Hope Speech Detection for Dravidian Languages Using Cross-Lingual Embeddings with Stacked Encoder Architecture

Arunima Sundar¹ · Akshay Ramakrishnan¹ · Avantika Balaji¹ · Thenmozhi Durairaj¹

Received: 14 July 2021 / Accepted: 15 October 2021 / Published online: 18 November 2021
© The Author(s), under exclusive licence to Springer Nature Singapore Pte Ltd 2021

Abstract

The task of hope speech detection has gained traction in the natural language processing field owing to the need for an increase in positive reinforcement online during the COVID-19 pandemic. Hope speech detection focuses on identifying texts among social media comments that could invoke positive emotions in people. Students and working adults alike posit that they experience a lot of work-induced stress further proving that there exists a need for external inspiration which in this current scenario, is mostly found online. In this paper, we propose a multilingual model, with main emphasis on Dravidian languages, to automatically detect hope speech. We have employed a stacked encoder architecture which makes use of language agnostic cross-lingual word embeddings as the dataset consists of code-mixed YouTube comments. Additionally, we have carried out an empirical analysis and tested our architecture against various traditional, transformer, and transfer learning methods. Furthermore a k-fold paired *t* test was conducted which corroborates that our model outperforms the other approaches. Our methodology achieved an F1-score of 0.61 and 0.85 for Tamil and Malayalam, respectively. Our methodology is quite competitive to the state-of-the-art methods. The code for our work can be found in our GitHub repository (<https://github.com/arunimasundar/Hope-Speech-LT-EDI>).

Keywords Hope speech · Code-mixed · Transformers · Language agnostic · Cross-lingual word embeddings · Dravidian languages

Introduction

According to studies [1], the average time a person spends on social media is approximately 2 h and 25 min on a daily basis. It is also a known fact that the majority of teenagers and young adults, roughly 84% [2] of them, utilize social

media regularly. Apart from the positive consequences of the advent of social media, there have been several negative impacts too. In a study conducted by Pew Research Center, it was revealed that around one in five internet users (22%) [3] have been victims of online harassment. Moreover, one of the most common methods by which cyber bullies harass their victims is by commenting on their posts on social media.

Considering that young adults are readily exposed to such vitriolic content on social media, it tends to create a snowball effect. Abusive comments tend to produce more abusive comments and ultimately result in an avalanche of online aggression. This also works the other way around, with positive comments inspiring more people to leave more positive comments [4]. Many studies have been done on this subject including the much disputed Facebook experiment [5] in which the tech-giant modified its “Newsfeed” algorithm to showcase more positive or negative posts to certain users. The results demonstrate that people tend to create more positive posts upon seeing happy posts on their newsfeed, and vice versa.

This article is part of the topical collection “Machine Learning for Offensive and Highly Emotional Content on Social Media” guest edited by Bharathi Raja Asoka Chakravarthi, Anand Kumar M, Sandip Modha, Thomas Mandl and Prasenjit Majumder.

✉ Arunima Sundar
arunima17016@cse.ssn.edu.in

Akshay Ramakrishnan
akshayramakrishnan17007@cse.ssn.edu.in

Avantika Balaji
avantika17021@cse.ssn.edu.in

Thenmozhi Durairaj
theni_d@ssn.edu.in

¹ Sri Sivasubramaniya Nadar College of Engineering, Chennai, India

Platforms such as Facebook and Instagram allow their users to create online communities of like-minded people which promote a sense of acceptance and belonging, especially for those who feel marginalized or lonely. Given the COVID pandemic situation, people in most parts of the world are forced to remain isolated and look to social media to assuage their fears. Therefore, it is important to reinforce the positivity that people seek by focusing on hope speech.

Keeping this in mind, our proposed work focuses on detecting hope speech in YouTube comments. Hope speech consists of texts which have the propensity to inspire and invigorate individuals. Motivational texts and encouragements all fall under the category of hope speech. Our dataset consists of three labels, Hope_speech, Non_hope_speech and Not in language. For example:

- *thalaivar ah mathunga ellam nallabadiya marum:* Change the leader, and everything will change for the better.—**Hope_speech.**
- *Bro neegha sonna eantha app um en kitta ialla:* I don't have any of the applications mentioned by you, brother.—**Non_Hope_Speech.**
- *madhan gowri fans hit like:* Fans of madhan gowri, hit like.—**not-Tamil.**

Our work of hope speech identification mainly focuses on Tamil and Malayalam which are Dravidian Languages predominantly spoken in South India. The official language of the Indian state of Tamil Nadu is Tamil. Moreover, it is also recognized as an official language in two other sovereign nations, Singapore and Sri Lanka. Malayalam is the official language of the Indian state of Kerala and the union territory of Lakshadweep. There are approximately 38 million native Malayalam speakers. As opposed to other Dravidian languages, the finite verb in Malayalam is inflected only for tense and not for person, number or gender. Both the Tamil and Malayalam languages are characterized by a sequence of retroflex consonants which are formed by curling the tip of your tongue to the palate.

Existing work [6–8] pertaining to this task of hope speech detection involves the usage of neural network based architectures as well as transformers. These works perform fairly well for all languages but fail to work well for code-mixed and transliterated text. The proposed approach takes this premise into consideration by utilizing language-agnostic cross-lingual word embeddings which map words from different languages with equivalent meanings to similar hidden representations. This mapping helps to better understand contextual relationships in code-mixed texts resulting in improved performance without much use of feature engineering, whereas other traditional and deep learning models require more data augmentation methods.

Related Work

The first multilingual hope speech dataset [9] for Equality, Diversity and Inclusion (HopeEDI) was created by sourcing comments from YouTube. This formed the basis for the workshop on hope speech detection, conducted by LTEDI-EACL-2021 [10], which brought to light the importance of identifying positive comments on social media. The task proposed in the workshop has promoted research in this field.

In [11], the problem of hope speech detection has approached by adopting character n-grams-based TF-IDF and MuRIL text representations for the sentences in the dataset. Using such text representations, the authors have classified the sentences into hope speech, non-hope speech or not in language. The authors of this paper have compared the different approaches, namely TF-IDF + LR, TF-IDF + SVM, MuRIL + LR and MuRIL + SVM for each language.

Apart from hope speech, we also referred to work related to hate speech and offensive language identification as the crux of these NLP classification tasks is to analyze texts and categorize them. This work [12] follows two machine learning approaches in identifying the offensive language in twitter. The multilingual dataset is preprocessed by using spacy tokenizer for English and German languages followed by the nltk Twitter tokenizer used for tokenizing the input. The approach consisted of an ensemble of SVM, Random forest and Adaboost classifiers with majority voting.

Moreover, the authors have observed that models which make use of MuRIL text representations have been outperformed by models which use TF-IDF representations which led to our choice of disregarding MuRIL for our work. However, even in the case of TF-IDF, it cannot capture semantics as compared to word embeddings and also does not work well in the case of large vocabularies.

These papers formed a basis for our research on traditional learning classifiers with feature engineering like the use of TF-IDF and MuRIL text representations. However, both of these papers use extensive data augmentation in order to train the dataset with traditional classifiers. Hence, we explored literature which use other end classifiers after having obtained semantically rich text embeddings from the dataset.

This work [13] utilizes the FLAIR framework consisting of pre-trained embeddings for all their language models. This approach focuses on using context-aware string embeddings for word representations in deep learning techniques. For text representation, Recurrent Neural Network (RNN) and pooled document embeddings are used. The authors of [14] experiment with Bi-LSTM and Dense architectures along with the incorporation of transformer embeddings like BERT, ULMFiT, MuRIL, etc. For the Tamil and Malayalam

languages, mBERT cased with BiLSTM architecture and mBERT uncased with BiLSTM architecture outperformed the other models, respectively.

BERT embeddings are used in [15] to feed into a CNN classifier. The embeddings have been obtained from the transliterated text where modules like language detection and linguistic rules are used. Convolutions are applied in the classifier network followed by a max-pooling layer. For output representation, the convolutions are passed through to a feed-forward network, and to counteract overfitting, a dropout layer is also added.

Balouchzahi et al. [16] created 3 models, namely CoHope-ML, CoHope-NN, and CoHope-TL, based on an ensemble of classifiers, Keras Neural Network (NN), and BiLSTM with Conv1d model respectively. The CoHope-ML and CoHopeNN models were trained on a feature set consisting of character sequences extracted from sentences combined with words for Ma-En (Malayalam-English) and Ta-En (Tamil-English) code-mixed texts and a combination of word and character n-grams along with syntactic word n-grams for the English dataset. The CoHope-ML model was developed as an ensemble of three classifiers—Logistic Regression, eXtreme Gradient Boosting (XGB) and Multi-Layer Perceptron (MLP) using the bagging technique. The CoHopeNN model employs a Keras dense Neural Network (NN) architecture. The CoHope-TL model comprised three parts: training tokenizers, BERT Language Model training followed using the pre-trained BERT Language Model as weights in BiLSTM-Conv1D model. The CoHope-ML model performed the best amongst the 3 proposed models.

From the above works, it is clear as to how embeddings are essential to grasp the context of the sentence. However, with transformers employing positional embeddings and self attention, it makes it a more attractive choice over using neural networks to accomplish the same. Moreover, given the code-mixed nature of the dataset, we were inspired to experiment with transformers while incorporating multilingual embeddings.

Sharma et al. [17], have contributed to the fields of hope speech detection and offensive language identification. The datasets were preprocessed by applying transformations from native script to Latin script using the Indic-trans library. The ULMFiT model has been trained on synthetically generated code-mixed languages. For the hope speech detection task, KNN algorithm is used to build the baseline model and the final model consists of a classifier trained on the fine-tuned ULMFiT language model. An ensemble of RoBERTa and ULMFiT was submitted for the offensive language identification task. The first classifier is trained on the fine-tuned language model, in this case RoBERTa, and the second classifier is obtained by training the ULMFiT model.

In [18], the authors have processed the dataset by applying stratified K-fold method prior to training and employed

XLM-RoBERTa with attention as their classifier for all three languages. A similar approach [19] also utilized XLM-RoBERTa framework. The labels are classified using the output obtained from the final layer of XLM-RoBERTa and the weighted semantic information of TF-IDF.

This work [20] segregates the process of identifying hope speech into two phases: one phase which consists of five language detection models to identify the language of the text, and another phase which detects the occurrence of hope speech. The authors generate SBERT embeddings and pass them to the feed-forward network for prediction

A transformer-based pretrained BERT model [21], with a rule-based language identification system, assists in the detection of “Other language” labels. The authors of this work also experiment with traditional learning models with TF-IDF and deep learning models with pretrained Glove and FastText embeddings.

Other works that employ transformer models include [22, 23]. In both of these works, the authors have implemented various methods and found that XLM-RoBERTa was the best performing method, achieving promising scores.

Keeping in mind the state-of-the-art performance of transformer-based models, we propose a language-agnostic transformer model capable of providing comparable results on low-resource languages like Tamil and Malayalam. Moreover, to capture the semantics of the different languages, we utilized cross-lingual word embeddings which map similar meanings to the same vector space, regardless of the language.

Data

The dataset [9] used in our methodology was provided by LT-EDI for the purpose of detecting hope speech in YouTube comments. The dataset consists of YouTube comments belonging to three languages, namely English, Tamil and Malayalam as seen in Fig. 1.

These YouTube comments were then segregated into Training, Validation and Test datasets for each language which has been tabulated in Table 1. Each dataset consists of two attributes, “text” and “label”, where “text” is the YouTube comment itself and “label” can be either one of “Hope_speech”, “Non_Hope_speech” and “Not in language” as shown below in Table 2.

In our proposed methodology, we have built two variants—*original* and *split*. In the first variant, the original datasets are used to train the model. In the second variant, the datasets are divided into two for both Dravidian languages, Tamil and Malayalam. The division into the two datasets, is achieved by the language detection module which is based on the langdetect API [24]. It separates the dataset on the basis of whether the text is written in its native

Fig. 1 Language-wise distribution of data **Table 1** Distribution of data

Language	Train	Validation	Test
English	22,762	2843	2846
Tamil	16,160	2018	2020
Malayalam	8564	1070	1071

Table 2 Class-wise distribution of data

Language	Hope_speech	Non_hope_speech	Not in language
English	2484	25,940	27
Tamil	7899	9816	2483
Malayalam	2052	7765	888

script or not. The text containing a majority of pure Tamil/Malayalam words is added in the Pure Tamil/Malayalam dataset and if not, the text is added in the Tanglish/Manglish dataset. We decided to implement the *split* dataset variation to test if our model's performance is impacted by training the pure script and code-mixed content separately.

Additionally, to test the language agnostic nature of our transformer model, we combined the English, Tamil, and Malayalam datasets provided by Language Technology for Equality, Diversity and Inclusion (LTEDI), and used them for training our model. This *combined* dataset was prepared by removing the not in language label. In addition to pre-processing steps mentioned in our proposed methodology, we handle out of vocabulary words (OOV) for the English dataset. OOV words are those that are abbreviated or misspelled, or not present in the English language dictionary. These OOV words can cause misclassifications as they

ഇത് മലയാളത്തിലാണ് 👍 ➡️
 ഇത് മലയാളത്തിലാണ് thumbs up ➡️
 ഇത് മലയാളം രംസ് അപ്പിലാണ്

Fig. 2 Translation of emoji to target language (Malayalam)

increase the randomness in input and do not have embeddings. This problem was handled by employing a custom dictionary that we created with the most frequently occurring OOV words and their corrected usage.

Preprocessing

Basic pre-processing steps performed are listed below:

- removing punctuations, extra spaces
- removing special characters, and lowercasing
- stopword removal
- converting emoticons to text using the demoji [25] library in the English language. For dealing with the Tamil and Malayalam datasets, this text is further translated to Tamil and Malayalam, respectively, based on the script the particular sentence is written in. For example, The first sentence in the above Fig. 2 translates to “This is in Malayalam” in English and transliterates to “ithu malayalathilana”. The emoticon in the first sentence is converted into text using the emoji library. The third sentence translates to “This is in Malayalam thumbs up”, where “thumbs up” from the second sentence has been translated to Malayalam in this third sentence.

- Normalizing contractions—contractions are shortened forms of words joined by an apostrophe. For example, “couldn’t”, “won’t” are contractions. The contractions were normalized for the English words present in all three datasets to standardize the text. For example, contractions like “didn’t” and “wouldn’t” were replaced with “did not” and “would not”.

Pre-processing Trade-offs and Analysis

There were several pre-processing stages that we questioned, taking into consideration our model’s objective and the training dataset. As the datasets consist of YouTube comments, the sentences are concise and in the form of “textspeak” where even stop words may contribute to the meaning of the sentence. As a result, stopwords for the English language were not removed. For the Tamil dataset, the stopwords which are part of the dictionary and written in the Tamil script were removed using the *adverttools* [26] library.

We also considered transliteration, but decided against it because upon doing so, we discerned that this practice does not contribute significantly to the context of sentences in this dataset. This is because the dataset in question comprises of YouTube comments where the majority of them contain code-mixed texts. Most of the comments are such that several common nouns are written as such in the English language itself as depicted in Fig. 3.

Even though the rest of the sentence consists of Tamil words, the word “college” is in the English language. After transliteration, since the word “college” is not in Tanglish (Tamil words written in English script), the library converts each letter in “college” into the corresponding Tamil alphabet. Doing so, would not result in a valid Tamil word. We can see that the actual translation for the word “college” in Tamil is not reflected after transliterating.

Methodology

The proposed methodology uses a transformer-based approach for identifying hope speech in YouTube comments. We have adopted a stacked transformer based encoder model

along with incorporating cross-lingual word embeddings. The methodology consists of two variations. In the first variant, the *original* datasets are used to train the model and the flow of our approach can be seen in Fig. 5. In the second variant, the *split* datasets of the two Dravidian languages are used and this approach is depicted in Fig. 4.

Figure 5 is the first variant proposed methodology where the input given to the model translates to “I am studying everything”.

The steps used in both these approaches are as follows:

1. Cross-lingual word embeddings

We have employed cross-lingual language embeddings that are derived from models pretrained on a combination of both masked language modeling (MLM) and translation language modeling (TLM) [27]. TLM works by concatenating parallel sentences belonging to different languages and tries to predict the masked token, thereby forcing it to take into consideration both languages, resulting in the embeddings of both languages being aligned. The Fig. 6 demonstrates the working of TLM considering two parallel sentences belonging to the Tamil and English languages. The masked Tamil word can be predicted by attending to its English translation and vice versa. The final word embeddings are an aggregation of the language (LE), position (PE), and token (TE) embeddings. The language embeddings are a representation of their respective language codes. The token embeddings consist of the original sentence along with the MASK and separator tokens. The position embeddings (PE) [28] to be encoded are calculated using the equations given below:

$$PE_{(pos,2i)} = \sin(pos/10,000^{2i/d_{model}}), \quad (1)$$

$$PE_{(pos,2i+1)} = \cos(pos/10,000^{2i/d_{model}}). \quad (2)$$

The length of the positional embedding vector is 512 and is the same as that of the word embedding vector. The positional values for the even elements are calculated using Eq. 1, whereas in the case of odd elements, positional values

Fig. 3 Transliteration examples

Fig. 4 Proposed methodology variation-2 for hope speech detection

are calculated using Eq. 2. The final word embeddings are then given as input to the transformer layer.

The transformer layer consists of a bidirectional dual encoder architecture [29] as seen in Fig. 7. The source text, in this case Tamil/Malayalam–English code-mixed content, is given as input to the first encoder. The English translation of the source text is given as input to the second encoder. These inputs are then passed through a group of hidden layers in the encoder. The last layer in both the encoders gives the final sentence embeddings which are s and t , respectively, for the first and second encoder.

According to the bidirectional nature of the encoders, the final embeddings s and t are translations of each other. The objective of this architecture is to rank the true translation of s , over all the sentences in T , containing the set of different sentences i.e. t_i . The following Eq. 3 is a log-linear model which depicts the probability distribution for every t_i given source text s , where $\phi(s_i, t_i)$ gives us the similarity between source and target.

$$P(t_i|s_i) = \frac{e^{\phi(s_i, t_i)}}{\sum_{\bar{t} \in T} e^{\phi(s_i, \bar{t})}}. \quad (3)$$

This probability distribution can be effectively approximated by training in-batch cross-accelerated negative samples as given in equation 4.

$$P_{\text{approx}}(t_i|s_i) = \frac{e^{\phi(s_i, t_i)}}{e^{\phi(s_i, t_i)} + \sum_{k=1, k \neq i}^K e^{\phi(s_i, t_n)}}. \quad (4)$$

The parameter sharing capacity of the two encoders ensures improved mapping of similar words belonging to different languages to similar hidden representations. The CLS tokens present in the last layers of the encoders contain the final embeddings which are combined to obtain the cross-lingual word embeddings.

2. Transformer network

Transformer [30] is an attention mechanism, that learns contextual relations between words. Transformer comprises two mechanisms, an encoder, and a decoder. The encoder used is bidirectional, as opposed to directional models, which reads the text input sequentially. This bidirectional behavior is responsible for allowing the model to learn the context of a word based on all of its surroundings (left and right of the word), which makes it useful for NLP tasks. Our transformer network consists of a stack of encoders that use a self-attention mechanism. An attention mechanism enables the model to highlight essential features by assigning weights to the input features based on their significance. It can be explained in terms of queries, keys, and values. Each word in our sequence can be considered as a query that

Fig. 5 Proposed methodology variation-1 for hope speech detection

Fig. 6 Working of translation language modeling

is mapped to a key-value pair in order of relevance. Therefore in each encoder, we perform the attention mechanism first. The number of queries depends on the number of words present in a sentence. The attention function is computed using the equation given below.

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V. \quad (5)$$

According to the equation, the attentions are computed concurrently by transforming queries, keys, and values into the matrix Q , K , and V respectively. For example, if our sentence

Fig. 7 Architecture of the bidirectional dual-encoder

Fig. 8 Computation of Q , K , V matrices

Fig. 9 Attention heads

contains 5 words, then each matrix will have a dimension of 5×512 vectors. This process is illustrated in Fig. 8.

The type of attention discussed above is of the single attention unit. Another variant of attention known as multi-head attention is used for our approach and it consists of 12 attention heads as seen in Fig. 9.

The different attention heads used imply that the network calculates different Q , K , and V matrices where each head corresponds to a single attention unit or a scalar dot product attention which can be calculated using Eq. 5. The output obtained from the concatenation of scalar dot product attentions undergoes a linear transformation with W^O as depicted in Eq. 6. Here, W is an additional weight matrix obtained during training.

$$\text{MultiHead}(Q, K, V) = \text{concat}(\text{Attention}_1, \dots, \text{Attention}_h)W^O. \quad (6)$$

The output from the attention layer is passed to the position-wise feed-forward network. The embedding vectors from the last encoder are fed into a fully connected layer followed by a softmax layer which computes the probability of each class label. The output corresponding to the CLS token is used.

The word embeddings are passed to the model where the CLS (classification) token is taken as input along with a sequence of words. The layers of the encoders in the stack pass the inputs progressively to the next layers, each of which applies a self-attention mechanism and propagates the result through a feed-forward neural network. This attention mechanism present in each encoder creates context-sensitive representations for each word. The final output of the model is a vector of hidden size (768). To adopt the model for classification tasks, the output corresponding to the CLS token is used.

The proposed methodology for our approach can be summarized by the Algorithm 1 given below.

Table 4 represents the scores obtained from the models trained on the *split* datasets. This table consists of the

Algorithm 1: Proposed Methodology

Input: Code-mixed sentence S

Output: Class label L where $L \in \{\text{Hope, NonHope, Not_in_language}\}$

```

1 Let  $S = w_1, w_2, \dots, w_n$  where  $w_i \in \{\text{ta/ml or ta-en/ml-en}\}$ 
2 Preprocess  $S$  to obtain  $S_p$ 
3 for  $w_i$  in  $S_p$  do
4   Obtain cross-lingual word embedding  $E[w_i]$  where  $\text{sizeof}(E[w_i]) = 512$ 
5    $Q = E[w_i] * W^Q$ 
6    $K = E[w_i] * W^K$ 
7    $V = E[w_i] * W^V$  where  $W^Q, W^K, W^V$  are projection weights learnt during
   training
8   Logically partition  $Q, K, V$  by the number of attention heads  $h$  where  $h = 12$ 
9   for  $i = 0$  to  $12$  do
10     $\text{Attention}_i(Q_i, K_i, V_i) = \text{softmax}(\frac{Q_i K_i^T}{\sqrt{d_{k_i}}}) V_i$ 
11  end
12   $\text{MultiHead}(Q, K, V) = \text{concat}(\text{Attention}_1, \dots, \text{Attention}_h) W^O$  where
    $W^O$  is an output matrix learnt during training
13 end
14 Repeat steps 6-16 for the stack of 12 encoders
15 Obtain hidden layer representation  $H$  from the last encoder layer where
    $\text{sizeof}(H) = 768$ 
16  $H$  is given as input to the classification layer with
    $\text{size} = \text{numberOfClassLabels} = 3$  to obtain output vector  $O$ 
17 Output vector  $O$  is fed into the softmax layer to obtain Probability vector  $P$ 
18 Obtain the class label  $L$  as  $\text{argmax}(P)$ 
19 Return  $L$ 

```

This methodology was implemented using the language-agnostic BERT sentence embedding (LaBSE) [31] framework from the Simple Transformers library. The cross-lingual word embeddings are obtained from LaBSE which is pretrained on 6 billion bilingual sentence pairs. This framework supports 109 languages with a 500 k vocabulary size and provides language-agnostic cross-lingual sentence embeddings for the same. Our language-agnostic transformer model consists of an encoder stack that contains 12 layers with 768 feed-forward networks and 12 attention heads which is used for classification.

Implementation

Following our approach mentioned in the previous section, we trained our model on the Dravidian language datasets for 3 epochs with a batch size of 32. The validation results are discussed below in Table 3. These results pertain to our transformer model which has been trained for *original* datasets.

evaluation of the combined predictions obtained from its two underlying models, one being trained on the dataset containing texts written in the native script and the other trained on code-mixed texts.

The scores achieved by both the variations are similar. This shows that multilingual texts can be given an input to the model without the need for division into different scripts. To test the language agnostic nature of our proposed approach, we trained the transformer model on the *combined* dataset and the results of its performance when tested against the *original* validation datasets of Tamil and Malayalam languages are given in Table 5.

Tables 3, 4, 5 contain weighted average precision, recall and F1 scores for comparison.

Table 3 Original dataset validation results

Language	Precision	Recall	Accuracy	F1-score
Tamil	0.622	0.625	0.625	0.622
Malayalam	0.844	0.848	0.848	0.846

Result Analysis

Empirical Analysis

To evaluate our proposed methodology against other popular text classification approaches in the NLP sphere, we have experimented with the following.

Traditional and Deep Learning Approaches

Apart from the general preprocessing steps followed as mentioned above, the dataset is further prepared by performing various steps to maximize the learning ability of the classifiers. The texts are represented in a vector format with the help of TF-IDF. The TF-IDF scores obtained a maximum of 5000 unique features. For the Malayalam dataset, an additional step was performed to overcome the class imbalance. This was done using a technique called SMOTE (Synthetic Minority Oversampling Technique) [32]. This procedure was not followed for the Tamil language datasets as they were comparatively more balanced than the other language datasets.

Subsequently, the support vector machine (SVM) [33] classifier was built on the oversampled data. Similarly, other traditional classifiers, namely Naive Bayes and logistic regression were also implemented. As a deep learning approach, we have built our model using the bidirectional long short term memory (Bi-LSTM) [34] architecture. The model which we have implemented has 4 layers, namely the embedding layer, Bi-LSTM, and two dense layers. The size of the embedding dimension was 32, and the vocabulary size was 35029. The activation function used in the first dense layer was ‘ReLU’ and the second dense layer had softmax output. The model was trained for 10 epochs with a batch size of 32. The results obtained on the validation set are discussed below in Table 6.

Table 4 Split dataset validation results

Dataset	Precision	Recall	Accuracy	F1-score
Pure Tamil	0.625	0.625	0.625	0.625
Tanglish	0.622	0.622	0.622	0.622
Complete Tamil Dataset	0.622	0.625	0.625	0.623
Pure Malayalam	0.844	0.845	0.845	0.844
Manglish	0.832	0.832	0.832	0.832
Complete Malayalam dataset	0.844	0.844	0.844	0.844

Table 5 Combined dataset validation results

Language	Precision	Recall	Accuracy	F1-score
Tamil	0.654	0.659	0.659	0.656
Malayalam	0.848	0.855	0.855	0.851

Transfer Learning Approach

Transfer learning is a machine learning technique for predictive modeling that involves using a model developed for a task and applying it for a model on a different task. It accelerates the training for the model and lowers the generalization error. We have implemented this technique for the Dravidian languages, i.e., Tamil and Malayalam datasets.

For all of the above-mentioned approaches, we have tokenized the text based on the frequency of occurrence in the given dataset. Now, we look to extend our tokenization methods using the pretrained models offered by the iNLTK [35] library. It contains pretrained language models along with support for other NLP tasks required for Indic languages. More importantly, the provided pre-trained language models also include code-mixed embeddings which would prove imperative when dealing with YouTube comments. In an attempt to utilize these code-mixed language models, we use the Tamil/ Malayalam split datasets, obtained from the process elaborated in Sect. 3.

The classifiers for Tamil/Malayalam and Tanglish/Manglish are built by fine-tuning a pre-trained language model using ULMFiT [36]. ULMFiT is a method provided by fast.ai [37] that enables transfer learning for any NLP task and provides key techniques that are required for fine-tuning language models. This model uses the AWD-LSTM [38] or AGSG weight dropped LSTM architecture for its language modeling. It is a type of LSTM that uses DropConnect and a variant of Average-SGD along with other well-known

Table 6 Validation results for traditional and deep learning models

Language	Model	Precision	Recall	Accuracy	F1-score
Tamil	SVM	0.601	0.608	0.608	0.598
	NB	0.608	0.612	0.612	0.596
	LR	0.587	0.594	0.594	0.586
	Bi-LSTM	0.566	0.566	0.566	0.566
	Proposed model	0.622	0.625	0.625	0.622
Malayalam	SVM	0.794	0.809	0.809	0.793
	NB	0.779	0.792	0.792	0.749
	LR	0.799	0.811	0.811	0.789
	Bi-LSTM	0.791	0.795	0.795	0.792
	Proposed model	0.844	0.848	0.848	0.846

Table 7 Validation results for transfer learning approach

Model	Dataset	Precision	Recall	Accuracy	F1-score
ULMFiT	Pure Tamil	0.615	0.615	0.615	0.615
	Tanglish	0.621	0.621	0.621	0.621
	Complete Tamil dataset	0.621	0.621	0.621	0.621
	Pure Malayalam	0.784	0.798	0.798	0.790
	Manglish	0.812	0.818	0.818	0.815
	Complete Malayalam dataset	0.795	0.812	0.812	0.803
Proposed model	Tamil	0.622	0.625	0.625	0.622
	Malayalam	0.844	0.848	0.848	0.846

regularization strategies. The classifiers for both Tamil and Malayalam have been trained for 5 epochs with dropout and learning rate set to 0.5 and $1e-3$, respectively. The results obtained on the validation set are tabulated in Table 7.

Transformer-Based Approach

Apart from the traditional and deep learning approaches, we also implemented transformer based models. We implemented the multilingual text-to-text transformer architecture (mT5) [39] for training our Dravidian language dataset. We trained this model with a batch size of 4 and evaluated the model by testing it against the validation data. We further experimented with mBERT, training the model by assigning balanced weights which were computed using the scikit-learning library [40] to overcome class imbalance. To extend our experimentation to models pre-trained solely for the task at hand, we used the indic-BERT architecture [41] to train our Tamil and Malayalam datasets. The following Table 8 discusses the validation results.

From the above experiments, we can observe that traditional, transfer learning and transformer-based approaches achieve comparable results with our model, but at the cost of a lot of time and effort being spent in feature engineering and data preparation. Moreover, the above approaches necessitate tailoring the model according to the language of the text in the dataset. Our approach, however, works well in a multilingual context due to the use of cross-lingual word embeddings, which makes the process of model-building simpler. The ability of our language agnostic model to produce satisfactory results while providing cross-lingual support even in the cases where training data is not available (zero-shot cases) [31], makes it an attractive choice when dealing with our dataset consisting of YouTube comments.

Statistical Analysis

To further ground our approach amongst other pre-existing models for text classification, we evaluated the learning ability of our models using k-fold (five-fold) cross validation. Amongst all the models that we implemented, we chose the

best performing models in each approach as elaborated in “Empirical Analysis”.

Tamil Language

Figures 10 and 11 depict the comparison of the performance between our proposed model and SVM, the best performing model under traditional implementation, and indic-BERT, the best performing model under transformer implementation, respectively. Table 9 contains the performance metrics of the various model for five folds.

Malayalam Language

Figures 12 and 13 depict the comparison of the performance between our proposed model and SVM, the best performing model under traditional implementation, and indic-BERT, the best performing model under transformer implementation, respectively. Table 10 contains the performance metrics of the various model for five folds.

T-test

To show that the improvement obtained by our model is statistically significant, we have performed k-fold paired t test. The t test was conducted for one-tailed, type 1 with

Table 8 Validation results for transformer-based models

Language	Model	Precision	Recall	Accuracy	F1-score
Tamil	mT5	0.556	0.445	0.445	0.494
	mBERT	0.512	0.554	0.554	0.532
	indic-BERT	0.543	0.554	0.554	0.548
	Proposed model	0.622	0.625	0.625	0.622
Malayalam	mT5	0.766	0.765	0.774	0.765
	mBERT	0.789	0.778	0.778	0.783
	indic-BERT	0.832	0.839	0.839	0.835
	Proposed model	0.844	0.848	0.848	0.846

Fig. 10 Comparison of our Tamil language approach against SVM

Fig. 11 Comparison of our Tamil language approach against indic-BERT

alpha value 0.5. The performance measures, namely precision, recall and F-score for the 5-folds were considered for computing the t test values.

Table 11 shows the p values obtained from the t test that was computed for our approach and SVM. It is observed that all the values are lesser that 0.05. This shows that our approach improved the performance over SVM and the improvement is statistically significant for both Tamil and Malayalam. Similarly, Table 12 also shows that the improvement in performance of our approach over indic-BERT is statistically significant.

Results and Performance Comparison

Results

Amongst the models tested on the validation data, we noticed that one model for each language performed marginally better than the others. Our observation is as follows: our

Table 9 Cross-validation results for Tamil language

Model	Fold	Precision	Recall	F1-score
SVM	1	0.578	0.577	0.560
	2	0.584	0.584	0.569
	3	0.592	0.597	0.580
	4	0.574	0.580	0.566
	5	0.581	0.584	0.570
indic-BERT	1	0.581	0.589	0.578
	2	0.546	0.561	0.544
	3	0.581	0.587	0.580
	4	0.561	0.573	0.557
	5	0.561	0.570	0.559
Our model	1	0.609	0.611	0.610
	2	0.606	0.609	0.606
	3	0.601	0.602	0.601
	4	0.598	0.599	0.598
	5	0.603	0.605	0.602

Fig. 12 Comparison of our Malayalam language approach against SVM

Fig. 13 Comparison of our Malayalam language approach against indic-BERT

Table 10 Cross-validation results for Malayalam language

Model	Fold	Precision	Recall	F1-score
SVM	1	0.797	0.805	0.778
	2	0.791	0.795	0.765
	3	0.805	0.811	0.782
	4	0.809	0.810	0.780
	5	0.790	0.791	0.758
indic-BERT	1	0.831	0.837	0.833
	2	0.822	0.829	0.824
	3	0.807	0.811	0.808
	4	0.836	0.842	0.837
	5	0.838	0.844	0.839
Our model	1	0.840	0.842	0.840
	2	0.831	0.832	0.831
	3	0.843	0.844	0.843
	4	0.862	0.866	0.863
	5	0.853	0.856	0.854

Table 11 SVM fivefold paired *t* test

Metrics	<i>p</i> value Tamil	<i>p</i> value Malayalam
Precision	0.00186	0.00027
Recall	0.005798	0.00096
F1-score	0.000925	0.00021

Table 12 indic-BERT fivefold paired *t* test

Metrics	<i>p</i> value Tamil	<i>p</i> value Malayalam
Precision	0.002656	0.01128
Recall	0.003446	0.02743
F1-score	0.002101	0.01532

Language Agnostic Transformer model performed the best for both Tamil and Malayalam languages. Table 13 depicts the results obtained by our model on the test sets belonging to the Tamil and Malayalam languages. The tables contain the weighted average precision, recall and F1 scores.

Figures 14 and 15 depict the confusion matrix of Tamil and Malayalam test sets respectively. On observing Fig. 14, we can conclude that out of 2020 instances, our transformer model for the Tamil language has predicted 1234 instances accurately. Moreover for Malayalam, our transformer model classified 907 out of 1071 total instances correctly.

Performance Comparison with State-of-the-Art Results

In this section, the state-of-the-art methods specified in [10] are compared with the performance of our proposed

Table 13 Test set results

Language	Model	Precision	Recall	Accuracy	F1-score
Tamil	Language agnostic trans-former model	0.609	0.611	0.611	0.607
Malayalam	Language agnostic trans-former model	0.848	0.847	0.847	0.847

Fig. 14 Confusion matrix for Tamil test set

methodology. The metrics chosen for comparison include weighted average precision, recall and F1-score. The values are presented in Tables 14 and 15. For the Tamil language, our methodology when compared with the baseline approach has achieved an improvement in the F1-score by 8.92%. Furthermore, an improvement of 16.43% has been observed in the Malayalam language. As seen in Table 14, the ULMFiT methodology [17] has achieved the highest scores and our methodology is at an equal footing with their work. The same is observed for the Malayalam language where our methodology's performance is equivalent to the Ensemble of LR, XGB and MLP model [16].

Conclusion

We have presented a language-agnostic transformer model to detect hope speech in the Dravidian languages datasets. The proposed model makes use of a bidirectional dual-encoder to produce the embeddings for texts in our dataset. By utilizing attention mechanisms, the network better understands

Fig. 15 Confusion matrix for Malayalam test set

Table 14 Performance comparison between methods for the Tamil language

Method	Precision	Recall	F1-score
Baseline-MNB [9]	0.60	0.58	0.56
XLM-RoBERTa [18]	0.59	0.59	0.59
FLAIR + PWE + RNN [13]	0.60	0.59	0.58
mBERT with CNN [15]	0.58	0.58	0.58
TF-IDF character n-grams with LR [11]	0.59	0.59	0.57
Ensemble of LR, XGB and MLP [16]	0.59	0.59	0.59
Tf-Idf scores + XLM-RoBERTa [19]	0.61	0.61	0.59
ULMFiT [17]	0.62	0.62	0.61
Our method	0.61	0.61	0.61

Table 15 Performance comparison between methods for the Malayalam language

Method	Precision	Recall	F1-score
Baseline-DT [9]	0.73	0.76	0.73
TF-IDF character n-grams with SVM [11]	0.78	0.79	0.75
3 parallel Bi-LSTMs [8]	0.79	0.79	0.79
mBERT with CNN [15]	0.83	0.83	0.83
XLM-RoBERTa [18]	0.84	0.84	0.84
Tf-Idf scores + XLM-RoBERTa [19]	0.84	0.85	0.84
FLAIR + PWE + Pooled [13]	0.84	0.85	0.84
Ensemble of LR, XGB and MLP [16]	0.85	0.85	0.85
Our method	0.85	0.85	0.85

contextual relationships between the words. This is evidenced by the comparison of our proposed approach with other different methodologies as seen in the k-fold paired *t* test scores. Moreover, this approach requires less data augmentation and reduces the overhead when compared with other traditional and transformer learning methods. The proposed model achieved an F1-score of 0.85 in Malayalam and 0.61 in Tamil which exceeds the baseline scores for the respective languages. Our study contributes to the reduction of abusive content on social media by filtering out negative comments so that only positive and hopeful comments are retained. In the future, we would like to expand our ideologies by performing an ensemble of transformer-based models using cross-lingual word embeddings.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

1. D. G. How much time do people spend on social media in 2021? 2021. <https://techjury.net/blog/time-spent-on-social-media/#gref>.
2. Dimock M. Demographics of social media users and adoption in the united states | pew research center. 2021. <https://www.pewresearch.org/internet/fact-sheet/social-media/>.
3. Aleksandra. How reading online comments affects us—social media psychology. 2016. <https://socialmediapsychology.eu/2016/10/05/onlineandsocialmediacomments/>.
4. Muchnik L, Aral S, Taylor SJ. Social influence bias: a randomized experiment. *Science*. 2013;341(6146):647–51.
5. Kramer AD, Guillory JE, Hancock JT. Experimental evidence of massive-scale emotional contagion through social networks. *Proc Natl Acad Sci*. 2014;111(24):8788–90.
6. Arunima S, Ramakrishnan A, Balaji A, Thenmozhi D. ssn_diBERTcity@LT-EDI-EACL2021: hope speech detection on multilingual YouTube comments via transformer based approach. In: *Proceedings of the first workshop on language technology for equality, diversity and inclusion*, (Kyiv). Association for Computational Linguistics. 2021. p. 92–7.
7. Upadhyay IS, Wadhawan A, Mamidi R. Hopeful men@LT-EDI-EACL2021: hope speech detection using indic transliteration and transformers. In: *Proceedings of the first workshop on language technology for equality, diversity and inclusion*, (Kyiv). Association for Computational Linguistics. 2021. p. 157–63.
8. Saumya S, Mishra AK. IIIT_DWD@LT-EDI-EACL2021: hope speech detection in YouTube multilingual comments. In: *Proceedings of the first workshop on language technology for equality, diversity and inclusion*, (Kyiv). Association for Computational Linguistics. 2021. p. 107–13.
9. Chakravarthi BR. 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*, (Barcelona, Spain (Online)). Association for Computational Linguistics. 2020. p. 41–53.

10. Chakravarthi BR, Muralidaran V. 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, (Kyiv). Association for Computational Linguistics. 2021. p. 61–72.
11. Dave B, Bhat S, Majumder P. IRNLP_DAICT@LT-EDI-EACL2021: hope speech detection in code mixed text using TF-IDF char n-grams and MuRIL. In: Proceedings of the first workshop on language technology for equality, diversity and inclusion, (Kyiv). Association for Computational Linguistics. 2021. p. 114–17.
12. Mujadia V, Mishra P, Sharma DM. IIIT-Hyderabad at HASOC 2019. Hate Speech Detection. 2019.
13. Junaida MK, Ajees AP. KU_NLP@LT-EDI-EACL2021: a multilingual hope speech detection for equality, diversity, and inclusion using context aware embeddings. In: Proceedings of the first workshop on language technology for equality, diversity and inclusion, (Kyiv). Association for Computational Linguistics. 2021. p. 79–85.
14. Puranik K, Hande A, Priyadarshini R, Thavareesan S, Chakravarthi BR. IIIT@LT-EDI-EACL2021-hope speech detection: there is always hope in transformers. In: Proceedings of the first workshop on language technology for equality, diversity and inclusion, (Kyiv). Association for Computational Linguistics. 2021. p. 98–106.
15. Dowlagar S, Mamidi R. EDIOne@LT-EDI-EACL2021: pre-trained transformers with convolutional neural networks for hope speech detection. In: Proceedings of the first workshop on language technology for equality, diversity and inclusion, (Kyiv). Association for Computational Linguistics. 2021. p. 86–91.
16. Balouchzahi F, Aparna BK, Shashirekha HL. MUCS@LT-EDI-EACL2021: coHope-hope speech detection for equality, diversity, and inclusion in code-mixed texts. In: Proceedings of the first workshop on language technology for equality, diversity and inclusion, (Kyiv). Association for Computational Linguistics. 2021. p. 180–87.
17. Sharma M, Arora G. Spartans@LT-EDI-EACL2021: inclusive speech detection using pretrained language models. In: Proceedings of the first workshop on language technology for equality, diversity and inclusion, (Kyiv). Association for Computational Linguistics. 2021. p. 188–92.
18. Zhao Y, Tao X. ZYJ@LT-EDI-EACL2021: XLM-RoBERTa-based model with attention for hope speech detection. In: Proceedings of the first workshop on language technology for equality, diversity and inclusion, (Kyiv). Association for Computational Linguistics. 2021. p. 118–21.
19. Huang B, Bai Y. TEAM HUB@LT-EDI-EACL2021: hope speech detection based on pre-trained language model. In: Proceedings of the first workshop on language technology for equality, diversity and inclusion, (Kyiv). Association for Computational Linguistics. 2021. p. 122–27.
20. Chinnappa D. dhivya-hope-detection@LT-EDI-EACL2021: multilingual hope speech detection for code-mixed and transliterated texts. In: Proceedings of the first workshop on language technology for equality, diversity and inclusion, (Kyiv). Association for Computational Linguistics. 2021. p. 73–8.
21. Gundapu S, Mamidi R. Autobot@LT-EDI-EACL2021: one world, one family: hope speech detection with BERT transformer model. In: Proceedings of the first workshop on language technology for equality, diversity and inclusion, (Kyiv). Association for Computational Linguistics. 2021. p. 143–48.
22. Ziehe S, Pannach F, Krishnan A. GCDH@LT-EDI-EACL2021: XLM-RoBERTa for hope speech detection in English, Malayalam, and Tamil. In: Proceedings of the first workshop on language technology for equality, diversity and inclusion, (Kyiv). Association for Computational Linguistics. 2021. p. 132–35.
23. Hossain E, Sharif O, Hoque MM. NLP-CUET@LT-EDI-EACL2021: multilingual code-mixed hope speech detection using cross-lingual representation learner. In: Proceedings of the first workshop on language technology for equality, diversity and inclusion, (Kyiv). Association for Computational Linguistics. 2021. p. 168–74.
24. Danilak M. langdetect · pypi. 2021. <https://pypi.org/project/langdetect/>.
25. Solomon B. Demoji-pypi. 2021. <https://pypi.org/project/demoji/>.
26. Dabbas E. Adverttools—pypi. 2021. <https://pypi.org/project/adverttools/>.
27. Lample G, Conneau A. Cross-lingual language model pretraining. CoRR. 2019; [arXiv:1901.07291](https://arxiv.org/abs/1901.07291).
28. Pei CT. Positional embeddings. 2019. <https://medium.com/nlp-trend-and-review-en/positional-embeddings-7b168da36605>.
29. Yang Y, Ábrego GH, Yuan S, Guo M, Shen Q, Cer D, Sung Y, Strophe B, Kurzweil R. Improving multilingual sentence embedding using bi-directional dual encoder with additive margin softmax. CoRR. 2019. [arXiv:1902.08564](https://arxiv.org/abs/1902.08564).
30. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser L, Polosukhin I. Attention is all you need. CoRR. 2017. [arXiv:1706.03762](https://arxiv.org/abs/1706.03762).
31. Feng F, Yang Y, Cer D, Arivazhagan N, Wang W. Language-agnostic BERT sentence embedding. CoRR. 2020. [arXiv:2007.01852](https://arxiv.org/abs/2007.01852).
32. Chawla NV, Bowyer KW, Hall LO, Kegelmeyer WP. SMOTE: synthetic minority over-sampling technique. J Artif Intell Res. 2002;16:321–57.
33. Suthaharan S. Support vector machine. In: Machine learning models and algorithms for big data classification. Springer; 2016. p. 207–35.
34. Huang Z, Xu W, Yu K. Bidirectional LSTM-CRF models for sequence tagging. CoRR. 2015. [arXiv:1508.01991](https://arxiv.org/abs/1508.01991).
35. Arora G. iNLTK: natural language toolkit for indic languages. CoRR. 2020. [arXiv:2009.12534](https://arxiv.org/abs/2009.12534).
36. Howard J, Ruder S. Fine-tuned language models for text classification. CoRR. 2018. [arXiv:1801.06146](https://arxiv.org/abs/1801.06146).
37. Howard J, Gugger S. Fastai: a layered API for deep learning. Information. 2020;11(2):108.
38. Merity S, Keskar NS, Socher R. Regularizing and optimizing lstm language models. 2017. [arXiv:1708.02182](https://arxiv.org/abs/1708.02182).
39. Xue L, Constant N, Roberts A, Kale M, Al-Rfou R, Siddhant A, Barua A, Raffel C. mt5: a massively multilingual pre-trained text-to-text transformer. CoRR. 2020. [arXiv:2010.11934](https://arxiv.org/abs/2010.11934).
40. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J, Passos A, Cournapeau D, Brucher M, Perrot M, Duchesnay E. Scikit-learn: machine learning in Python. J Mach Learn Res. 2011;12:2825–30.
41. Kakwani D, Kunchukuttan A, Golla S, Gokul NC, Bhattacharyya A, Khapra MM, Kumar P. IndicNLP Suite: monolingual corpora, evaluation benchmarks and pre-trained multilingual language models for Indian languages. In: Findings of the association for computational linguistics: EMNLP 2020 (Online). Association for Computational Linguistics. 2020. p. 4948–61.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.