Hopeful_Men@LT-EDI-EACL2021: Hope Speech Detection Using Indic Transliteration and Transformers

Ishan Sanjeev Upadhyay*[1], **Nikhil E***[1], **Anshul Wadhawan**[2], and **Radhika Mamidi**[3]

[1,3]International Institute of Information Technology, Hyderabad
[2]Flipkart Private Limited
[1]{ishan.sanjeev, nikhil.e}@research.iiit.ac.in
[2]anshul.wadhwan@flipkart.com
[3]radhika.mamidi@iiit.ac.in

Abstract

This paper aims to describe the approach we used to detect hope speech in the HopeEDI dataset. We experimented with two approaches. In the first approach, we used contextual embeddings to train classifiers using logistic regression, random forest, SVM, and LSTM based models.The second approach involved using a majority voting ensemble of 11 models which were obtained by fine-tuning pre-trained transformer models (BERT, ALBERT, RoBERTa, IndicBERT) after adding an output layer. We found that the second approach was superior for English, Tamil and Malayalam. Our solution got a weighted F1 score of 0.93, 0.75 and 0.49 for English,Malayalam and Tamil respectively. Our solution ranked first in English, eighth in Malayalam and eleventh in Tamil.

1 Introduction

The spread of hate speech on social media is a problem that still exists today. While there have been attempts made at hate speech detection (Schmidt and Wiegand, 2017; Lee et al., 2018) to stop the spread of negativity, this form of censorship can also be misused to obstruct rights and freedom of speech. Furthermore, hate speech tends to spread faster than non-hate speech (Mathew et al., 2019).While there has been a growing amount of marginalized people looking for support online (Gowen et al., 2012; Wang and Jurgens, 2018), there has been a substantial amount of hate towards them too (Mondal et al., 2017). Therefore, detecting and promoting content that reduces hostility and increases hope is important. Hope speech detection can be seen as a rare positive mining task because hope speech constitutes a low percentage of overall content (Palakodety et al., 2020a). There has been work done on hope speech or help speech detection before that has used logistic regression and active learning techniques (Palakodety et al., 2020a,b). In our paper, we will be doing the hope speech detection task on the HopeEDI dataset (Chakravarthi, 2020; Chakravarthi and Muralidaran, 2021) which consists of user comments from Youtube in English, Tamil and Malayalam.

In this paper, we will first look at the task definition, followed by the methodology used. We will then look at the experiments and results followed by conclusion and future work.

2 Task Definition

The given problem is a comment level classification task for the identification of "hope speech" within YouTube comments, wherein they are to be classified as "Hope speech", "Not hope speech" and "Not in intended language". The data provided in the task was annotated at a per-comment basis wherein a comment could be composed of more than one sentence.

3 Methodology

This section talks about the methodology that we have used to solve the task. As shown in Figure 1, the pipeline involves preprocessing, language detection, transliteration (for Indian languages), and hope speech detection. These steps are described in this section.

3.1 Pre-processing Module

The preprocessing module involved the following:

- Removing special characters and excess whitespaces
- Removing emojis
- Make text lowercase.

*These authors contributed equally to this work.

Proceedings of the First Workshop on Language Technology for Equality, Diversity and Inclusion, pages 157–163

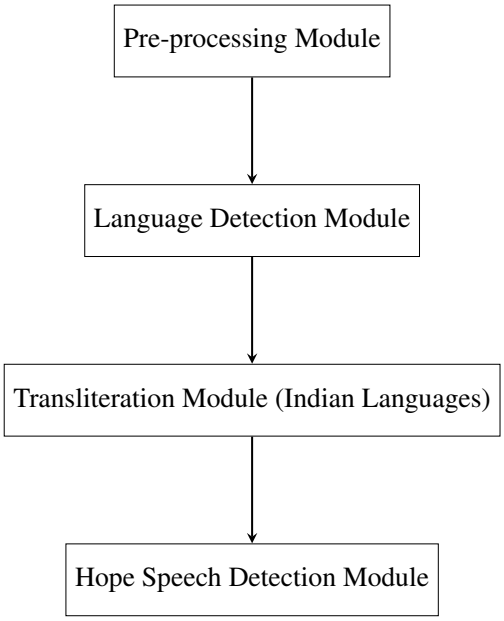

Figure 1: Methodology Pipeline

These steps were taken to make the text more uniform. Special characters like "@" and "#" were removed because they did not serve as good features for classification and language detection. Emojis were removed because they were sparsely used in the dataset.

3.2 Language Detection Module

The task involves classifying text into hope, not-hope and not-language. Language detection module marks the not-language sentences. We use Google's language detection library (Shuyo, 2010) to do this. The Tamil and Malayalam datasets are code-mixed. Inter-sentential,intra-sentential, tag code-mixing and code-mixing between Latin and native script is observed in the Tamil and Malayalam datasets. Google's language detection library does not work on such code-mixed data. Since the Tamil and Malayalam sentences involve code-mixed data, language detection can not be done on them using the Google language detection library. We observed that sentences that were marked as not-Tamil and not-Malayalam were mostly English sentences with some of them being Hindi and other languages. Hence, we adopted a heuristic where we marked sentences as not-Tamil or not-Malayalam if the sentences were detected to be in English or Hindi, other sentences were assumed to belong to the respective language.

3.3 Transliteration Module

After language detection, sentences that are classified to be in Tamil and Malayalam undergo transliteration. Tamil and Malayalam text have code-mixing between Latin and native script, hence transliteration is done to make the entire text in the native script. This step is also important because it makes the text closer to the kind of text IndicBert is trained on. Transliteration was done by using the indic-transliteration library .

3.4 Hope Speech Detection Module

After preprocessing and transliteration (for Indian languages), the text is sent to the hope speech detection module. The hope speech detection module is responsible for predicting if a text is hope speech or not hope speech. We have used the following for our experiment.

3.4.1 Models

Transformers (Vaswani et al., 2017) have performed well in various natural language processing (NLP) tasks. Unlike recurrent neural networks (RNN), transformers are non-sequential (ie. sentences are processed as a whole rather than word by word) and use self-attention at each input time step. Hence, they do not suffer from long dependency issues. Query, Key and Value are three different ways in which input vectors are used in the self-attention mechanism. The attention score for every input vector is calculated using a compatibility function which takes as input the query vector and all the keys. The final output is a weighted sum of values where the weights are the attention scores calculated by the compatibility function. We have used the following transformers in our experiment. We chose RoBERTa for our final model in English and ALBERT (IndicBERT) for Tamil and Malayalam.

BERT (Devlin et al., 2019) is based on the transformer architecture. Using its multi-layer encode module,It is able to jointly utilize both left and right contexts across all layers to pre-train its bidirectional representations. BERT is trained on two unsupervised prediction tasks, next sentence prediction and masked language modelling. We have fine tuned "bert-base-uncased" model on the dataset for one of our experiments.

RoBERTa (Liu et al., 2019) is a transformer architecture which is based on optimizations made to

https://github.com/sanskrit-coders/indic_transliteration

the BERT approach. It trains on more data and bigger batches, removes next sentence prediction objective that BERT used, trains on longer sequences and introduces dynamic masking (ie. mask tokens change during training epochs). RoBERTa outperforms BERT and XLNet on the GLUE benchmark. For the hope speech classification task in English, we fine-tuned the "roberta-base" model on the provided data. The roberta-base model is trained on 160 GB of English text from five different datasets.

ALBERT (Lan et al., 2020) is a transformer architecture based on BERT but with fewer parameters. There are two key changes made to ALBERT. The first is factorized embeddings parameterization, which decomposes the large vocabulary embedding matrix into smaller matrices. This makes it easier to grow hidden size without increasing the parameter size of vocabulary embeddings. This step leads to a reduction in parameters by 80% compared to BERT. The second is cross-layer parameter sharing, which prevents the parameters from increasing as the depth of the network increases. We fine-tuned the IndicBERT model for the hope speech classification task in Tamil and Malayalam. We used IndicBERT (Kakwani et al., 2020) which is a multilingual ALBERT model pre-trained on 12 major Indian languages. We also fine-tuned "albert-base-v2" model for our experiment in English.

LSTM Long Short-Term Memory (Hochreiter and Schmidhuber, 1997) networks seek to solve the short-term memory or vanishing gradient problem that RNNs face. They do so by having internal gates that regulate the flow of information. Information flows through a mechanism known as cell states. The cell can make decisions about what to store, what to forget and what the next hidden state should be. This is done through internal mechanisms called gates which contain sigmoid activations.

Random Forest Classifier Random forests (Breiman, 2001) use an ensemble of a large number of decision trees generally trained with the bagging method. These decision trees are created using random subsamples of the given dataset with replacement (bootstrap dataset) and a random subset of the features. New samples are classified by choosing the prediction made by most decision trees (majority voting).

Support Vector Machine Support vector machine (SVM) (Hearst, 1998) is a supervised learning method that can be used for classification or regression. We have used SVM for classification. The objective of the SVM classification algorithm is to find the hyper-plane that most accurately differentiates two classes that have been plotted on a f dimensional plane where f is the number of features.

Logistic Regression Logistic regression (McCullagh and Nelder, 1989)is a statistical model used for binary classification. It does so by using a logistic function to model the binary outcome. It can be extended for multiclass classification problems.

3.4.2 Ensemble Process

Ensembles can help make better predictions by reducing the spread of predictions. Hence, lowering variance and improving accuracy. We used a voting based ensemble method where we trained N models on N different training and validation data obtained by random shuffling. We then chose the majority voting as the merging technique to produce our final prediction y. In majority voting, the final prediction y is decided based on which prediction is made by the majority of the models . We made two ensembles, one each of 7 models and 11 models and chose the ensemble that gave the best weighted F1 score.

4 Experiments

Initially, the entire database is preprocessed to remove extra tab spaces, punctuations, emojis, mentions and links. In the case of Malayalam and Tamil, we also transliterate the entire database. Then we distributed our experimentation procedure into two different approaches. In the first approach, we fine-tune our pre-trained masked language models using the train and validation splits for the purpose of making them more suitable to the subsequent classification task. Thereafter, contextual embeddings for each sentence in the dataset are produced by calculating the average of the second to last hidden layer for every single token in the sentence. We then trained Logistic Regression, Random Forest, SVM and RNN-based classifier models using these embeddings. In the second approach, all the sentences are encoded into tokens using the respective tokenizers and then we add a linear layer on top of the pre-trained model layers after dropout. All the layers of the devised model are then trained such that the error is back propagated through the entire architecture and the pre-trained weights of the model are modified to reflect the

new database. For both these approaches, we then calculated predictions for the test split and reported performance metrics. For English, we try out three different pre-trained models: "roberta-base", "bert-base-uncased", and "albert-base-v2" for both the approaches. For Tamil and Malayalam however, only the IndicBERT model is applicable for either approach.

Language	Database	Hope	Not Hope	Other Lang.
English	Train	1962	20778	22
	Dev	242	2569	2
Tamil	Train	6327	7872	1961
	Dev	757	998	263
Malayalam	Train	1668	6205	691
	Dev	190	784	96

Table 1: Data distribution by class

	English	Tamil	Malayalam
Training	22762	16160	2564
Development	2843	2018	1070
Test	2846	2020	1071
Total	28451	20198	10705

Table 2: Data distribution by language

4.1 Dataset

The HopeEDI dataset consists of Youtube comments marked as "hope", "not hope" and "other language" in three languages: English, Tamil and Malayalam. The distribution of hope, not hope and other language tag in the training and development datasets is shown in table 1. The ratio of hope to not hope is around 0.09 in English, 0.26 in Malayalam and 0.79 in Tamil. Table 2 shows the data distribution between training, development and test datasets. There are a total of 28,451 comments in English, 10,705 comments in Malayalam and 20,198 comments in Tamil. Data in Tamil and Telugu has code-mixing. In the English dataset, there are instances where English comments are annotated as not English. For example, "Fox News is pure Garbage!" is annotated as not English in the training set. This contributes some noise to the English dataset.

4.2 System Settings

In the first approach, we run the task of masked language modelling on our database for 4 epochs for each of the 5 model-database combinations. Afterwards, the sentence input token length is limited to 512 and the embeddings extracted by evaluation on the input sequences by the model are of length 768. The RNN based classifier is composed of an LSTM layer and two dense layers. In the second approach, the encoded sentences are in the form of a data loader class, containing the respective input IDs and attention masks, with a batch size of 16. These are then passed into a model that implements a dropout of 30% and the output from the final linear layer is used for classification.

4.3 Evaluation Metrics

We used F1 and weighted F1 scores for evaluating our model.

$$F1\ Score = 2 \times \frac{(precision \times recall)}{(precision + recall)}$$

Weighted F1 scores are calculated by taking the F1 scores for each label and then doing a weighted average by the number of true instances of each label.

$$weighted\ F1 = \frac{(F1_i \times y_i + F1_j \times y_j)}{(y_i + y_j)}$$

y_i and y_j are the number of true instances of class i and class j respectively and $F1_i$ and $F1_j$ are the F1 scores of class i and j respectively.

4.4 Results

Our experimentation involved two approaches. In the first approach, we used contextual embeddings (E) to train classifiers using logistic regression (LR), random forest (RF), SVM, and LSTM based models. In the second approach we used an ensemble of 11 models which were generated by fine-tuning (FT) pre-trained transformer models after adding an output layer. We used majority voting to get our final prediction. Results for both the approaches on our test split generated from the provided train and dev datasets are reported in Tables 3, 4 and 5 for English, Tamil and Malayalam respectively. We report the macro-averaged and weighted recall, precision and F1-score for each possible model-method combination. While the weighted F1 scores are more representative of how well a model performs, the disparity between the weighted and macro-averaged scores demonstrates how disproportionate a certain model's effectiveness is in predicting the different classes. For English, the second approach involving finetuning is the best performing one for each of the models tested, closely followed by the Logistic Regression and LSTM-based methods in the first approach. The roberta-base model seems to have a slight edge over the other two tested models. For Tamil and

Model	Method Used	Macro Precision	Weighted Precision	Macro Recall	Weighted Recall	Macro F1-Score	Weighted F1-Score
BERT	E + LR	0.778	0.911	0.656	0.924	0.695	0.913
	E + RF	0.834	0.902	0.526	0.916	0.528	0.881
	E + SVM	0.771	0.86	0.489	0.866	0.488	0.837
	E + LSTM	0.456	0.833	0.500	0.913	0.477	0.871
	FT	0.759	0.915	0.728	0.915	0.742	0.915
ALBERT	E + LR	0.703	0.881	0.538	0.912	0.549	0.883
	E + RF	0.832	0.900	0.506	0.914	0.491	0.874
	E + SVM	0.456	0.833	0.500	0.913	0.477	0.871
	E + LSTM	0.657	0.878	0.571	0.905	0.591	0.887
	FT	0.755	0.916	0.705	0.924	0.725	0.919
RoBERTa	E + LR	0.794	0.914	0.657	0.926	0.700	0.915
	E + RF	0.840	0.905	0.535	0.917	0.544	0.885
	E + SVM	0.821	0.899	0.517	0.915	0.512	0.878
	E + LSTM	0.791	0.918	0.693	0.928	0.729	0.921
	FT	0.753	0.915	0.748	0.922	0.745	**0.923**

Table 3: Metrics for English language

Model	Method Used	Macro Precision	Weighted Precision	Macro Recall	Weighted Recall	Macro F1-Score	Weighted F1-Score
Indic -BERT	E + LR	0.473	0.482	0.484	0.520	0.441	0.464
	E + RF	0.511	0.516	0.506	0.544	0.458	0.482
	E + SVM	0.278	0.309	0.500	0.556	0.357	0.397
	E + LSTM	0.591	0.587	0.501	0.557	0.364	0.403
	FT	0.635	0.637	0.627	0.636	0.623	**0.629**

Table 4: Metrics for Tamil language

Model	Method Used	Macro Precision	Weighted Precision	Macro Recall	Weighted Recall	Macro F1-Score	Weighted F1-Score
Indic -BERT	E + LR	0.645	0.729	0.501	0.790	0.447	0.699
	E + RF	0.395	0.623	0.499	0.788	0.440	0.696
	E + SVM	0.386	0.610	0.492	0.777	0.433	0.683
	E + LSTM	0.367	0.579	0.471	0.745	0.413	0.652
	FT	0.776	0.842	0.743	0.842	0.756	**0.837**

Table 5: Metrics for Malayalam language

Malayalam, the second approach is still the best performer, but by a greater margin.

5 Conclusion and Future Work

In this paper, we presented our approach for hope speech detection in English, Tamil, and Malayalam on the HopeEDI dataset. We used two approaches. The first approach involved using contextual embeddings to train various classifiers. The second approach involved using a majority voting ensemble of 11 models which were obtained by fine-tuning pre-trained transformer models. The second approach using the roberta-base model was the best performing model for English, giving a weighted F1 score of 0.93. The second approach using IndicBERT model gave the best performance for Tamil and Malayalam, giving a weighted F1 score of 0.75 for Malayalam and 0.49 for Tamil. In the future, we plan to fine-tune transformers pre-trained on code mixed data. Data augmentation methods like synonym replacement and random insertion could be used to fine-tune the model on more data.

References

Leo Breiman. 2001. Random forests. *Mach. Learn.*, 45(1):5–32.

Bharathi Raja Chakravarthi. 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*, pages 41–53, Barcelona, Spain (Online). Association for Computational Linguistics.

Bharathi Raja Chakravarthi and Vigneshwaran Muralidaran. 2021. 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.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Kris Gowen, Matthew Deschaine, Darcy Gruttadara, and Dana Markey. 2012. Young adults with mental health conditions and social networking websites: Seeking tools to build community. *Psychiatric Rehabilitation Journal*, 35(3):245–250.

Marti A. Hearst. 1998. Support vector machines. *IEEE Intelligent Systems*, 13(4):18–28.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Divyanshu Kakwani, Anoop Kunchukuttan, Satish Golla, Gokul N.C., Avik Bhattacharyya, Mitesh M. Khapra, and Pratyush Kumar. 2020. IndicNLPSuite: Monolingual corpora, evaluation benchmarks and pre-trained multilingual language models for Indian languages. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4948–4961, Online. Association for Computational Linguistics.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. Albert: A lite bert for self-supervised learning of language representations.

Younghun Lee, Seunghyun Yoon, and Kyomin Jung. 2018. Comparative studies of detecting abusive language on Twitter. In *Proceedings of the 2nd Workshop on Abusive Language Online (ALW2)*, pages 101–106, Brussels, Belgium. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach.

Binny Mathew, Ritam Dutt, Pawan Goyal, and Animesh Mukherjee. 2019. Spread of hate speech in online social media. In *Proceedings of the 10th ACM Conference on Web Science*, WebSci '19, page 173–182, New York, NY, USA. Association for Computing Machinery.

P. McCullagh and J.A. Nelder. 1989. *Generalized Linear Models, Second Edition*. Chapman and Hall/CRC Monographs on Statistics and Applied Probability Series. Chapman & Hall.

Mainack Mondal, Leandro Araújo Silva, and Fabrício Benevenuto. 2017. A measurement study of hate speech in social media. In *Proceedings of the 28th ACM Conference on Hypertext and Social Media*, HT '17, page 85–94, New York, NY, USA. Association for Computing Machinery.

Shriphani Palakodety, Ashiqur R. KhudaBukhsh, and Jaime G. Carbonell. 2020a. Hope speech detection: A computational analysis of the voice of peace.

Shriphani Palakodety, Ashiqur R. KhudaBukhsh, Jaime G. Carbonell, Shriphani Palakodety, Ashiqur R. KhudaBukhsh, and Jaime G. Carbonell. 2020b. Voice for the voiceless: Active sampling to detect comments supporting the rohingyas. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(01):454–462.

Anna Schmidt and Michael Wiegand. 2017. A survey on hate speech detection using natural language processing. In *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media*, pages 1–10, Valencia, Spain. Association for Computational Linguistics.

Nakatani Shuyo. 2010. Language detection library for java.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need.

Zijian Wang and David Jurgens. 2018. It's going to be okay: Measuring access to support in online communities. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 33–45, Brussels, Belgium. Association for Computational Linguistics.