Gauravarora@HASOC-Dravidian-CodeMix-FIRE2020: Pre-training ULMFiT on Synthetically Generated Code-Mixed Data for Hate Speech Detection

Gaurav Arora

Jio Haptik Technologies Limited

Abstract

This paper describes the system submitted to Dravidian-Codemix-HASOC2020: Hate Speech and Offensive Content Identification in Dravidian languages (Tamil-English and Malayalam-English). The task aims to identify offensive language in code-mixed dataset of comments/posts in Dravidian languages collected from social media. We participated in both Sub-task A, which aims to identify offensive content in mixed-script (mixture of Native and Roman script) and Sub-task B, which aims to identify offensive content in Roman script, for Dravidian languages. In order to address these tasks, we proposed pre-training ULMFiT on synthetically generated code-mixed data, generated by modelling code-mixed data generation as a Markov process using Markov chains. Our model achieved 0.88 weighted F1-score for code-mixed Tamil-English language in Sub-task B and got 2nd rank on the leader-board. Additionally, our model achieved 0.91 weighted F1-score (4th Rank) for mixed-script Malayalam-English in Sub-task A and 0.74 weighted F1-score (5th Rank) for code-mixed Malayalam-English language in Sub-task B.

Keywords

Hate speech, offensive language, ULMFiT, markov chains, code-mix data,

1. Introduction

In recent years, as more and more people from all walks of life come online, it has become very important to monitor their behaviour in order to flag behaviour that is hateful, offensive or encourages violence towards a person or group based on something such as race, religion, sex, or sexual orientation, to ensure Internet remains an inclusive place and to promote diversity of content, thoughts and encourage creativity. It is a challenging task to develop systems that can effectively flag hateful and offensive content accurately [1]. Additionally, a substantial amount of work has been done for hate speech detection in languages like English, but no work has been done for Indic languages [2], [3].

Dravidian-Codemix-HASOC2020: Hate Speech and Offensive Content Identification in Dravidian languages (Tamil-English and Malayalam-English) proposes to bridge this gap. It's goal is to identify the offensive language in code-mixed dataset of comments/posts in Dravidian

FIRE 2020 - Forum for Information Retrieval Evaluation, December 16-20, 2020, Hyderabad, India
EMAIL: gaurav@haptik.ai (G. Arora)
URL: https://goru001.github.io/ (G. Arora)

languages collected from social media. Each comment/post is annotated with offensive/not-offensive label at the comment/post level. The data set has been collected from YouTube comments and tweets. It has 2 sub-tasks. Sub-task A is a message-level label classification task in which given a YouTube comment in Code-mixed (Mixture of Native and Roman Script) Malayalam, systems have to classify it into offensive or not-offensive. Sub-task B is also a message-level label classification task in which given a tweet or YouTube comment in Tanglish or Manglish (Tamil and Malayalam written using Roman Script), systems have to classify it into offensive or not-offensive [2], [3].

In this competition, we participated in both sub-task A and sub-task B. We pre-trained ULMFiT [4] on synthetically generated code-mixed data and used transfer learning to train the model on downstream task of hate-speech classification. We propose a Markov model to synthetically generate code-mixed data from Wikipedia articles in native form and their translated and transliterated versions. We use this synthetically generated code-mixed data for Malayalam and Tamil to pre-train ULMFiT from scratch. With this approach, we achieve 2nd Rank on the leader-board for code-mixed Tamil-English language in Sub-task B, 4th Rank on the leader-board for code-mixed Malayalam in mixed-script in Sub-task A and 5th Rank on the leader-board for code-mixed Malayalam-English language in Sub-task B. We've open-sourced our code for dataset preparation, language model pre-training and classification model training on GitHub[1,2].

2. Related Work

Lot of researchers and practitioners from industry and academia have been attracted towards the problem of automatic identification of hateful and offensive speech. Authors in [5] discuss the complexity of the concept of hate speech and its unquestionable potential for societal impact, particularly in online communities and digital media platforms. There have been previous attempts at developing models for hate speech detection in English, Hindi and German [6], Italian [7] but not much work has been done for Dravidian code-mix languages. Having said that, attempts are being made to accelerate progress in NLP in Dravidian languages, like in this competition or in [8], [9].

There have been attempts at synthetically generating code-mixed data based on linguistic theory [10]. But in this paper, we use a novel technique to generate synthetic code-mixed data and pre-train ULMFiT model for Dravidian languages.

3. Methodology

In this section we discuss data preparation for pre-training ULMFiT language model, get insights into the Dravidian code-mix HASOC classification train, dev and test sets and discuss modeling details.

[1]https://github.com/goru001/nlp-for-tanglish
[2]https://github.com/goru001/nlp-for-manglish

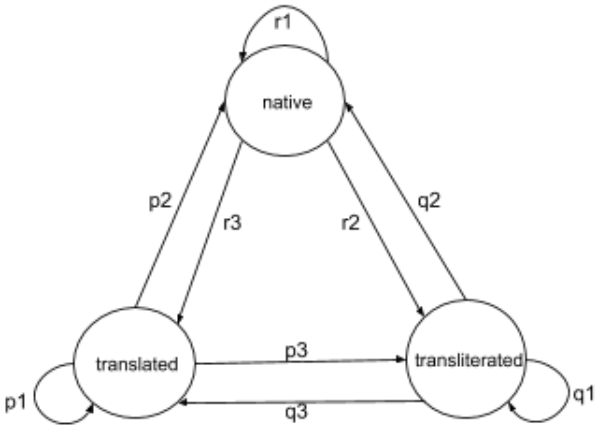

Figure 1: Markov chain modelling code-mixed data generation process

Table 1
State transition probabilities of Markov model used for synthetically generating Malayalam code-mixed data

Model 1						Model 2					
p1	0	q1	0	r1	0	p1	0	q1	0	r1	0
p2	1	q2	0	r2	1	p2	0	q2	1	r2	0
p3	0	q3	1	r3	0	p3	1	q3	0	r3	1

3.1. Dataset

Data preparation for pre-training ULMFiT language model. Firstly, we obtain parallel sets of Wikipedia articles in native, transliterated and translated form. This gives us one to one mapping between sentences in native, transliterated and translated form. This allows us to model code-mixed data generation process as a Markov Process with 3 states; `native, translated` and `transliterated` with `p1, p2, p3, q1, q2, q3, r1, r2, r3` as state transition probabilities as shown in Figure 1. We sample from this Markov process from time `T=0` to time `T=N`, where `N=total no. of sentences in all the Wikipedia articles`, which gives us a output sequence of states $S_{out\,put}$={native, transliterated, transliterated, native, translated, ...} depending upon the values of state transition probabilities. Using $S_{out\,put}$, we form code-mixed data by picking i^{th} sentence at $T = i$ from $S_{out\,put}$'s i^{th} state. Example of state transition probabilities used for synthetically generating Malayalam code-mixed data are shown in Table 1.

Dravidian code-mix HASOC classification dataset. Table 2 shows statistics of the dataset used in this competition for Sub-task A and Sub-task B. Dataset statistics are largely consistent across train, dev and test sets and contain unbalanced classes, varied length sentences. Malayalam and Tamil training datasets for sub-task B contain a few mixed script sentences.

Table 2
HASOC Dravidian code-mix Dataset statistics

	Sub-task A			**Sub-task B**			
	Malayalam mixed-script			**Malayalam**		**Tamil**	
no. of classes	2			2		2	
	Train	**Dev**	**Test**	**Train**	**Test**	**Train**	**Test**
no. of examples	3200	400	400	4000	951	4000	940
%age of examples containing only Roman characters	66.8%	70.2%	64.5%	99.1%	100%	99.7%	100%
min. no. of examples in a class	567	72	-	1952	-	1980	-
max. no. of examples in a class	2633	328	-	2047	-	2020	-
avg. no. of examples in a class	1600	200	-	2000	-	2000	-
min no. of tokens in an example	1	2	1	1	1	1	1
max no. of tokens in an example	186	56	186	62	526	66	55
avg no. of tokens in an example	9.009	9.44	9.87	9.99	10.47	18.33	17.31
median no. of tokens in an example	8	8	8	8	9	16	15

Table 3
Dropout applied to AWD-LSTM model

Layer	Embedding Layer	Input Layer	LSTM's Internal	Between LSTMs
Dropout	0.02	0.1	0.2	0.2

3.2. Modeling Details

3.2.1. Model Architecture

Weight-dropped AWD-LSTM variant [11] of the long short-term memory network [12] and [13] was used for the language modeling task. Its embedding size was 400, number of hidden activations per layer was 1152 and the number of layers was 3. Two linear blocks with batch normalization and dropout were added as custom head for the classifier with rectified linear unit activations for the intermediate layer and a softmax activation at the last layer [4]. Table 3 shows dropout applied to AWD-LSTM model. We use *Dropout Multiplicity* as a parameter in our experiments to proportionately change the magnitude of all dropouts.

3.2.2. Pipeline

Tokenization. We create subword vocabulary by training a SentencePiece[3] tokenization model on the dataset prepared, using unigram segmentation algorithm [14]. Since we're using subword

[3]https://github.com/google/sentencepiece

Table 4
Vocab size of the model

Language	Tamil	Malayalam	
Script	Latin	Latin	Mixed
Vocab size	8000	15000	25000

Table 5
Validation set perplexity of ULMFiT, trained on synthetically generated code-mixed data

Language	Tamil	Malayalam	
Script	Latin	Latin	Mixed
Perplexity	37.50	45.84	41.22

tokenization, our models can handle different spellings in transliteration or spelling errors in test set. Table 4 shows vocabulary size of the models trained for each one of the sub-tasks.

ULMFiT Language model pre-training. Our model is based on the Fastai[4] implementation of ULMFiT. Table 5 shows perplexity of the trained models on validation set. Pre-trained models along with datasets can be downloaded and used for several other downstream classification tasks from the GitHub repository.

Preprocessing of Dravidian code-mix HASOC dataset. Basic preprocessing was done by lower-casing, removing @username mentions etc. Exact reproduction details are available in the GitHub repository.

Classifier training. Classifier was trained by first fine-tuning the pre-trained ULMFiT language model and then training the classifier.

4. Experiment and Results

4.1. Experiment setting

We did basic pre-processing detailed in the previous section. For each one of the sub-tasks, we split the training data into standard train-validation splits in the ratio of 80:20, in case explicit validation set was not given. Using this split, we perform experiments to identify the best set of hyperparameters, shown in Table 6. Gradual unfreezing epochs and learning rates are accessible from notebooks in GitHub repository. Backpropagation through time (BPTT) was set to 70. For language model fine-tuning, dropout multiplicity was set to 0.3, batch size was set to 64, the model was trained for 1 epoch with learning rate 1e-2 with only last layer unfreezed and 5 epochs with learning rate 1e-3 after unfreezing all layers.

4.2. Results

In this competition, teams were ranked by the weighted F1 score of their classification system. Tables 7, 8 and 9 show Top-5 ranked teams in the competition with their weighted Precision, weighted Recall and weighted F1-score for each one of the sub-tasks. Difference between top

[4]https://github.com/fastai/fastai

Table 6
Hyperparameters for Classifier training

Task		Dropout Multiplicity	Batch Size	Model Unfreezed	
				Epochs	Learning Rate
Sub-Task A	Malayalam Mixed Script	0.5	16	5	1e-3
Sub-Task B	Malayalam	0.7	16	5	1e-3
	Tamil	0.5	16	5	1e-3

Table 7
Top-5 of the official leader-board in Sub-task A for code-mixed Malayalam

TeamName	Precision	Recall	F-Score	Rank
SivaSai@BITS, IIITG-ADBU	0.95	0.95	0.95	1
CFILT_IITBOMBAY, SSNCSE-NLP	0.94	0.94	0.94	2
CENMates, NIT-AI-NLP, YUN, Zyy1510	0.93	0.93	0.93	3
Gauravarora	**0.92**	**0.91**	**0.91**	**4**
WLV-RIT	0.89	0.9	0.89	5

Table 8
Top-5 of the official leader-board in Sub-task B for code-mixed Malayalam

TeamName	Precision	Recall	F-Score	Rank
CENmates	0.78	0.78	0.78	1
SivaSai, KBCNMUJAL	0.79	0.75	0.77	2
IIITG-ABDU	0.77	0.76	0.76	3
SSNCSE-NLP	0.78	0.74	0.75	4
Gauravarora	**0.76**	**0.72**	**0.74**	**5**

Table 9
Top-5 of the official leader-board in Sub-task B for code-mixed Tamil

TeamName	Precision	Recall	F-Score	Rank
SivaSaiBITS	0.9	0.9	0.9	1
Gauravarora	**0.88**	**0.88**	**0.88**	**2**
SSNCSE-NLP	0.88	0.88	0.88	2
KBCNMUJAL, IIITG-ADBU, zyy1510	0.87	0.87	0.87	3
CENmates, CFILT	0.86	0.86	0.86	4
YUN	0.85	0.85	0.85	5

ranked team's weighted F1-score and our weighted F1-score is less than 0.04 in all the sub-tasks. Perplexity of ULMFiT language model after fine-tuning was 57.53 for mixed-script Malayalam in sub-task A, 71.02 for code-mixed Tamil and 89.12 for code-mixed Malayalam in Latin script in sub-task B.

5. Conclusion

In this paper, we presented a Markov model to facilitate synthetic generation of code-mixed data and used it to pre-train ULMFiT language model on Dravidian languages. Using transfer learning with the pre-trained ULMFiT model on downstream task of hate speech detection, we achieved Rank 2 for Tamil in Sub-task B, Rank 4 for Malayalam in Sub-task A and Rank 5 for Malayalam in Sub-task B. In future research, we will consider improving synthetically generating code-mixed data by allowing code-mixing within a single sentence and using Transformer based models.

References

[1] R. Kumar, A. K. Ojha, S. Malmasi, M. Zampieri, Benchmarking aggression identification in social media, in: Proceedings of the First Workshop on Trolling, Aggression and Cyberbullying (TRAC-2018), Association for Computational Linguistics, Santa Fe, New Mexico, USA, 2018, pp. 1–11. URL: https://www.aclweb.org/anthology/W18-4401.

[2] B. R. Chakravarthi, M. A. Kumar, J. P. McCrae, P. B, S. KP, T. Mandl, Overview of the track on HASOC - Offensive Language Identification - DravidianCodemix, in: Working Notes of the Forum for Information Retrieval Evaluation (FIRE 2020). CEUR Workshop Proceedings. In: CEUR-WS. org, Hyderabad, India, 2020.

[3] B. R. Chakravarthi, M. A. Kumar, J. P. McCrae, P. B, S. KP, T. Mandl, Overview of the track on HASOC - Offensive Language Identification - DravidianCodemix, in: Proceedings of the 12th Forum for Information Retrieval Evaluation, FIRE '20, 2020.

[4] J. Howard, S. Ruder, Fine-tuned language models for text classification, CoRR abs/1801.06146 (2018). URL: http://arxiv.org/abs/1801.06146. arXiv:1801.06146.

[5] P. Fortuna, S. Nunes, A survey on automatic detection of hate speech in text, ACM Computing Surveys (CSUR) 51 (2018) 1 – 30.

[6] T. Mandl, S. Modha, P. Majumder, D. Patel, M. Dave, C. Mandlia, A. Patel, Overview of the hasoc track at fire 2019: Hate speech and offensive content identification in indo-european languages, in: Proceedings of the 11th Forum for Information Retrieval Evaluation, FIRE '19, Association for Computing Machinery, New York, NY, USA, 2019, p. 14–17. URL: https://doi.org/10.1145/3368567.3368584. doi:10.1145/3368567.3368584.

[7] M. Corazza, S. Menini, E. Cabrio, S. Tonelli, S. Villata, A multilingual evaluation for online hate speech detection, ACM Trans. Internet Technol. 20 (2020). URL: https://doi.org/10.1145/3377323. doi:10.1145/3377323.

[8] B. R. Chakravarthi, R. Priyadharshini, V. Muralidaran, S. Suryawanshi, N. Jose, E. Sherly, J. P. McCrae, Overview of the track on Sentiment Analysis for Dravidian Languages in Code-Mixed Text, in: Working Notes of the Forum for Information Retrieval Evaluation (FIRE 2020). CEUR Workshop Proceedings. In: CEUR-WS. org, Hyderabad, India, 2020.

[9] B. R. Chakravarthi, R. Priyadharshini, V. Muralidaran, S. Suryawanshi, N. Jose, E. Sherly, J. P. McCrae, Overview of the track on Sentiment Analysis for Dravidian Languages in Code-Mixed Text, in: Proceedings of the 12th Forum for Information Retrieval Evaluation, FIRE '20, 2020.

[10] A. Pratapa, G. Bhat, M. Choudhury, S. Sitaram, S. Dandapat, K. Bali, Language modeling for code-mixing: The role of linguistic theory based synthetic data, 2018, pp. 1543–1553. doi:10.18653/v1/P18-1143.

[11] S. Merity, N. S. Keskar, R. Socher, Regularizing and optimizing LSTM language models, CoRR abs/1708.02182 (2017). URL: http://arxiv.org/abs/1708.02182. arXiv:1708.02182.

[12] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 9 (1997) 1735–1780. URL: https://doi.org/10.1162/neco.1997.9.8.1735. doi:10.1162/neco.1997.9.8.1735.

[13] F. A. Gers, J. A. Schmidhuber, F. A. Cummins, Learning to forget: Continual prediction with lstm, Neural Comput. 12 (2000) 2451–2471. URL: https://doi.org/10.1162/089976600300015015. doi:10.1162/089976600300015015.

[14] T. Kudo, J. Richardson, Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing, CoRR abs/1808.06226 (2018). URL: http://arxiv.org/abs/1808.06226. arXiv:1808.06226.