Multimodal Sentiment Analysis of Tamil and Malayalam

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

We present several models for sentiment analysis of multimodal movie reviews in Tamil and Malayalam into 5 separate classes: highly negative, negative, neutral, positive, and highly positive, based on the shared task, ”Multimodal Abusive Language Detection and Sentiment Analysis” at RANLP-2023. We use transformer language models to build text and audio embeddings and then compare the performance of multiple classifier models trained on these embeddings: a Multinomial Naive Bayes baseline, a Logistic Regression, a Random Forest, and an SVM. To account for class imbalance, we use both naive resampling and SMOTE. We found that without resampling, the baseline models have the same performance as a naive Majority Class Classifier. However, with resampling, logistic regression and random forest both demonstrate gains over the baseline. In the shared task, our best-performing model outperformed all ranked models with a macro-F_1 of 0.29 for Tamil and of 0.28 for Malayalam. Nevertheless, we found that this result was not stable across experimental random seeds.

1 Introduction

Sentiment analysis is the application of natural language processing techniques to identify, quantify, and examine the subjective attitudes and affective content of language. It has a rich history and many methods have been used in the past (Cui et al., 2023). While sentiment analysis tasks most frequently occur in the text domain, 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. Sentiment analysis models trained for such contexts must reflect features of all modalities concerned. Practically speaking, this task can be constructed in many ways: as a binary classification task (i.e. categorizing language into either Positive or Negative sentiment), an ordinal regression problem, or a multi-class classification problem. In our case, the task was the latter. We are classifying multimodal (text, audio, and video) data in Tamil and Malayalam into 5 separate sentiment classes: highly negative, negative, neutral, positive, and highly positive (B et al., 2023).

1.1 Motivation

Although Dravidian languages, such as Telugu, Tamil, Kannada, or Malayalam, are spoken by more than 250 million people, very few natural language processing resources exist for them. This paper considers data in two Dravidian languages, Tamil and Malayalam. The agglutination and high degree of morphological complexity exhibited by both languages present 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 Dravidian languages, morphologically-complex languages, or low-resource languages more generally.

2 Related Work

Previous work on sentiment analysis on Dravidian languages was done in last year's shared task where only one team submitted their sentiment analysis project (Premjith et al., 2022). Also, previous work on sentiment analysis on Dravidian languages has used code-switched data (English

Third Workshop on Speech and Language Technologies for Dravidian Languages, pages 250–257, Varna, Bulgaria, Sep 7, 2023.
https://doi.org/10.26615/978-954-452-085-4_037

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

Text Vectorization Synthesizing text documents into feature vectors for text classification is a difficult problem; common methods (Rani et al., 2022; Anita and Shashi, 2019) for doing so include Bag-of-Words, N-Gram, TF-IDF, Word2Vec (Mikolov et al., 2013), GloVE (Pennington et al., 2014), and Sentence2Vec/Doc2Vec (Le and Mikolov, 2014). Previous work has used features present in the orthography of low resource languages, especially Dravidan languages (Chakravarthi et al., 2021b). Using pre-trained ELMo (Peters et al., 2018) or BERT (Devlin et al., 2019) word embeddings has become increasingly popular, since they better encode semantic information (Anita and Shashi, 2019). Sun and Zhou (2020) uses the last three hidden layers of the pretrained multilingual model XLM-RoBERTa (Conneau et al., 2019) in conjunction with a convolutional neural network (CNN) to encode Tamil-English, as well as Malayalam-English code-mixed data into feature vectors.

Audio Vectorization Speech technologies have progressed a lot in the recent years, especially in multilingual and low-resource areas. The introduction of pseudo-labeling has improved the efficiency of semi-supervised deep neural network learning (Lee et al., 2013). Multitask learning approaches in conjunction with deep neural networks has enabled improvements in speech recognition for low-resource languages (Chen and Mak, 2015). Wav2Vec2 utilizes transformer-based unsupervised pre-training (Jiang et al., 2019) which works well with little training data. It also does this pre-training via masked reconstruction loss which has benefits similar to data augmentation methods (Wang et al., 2020). M-CTC-T, XLS-R (Cross-lingual Speech Representations), and UniSpeech, on the other hand, make use of supervised/self-supervised pre-training methods which often allow for better accuracy (Baevski et al., 2019).

3 Dataset 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 (Chakravarthi et al., 2021c). Each sample consists of an audio file of speech and a (sometimes partial) transcript of it. (They have also provided video files which we do not use.) Every sample is a movie review collected from YouTube and labeled by human annotators.

They have also 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 combine train and dev data and use k-fold validation.

4 Methodology

4.1 System Overview

Our system pipeline has four main stages (see Figure 1): 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. 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 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, 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.

For the audio data, 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., 2022) using HuggingFace (Wolf et al., 2020). We then pass the output of the feature extractors into the three different pretrained models: XLS-R (Conneau et al., 2019), UniSpeech (Wang et al., 2021), and SpeechBrain (Lugosch et al., 2022). To extract document feature vectors, we then perform an element-wise average of the second to last hidden layer of the chosen pretrained model.

Third, we do resampling, using two methods, namely random resampling with replacement as

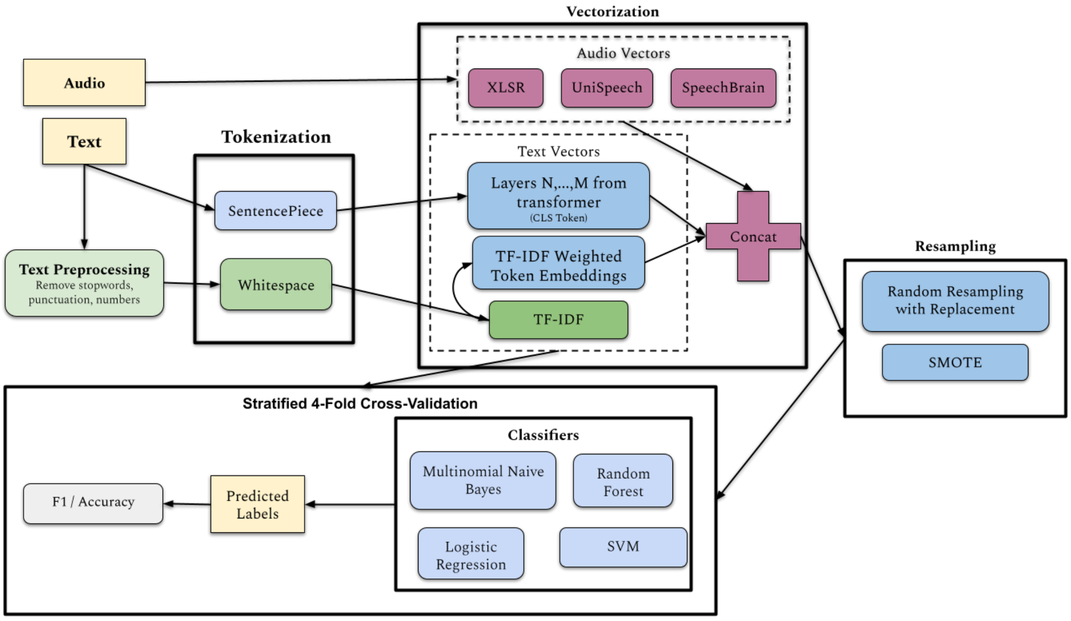


Figure 1: End-to-End System Architecture

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 first train various models using the training data, then use these 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.

4.2 Language Pooling

To create more training data, we pooled the data for Tamil and Malayalam together to create a new larger training data set and with which we trained a single model. Doing so produced better results than for language-specific models.

4.3 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`[1] 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.

4.4 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`.[2]

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:

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

$$\text{tf}(t, d) = \text{count}(t) \in d \tag{2}$$

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

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

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

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

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 we present these results below.

TF-IDF Weighted XLM-RoBERTa 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 4.3.

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[3], $\mathrm{tf\_idf}(t_i, d)$, corresponding to each token t_i in document d. Then the document feature vector $\vec{f_d}$ for document d is exactly:

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

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

4.5 Audio Vectorization

XLS-R For XLS-R we used Facebook's XLSR-Wav2Vec2 pretrained model `facebook/wav2vec2-large-xlsr-53`[4]. This model builds on and shares the same architecture as Wav2Vec2. This model was trained on CommonVoice, Babel, and Multilingual LibriSpeech (MLS) (53 total languages), including Tamil but not Malayalam.

UniSpeech For UniSpeech, we used Microsoft's pretrained model `microsoft/unispeech-large-multi-lingual-1500h-cv`[5]. This model builds on and shares the same architecture as Wav2Vec2. This model was trained on languages from CommonVoice, not including Tamil and Malayalam. 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 pretrained model `speechbrain/m-ctc-t-large`[6]. This model builds on and shares the same architecture as M-CTC-T. This model was trained on all languages from CommonVoice 6.1 (Ardila et al., 2020) and VoxPopuli (79 total languages), including Tamil but not Malayalam.

4.6 Data Augmentation

SMOTE For Tamil, we also resampled the data using SMOTE 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.

4.7 Ridge Regression Feature Selection

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 strength hyperparameter (α) of 0.30. In all experimental settings, this reduced the number of features by between 35-75%, thereby decreasing the immense feature sparsity that we were faced with in this task.

4.8 Classifiers

Multinomial Naive Bayes For our baseline, we trained a multinomial Naive Bayes classifier on the TF-IDF feature vectors from section 4.4. To run Naive Bayes, we used the `sklearn` library (Pedregosa et al., 2011).

[3] As calculated in section 4.3

[4] HF: wav2vec2-large-xlsr-53

[5] HF: UniSpeech

[6] HF: m-ctc-t-large

	Tamil				Malayalam				Tamil+Malayalam			
	Dev		Test		Dev		Test		Dev		Test	
	Acc	F_1	Acc	F_1	Acc	F_1	Acc	F_1	Acc	F_1	Acc	F_1
Baseline (Majority classifier)	0.61	0.15	0.30	0.09	0.60	0.15	0.50	0.13	0.61	0.15	0.40	0.11
Text Only (5-seed average)	0.28	0.26	0.35	**0.19**	0.53	0.40	0.30	**0.14**	–	–	–	–
Text Only (seed=573)	0.26	0.26	0.50	**0.29**	0.40	0.31	0.40	**0.28**	–	–	–	–
Text+Audio (5-seed average)	–	–	0.50	**0.13**	–	–	0.61	**0.15**	0.60	0.35	0.56	0.14

Table 1: Results on the test set comparing our baselines (equivalent performance to a majority vote classifier), and our best systems using text as well as text and audio data. The "Dev" columns refer are the averages of four-fold cross-validation in the train set, while the test columns are calculated based on inference over the test set. See Appendix A.1 for hyperparameters.

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, they never outperformed it.

5 Results

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 Multinomial Naive Bayes as presented in section 4.8.

We have selected the best-performing model in terms of the pooled dev macro-F_1 score, training just on text data, as well as training on text in conjunction with audio data. For this model, we report the validation pooled F_1 scores and accuracy averaged across five seeds.[7] For the test data, we report the same metrics averaged across the same seeds in table 1.

We also report the dev and test results of the best model for any specific seed in terms of F_1 score on the test data. This "best model" was trained with only the text modality (across the board, the text audio models did worse due to greater feature sparsity). Note that this single seed happened to perform especially well, but is not representative of overall model performance.

6 Discussion

6.1 Data Set Imbalance

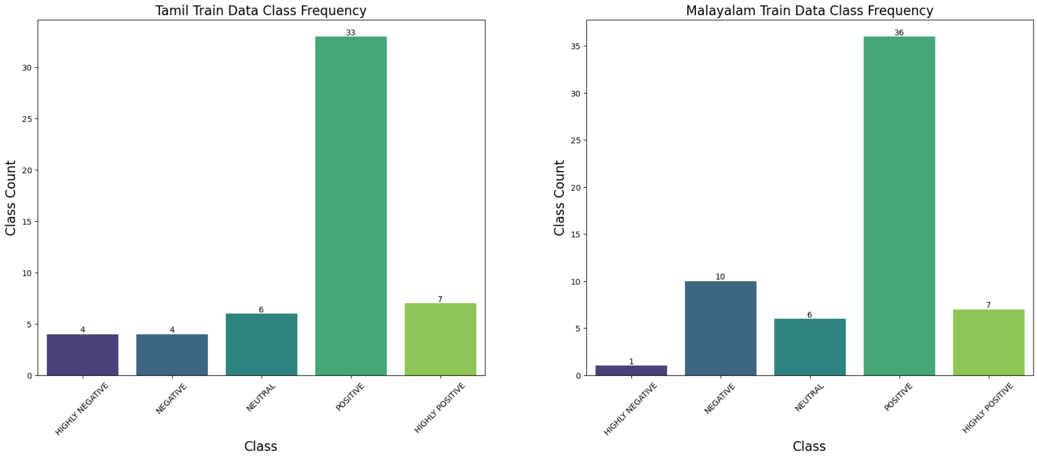
As we see in figure 2, the data set was both very small and highly imbalanced. Out of 70 Malayalam training samples, only one was of the 'Highly Negative' class, while thirty-six of them were 'Positive'. The fifty-four Tamil samples were likewise highly imbalanced, with 'Highly Negative' and 'Negative' both appearing only four times each and 'Positive' appearing thirty-three times. This imbalance could have negatively impacted the accuracy of our model to a great extent.

6.2 Discussion of Results

Without resampling, our baseline 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 instances are POSITIVE.

Because our models were trained on so few samples, the use of too many features resulted in over-

[7] 0, 42, 100, 573, 2023.



(a) Tamil (total: 54)

(b) Malayalam (total: 60)

Figure 2: Training Data Class Distribution

fitting; for this reason, models with fewer features generally performed better. Because there are only ten test instances 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 setting, but we do not feel that these are necessarily indicative of our model performance in general.

As we see in Table 1, when including text and audio data, 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, pooling 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 to better classify sentiment. However, Tamil and Malayalam use different orthographies, so pooling their data only increased performance when using text and audio, and made no impact when using only text data.

Because there was only one HIGHLY NEGATIVE Malayalam instance, it cannot be in the train and dev data simultaneously. 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.

6.3 Future Work

One clear route for future work is to focus on mitigating the limitations of such a small and imbalanced data set. For example, one may conduct a more thorough exploration of data augmentation strategies. 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 and/or audio level may improve results.

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.

7 Conclusion

We performed multimodal 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 using only text data, as well as jointly using text and audio data.

We created baseline systems using TF-IDF vectors and a Multinomial Naive Bayes classifier with no resampling and found that these baseline models performed no better than a majority class 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. We were not able to be formally ranked however our model did outperform the other ranked models.

References

Kumari S. Anita and Mogalla Shashi. 2019. Vectorization of text documents for identifying unifiable news articles. *International Journal of Advanced Computer Science and Applications*, 10(7). Copyright - © 2019. This work is licensed under https://creativecommons.org/licenses/by/4.0/ (the "License"). Notwithstanding the ProQuest Terms and Conditions, you may use this content in accordance with the terms of the License; Last updated - 2023-05-01.

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, Jyothish Lal G, Bharathi Raja Chakravarthi, Nandhini K, Rajeswari Natarajan, Abirami Murugappan, Bharathi B, Kaushik M, Prasanth S.N, Aswin Raj R, and Vijai Simmon S. 2023. Findings of the Multimodal Abusive Language Detection and Sentiment Analysis in Dravidian Languages @ dravidianlangtech-ranlp 2023. In *Proceedings of the Third Workshop on Speech and Language Technologies for Dravidian Languages DravidianLangTech 2023*. Recent Advances in Natural Language Processing.

Alexei Baevski, Michael Auli, and Abdelrahman Mohamed. 2019. Effectiveness of self-supervised pretraining for speech recognition. *arXiv preprint arXiv:1911.03912*.

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. 2021a. 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.

Bharathi Raja Chakravarthi, Priya Rani, Mihael Arcan, and John P. McCrae. 2021b. A survey of orthographic information in machine translation. *SN Computer Science*, 2(4):330.

Bharathi Raja Chakravarthi, KP Soman, Rahul Ponnusamy, Prasanna Kumar Kumaresan, Kingston Pal Thamburaj, John P McCrae, et al. 2021c. Dravidianmultimodality: A dataset for multi-modal sentiment analysis in tamil and malayalam. *arXiv preprint arXiv:2106.04853*.

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.

Dongpeng Chen and Brian Kan-Wing Mak. 2015. Multitask learning of deep neural networks for low-resource speech recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 23(7):1172–1183.

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*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding.

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.

Dongwei Jiang, Xiaoning Lei, Wubo Li, Ne Luo, Yuxuan Hu, Wei Zou, and Xiangang Li. 2019. Improving transformer-based speech recognition using unsupervised pre-training. *arXiv preprint arXiv:1910.09932*.

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.

Quoc V. Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents.

Dong-Hyun Lee et al. 2013. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on challenges in representation learning, ICML*, volume 3, page 896. Atlanta.

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

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space.

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.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations.

B Premjith, Bharathi Raja Chakravarthi, Malliga Subramanian, B Bharathi, Soman Kp, V Dhanalakshmi, K Sreelakshmi, Arunaggiri Pandian, and Prasanna Kumaresan. 2022. Findings of the shared task on multimodal sentiment analysis and troll meme classification in dravidian languages. In *Proceedings of the Second Workshop on Speech and Language Technologies for Dravidian Languages*, pages 254–260.

Deepa Rani, Rajeev Kumar, and Naveen Chauhan. 2022. Study and comparision of vectorization techniques used in text classification. In *2022 13th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, pages 1–6.

Ruijie Sun and Xiaobing Zhou. 2020. Srj @ dravidian-codemix-fire2020: Automatic classification and identification sentiment in code-mixed text. In *Fire*.

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.

Weiran Wang, Qingming Tang, and Karen Livescu. 2020. Unsupervised pre-training of bidirectional speech encoders via masked reconstruction. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6889–6893. IEEE.

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.

A Appendix

A.1 Model Hyperparameters

Text Only: Logistic Regression ($C = 5.0$), Ridge Regression feature selection, random resampling with replacement; text vectorization: second-to-last layer.

Text+Audio: Pooled language training, Logistic Regression ($C = 5.0$), Ridge Regression feature selection, random resampling with replacement; text vectorization: second-to-last layer; audio vectorization: XLSR, third-to-last layer.