Meta-Learning for Offensive Language Detection in Code-Mixed Texts

Gautham Vadakkekara Suresh
National University of Ireland Galway
Galway, Ireland
gautham.suresh09@yahoo.com

Bharathi Raja Chakravarthi
Insight SFI Research Centre for Data Analytics,
National University of Ireland Galway
Galway, Ireland
bharathi.raja@insight-centre.org

John P. McCrae
Insight SFI Research Centre for Data Analytics,
National University of Ireland Galway
Galway, Ireland
john.mccrae@insight-centre.org

ABSTRACT

This research investigates the application of Model-Agnostic Meta-Learning (MAML) and ProtoMAML to identify offensive code-mixed text content on social media in Tamil-English and Malayalam-English code-mixed texts. We follow a two-step strategy: The XLM-RoBERTa (XLM-R) model is trained using the meta-learning algorithms on a variety of tasks having code-mixed data, monolingual data in the same language as the target language and related tasks in other languages. The model is then fine-tuned on target tasks to identify offensive language in Malayalam-English and Tamil-English code-mixed texts. Our results show that meta-learning improves the performance of models significantly in low-resource (few-shot learning) tasks[1]. We also introduce a weighted data sampling approach which helps the model converge better in the meta-training phase compared to traditional methods.

CCS CONCEPTS

• **Information systems** → **Clustering and classification**; • **Computing methodologies** → *Machine learning algorithms.*

KEYWORDS

Deep Learning; Code-mixing; Meta Learning; Dravidian Languages; Malayalam; Tamil

ACM Reference Format:
Gautham Vadakkekara Suresh, Bharathi Raja Chakravarthi, and John P. McCrae. 2021. Meta-Learning for Offensive Language Detection in Code-Mixed Texts. In *Forum for Information Retrieval Evaluation (FIRE 2021), December 13–17, 2021, Virtual Conference , India.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3503162.3503167

1 INTRODUCTION

The advent of the internet in developing countries has resulted in increased usage of social media. Social media platforms such as Facebook, Twitter, YouTube and Instagram provide an inexpensive and open platform for people to share, communicate and discuss various topics [20]. The open nature of these platforms has given rise to hate speech content in online social media resulting in inflamed discussions between users, racism and extremism.

Though English is the principal language used for communication on the internet, the usage of other languages is common in developing countries and multilingual communities. Hence, there is a need to develop technologies for these languages as well [12, 25]. In addition to this, the users of multilingual communities often mix the linguistic units of a language with another in an utterance [24]. This is commonly referred to as code-mixing (CM). CM can occur in different ways. It is called inter-sentential or code-switching when it occurs in the same utterance but different sentences. It can also happen in a single sentence and in such cases, it is called intra-sentential or code-mixing. We use the general term code-mixing to mean both types. It can also happen at tag-level when a tag phrase (short phrase) in one language is mixed with an utterance from another language. This type of code-mixing usually occurs at an intra-sentential level [9]. Sometimes, intra-sentential code-mixing can have the switching take place in a single word making it intra-word switching [5]. Along with CM, social media users sometimes mix sentences in native script with words or phrases from another language, such as English. This makes code-mixed content complex for natural language processing (NLP) tasks such that even a simple task such as word-level language identification is considered difficult [12]. Deep learning models for NLP are usually trained on large amount of text data obtained from sources such as Wikipedia, CommonCrawl and BookCorpus [39]. But, CM is mostly used for informal communication and rarely used in literature and media such as news which results in a low amount of available training data. This scarcity of data along with the complexity added by CM makes it difficult to develop systems that can handle code-mixed data. The goal of this research is to use meta-learning to improve the ability of models offensive language detection task for Malayalam-English and Tamil-English code-mixed texts.

The Dravidian languages form a distinct family of languages that are spoken by more than 300 million people [16, 31]. Both Tamil and Malayalam are Dravidian languages. The Tamil language has official status in the Indian state of Tamil Nadu, Indian union territory Puducherry, Sri Lanka and Singapore [7, 28, 29]. The Tamil language is written alphasyllabically and has rich morphology [30, 34]. The Malayalam language is spoken in the southern region of India, mainly in the state of Kerala, the union territories Lakshadweep and Puducherry [9]. Malayalam and Tamil are morphologically very rich and exhibits high agglutination [21, 32, 33]. Both Tamil and Malayalam are under-resourced languages. The users of these

[1]https://github.com/gauthamsuresh09/meta_cm_offensive_detection

FIRE 2021, December 13-17, 2021, Virtual Event, India

ACM ISBN 978-1-4503-9596-0/21/12…$15.00
https://doi.org/10.1145/3503162.3503167

languages often use Latin script for typing instead of native script as well as mix with words in English.

We propose a meta-learning approach to identify offensive language in code-mixed texts. Though our approach is generic, we evaluate our methods on datasets containing Tamil-English and Malayalam-English code-mixed texts. We train the model using meta-learning on a variety of datasets related to the target task and then fine-tune it on the publicly available dataset for target task. These datasets could be from same language as target language and also from other languages but for a similar task. We also propose a modification to the sampling strategy during meta-training phase by assigning higher weights to datasets that are more related to the target task. We evaluate the models on publicly available datasets and report our results on the same.

2 RELATED WORK

A few years back, there wasn't much research in the area of offensive language and hate speech detection for Malayalam-English and Tamil-English code-mixed texts[10]. This can be mainly attributed to the lack of a standard annotated corpus to train and benchmark the systems[3]. The shared tasks for offensive language identification conducted as part of Dravidian CodeMix - HASOC 2020 [18] and Dravidian Lang Tech 2021 workshop [8] introduced annotated datasets in these languages. In Dravidian CodeMix - HASOC 2020 shared task, Sai and Sharma [27] proposed a system that does selective translation and transliteration approach to normalise the input. They identified the English words in the input text using a vocabulary and translated it to the target language, which is either Malayalam or Tamil in this case. The remaining words in the Latin script were transliterated to the native script. An ensemble of XLM-R models is used to handle the instability and variance observed while evaluating them separately. Pathak et al. [23] used a combination of word-level and character-level n-gram with Term Frequency - Inverse Document Frequency (TF-IDF) for feature extraction. The extracted features were then used to train a Multinomial Naive Bayes (MNB) model. Their method was simple and effective considering the amount of training data available.

Shared tasks for offensive language detection for Malayalam, Tamil and Kannada languages were organized by the Dravidian Lang Tech 2021 workshop [4]. The work from Saha et al. [26] pre-trained the XLM-R model on the target dataset to capture the semantics of code-mixed corpus. Later, this pre-trained model is fine-tuned on the target dataset for the classification task. Similar to findings by Sai and Sharma [27], they also noticed variation in model performance for different random seeds and used a genetic algorithm to set the weights for different models in the ensemble. Kedia and Nandy [15] found that pre-training transformer models using a combined dataset of all three languages gives improved performance. This suggests that learning from one language could help with tasks for other related languages.

The Model-Agnostic Meta-Learning (MAML) proposed by Finn et al. [14] trains the parameters of Deep Neural Networks (DNN) on a variety of related tasks such that fine-tuning on a new task for a small number of gradient steps gives a significant performance improvement. The authors also mentioned the usage of first-order approximation for backpropagation in the gradient descent step during the meta-learning step. The original MAML algorithm needs a second-order derivative for the parameter updates which is computationally expensive. They were able to achieve significant speed-up in the network by using first-order approximation without losing much of the performance gains. Later, Nichol et al. [22] introduced the Reptile algorithm which samples the tasks repeatedly for training during the meta-learning step, moving the model parameters towards the values preferred by the task for better initialization. Dou et al. [13] studied the application of meta-learning algorithms on low-resource natural language understanding tasks. Triantafillou et al. [35] introduced the ProtoMAML algorithm which combines the usage of prototypes in Prototypical Networks with MAML algorithm for better initialization of final linear layer. Later, van der Heijden et al. [36] proposed ProtoMAMLn, a simple modification to the ProtoMAML algorithm by applying L_2 normalisation to the prototypes for better and stable learning over multilingual classification tasks.

The current systems require large number of annotated samples during training to handle the complexity of code-mixed texts. Our objective is to adapt the model to handle the code-mixed text and produce better results on the target task of offensive language detection. Meta-learning appears to be a promising strategy in this case to adapt the model to target task domain to provide better performance in low-resource scenario compared to multi-task learning approaches Bansal et al. [2]. We can utilize datasets for related tasks or languages to train the model with algorithm such as MAML and its variants. This could improve the performance of systems on tasks such as offensive language detection in low-resource code-mixed texts by applying meta-learning on datasets for similar tasks such as sentiment-analaysis and hate speech detection. This approach also allows us to train the model using both code-mixed datasets as well as monolingual datasets.

3 DATASETS

The datasets for this research can be split mainly into two categories: a) Meta-training datasets and b) Meta-testing (Offensive language detection datasets). This section covers their details and the preprocessing steps applied to them.

3.1 Meta-Training

The datasets are selected in such a way that they are either for related tasks such as hate speech detection, sentiment analysis or contain data in the same or related language when compared to target tasks. This includes publicly available Malyalam-English and Tamil-English code-mixed datasets for sentiment analysis. Secondly, monolingual datasets for Malayalam and Tamil classification tasks are also used. Apart from these, datasets for similar tasks from other languages are also used for cross-lingual transfer. The details of these datasets including the languages and classes selected from them are provided in Table 1. Each language in a dataset is annotated with a number which will be used as an identifier in the next sections.

3.2 Meta-Testing

Once the model is trained using meta-learning, it needs to be tested on the target tasks. For this, the following offensive language detection datasets provided as part of Tamil and Malayalam sub-track in HASOC 2020 [18] are selected as target tasks: a) Task 1 containing Malayalam-English code-mixed texts collected from YouTube comments, b) Task 2 Tamil-English containing Tamil-English code-mixed texts collected from Twitter tweets and Helo App comments, and c) Task 2 Malayalam-English containing Malayalam-English code-mixed texts collected from YouTube comments. All the three tasks are utterance level binary classification tasks. The datasets contain all types of code-mixing. The details of train, validation and test splits of these datasets are given in Table 2.

3.3 Pre-processing

The datasets selected for meta-learning and fine-tuning contain mostly short texts collected from social media. The pre-processing step is important for these datasets as most of them contain noise such as URLs and repeated characters. To decide the pre-processing steps, the XLM-RoBERTa (XLM-R) [11] "base" model is fine-tuned on Task 2 Tamil-English code-mixed data after applying different combination of pre-processing techniques. Based on the results from this step, the following pre-processing steps are selected for the meta-learning datasets: remove URLs, remove hashtags (eg. #tamilmovie) and user mentions (eg. @user1), remove emoji from the text, replace repeated whitespace and full stop characters with a single occurrence, convert text to lowercase and shorten the elongated words where the same character repeats consecutively more than 3 or more times. Some of the pre-processing steps mentioned above differ for the fine-tuning phase on offensive language detection datasets: Instead of removing hashtags (eg. #tamilmovie) and user mentions (eg. @user1), only the "#" or "@" is removed. For example, "#tamilmovie" becomes "tamilmovie" after applying this step. Also, all the punctuation characters are removed.

4 METHODOLOGY

The implementation strategy can be broadly divided into meta-training and meta-testing steps. The meta-training phase also involves meta-validation in which the model is evaluated for fast adaptation on the validation datasets after each epoch. In the meta-training procedure, our objective is to optimize the model f_θ parameterized by θ for fast adaptation on validation tasks. For this, we sample a batch of tasks from a distribution $p(D)$ over set of datasets D. Each task in the batch is split into support (S) and query (Q) sets. The model is trained and evaluated for fast adaption on the support and query sets, respectively. To achieve this, the model is trained for k steps on support samples for a task and then evaluated on corresponding query set. Finally, the model parameters are updated using the loss on query set over the batch of tasks. Our baseline is the XLM-R model with only meta testing phase applied to evaluate it on the target tasks. This helps us to know the improvement that meta-learning provides for these tasks by comparing the metrics of the baseline model with the ones that are trained using procedure described below.

4.1 Meta-Training

We experiment with two meta-learning algorithms in the meta-training phase: MAML and ProtoMAML. These algorithms are selected based on their performance on multilingual document classification tasks compared to other algorithms such as Reptile [36]. We experiment with two different approaches using these algorithms. In the approach description, we refer to the datasets from Section 3 with identifiers defined in Table 1.

Algorithm 1 - MAML: The Model-Agnostic Meta-Learning (MAML) algorithm introduced by Finn et al. [14] has shown promising results over the years on various tasks. The algorithm is model-agnostic in the sense that it can be easily applied to any DNN architecture which uses gradient-based optimization or any other differentiable objective for training. During the meta-learning phase, the model parameters are updated to generalize better on the domain of tasks. In effect, it does weight initialisation for the model that adapts easily on downstream tasks. After successful meta-learning, the model will be sensitive to target domain tasks and even a few gradient update steps will help it easily adapt to the given task. This is particularly helpful for approaches such as K shot learning where we have only a limited number of samples for a given task. In such cases, we can adapt the model to the target domain by training it on similar tasks using MAML. This helps the model to use knowledge from the tasks in the meta-learning phase for fast adaptation to the target task. We use an implementation[2]with first-order approximation of MAML. In the upcoming sections, this is referred as foMAML.

Algorithm 2 - ProtoMAML: ProtoMAML combines the simple yet effective inductive bias of Prototypical Networks with fast adaptation of MAML algorithm. van der Heijden et al. [36] found promising results by applying ProtoMAML on benchmark multilingual document classification tasks. They made a simple modification to the ProtoMAML algorithm by adding L_2 normalization to the prototypes, referred to as ProtoMAMLn. Similar to MAML, a first-order approximation[2] of the algorithm is implemented for the experiments which is referred to as foProtoMAMLn.

Approach 1 - Cross-lingual: In this approach (Algorithm 1 from van der Heijden et al. [36]), hate speech and offensive language detection datasets from various languages are used for meta-learning. This includes Datasets 5, 6, 7, 8, 9, 10. During the meta-learning phase, these datasets could adapt the model to the target task which can later help for cross-lingual transfer. Along with these datasets, the sentiment analysis Datasets 3 and 4 are also used to help the model adapt to code-mixed data.

Approach 2 - Language-oriented: Approach 2 (Algorithm 2) implements a language-oriented method for the selection of datasets in which we select the datasets with same or similar language as the target tasks. For example, in the case of target tasks in Malayalam-English code-mixing, same language refers to Malayalam and similar language refers to Tamil. This mapping results in two different models, one for Malayalam-English and another for Tamil-English code-mixed texts. According to this criteria, the following datasets are selected: 1, 2, 3, 4, 5, 11, 12, 13 and 14 . The only additional dataset included is Hindi hate speech detection

[2]https://github.com/mrvoh/meta_learning_multilingual_doc_classification

Table 1: Details of datasets selected for meta-learning.

Meta-learning Datasets			
Dataset	Type	Languages selected	Classes selected
Dravidian-CodeMix, FIRE 2021 [5, 6]	Sentiment analysis	Malayalam-English code-mixed (1), Tamil-English code-mixed (2)	Positive, Negative, Mixed feelings
Dravidian-CodeMix, FIRE 2020 [5, 6]	Sentiment analysis	Malayalam-English code-mixed (3), Tamil-English code-mixed (4)	Positive, Negative, Mixed feelings
HASOC, FIRE 2019 [19]	Hate speech detection	Hindi (5), English (6), German (7)	None, Hate, Profane, Offensive
OffensEval 2020 [38]	Offensive language detection	Arabic (8), Greek (9), Turkish (10)	None, Offensive
iNLTK headlines corpus [1]	News headlines classification	Malayalam (11), Tamil (12)	Business, Sports, Entertainment, Tamil-cinema and Spirituality
IndicNLP classification dataset [17]	News article classification	Malayalam (13), Tamil (14)	Business, Sports, Entertainment, Technology, Politics

Table 2: Number of samples in datasets for offensive language detection tasks (Meta-testing). The dataset splits which are not present are marked with N/A. In this case, each sample is an utterance.

Dataset Statistics						
Split	Task 1		Task 2 Tamil-English		Task 2 Malayalam-English	
	NOT	OFF	NOT	OFF	NOT	OFF
Train	2633	567	2019	1979	2047	1953
Val	328	72	N/A	N/A	N/A	N/A
Test	334	66	465	475	473	478

Algorithm 1 Meta-training procedure for Approach 1

Require: $p(D)$: Distribution over tasks
Require: α, β: step size hyper-parameters
  Initialize θ
  while not done **do**
    Sample batch of tasks $\{D^l\} = \{(S^l, Q^l)\}\ \ p(D)$
    for all (S^l, Q^l) **do**
      Initialize $\theta_l^{(0)} = \theta$
      for all steps k **do**
        Compute: $\theta_l^{(k+1)} = \theta_l$ - $\alpha(\nabla_{\theta_l^{(k)}} \mathcal{L}_{S_l}(f_{\theta_l^{(k)}}))$
      end for
    end for
    Update $\theta = \theta$ - β(MetaUpdate$(f_{\theta_l^{(K)}}, Q^l)$)
  end while

dataset. This approach also implements a weighted sampling strategy during the meta-training phase to upsample the datasets for target language. This assigns a higher weight of 4 to datasets having the same code-mixing as the target language, a lower weight of 2 to datasets of same language but contain monolingual data and a lowest weight value of 1 to datasets from other languages if present. These weights are normalized later so that their sum is 1.

In the meta-training step of both approaches, all the classes from these datasets are used during training, though only two classes from a selected dataset are randomly sampled in each iteration. This is because the target tasks are binary classification tasks. The meta-validation step takes samples from datasets with the same code-mixing as the target task rather than all the datasets utilized in training. The test and validation splits of the datasets are merged to increase the number of samples. The two different models for this approach are as explained below:

- **Malayalam-English code-mixing:** As per the weight configuration, datasets 1 and 3 get weight 4, datasets 11 and 13 get 2, and datasets 2, 4, 5, 12 and 14 get 1. For the meta-validation step, only Malayalam-English sentiment analysis code-mixed datasets 1 and 3 are used.
- **Tamil-English code-mixing:** Using the weight configuration, the datasets 2 and 4 get a weight of 4, datasets 12 and 14 get 2, and datasets 1, 3, 5, 11 and 13 get 1. For the meta-validation step, the Tamil-English sentiment analysis code-mixed datasets 2 and 4 are used.

In the meta-validation phase, we select only the samples having output classes "Positive" and "Negative" from these datasets. These classes are mapped to constant classifier outputs contrary to the shuffling approach implemented by van der Heijden et al. [36]. The hypothesis is that it helps the model to learn faster during the validation step. It should be noted that we haven't added any specific modifications in the approach or dataset pre-processing to handle mixing of scripts. The models are trained with such data during the meta-training phase if it exists in the selected code-mixed datasets.

4.2 Meta-Testing

The meta-testing phase involves fine-tuning and evaluating our model on the target tasks. The models are fine-tuned on the target datasets for a pre-defined number of epochs with early stopping.

Algorithm 2 Meta-training procedure for Approach 2

Require: f_θ: Base-learner
Require: α, β: step size hyper-parameters
Require: D: Set of datasets
  if target task is in Malayalam-English code-mixing **then**
    $U \leftarrow$ ml-en
    $V \leftarrow$ ml
  else if target task is in Tamil-English code-mixing **then**
    $U \leftarrow$ ta-en
    $V \leftarrow$ ta
  end if
  for all d in D **do**
    if $d.language == U$ **then**
      $d.weight \leftarrow 4$
    else if $d.language == V$ **then**
      $d.weight \leftarrow 2$
    else
      $d.weight \leftarrow 1$
    end if
  end for
  $W \leftarrow$ Normalised weights for tasks in D
  Initialize θ
  $p(D) \leftarrow$ distribution over tasks using probability values from W

  while not done **do**
    Sample batch of tasks $\{D^l\} = \{(S^l, Q^l)\}\ \ p(D)$
    for all (S^l, Q^l) **do**
      Initialize $\theta_l^{(0)} = \theta$
      for all steps k **do**
        Compute: $\theta_l^{(k+1)} = \theta_l$ - $\alpha(\nabla_{\theta_l^{(k)}} \mathcal{L}_{S_l}(f_{\theta_l^{(k)}}))$
      end for
    end for
    Update $\theta = \theta$ - β(MetaUpdate($f_{\theta_l^{(K)}}, Q^l$))
  end while

The fine-tuning approach presented in this study varies from the one implemented in van der Heijden et al. [36], where the model is trained on the datasets using meta-learning algorithm. Instead, the input data is split into batches and the model is fine-tuned using the Adam optimizer with weight decay and a linear scheduler with warmup. This fine-tuning approach is implemented using the Hugging Face Transformers library [37]. Due to this difference in training procedure, the parameters of layer normalization cannot be used here as the meta-learning strategy uses per step layer normalization instead of shared parameters. This method maintains sets of parameters for the layer normalization, each for a step in the inner loop of meta-learning. Instead, these parameters are initialised with values from the XLM-R "base" model. The models are fine-tuned separately on each of the offensive language detection task both at high-resource and low-resource settings. We also perform Welch's t-test between the scores of baseline and meta-trained models to report the statistical significance of our results.

Low-resource setting: The low-resource setting involves fine-tuning the models with a very low number of samples from each of the target classes. This requires undersampling the train split of task-specific datasets to the required count. To achieve this, two different undersampling approaches are implemented. The first approach undersamples at target class level to a required count ensuring the resulting dataset is balanced. This approach can be applied to datasets that are highly imbalanced. The second approach performs a stratified splitting based on the target class values for the samples. This approach is implemented for datasets that are balanced or nearly balanced by providing a required proportion to which undersampling needs to be done. The details of datasets after undersampling is mentioned in given in Table 3. For each of the three tasks to be tested, the dataset is undersampled using 5 different random seed values. The model is then fine-tuned and evaluated on each of these datasets separately.

Table 3: Number of samples (utterances) for the tasks in low-resource setting. The validation split is created for Task 2 by splitting the original training data into 85% and 15% for training and validation, respectively. The resulting training data is undersampled to required amount for low-resource setting.

Dataset Statistics for low-resource setting						
Split	Task 1		Task 2 Tamil-English		Task 2 Malayalam-English	
	NOT	OFF	NOT	OFF	NOT	OFF
Train	150	150	171	168	174	166
Val	328	72	303	297	307	293
Test	334	66	465	475	473	478

Table 4: Number of samples (utterances) for the tasks in high-resource setting. The validation split is created for Task 2 by splitting the original training data into 85% and 15% for training and validation, respectively.

Dataset Statistics for high-resource setting						
Split	Task 1		Task 2 Tamil-English		Task 2 Malayalam-English	
	NOT	OFF	NOT	OFF	NOT	OFF
Train	2633	567	1716	1683	1740	1660
Val	328	72	303	297	307	293
Test	334	66	465	475	473	478

High-resource setting: In the high-resource setting, the complete training data for the tasks is used during fine-tuning procedure. In this way, the performance of the models can be evaluated in cases where large number of annotated samples are available for the task. The details of datasets for this approach is given in Table 4. The models are trained by initializing the trainer with 5 different random seeds and an average of the scores from these on test split is presented. The performance of models could be similar though an improvement is expected for meta-trained models.

Weighted Loss Function: The fine-tuning approach in meta-testing phase is implemented with a weighted cross-entropy loss

[26]. The weighted approach implementation for cross-entropy loss assigns higher weights to class labels with lower number of samples. This method helps the model to learn better on minority classes rather than overfitting on majority classes.

Weighted average F1 score: In all the experiments, there is a need for a common metric with which the performance of fine-tuned models can be compared. One of the most common metric to evaluate the model performance is F1 score. It is defined as the harmonic mean of precision and recall. Its lowest possible value is 0 and highest value is 1. Finally, an average of F1 score across the target classes is taken as the score of the model. A downside of this approach is that it doesn't take into account the number of true samples in each class. Hence, applying this metric when the evaluation datasets are imbalanced could result in all the classes getting equal importance in the average score, which is not the case. A way to tackle this problem is to use the weighted average of F1 score as suggested in Mandl et al. [18] to evaluate the models for the three offensive language identification tasks. This method takes the average of support-weighted mean per class instead of giving equal weights to all samples. The weight values are calculated using number of support samples for each class. Hence, a class with larger number of true samples gets higher weight compared to others. These weights determine the contribution of the associated class to the average. As mentioned before, the validation and test splits for the tasks are imbalanced with samples for "Not offensive" being much higher compared to "Offensive" for Task 1. Even the dataset splits for Task 2 is not perfectly balanced. So, weighted average of F1 score is used to evaluate the fine-tuned models in our experiments.

5 EXPERIMENTAL SETTINGS

As explained in previous section, the meta-learning is split at a high level into meta-training and meta-testing. In the meta-testing phase, the meta-trained model is evaluated in two different scenarios: a) Low-resource and b) High-resource.

5.1 Meta-training

For the meta-training procedure, most of the parameters for experiments are taken from the best performing values found by van der Heijden et al. [36]. These are mentioned in Table 5. We apply cosine annealing to the learning rate for meta-learning. All the models are trained with early stopping for 50 epochs with 100 iterations per epoch. The patience for early stopping is set to 3. The batch size is set to 4 so that each iteration takes 4 support-query sets. Each support-query set is sampled by randomly selecting a dataset and then randomly choosing two classes (or tasks) for that dataset. For each selected class, 8 samples are selected randomly from a uniform distribution for both support and query sets. A total of 32 evaluation tasks are taken in the meta-validation phase with 5 steps per iteration. The maximum length for each input sample is set to 128 since our target tasks majorly consist of short texts.

Following the approach implemented by van der Heijden et al. [36], per-step layer normalization is applied and a separate learning rate is used for each inner loop step in every layer rather than sharing a single learning rate across all the model parameters. The XLM-R model is trained using foMAML and foProtoMAMLn with both approaches 1 and 2. For Approach 2, the models are trained separately for Malayalam-English and Tamil-English code-mixed data. This results in a total of 6 models after meta-training.

Table 5: Experimental settings for meta-training procedure.

Experimental Settings	
Parameter	Value
Inner loop steps	5
Initial inner-loop learning rate	1e-5
Class-head learning rate multiplier	10
Inner-optimizer learning rate	6e-5
Meta-learning learning rate	3e-5

5.2 Meta-testing

The models are fine-tuned and evaluated on all the three offensive language identification tasks. The datasets for Task 2 doesn't contain validation set. Hence, the training set of these datasets is split using stratified approach with 85% and 15% for the resulting training and validation sets. For all these tasks, the models are evaluated in both low-resource and high-resource settings.

Low-resource: The undersampling of the train split for task-specific datasets is performed as defined below.

- **Task 1:** As mentioned before, the dataset for Task 1 is highly imbalanced with fewer samples for the offensive (OFF) class compared to not-offensive (NOT) class. Hence, the dataset is undersampled to a state where the number of samples for each of the class is 150.
- **Task 2 Tamil-English:** The training split of dataset for this task is almost balanced across the classes. Hence, a stratified undersampling is applied to take 10% of the dataset.
- **Task 2 Malayalam-English:** Similar to the dataset for Task 2 Tamil-English, a stratified undersampling to take 10% of the dataset is done.

The models are fine-tuned using 5 different undersampled datasets for each task. The values 42 through 46 (both inclusive) are used as random seeds. The model trainer is initialized with seed value of 42 for all the experiments. The models are trained for 15 epochs without early stopping. Once the training is complete, the model with the lowest validation loss is loaded for evaluation.

High-resource: In the high-resource setting, the models are fine-tuned for 20 epochs and early stopping with a patience of 3 epochs. The models are trained with 5 different random seeds having values 42 through 46 (both inclusive). After training, the best model is loaded based on loss on the validation dataset.

In both low-resource and high-resource scenarios, the parameters of the final classification layer are initialized with values from the model after meta-training except for the baseline XLM-R model. The models are trained with a batch size of 32. A learning rate of 2e-5 is set using a linear scheduler with warmup. The number of warmup steps is 250 for low-resource setting and 500 for high-resource setting. A weight decay of 0.01 is used and the models are evaluated on validation dataset after each epoch. Finally, the best model checkpoint based on validation loss is evaluated on the test dataset.

Table 6: Results for the fine-tuned models in low-resource and high-resource settings. These are mean values of weighted average F1 score on test set over 5 different seeds used for undersampling the training dataset in low-resource setting whereas it is used for trainer initialization in high-resource setting.

Results						
	Low-resource			High-resource		
Model	Task 1	Task 2 Tamil-English	Task 2 Malayalam-English	Task 1	Task 2 Tamil-English	Task 2 Malayalam-English
Baseline (XLM-R Base)	0.760	0.708	0.522	**0.870**	0.858	0.704
Approach 1: foMAML	0.770	0.758	0.458	0.842	0.870	0.728
Approach 1: foProtoMAMLn	0.782	0.736	**0.576**	0.820	**0.884**	0.726
Approach 2: foMAML	**0.820**	**0.772**	0.574	0.856	0.872	0.696
Approach 2: foProtoMAMLn	0.792	0.738	0.548	**0.870**	0.870	**0.738**

Table 7: Results of Welch's t-test between the given model and baseline (XLM-R) using the weighted average F1 score over different seeds. It shows the statistical significance of our results with a lower value meaning more statistically significant. The models that have lower F1 score compared to baseline are marked with N/A.

Results of Welch's t-test						
	Low-resource			High-resource		
Model	Task 1	Task 2 Tamil-English	Task 2 Malayalam-English	Task 1	Task 2 Tamil-English	Task 2 Malayalam-English
Approach 1: foMAML	0.714	0.017	N/A	N/A	0.593	0.511
Approach 1: foProtoMAMLn	0.387	0.215	**0.183**	N/A	**0.196**	0.563
Approach 2: foMAML	**0.004**	**0.007**	0.194	N/A	0.460	N/A
Approach 2: foProtoMAMLn	0.152	0.219	0.537	**1.000**	0.499	**0.349**

As mentioned previously, the fine-tuning procedure is done for each model trained using meta-learning on all the three tasks. The models for Approach 2 are an exception, where they are fine-tuned and evaluated only on Malayalam-English (Task 1, Task 2 Malayalam-English) or Tamil-English (Task 2 Tamil-English) according to meta-training carried out for the model.

6 RESULTS AND ANALYSIS

This section presents the results of the experiments conducted for the meta-training and meta-testing phases.

6.1 Meta-Training

The training for models implementing Approach 1 finished within first few epochs even though the total number of training epochs was set to 50. This is because early stopping is applied when there is no improvement in validation score for these models. The models trained using both the algorithms are having low scores of training and validation accuracy with an average of 0.500 for the best performing checkpoint. When trained using Approach 2, the model for Malayalam-English data lasted for 13 epochs for foMAML and 8 for foProtoMAMLn achieving 0.620 and 0.497 of train accuracy, 0.687 and 0.526 of validation accuracy, 0.615 and 0.693 validation loss for best performing checkpoint. Similarly for the Tamil-English model, the training lasted for 28 epochs for foMAML and 18 for foProtoMAMLn achieving 0.979 and 0.502 of train accuracy, 1.0 and 0.528 of validation accuracy, 0.005 and 0.693 validation loss for best performing checkpoint. This improvement of model performance across the meta-training phase can be attributed to the changes in Approach 2 regarding upsampling of Malayalam-English code-mixed data and selection of appropriate validation datasets. The higher scores on Tamil-English data could be due to the inclusion of romanized Tamil texts in XLM-R pre-training. From the results, the model seems to be overfitting on the data, but an improvement in scores during meta-testing is still expected. It can also be observed that the models trained with foMAML in Approach 2 performs better in meta-training compared to the ones trained with foProtoMAMLn.

6.2 Meta-Testing

The results of meta-testing phase is presented in this section for both low-resource and high-resource settings. The results for all the models in both low-resource and high-resource setting are presented in the Table 6. Also, the results for statistical significance of model performance using Welch's t-test is given in Table 7.

Low-resource: The results show that the our models consistently outperform the baseline score of XLM-R "base" model in the low-resource setting. The models perform better than the baseline on all tasks except for the foMAML model trained with Approach 1 on Task 2 Malayalam-English. Overall, the model trained using foMAML with Approach 2 performs the best on Task 1 and Task 2

Tamil-English with a statistically significant improvement of 0.060 and 0.64 respectively. In Task 2 Malayalam English, it gives results on par with foProtoMAMLn model on Task 2 Malayalam-English, which performs the best with an improvement of 0.054 in F1 score. This shows that Approach 2 helps the model learn better during meta-learning. It also shows that the models are able to successfully transfer their knowledge gained from meta-learning on related tasks such as sentiment analysis to offensive language detection task. It should be noted that the performance of models on Task 2 Malayalam-English is less statistically significant according to Welch's t-test.

High-resource: In the high-resource setting, all training samples available in the dataset for the task is utilized during fine-tuning procedure. It can be observed that the models trained with ProtoMAML algorithm achieve higher scores compared to MAML models. All the meta-trained models outperform the baseline on Task 2 Tamil-English though improvement is not highly significant from a statistical point of view. Similarly, they show a notable improvement in Task 2 Malayalam-English. Again, the results for these models have high value for Welch's t-test meaning they are less statistically significant. The models perform worse on Task 1 compared to the baseline though this task also contains Malayalam-English code-mixed texts. A possible reason for this is because the Task 1 is comparatively easier compared to the Task 2 counterpart, which is explained by the strong baseline performance on this task. It could also be because the models adapted more towards datasets provided during meta-training.

7 CONCLUSION

In this research, we studied the application of meta-learning to identify offensive language in Malayalam-English and Tamil-English code-mixed texts. The results show significant improvement in few-shot classification for the models trained using meta-learning over the XLM-R "base" model. Similar improvements can also be seen for high-resource tasks, especially if the model has been pre-trained with romanized texts of target language. We also introduced a weighted sampling approach in the meta-training phase which improves the model performance on few-shot classification significantly. The results show that meta-learning can improve the performance of models on code-mixed texts using datasets for similar tasks and related languages.

In future, we plan to explore methods to assign the optimal weights for meta-learning datasets. Based on our observations, we also plan to pre-train the XLM-R model on Malayalam-English code-mixed data to improve its performance on downstream tasks.

ACKNOWLEDGMENTS

This publication has been supported in part by a research grant from Science Foundation Ireland (SFI) (SFI/12/RC/2289_P2 Insight_2), co-funded by the European Regional Development Fund and Irish Research Council grant IRCLA/2017/129 (CARDAMOM-Comparative Deep Models of Language for Minority and Historical Languages).

REFERENCES

[1] Gaurav Arora. 2020. iNLTK: Natural Language Toolkit for Indic Languages. In *Proceedings of Second Workshop for NLP Open Source Software (NLP-OSS).* Association for Computational Linguistics, Online, 66–71. https://doi.org/10.18653/v1/2020.nlposs-1.10

[2] Trapit Bansal, Rishikesh Jha, and Andrew McCallum. 2020. Learning to Few-Shot Learn Across Diverse Natural Language Classification Tasks. arXiv:1911.03863 [cs.CL]

[3] Bharathi Raja Chakravarthi. 2020. *Leveraging orthographic information to improve machine translation of under-resourced languages.* Ph.D. Dissertation. NUI Galway.

[4] Bharathi Raja Chakravarthi, Dhivya Chinnappa, Ruba Priyadharshini, Anand Kumar Madasamy, Sangeetha Sivanesan, Subalalitha Chinnaudayar Navaneethakrishnan, Sajeetha Thavareesan, Dhanalakshmi Vadivel, Rahul Ponnusamy, and Prasanna Kumar Kumaresan. 2021. Developing Successful Shared Tasks on Offensive Language Identification for Dravidian Languages. *arXiv preprint arXiv:2111.03375* (2021).

[5] Bharathi Raja Chakravarthi, Navya Jose, Shardul Suryawanshi, Elizabeth Sherly, and John Philip McCrae. 2020. A Sentiment Analysis Dataset for Code-Mixed Malayalam-English. In *Proceedings of the 1st Joint Workshop on Spoken Language Technologies for Under-resourced languages (SLTU) and Collaboration and Computing for Under-Resourced Languages (CCURL).* European Language Resources association, Marseille, France, 177–184. https://www.aclweb.org/anthology/2020.sltu-1.25

[6] Bharathi Raja Chakravarthi, Vigneshwaran Muralidaran, Ruba Priyadharshini, and John Philip McCrae. 2020. Corpus Creation for Sentiment Analysis in Code-Mixed Tamil-English Text. In *Proceedings of the 1st Joint Workshop on Spoken Language Technologies for Under-resourced languages (SLTU) and Collaboration and Computing for Under-Resourced Languages (CCURL).* European Language Resources association, Marseille, France, 202–210. https://aclanthology.org/2020.sltu-1.28

[7] Bharathi Raja Chakravarthi, Ruba Priyadharshini, Shubhanker Banerjee, Richard Saldanha, John P. McCrae, Anand Kumar M, Parameswari Krishnamurthy, and Melvin Johnson. 2021. Findings of the Shared Task on Machine Translation in Dravidian languages. In *Proceedings of the First Workshop on Speech and Language Technologies for Dravidian Languages.* Association for Computational Linguistics, Kyiv, 119–125. https://www.aclweb.org/anthology/2021.dravidianlangtech-1.15

[8] Bharathi Raja Chakravarthi, Ruba Priyadharshini, Navya Jose, Anand Kumar M, Thomas Mandl, Prasanna Kumar Kumaresan, Rahul Ponnusamy, Hariharan R L, John P. McCrae, and Elizabeth Sherly. 2021. Findings of the Shared Task on Offensive Language Identification in Tamil, Malayalam, and Kannada. In *Proceedings of the First Workshop on Speech and Language Technologies for Dravidian Languages.* Association for Computational Linguistics, Kyiv, 133–145. https://aclanthology.org/2021.dravidianlangtech-1.17

[9] Bharathi Raja Chakravarthi, Ruba Priyadharshini, Vigneshwaran Muralidaran, Shardul Suryawanshi, Navya Jose, Elizabeth Sherly, and John P. McCrae. 2020. Overview of the Track on Sentiment Analysis for Dravidian Languages in Code-Mixed Text. In *Forum for Information Retrieval Evaluation* (Hyderabad, India) *(FIRE 2020).* Association for Computing Machinery, New York, NY, USA, 21–24. https://doi.org/10.1145/3441501.3441515

[10] Bharathi Raja Chakravarthi, Ruba Priyadharshini, Rahul Ponnusamy, Prasanna Kumar Kumaresan, Kayalvizhi Sampath, Durairaj Thenmozhi, Sathiyaraj Thangasamy, Rajendran Nallathambi, and John Phillip McCrae. 2021. Dataset for Identification of Homophobia and Transophobia in Multilingual YouTube Comments. *arXiv preprint arXiv:2109.00227* (2021).

[11] Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised Cross-lingual Representation Learning at Scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics.* Association for Computational Linguistics, Online, 8440–8451. https://doi.org/10.18653/v1/2020.acl-main.747

[12] Amitava Das and Björn Gambäck. 2013. Code-Mixing in social media text: the last language identification frontier? *Trait. Autom. des Langues* 54 (2013), 41–64.

[13] Zi-Yi Dou, Keyi Yu, and Antonios Anastasopoulos. 2019. Investigating meta-learning algorithms for low-resource natural language understanding tasks. *arXiv preprint arXiv:1908.10423* (2019).

[14] Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*. PMLR, 1126–1135.

[15] Kushal Kedia and Abhilash Nandy. 2021. indicnlp@kgp at DravidianLangTech-EACL2021: Offensive Language Identification in Dravidian Languages. In *Proceedings of the First Workshop on Speech and Language Technologies for Dravidian Languages.* Association for Computational Linguistics, Kyiv, 330–335. https://www.aclweb.org/anthology/2021.dravidianlangtech-1.48

[16] Arun Kumar, Ryan Cotterell, Lluís Padró, and Antoni Oliver. 2017. Morphological Analysis of the Dravidian Language Family. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers.* Association for Computational Linguistics, Valencia, Spain, 217–222. https://www.aclweb.org/anthology/E17-2035

[17] Anoop Kunchukuttan, Divyanshu Kakwani, Satish Golla, Gokul N.C., Avik Bhattacharyya, Mitesh M. Khapra, and Pratyush Kumar. 2020. AI4Bharat-IndicNLP

Corpus: Monolingual Corpora and Word Embeddings for Indic Languages. *arXiv preprint arXiv:2005.00085* (2020).

[18] Thomas Mandl, Sandip Modha, Anand Kumar M, and Bharathi Raja Chakravarthi. 2020. Overview of the HASOC Track at FIRE 2020: Hate Speech and Offensive Language Identification in Tamil, Malayalam, Hindi, English and German. In *Forum for Information Retrieval Evaluation* (Hyderabad, India) *(FIRE 2020)*. Association for Computing Machinery, New York, NY, USA, 29–32. https://doi.org/10.1145/3441501.3441517

[19] Thomas Mandl, Sandip Modha, Prasenjit Majumder, Daksh Patel, Mohana Dave, Chintak Mandlia, and Aditya Patel. 2019. 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* (Kolkata, India) *(FIRE '19)*. Association for Computing Machinery, New York, NY, USA, 14–17. https://doi.org/10.1145/3368567.3368584

[20] 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* (Prague, Czech Republic) *(HT '17)*. Association for Computing Machinery, New York, NY, USA, 85–94. https://doi.org/10.1145/3078714.3078723

[21] Deepu S. Nair, Jisha P. Jayan, R. R. Rajeev, and Elizabeth Sherly. 2014. SentiMa - Sentiment extraction for Malayalam. In *2014 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*. 1719–1723. https://doi.org/10.1109/ICACCI.2014.6968548

[22] Alex Nichol, Joshua Achiam, and John Schulman. 2018. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999* (2018).

[23] Varsha Pathak, Manish Joshi, Prasad Joshi, Monica Mundada, and Tanmay Joshi. 2021. KBCNMUJAL@ HASOC-Dravidian-CodeMix-FIRE2020: Using Machine Learning for Detection of Hate Speech and Offensive Code-Mixed Social Media text. *arXiv preprint arXiv:2102.09866* (2021).

[24] Adithya Pratapa, Gayatri Bhat, Monojit Choudhury, Sunayana Sitaram, Sandipan Dandapat, and Kalika Bali. 2018. Language Modeling for Code-Mixing: The Role of Linguistic Theory based Synthetic Data. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Melbourne, Australia, 1543–1553. https://doi.org/10.18653/v1/P18-1143

[25] Ruba Priyadharshini, Bharathi Raja Chakravarthi, Sajeetha Thavareesan, Dhivya Chinnappa, Thenmozhi Durairaj, and Elizabeth Sherly. 2021. Overview of the DravidianCodeMix 2021 Shared Task on Sentiment Detection in Tamil, Malayalam, and Kannada. In *Forum for Information Retrieval Evaluation* (Online) *(FIRE 2021)*. Association for Computing Machinery.

[26] Debjoy Saha, Naman Paharia, Debajit Chakraborty, Punyajoy Saha, and Animesh Mukherjee. 2021. Hate-Alert@DravidianLangTech-EACL2021: Ensembling strategies for Transformer-based Offensive language Detection. In *Proceedings of the First Workshop on Speech and Language Technologies for Dravidian Languages*. Association for Computational Linguistics, Kyiv, 270–276. https://www.aclweb.org/anthology/2021.dravidianlangtech-1.38

[27] Siva Sai and Yashvardhan Sharma. 2020. Siva@ HASOC-Dravidian-CodeMix-FIRE-2020: Multilingual Offensive Speech Detection in Code-Mixed and Romanized Text.. In *FIRE (Working Notes)*. 336–343.

[28] Ratnasingam Sakuntharaj and Sinnathamby Mahesan. 2016. A novel hybrid approach to detect and correct spelling in Tamil text. In *2016 IEEE International Conference on Information and Automation for Sustainability (ICIAfS)*. IEEE, 1–6.

[29] Ratnasingam Sakuntharaj and Sinnathamby Mahesan. 2017. Use of a novel hash-table for speeding-up suggestions for misspelt Tamil words. In *2017 IEEE International Conference on Industrial and Information Systems (ICIIS)*. IEEE, 1–5.

[30] Ratnasingam Sakuntharaj and Sinnathamby Mahesan. 2018. Detecting and correcting real-word errors in Tamil sentences. *Ruhuna Journal of Science* 9, 2 (2018).

[31] Ratnasingam Sakuntharaj and Sinnathamby Mahesan. 2018. A Refined POS Tag Sequence Finder for Tamil Sentences. In *2018 IEEE International Conference on Information and Automation for Sustainability (ICIAfS)*. IEEE, 1–6.

[32] Sajeetha Thavareesan and Sinnathamby Mahesan. 2019. Sentiment Analysis in Tamil Texts: A Study on Machine Learning Techniques and Feature Representation. In *2019 14th Conference on Industrial and Information Systems (ICIIS)*. 320–325. https://doi.org/10.1109/ICIIS47346.2019.9063341

[33] Sajeetha Thavareesan and Sinnathamby Mahesan. 2020. Sentiment Lexicon Expansion using Word2vec and fastText for Sentiment Prediction in Tamil texts. In *2020 Moratuwa Engineering Research Conference (MERCon)*. 272–276. https://doi.org/10.1109/MERCon50084.2020.9185369

[34] Sajeetha Thavareesan and Sinnathamby Mahesan. 2020. Word embedding-based Part of Speech tagging in Tamil texts. In *2020 IEEE 15th International Conference on Industrial and Information Systems (ICIIS)*. 478–482. https://doi.org/10.1109/ICIIS51140.2020.9342640

[35] Eleni Triantafillou, Tyler Zhu, Vincent Dumoulin, Pascal Lamblin, Utku Evci, Kelvin Xu, Ross Goroshin, Carles Gelada, Kevin Swersky, Pierre-Antoine Manzagol, and Hugo Larochelle. 2020. Meta-Dataset: A Dataset of Datasets for Learning to Learn from Few Examples. In *International Conference on Learning Representations*. https://openreview.net/forum?id=rkgAGAVKPr

[36] Niels van der Heijden, Helen Yannakoudakis, Pushkar Mishra, and Ekaterina Shutova. 2021. Multilingual and cross-lingual document classification: A meta-learning approach. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*. Association for Computational Linguistics, Online, 1966–1976. https://aclanthology.org/2021.eacl-main.168

[37] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-Art Natural Language Processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Association for Computational Linguistics, Online, 38–45. https://www.aclweb.org/anthology/2020.emnlp-demos.6

[38] Marcos Zampieri, Preslav Nakov, Sara Rosenthal, Pepa Atanasova, Georgi Karadzhov, Hamdy Mubarak, Leon Derczynski, Zeses Pitenis, and Çağrı Çöltekin. 2020. SemEval-2020 Task 12: Multilingual Offensive Language Identification in Social Media (OffensEval 2020). In *Proceedings of SemEval*.

[39] Yukun Zhu, Ryan Kiros, Richard Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning Books and Movies: Towards Story-like Visual Explanations by Watching Movies and Reading Books. arXiv:1506.06724 [cs.CV]