Manchester Metropolitan University

e-space

Manchester Metropolitan University's Research Repository

Please cite the Published Version

Jain, Deepak Kumar, Kumar, Akshi and Sangwan, Saurabh Raj (2022) TANA: The amalgam neural architecture for sarcasm detection in indian indigenous language combining LSTM and SVM with word-emoji embeddings. Pattern Recognition Letters, 160. pp. 11-18. ISSN 0167-8655

DOI: https://doi.org/10.1016/j.patrec.2022.05.026

Publisher: Elsevier

Version: Accepted Version

Downloaded from: https://e-space.mmu.ac.uk/629808/

Usage rights: BY-NC-ND Creative Commons: Attribution-Noncommercial-No Derivative Works 4.0

Additional Information: This is an Author Accepted Manuscript of an article published in Pattern Recognition Letters, by Elsevier.

Enquiries:
If you have questions about this document, contact openresearch@mmu.ac.uk. Please include the URL of the record in e-space. If you believe that your, or a third party's rights have been compromised through this document please see our Take Down policy (available from https://www.mmu.ac.uk/library/using-the-library/policies-and-guidelines)

Research Highlights

- TANA, A Hindi sarcasm detection model that combines LSTM and hinge loss of SVM
- Use of pre-trained fastText and emoji2vec embeddings for training
- Performance validation using various metrics, accuracy, F1, recall, precision
- Superior performance in comparison to existing works

Pattern Recognition Letters

TANA: The Amalgam Neural Architecture for Sarcasm Detection in Indian Indigenous Language combining LSTM and SVM with Word-Emoji Embeddings

Deepak Kumar Jain[a], Akshi Kumar[b] and Saurabh Raj Sangwan[c1]

[a] *Key Laboratory of Intelligent Air-Ground Cooperative Control for Universities in Chongqing, College of Automation, Chongqing University of Posts and Telecommunications, Chongqing, China*
[b] *Department of Computing and Mathematics, Faculty of Science and Engineering, Manchester Metropolitan University, Manchester, United Kingdom*
[c] *Department of Computer Science and Engineering, Netaji Subhas University of Technology, New Delhi, India*

ABSTRACT

Sentiment analysis is indeed a difficult task owing to the playful language mannerism, altered vocabulary and speak-text used on online forums. Humans tend to use words and phrases in ways that are incomprehensible to those who are not involved in the discourse. Sarcastic remarks in conversations are often utilized to mock others by saying something that isn't pleasant. Sardonic or humorous statements/ tones are used to insult or make others appear puerile. Automated sarcasm detection is considered as one of the key tasks to tweak sentiment analysis and extending it to a morphologically rich and free-order dominant indigenous Indian language Hindi is another challenge. This research puts forward '*T*he *A*malgam *N*eural *A*rchitecture', TANA, to detect sarcasm in Hindi tweets. The architecture is trained using two embeddings, namely word and emoji embeddings and combines an LSTM with the loss function of SVM for sarcasm detection. We use the Sarc-H dataset, which is built by scrapping Hindi language tweets and manually annotating based on the hashtags '#कटाक्ष' (pronounced as kataaksh, which means sarcasm in Hindi) and '#व्यंग्य' (pronounced as vyangya, another word for sarcasm in Hindi) used by the tweeters and the results are evaluated using various classification performance metrics and achieves a F-score of 0.9675 outperforming LSTM using last layer as softmax as well as the existing works.

1. Introduction

The abundance of social media generates a colossal quantity of content for consumption, clarification, and communication. Sentiment analysis is contextual mining of text which classifies polarity and subjectivity in user-generated content. It monitors conversations and evaluates language and voice inflections to quantify attitudes, opinions, and emotions related to a business, brand, product or service, or topic. Indeed, sentiment analysis is used across a variety of applications and for myriad purposes. Computationally identifying and categorizing polarity expressed in piece of text can either be in a subjective or objective sentence and analyzed at message, sentence or at entity and aspect level for finding regular, comparative, implicit or explicit opinion. But human sentiments are not just restricted to the discrete classes of positive, negative, or neutral. There are more complex emotions and communicative expressions within these defined categories which are difficult to categorize. As a key challenge that analysts in sentiment analysis continue to face is how to detect sarcasm in real-time user-generated text [1, 2].

One of the most promising solutions for training machines to identify sarcasm is the one that looks for context around tweets. As a focused NLP task, context-aware weighting for automatic detection of sarcasm intends to find accompanying hints from users' linguistic input that are aware of 'context' to aid right interpretation [3]. This is important as for the word *'excellent'* a tweet like "*He is amazingly excellent at cheating"* can be regarded as positive without context. The above tweet can only be classified as negative when the context of the word *'excellent'* is considered, as the word *'cheating'* is a negative polarity term. Evidently, the context of each word can aid in properly categorizing its polarity, increasing the accuracy of conventional sentiment analysis. Basically, one needs more than just words to assist in deciding the true nature of the sentence as sarcasm is not in your face and obvious at textual level. As a result, more attention needs to be paid to the semantic relationships between the words in the sentence which could elucidate the presence of certain incongruence and serve as potential indicators of sarcasm [4]. Another accentuating online phenomenon is the emergence of the use of visual language emoji in social media conversations. Emojis work with human psychology and compensate for the 70% of human emotional expression that might come from non-verbal cues, such as facial expressions, body language, and tone in face-to-face conversations. Emojis can express emotion that would be difficult to convey via text alone. For example, "*He is amazingly excellent at cheating* 😛" is a more lucid representation of a goofiness associated with the statement thus making it less vulnerable to incorrect interpretations. Emojis are advantageous as

[a]deepak@cqupt.edu.cn ; [b]akshi.kumar@mmu.ac.uk
[1c] saurabh.trf18@nsut.ac.in (Corresponding author: S.R. Sangwan)

these are almost completely universal in their meanings and can be unanimously understood by hugely diverse audience.

Most of the work on detecting sarcasm in textual data has been done only on English language [5-7]. Hindi is numerically and proportionally the largest indigenous language community in the Indian sub-continent. It is the official language of India and a sizeable population speaks/ writes Hindi while the rest are comfortable in their regional language [8]. But like other Indian languages, Hindi is a resource poor language [9]. The lack of annotated dataset, various analysis tools like POS tagger and sentiment scores have restricted the scope of research in sentiment analysis and subsequent sarcasm detection in Hindi. Also, like most of the Indian languages, Hindi too has free-word order. For example, कपड़े अच्छे हैं इस दुकान के (pronounced as Kapde acche hain iss dukaan ke), इस दुकान के कपड़े अच्छे हैं (pronounced as Iss dukaan ke kapde acche hain), अच्छे कपड़े हैं इस दुकान के (pronounced as Acche kapde hain iss dukaan ke) all three statements convey the same meaning "This shop has good clothes" with different word order. Considering such limitations, only few research efforts are available publicly for sarcasm detection in Hindi [5, 10, 11]. Most of the previous work on sarcasm detection in Hindi rely lexical language resources using statistical or the semantic relations between the words for polarity annotation and are efficient and simple methods with acceptable results. These are unsupervised methods that do not need training data. On the flip side lexicon-based techniques report low recall and suffer due to out-of-vocabulary words, microtext, speak text and emojis. Moreover, such methods are not only time-consuming and cumbersome, but often fail to correctly interpret the context of each word with respect to its neighbors or the entire sentence [12, 13]. More recently, deep neural networks achieve state-of-the-art results at discerning patterns and discriminative features in sarcasm detection task, but limited studies have explored indigenous language datasets.

Thus, motivated by the need to develop more efficient models for sarcasm detection in the resource-poor indigenous language, Hindi, this research proposes a TANA model. That is, the amalgam neural architecture which combines LSTM and SVM by automatically learning features with the help of word-emoji embedding. Additionally, the word 'TANA' signifies the functionality that the model serves as the Hindi word 'ताना' (pronounced as Tana) means sarcasm, taunt, or gibe in English.

The pre-trained fastText[2] Hindi word embeddings are utilized in this research. Additionally, to achieve added confidence in the intended sentiment of any sentence, information conveyed by emojis is also considered. For this purpose, the pre-trained emoji2vec [14] emoji embedding is used. The embeddings are concatenated to form an integer-encoded word-emoji embedding vector. The model combines the long short-term memory (LSTM) model with the loss function of support vector machine (SVM) with the squared Hinge loss function to identify sarcasm. The Sarc-H dataset built by scrapping Hindi language tweets and manually annotating based on the hashtags used by the tweeters is used. The classification performance of baselines and the hybrid model is evaluated using accuracy, F1 Score, precision and recall as metrics.

The rest of the paper is organized as follows: Section 2 surveys the related work done in this domain; Section 3 discusses the TANA model for Hindi sarcasm detection. Section 4 focuses on the experimental settings and the results obtained. The paper concludes the work with a brief discussion on the future work in Section 6.

2. Related Work

Deep learning approaches have obtained very high performance across many different NLP tasks including sarcasm detection. Pertinent studies reveal most of the work on sarcasm detection has been done on English language. Few studies have been reported on Arabic, Dutch and Chinese languages. In 2018, Alayba et al. [15] used a hybrid of CNN and LSTM for sarcasm detection in Arabic posts. Ptáček et al. [16] used machine learning for sarcasm detection in Dutch tweets. Liu et al. [17] proposed a multi-strategy ensemble learning approach (MSELA) for handling imbalanced datasets in Chinese and introduced a set of features specifically for detecting sarcasm in social media. Similar research has been carried out languages like Spanish [18] and Indonesian [19]. As far as the low-resource Hindi language is concerned, research on code-mix social media text which is a linguistic anglicization of Hindi (transliteration based on pronunciation, not meaning) has been notably done [1, 20]. One of the pioneer works was reported by Desai and Dave [11] where the authors built a dataset of sarcastic sentences in Hindi and used various lexical features like emoticons, punctuation marks polarity lists to train an SVM classifier for categorizing sarcasm. Bharti et al. [5] used a context-based approach by using input tweet and its related news to count the number of positive and negative keywords in both news and tweet using a predefined list of Hindi words with polarity value to determine if the given tweet is sarcastic or not. In 2017, Bharti et al. [10] presented a pattern-based framework to predict sarcasm in Hindi tweets. This research is the primary effort in the same direction where the strength of embeddings is harnessed to better comprehend the sentiments being manifested by the text.

3. TANA for Sarcasm Detection in Hindi Tweets

TANA implements SVM in last layer while using LSTM for binary text classification. That is, instead of using the conventional cross-entropy loss for binary classification problem, we use a popular extension of the hinge loss function, known as the squared hinge loss, that simply calculates the square of the score hinge loss. The architectural flow of the LSTM-SVM model for sarcasm detection in Hindi tweets using word-emoji embeddings is shown in fig.1.

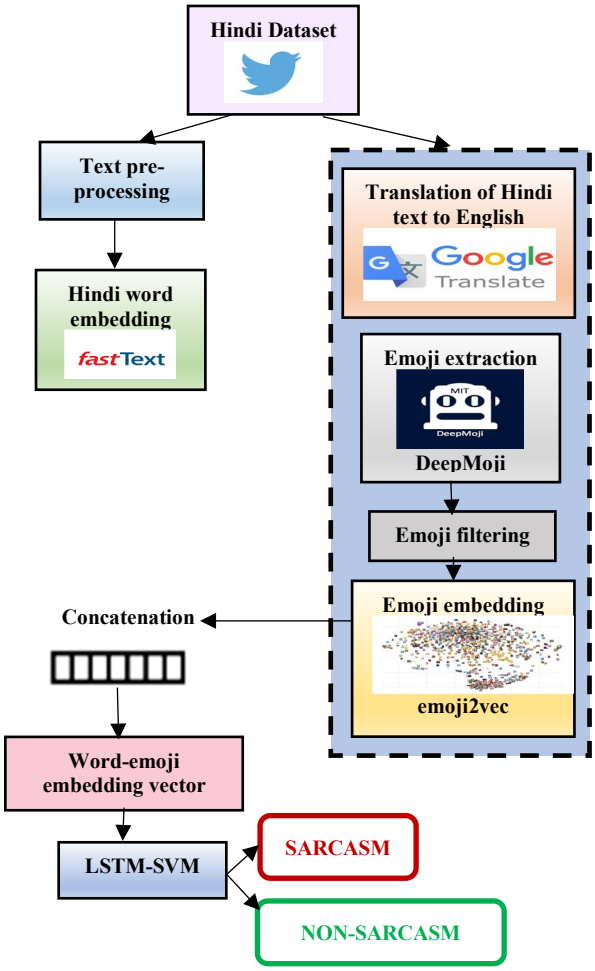

Fig.1. Architectural flow of the TANA model

[2] https://fasttext.cc/

The following sub-sections explain the model in detail.

3.1 Sarc-H Dataset and Pre-processing

The dataset, Sarc-H is created by extracting Hindi language tweets using explicit sarcasm hashtags ('#कटाक्ष' and '#व्यंग्य') for the sarcastic category whereas the negative category includes tweets from popular Hindi news channels. The class annotations are added as labels in the dataset. TweetScraper[3] was used for extracting tweets and dataset with a total of 1004 tweets were built with 414 tweets labelled as sarcastic and 590 labelled as non-sarcastic Given the informal style of writing style that users prefer, a lot of noise is added to the tweet in terms of URL references, mentions of users with @, and use of English words in Hindi sentences. Thus, the fetched tweets are cleaned and pre-processed by removal of strings starting with @ to refer to a user, URLs, hashtag '#' and removal of the string following the hashtag in case of trailing hashtags only [6]. This prevents the employed classification model from predicting every sentence every tweet with '#कटाक्ष' or '#व्यंग्य' as sarcastic and thus focus on extracting actual semantic or syntactic features for distinguishing between sarcastic and non-sarcastic tweets.

3.2 Embeddings

We use a pre-trained word embedding for Hindi language provided by fastText (AN NLP library by Facebook). This model was trained using CBOW with position-weights, in dimension 300, with character n-grams of length 5, a window of size 5 and 10 negatives.

Similarity score is an implementation provided by gensim library which essentially helps us determine how similar two words are to each other. Since these scores are a measure of cosine similarity between the word vectors, a larger value depicts a closer relationship. Just like the word embeddings, emoji embeddings are essentially points in the vector space such that similar emojis exist close to each other while dissimilar ones are relatively more distant. Emoji2vec allows us to do this by representing emojis as vectors in a 300-dimensional vector space. It is a pre-trained model that has been trained on the description of all the emojis in the Unicode emoji standard. The embedding is the sum of word embeddings of words in description. As an example, table 1 showcases the cosine similarity between the selected pair of emojis to depict how similar (higher score) and disparate (lower score) emojis fare with respect to emoji2vec.

Table 1. Similarity score of various emoji pairs according to emoji2vec embeddings

Emoji Pairs	Similarity Score
(😂,😆)	0.6827
(😐,😩)	0.5617
(😆,😩)	0.3619

Not all extracted tweets contain emojis. But as the language of visual symbols effectually substitute body language and tone of voice in text-based communication, their use as contextual cues to detect sarcasm is obligatory. Motivated by the merits of inclusion of emojis along with the availability of the requisite tools to incorporate them into our task, we use DeepMoji [21] for generating emojis relevant to the respective tweet. However, DeepMoji does not support Hindi language and therefore we used the Google Translation API for translating the extracted Hindi tweets to English. However, not all translations were precise enough in terms of the intended meaning and therefore, we refined the translations to better convey the meaning of the given tweet. Following the translation, the entire list of tweets which only comprised the text exclusive of the hashtags and user annotations was fed into the DeepMoji code to generate emojis. While reasonably appropriate emojis were obtained in the case of non-sarcastic tweets, the ones generated for sarcastic tweets represented a wide variety of emotions for most of the tweets. Such a behavior was plausible as DeepMoji was not aware of the appended hashtags of '#sarcasm'. Thus, to include this information in the generation of emojis, we appended each sarcastic tweet with the word 'sarcasm' to obtain relevant emojis. DeepMoji generated 5 emojis to the given input which it deemed most pertinent being mindful of the possible sentiments the author might be intending to manifest. As a result, some of the emojis obtained were indeed irrelevant and hence discarded.

Next to utilize both embeddings for our task, we implemented a concatenation procedure as follows: A word-emoji embedding vector Ef is constructed which has a size equal to the sum of the embedding vector size of word embeddings, Ew, and emoji embeddings, Ee and is initialized with 0s. We have used the upper half to represent the word embeddings and the lower half to represent the emoji embeddings. For any given word, only the top half of the vector is assigned to the respective value of the word embedding while the lower half is untouched. Similarly in case of emojis, only the lower half of the final embedding is assigned to the respective emoji embedding value while the upper half remains set to 0.

3.3 LSTM-SVM

Previous studies demonstrate SVM as an alternative to softmax function for classification and claim improved performance with the use of SVM in an artificial neural network (ANN) architecture in comparison to the conventional softmax function. The approach is suitable for binary classification, as in this research the tweets are classified as sarcastic or non-sarcastic. The combination of LSTM with SVM is shown in fig.2.

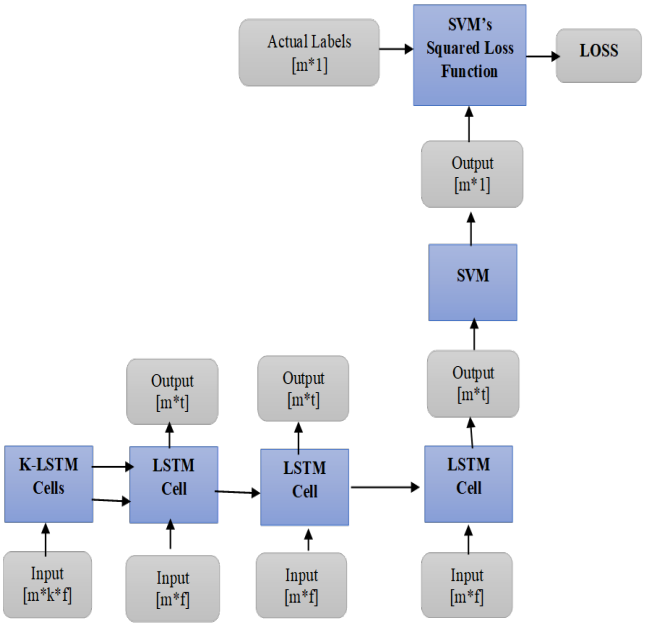

Fig.2. LSTM-SVM

Where,
m: Batch size
k: No. of LSTM cells
f: No. of features
t: Output dimension of a single LSTM cell
Input dimension: m * k * f
Output of LSTM network at one time step: m * t

Long Short-term memory or LSTM [22], a variant of RNN, helps solve the problems of vanishing and exploding gradients thus allowing it to plot long-term dependencies by defining each memory cell with a set of gates R_d, where d is the memory

[3] https://pypi.org/project/tweetscraper/1.2.0/

dimension of hidden state of LSTM. LSTM consists of three gates, which are functions of x_t, input at the current time step and h_t–1, the hidden state: input gate i_t, which decides by how much each memory cell has to be updated, forget gate f_t ,which decides whether or not to discard the memory cell state information that came from h_t–1 and output gate o_t , which takes the decision of passing the memory state to the rest of the network. The gates jointly decide on the memory update mechanism. The LSTM transition functions are as shown in (1) to (6):

$$i_t = \sigma(W_i[h_{t-}1,x_t] + b_i) \quad (1)$$
$$f_t = \sigma(W_f[h_{t-}1,x_t] + b_f) \quad (2)$$
$$q_t = \tanh(W_q[h_{t-}1,x_t] + b_q) \quad (3)$$
$$o_t = \sigma(W_o[h_{t-}1,x_t] + b_o) \quad (4)$$
$$ct = ft.ct_{-}1 + it \odot qt \quad (5)$$
$$h_t = o_t \odot \tanh(c_t) \quad (6)$$

where σ denotes the logistic sigmoid function that provides an output in [0,1], tanh denotes the hyperbolic tangent function with the output in the range [-1,1], and $\odot$ denotes element wise multiplication.

The softmax layer is usually used as a network output layer and the function of cross-entropy loss is used as the decision function. However, we have used linear SVM as the final layer in this work as given by Tang [23], and instead of optimizing the cross-entropy loss function, we have optimized the SVM decision function provided as given in (7)

$$Loss = min\frac{1}{2}\|w\|^2 + C \cdot \sum_{i=1}^{n} \max(0, 1 - t \cdot y) \quad (7)$$

This is called L1-SVM unconstrained optimization problem with the hinge loss. This objective function is not differentiable therefore another variation of it, called L2-SVM, with squared hinge loss is used as given in (8)

$$Loss = min\frac{1}{2}\|w\|^2 + C \cdot \sum_{i=1}^{n} \max(0, 1 - t \cdot y)^2 \quad (8)$$

where, C = regularization factor,
t = target labels [-1,1],
y = input to the SVM layer = (w · x + b) where x is the output of LSTM

This loss function is a margin-based loss function that is a bigger loss is imposed on points which violate the decision margin. The parameters of LSTM are updated by various gates as well as by optimizing the loss function of SVM.

4. Results and Discussion

The dataset was split into 70:30 training and testing datasets. Though the dataset was small but deep learning allowed easy incorporation of the problem-specific constraints directly into the model to reduce variance. At the same time neural nets have a large library of techniques to combat overfitting. These techniques helped mitigate the variance issue, while still benefitting from the flexibility.

A simplistic ANN and LSTM were applied discretely to define the baselines for the study. For hyperparameter tuning, we made use of automated hyperparameter optimization that is provided by a python library, Hyperopt [24]. The models were evaluated using four performance metrics: accuracy, precision, recall and F-score. The confusion matrix is used to visualize the output of the device [25]. The table for the confusion matrix includes the following values:

- True Positive (TP): total no. of tweets which were "sarcastic" and predicted "sarcastic".
- False Negative (FN): total no. of tweets which were "non-sarcastic" and predicted "sarcastic".
- False Positive (FP): total no. of tweets which were "sarcastic" and predicted "non-sarcastic".
- True Negative (TN): total no. of tweets which were "non-sarcastic" and predicted "non-sarcastic".

Table 2, 3 and 4 depict the confusion matrix for ANN, LSTM and LSTM-SVM respectively.

Table 2. Confusion matrix for ANN

	Non-Sarcastic	***Sarcastic***
Non-Sarcastic	153	10
Sarcastic	12	127

Table 3. Confusion matrix for LSTM

	Non-Sarcastic	***Sarcastic***
Non-Sarcastic	155	8
Sarcastic	4	135

Table 4. Confusion matrix for LSTM-SVM

	Non-Sarcastic	***Sarcastic***
Non-Sarcastic	159	4
Sarcastic	5	134

The ROC curve for the TANA model is shown in fig.3.

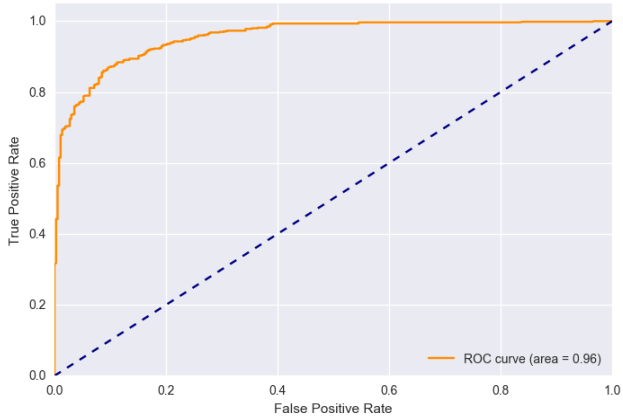

Fig.3. ROC curve of LSTM-SVM

Table 5 presents the performance comparison of all the models using the evaluation metrics. The proposed TANA model outperforms all the models, achieving an accuracy of 97.01% and F-score of 0.9675. ANN, on the other hand, falls on the lower end of the spectrum, achieving the lowest, though an appreciable accuracy of 92.72% and F-score of 0.9203.

Table 5. Performance comparison of all models

Model	***Precision***	***Recall***	***F-Score***	***Accuracy***
ANN	0.9270	0.9137	0.9203	0.9272
LSTM	0.9440	0.9712	0.9574	0.9603
TANA (LSTM-SVM)	0.9640	0.9710	0.9675	0.9701

The results of TANA were also evaluated without using emojis. Table 6 shows the confusion matrix for LSTM-SVM based TANA without the use of emojis.

Table 6. Confusion matrix for LSTM-SVM without emojis

	Non-Sarcastic	***Sarcastic***
Non-Sarcastic	152	11
Sarcastic	13	126

A F-score of 0.9131 and an accuracy of 92.05% was observed without the use of emojis, clearly establishing the role of emojis in discerning sarcasm in written content. Fig.4 shows the results of using TANA with and without emojis.

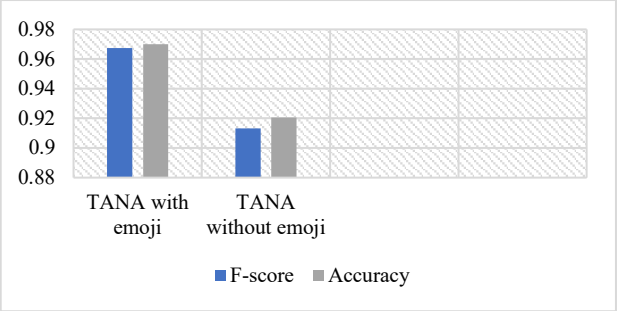

Fig.4. Comparison of proposed TANA model with and without emojis

The results were also compared with the existing work done based on performance metrics. Fig.5 presents the comparison along with the details of dataset, features and techniques utilized in each study.

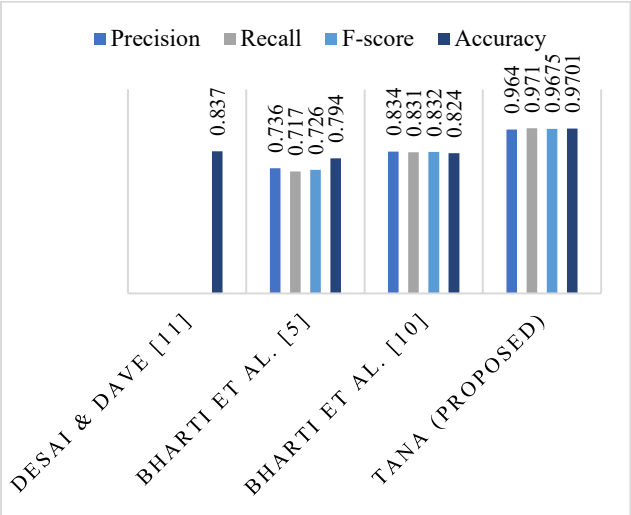

Fig.5. Comparison of proposed TANA model with existing state-of-the-art

5. Conclusion

Sarcasm is a pivotal natural language challenge to analyze sentiments accurately as most text-based conversation is flat-toned. The exact emotion or intention is difficult to comprehend, and this task becomes even more strenuous for indigenous languages like Hindi which are complex in morphology and lack sufficient resources to facilitate analytics. The ability to analyze text has substantially increased with the advances in deep learning and this work described one such deep learning based model, TANA. As context incongruity signaled by words and emojis can be used to detect sarcasm in online data streams, we used a combination of fastText and emoji2vec embeddings to generate an integer-encoded word-emoji vector that trained a LSTM-SVM. That is, in TANA the L2-SVM is implemented in the output layer of the LSTM instead of the conventional softmax function with the cross-entropy function (for computing loss) to detect sarcasm. The model was validated on a Hindi dataset created for the purpose for detecting sarcasm detection. TANA performed superiorly with 97.01% accuracy, 0.9675 F-score, 0.9710 recall and 0.9640 precision. On comparison with the existing works too, the TANA model demonstrated superlative results. The research validates that automated feature engineering facilitates detecting sarcasm in indigenous, low-resource languages and at the same time, emojis can prove to be quite pivotal in the determination of the true sentiment of any tweet.

The dataset size is surely a limitation, and we intend to increase the dataset for generalization. Further the use features such as number of punctuations and number of hashtags can also be evaluated for finding signs of sarcasm in text. The study on transliterated text and code-switched with Hindi are some promising directions of future work.

Funding

The author(s) received no financial support for the research, authorship, and/or publication of this article.

References

1. Jain, D., Kumar, A., & Garg, G. (2020). Sarcasm detection in mash-up language using soft-attention based bi-directional LSTM and feature-rich CNN. Applied Soft Computing, 91, 106198.
2. Majumder, N., Poria, S., Peng, H., Chhaya, N., Cambria, E., & Gelbukh, A. (2019). Sentiment and sarcasm classification with multitask learning. IEEE Intelligent Systems, 34(3), 38-43.
3. Kumar, A., & Garg, G. (2019). Empirical study of shallow and deep learning models for sarcasm detection using context in benchmark datasets. Journal of Ambient Intelligence and Humanized Computing, 1-16.
4. Bliss-Carroll, N. L. (2016). The nature, function, and value of emojis as contemporary tools of digital interpersonal communicate
5. Bharti, S. K., Babu, K. S., & Jena, S. K. (2015, August). Parsing-based sarcasm sentiment recognition in twitter data. In 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM) (pp. 1373-1380). IEEE.
6. Kumar, A., & Garg, G. (2019). Empirical study of shallow and deep learning models for sarcasm detection using context in benchmark datasets. Journal of Ambient Intelligence and Humanized Computing, 1-16.
7. Davidov, D., Tsur, O., & Rappoport, A. (2010, July). Semi-supervised recognition of sarcasm in Twitter and Amazon. In Proceedings of the fourteenth conference on computational natural language learning (pp. 107-116).
8. Parshad, R. D., Bhowmick, S., Chand, V., Kumari, N., & Sinha, N. (2016). What is India speaking? Exploring the "Hinglish" invasion. Physica A: Statistical Mechanics and its Applications, 449, 375-389.
9. Kumar, A. & Albuquerque., VHC. (2021). Sentiment Analysis Using XLM-R Transformer and Zero-shot Transfer Learning on Resource-poor Indian Language. ACM Trans. Asian Low-Resour. Lang. Inf. Process. 20, 5, Article 90 (September 2021), 13 pages. DOI: https://doi.org/10.1145/3461764
10. Bharti, S. K., Babu, K. S., & Jena, S. K. (2017, December). Harnessing online news for sarcasm detection in hindi tweets. In International Conference on Pattern Recognition and Machine Intelligence (pp. 679-686). Springer, Cham.
11. Desai, N., & Dave, A. D. (2016). Sarcasm detection in Hindi sentences using support vector machine. International Journal, 4(7), 8-15.
12. Kumar, A., Bhatia, M. P. S., & Sangwan, S. R. (2021). Rumour detection using deep learning and filter-wrapper feature selection in benchmark twitter dataset. Multimedia Tools and Applications, 1-18.
13. Sangwan, S. R., & Bhatia, M. P. S. (2020). D-BullyRumbler: a safety rumble strip to resolve online denigration bullying using a hybrid filter-wrapper approach. Multimedia Systems, 1-17.
14. Eisner, B., Rocktäschel, T., Augenstein, I., Bošnjak, M., & Riedel, S. (2016). emoji2vec: Learning emoji representations from their description. arXiv preprint arXiv:1609.08359.
15. Alayba, A. M., Palade, V., England, M., & Iqbal, R. (2018, August). A combined CNN and LSTM model for arabic sentiment analysis. In International cross-domain conference for machine learning and knowledge extraction (pp. 179-191). Springer, Cham.
16. Ptáček, T., Habernal, I., & Hong, J. (2014, August). Sarcasm detection on czech and english twitter. In Proceedings of COLING 2014, the 25th international conference on computational linguistics: Technical papers (pp. 213-223).
17. Liu, P., Chen, W., Ou, G., Wang, T., Yang, D., & Lei, K. (2014, June). Sarcasm detection in social media based on imbalanced classification. In International Conference on Web-Age Information Management (pp. 459-471). Springer, Cham.
18. Justo, R., Alcaide, J. M., Torres, M. I., & Walker, M. (2018). Detection of sarcasm and nastiness: new resources for Spanish language. Cognitive Computation, 10(6), 1135-1151.
19. Lunando, E., & Purwarianti, A. (2013, September). Indonesian social media sentiment analysis with sarcasm detection. In 2013 International Conference on Advanced Computer Science and Information Systems (ICACSIS) (pp. 195-198). IEEE.
20. Swami, S., Khandelwal, A., Singh, V., Akhtar, S. S., & Shrivastava, M. (2018). A corpus of english-hindi code-mixed tweets for sarcasm detection. arXiv preprint arXiv:1805.11869.
21. Felbo, B., Mislove, A., Søgaard, A., Rahwan, I., & Lehmann, S. (2017). Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm. arXiv preprint arXiv:1708.00524.
22. Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural computation, 9(8), 1735-1780.
23. Tang, Y. "Deep learning using linear support vector machines," arXiv preprint arXiv:1306.0239, 2013.
24. Bergstra, J., Yamins, D., & Cox, D. (2013, February). Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In International conference on machine learning (pp. 115-123). PMLR.
25. Kumar, A., Bhatia, M. P. S., & Sangwan, S. R. (2021). Rumour detection using deep learning and filter-wrapper feature selection in benchmark twitter dataset. *Multimedia Tools and Applications*, 1-18.