Multimodal Sentiment Analysis of Tamil and Malayalam D4

Abhinav Patil, Sam Briggs, Tara Wueger, D. D. O'Connell
Department of Linguistics, University of Washington
Seattle, WA
{abhinavp, briggs3, taraw28, danieloc}@uw.edu

Abstract

Sentiment analysis consists of categorizing a unit of language about a certain topic according to the author's attitude towards that topic. Our task is the classification of multi-modal data – text and audio data – in the Dravidian languages Tamil and Malayalam into 5 separate sentiment classes: highly negative, negative, neutral, positive, and highly positive. We implement and evaluate multiple classifier models: a Multinomial Naive Bayes baseline, a Logistic Regression, a Random Forest, and an SVM for comparison. We also implement and evaluate a finetuned XLM-RoBERTA model. To account for class imbalance, we use both naive resampling and SMOTE. We found that without resampling, both the baseline and the Neural Network have the same performance as a naive Majority Class Classifier baseline. However, with resampling, logistic regression and random forest both demonstrate gains over the baseline. Finally, we perform error analysis of the models.

1 Introduction

Sentiment analysis is a natural language processing task which consists of categorizing a unit of language about a certain topic according to the author's attitude towards that topic; it has a rich history and many methods have been used in the past (Cui et al., 2023). Although the unit of language in question typically takes the form of text, sentiment analysis of multimodal content has become an increasingly important task in recent decades as such content has become much more common. Users of popular social media websites, for instance, have long grown accustomed to creating and interacting with content which has either a textual, auditory, or visual form, or some combination of these three modalities, e.g., a YouTube video with subtitles features all three at once. Training models for sentiment analysis in such contexts requires consideration of features of all modalities concerned. Practically speaking, this task can be constructed in many ways, e.g. as a binary classification task (i.e., categorizing language in two classes, Positive and Negative sentiment), an ordinal regression problem, or — as in our case — a multi-class classification problem.

1.1 Motivation

Although Dravidian languages, such as Telugu, Tamil, Kannada, or Malayalam, are spoken by more than 250 million people, they remain decidedly low-resource, in the sense that very few natural language processing resources exist for them. This paper considers data in two Dravidian languages, namely Tamil and Malayalam. The agglutination and high degree of morphological complexity exhibited by both languages presents significant challenges in the development of useful NLP resources. This makes them ideal candidates for a sentiment analysis task, firstly because approaches which can more effectively extract useful information from limited data are especially useful in this kind of low-resource context, and secondly because progress in the development of such approaches may prove useful to those working with either Dravidian languages, morphologically-complex languages, or low-resource languages more generally.

2 Task Description

We will participate in the "Multimodal Abusive Language Detection and Sentiment Analysis: DravidianLangTech@RANLP 2023" shared task hosted on CodaLab (B. et al., 2023). [1]

This shared task has two subtasks: abusive language detection in Tamil, and sentiment analysis in

[1]This is last year's citation for the same shared task. The official citation has yet to be released as of June 7, 2023.

both Tamil and Malayalam (members of the Dravidian language family). Both tasks are multimodal, consisting of videos paired with files containing just their audio tracks as well as (sometimes partial) text transcripts. For the purposes of this class, we will be working on the second subtask of sentiment analysis in Tamil and Malayalam, and using only text and audio data.

2.1 Shared Task Evaluation

The organizers of the competition have stated that they will use an F1 metric for evaluation purposes. They have not release evaluation tools or scripts at this time, but suggest teams use Sklearn's classification report function (which lists per-class accuracy and overall micro/weighted average/macro precision, recall, and F1) in evaluating performance, suggesting that they may use the same function in evaluation tools (whether eventually released or not). We have followed their advice in this matter. In this paper, we largely focus on macro-F1 and accuracy only, for conciseness.

2.2 Primary Task: Sentiment Analysis of Text Data

Our task is to categorize text written in either Tamil or Malayalam into 5 different categories: Highly Negative, Negative, Neutral, Positive, or Highly Positive. The data is ordinal, that is, the categories are discrete and there exists a total order over them, but the distances between the categories are taken to be unknown or not well-defined. In our initial attempts, we tried formulating the task both in terms of ordinal regression and in terms of multiclass classification.

Previous work on sentiment analysis on Dravidian languages has been done using code-switched data (English and either Tamil or Malayalam) (Chakravarthi et al., 2021). Our data is not code-switched, and only contains one language. Like many tasks in Natural Language Processing, different Neural Network architectures have been used to perform sentiment analysis (Habimana et al., 2019).

2.3 Adaptation Task: Sentiment Analysis of Multimodal Data: Text and Audio

For our adaptation task, we will continue to work on sentiment analysis with audio data in addition to the text data from the primary task. This task will also be multilingual, using data from both Tamil and Malayalam, and the same five categories will be used in the adaptation task as in the primary task.

2.4 Data set

2.4.1 Description

The shared task organizers have provided 54 training samples for Tamil and 60 training samples for Malayalam, as well as 10 test samples for both. Each sample consists of an audio file containing speech in the given language and a (sometimes partial) transcript of that audio. (Although they have provided video files in addition to audio and text, we do not make use of them.) Every sample is a movie review collected from YouTube and labeled by human annotators.

2.4.2 Split

The task organizers have provided a train and dev split over the data set, which is further subdivided by language (Tamil and Malayalam). The 54 Tamil samples are split 44/10 train/dev while the Malayalam samples are split 50/10.

However, instead of using the official split, we combined the train and dev data into a single train set, and then used k-fold validation as we explain later.

For evaluation, an official evaluation data set will be provided in the coming weeks by the organizers.

The official data can be found in Google Drive folder linked below:

- Tamil: train, dev, test
- Malayalam: train, dev, test

3 System Overview

As you see in Figure 1, there are four main stages to our system pipeline, namely preprocessing/tokenization, vectorization, resampling, and K-fold Cross Validation.

First, we preprocess and tokenize the data. To preprocess the data we use two different tokenization methods the data. For the baseline models, we use whitespace tokenization after removing stopwords, punctuation, and numbers. For more sophisticated models, we use the SentencePiece algorithm used in XLM-RoBERTa (Conneau et al., 2019).

Second, we then vectorize each input sample to produce document feature vectors. We have three main vectorization methods. The baseline models use TF-IDF vectorization. We also try two different vectorization methods using XLM-RoBERTa,

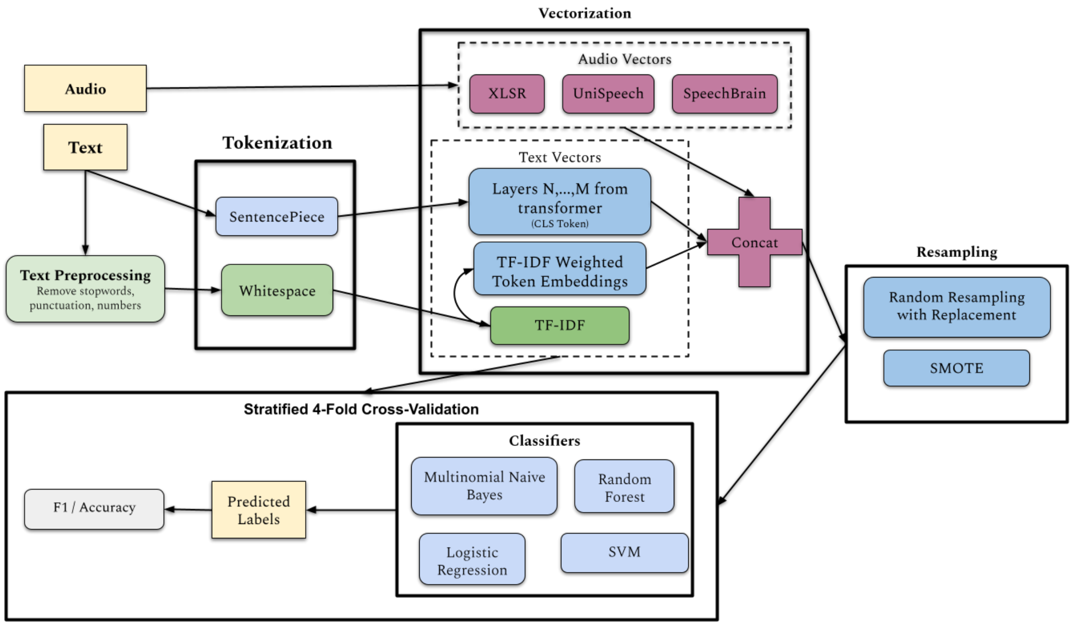


Figure 1: End-to-End System Architecture

namely using TF-IDF weighted average token embeddings from the first layer of XLM-RoBERTa base model, as well as using CLS token representations from hidden layers n through m of the XLM-RoBERTa base model.

Third, we do resampling, using two methods, namely random resampling with replacement as well as Synthetic Minority Over-sampling Technique (SMOTE) (Chawla et al., 2002).

Finally, we run stratified k-fold cross validation with $k = 4$. K-fold cross validation consists of two main components: classification and evaluation. For classification, we classify each document into the 5 different sentiment categories. To classify, we first train various models using the training data, and then use the various trained models to predict the sentiment of the development documents specified by the current fold. For evaluation, we evaluate the performance of each model using accuracy and F_1 score.

4 Approach

4.1 D2

4.1.1 Preprocessing/Tokenization

WhiteSpace During preprocessing for the baseline, we first tokenized the data by whitespace. We then removed any tokens containing punctuation or numbers, as well as stop words. We used the list of stop words provided by `spaCy`[2] for both Tamil and Malayalam.

SentencePiece For the more sophisticated models, we use the XLM-RoBERTa (Conneau et al., 2019) tokenizer, which was trained using the SentencePiece algorithm (Kudo and Richardson, 2018), as well as IndicBert (Kakwani et al., 2020; Doddapaneni et al., 2022). Text longer than the model maximum input length (512, or 510 after accounting for special tokens) is truncated, while text shorter than it is padded.

4.1.2 Text Vectorization

TF-IDF Vectors We then create one TF-IDF vector per document in the data set. To create the TF-IDF vectors, we run each document through the TF-IDF vectorizer provided by `sklearn`.[3]

To calculate TF-IDF, we used the unsmoothed TF-IDF provided by `sklearn`. Given a document set D with n documents, a document $d \in D$, and a term t with document frequency $df(t)$, we calculate TF-IDF for term t as follows:

[2]We used spaCy v3.*. The Tamil and Malayalam language models can be found here.

[3]We used sklearn v1.*. `Sklearn`'s TF-IDF vectorizer documentation can be found here

$$\text{TFIDF}(t,d,D) = tf(t,d) \cdot \text{idf}(t) \quad (1)$$

$$tf(t,d) = count(t) \in d \quad (2)$$

$$\text{idf}(t,D) = \log\left(\frac{n}{df(t)}\right) + 1 \quad (3)$$

4.1.3 Classifiers

Multinomial Naive Bayes For our baseline, we trained a multinomial Naive Bayes classifier on the TF-IDF feature vectors from section 4.1.2. To run Naive Bayes, we used the `sklearn` library (Pedregosa et al., 2011).

Fine-Tuned XLM-RoBERTa Model We also finetuned an XLM-RoBERTa model (Conneau et al., 2019). We tried training across a variety of hyperparameter settings: we varied the number of epochs from 5 to as many as 1000, we tried freezing or unfreezing various layers at various points during training, and we tried training with both a multiclass classifier head and a regression head. However, in terms of actual accuracy and macro-F1 over the k-fold validation sets, results did not vary.

4.2 D3

4.2.1 Text Vectorization

Pretrained Multilingual Model Concatenated Hidden State Embeddings In this approach, we pass each document through a pretrained transformer language model and extract the hidden state representation for a specific set of layers, e.g. the last four, final, the second-to-last, third-to-last, etc. Then, we extract the vector corresponding to the CLS token from each layer and concatenate them together. We tried a variety of model and layer number combinations. For models, we looked at XLM-RoBERTA (base and large and Indic Bert. For the layers, we looked at the last four concatenated, and each of the last four separately. We found the second-to-last layer of XLM-RoBERTA (base) performed the best, and it is these results we will be presenting.

4.2.2 Resampling

SMOTE For Tamil, we also resampled the data using SMOTE (Chawla et al., 2002) with $k = 2$. In cross-validation, we could not use SMOTE for Malayalam due to the fact that there was only a single HIGHLY NEGATIVE sample, meaning three folds lacked it altogether. We were constrained to a small k for a similar reason.

Random Resampling with Replacement For both languages, we also tried random resampling with replacement, where we augmented each minority label with duplicates randomly drawn with replacement from the same label, until all classes had an equal number of samples.

4.2.3 Classifiers

Logistic Regression, Random Forest, and SVM We trained a linear Logistic Regression classifier, a Random Forest Classifier, and an SVM using `sklearn` (Pedregosa et al., 2011). We tested different combinations of hyperparameters, such as the solver, penalty, regularization strength, C, loss, alpha, learning rate, and the maximum number of iterations. Although these classifiers, under certain (relatively rare) conditions, could replicate the results of the Naive Bayes classifier on default settings, it never outperformed it.

Fine-Tuned Indic Bert Model We also finetuned a IndicBert [4] model. Similar to XLM-RoBERTa, we tried training across a variety of hyperparameter settings. The results were the same in that adjusting these hyperparameter settings did not affect overall accuracy very much.

4.3 D4

4.3.1 Text Vectorization

TF-IDF Weighted Token Embeddings We implemented a tokenization strategy using TF-IDF weighted average token embeddings from the embedding (first) layer of the XLM-RoBERTa base model. To obtain document feature vectors, we tokenize a document d using SentencePiece as outlined in section 4.1.1.

We thus have a sequence of n tokens, $\{t_n\} = d$. For each token $t_i \in d$, we obtain the embedding, $\vec{e_i}$, corresponding to token t_i. To obtain $\vec{e_i}$, we pull $\vec{e_i}$ from the embedding (first) layer of XLM-RoBERTa. We also obtain the TF-IDF score[5], $\text{tf\_idf}(t_i, d)$, corresponding to each token t_i in document d. Then the document feature vector $\vec{f_d}$ for document d is exactly:

$$\vec{f_d} = \sum_{i=1}^{n} \mathrm{tf\_idf(t_i, d)} \cdot \vec{e_i} \quad (4)$$

In other words, each document feature vector is the weighted average of token embeddings, using TF-IDF weights.

[4]HF: indic-bert (Wolf et al., 2020)

[5]As calculated in section 4.1.1

4.3.2 Audio Vectorization

We tried three main audio vectorization methods from three different pretrained models, namely XLS-R (Conneau et al., 2020), UniSpeech (Wang et al., 2021), and SpeechBrain (Lugosch et al., 2022a). To obtain document feature vectors, we first pass raw audio into the built-in feature extractors from two models, Wav2Vec2 (Baevski et al., 2020) and M-CTC-T (Lugosch et al., 2022b) using HuggingFace (Wolf et al., 2020). We then pass the output of the feature extractors into the three different pre-trained models detailed below. To extract document feature vectors, we then perform an element-wise average of the second to last hidden layer of the chosen pretrained model.

XLS-R For XLS-R we used Facebook's XLSR-Wav2Vec2 pretrained model `facebook/wav2vec2-large-xlsr-53`[6]. This model builds on and shares the same architecture as Wav2Vec2 (Baevski et al., 2020). This model was trained on CommonVoice, Babel, and Multilingual LibriSpeech (MLS), for a total of 53 languages, including Tamil but not Malayalam.

UniSpeech For UniSpeech, we used Microsoft's pretrained model `microsoft/unispeech-large-multi-lingual-1500h-cv`[7]. This model builds on and shares the same architecture as Wav2Vec2 (Baevski et al., 2020). This model was trained on four languages from CommonVoice: English, Spanish, French, and Italian. Although Tamil and Malayalam are not closely related to those four languages, this model was trained on the phoneme level rather than the character level, and thus we believe that this model is good for transfer learning.

SpeechBrain For SpeechBrain, we used Meta's pretained model `speechbrain/m-ctc-t-large`[8]. This model builds on and shares the same architecture as M-CTC-T proposed by Lugosch et al. (2022b). This model was trained on all 60 langauges from CommonVoice 6.1 (Ardila et al., 2020) and 19 (mainly) European langauges from VoxPopuli for a total of 79 different languages, including Tamil but not Malayalam.

[6]HF: wav2vec2-large-xlsr-53
[7]HF: UniSpeech
[8]HF: m-ctc-t-large

4.3.3 Language Pooling

For the adaptation task, we pooled both the training data for Tamil and Malayalam language data together to create a new larger training data set and trained a single model on the pooled training data. Pooling the data together produced better results than language-specific models.

4.3.4 Ridge Regression Feature Selection

For the D4 models, we used a ridge regression model (linear regression with L2 regularization) for feature selection. All coefficients of the ridge regression model whose absolute values were smaller than the mean absolute value of all coefficients were dropped from consideration in the final classifier. We used a regularization strengh hyperparameter (α) of 0.30. This helped decrease the immense feature sparsity that we were faced with in this task.

5 Results

5.1 Evaluation

To evaluate our classifiers, we pooled the official train and dev splits into one large training set, and then ran k-fold Cross-Validation. This allowed us to train our models on more training data, as there was not a lot of training data provided. For k-fold Cross-Validation, we used 4 folds. To be able to compare the different models that we created, we used a deterministic algorithm to shuffle the data. This ensures that the same four train and dev splits were used for all the models, making the performance of our different models comparable.

First, for reference, in table 1 we provide the results of a majority-class classifier (one that assigns all classes the majority class, `POSITIVE`) as a baseline. This is identical to the performance of all of our D2 models.

For both the primary and adaptation tasks, we have selected the best-performing model in terms of the pooled dev macro-f_1 score. For this model, we report the validation pooled f_1 scores and accuracy averaged across five seeds.[9] For the test data, we report the same metrics averaged across the same seeds. These are in Tables 2 and 3 for the primary and adaptation tasks, respectively.

Additionally, in Table 4 we report the dev and test results of the best model for any specific seed in terms of f_1 score on the test data, whose results we submitted to the shared task. This "best model" was for our primary task, i.e., its input was only the text

[9]0, 42, 100, 573, 2023.

(across the board, the text audio models did words due to greater feature sparsity). Note that this single seed happened to perform especially well, but is not representative of overall model performance.

	Dev		Test	
	Acc	F_1	Acc	F_1
Tamil	0.61	0.15	0.30	0.09
Malayalam	0.60	0.15	0.50	0.13

Table 1: Majority-class classifier / D2 Models

	Dev		Test	
	Acc	F_1	Acc	F_1
Tamil	0.28	0.26	0.35	0.19
Malayalam	0.53	0.40	0.30	0.14

Table 2: (Five-seed average) Primary task: Logistic Regression ($C = 5.0$), Ridge Regression feature selection, random sampling with replacement; text vectorization: XLM-RoBERTa, second-to-last layer.

	Dev		Test	
	Acc	F_1	Acc	F_1
Tamil+Malayalam	0.60	0.35	0.56	0.14
Tamil	N/A	N/A	0.50	0.13
Malayalam	N/A	N/A	0.61	0.15

Table 3: (Five-seed average) Adaptation task (pooled language training, text+audio) - Logistic Regression ($C = 5.0$), Ridge Regression feature selection, random sampling with replacement; text vectorization: XLM-RoBERTa, second-to-last layer; audio vectorization: XLSR, third-to-last layer..

6 Discussion

In our D2 baselines, without resampling, our classifiers did not perform better than a majority class classifier due to the fact that the amount of data available to train on is fairly small and most of the true labels are POSITIVE. For D3, we found using SMOTE for Tamil increased macro F1.

Because our models were trained on so few samples, the use of too many features resulted in overfitting; for this reason, models with fewer features generally performed better.

Because there are so few samples, our results display high variance. This was true for both the training data and the test data. Because there are only ten test samples per language, across seeds we saw a great amount of variance in scores. We chose to submit the best results among all random seeds, achieved in the text-only (primary) task setting, but we do not feel that these are necessarily indicative of our model performance in general. To make an informed statement about the quality of our model would require access to significantly more test data.

	Dev		Test	
	Acc	F_1	Acc	F_1
Tamil	0.26	0.26	0.50	0.29
Malayalam	0.40	0.31	0.40	0.28

Table 4: Best overall test results - Text only, Logistic Regression ($C = 5.0$), Ridge Regression feature selection, random sampling with replacement; text vectorization: XLM-RoBERTa, second-to-last layer (Seed=573).

For the adaptation task, we found that pooling the language data into one large training data set featuring both Malayalam and Tamil improved the performance of our models. We believe that this occurred for two reasons. Firstly, the language-specific training data for both Tamil and Malayalam is incredibly small, and pooling it therefore enabled us to train our models on a substantially larger amount of data. Secondly, Tamil and Malayalam are very closely related languages, and the model may have been able to leverage salient linguistic features present in both languages in this particular data set which help in classifying sentiment. It is important to note, however, that Tamil and Malayalam use different orthographies, and as such pooling their data only increased performance in the adaptation task when using audio data, and made no impact when using only text data in the primary task.

Because there was only one HIGHLY NEGATIVE Malayalam instance, either it in the train data and not in the dev data, or it is in the dev data and not in the train data. This means that the maximum F_1 score for Malayalam dev data is 0.8, and the evaluation of the models on Malayalam dev data might not be indicative of the actual performance of our models.

6.1 Future Work

One clear route for future work is to focus on mitigating the limitations of such a small and imbalanced data set. A future approach to this task may, for instance, productively avail of resources specially developed for Tamil and Malayalam which do not yet exist.

If we had more time, we may have tried exploring data augmentation strategies more thoroughly. The sole augmentation procedure we used (besides simple random resampling) operated at the vector level (SMOTE). However, other data augmentation strategies that operate at the text or audio level may have improved our results. For example, we might have used abstractive summarization of existing samples to generate more samples, or replaced individual words with similar words on the basis of their word embedding vectors.

In pooling our language data, we did not convert the text data for Tamil and Malayalam into a script compatible with both, e.g., via Romanization. Doing so may have improved results when pooling, not just for audio data, but for text data as well.

Another route one could take would be to translate all the data into a high-resource language (e.g. English) and then use existing and proven NLP tools on the translations. This might provide better results if one was able to use NLP tools specializing in a certain language.

6.2 Error Analysis

Doing an extensive error analysis is difficult because of the size of the dataset. The very small number of test samples means it is difficult to nearly impossible to make conclusive statements about why a specific sample or set of samples was misclassified, because of the inherently high variance that goes with inference on a small test set.

Additionally, doing qualitative error analysis is also difficult, because none of the authors understand or speak either Tamil or Malayalam. Therefore finding generalizable and salient linguistic features which might guide the models to classify something incorrectly is not possible with the author's current resources.

7 Limitations and Ethical Considerations

7.1 Limitations

7.1.1 Data Set Imbalance

As we see in figure 2, the data set was both very small and highly imbalanced. Out of seventy Malayalam training samples, only one was of the 'Highly Negative' class, while thirty-six of them were 'Positive'. The fifty-four Tamil samples were likewise highly imbalanced, with 'Highly Negative' and 'Negative' both appearing only four times each and 'Positive' appearing thirty-three times. If there had been, say, one-hundred or a thousand times more of each class for both languages, this degree of imbalance would not have posed nearly so great of an issue; however, in conjunction with the sparsity of training samples, it likely negatively impacted the accuracy of our model to a great extent, as it is very difficult to extract meaningful and useful information from few instances — or just a single instance — of a class.

7.2 Ethical Considerations

7.2.1 Data Collection Methods

One must consider the way in which multimodal data is collected. Because multimodal data collection for sentiment analysis involves human participants, one must consider both the privacy and the informed consent of the human participants who engage in the data collection process.

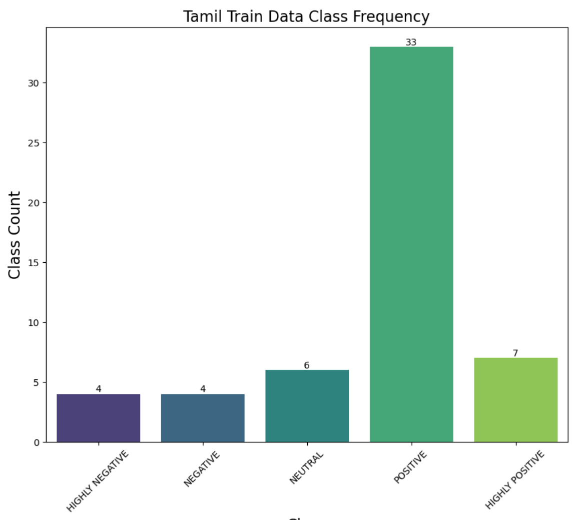
Privacy Unlike text data, where one can scrape the internet and grab data from an anonymous source, multimodal data uses both audio and video visuals, both which are means of identifying an individual. In the sentiment analysis task, using methods such as disguising a participants voice or hiding their face is undesirable, as both the voice as well as the facial expressions of the individual are informative and can help the model make decisions concerning the participant's sentiment. Therefore, even without associating a certain sample with a certain individual, the individual might still be identified by a family member, friend, or acquaintance.

For video in particular, researchers must also ensure that unwanted background is not visible, such as other people, commodities in the person's home, or items which might help identify the location of the participant. One method might be to blur the video background, but this can negatively influence the generalizability of machine learning models (Schuller et al., 2016).

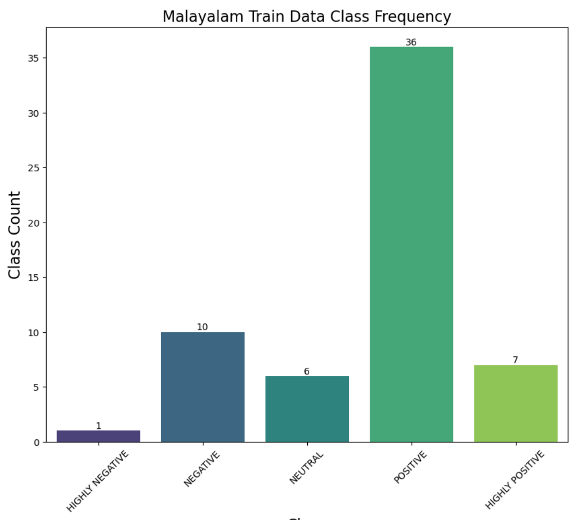
Although we did not avail of the video data with which we were provided, nevertheless we had access to it as participants in this task; the privacy concerns discussed above therefore still apply.

Informed Consent Because this task uses modes of data by which a participant can be identified afterwards and in which anonymity of participants is nearly impossible to ensure, researchers must acquire informed consent from all participants.

Given that people act differently when aware that they are being recorded, participants in a data collection process are ideally unaware of this fact



(a) Tamil (total: 54)



(b) Malayalam (total: 60)

Figure 2: Training Data Class Distribution

while samples which feature their data or information are being recorded, because this would obtain more organic responses. In other words, this requires recording someone first, and securing their informed consent at some later time. This method of data collection raises many important ethical and legal concerns (Schuller et al., 2016).

By contrast, if samples are recorded after informed consent takes place, participants may adjust their behavior accordingly and produce less genuine or spontaneous responses. In the context of sentiment analysis data, such data may not accurately capture a participant's true sentiment.

7.2.2 Sentiment Analysis Technology Applications

One concern held by Schuller et al. (2016) is that one would feel that the models used for sentiment analysis as objective, while a model is only as good as the data it was trained on. This has the implication that one must carefully consider the application of the sentiment analysis technology, and the effect that different tasks have when incorrect decisions are inevitably made by the model. For instance, an incorrect sentiment decision made by the personal recommendation system on Netflix would not have a very large implication, while an incorrect sentiment decision made during the President's State of the Union Address might have massive implications.

Another concern held by Schuller et al. (2016), is that computing models which perform sentiment analysis often make simplifying assumptions, thereby reducing the intricacies of human sentiment to a level which is not representative of true human sentiment.

8 Conclusion

We performed multi-modal sentiment analysis on text and audio data from the Dravidian languages Tamil and Malayalam. To perform sentiment analysis, we built an end-to-end system, trying different vectorization methods, resampling techniques, and classifiers. We evaluated the system's performance on the primary task using text data only, as well as the adaptation task using text and audio data in conjunction.

We created baseline systems using TF-IDF vectors and a Multinomial Naive Bayes classifier with no resampling, as well as a fine-tuned classification head from XLM-RoBERTa. We found that these baseline models performed no better than a majority class classifier.

We improved our performance using resampling techniques such as SMOTE, and we found that the best method was using the combination of [insert vectorization method], [insert resampling technique], and [insert classifier].

As discussed elsewhere in this paper, we found that significant challenges stemming from the small size of our data set and its highly imbalanced distribution of classes proved largely insurmountable in improving the success of our models.

References

Rosana Ardila, Megan Branson, Kelly Davis, Michael Kohler, Josh Meyer, Michael Henretty, Reuben Morais, Lindsay Saunders, Francis Tyers, and Gregor Weber. 2020. Common voice: A massively-multilingual speech corpus. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 4218–4222, Marseille, France. European Language Resources Association.

Premjith B., Sowmya V., Joyithish Lal. G., Bharathi Raja Chakravarthi, K. Nandhini, Rajeswari Natarajan, Abirami Murugappan, and Bharathi B. 2023. Multimodal Abusive Language Detection and Sentiment Analysis:DravidianLangTech@RANLP 2023.

Alexei Baevski, Henry Zhou, Abdelrahman Mohamed, and Michael Auli. 2020. wav2vec 2.0: A framework for self-supervised learning of speech representations.

Bharathi Raja Chakravarthi, Ruba Priyadharshini, Vigneshwaran Muralidaran, Shardul Suryawanshi, Navya Jose, Elizabeth Sherly, and John P. McCrae. 2021. Overview of the track on sentiment analysis for dravidian languages in code-mixed text. In *Proceedings of the 12th Annual Meeting of the Forum for Information Retrieval Evaluation*, FIRE '20, page 21–24, New York, NY, USA. Association for Computing Machinery.

Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. 2002. Smote: Synthetic minority over-sampling technique. *J. Artif. Int. Res.*, 16(1):321–357.

Alexis Conneau, Alexei Baevski, Ronan Collobert, Abdelrahman Mohamed, and Michael Auli. 2020. Unsupervised cross-lingual representation learning for speech recognition.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Unsupervised cross-lingual representation learning at scale. *CoRR*, abs/1911.02116.

Jingfeng Cui, Zhaoxia Wang, Seng-Beng Ho, and Erik Cambria. 2023. Survey on sentiment analysis: evolution of research methods and topics. *Artificial Intelligence Review*.

Sumanth Doddapaneni, Rahul Aralikatte, Gowtham Ramesh, Shreyansh Goyal, Mitesh M. Khapra, Anoop Kunchukuttan, and Pratyush Kumar. 2022. Indicxtreme: A multi-task benchmark for evaluating indic languages. *ArXiv*, abs/2212.05409.

Olivier Habimana, Yuhua Li, Ruixuan Li, Xiwu Gu, and Ge Yu. 2019. Sentiment analysis using deep learning approaches: an overview. *Science China Information Sciences*, 63(1):111102.

Divyanshu Kakwani, Anoop Kunchukuttan, Satish Golla, Gokul N.C., Avik Bhattacharyya, Mitesh M. Khapra, and Pratyush Kumar. 2020. IndicNLPSuite: Monolingual Corpora, Evaluation Benchmarks and Pre-trained Multilingual Language Models for Indian Languages. In *Findings of EMNLP*.

Taku Kudo and John Richardson. 2018. SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing. *CoRR*, abs/1808.06226.

Loren Lugosch, Tatiana Likhomanenko, Gabriel Synnaeve, and Ronan Collobert. 2022a. Pseudo-labeling for massively multilingual speech recognition. *ICASSP*.

Loren Lugosch, Tatiana Likhomanenko, Gabriel Synnaeve, and Ronan Collobert. 2022b. Pseudo-labeling for massively multilingual speech recognition.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Björn Schuller, Jean-Gabriel Ganascia, and Laurence Devillers. 2016. Multimodal sentiment analysis in the wild: Ethical considerations on data collection, annotation, and exploitation. In *Proceedings of the 1st International Workshop on ETHics In Corpus Collection, Annotation and Application (ETHI-CA2 2016), satellite of the 10th Language Resources and Evaluation Conference (LREC 2016)*, pages 29–34.

Chengyi Wang, Yu Wu, Yao Qian, Kenichi Kumatani, Shujie Liu, Furu Wei, Michael Zeng, and Xuedong Huang. 2021. Unispeech: Unified speech representation learning with labeled and unlabeled data.

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*, pages 38–45, Online. Association for Computational Linguistics.