Sarcasm Identification in Dravidian Languages Tamil and Malayalam

Poorvi Shetty[1]

[1]JSS Science and Technology University, Mysore, India

Abstract

Sarcasm poses a formidable challenge to sentiment analysis systems as it conveys opinions indirectly, often diverging from their literal interpretation. The escalating demand for sarcasm and sentiment detection in social media, particularly within code-mixed content in Dravidian languages, underscores the significance of this research. Code-mixing is widespread within multilingual communities, and code-mixed texts frequently incorporate non-native scripts. The primary objective of this study is to discern sarcasm within a dataset comprising comments and posts in Tamil-English and Malayalam-English, sourced from social media platforms. Our research investigates combinations of various embeddings and models, yielding promising results. Notably, the top-performing system i.e., TF-IDF Vectorizer coupled with a stacking classifier (composed of Linear Support Vector Classifier, Random Forest Model, and K-Nearest Neighbors Model, with Logistic Regression serving as the meta classifier) achieved a weighted average F1 score of 0.79 for Tamil and 0.78 for Malayalam, showcasing its effectiveness in sarcasm and sentiment analysis within code-mixed content. The system proposed ranked 2nd for Tamil and 3rd for Malayalam in the shared task.

Keywords

sarcasm identification, code-mixing, multilingual communication, word embeddings, classifiers, tamil, malayalam,

1. Introduction

Detecting sarcasm poses a significant challenge for sentiment analysis systems because it involves conveying an opinion indirectly, often with a meaning that diverges from the literal interpretation. As the demand for sarcasm and sentiment detection on social media texts, particularly in Dravidian languages, continues to rise, addressing this challenge becomes crucial.

Tamil is a Dravidian language spoken as the native tongue by more than 78 million people worldwide, with a presence in countries including India, Sri Lanka, Malaysia, Singapore, and Mauritius [1]. Meanwhile, Malayalam serves as the official language in Kerala and is spoken by over 37 million people globally. The written tradition of Malayalam has a rich historical legacy spanning centuries [2]. Despite their significant speaker populations and extensive historical backgrounds, these languages face a shortage of resources in the realm of Natural Language Processing (NLP).

Forum for Information Retrieval Evaluation, December 15-18, 2023, India

poorvishetty1202@gmail.com (P. Shetty)

0009-0004-2243-5176 (P. Shetty)

Sarcasm detection in Natural Language Processing (NLP) is crucial for improving machine-human communication. It involves identifying contradictions within sarcastic statements, where the intended meaning contradicts the literal interpretation. Accurate sarcasm detection enhances various applications, including virtual assistants, sentiment analysis, and social media monitoring. It enables machines to better understand the subtleties of human language and emotions, making interactions more natural and insightful [3].

Code-mixing is a prevalent linguistic phenomenon in multilingual communities, and code-mixed texts are frequently composed using non-native scripts. In this context, our research aims to discern sarcasm and determine the sentiment polarity within a dataset of comments and posts that are code-mixed in Tamil-English and Malayalam-English, sourced from various social media platforms. This was part of the Sarcasm Identification of Dravidian Languages (Malayalam and Tamil) in DravidianCodeMix 2023 (DravidianCodeMix) shared task [4].

2. Related Work

Several advancements have been made in the field of sarcasm detection by various researchers. Joshi et al. [5] explored three key developments in their research, including the use of semi-supervised pattern extraction to unveil implicit sentiment, the incorporation of hashtag-based supervision, and the integration of contextual information beyond the target text. Their comprehensive investigation encompassed datasets, methodologies, emerging trends, and challenges in the domain of sarcasm detection. Elgabry et al. [6] introduced an innovative approach to enhance Arabic sarcasm detection, which involved data augmentation, contextual word embeddings, and a random forests model, achieving superior performance in identifying sarcasm.

Apon et al. [7] found that the Random Forest classifier produced the most favorable results when applied to their original Bengali sarcasm detection dataset. Ravikiran et al. [8] established an experimental benchmark using state-of-the-art multilingual language models like BERT, DistillBERT, and XLM-RoBERTA for identifying offensive language spans in Dravidian languages. Kumar et al. [9] presented a hybrid deep learning model trained with both word and emoji embeddings to identify sarcasm, emphasizing the significance of incorporating emojis in sarcasm detection. Eke et al. [10] conducted an investigation into the GloVe word vector model, revealing that it not only captures semantics and grammar but also retains contextual information and global corpus statistics. In a comprehensive experimental analysis, Onan [11] explored six subsets of Twitter messages, varying in size from 5,000 to 30,000, and found that employing topic-enriched word embedding schemes alongside conventional feature sets holds promise for sarcasm identification.

3. Dataset Description

The dataset [12] [13] used comprised of YouTube video comments in both Tamil-English and Malayalam-English. This dataset encompasses a wide range of code-mixed sentences. The comments predominantly appear in two forms: written in the native script and the Roman script, featuring either Tamil/Malayalam grammar and English vocabulary or conversely, English grammar with Tamil/Malayalam vocabulary. Additionally, some comments are composed in

Table 1
Training dataset statistics

Language	Non-Sarcastic	Sarcastic
Tamil	24805	8990
Malayalam	12225	2847

Tamil/Malayalam script, interspersed with English expressions. 1 has the statistics for both languages. Each entry can be labelled as 'Non-Sarcastic' or 'Sarcastic'. We notice that 'Non-Sarcastic' entries significantly outnumber 'Sarcastic' entries, indicating class imbalance.

4. Data Preprocessing

In the preprocessing stage, several steps were applied to prepare the data for machine learning, namely punctuation removal, numeric characters removal, whitespace cleanup and label encoding of target series. By implementing these preprocessing steps, the text data was cleaned, standardized, and made ready for further feature extraction and model training.

In the process of text data preprocessing and feature extraction, four different methods were considered: TFIDF Vectorizer, CountVectorizer, Word2Vec, and FastText.

4.1. TF-IDF Vectorizer

This method calculates the Term Frequency-Inverse Document Frequency (TF-IDF) scores for each word in the corpus, reflecting the importance of words within individual documents and across the entire dataset. It is particularly useful for capturing the uniqueness and significance of words in a document [14]. Implementation was done using the Scikit-Learn library.

4.2. CountVectorizer

Unlike TFIDF, CountVectorizer simply counts the frequency of each word in the text. This method is straightforward and efficient, making it a good choice when you want to consider the raw word counts as features [15]. Implementation was done using the Scikit-Learn library.

4.3. Word2Vec

Word2Vec is a deep learning-based technique that learns word embeddings by predicting the context of words in a large corpus. It captures semantic relationships between words and represents them as dense vectors in a continuous vector space. This method is effective at capturing semantic meanings and word associations [16]. Implementation was done using the Gensim library.

4.4. FastText

FastText is an extension of Word2Vec that also considers sub-word information. It breaks words into smaller sub-word units (n-grams) and learns embeddings for these units as well as for full words. This approach is beneficial for handling out-of-vocabulary words and capturing morphological information [17]. Implementation was done using the fastText library.

5. Models Used

Diverse machine learning models were employed for the task, aiming for a thorough comparison and selection of the most effective approaches. The Scikit-Learn library was employed for model implementation. Default hyperparameters from the library were used for all models. TF-IDF vectorizer was used as it gave the best results.

5.1. Random Forest

Random Forest is an ensemble learning method based on decision trees [18]. It builds multiple decision trees and combines their predictions to improve accuracy and reduce overfitting. Gini criterion was used, with n_estimators as 100 and min_samples_split value as 2. Max_depth parameter was set to None.

5.2. Logistic Regression

Logistic Regression is a simple yet effective linear classification algorithm used for binary and multi-class classification [19]. It models the relationship between the dependent variable and one or more independent variables using the logistic function, which transforms linear combinations into probabilities. L2 penalty term was utilised, C value was set to default value of 1.0 and lbfgs solver was used.

5.3. Linear Support Vector Classifier (LinearSVC)

LinearSVC is a linear classification algorithm that aims to find the hyperplane that best separates classes in high-dimensional space [20]. It's particularly useful when the data is linearly separable and can handle large datasets efficiently. Here too, L2 penalty term was used, the loss function used was squared_hinge and C value was 1.0.

5.4. Decision Tree

Decision Trees are non-linear models that partition the data into subsets based on feature values [21]. They are used for both classification and regression tasks and are interpretable. However, they can be prone to overfitting. Similar to Decion Tree, gini criterion was used, with n_estimators as 100 and min_samples_split value as 2. Max_depth parameter was set to None.

5.5. K-Nearest Neighbors (KNN)

KNN is a simple instance-based learning algorithm used for classification and regression [22]. It assigns a class label to a data point based on the majority class among its k-nearest neighbors in the feature space. The n_neighbours value was set to 5 with uniform weights. The algorithm parameter was set to auto. Minkowski metric was used.

5.6. AdaBoost

AdaBoost is an ensemble learning method that combines the predictions of multiple weak learners (often decision trees) to create a strong classifier [23]. It assigns different weights to data points and focuses on those that are misclassified in previous iterations. The n_estimators value is set to 50.

5.7. One-Versus-Rest (OneVsRest)

This is a technique for handling multi-class classification problems by training multiple binary classifiers, one for each class, and then combining their results [24]. Logistic Regression is often used as the base binary classifier. The parameters for the LR model within are the same as mentioned above.

5.8. Gradient Boosting

Gradient Boosting is an ensemble learning method that builds an additive model by iteratively training weak learners and adjusting their weights based on the errors of the previous iterations [25]. It's known for its high predictive accuracy. The min_samples_split value is set to 2, and log_loss is used as the loss function.

5.9. Stacking Classifier

Stacking is an ensemble technique that combines multiple base classifiers with a meta-classifier to improve predictive performance [26]. In this case, LinearSVC and RandomForest were used as base classifiers, and logistic regression as the meta-classifier. The parameters for the models within are the same as mentioned above.

5.10. Voting Classifier

Voting is another ensemble technique that combines the predictions of multiple classifiers by taking a majority vote (hard voting) or weighted vote (soft voting) [27]. In this study, Logistic Regression, Random Forest, and Support Vector Classifier (SVC) were combined. The parameters for the models within are the same as mentioned above.

5.11. Bagging Classifier

Bagging is an ensemble technique that trains multiple instances of the same base classifier on different subsets of the data and aggregates their predictions [28]. Here, KNN model was used

Table 2
F1 score of models on the Tamil dataset with various embeddings

Classifier	CountVectorizer	Word2Vec	FastText	TF-IDF
Random Forest	0.74	0.61	0.73	0.75
Logistic Regression	0.78	0.61	0.73	0.77
Linear SVC	0.77	0.61	0.73	0.78
Decision Tree	0.74	0.62	0.66	0.73
KNN	0.65	0.61	0.73	0.63
AdaBoost	0.72	0.61	0.73	0.72
One Vs Rest	0.78	0.62	0.73	0.77
Gradient Boost	0.69	0.61	0.74	0.69
Stacking	0.77	0.63	0.77	**0.79**
Voting	0.65	0.61	0.75	0.77
Bagging	0.65	0.61	0.74	0.63

Table 3
F1 score of models on the Malayalam dataset with various embeddings

Classifier	CountVectorizer	Word2Vec	FastText	TF-IDF
Random Forest	0.75	0.73	0.75	0.75
Logistic Regression	0.77	0.73	0.76	0.75
Linear SVC	0.77	0.73	0.76	0.78
Decision Tree	0.76	0.72	0.73	0.76
KNN	0.74	0.72	0.77	0.73
AdaBoost	0.75	0.73	0.78	0.75
One Vs Rest	0.77	0.73	0.76	0.75
Gradient Boost	0.75	0.74	0.77	0.74
Stacking	0.77	0.73	0.78	**0.78**
Voting	0.76	0.72	0.76	0.75
Bagging	0.73	0.72	0.78	0.73

as the base classifier, and ten instances were created. The parameters for the KNN within are the same as mentioned above.

6. Experiments and Results

Our research methodology involved data preprocessing phase as outlined in the earlier section. Subsequently, we conducted an exhaustive exploration of various word embedding techniques (TF-IDF, Count, Word2Vec, FastText), pairing each with various models to evaluate their performance. We aimed to assess the effectiveness of these combinations, and we carefully recorded the weighted average F1 scores as a key performance metric.

We use weighted average F1 scores (refer 2 3) to evaluate the performance of the models. It is a good metric for evaluating models in a class-imbalanced binary classification problem because it considers both precision and recall, making it robust to situations where one class

is much smaller than the other. It provides a balanced assessment of a model's performance by penalizing false positives and false negatives, making it more informative than accuracy in such scenarios.

The best result for both Tamil and Malayalam was achieved using TFIDFVectorizer to convert text data into numerical form and a stacking classifier combining LinearSVC, RandomForest, and KNN as base models, with logistic regression as the meta classifier. TFIDFVectorizer is used to preserve semantic information in the model, enabling the extraction of meaningful text features for better discrimination between classes. Additionally, the choice of a stacking classifier leverages the strengths of diverse base models, effectively combining their predictions and enhancing overall predictive power. Logistic Regression, though simple, proves effective in aggregating base model predictions, promoting balanced and well-regulated final predictions, which helps mitigate overfitting and bias issues, resulting in improved classification accuracy. The weighted average F1 score for this configuration was 0.79 for Tamil and 0.78 for Malayalam.

7. Conclusion

In this research study, we addressed the task of sarcasm identification in Dravidian languages, specifically Tamil and Malayalam. To achieve this, we explored various combinations of word embeddings and classifiers, systematically analyzing their effectiveness in the context of the problem. Notably, the best-performing model emerged as a result of our experimentation, featuring the TFIDFVectorizer for text representation and a powerful ensemble stacking classifier composed of LinearSVC, RandomForest, and K-Nearest Neighbors, with Logistic Regression serving as the meta classifier.

References

[1] P. Krishnamurthy, K. Sarveswaran, Towards building a modern written Tamil treebank, in: Proceedings of the 20th International Workshop on Treebanks and Linguistic Theories (TLT, SyntaxFest 2021), Association for Computational Linguistics, Sofia, Bulgaria, 2021, pp. 61–68. URL: https://aclanthology.org/2021.tlt-1.6.

[2] A. Rojan, E. Alias, G. M. Rajan, J. Mathew, D. Sudarsan, Natural language processing based text imputation for malayalam corpora, in: 2020 International Conference on Electronics and Sustainable Communication Systems (ICESC), 2020, pp. 161–165. doi:10.1109/ICESC48915.2020.9156036.

[3] A. A, S. G, S. H. R, M. Upadhyaya, A. P. Ray, M. T. C, Sarcasm detection in natural language processing, Materials Today: Proceedings 37 (2021) 3324–3331. URL: https://doi.org/10.1016/j.matpr.2020.09.124. doi:10.1016/j.matpr.2020.09.124.

[4] B. R. Chakravarthi, N. Sripriya, B. Bharathi, K. Nandhini, S. Chinnaudayar 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.

[5] A. Joshi, P. Bhattacharyya, M. J. Carman, Automatic sarcasm detection: A survey, CoRR abs/1602.03426 (2016). URL: http://arxiv.org/abs/1602.03426. arXiv:1602.03426.

[6] H. Elgabry, S. Attia, A. Abdel-Rahman, A. Abdel-Ate, S. Girgis, A contextual word embedding for Arabic sarcasm detection with random forests, in: Proceedings of the Sixth Arabic Natural Language Processing Workshop, Association for Computational Linguistics, Kyiv, Ukraine (Virtual), 2021, pp. 340–344. URL: https://aclanthology.org/2021.wanlp-1.43.

[7] T. S. Apon, R. Anan, E. A. Modhu, A. Suter, I. J. Sneha, M. G. R. Alam, Banglasarc: A dataset for sarcasm detection, 2022. arXiv:2209.13461.

[8] M. Ravikiran, S. Annamalai, DOSA: Dravidian code-mixed offensive span identification dataset, in: Proceedings of the First Workshop on Speech and Language Technologies for Dravidian Languages, Association for Computational Linguistics, Kyiv, 2021, pp. 10–17. URL: https://aclanthology.org/2021.dravidianlangtech-1.2.

[9] A. Kumar, S. R. Sangwan, A. K. Singh, G. Wadhwa, Hybrid deep learning model for sarcasm detection in indian indigenous language using word-emoji embeddings, ACM Trans. Asian Low-Resour. Lang. Inf. Process. 22 (2023). URL: https://doi.org/10.1145/3519299. doi:10.1145/3519299.

[10] C. I. Eke, A. Norman, L. Shuib, F. B. Fatokun, I. Omame, The significance of global vectors representation in sarcasm analysis, in: 2020 International Conference in Mathematics, Computer Engineering and Computer Science (ICMCECS), 2020, pp. 1–7. doi:10.1109/ICMCECS47690.2020.246997.

[11] A. Onan, Topic-enriched word embeddings for sarcasm identification, in: R. Silhavy (Ed.), Software Engineering Methods in Intelligent Algorithms, Springer International Publishing, Cham, 2019, pp. 293–304.

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

[13] 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.

[14] J. E. Ramos, Using tf-idf to determine word relevance in document queries, 2003. URL: https://api.semanticscholar.org/CorpusID:14638345.

[15] O. Shahmirzadi, A. Lugowski, K. Younge, Text similarity in vector space models: A comparative study, 2018. arXiv:1810.00664.

[16] T. Mikolov, K. Chen, G. Corrado, J. Dean, Efficient estimation of word representations in vector space, 2013. arXiv:1301.3781.

[17] P. Bojanowski, E. Grave, A. Joulin, T. Mikolov, Enriching word vectors with subword information, 2017. arXiv:1607.04606.

[18] L. Breiman, Machine Learning 45 (2001) 5–32. URL: https://doi.org/10.1023/a:1010933404324. doi:10.1023/a:1010933404324.

[19] A. A. T. Fernandes, D. B. F. Filho, E. C. da Rocha, W. da Silva Nascimento, Read this paper if you want to learn logistic regression, Revista de Sociologia e Política 28 (2020). URL: https://doi.org/10.1590/1678-987320287406en. doi:10.1590/1678-987320287406en.

[20] S. Ghosh, A. Dasgupta, A. Swetapadma, A study on support vector machine based linear and non-linear pattern classification, in: 2019 International Conference on Intelligent

Sustainable Systems (ICISS), 2019, pp. 24–28. doi:10.1109/ISS1.2019.8908018.

[21] L. Rokach, O. Maimon, Decision trees, in: Data Mining and Knowledge Discovery Handbook, Springer-Verlag, ????, pp. 165–192. URL: https://doi.org/10.1007/0-387-25465-x_9. doi:10.1007/0-387-25465-x_9.

[22] K. Taunk, S. De, S. Verma, A. Swetapadma, A brief review of nearest neighbor algorithm for learning and classification, in: 2019 International Conference on Intelligent Computing and Control Systems (ICCS), 2019, pp. 1255–1260. doi:10.1109/ICCS45141.2019.9065747.

[23] R. Wang, AdaBoost for feature selection, classification and its relation with SVM, a review, Physics Procedia 25 (2012) 800–807. URL: https://doi.org/10.1016/j.phpro.2012.03.160. doi:10.1016/j.phpro.2012.03.160.

[24] Y. Alhessi, R. Wicentowski, SWATAC: A sentiment analyzer using one-vs-rest logistic regression, in: Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015), Association for Computational Linguistics, 2015. URL: https://doi.org/10.18653/v1/s15-2106. doi:10.18653/v1/s15-2106.

[25] Z. He, D. Lin, T. Lau, M. Wu, Gradient boosting machine: A survey, 2019. arXiv:1908.06951.

[26] S.-A. N. Alexandropoulos, C. K. Aridas, S. B. Kotsiantis, M. N. Vrahatis, Stacking strong ensembles of classifiers, in: IFIP Advances in Information and Communication Technology, Springer International Publishing, 2019, pp. 545–556. URL: https://doi.org/10.1007/978-3-030-19823-7_46. doi:10.1007/978-3-030-19823-7_46.

[27] S. Kumari, D. Kumar, M. Mittal, An ensemble approach for classification and prediction of diabetes mellitus using soft voting classifier, International Journal of Cognitive Computing in Engineering 2 (2021) 40–46. URL: https://doi.org/10.1016/j.ijcce.2021.01.001. doi:10.1016/j.ijcce.2021.01.001.

[28] P. Bühlmann, Bagging, boosting and ensemble methods, in: Handbook of Computational Statistics, Springer Berlin Heidelberg, 2011, pp. 985–1022. URL: https://doi.org/10.1007/978-3-642-21551-3_33. doi:10.1007/978-3-642-21551-3_33.