Detecting Sarcasm in Social Media Text Using Indic Transliteration and Machine Learning Techniques

Kogilavani S V[1], Malliga Subramnanian[1], Prenitha S P[2], Varshini S H*[3] and Arunachalam M[4]

Abstract

Sarcasm, which is frequently characterized by irony or ridicule, is a complicated phenomena in language when the intended meaning differs from the literal utterance. Sarcasm detection in text poses major hurdles to natural language processing (NLP), especially in social media situations. In contrast to conventional sentiment analysis, sarcasm frequently hides its true meaning beneath colloquial language, making it challenging to identify without taking into account the context and minute changes in meaning.Using transliterated datasets, this study focuses on sarcasm recognition in Indic languages, particularly Tamil and Malayalam. We improve the text's accessibility for machine learning models by transliterating the native scripts into standard forms. The efficacy of four models in identifying sarcasm was tested: Decision Tree (DT), K-Nearest Neighbors (KNN), Support Vector Machine (SVM), and Logistic Regression (LR). With an F1-score of 0.3710 and accuracy rates of 83.62% for Malayalam and 75.50% for Tamil, Logistic Regression was the model that performed the best overall.Our system demonstrated the resilience of our method by achieving a noteworthy rank 9 in a competitive sarcasm detection assignment.These findings highlight how crucial it is to combine customized machine learning models with transliteration in order to identify sarcasm in regional languages.

Keywords

Sarcasm detection, Transliteration, Natural Language Processing (NLP), Tamil, Malayalam, Code-mixed languages, Machine Learning, Logistic Regression, K-Nearest Neighbors (KNN), Support Vector Machine (SVM), Decision Tree (DT)

1. Introduction

The practice of translating a word or phrase from one writing system to another while preserving the original writing system's sounds or letters is known as transliteration.When used skillfully, transliterations and code-switching in languages such as Tamil and Malayalam can yield insightful linguistic information that enhances the ability to recognize sarcasm [1]. Although these components appear to add more labour to the task at hand, they actually create a more realistic representation of language by combining phonetic and cultural elements with local linguistic patterns using English script. Transliterations reveal contextual details that would otherwise be lost in translation since they accurately capture the sound of words in their native language.By incorporating advanced machine learning techniques, transliterations become a benefit rather than a disadvantage. Transliterations can be recognized and interpreted by machine learning algorithms, especially those built to handle multilingual data. This successfully bridges the gap between the original language and its English-script counterpart. These algorithms are able to identify patterns that disclose the speaker's intention, even when it is obscured by sarcastic tones, by examining both the transliterated text and its surface-level content. This is made possible by the fact that, provided phonetic consistencies are preserved, feature extraction models—like those trained on English data—can nevertheless extract semantic information from transliterated text. Text that has been transliterated into the Roman script adheres to a systematic, phonetically consistent sound mapping. Therefore, despite the script variation, feature extraction algorithms use these phonetic signals to identify well-known patterns, including sentiment changes or tonal nuances. Models are taught to identify characteristics such as contextual dependencies, n-grams, and word embeddings in transliteration cases, which preserve important linguistic elements of the source language. Furthermore, Both native and transliterated data are used to train embedding methods such as Word2Vec, which

Forum for Information Retrieval Evaluation, December 12-15, 2024, India

Kogilavanisv@gmail.com (K. S. V); mallinishanth72@gmail.com (M. Subramnanian); prenithasp.22aid@kongu.edu (P. S. P); varshinish.22aid@kongu.edu (V. S. H*); arunachalamm.22aid@kongu.edu (A. M)

capture both surface-level features and more profound semantic implications. remove a line from this and it should stil make sense

2. Literature Review

Sarcasm recognition in Indian languages has been a major focus of research. The study [2] on detecting sarcasm in Hindi used language-specific methods such as transliteration and tokenisation from the Indic NLP toolbox. Sarcasm detection

Language nuances and code switching cause problems in Hindi, as they do in other Indian languages. The study discovered that collecting features from Indic scripts using models like SVM and Random Forest can improve sarcasm detection by leveraging language-specific characteristics.

The author at [3] investigated a multimodal approach to sarcasm detection, using textual and visual data. This study discovered that sarcasm is frequently expressed through imagery, emoticons, and social media replies. By combining text-based characteristics with visual sentiment signals, their technique outperformed purely text-based models in accuracy. The use of multimodal approaches allows for a more full investigation of sarcasm, as it captures context that text alone may overlook.

Both the studies by [4, 5] used machine learning models including Decision Trees, SVM, and Neural Networks to identify sarcasm. The author used these models to anticipate sarcastic intent by analysing the text's syntactic and semantic components. In [5], the author applied neural networks to identify sarcasm across code-mixed languages, highlighting the significance of deep learning in difficult multilingual contexts.

The author [6] emphasised the role of transliteration in handling Indian scripts such as Tamil. Transliteration bridges the gap between native scripts and their Romanised counterparts, which is required for sarcasm detection in code-mixed datasets. Accurate transliteration not only protects phonetic integrity, but it also improves feature extraction for machine learning models by retaining important linguistic nuances. This, in turn, enhances the model's capacity to recognise sophisticated language elements like sentiment, intent, and sarcasm in multilingual settings. Transliteration also increases performance on NLP tasks such as named entity recognition (NER), tokenisation, and cross-lingual transfer learning by providing a consistent representation of words across languages and scripts. Transliteration has been shown in studies to reduce text ambiguity and improve overall classification accuracy, especially in code-mixed circumstances where Romanised and local characters are merged.

Finally, the author of [7] emphasised the role of transliteration in managing Indic scripts like Tamil. Transliteration bridges the gap between native scripts and their Romanised counterparts, which is required for sarcasm detection in code-mixed datasets. Their findings demonstrate that correct transliteration preserves phonetic integrity, allowing for superior feature extraction in machine learning models.

These researches emphasise the significance of dealing with linguistic diversity, exploiting multimodal inputs, and using advanced machine learning algorithms. Combining these approaches considerably increases model robustness and accuracy in identifying sarcasm in Indian languages.

3. Dataset Description

The dataset used in this research features mixed-language utterances that include Tamil-English and Malayalam-English combinations, categorized as either sarcastic or non-sarcastic. The dataset is available at: https://codalab.lisn.upsaclay.fr/competitions/19310#participate. Text samples in Malayalam and Tamil that have been separated into training, validation, and test subsets make up the sarcasm recognition dataset. There are 21,740 non-sarcastic and 7,830 sarcastic text in the training set and 4,630 non-sarcastic and 1,706 sarcastic text in the validation set for the Tamil dataset. 6,338 text samples make up the test set, which will be used for label-free evaluation. There are 10,689 non-sarcastic and 2,499 sarcastic text in the training set and 2,305 non-sarcastic and 521 sarcastic text in the validation set of the Malayalam dataset. The model's performance on unseen data will also be evaluated using the 2,826 unlabeled text samples in the test set. This dataset is set up to train and validate models for sarcasm detection in both languages.The dataset is an extensive collection for sarcasm identification in Tamil and Malayalam since it spans a wide range of situations and domains, from official writing to social media posts. The model's capacity to generalize well to new data is improved by the quantity of the dataset, which guarantees that it is trained on a variety of samples. Furthermore, a different validation set makes sure that the models are adjusted to prevent overfitting, which strengthens their resilience.

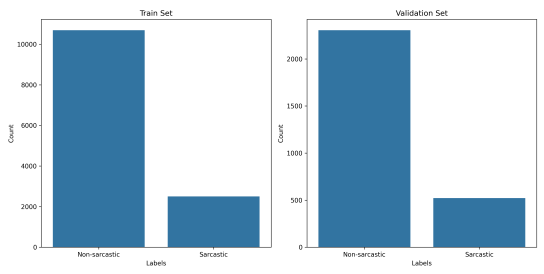

Figure 1: Class distribution for Malayalam text

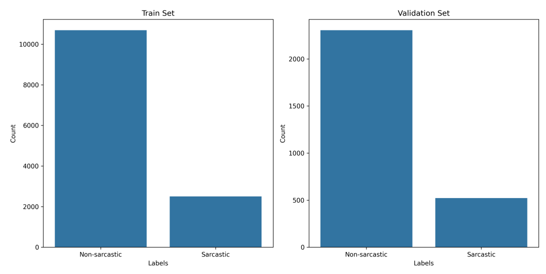

Figure 2: Class distribution for Tamil text

4. Methodology

4.1. Dataset Preprocessing

A thorough preparation pipeline is crucial when working with both native-script and Romanized Tamil text data. The first step is transliterating Romanized Tamil into native Tamil script using the indic-transliteration module to ensure a consistent language representation during feature extraction.

Following transliteration, preprocessing is carried out to clean and standardize the data. All text is converted to lowercase to maintain uniformity, and punctuation, special characters, and non-alphabetical symbols are removed, as they don't contribute to the text's semantic meaning. Stopwords, common words like "and" ,"the" and "is" which don't add significant meaning, are also removed using a specially compiled Tamil stopwords list, expanded from publicly available sources. This step reduces dimensionality and allows models to focus on informative words.

Tokenization is performed using the indic_tokenize module, designed to handle the complexities of Indian languages. The TfidfVectorizer then converts each sentence into meaningful tokens or words and transforms them into numerical features. By giving words that appear frequently within a document but rarely across the entire corpus higher weights, TfidfVectorizer captures the relative importance of each word. To maintain computational feasibility, the vocabulary is limited to the top 5,000 most frequent terms.

Four machine learning models are employed for sarcasm detection in Tamil text: K-Nearest Neighbors (KNN), Support Vector Machines (SVM), Decision Trees (DT), and Logistic Regression (LR). These models are selected for their ability to handle high-dimensional feature spaces and their efficiency in text classification tasks.

4.2. Feature engineering and data splitting

Initially, a bigger and more varied dataset was produced by combining the initial training and validation datasets. In order to improve the model's ability to generalize, more data was made accessible for training. Following the datasets' merger, an 80-20 split was implemented, whereby 20% of the merged data was set aside for testing and 80% was used for training. This method made it possible to train the model on a greater range of instances while still offering a distinct test set for assessing how well it performed on data that wasn't used for training.

For feature extraction, TfidfVectorizer was utilized to convert text into a numerical matrix, evaluating the significance of words in relation to the entire dataset. Word embeddings were also employed to

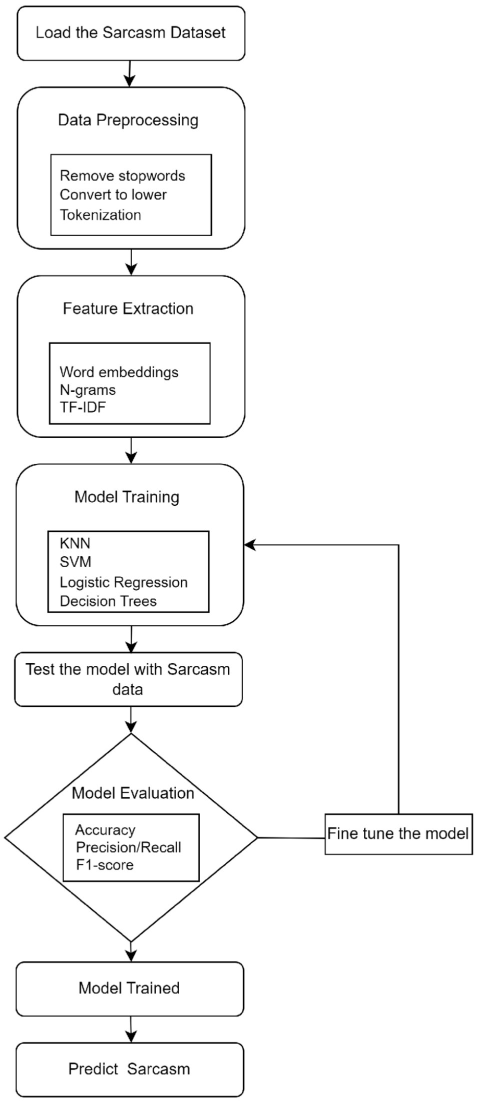

Figure 3: Workflow of proposed solution

translate words into continuous vector representations, capturing semantic relationships based on context. To preserve local context and better understand word associations, N-grams, particularly trigrams, were used. The selection of trigrams was aimed at incorporating patterns involving sequences of three words, which can provide a deeper understanding of context, especially for capturing nuanced expressions like sarcasm. The extracted features from these techniques were then used to train and test the machine learning models.

4.3. Model Selection

4.3.1. Logistic Regression

A popular classification technique for sarcasm detection, even in contexts with mixed language codes, is logistic regression. The model uses a probability score between [0,1] to determine if a text is sarcastic or not. Texts that meet a predetermined threshold (e.g., 0.5) are classed as sarcastic. The code-mixed text's sentiment scores, word frequencies, and context-based signals are among the features that are utilized to produce predictions. While LR yields interpretable findings, it is not always able to handle the delicate, context-dependent nature of sarcasm, particularly in texts with mixed codes. Sarcasm frequently conveys contradicting or weak feelings that are difficult to define with straightforward linear bounds. We have obtained an accuracy of 75.49% with F1 score of 0.371 in Tamil dataset and an accuracy of 83.61% in the Malayalam dataset

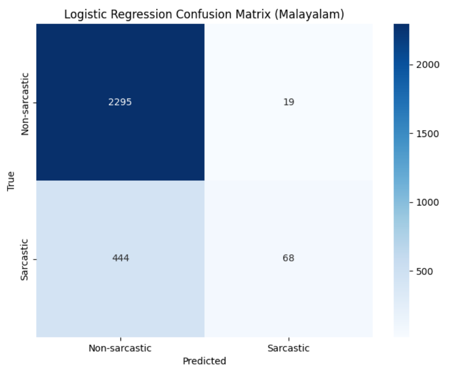

Figure 4: LR - Confusion Matrix - Malayalam

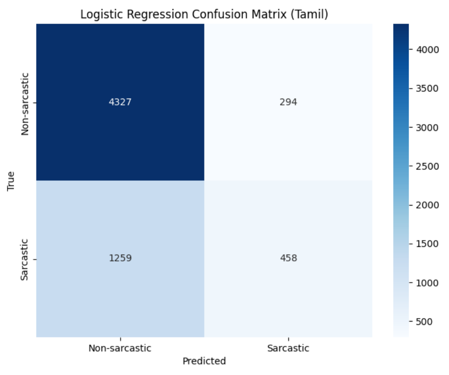

Figure 5: LR - Confusion Matrix - Tamil

4.3.2. K-Nearest Neignbour(KNN)

By categorizing texts based on their similarity to pre-labeled sarcastic or non-sarcastic samples, the k-Nearest Neighbors method (KNN) can detect sarcasm without needing complex frameworks. When applied to code-mixed languages, KNN analyzes features from both languages, such as word frequency, sentence structure, and contextual patterns, to find the closest matches. The final label is determined by the majority class of the nearest neighbors.

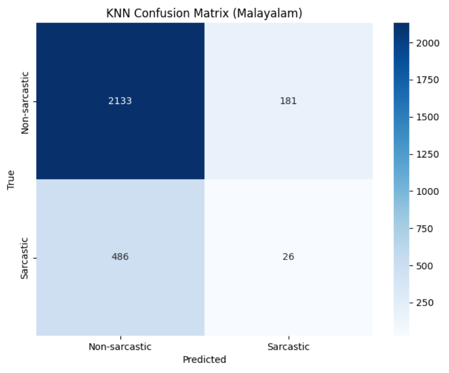

Figure 6: KNN - Confusion Matrix Malayalam

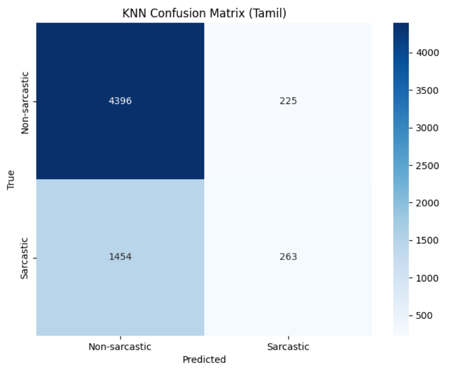

Figure 7: KNN - Confusion Matrix Tamil

Although simple and easy to implement, KNN struggles to accurately identify sarcasm, especially in code-mixed texts. Traditional similarity metrics often fail to capture subtle cues like tone, wordplay, or the context of language switching, which are critical for detecting sarcasm. As a result, KNN's effectiveness is limited in these scenarios. On the Tamil dataset, we achieved an accuracy of 73.51% with an F1 score of 0.238. For the Malayalam dataset, the accuracy reached 76.39%. This shows KNN's limitations in identifying sarcasm in complex language settings.

4.3.3. Support Vector Machines(SVM)

When it comes to code-mixed languages like Tamil-English, Support Vector Machines (SVM) are especially helpful in differentiating between messages that are sarcastic and those that are not. By converting the text into numerical features, such TF-IDF scores, and extracting linguistic traits unique to the code-mixed language, SVM finds the ideal hyperplane that divides the two groups. SVM is a good option since sarcasm in these kinds of writings frequently depends on how both languages interact with one another and their contextual meanings. It can handle sparsity and high-dimensional feature spaces well, which makes it appropriate for sarcasm detection in code-mixed, complicated, context-rich datasets. We have obtained an accuracy of 66.424% with F1 score of 0.321% in Tamil dataset and an accuracy of 71.019% in the Malayalam dataset

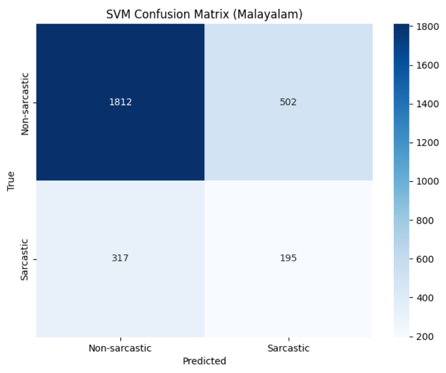

Figure 8: SVM - Confusion Matrix Malayalam

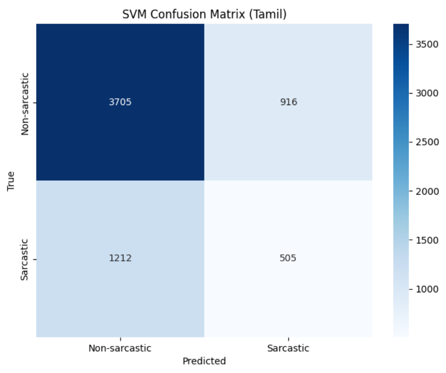

Figure 9: SVM - Confusion Matrix Tamil

4.3.4. Decision Tree

Decision Trees (DT) in sarcasm detection function by recursively segmenting the data according to the characteristics that most effectively distinguish sarcastic from non-sarcastic occurrences. These characteristics for texts with mixed codes could be word choice, punctuation, mood changes, and grammatical patterns in both languages. The feature providing the most information gain is chosen at each node in the tree, resulting in an organized sequence of choices that establish categorization.

Although DTs are easily interpreted and shown, a basic tree structure may not be able to properly capture the complex and context-dependent nature of sarcasm in code-mixed language. In these situations, feature engineering and the incorporation of DTs into ensemble models can enhance their efficacy in sarcasm detection. We have obtained an accuracy of 69.359% with F1 score of 0.403 in Tamil dataset and an accuracy of 77.17% in the Malayalam dataset.

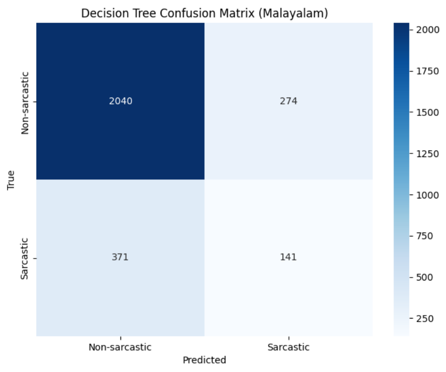

Figure 10: DT - Confusion Matrix - Malayalam

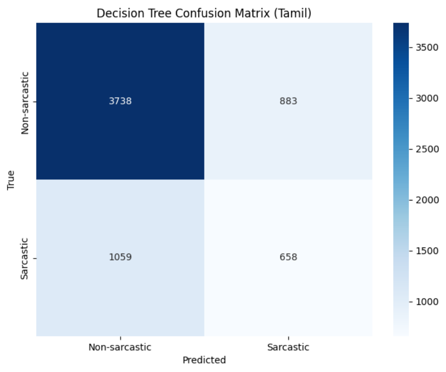

Figure 11: DT - Confusion Matrix - Tamil

5. Conclusion

The methodology involved comparing the actual results published on the website with the outcomes produced by each model. This approach facilitated a comprehensive assessment of the models' performance relative to established benchmarks.Identification of sarcasm is essential, with Logistic Regression (LR) achieving the highest accuracies of 75.50% for Tamil and 83.62% for Malayalam. Despite its straightforward nature, LR effectively handled code-mixed data, delivering consistent performance

Table 1
Classification Report for Malayalam language using Machine Learning models (Non - Sarcastic)

Model	F1 Score	Precision	Recall	Support
Decision Tree	0.8635	0.8461	0.8816	2314
KNN	0.8648	0.8144	0.9218	2314
Logistic Regression	0.9084	0.8379	0.9918	2314
SVM	0.8157	0.8511	0.7831	2314

Table 2
Classification Report for Malayalam language using Machine Learning models (Sarcastic)

Model	F1 Score	Precision	Recall	Support
Decision Tree	0.3042	0.3398	0.2754	512
KNN	0.0723	0.1256	0.0508	512
Logistic Regression	0.2270	0.7816	0.1328	512
SVM	0.3226	0.2798	0.3809	512

Table 3
Classification Report for Tamil language using Machine Learning models (Non-sarcastic)

Model	F1 Score	Precision	Recall	Support
Decision Tree	0.7938	0.7792	0.8089	4621
KNN	0.8397	0.7515	0.9513	4621
Logistic Regression	0.8478	0.7746	0.9364	4621
SVM	0.7769	0.7535	0.8018	4621

Table 4
Classification Report for Tamil language using Machine Learning models (Sarcastic)

Model	F1 Score	Precision	Recall	Support
Decision Tree	0.4039	0.4270	0.3832	1717
KNN	0.2385	0.5389	0.1532	1717
Logistic Regression	0.3710	0.6090	0.2667	1717
SVM	0.3219	0.3554	0.2941	1717

across both languages. In comparison, K-Nearest Neighbors (KNN) struggled with context identification in code-switching scenarios, resulting in lower accuracies of 73.51% for Tamil and 76.39% for Malayalam. Decision Trees (DT) outperformed KNN in capturing context-based signals, while Support Vector Machines (SVM) demonstrated satisfactory performance with high-dimensional data, although both models faced difficulties in grasping the nuances of sarcasm. Ultimately, the incorporation of transliteration-based techniques and tailored machine learning models contributed to improved system accuracy, highlighting the potential for enhanced sarcasm detection across multilingual datasets.

Declaration on Generative AI

During the preparation of this work, the author(s) used ChatGPT in order to: drafting content, grammar and spelling check, etc. After using this tool/service, the author(s) reviewed and edited the content as needed and take(s) full responsibility for the publication's content.

References

[1] 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.

[2] 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.

[3] D. Das, A multimodal approach to sarcasm detection on social media (2019).

[4] S. M. Sarsam, H. Al-Samarraie, A. I. Alzahrani, B. Wright, Sarcasm detection using machine learning algorithms in twitter: A systematic review, International Journal of Market Research 62 (2020) 578–598.

[5] D. Vinoth, P. Prabhavathy, An intelligent machine learning-based sarcasm detection and classification model on social networks, The Journal of Supercomputing 78 (2022) 10575–10594.

[6] H. Thapliyal, Sarcasm Detection System for Hinglish Language (SDSHL), Ph.D. thesis, IIIT Hyderabad, 2020.

[7] R. Viswanadha, Transliteration of tamil and other indic scripts, Tamil Internet 2002 (2002).

[8] T. Ptáek, I. Habernal, J. Hong, Sarcasm detection on czech and english twitter, in: COLING 2014, the 25th International Conference on Computational Linguistics, 2014, pp. 213–223.

[9] S. K. Bharti, K. S. Babu, Sarcasm as a contradiction between a tweet and its temporal facts: a pattern-based approach, International Journal on Natural Language Computing (IJNLC) Vol 7 (2018).

[10] S. G. Shamay-Tsoory, R. Tomer, J. Aharon-Peretz, The neuroanatomical basis of understanding sarcasm and its relationship to social cognition., Neuropsychology 19 (2005) 288.

[11] A. Kumar, S. Dikshit, V. H. C. Albuquerque, Explainable artificial intelligence for sarcasm detection in dialogues, Wireless Communications and Mobile Computing 2021 (2021) 2939334.

[12] N. Sripriya, T. Durairaj, K. Nandhini, B. Bharathi, K. Ponnusamy, C. Rajkumar, P. Kumaresan, R. Ponnusamy, S. C. Navaneethakrishnan, B. R. Chakravarthi, Findings of shared task on sarcasm identification in code-mixed dravidian languages (2023) 22–24. doi:10.1145/3632754.3633077.

[13] O. Vitman, Y. Kostiuk, G. Sidorov, A. Gelbukh, Sarcasm detection framework using context, emotion and sentiment features, Expert systems with applications 234 (2023) 121068–121068. doi:10.1016/j.eswa.2023.121068.

[14] S. Bhosale, A. Chaudhuri, A. L. R. Williams, D. Tiwari, A. Dutta, X. Zhu, P. Bhattacharyya, D. Kanojia, 16. sarcasm in sight and sound: Benchmarking and expansion to improve multimodal sarcasm detection, arXiv.org (2023). doi:10.48550/arxiv.2310.01430.

[15] B. P. Singh, D. K. Sharma, A survey of sarcasm detection techniques in natural language processing, 2023, pp. 1–6. doi:10.1109/ISCON57294.2023.10112176.

[16] J. Wang, L. Sun, Y. Y. Liu, M. Shao, Z. W. Zheng, 24. multimodal sarcasm target identification in tweets, 2022. doi:10.18653/v1/2022.acl-long.562.

[17] 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.

[18] 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.

[19] 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.