# Text Classification Based on Ensembles of Pre-Trained Models for Sarcasm Identification in Dravidian Languages

Jaime Cerda-Flores[1], Daniel Castro-Pineda[1], Miguel G. Juarez[1], Rodrigo I Hernandez-Mazariegos[1], Jaime Cerda-Jacobo[1], Mario Graff[2] and José Ortiz-Bejar[1]

[1]*Universidad Michoacana de San Nicolas de Hidalgo (UMSNH)*

[2]*CONAHCyT-INFOTEC*

**Abstract**

This manuscript describes the participation of the UMSNH NLP Team in the *Sarcasm Identification of Dravidian Languages: Tamil & Malayalam task at FIRE 2024.* Our approach combines bag-of-words and deep learning models, solving the task independently. We then construct a new feature space by leveraging the decision functions of the individual models. This new feature space is fed into an XGBoost classifier to make a final prediction. The generic text categorization system, FastText, achieves the best performance for the Tamil-based task with a macro F1-score of 0.74. However, our combined model improves upon individual performances for the Malayalam task with a macro F1-score of 0.76.

**Keywords**

Stacking, XGBoost, FastText, mTC

## 1. Introduction

Recent achievements in natural language processing, especially in generative models, have obscured many details about human communications. Analyzing the emotional load in written sentences is not only about identifying patterns; there are complex dependencies on the social and cultural context, educational level, and many other subjective aspects. The set of tasks related to identifying emotions in written/spoken communication is called Sentiment Analysis; this task ranges from identifying if a text/comment has a positive, negative, or neutral to more complex tasks like identifying its level of humor.

In the same line, sarcasm identification is one of the most complex sentiment analysis tasks. Usually, sarcasm is expressed as ironic comments that pretend to convey the opposite of its real meaning. The latter implies that identifying a sarcastic phrase is practically impossible without the appropriate context. Furthermore, due to globalization, it is becoming more common to use slang and terminology from other languages (mainly English). This last point increases the complexity of context identification. Working with mixed languages is denominated as Code-Mixed.

*Forum for Information Retrieval Evaluation, December 12–15, 2024, India*

[†]These authors contributed equally.

1130834h@umich.mx (J. Cerda-Flores); 2220220h@umich.mx (D. Castro-Pineda); 1614873c@umich.mx (M. G. Juarez); 1301441a@umich.mx (R. I. Hernandez-Mazariegos); jaime.cerda@umich.mx (J. Cerda-Jacobo); mario.graff@infotec.mx (M. Graff); jose.ortiz@umich.mx (J. Ortiz-Bejar)

## 2. Task description

The task Sarcasm Identification of Dravidian Languages Tamil & Malayalam aims to identify sarcasm and sentiment polarity in code-mixed YouTube comments/posts written in Tamil-English and Malayalam-English. Each corpus entry is annotated as *Sarcastic*, *Non-Sarcastic*. More details about corpus creation and labeling can be seen at references [1, 2, 3, 4, 5].

For the task, organizers provide two datasets per language: one labeled dataset for training and an unlabeled one for evaluation. Furthermore, the training set of each language is split into a smaller validation set and a larger training set. Table 2 shows the composition of the training and testing sets.

| Language | Dataset | Total Instances | Non-sarcastic | Sarcastic |
|---|---|---|---|---|
| Malayalam | Train | 16014 | 12994 | 3020 |
| | Test | 2826 | No labels available | |
| Tamil | Train | 35906 | 26370 | 9536 |
| | Test | 6338 | No labels available | |

**Table 1**
Dataset description for the Sarcasm detection task in Tamil and Malayalam.

## 3. Related Work

Transformers, introduced by Vaswani et al. [6], revolutionized natural language processing by relying on a self-attention mechanism to capture long-range dependencies in text. Since their introduction, transformers have been successfully applied to various tasks, such as machine translation as implemented through BERT [7] and text generation through GPT-2 [8], demonstrating state-of-the-art performance across multiple domains.

As of recent, transformer based models have also been widely used for text classification as well, which can be observed in the past edition of the Sarcasm Identification of Dravidian Languages task from the Forum for Information Retrieval Evaluation 2023 [1]. Amongst these, the best performing system for the Tamil-English language was achieved by Bhaumik and Das [9] through the use of MuRIL [10], with which they also achieved second place in the Malayalam-English language.

However, not all participants chose to employ these transformer based models. The work presented by Krishnan et al. [11] saw the use of simpler classification methods; MLPs, Random Forests, KNN and SVMs, along with count and frequency based methods for feature extraction, as opposed to word embeddings used in transformers. This implementation proved to be effective as it placed the authors in first place for the Malayalam-English language, demonstrating the continued value of shallow-learning approaches.

FastText, introduced by Joulin et al. proposed an approach to text classification which can be considered a modification of the popular Word2Vec algorithm [12]. Instead of using a Continuous Bag of Words model to train word embeddings to predict surrounding words, it trains them to predict the predefined labels directly. From [12], it has been shown to perform competitively in sentiment analysis tasks.

XGBoost is an optimized gradient boosting framework that has become highly popular in machine learning due to its efficiency and performance in predictive modeling [13]. XGBoost has achieved state-of-the-art results in various structured data tasks, such as the classification of fake news [14].

Ensembles from stacking outputs of different classifiers have been used in [15, 16]. In [15], the authors used this method to classify sarcastic tweets that did not contain the string “#sarcasm”. The approach in [16] sought to enhance the performance of individual classifiers and showed competitive results.

Nevertheless, performance seemed to be affected by the use of sub-sets from the original database for model training due to computational limitations. The latter is addressed in this work.

## 4. Our approach

Roughly, our approach consists of first applying text pre-processing. The processed text is used to train and optimize multiple text classification models $LM$. Then, the decision function $f_{LM}$ values are stacked to produce a new feature space, which is used to train an XGBoost instance; this flow is depicted by the orange line in the Figure 1. For the prediction phase, after performing the pre-processing, the stacked space is created by using the prediction made for each model $LM_i$ and finally fed to the XGBoost to perform the final decision; this phase is indicated by the blue line in Figure 1.

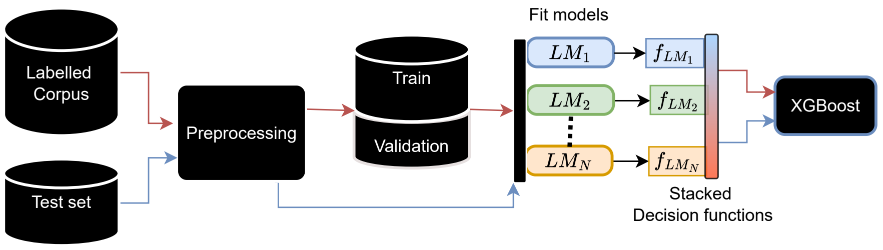

**Figure 1:** Proposed pipeline for ensemble of text classifiers

The rest of the document is organized as follows: Section 4.1 explains the pre-preprocessing details. The basis of the different evaluated classifiers is described in Section 4.2. Experimental setup and results are presented in Section 5. Finally, in Section 6, some conclusions derived from the results are given.

### 4.1. Pre-processing

One pre-processing approach is to transliterate a language into a specific alphabet, such as the Malayalam script or Latin script. This method, while less robust, offers a high level of accuracy, lower computational cost, and superior consistency for this task.

Text to Tamil/Malayalam script means transliterating the text into the Tamil/Malayalam alphabet, Text to Latin Script means transliterating the text from the Tamil/Malayalam alphabet to Latin, Raw text means no pre-processing was done. Text normalization means the text was treated even further, such as removing punctuation marks, URLs, or lower casing.

#### 4.1.1. Tamil/Malayalam Script

The primary objective of this pre-processing task is to identify the alphabet in which the phrase is written and, if needed, transliterate it to the desired alphabet. This is to ensure that the entire data set, in both Malayalam and Tamil languages, is in a single alphabet, specifically the Latin alphabet.

#### 4.1.2. To Latin Script

This pre-processing task involves translating Malayalam and Tamil texts from their source into the Latin alphabet. This is to take advantage of the tokenizers of some multilingual BERT models that require the input in Latin script.

#### 4.1.3. Text Normalization

Text normalization aims to transform raw text into a standardized format easier for machine learning algorithms to process. It reduces complexities and differences in text, such as spelling, formatting, or casing. It allows tokenization and model training to focus on patterns in the text rather than noise.

Normalization procedures, such as lower casing, punctuation and special character removal, stemming, lemmatization, stop word removal, handling of numbers, usernames, or emails, dealing with emojis or emoticons, and handling spelling mistakes, play a vital role in enhancing the quality of text data for machine learning.

## 4.2. Classification Approaches

### 4.2.1. $\mu$TC

$\mu$TC is a text classification framework designed to efficiently find a competitive text classifier by treating the process as a combinatorial optimization problem [17]. It defines a large configuration space, which includes text transformation, tokenization, vectorization, and classification functions. Given the vast number of possible configurations (over 45 million at the time of writing), evaluating all of them is computationally impractical. To address this, $\mu$TC employs the meta-heuristic search techniques Random Search and Hill Climbing to navigate the configuration space efficiently.

In this framework, the objective is to maximize the score function, set as the Macro F1-score, but this setting is customizable. This score evaluates a classifier trained on a given dataset by comparing predicted labels against true labels.

The optimization process begins with Random Search, which selects a subset of configurations and finds the best-performing one. Hill Climbing then explores the neighborhood of this best configuration to find further improvements.

The configuration space includes functions for text transformation (e.g., handling hashtags, URLs, and emojis), tokenization (e.g., n-word grams, q-character grams), vectorization (e.g., TF, TF-IDF), and classification (using an unoptimized SVM with a linear kernel). This flexible setup allows $\mu$TC to handle various text preprocessing and classification tasks.

### 4.2.2. BERT

BERT (Bidirectional Encoder Representations from Transformers) [7] and RoBERTa (Robustly Optimized BERT Pretraining Approach)[18] are transformer-based text models that have shown excellent performance at different classification benchmarks like GLUE, RACE and SQuAD, to name a few. Both RoBERTa and BERT models are designed as bidirectional models that can predict words conditioned on both the left and right contexts. This feature has shown outstanding results in multiple natural language processing tasks, such as text classification.

The BERT transformer model is pre-trained using a static MLM (masked language modeling) and NSP (next-sentence prediction). In the MLM task, about 15% of the words in each sequence are masked, and the model is trained to predict them. On the other hand, RoBERTa is built on top of BERT and modifies key hyperparameters. In the pre-training stage, it uses a dynamic MLM where the masked token is constantly changed. It also completely removes the NSP step.

**Fine-tuning**

Fine-tuning is the process in which the parameters learned from a previous pre-trained model are then transferred to a new model that will work as a starting point for a Natural Language task such as text classification.

| Model | Pre-training approach | Tokenizer |
|---|---|---|
| BERT | Static MLM, NSP | Wordpiece |
| RoBERTa | Dynamic MLM | BPE |

**Table 2**
BERT and RoBERTa text models main differences at pretraining approaches and tokenizers utilized.

For the fine-tuning task done in this study, a sequence classification instance using BERT was generated for each pre-trained BERT or RoBERTa model. These instances contained labels for two categories:"sarcastic" and "nonsarcastic". Data sets were loaded, pre-tokenized, and padded to the max length of the models (512 tokens). Table 3 shows the different models fine-tuned for the sarcasm detection task.

| Model Name | Target Language | Preprocessing | Hyperparameters |
|---|---|---|---|
| robert-malayalam | Malayalam | Text to Tamil Script | Stock Hyperparameters |
| bert-malayalam | Malayalam | Text to Latin Script | |
| l3cube-pune/tamil-bert | Tamil | Text to Tamil Script | |
| l3cube-pune/malayalam-bert | Malayalam | Text to Tamil Script | Warm up steps: 500 |
| l3cube-pune/tamil-bert | Tamil | Raw Text | Learning rate: $1e^{-5}$ |
| roBERTa XLM | Tamil | Text to Tamil Script | Epochs: 7 |
| roBERTa XLM | Malayalam | Text to Latin Script | |

**Table 3**
Preprocessing and hyperparameters for fine-tuning different BERT models. Stock hyperparameters for BERT for text classification are available in https://huggingface.co/docs/transformers/model_doc/bert#transformers.BertForSequenceClassification

**MuRIL**

MuRIL is a BERT-based model pre-trained on 17 Indian languages and their transliterated versions[10]. It has been used in previous text classification competitions with positive results [9]. MuRIL follows a similar training approach to multilingual BERT but includes some modifications; training uses translation and transliteration segment pairs and applies an upsampling factor of 0.3, enhancing performance for low-resource languages. It is trained on both monolingual data (Wikipedia and Common Crawl), translated data (translations of the monolingual data and the PMINDIA dataset), and transliterated data (transliterations from Wikipedia and the Dakshina dataset). The model was trained using whole word masking for 1000K steps, with a batch size of 4096 and a sequence length of 512. It focuses on self-supervised masked language modeling.

### 4.2.3. FastText

FastText is a library predominantly used for generating word embeddings. It is also effective for text classification through techniques like bag-of-words and subword information. It includes various unsupervised and supervised learning algorithms. FastText has pre-trained models for nearly 294 languages, including German, Spanish, French, and Czech. One main characteristic of fastText is its ability to generate word vectors even for unknown, out-of-vocabulary (OOV) words or concatenation of different words. This is due to how word vectors are created by joining substrings of characters in the OOV words.

## 4.3. Ensemble

The ensemble approach consisted of creating a new Vector Space Model (VSM) by horizontally stacking the output probabilities and decision functions obtained from the previous classification approaches, which were observed to perform relatively well individually. The VSM is then used to fit an XGBoost classifier optimized through randomized search cross-validation, which performs the final prediction task. The dimension of the VSM used as input data is given by $C \times x \times n_m$, where $C$ is the number of classes, $x$ is the number of samples in the dataset, and $n_m$ is the number of used outputs or the amount of selected approaches used to create the VSM. In the case of this task, $C$ is always equal to 2 for every approach, save for the case of $\mu$TC, as the output decision function from its SVM classifier is a positive number for a sarcastic prediction and a negative number for a non-sarcastic prediction, whereas the rest of the approaches output a pair of probabilities for a sample being of each class.

In order to obtain the outputs used to train the XGBoost classifiers, as well as to validate them and get the final predictions from the test set, the models obtained from each approach were used on the training sets, the validation sets, and the test sets for each language.

### 4.3.1. XGBoost

XGBoost is an open source machine learning library designed to implement efficient distributed gradient boosting algorithms [19]. It has been widely used by winning teams in machine learning competitions. The core concept behind gradient boosting is using weak models, in this case shallow decision trees, to build a strong decision tree. This building process is done sequentially, until a specified number of iterations is met. Each new built model tries to correct the errors made by the previous models by minimizing a specified loss function using gradient descent, the first derivative of the loss function.

Unlike the normal gradient boosting algorithm, XGBoost uses the second derivative of the loss function, it's Hessian, as well as the first derivative, essentially turning the gradient descent optimization into Newton-Raphson optimization. It also uses L1 and L2 regularization to control model complexity, making it less prone to overfitting by penalizing large trees and overly complex models.

## 5. Experiments and results

For our experiments, we used only the training and validation sets as provided by the competition organizers. The compositions of each set are shown in Table 5.

| Language | Dataset | Total Instances | Non-sarcastic | Sarcastic |
|---|---|---|---|---|
| Malayalam | Train | 13188 | 10689 | 2499 |
| | Validation | 2826 | 2305 | 521 |
| Tamil | Train | 29570 | 21740 | 7830 |
| | Validation | 6336 | 4630 | 1706 |

**Table 4**
Dataset split for the Sarcasm detection task in Tamil and Malayalam.

### 5.1. $\mu$TC

As the $\mu$TC classification pipeline involves a series of text pre-processing steps, the raw datasets were fed to it as is, specifying the use of 5-fold cross-validation and the space configuration search to take place in 80 points. A single model was trained per-language. Table 5 shows the parameters obtained for Malayalam and Tamil after optimization.

### 5.2. FastText

For the use of FastText, all training and validation sets, as well as the test sets, were preprocessed by performing the following text normalization procedures using a custom function:

| Parameter name | Dravidian dialect | |
|---|---|---|
| | Malayalam | Tamil |
| lower-case | false | true |
| emojis-handler | none | none |
| hashtag-handlers | none | delete |
| url-handler | delete | delete |
| user-handler | group | delete |
| number-handler | none | delete |
| diacritic-removal | true | true |
| duplication-removal | true | true |
| punctuation-removal | true | false |
| $q$-grams | $1, 3, 4$ | $1, 2, 3, 5, 9$ |
| $n$-grams | $3, 2, 1$ | $3, 2$ |
| skip-grams | (3,1) | none |
| weighting scheme | tfidf | tfidf |
| token-max-filter | 1 | 1 |
| token-min-filter | $-1$ | $-1$ |

**Table 5**
$\mu$TC best configuration parameters for Malayalam and Tamil texts

- Lower casing
- URL removal
- Removal of usernames
- Removal of non-alphanumerical characters
- Conversion of emojis and emoticons to textual representation

In addition to this, the training and validation sets were transformed to comply with the default required format for the use FastText. This is to say, the string "__label__*c*" was placed at the beginning of every sample in each set, where *c* represents the real label for a given sample.

A single model was trained per-language using both the default FastText parameters and using the hyperparameter optimization option provided by FastText, setting the time limit to 10 minutes. This resulted in 2 models per-language.

## 5.3. BERT

All BERT and RoBERTa models, save for MuRIL and Multilingual, were fine-tuned for a text classification task. Subsequently, each model was tested across the target languages to evaluate its performance in detecting sarcasm. This involved the following methodology:

- Both the BERT and RoBERTa models were fine-tuned using the prepared dataset. This process involved adjusting the pre-trained models with warm-up steps and learning rates for the sarcasm classification task.
- After fine-tuning, each model was tested on the test sets provided for each target language. Performance metrics such as accuracy, precision, recall, and F1 score were calculated to select the most effective models to be later used in the ensemble.
- Once the most effective models were identified, the probability scores for each dataset were computed. These scores were then used to create the ensemble model, a significant outcome of our methodology.

In the case of MuRIL and BERT Multilingual, the pre-trained model was used to obtain document embeddings from each document in every dataset. Instead of using the document embedding provided by the model, the token embeddings from every token in a given document were summed and averaged;

this averaged embedding was used as the document embedding. These document embeddings were used in the training process for each language to train a logistic regression classifier for each language with the use of 5-fold stratified cross-validation, which were then used to make predictions and obtain probability scores from the training, validation, and test datasets for later use in the ensemble model. It is important to note that prior to passing the text through MuRIL, it was normalized by performing the following operations: removing URLs, converting emojis and emoticons to textual descriptions, converting the text to lowercase, replacing usernames with a placeholder, removing specified punctuation and symbols, and normalizing whitespace. The text was left in code-mixed format instead of transliterating, as opposed to the other uses of BERT models. For MuRIL, an experiment was also done with transliterated text.

## 5.4. XGBoost

The selected configurations to create the VSMs for use with XGBoost are detailed in table 6. Please note that a model was created for each language in all configurations, using the output probabilities from the respective language to build the VSM.

| XGBoost Configuration |
|---|
| MuRIL + fastText |
| MuRIL + fastText + Multilingual BERT |
| MuRIL + fastText + $\mu$TC + l3cube-pune |
| MuRIL + fastText + $\mu$TC + l3cube-pune + RoBERTaXLM |

**Table 6**
Configurations used to train XGBoost classifiers

Each XGBoost model was built by randomized search cross-validation using stratified k-folds with five splits. The parameters with their respective values that were chosen for the random search are detailed in table 7.

| Parameter | Values |
|---|---|
| n_estimators | 100, 600, 1000 |
| min_child_weight | 1, 5, 10 |
| gamma | 0.5, 1, 1.5, 2, 5 |
| learning_rate | 0.01, 0.05, 0.1, 0.15, 0.2 |
| subsample | 0.6, 0.8, 1.0 |
| colsample_bytree | 0.6, 0.8, 1.0 |
| max_depth | 3, 4, 5 |

**Table 7**
Parameter Grid for Randomized Search in XGBoost

## 5.5. Results

Table 8 shows the results of all trained models and ensembles for both languages. As seen in Table 8, the ensemble models perform better in at least one of the metrics than individual models for Malayalam and exhibit the best F1 performance for Tamil.

As organizers allow three submissions, we decide to submit the results obtained with the models boldfaced in Table 8. For the Tamil case, the ensemble model was ranked at the top of the contestants, as shown in Table 9. However, for the Malayalam dialect, the optimized FastText approach was ranked at the top tied with the other five teams, as shown in Table 10.

| Model Name | Malayalam | | Tamil | |
|---|---|---|---|---|
| | macro-F1 | Accuracy | macro-F1 | Accuracy |
| Default fastext+prepocessing | 0.69 | 0.85 | 0.7 | 0.78 |
| 2gram fastext+prepocessing | 0.65 | 0.85 | 0.7 | 0.79 |
| bert-(malayalam/tamil) | 0.74 | 0.86 | 0.73 | 0.79 |
| $\mu$tc | 0.73 | 0.86 | 0.75 | 0.81 |
| l3cube-pune/(malayalam/tamil)-bert | 0.73 | 0.86 | 0.74 | 0.80 |
| roBERTa XLM | 0.74 | 0.85 | 0.74 | 0.80 |
| **optimized_fasttext** | 0.72 | 0.85 | 0.75 | 0.81 |
| bert_muril_malayalam | 0.72 | 0.86 | - | - |
| bert_muril_tamil | - | - | 0.69 | 0.76 |
| optimized xgboost (bert muril + fasttext) | 0.73 | 0.86 | 0.75 | 0.80 |
| optimized xgboost (bert muril + fasttext + bert multilingual) | 0.73 | 0.86 | 0.75 | 0.81 |
| **optimized xgboost (bert muril + fasttext + mtc + bert/l3cube-pune)** | 0.74 | 0.86 | 0.76 | 0.81 |
| **optimized xgboost (bert muril + fasttext + mtc + bert/l3cube-pune + RoBERTaXLM)** | 0.74 | 0.86 | 0.76 | 0.81 |

**Table 8**
Accuracy and Macro-F1 results on validation set across the different trained models and formed ensembles

| Team Name | F1-Score | Rank |
|---|---|---|
| UMSNH_NLP | 0.76 | 1 |
| Awsathama | – | 2 |
| Codespark | – | 3 |
| IRLab@IITBHU | – | 3 |
| Sarcasm_NLP | – | 4 |
| MUCS | – | 4 |
| PixelPhrase | – | 4 |

**Table 9**
Final ranking for Malayalam language

| Team Name | F1-Score | Rank |
|---|---|---|
| Awsathama | – | 1 |
| Team_catalysts | – | 1 |
| Change_makers | – | 1 |
| MUCS | – | 1 |
| UMSNH_NLP | 0.74 | 1 |
| IRLab@IITBHU | – | 1 |
| Sarcasm_NLP | – | 2 |

**Table 10**
Final ranking for Tamil language

## 6. Conclusions

At this point, there is a rapid and constant rise of new and efficient language models. The latter provides researchers with a comprehensive variety of tools; integrating them could be challenging. Our team integrated the knowledge using XGBoost over stacked VSM to take advantage of multiple text classification approaches. Our approach ranks at the top in both languages. However, even though ensembles perform consistently better in our validation partition, the ensemble only outperforms other approaches for the Malayalam task, while in the Tamil task, FastText is the one on rank 1. The latter suggest that the correct model can depend on the input data, or possible overfitting from the ensemble model, but more research is needed to rectify this. Also, while this approach proves to be effective, the setup is relatively complex to implement and computationally intensive. This leaves open the possibility of further optimizing the process to simplify our implementation.

## Acknowledgment

Authors from the Universidad Michoacana acknowledge the support provided by CONAHCYT through Project CF-2023-I-1174 within the framework of the "Ciencia de Frontera 2023".

## Declaration on Generative AI

During the preparation of this work, the authors used Grammarly to grammar and spelling check. After, the authors reviewed and edited the content as needed and took full responsibility for the publication's content.

## References

[1] B. R. Chakravarthi, N. Sripriya, B. Bharathi, K. Nandhini, S. C. Navaneethakrishnan, T. Durairaj, R. Ponnusamy, P. K. Kumaresan, K. K. Ponnusamy, C. Rajkumar, Overview of the shared task on sarcasm identification of dravidian languages (malayalam and tamil) in dravidiancodemix, in: Forum of Information Retrieval and Evaluation FIRE-2023, 2023.

[2] N. Sripriya, T. Durairaj, K. Nandhini, B. Bharathi, K. K. Ponnusamy, C. Rajkumar, P. K. Kumaresan, R. Ponnusamy, C. Subalalitha, B. R. Chakravarthi, Findings of shared task on sarcasm identification in code-mixed dravidian languages, FIRE 2023 16 (2023) 22.

[3] B. R. Chakravarthi, Hope speech detection in youtube comments, Social Network Analysis and Mining 12 (2022) 75.

[4] B. R. Chakravarthi, A. Hande, R. Ponnusamy, P. K. Kumaresan, R. Priyadharshini, How can we detect homophobia and transphobia? experiments in a multilingual code-mixed setting for social media governance, International Journal of Information Management Data Insights 2 (2022) 100119.

[5] B. R. Chakravarthi, S. N, B. B, N. K, T. Durairaj, R. Ponnusamy, P. K. Kumaresan, K. K. Ponnusamy, C. Rajkumar, Overview of sarcasm identification of dravidian languages in dravidiancodemix@fire-2024, in: Forum of Information Retrieval and Evaluation FIRE - 2024, DAIICT , Gandhinagar, 2024.

[6] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, I. Polosukhin, Attention is all you need, 2023. URL: https://arxiv.org/abs/1706.03762. arXiv:1706.03762.

[7] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, Bert: Pre-training of deep bidirectional transformers for language understanding, 2019. URL: https://arxiv.org/abs/1810.04805. arXiv:1810.04805.

[8] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, Language models are unsupervised multitask learners, 2019. URL: https://api.semanticscholar.org/CorpusID:160025533.

[9] M. D. Anik Basu Bhaumik, Sarcasm detection in dravidian code-mixed text using transformer-based models, FIRE 2023 16 (2023) 10.

[10] S. Khanuja, D. Bansal, S. Mehtani, S. Khosla, A. Dey, B. Gopalan, D. K. Margam, P. Aggarwal, R. T. Nagipogu, S. Dave, et al., Muril: Multilingual representations for indian languages, arXiv preprint arXiv:2103.10730 (2021).

[11] B. B. Dhanya Krishnan, Krithika Dharanikota, Cross-linguistic sarcasm detection in tamil and malayalam: A multilingual approach, FIRE 2023 16 (2023) 11.

[12] A. Joulin, E. Grave, P. Bojanowski, T. Mikolov, Bag of tricks for efficient text classification, arXiv preprint arXiv:1607.01759 (2016).

[13] T. Chen, C. Guestrin, Xgboost: A scalable tree boosting system, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '16, Association for Computing Machinery, New York, NY, USA, 2016, p. 785–794. URL: https://doi.org/10.1145/2939672.2939785. doi:10.1145/2939672.2939785.

[14] J. P. Haumahu, S. D. H. Permana, Y. Yaddarabullah, Fake news classification for indonesian news using extreme gradient boosting (xgboost), IOP Conference Series: Materials Science and Engineering 1098 (2021) 052081. URL: https://dx.doi.org/10.1088/1757-899X/1098/5/052081. doi:10.1088/1757-899X/1098/5/052081.

[15] R. Bagate, S. Ramadass, Sarcasm detection of tweets without #sarcasm: Data science approach, Indonesian Journal of Electrical Engineering and Computer Science 23 (2021) 993. doi:10.11591/ijeecs.v23.i2.pp993-1001.

[16] J. Cerda-Flores, R. Hernández-Mazariegos, J. Ortiz-Bejar, F. Calderón-Solorio, J. Ortiz-Bejar, Umsnh at restmex 2023: An xgboost stacking with pre-trained word-embeddings over data batches, 2023, pp. 1–10. URL: https://ceur-ws.org/Vol-3496/restmex-paper19.pdf.

[17] E. S. Tellez, D. Moctezuma, S. Miranda-Jiménez, M. Graff, An automated text categorization framework based on hyperparameter optimization, Knowledge-Based Systems 149 (2018) 110–123.

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

[19] T. Chen, T. He, M. Benesty, V. Khotilovich, Package 'xgboost', R version 90 (2019) 40.