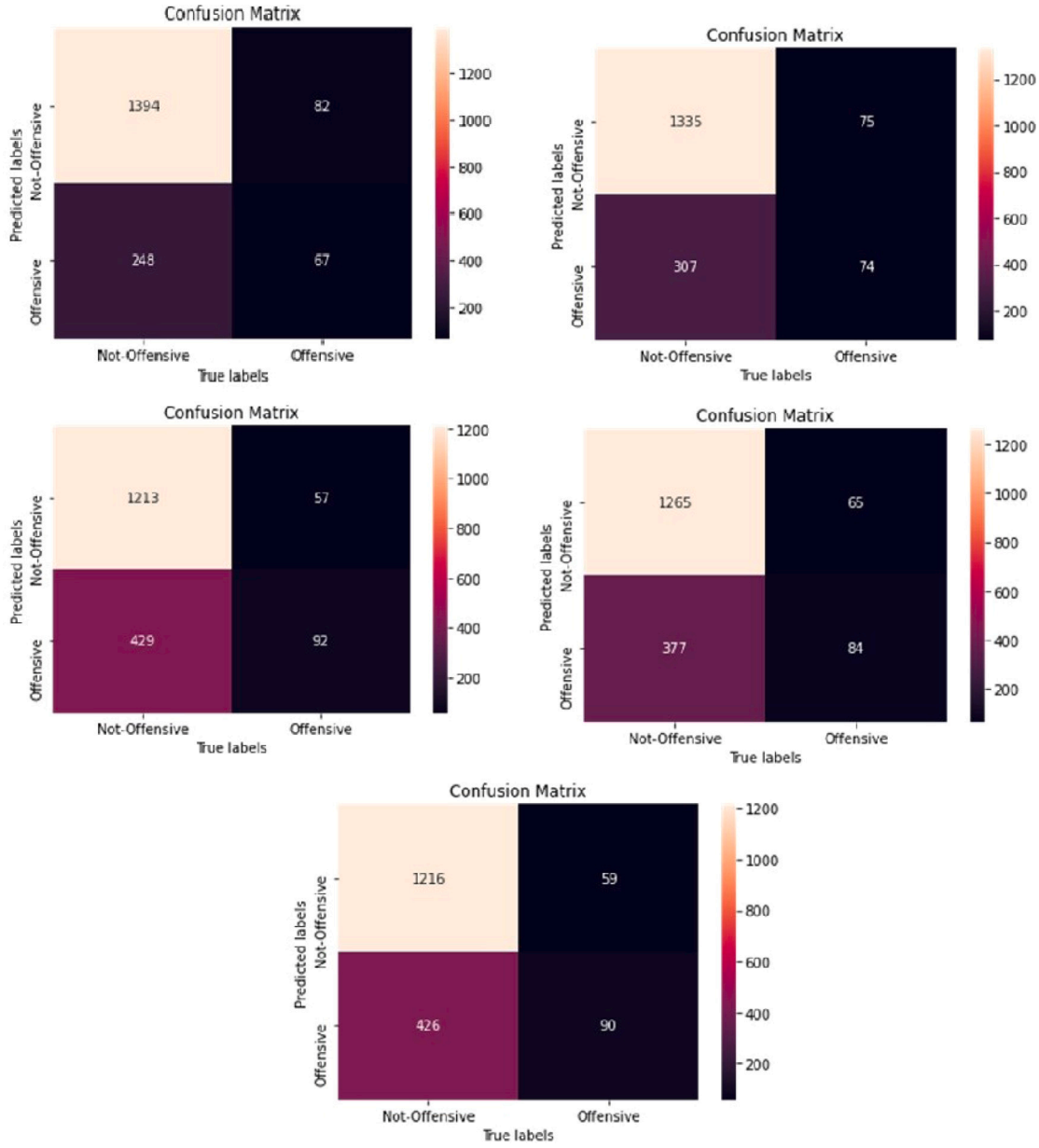

Fig. 8. Confusion Matrices for Adapter based M-BERT, XLMRoberta base, XLMRoberta large, MuRIL - base, MuRIL - large for Cross-domain data set.

demonstrates the need for developing a context-free model of language that can be used across domains and can extract similar features from various domains.

In all the models, values of the performance metrics were found to be low for the Offensive class when compared to the Not Offensive class. This is because the number of samples for the Offensive class was low. Data augmentation techniques such as SMOTE may be applied, but upon trial, it was found that this method could be applied only for numerical features. There are a few text augmentation techniques such as lexical substitution, word-embeddings substitution, Thesaurus-based substitution, etc. which may be attempted in the future. When compared to all the adapter models, the performance of machine learning algorithms is not appreciable. It is because transformer models have great advantages over traditional machine learning methods in text classification. It is not easy for traditional machine learning methods to extract text features. Moreover, these features cannot adequately represent the semantics of the text, and a large part of useful information may be lost. But in transformer models, the features are automatically extracted from the data set. An exhaustive set of experiments have been conducted to investigate various machine learning and adapter transformer models, and the findings show the discriminating capability of the adapter transformer models to deal with the text classification in the offensive language detection field. As we can observe from the results reported in Table 9, the pre-trained

Comment	Actual tag	Predicted Tag
"சிங்கம் 3னு சொன்னாங்க ஆனா ஒரு சிங்கத்தையும் காட்டல"	Offensive	Not Offensive
It was said as Lion3, but not even a single lion was shown.		
"போராடமா அவன் போக மாட்டான் சாக மாட்டான்!!! Pheonix பறவை மாதிரி எரிக்க எரிக்க எழுந்து வருவான்!!!	Not-Offensive	Offensive (mBERT)
He will not go to fight or die !!! Pheonix will wake up to burn like a bird !!!		
"ஆயிரம் ரகுமான் வந்தாலும் ஒரு இளையராஜா ஆக முடியாது ,உயிர் குடுத்து இருக்காரு தன்னேட பின்னணி இசையில்"	Offensive.	Not-Offensive
Even if a thousand Raghumans (musician) come, you will not be able to become Illayaraja, a musician		

Fig. 9. Error analysis – A few cases.

models transfer notably well to any other data set. These results indicate that linguistic offensive speech indicators are well retained across different data sets. This may be due to the presence of common words between the data sets. Cross-domain transfer indicates that the developed models can be generalized across domains regardless of the source and target domains.

7. Error analysis

We conducted an additional study of the errors caused by our models to better grasp the challenges of this undertaking. The practice of evaluating the test set examples that the models misclassified in order to uncover the underlying reasons for the errors is referred to as error analysis. The outcomes of a classification model on new data can be categorized into one of four categories, namely true positives, false positives, true negatives, and false negatives. The terms true and false refer to whether the predicted class matches the actual class in all four cases, and positive and negative refer to the classification the model gives to an observation. For example, in the confusion matrix for the adapter-based mBERT model, we can see that the true positive value is 509. This means that out of 536 non-offensive test occurrences, 509 were correctly classified as non-offensive, while 27 were incorrectly classified as offensive. Similarly, only 47 instances of the Offensive class were accurately identified as offensive, while 71 examples were incorrectly classed as non-offensive. A few examples are provided in Fig. 9. Consider the observation in Fig. 9(a): The actual tag for this comment was Offensive, but this comment was predicted as Not Offensive by all the pre-trained transformer models. An analysis of the comment revealed no offensive or abusive words. It is not clear why this comment was tagged as offensive. This could confuse the classifiers and cause their performance to be degraded. Similarly, another comment listed in Fig. 9(b) was actually classified as Not Offensive, while mBERT predicted it as Offensive. The same comment was present multiple times in the train data and was classified as Not Offensive. Irrespective of its repeated presence and it being tagged as Not Offensive, mBERT predicted it as Offensive. Class imbalance may be one of the reasons for misclassification, and it highly affects the result of classification. Instances in under-sampled classes are more prone to misclassification. Interestingly, the same comment appeared again in the data set, and mBERT correctly predicted it as Not Offensive. We also found that even humans would find it difficult to classify certain tweets as offensive or not without being provided with the full context. The classifier misclassified nearly 5% of the tweets that did not have offensive terms.

We also performed further analysis to understand the cases in which the classifiers misclassified Offensive text as Not Offensive. The comment in Fig. 9(c), which has no offensive terms, was actually tagged as Offensive. Such tagging confused the classifiers, causing their performance to be degraded. Hence, mBERT misclassified 60% of the tweets that were tagged as offensive. On the other hand, the misclassification rate of XLM-RoBERTa (Large) was significantly reduced compared to the other models. This classifier misclassified 32% of tweets as Not Offensive that were actually tagged as Offensive. This rate is low among all the proposed models. XLM-RoBERTa combines the benefits of XLM and RoBERTa; hence, it is integrated with the adapter and takes advantage of the multitask training to increase thee prediction performance. This enables XLM-RoBERTa to classify offensive texts with high accuracy.

8. Conclusion and future work

With the ever-increasing usage of social media, it has become increasingly common for people to hide behind a mask and post offensive comments. Detection of offensive tweets and comments in social media posts present to be a significant task. In the present study, we have described and analyzed various machine learning techniques and neural network models to detect offensive texts in YouTube comments. To start with, machine learning approaches, namely BNB, SVM, LR, and KNN, were used for creating models for detecting offensive contents. Of these models, KNN was found to give the highest accuracy of 81.651%. Since machine learning models require feature extraction to be done explicitly, TF-IDF was used for extracting features from the preprocessed data set. With an attention mechanism, transformers process an input sequence of words all at once and use contextual relations between words in a text for NLP tasks. A few pre-trained transformer- and BERT-based models that do not require explicit feature extraction were also created. These models were used in two modes: fine-tuning, wherein the models were re-trained in entirety, and adapters, wherein small, learned bottleneck layers were inserted within each layer of a pre-trained model, thus avoiding full fine-tuning of the entire model. Such adapter models require a smaller number of parameters to be trained when compared to fine-tuning models. Among the different models, XLM-RoBERTa (Large) gave the highest accuracy of 88.532%. Since this model has more layers, the number of trainable parameters is nearly three times more than that of other adapters. Further, the performance of the adapter-based

transformer models was also tested for a cross-domain data set. For this new data set, these models gave a very appreciable accuracy, which shows the adaptability of the models for a new domain.

According to the results of the proposed models, transformer models, notably pre-trained adapter models, outperform machine learning models. In future, we plan to address the data set imbalance issues using techniques such as text augmentations and oversampling the minority class. We also plan to investigate how adapters can be utilized for further studies, such as migrating an adapter to another downstream task, stacking several adapters, and integrating the information from multiple adapters. Additionally, we intend to explore other transformers models. We believe that a suitable framework has been provided for the NLP community on the usage of adapter model NLP tasks.

CRediT authorship contribution statement

Malliga Subramanian: Conceptualization, Investigation, Methodology, Software, Visualization, Writing – original draft, Writing – review & editing. **Rahul Ponnusamy:** Methodology, Visualization, Software, Writing – original draft, Writing – editing. **Sean Benhur:** Methodology, Software. **Kogilavani Shanmugavadivel:** Investigation, Methodology, Software, Writing – original draft, Writing – review & editing. **Adhithiya Ganesan:** Methodology, Software. **Deepti Ravi:** Methodology, Software. **Gowtham Krishnan Shanmugasundaram:** Methodology, Software. **Ruba Priyadharshini:** Data curation, Writing- reviewing. **Bharathi Raja Chakravarthi:** Conceptualization, Investigation, Methodology, Supervision, Data curation, Writing- origianal draft, reviewing & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

Abro, S., Sarang Shaikh, Z.A., Khan, S., Mujtaba, G., Khand, Z.H., 2020. Automatic hate speech detection using machine learning: A comparative study. Mach. Learn. 10 (6).

Alkiviadou, N., 2019. Hate speech on social media networks: towards a regulatory framework? Inf. Commun. Technol. Law 28 (1), 19–35.

Andrew, J.J., 2021. Judithjeyafreedaandrew @ dravidianlangtech-eacl2021: offensive language detection for dravidian code-mixed youtube comments. In: Proceedings of the First Workshop on Speech and Language Technologies for Dravidian Languages, pp. 169–174.

Anita, R., Subalalitha, C., 2019a. Building discourse parser for Thirukkural. In: Proceedings of the 16th International Conference on Natural Language Processing, pp. 18–25.

Anita, R., Subalalitha, C., 2019b. An approach to cluster Tamil literatures using discourse connectives. In: 2019 IEEE 1st International Conference on Energy, Systems and Information Processing (ICESIP). IEEE, pp. 1–4.

Artetxe, M., Ruder, S., Yogatama, D., 2019. On the cross-lingual transferability of monolingual representations. arXiv preprint arXiv:1910.11856.

Aßenmacher, M., Heumann, C., 2020. On the comparability of pre-trained language models. arXiv preprint arXiv:2001.00781.

Ayo, F.E., Folorunso, O., Ibharalu, F.T., Osinuga, I.A., 2020. Machine learning techniques for hate speech classification of twitter data: State-of-the-art, future challenges and research directions. Comput. Sci. Rev. 38, 100311.

Benhur, S., Sivanraju, K., 2021. Psg@ dravidian-codemix-hasoc2021: Pretrained transformers for offensive language identification in tanglish. arXiv preprint arXiv:2110.02852.

Bharathi, B., Chakravarthi, B.R., Chinnaudayar Navaneethakrishnan, S., Sripriya, N., Pandian, A., Valli, S., 2022. Findings of the shared task on speech recognition for vulnerable individuals in tamil. In: Proceedings of the Second Workshop on Language Technology for Equality, Diversity and Inclusion. Association for Computational Linguistics.

Bharathi, B., et al., 2021. Ssncse_nlp@ dravidianlangtech-eacl2021: offensive language identification on multilingual code mixing text. In: Proceedings of the First Workshop on Speech and Language Technologies for Dravidian Languages, pp. 313–318.

Blair, J., 2003. New breed of bullies torment their peers on the internet. Educ. Week 22 (21), 6.

Chakravarthi, B.R., 2020. HopeEDI: A multilingual hope speech detection dataset for equality, diversity, and inclusion. In: Proceedings of the Third Workshop on Computational Modeling of People's Opinions, Personality, and Emotion's in Social Media. Association for Computational Linguistics, Barcelona, Spain (Online), pp. 41–53, URL https://aclanthology.org/2020.peoples-1.5.

Chakravarthi, B.R., M, A.K., McCrae, J.P., Premjith, B., Soman, K., Mandl, T., 2020a. Overview of the track on HASOC-offensive language identification-DravidianCodeMix. In: FIRE (Working Notes). pp. 112–120.

Chakravarthi, B.R., Muralidaran, V., 2021a. Findings of the shared task on hope speech detection for equality, diversity, and inclusion. In: Proceedings of the First Workshop on Language Technology for Equality, Diversity and Inclusion. Association for Computational Linguistics, Kyiv, pp. 61–72, URL https://aclanthology.org/2021.ltedi-1.8.

Chakravarthi, B.R., Muralidaran, V., Priyadharshini, R., McCrae, J.P., 2020b. Corpus creation for sentiment analysis in code-mixed tamil-english text. In: Proceedings of the 1st Joint Workshop on Spoken Language Technologies for under-Resourced Languages (SLTU) and Collaboration and Computing for under-Resourced Languages (CCURL). European Language Resources association, Marseille, France, ISBN: 979-10-95546-35-1, pp. 202–210, URL https://aclanthology.org/2020.sltu-1.28.

Chakravarthi, B.R., Priyadharshini, R., Durairaj, T., McCrae, J.P., Buitaleer, P., Kumaresan, P.K., Ponnusamy, R., 2022a. Findings of the shared task on homophobia transphobia detection in social media comments. In: Proceedings of the Second Workshop on Language Technology for Equality, Diversity and Inclusion. Association for Computational Linguistics.

Chakravarthi, B.R., Priyadharshini, R., Jose, N., Mandl, T., Kumaresan, P.K., Ponnusamy, R., Hariharan, R., McCrae, J.P., Sherly, E., et al., 2021. Findings of the shared task on offensive language identification in Tamil, Malayalam, and Kannada. In: Proceedings of the First Workshop on Speech and Language Technologies for Dravidian Languages, pp. 133–145.

Chakravarthi, B.R., Priyadharshini, R., Muralidaran, V., Jose, N., Suryawanshi, S., Sherly, E., McCrae, J.P., 2022b. Dravidiancodemix: Sentiment analysis and offensive language identification dataset for dravidian languages in code-mixed text. Lang. Resour. Eval. 1–42.

Cieri, C., Maxwell, M., Strassel, S., Tracey, J., 2016. Selection criteria for low resource language programs. In: Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16), pp. 4543–4549.

Conneau, A., Khandelwal, K., Goyal, N., Chaudhary, V., Wenzek, G., Guzmán, F., Grave, E., Ott, M., Zettlemoyer, L., Stoyanov, V., 2019. Unsupervised cross-lingual representation learning at scale. arXiv preprint arXiv:1911.02116.

Dave, B., Bhat, S., Majumder, P., 2021. Irnlp_daiict @ dravidianlangtech-eacl2021: offensive language identification in Dravidian languages using TF-IDF char n-grams and MuRIL. In: Proceedings of the First Workshop on Speech and Language Technologies for Dravidian Languages, pp. 266–269.

Davidson, T., Warmsley, D., Macy, M., Weber, I., 2017. Automated hate speech detection and the problem of offensive language. arXiv:1703.04009.

De Gibert, O., Perez, N., García-Pablos, A., Cuadros, M., 2018. Hate speech dataset from a white supremacy forum. arXiv preprint arXiv:1809.04444.

De Smedt, T., Jaki, S., Kotzé, E., Saoud, L., Gwóźdź, M., De Pauw, G., Daelemans, W., 2018. Multilingual cross-domain perspectives on online hate speech. arXiv preprint arXiv:1809.03944.

Del Vigna, F., Cimino, A., Dell'Orletta, F., Petrocchi, M., Tesconi, M., 2017. Hate me, hate me not: Hate speech detection on facebook. In: Proceedings of the First Italian Conference on Cybersecurity (ITASEC17), pp. 86–95.

Devlin, J., Chang, M.-W., Lee, K., Toutanova, K., 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.

Dowlagar, S., Mamidi, R., 2021. Hasocone@ fire-hasoc2020: Using BERT and multilingual BERT models for hate speech detection. arXiv preprint arXiv:2101.09007.

Gao, L., Huang, R., 2017. Detecting online hate speech using context aware models. arXiv preprint arXiv:1710.07395.

Gaydhani, A., Doma, V., Kendre, S., Bhagwat, L., 2018. Detecting hate speech and offensive language on twitter using machine learning: An n-gram and tfidf based approach. arXiv preprint arXiv:1809.08651.

Ginting, P.S.B., Irawan, B., Setianingsih, C., 2019. Hate speech detection on twitter using multinomial logistic regression classification method. In: 2019 IEEE International Conference on Internet of Things and Intelligence System (IoTaIS). IEEE, pp. 105–111.

Hande, A., Puranik, K., Yasaswini, K., Priyadharshini, R., Thavareesan, S., Sampath, A., Shanmugavadivel, K., Thenmozhi, D., Chakravarthi, B.R., 2021. Offensive language identification in low-resourced code-mixed dravidian languages using pseudo-labeling. arXiv preprint arXiv:2108.12177.

He, R., Liu, L., Ye, H., Tan, Q., Ding, B., Cheng, L., Low, J.-W., Bing, L., Si, L., 2021. On the effectiveness of adapter-based tuning for pretrained language model adaptation. arXiv preprint arXiv:2106.03164.

Ho, Y.-C., Pepyne, D.L., 2002. Simple explanation of the no-free-lunch theorem and its implications. J. Optim. Theory Appl. 115 (3), 549–570.

Hosmer, D., Lemeshow, S., 2000. Applied Logistic Regression. John Wiley & Sons Inc, New York, NY.

Houlsby, N., Giurgiu, A., Jastrzebski, S., Morrone, B., De Laroussilhe, Q., Gesmundo, A., Attariyan, M., Gelly, S., 2019. Parameter-efficient transfer learning for NLP. In: International Conference on Machine Learning. PMLR, pp. 2790–2799.

Hu, J., Ruder, S., Siddhant, A., Neubig, G., Firat, O., Johnson, M., 2020. Xtreme: A massively multilingual multi-task benchmark for evaluating cross-lingual generalisation. In: International Conference on Machine Learning. PMLR, pp. 4411–4421.

Jada, P.K., Yasaswini, K., Puranik, K., Sampath, A., Thangasamy, S., Thamburaj, K.P., 2021. "Pegasus@ dravidian-codemix-hasoc2021: Analyzing social media content for detection of offensive text". arXiv preprint arXiv:2111.09836.

Joachims, T., 1998. Text categorization with support vector machines: Learning with many relevant features. In: European Conference on Machine Learning. Springer, pp. 137–142.

Kim, S., Shum, A., Susanj, N., Hilgart, J., 2021a. Revisiting pretraining with adapters. In: Proceedings of the 6th Workshop on Representation Learning for NLP (RepL4NLP-2021), pp. 90–99.

Kotsiantis, S.B., Zaharakis, I., Pintelas, P., et al., 2007. Supervised machine learning: A review of classification techniques. Emerg. Artif. Intell. Appl. Comput. Eng. 160 (1), 3–24.

Kovács, G., Alonso, P., Saini, R., 2021. Challenges of hate speech detection in social media. SN Comput. Sci. 2 (2), 1–15.

Kumaresan, P.K., Sakuntharaj, R., Thavareesan, S., Navaneethakrishnan, S., Madasamy, A.K., Chakravarthi, B.R., McCrae, J.P., 2021. Findings of shared task on offensive language identification in tamil and malayalam. In: Forum for Information Retrieval Evaluation. pp. 16–18.

Lample, G., Conneau, A., 2019. Cross-lingual language model pretraining. arXiv preprint arXiv:1901.07291.

Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., Soricut, R., 2019. Albert: A lite bert for self-supervised learning of language representations. arXiv preprint arXiv:1909.11942.

Lee, S.-H., Kim, H.-W., 2015. Why people post benevolent and malicious comments online. Commun. ACM 58 (11), 74–79.

Liu, P., Li, W., Zou, L., 2019a. NULI at SemEval-2019 task 6: Transfer learning for offensive language detection using bidirectional transformers. In: SemEval@ NAACL-HLT. pp. 87–91.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., Stoyanov, V., 2019b. Roberta: A robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692.

MacAvaney, S., Yao, H.-R., Yang, E., Russell, K., Goharian, N., Frieder, O., 2019. Hate speech detection: Challenges and solutions. PLoS One 14 (8), e0221152.

Mahabadi, R.K., Ruder, S., Dehghani, M., Henderson, J., 2021b. Parameter-efficient multi-task fine-tuning for transformers via shared hypernetworks. arXiv preprint arXiv:2106.04489.

Mohapatra, S.K., Prasad, S., Bebarta, D.K., Das, T.K., Srinivasan, K., Hu, Y.-C., 2021. Automatic hate speech detection in english-odia code mixed social media data using machine learning techniques. Appl. Sci. 11 (18), 8575.

Mozafari, M., Farahbakhsh, R., Crespi, N., 2019. A BERT-based transfer learning approach for hate speech detection in online social media. In: International Conference on Complex Networks and their Applications. Springer, pp. 928–940.

Narasimhan, A., Anandan, A., Karky, M., Subalalitha, C., 2018. Porul: Option generation and selection and scoring algorithms for a tamil flash card game. Int. J. Cogn. Lang. Sci. 12 (2), 225–228.

Nayel, H.A., Shashirekha, H., 2019. Deep at HASOC2019: A machine learning framework for hate speech and offensive language detection. In: FIRE (Working Notes). pp. 336–343.

Obadimu, A.M., 2020. Assessing the role of social media platforms in the propagation of toxicity. (Ph.D. thesis). University of Arkansas at Little Rock.

Peters, M.E., Ruder, S., Smith, N.A., 2019. To tune or not to tune? adapting pretrained representations to diverse tasks. arXiv preprint arXiv:1903.05987.

Pfeiffer, J., Kamath, A., Rücklé, A., Cho, K., Gurevych, I., 2020a. Adapterfusion: Non-destructive task composition for transfer learning. arXiv preprint arXiv:2005.00247.

Pfeiffer, J., Rücklé, A., Poth, C., Kamath, A., Vulić, I., Ruder, S., Cho, K., Gurevych, I., 2020b. Adapterhub: A framework for adapting transformers. arXiv preprint arXiv:2007.07779.

Pfeiffer, J., Vulić, I., Gurevych, I., Ruder, S., 2020c. Mad-x: An adapter-based framework for multi-task cross-lingual transfer. arXiv preprint arXiv:2005.00052.

Pires, T., Schlinger, E., Garrette, D., 2019. How multilingual is multilingual bert? arXiv preprint arXiv:1906.01502.

Priyadharshini, R., Chakravarthi, B.R., Chinnaudayar Navaneethakrishnan, S., Durairaj, T., Subramanian, M., Shanmugavadivel, K., U Hegde, S., Kumaresan, P.K., 2022. Findings of the shared task on abusive comment detection in tamil. In: Proceedings of the Second Workshop on Speech and Language Technologies for Dravidian Languages. Association for Computational Linguistics.

Putri, T., Sriadhi, S., Sari, R., Rahmadani, R., Hutahaean, H., 2020. A comparison of classification algorithms for hate speech detection. In: Iop Conference Series: Materials Science and Engineering. Vol. 830, (3), IOP Publishing, 032006.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., Liu, P.J., 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. arXiv preprint arXiv:1910.10683.

Ravikiran, M., Chakravarthi, B.R., Madasamy, A.K., Sivanesan, S., Rajalakshmi, R., Thavareesan, S., Ponnusamy, R., Mahadevan, S., 2022. Findings of the shared task on offensive span identification in code-mixed tamil-english comments. In: Proceedings of the Second Workshop on Speech and Language Technologies for Dravidian Languages. Association for Computational Linguistics.

Razavi, A.H., Inkpen, D., Uritsky, S., Matwin, S., 2010. Offensive language detection using multi-level classification. In: Canadian Conference on Artificial Intelligence. Springer, pp. 16–27.

Rücklé, A., Geigle, G., Glockner, M., Beck, T., Pfeiffer, J., Reimers, N., Gurevych, I., 2020. Adapterdrop: On the efficiency of adapters in transformers. arXiv preprint arXiv:2010.11918.

Sakuntharaj, R., Mahesan, S., 2016. A novel hybrid approach to detect and correct spelling in tamil text. In: 2016 IEEE International Conference on Information and Automation for Sustainability (ICIAfS). pp. 1–6. http://dx.doi.org/10.1109/ICIAFS.2016.7946522.

Sakuntharaj, R., Mahesan, S., 2017. Use of a novel hash-table for speeding-up suggestions for misspelt tamil words. In: 2017 IEEE International Conference on Industrial and Information Systems (ICIIS). pp. 1–5. http://dx.doi.org/10.1109/ICIINFS.2017.8300346.

Sakuntharaj, R., Mahesan, S., 2021. Missing word detection and correction based on context of tamil sentences using N-grams. In: 2021 10th International Conference on Information and Automation for Sustainability (ICIAfS). pp. 42–47. http://dx.doi.org/10.1109/ICIAfS52090.2021.9606025.

Sampath, A., Durairaj, T., Chakravarthi, B.R., Priyadharshini, R., Chinnaudayar Navaneethakrishnan, S., Shanmugavadivel, K., Thavareesan, S., Thangasamy, S., Krishnamurthy, P., Hande, A., Benhur, S., Ponnusamy, K.K., Pandiyan, S., 2022. Findings of the shared task on emotion analysis in tamil. In: Proceedings of the Second Workshop on Speech and Language Technologies for Dravidian Languages. Association for Computational Linguistics.

Sanh, V., Debut, L., Chaumond, J., Wolf, T., 2019. Distilbert, a distilled version of BERT: smaller, faster, cheaper and lighter. arXiv preprint arXiv:1910.01108.

Saroj, A., Mundotiya, R.K., Pal, S., 2019. Irlab@ IITBHU at HASOC 2019: Traditional machine learning for hate speech and offensive content identification. In: FIRE (Working Notes). pp. 308–314.

Sarzynska-Wawer, J., Wawer, A., Pawlak, A., Szymanowska, J., Stefaniak, I., Jarkiewicz, M., Okruszek, L., 2021. Detecting formal thought disorder by deep contextualized word representations. Psychiatry Res. 304, 114135.

Schmidt, A., Wiegand, M., 2019. A survey on hate speech detection using natural language processing. In: Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media, April 3, 2019, Valencia, Spain. Association for Computational Linguistics, pp. 1–10.

Semnani, S., Sadagopan, K.R., Tlili, F., 2019. BERT-A: Finetuning BERT with Adapters and Data Augmentation. Standford University.

Silva, A., Roman, N., 2020. Hate speech detection in portuguese with naïve Bayes, SVM, MLP and logistic regression. In: Anais Do XVII Encontro Nacional de Inteligência Artificial e Computacional. SBC, pp. 1–12.

Singh, G., Kumar, B., Gaur, L., Tyagi, A., 2019. Comparison between multinomial and Bernoulli naïve Bayes for text classification. In: 2019 International Conference on Automation, Computational and Technology Management (ICACTM). IEEE, pp. 593–596.

Srinivasan, R., Subalalitha, C., 2019. Automated named entity recognition from tamil documents. In: 2019 IEEE 1st International Conference on Energy, Systems and Information Processing (ICESIP). IEEE, pp. 1–5.

Subalalitha, C.N., 2019. Information extraction framework for Kurunthogai. Sādhanā (ISSN: 0973-7677) 44 (7), 156. http://dx.doi.org/10.1007/s12046-019-1140-y.

Subalalitha, C., Poovammal, E., 2018. Automatic bilingual dictionary construction for tirukural. Appl. Artif. Intell. 32 (6), 558–567.

Suryawanshi, S., Chakravarthi, B.R., 2021. Findings of the shared task on Troll Meme Classification in Tamil, in: Proceedings of the First Workshop on Speech and Language Technologies for Dravidian Languages, pp. 126–132.

Suryawanshi, S., Chakravarthi, B.R., Arcan, M., Buitelaar, P., 2020. Multimodal meme dataset (multioff) for identifying offensive content in image and text. In: Proceedings of the Second Workshop on Trolling, Aggression and Cyberbullying, pp. 32–41.

Thavareesan, S., Mahesan, S., 2019. Sentiment analysis in tamil texts: A study on machine learning techniques and feature representation. In: 2019 14th Conference on Industrial and Information Systems (ICIIS). pp. 320–325. http://dx.doi.org/10.1109/ICIIS47346.2019.9063341.

Thavareesan, S., Mahesan, S., 2020a. Sentiment lexicon expansion using word2vec and fasttext for sentiment prediction in tamil texts. In: 2020 Moratuwa Engineering Research Conference (MERCon). pp. 272–276. http://dx.doi.org/10.1109/MERCon50084.2020.9185369.

Thavareesan, S., Mahesan, S., 2020b. Word embedding-based part of speech tagging in Tamil texts. In: 2020 IEEE 15th International Conference on Industrial and Information Systems (ICIIS). pp. 478–482. http://dx.doi.org/10.1109/ICIIS51140.2020.9342640.

Thavareesan, S., Mahesan, S., 2021. Sentiment analysis in tamil texts using k-means and k-nearest neighbour. In: 2021 10th International Conference on Information and Automation for Sustainability (ICIAfS). pp. 48–53. http://dx.doi.org/10.1109/ICIAfS52090.2021.9605839.

Tsvetkov, Y., 2017. Opportunities and challenges in working with low-resource languages. Slides Part-1.

Vandersmissen, B., 2012. Automated detection of offensive language behavior on social networking sites. IEEE Trans..

Vasantharajan, C., Thayasivam, U., 2022. Towards offensive language identification for tamil code-mixed YouTube comments and posts. SN Comput. Sci. 3 (1), 1–13.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I., 2017. Attention is all you need. Adv. Neural Inf. Process. Syst. 30.

Xu, L., Xeng, J., Chen, S., 2020. Yasuo at HASOC2020: Fine-tune XML-roberta for hate speech identification. In: FIRE (Working Notes). pp. 311–318.

Zampieri, M., Malmasi, S., Nakov, P., Rosenthal, S., Farra, N., Kumar, R., 2019. Semeval-2019 task 6: Identifying and categorizing offensive language in social media (offenseval). arXiv preprint arXiv:1903.08983.