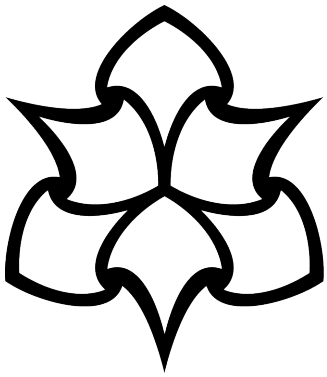

Hybrid Deep Learning Model for Sarcasm Detection in Indian Indigenous Language Using Word-Emoji Embeddings

AKSHI, A.K., KUMAR

Department of Information Technology, Netaji Subhas University of Technology, Delhi, India, akshi.kumar@mmu.ac.uk

SAURABH, S.R.S, RAJ SANGWAN

Department of Computer Science & Engineering, Netaji Subhas University of Technology, Delhi, India, saurabh.trf18@nsut.ac.in

ADARSH, A.K.S., KUMAR SINGH

Department of Information Technology, Delhi Technological University, Delhi, India, kumar.adarsh96@gmail.com

GANDHARV, G.W., WADHWA

Department of Information Technology, Delhi Technological University, Delhi, India, ganswadhwa19@gmail.com

Automated sarcasm detection is deemed as a complex natural language processing task and extending it to a morphologically-rich and free-order dominant indigenous Indian language Hindi is another challenge in itself. The scarcity of resources and tools such as annotated corpora, lexicons, dependency parser, Part-of-Speech tagger and benchmark datasets engorge the linguistic challenges of sarcasm detection in low-resource languages like Hindi. Furthermore, as context incongruity is imperative to detect sarcasm, various linguistic, aural and visual cues can be used to predict target utterance as sarcastic. While pre-trained word embeddings capture the meanings, semantic relationships and different types of contexts in the form of word representations, emojis can also render useful contextual information, analogous to human facial expressions, for gauging sarcasm. Thus, the goal of this research is to demonstrate the use of a hybrid deep learning model trained using two embeddings, namely word and emoji embeddings to detect sarcasm. The model is validated on a Hindi tweets dataset, Sarc-H, manually annotated with sarcastic and non-sarcastic labels. The preliminary results clearly depict the importance of using emojis for sarcasm detection, with our model attaining an accuracy of 97.35% with an F-score of 0.9708. The research validates that automated feature engineering facilitates efficient and repeatable predictive model for detecting sarcasm in indigenous, low-resource languages.

CCS CONCEPTS • Information systems~Information retrieval~Specialized information retrieval~Structure and multilingual text search~Multilingual and cross-lingual retrieval • Information systems~Information systems applications~Collaborative and social computing systems and tools~Social networking sites • Computing methodologies~Artificial intelligence~Natural language processing • Computing methodologies~Machine learning~Machine learning approaches~Neural networks

Additional Keywords and Phrases: Sarcasm, Indigenous, Embeddings, Emojis, Deep learning

ACM Reference Format:

Akshi Kumar, Saurabh Raj Sangwan, Adarsh Kumar Singh and Gandharv Wadhwa, Hybrid Deep Learning Model for Sarcasm Detection in Indian Indigenous Language Using Word-Emoji Embeddings, In ACM TALLIP, mm dd, yy. ACM, New York, NY, USA, 18 pages. https://doi.org/xxx.

1 INTRODUCTION

With the copiousness of social media platforms such as Facebook, Instagram and Twitter, an unprecedented amount of content available for analysis. Sentiment analysis as one of the most prevalent social media mining task that aims to find the polarity of the online post to categorize it as positive, negative or neutral [1-3]. However, human sentiments are not just limited to such discrete classes but also consist of more complex emotions and communicative expressions which are harder to categorize within these defined categories. The expression on social media is informal and has created a pseudo-language with cyberslangs, emojis, neologism, morphological shortenings and metaphorical constructs such as sarcasm, irony and humour. Moreover, the culturally diverse, country-specific trending topics and hashtags and accessibility of aboriginal language keyboards add to the variety of content in numerous languages and dialects on social media platforms [4]. Emotion recognition [5, 6] and sarcasm detection [7-9] in real-time user-generated text are certainly two distinguished NLP tasks imperative for accurate analysis of sentiments.

Automated detection of sarcasm has enticed mounting importance over the past decade as it benefits upright analytics in social media posts [4, 7]. As a rhetorical literary device, sarcasm is highly subjective and contextual. It is vital to comprehend supplementary cues from users' linguistic input that are aware of 'context' to aid right interpretation [10]. So a sentence like "I love working on weekends" seems to have a positive connotation at first as someone is simply expressing his love towards a certain activity. However, we know, for a fact, that generally, people do not really like going to work on weekends as weekends serve as off-days from work when one prefers to relax and spend more time with their family and friends. Therefore, it could be quite inaccurate to classify such a sentence as positive owing to the lack of any additional evidence about the actual sentiment that the author intends to convey. Thus, one needs more than just words to assist in deciding the true nature of the sentence. There is a certain level of intangible acquaintance that one needs to have in order to assess the true sentiment behind a sentence. At the same time, tonal variations, body language and other aural-visual cues play an important role in determining the presence of sarcasm in spoken text. Face-to-face communications are multimodal, that is, the meaning is conveyed not just with words but with gestures like a fake laugh or rolling one's eyes and other facial expressions. Even acoustic markers such as the tone of the speaker, voice pitch, frequency, empathetic stress and pauses hint towards the sarcastic intentions. Such bodily cues can prove to be an effective indicator of the true sentiment the speaker wishes to manifest. However, detecting sarcasm on a textual level that is in written text is comparatively more demanding due to absence of such patent aural-visual cues, making it prone to misinterpretations. 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, 10]. So reconsidering our example, "relax" and "weekend" are more likely a pair that will be used together with a positive verb like "love" or "like" than the pair of "work" and "weekend". The comprehensive analysis of such word associations can help to comprehend the context and eventually detect sarcasm.

On social media, people often use hashtags and emojis or emoticons to add extra layers of meaning to their posts. A hashtag can help pull social media posts into topic-specific feeds, convey irony or sarcasm, suggest emotion or mood, pose an answer to an implied or rhetorical question, or even directly contradict the actual tweet. In case of sarcasm, a lot of such posts are affixed with hashtags like '#not', '#sarcasm' or '#sarcastic' which allows readers to correctly interpret an otherwise flat text. Concurrently, emojis or emoticons are also fast replacing the textual forms of internet slang. These

iconographic set of features convey tone in text conversations similar to those that are observed in face-to-face communication [11]. For example, "I love working on weekends😒" is a more lucid representation of how disappointed the author truly feels about working on weekends thus making it less vulnerable to incorrect interpretations. So, along with the semantic relationships present at textual level, these emojis serve as the visual cues which aid in a more accurate sentiment analysis.

A lot of research has been done on detecting sarcasm in textual data using a myriad of features [12-14]. Many datasets[1] have been made open source to facilitate research enthusiasts. However, most of the work done is only on English language. Many social networking platforms, like Twitter, now provide the users with an option of expressing in a wide range of languages. But as far as other languages are concerned, the exploration and experimentation has been rather limited owing to the inadequate amount of resources that could precisely encapsulate the subtle characteristics of the respective language. One such language is Hindi. 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. The availability of keyboards with 'Devanagari' scripts on mobile phones has made it a popular language choice [15]. But like other Indian languages, Hindi is also a low-resource language [16]. 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, लेख अच्छे हैं इस किताब के, इस किताब के लेख अच्छे हैं, अच्छे लेख हैं इस किताब के, all three statements convey the same meaning with different word order. In light of such limitations, only few research efforts are available publicly for sarcasm detection in Hindi [17, 18, 51, 52, 53]. Most of the previous work on sarcasm detection in Hindi rely on manual feature extraction methods. Such methods are not only time-consuming and cumbersome, but often fail to correctly interpret the context of each word with respect to its neighbours or the entire sentence [19, 20]. Few lexicon-based methods using lists of words such as list of positive-negative words or word-antonym pairs have also been used. Though these methods aim at bringing out the discordant nature of the words used together based on the a binary representation of the sentiment they express, but are quite limited in scope and are less incorporative of the language used by the wider populace. Moreover, none of the studies on Hindi sarcasm detection have used the allied emojis as contextual cues. Interpretation of emojis is important to truly understand and feel any communication. These add a touch of emotion to emphasize the communication analogous to physical cues such as body language and facial expressions in face-to-face communication.

Thus, motivated by the need to develop more efficient and repeatable predictive models for sarcasm detection in a resource-poor indigenous language, Hindi, this research demonstrates the use of a hybrid deep learning model which automatically learns features with the help of word-emoji embedding. Deep learning models rely on the machine or the model itself to run and identify patterns and high-level features which are crucial for the given task [2, 21]. This obviates the need of a domain-expert and solely depends on the data that is fed into the model. Essentially, word embeddings are vector representations of words such that contextually similar words occupy close spatial positions [22]. These embeddings are indicative of the relationships that a given word holds with all the other words to express similarities or disparities among them on a syntactic as well as semantic level and are thus competent to capture the context of a word with respect to a sentence or even a document. This semantic and syntactic knowledge allows us to identify incongruence within a

[1] https://paperswithcode.com/task/sarcasm-detection

sentence thus making the presence of sarcasm more apparent. The pre-trained fastText[2] Hindi word embeddings are utilized in this research. Additionally, in order to achieve added confidence in the intended sentiment of any sentence, information imparted by emojis is also considered. Just like word embeddings, emoji embeddings allow us to represent emojis in an n-dimensional vector space. Such representations when combined with word embeddings and given as input to the learning models allow capturing valuable relationships between the input texts and the accompanying emojis, eventually assisting sarcasm detection. The pre-trained emoji2vec [23] emoji embeddings are used in this research. The embeddings are concatenated to form an integer-encoded word-emoji embedding vector. This input vector is then used to train a hybrid of convolutional neural network (CNN) [24] and long short-term memory (LSTM) [25]. The architecture basically has two sub-models: the CNN model for feature extraction and the LSTM model for interpreting the features across time steps.

A dataset, referred to as Sarc-H where 'Sarc' refers to sarcasm and 'H' signifies Hindi language, is built by scrapping Hindi language tweets and manually annotating based on the hashtags used by the tweeters as follows:

- hashtags '#कटाक्ष ' (pronounced as kataaksh, which means sarcasm in Hindi) and hashtag '#व्यंग्य' (pronounced as vyangya, another word for sarcasm in Hindi) for extracting sarcastic tweets
- the non-sarcastic tweets are extracted tweets from prominent Hindi news channels (Aajtak, NDTV India etc.) official handles. The tweets on these channels generally aim at providing fellow subscribers with true and unambiguous news and hence are free from such nuanced emotion representations keeping them simple and straight-forward.

The classification performance of baselines and the hybrid model is evaluated using accuracy, F1 Score, precision and recall as metrics and clearly establish the importance of automatically extracting useful and meaningful features that help to achieve added confidence in the intended sentiment of any sentence, including sarcasm. The rest of the paper is organized as follows: Section 2 surveys the related work done in this domain; Section 3 discusses the hybrid deep model for Hindi sarcasm detection. Section 4 focuses on the experimental settings and the results obtained. The paper culminates with concluding remarks and a short discussion about the intended future work in Section 6.

2 RELATED WORK

An early work done in the field of sarcasm detection explored the tone of voice used for the expression "yeah right" to establish certain spectral, contextual, and prosodic features of speech which were then used to determine if a spoken sentence was sarcastic or not [26]. Kreuz and Caucci [27] explored lexical features i.e., presence of certain adjectives and adverbs, punctuation and interjections for detection of sarcasm in excerpts from long narratives. Davidov et al. [14] demonstrated sarcasm detection using a semi-supervised classification algorithm on two datasets: a total of 66000 Amazon product reviews and 5.9 million tweets from Twitter from which syntactic and pattern-based features. González-Ibáñez et al. [28] also worked on twitter dataset and used hashtag-based supervision for tweets. They also used unigrams, dictionary-based lexical and pragmatic factors and the frequency of the same with Support Vector Machine (SVM) for the purpose of classification. Riloff et al. [29] employed juxta-positioning of positive sentiment word and a negative situation or state as a deciding factor for presence of sarcasm in tweets. Liebrecht et al. [30] used a balanced winnow algorithm on their dataset of Dutch tweets (which were annotated on the basis of the hashtags used) for classification. They used hyperbole, intensifiers, and exclamation as features.

[2] https://fasttext.cc/

In 2015, Joshi et al. [31] proposed a sarcasm detection system which incorporated explicit and implicit incongruity features that achieved superlative results. In another work, Joshi et al. [32] presented a hypothesis of viewing sarcasm detection in dialogue as a sequence modelling model (SVM-HMM and SEARN) rather than a classification task. They supported the hypothesis by showing an improvement in performance for dataset derived features. Kumar and Garg [33] in 2019 reported a comparative empirical study on various supervised ML techniques for sarcasm detection on Twitter and Reddit datasets. With the rise in multimodal content being shared on social media, wherein the online post is a combination of text and audiovisual content, the task of multimodal sarcasm detection is also being explored extensively [34-36].

Recently, deep learning approaches have obtained very high performance across many different NLP tasks including sarcasm detection. Joshi et al. [37] showcased the use of word embedding features as an indicator of contextual incongruity for enhancing performance on sarcasm detection. Ghosh and Veale [38] used a combination of CNN, LSTM followed by an ANN. They applied the model on publicly available datasets and showed superior results. A secondary study on the past work done on sarcasm detection was given by Joshi et al. [39] in 2016. Context has been used as an essential parameter in sentiment analysis tasks, especially in the case of complex sentiments such as irony or sarcasm where the meaning of any word is not solely based on its literal meaning but also on its frame of reference. A study on context in sarcasm using manual feature extraction and automated feature learning was given by Kumar and Garg [13] in 2020. Attention layer is another mechanism which has proven to be very helpful in capturing contextual knowledge by being considerate of the fact that different words contribute differently to overall intended sentiment of the sentence. Kumar et al. [40] reported a hybrid deep neural model with softmax attention for sarcasm detection.

Literature survey of 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. [41] used a hybrid of CNN and LSTM for sarcasm detection in Arabic posts. Ptáček et al. [42] used machine learning for sarcasm detection in Dutch tweets. Liu et al. [43] 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 [44] and Indonesian [45]. 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 [4, 46]. The research works on Hindi only text are very limited. One of the pioneer works was reported by Desai and Dave [18] 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 which categorized the sentences into 5 classes based on the varying nature of sarcasm. Bharti et al. [17] 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 2018, Bharti et al. [51] presented a pattern-based framework wherein the contradiction between the temporal fact and the corresponding tweet is used to predict sarcasm in Hindi tweets. Katyayan et al. [53] also goes on to explore sarcasm detection in Hindi using 1000 sentences, extracted from social media platforms such as Facebook, Instagram and Twitter. The work utilises POS tagger and bag-of-words technique for feature extraction and, explores a neural network (without any specification of the involved hyper parameters or parameters) and 3 basic machine learning techniques, namely Naive Bayes, SVM and decision tree for classification. None of the previous studies report the use of automated feature learning using embeddings. 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 HYBRID DEEP LEARNING FOR SARCASM DETECTION IN HINDI TWEETS

As the generalization and automated feature learning capabilities of deep learning models exhibit high performance across many different NLP tasks, the research undertaken in the paper demonstrates the same for a convoluted form of expression in written text called sarcasm. The architectural flow of the hybrid CNN-LSTM model for sarcasm detection in Hindi tweets using word-emoji embeddings is shown in Figure 1.

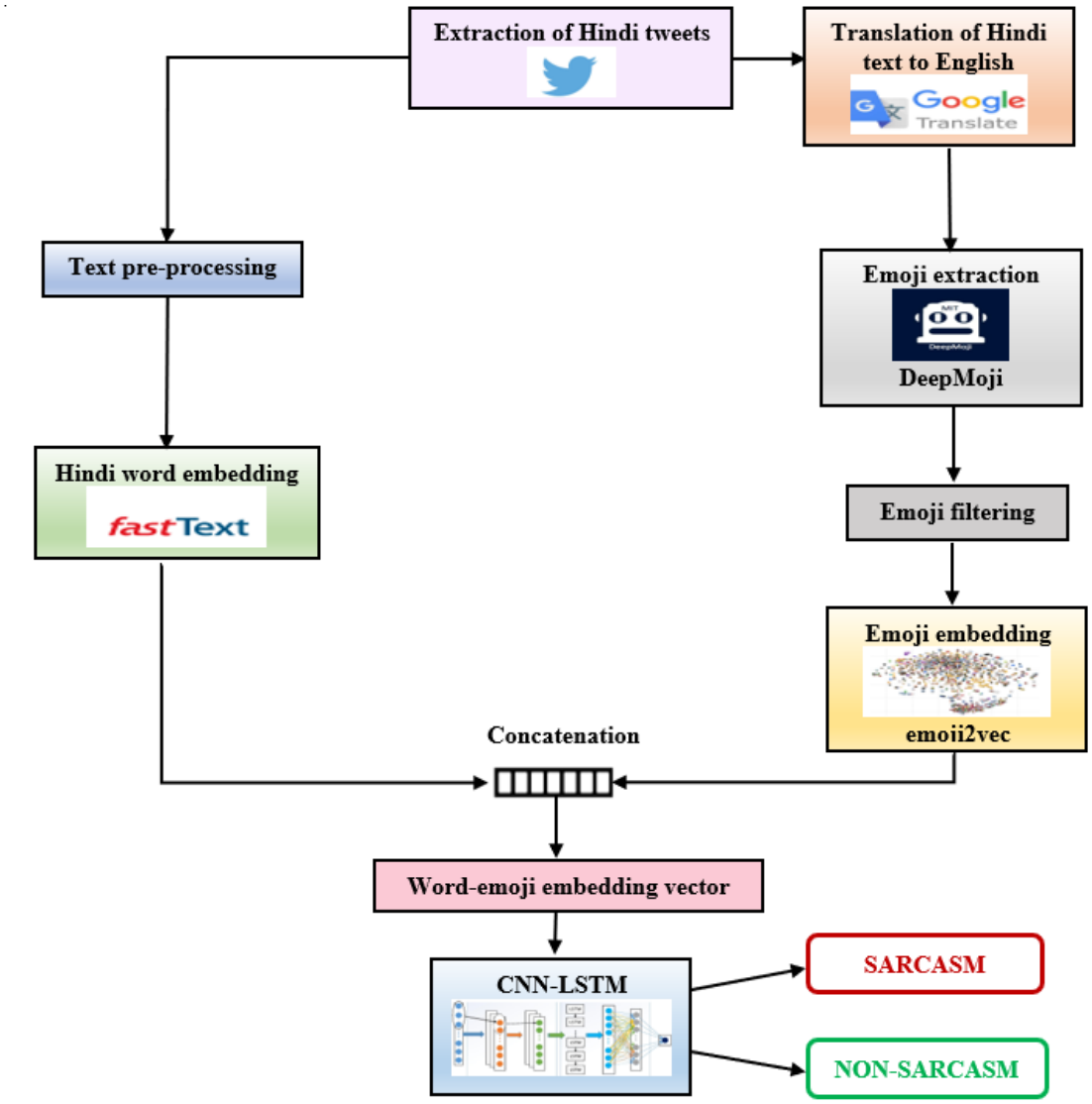

Figure 1: Architectural flow of the hybrid model

The following sub-sections explain the model in detail.

3.1 Dataset Creation and Pre-processing

Global social media statistics reveal that Twitter is one of the most popular social media platforms with 353 million monthly active users worldwide as of Q3 of 2020. Statistics[3] also reveal that as of January 2021, India ranked third based on number of Twitter users with a total of 17.5 million users. Moreover in 2019, Twitter localised the web UI experience in seven Indian regional languages including Hindi, Gujarati, Marathi, Urdu, Tamil, Bengali and Kannada to boost its usage in India. Twitter definitely serves as a suitable source of Hindi data for various text analytics tasks. Further advantages of using Twitter for building datasets for NLP tasks like sarcasm detection, are as follows:

- Twitter sets a limit to the length of any tweet to a maximum of 280 characters which means all the information required to uncover polarity incongruence, in most cases, will be contained within the tweet itself.
- Twitter gives us the choice of narrowing down the search field by specifying definitive keywords with the help of hashtags, Twitter handler's name, timestamp etc.
- The availability of numerous APIs for efficient tweet scraping.

The Sarc-H dataset 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.

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 performing the following tasks:

- Removal of strings beginning starting with @ to refer to a user
- Removal of URLs
- Removal of the hashtag '#'
- Removal of the string following the hashtag in case of trailing hashtags only. 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] https://www.statista.com/statistics/242606/number-of-active-twitter-users-in-selected-countries/

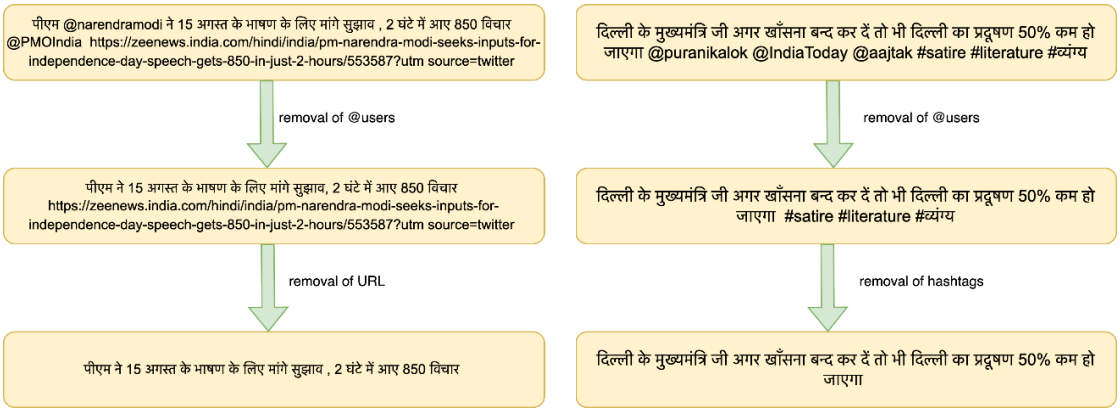

Figure 2: Pre-processing flow for two sample tweets

3.2 Embeddings

In general, an embedding is a low-dimensional, learned continuous vector representation of discrete variables into which we can translate high-dimensional vectors. In this research we use two such embeddings, namely word embedding and emoji embedding.

3.2.1 Word Embeddings

Word embeddings give us vector representations of a word in a vector space where words sharing certain semantic or syntactic relationships exist in close vicinity of each other. Such knowledge allows us to do away with manual feature engineering required to gain semantic and local contextual insight. This vector representation of words, learned using neural network models, was originally proposed by Mikolov et al. [47] but was solely for English language. 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. As an example, we consider three words, namely पुरुष (man), गधा (donkey) and महिला (woman) and calculate the similarity scores of various word pairs using the fastText Hindi word embeddings as shown in table 1.

Table 1: Similarity scores using FastText Hindi word embeddings

Word pair	Similarity Score
(पुरुष, महिला)	0.621
(पुरुष, गधा)	0.187
(खाना, पीना)	0.557
(गाना, नाचना)	0.505
(खाना, नाचना)	0.306

Now comparing the similarity scores between the pairs (पुरुष, महिला) and (पुरुष, गधा) as given in table 1, we observe that पुरुष is more related to महिला than it is to गधा. Such semantic relationships can also be observed in case of verbs. It stands to reason that eating (खाना) and drinking (पीना) and more closely related actions than eating (खाना) and dancing (नाचना). Similarly, dancing (नाचना) and singing (गाना) are more probable to be used together than eating (खाना) and dancing (नाचना). The above two examples showcase the semantic dissension in the usage of words like पुरुष and गधा together or खाना and नाचना which further points to context incongruity [30]. Such discordance tends to occur quite frequently in sarcastic sentences and thus could prove to be helpful in detection.

3.2.2 Emoji Embeddings

Emojis are defined as pictograms or icons which are mainly used in digital messages to express an idea or emotion. These originated on Japanese mobile phones in 1997 and became increasingly popular worldwide in 2010s after being added to the mobile operating system. Apart from their visual appeal, there are two more reasons for their growing popularity. Firstly, any messaging application generally has an upper limit on the number of characters that one can use. In light of this constraint, emojis were of great help, as a lot of them could represent and thus replace words, phrases and even entire sentences. Since their introduction to the digital world, the number of emojis have only increased from a few hundreds during later 1990s to a staggering 3521 emojis[4] as of October 2020. A simple example is the use of '❤' , which can be used as an representative the phrase "I love you" or "I love this" where 'this' could refer to anything depending on the context of the conversation thus reducing the number of characters by at least 6 characters (i.e 'I love'). Secondly, emojis help users to better express the emotion of an otherwise toneless sentence. The intended sentiment or tone is often lost in plain textual messages and thus the addition of an emoji can help the recipient of the message to correctly interpret the polarity of the message. This benefit becomes more pronounced when it comes to expressing more complex and subtle sentiments like sarcasm as it is not understood solely by the words themselves but by the tone in which they are spoken. For example, in a statement, "गर्लफ्रेंड बनाने के बाद पता चला की 100 रुपये की चॉकलेट आती है" (which means "When I had a girlfriend, I got to know that a chocolate with a price of Rs. 100 is available") seems to be neutral statement. However, the author could have his/her own reasons to despise or make fun of such expenses and in such a case, the intended tone would be sarcastic which is not apparent from the sentence. The incorporation of an emoji could help clear this ambiguity. So, had the author intended to simply express his apprised knowledge and comfort with the expense, he/she could have written "गर्लफ्रेंड बनाने के बाद पता चला की 100 रुपये की चॉकलेट आती है। 🙂". On the other hand, if the author meant the same in a sarcastic way, "गर्लफ्रेंड बनाने के बाद पता चला की 100 रुपये की चॉकलेट आती है। 😆" more clearly conveys the banter.

Thus, considering emojis can greatly boost the performance of the learning models. However, there is a need to represent the emojis in an appropriate numeric manner which can be given as input to train the models. Emoji2vec [23] 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 (Figure 2). The embedding is the sum of word embeddings of words in description

[4] https://emojipedia.org/stats/

Figure 3: Snapshot of emoji2vec emoji embedding

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. As an example, table 2 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 2: Similarity score of various emoji pairs according to emoji2vec embeddings

Emoji Pairs	Similarity Score
(😂,😆)	0.6827
(😔,😕)	0.5617
(😆,😕)	0.3619

Both, ‘face with tears of joy’, 😂 and ‘grinning squinting face’, 😆 are jovial expressions and therefore closely related as depicted by the high value of the similarity score. Similarly, ‘pensive face’, 😔 and ‘slightly frowning face’, 😕 are representative of an unhappy or gloomy emotion and therefore are in close vicinity of each other in the vector space as depicted by the high similarity score. However, 😆 and 😕 manifest contrasting emotions and thus have a lower similarity score.

Not all extracted tweets contained 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 [48] 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 the majority 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 of aforementioned 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 Hybrid CNN-LSTM Model

Convolutional neural networks (CNNs) are good at extracting local and deep features from a given text while LSTMs are able to learn the long-term dependencies in sequential data (as are words in a sentence). On the flip side, CNN is incapable to analyze differing and long length dependencies, and RNN, specifically LSTM, can step in to tackle this shortcoming owing to the presence of memory units. Thus using these together can actually complement each other having a synergic effect thus resulting in an overall improvement in the performance. So, we primarily use CNN for high-level feature extraction and these local feature sequences are then fed into LSTM, which now has finer and better features to work with rather than raw and flat textual sentences. The architecture of hybrid CNN-LSTM model is shown in Figure 4.

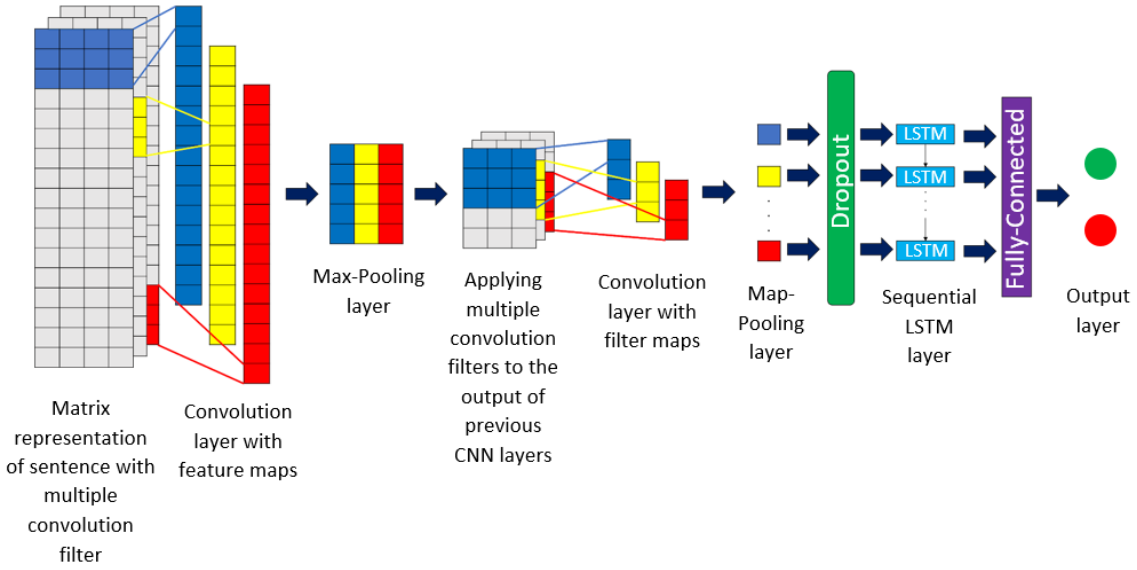

Figure 4: The hybrid CNN-LSTM model

3.3.1 CNN

Convolutional neural network (ConvNet or CNN) is a class of deep neural networks which has achieved laudable accomplishments in object detection and recognition tasks. Motivated by its immense success in the computer vision field, NLP researchers soon started to draw analogies between images and texts by acknowledging the fact that texts too are nothing but sequences of words or sentences as is an image a sequence of pixels. Once the texts have been converted to their vector representations, the operations being performed on images could similarly be performed on text sequences. The only difference would be that these text sequences would only by represented as 1-dimensional arrays and thus all the operations would need to be performed along just this one dimension. This idea of using CNN for text classification was

first proposed by Kim in 2018 [28]. CNN typically performs two operations convolution and pooling, which are primarily feature extraction methods:

- Convolution: For NLP tasks we use a 1-D CNN since any text is essentially a one-dimensional input i.e., a sequence of words. Let there be a sequence of words w1:n=w1, w2, … , wn where each word wi has a word embedding of dimension d, xi ϵRd, associated with it. Let the entire sentence be represented as x ϵRLxd, where L is the length of the sentence. A convolutional filter of size k is applied to each window in the sequence, i.e. dot product of embedding vectors in that window and filter u, followed by an activation function. Concatenation of the word embeddings of the ith window (xi) is as given in (1)

$$x_i = [w_i + w_{i+1} + \ldots + w_{i+k}] \in R_{kxd} \quad (1)$$

  where the $+$ sign represents row vector concatenation.
  The dot product of this x_i with filter u and application of an activation function, g, results in a feature map $r \in R^{L-k+1}$, where each element r_i is generated as shown in (2),

$$r_i = \mathrm{g}(x_i.\mathrm{u}) \in R \quad (2)$$

  where '.' represents element-wise multiplication.
  Usually multiple filters $u_1, u_2, \ldots, u_l$ are applied and are represented as matrix U as given in (3)

$$r_i = \mathrm{g}(x_i.\mathrm{U} + \mathrm{b}) \in R \quad (3)$$

  where b is the bias.
- Pooling: For the purpose of combination of the vectors resulting from different convolution windows into a single one-dimensional vector, pooling operation is used. This tries to reduce the size of each feature map and provide the most relevant features of the sentence/document. This is done by either taking the max or the average value observed in the resulting vector from the convolutions. We have incorporated max-pool in our experimentation.

CNN essentially provides an architecture for high level feature-extraction captured using consecutive fixed-sized windows. It is these fixed-size filters that prevent CNN to learn long-term dependencies and thus limiting its capabilities to extraction of local features and correlations only.

3.3.2 LSTM

Recurrent neural network or RNN is a class of artificial neural networks where connections between nodes form a directed graph along a sequence. It essentially forms a sequence of neural network blocks that are linked to each other like a chain. This architecture allows RNN to exhibit temporal behaviour and capture sequential data taking the previous output of hidden state, $h_t - 1$ along with the current input x_t. However, during the training of an RNN, where gradients are being propagated back in time all the way to the initial layer, the gradients coming from the deeper layers have to go through continuous matrix multiplication. As they approach the earlier layers, in case of small values (<1), they shrink exponentially until they vanish and make it impossible for the model to learn. This problem is referred to as the vanishing gradient problem. On the other hand, for large values (>1) gradients get larger and eventually blow up and crash the model leading to an exploding gradient problem. This makes it hard for RNN to capture long-term dependencies in the sequential data.

Long Short-term memory or LSTM [25], 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 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 it, 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 (4) to (9):

$$i_t = \sigma(W_i[h_t - 1, x_t] + b_i) \quad (4)$$

$$f_t = \sigma(W_f[h_t - 1, x_t] + b_f) \quad (5)$$

$$q_t = \tanh(W_q[h_t - 1, x_t] + b_q) \quad (6)$$

$$o_t = \sigma(W_o[h_t - 1, x_t] + b_o) \quad (7)$$

$$c_t = f_t . c_t - 1 + i_t \odot q_t \quad (8)$$

$$h_t = o_t \odot \tanh c_t \quad (9)$$

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 ⊙ denotes element wise multiplication.

Figure 5 shows the structure of the LSTM neural network.

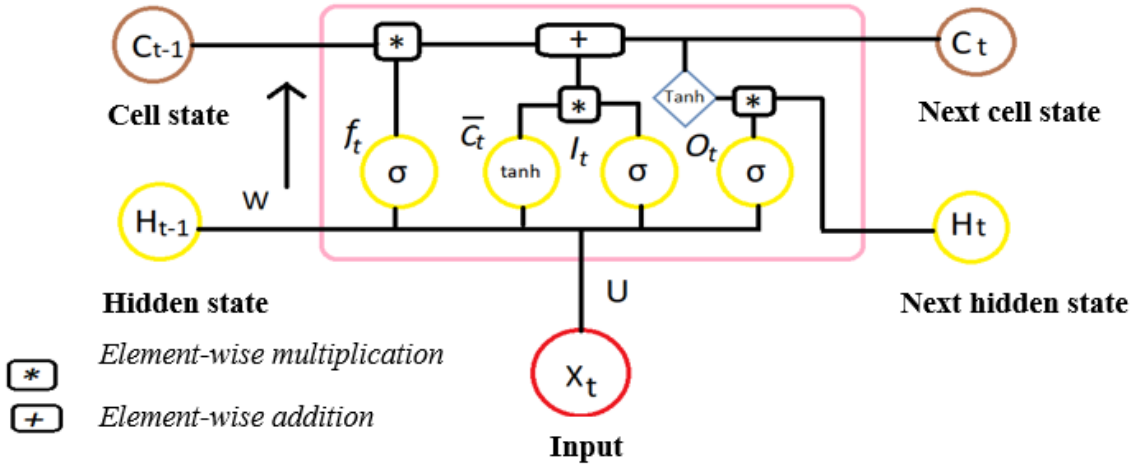

Figure 5. The structure of the LSTM neural network.

Since LSTM is designed for learning long term dependencies, it is only imperative to choose it upon the convolution layer to learn such dependencies in the obtained sequence of higher-level features.

4 RESULTS AND DISCUSSION

TweetScraper[5] was used for extracting tweets and Sarc-H dataset with a total of 1004 tweets was built with 414 tweets labelled as sarcastic and 590 labelled as non-sarcastic. The dataset, once processed suitably, was split into 70:30 training and testing datasets. Though the dataset was small but deep learning allows to easily incorporate problem-specific

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

constraints directly into the model to reduce variance and neural nets have a large library of techniques to combat overfitting. These techniques help mitigate the variance issue, while still benefitting from the flexibility and can be used to train models even if we have less samples.

A simplistic artificial neural network (ANN), CNN and LSTM were applied discretely to define the baselines for the study. Hyperparameter setting plays a significant role in the overall performance of any neural model and therefore an extensive experimentation with the same was done following data preparation. For the purpose of hyperparameter tuning, we make use of automated hyperparameter optimization that is provided by a python library, Hyperopt [49]. This obviated the need of manual tweaking of the values thus eliminating the requirement of rigorous and recurrent experimentation with each possible combination of the involved hyperparameters. Since four different models were experimented with, the tuning of each model's respective hyperparameters was performed. Table 3 shows the investigated hyperparameters values.

Table 3: Investigated hyperparameter values

Hyperparameters	Investigated values
Number of nodes in each layer (ANN, CNN and LSTM)	16, 32, 64, 128
Optimizer (ANN, CNN, LSTM)	rmsprop, adam, sgd
Batch size (ANN, CNN and LSTM)	32, 64
Filter size (for CNN)	2,3,4,10
Max pooling size (for CNN)	2,3,4
Dropout (only after both layers of CNN)	(0,1)
Activation function	Rectified Linear Function

Following hyperparameter setting, the models were evaluated using four performance metrics: accuracy, precision, recall and F-score. Table 3, 4, 5 and 6 depicts the confusion matrix for ANN, CNN, LSTM and hybrid CNN+LSTM respectively. As a primary research objective was to probe the significance of emojis in textual conversations for sarcasm detection, the results were evaluated with and without emojis. The tables include the matrices for both the cases.

Table 4: Confusion matrix for ANN

Without Emoji		
	Non-Sarcastic	Sarcastic
Non-Sarcastic	153	10
Sarcastic	22	117
With Emoji		
	Non-Sarcastic	Sarcastic
Non-Sarcastic	153	10
Sarcastic	12	127

Table 5: Confusion matrix for CNN

Without Emoji		
	Non-Sarcastic	Sarcastic
Non-Sarcastic	152	11
Sarcastic	15	124
With Emoji		

	Without Emoji	
	Non-Sarcastic	Sarcastic
	Non-Sarcastic	Sarcastic
Non-Sarcastic	159	4
Sarcastic	7	132

Table 6: Confusion matrix for LSTM

	Without Emoji	
	Non-Sarcastic	Sarcastic
Non-Sarcastic	150	13
Sarcastic	18	121
	With Emoji	
	Non-Sarcastic	Sarcastic
Non-Sarcastic	155	8
Sarcastic	4	135

Table 7: Confusion matrix for CNN-LSTM

	Without Emoji	
	Non-Sarcastic	Sarcastic
Non-Sarcastic	154	9
Sarcastic	16	123
	With Emoji	
	Non-Sarcastic	Sarcastic
Non-Sarcastic	161	2
Sarcastic	6	133

Table 8 presents the performance comparison of all the models using the evaluation metrics with and without emojis. The hybrid model outperforms all the models, achieving an accuracy of 91.72% and F-score of 0.9077 in absence of emojis and 97.35% accuracy and 0.9708 F-score with emojis. ANN, on the other hand, falls on the lower end of the spectrum, achieving the lowest, though an appreciable accuracy of 89.40% and F-score of 0.8797.

Table 8: Performance comparison of all models

Model	Without Emojis				With Emojis			
	Precision	Recall	F-Score	Accuracy	Precision	Recall	F-Score	Accuracy
ANN	0.9212	0.8417	0.8797	0.8940	0.9270	0.9137	0.9203	0.9272
LSTM	0.9030	0.8705	0.8864	0.8974	0.9440	0.9712	0.9574	0.9603
CNN	0.9185	0.8921	0.9051	0.9139	0.9706	0.9496	0.9600	0.9636
CNN-LSTM	0.9318	0.8849	0.9077	0.9172	0.9852	0.9568	0.9708	0.9735

In terms of model training time, it was noted that, on an average ANN took the least amount of time, followed by CNN and the hybrid model whereas LSTM took the maximum time. The ROC curves for the hybrid model with emojis and without emojis is shown in fig.6 and fig.7 respectively.

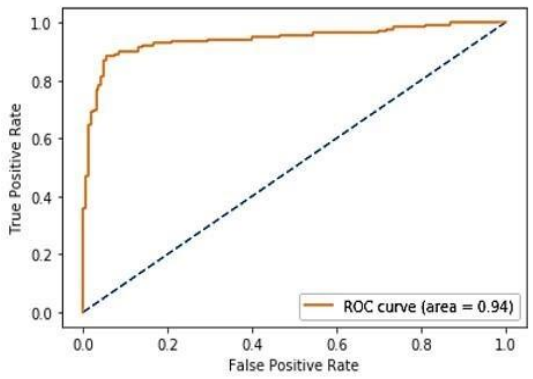

Figure 6. ROC curve of CNN-LSTM without emojis

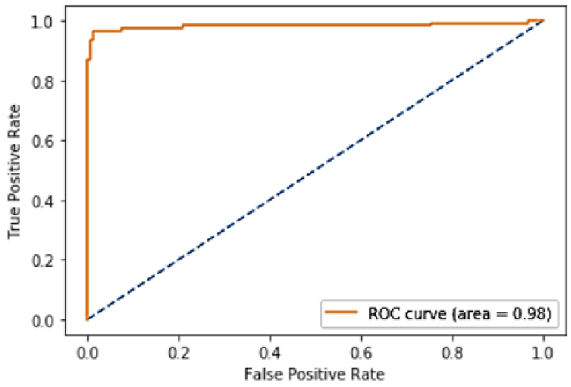

Figure 7. ROC curve of CNN-LSTM with emojis

Table 9 depicts the values of hyperparameters for optimal performance chosen by Hyperopt.

Table 9: Model-wise Optimal hyperparameter values

Model	Hyper-parameter	Without Emojis	With Emojis
ANN	Number of layers	1	1
	Number of nodes in a layer	16	32
	Batch size	64	32
	Optimizer	adam	adam
CNN	Number of layers	2	2
	Number of nodes in layer1, layer2	128,128	32, 16
	Filter size for layer1, layer2	10,10	4,4
	Max pooling for layer1, layer2	4,4	4,4
	Batch size	32	32
	Optimizer	rmsprop	adam
	Dropout value	0.221	0.308
LSTM	Number of layers	1	2
	Number of nodes in a layer	32	32,32
	Batch size	128	32
	Optimizer	adam	adam
CNN-LSTM	Number of layers of CNN and LSTM	1,1	2,1
	Number of nodes (CNN)	64	64,128
	Filter size for CNN layer	10	10,10
	Max pooling for CNN layer	4	4,2
	Dropout value	0.022	0.439
	Number of nodes in a layer LSTM)	64	64
	Number of nodes in a layer (FC)	16	128
	Batch size	32	32
	Optimizer	adam	rmsprop

As far as the dimensions of the utilized emojis embedding is concerned we used the default value which was 300. As for both word as well as emoji embeddings the dimensions were set to 300 each, on concatenation it resulted in

600-dimensional representations. Additional experimentation was performed to analyze the change in performance when embeddings of lower dimensions were utilized. For the purpose we implemented the embedding dimensionality reduction algorithm provided by Raunak et al. [50] and obtained embeddings of dimensions 50, 100 and 200 for both Hindi words and emojis. The hybrid model was tested once again with the embeddings of reduced dimensionality. Table 10 presents the results and it is observed that the best performance was achieved by opting a 300-dimensional representation of each of the embedding type.

Table 10: Variation in performance of hybrid model with embedding matrix of varying dimensionality

Dimension of embedding matrix (word embedding dimension + emoji embedding dimension)	Precision	Recall	F-Score	Accuracy
100 (50+50)	0.9845	0.9137	0.9478	0.9536
200 (100+100)	0.9565	0.9496	0.9531	0.9570
400 (200+200)	0.9706	0.9496	0.9600	0.9636
600 (300+300)	0.9852	0.9568	0.9708	0.9735

The results were also compared with the existing work done on the basis of performance metrics. Table 11 presents the comparison along with the details of dataset, features and techniques utilized in each study.

Table 11: Comparison of proposed CNN-LSTM hybrid model with existing state-of-the-art

Study	Dataset	Features	Technique	Performance Metrics			
				Precision	Recall	F-Score	Accuracy
Bharti et al., 2015, [51]	2000 one liner Hindi news obtained from online sources with 5000 related Hindi tweets	1.Count of positive and negative keywords 2.Polarity score in case of clash, using HindiSentiWordNet	Keyword matching in an online news-based context aware framework	0.736	0.717	0.726	0.794
Desai & Dave, 2016, [18]	Self-generated 1410 Hindi sentences	1.TF- IDF 2.Positive and negative scoring of words using HindiSentiWordNet 3.Boolean feature to denote presence of #Kataksh 4.Intensity of emoticons, if present	Machine Learning model- LibSVM	-	-	-	0.837
Bharti et al., 2017 [52]	1000 Hindi tweets	Not Publically Available	Context-based in terms of contradiction between tweet and its related news in the same timestamp	0.848	0.836	0.842	0.87
Bharti et al., 2018, [17]	500 Hindi news from online sources along with 2500 tweets with same timestamp	1.Triplet extraction of <subject, verb, object> using Rusu_Triplets extraction algorithm by	Comparison of tweets and its temporal facts using generated key-value pairs	0.834	0.831	0.832	0.824

Study	Dataset	Features	Technique	Performance Metrics			
				Precision	Recall	F-Score	Accuracy
		taking news as temporal facts 2.Generation of key-value pairs from the triplets					
Katyayan et al. , 2021 [53]	1000 sentences from social media like Facebook, Instagram, and Twitter	1. POS tagging 2. bag-of-words	Four separate machine learning techniques: Naive Bayes SVM Decision tree Neural Network	0.72 (highest among all, achieved by decision tree)	0.45 (highest among all, achieved by neural network)	0.54 (highest among all for both decision tree and neural network	-
Proposed CNN-LSTM model	1004 tweets extracted using hashtags and handle names	1.Hindi embedding for words 2.Emoji embedding for emojis	ANN, CNN, LSTM, CNN-LSTM (Best performance CNN-LSTM)	**0.985**	**0.957**	**0.971**	**0.973**

As evident, the previous studies were heavily reliant on manual feature extraction. However, the use of embeddings obviates the need for such feature engineering techniques as embeddings are highly competent in understanding the semantic and syntactic information present in any given sentence. Furthermore, not enough importance has been given to emojis in the previous works and, as explained in earlier sections, they can prove to be quite pivotal in the determination of the true sentiment of any tweet. Our work makes use of this important attribute, once again in the form of embeddings. Additionally, utilizing deep learning models further trivializes the need of explicit feature extraction techniques as these models are highly skillful and fast in retrieval of essential features and patterns by themselves. Thus, the utilization of embeddings along with these efficient deep learning models has given our proposed approach an edge over the previously proffered methodologies.

5 CONCLUSION AND FUTURE WORK

With the accelerated use of social listening tools for market intelligence, research on sentiment analysis technologies has gained momentum. Sarcasm is a pivotal natural language challenge to analyze sentiments accurately as most text-based conversation are 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. 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 CNN-LSTM model to detect sarcasm. The model was validated on a Sarc-H dataset created for the purpose for detecting sarcasm detection in Hindi. The hybrid CNN-LSTM performed superiorly with 97.35% accuracy, 0.9852 F-score, 0.9708 precision and 0.9508 recall. On comparison with the existing works too, the CNN-LSTM hybrid demonstrated superlative results. The experiments clearly indicate that emojis add clarity to the written content and their use improve the sarcasm classifier performance. Also, the use of embeddings address problems like word semantics, context and data sparsity for sarcasm detection in a low-resource language like Hindi.

The current work has been done on a small dataset which certainly is a limitation. However, we intend to increase our dataset in the future and employ models that generalize better and learn more high-level features with the help of contextualized word embeddings. Further, as the use of sarcasm in dialogues and conversational threads have further added to the challenges making it vital to capture the knowledge of the domain of discourse, context propagation during the course of dialogue as well as situational context and tone of the speaker, we intend to build models that predict sarcasm in Hindi conversational data using its chronological nature.

REFERENCES

[1] Cambria, E., Schuller, B., Xia, Y., & Havasi, C. (2013). New avenues in opinion mining and sentiment analysis. IEEE Intelligent systems, 28(2), 15-21.

[2] Kumar, A., Srinivasan, K., Cheng, W. H., & Zomaya, A. Y. (2020). Hybrid context enriched deep learning model for fine-grained sentiment analysis in textual and visual semiotic modality social data. Information Processing & Management, 57(1), 102141.

[3] Kumar, A. (2021). Contextual semantics using hierarchical attention network for sentiment classification in social internet-of-things. Multimed Tools Appl https://doi.org/10.1007/s11042-021-11262-8

[4] 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.

[5] Akhtar, M. S., Ekbal, A., & Cambria, E. (2020). How intense are you? Predicting intensities of emotions and sentiments using stacked ensemble [application notes]. IEEE Computational Intelligence Magazine, 15(1), 64-75.

[6] Poria, S., Chaturvedi, I., Cambria, E., & Hussain, A. (2016, December). Convolutional MKL based multimodal emotion recognition and sentiment analysis. In 2016 IEEE 16th international conference on data mining (ICDM) (pp. 439-448). IEEE.

[7] 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.

[8] Ghosh, D., Guo, W., & Muresan, S. (2015, September). Sarcastic or not: Word embeddings to predict the literal or sarcastic meaning of words. In proceedings of the 2015 conference on empirical methods in natural language processing (pp. 1003-1012).

[9] Kumar, A., Dikshit, S., & Albuquerque, V. H. C. (2021). Explainable Artificial Intelligence for Sarcasm Detection in Dialogues. Wireless Communications and Mobile Computing, 2021.

[10] Kumar, A., & Garg, G. (2020). The multifaceted concept of context in sentiment analysis. In Cognitive Informatics and Soft Computing (pp. 413-421). Springer, Singapore.

[11] Bliss-Carroll, N. L. (2016). The nature, function, and value of emojis as contemporary tools of digital interpersonal communication.

[12] 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.

[13] 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.

[14] 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).

[15] 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.

[16] 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

[17] 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.

[18] Desai, N., & Dave, A. D. (2016). Sarcasm detection in Hindi sentences using support vector machine. International Journal, 4(7), 8-15.

[19] 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.

[20] 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.

[21] Kumar, A., & Jaiswal, A. (2020). Deep learning based sentiment classification on user-generated big data. Recent Advances in Computer Science and Communications (Formerly: Recent Patents on Computer Science), 13(5), 1047-1056.

[22] Kumar, A., & Sachdeva, N. (2020). Multi-input integrative learning using deep neural networks and transfer learning for cyberbullying detection in real-time code-mix data. Multimedia Systems, 1-15.

[23] Eisner, B., Rocktäschel, T., Augenstein, I., Bošnjak, M., & Riedel, S. (2016). emoji2vec: Learning emoji representations from their description. arXiv preprint arXiv:1609.08359.

[24] Gu J, Wang Z, Kuen J, Ma L, Shahroudy A, Shuai B, Liu T, Wang X, Wang G, Cai J, Chen T. Recent advances in convolutional neural networks.

Pattern Recognition. 2018 May 1; 77:354-77.

[25] Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural computation, 9(8), 1735-1780.

[26] Tepperman, J., Traum, D., & Narayanan, S. (2006). " Yeah Right": Sarcasm Recognition for Spoken Dialogue Systems. In Ninth international conference on spoken language processing.

[27] Kreuz, R., & Caucci, G. (2007, April). Lexical influences on the perception of sarcasm. In Proceedings of the Workshop on computational approaches to Figurative Language (pp. 1-4).

[28] González-Ibánez, R., Muresan, S., & Wacholder, N. (2011, June). Identifying sarcasm in Twitter: a closer look. In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (pp. 581-586).

[29] Riloff, E., Qadir, A., Surve, P., De Silva, L., Gilbert, N., & Huang, R. (2013, October). Sarcasm as contrast between a positive sentiment and negative situation. In Proceedings of the 2013 conference on empirical methods in natural language processing (pp. 704-714).

[30] Liebrecht, C. C., Kunneman, F. A., & van Den Bosch, A. P. J. (2013). The perfect solution for detecting sarcasm in tweets# not.

[31] Joshi, A., Sharma, V., & Bhattacharyya, P. (2015, July). Harnessing context incongruity for sarcasm detection. In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers) (pp. 757-762).

[32] Joshi, A., Tripathi, V., Bhattacharyya, P., & Carman, M. (2016, August). Harnessing sequence labeling for sarcasm detection in dialogue from tv series 'friends'. In Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning (pp. 146-155).

[33] Kumar, A., & Garg, G. (2020). Sarcasm Detection Using Feature-Variant Learning Models. In Proceedings of ICETIT 2019 (pp. 683-693). Springer, Cham.

[34] Schifanella, R., de Juan, P., Tetreault, J., & Cao, L. (2016, October). Detecting sarcasm in multimodal social platforms. In Proceedings of the 24th ACM international conference on Multimedia (pp. 1136-1145).

[35] Castro, S., Hazarika, D., Pérez-Rosas, V., Zimmermann, R., Mihalcea, R., & Poria, S. (2019). Towards multimodal sarcasm detection (an _Obviously_ perfect paper). arXiv preprint arXiv:1906.01815.

[36] Kumar, A., & Garg, G. (2019, March). Sarc-m: Sarcasm detection in typo-graphic memes. In International Conference on Advances in Engineering Science Management & Technology (ICAESMT)-2019, Uttaranchal University, Dehradun, India.

[37] Joshi, A., Tripathi, V., Patel, K., Bhattacharyya, P., & Carman, M. (2016). Are word embedding-based features useful for sarcasm detection?. arXiv preprint arXiv:1610.00883.

[38] Ghosh, A., & Veale, T. (2016, June). Fracking sarcasm using neural network. In Proceedings of the 7th workshop on computational approaches to subjectivity, sentiment and social media analysis (pp. 161-169).

[39] Joshi, A., Bhattacharyya, P., & Carman, M. J. (2017). Automatic sarcasm detection: A survey. ACM Computing Surveys (CSUR), 50(5), 1-22.

[40] Kumar, A., Sangwan, S. R., Arora, A., Nayyar, A., & Abdel-Basset, M. (2019). Sarcasm detection using soft attention-based bidirectional long short-term memory model with convolution network. IEEE access, 7, 23319-23328.

[41] 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.

[42] 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).

[43] 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.

[44] 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.

[45] 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.

[46] 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.

[47] Mikolov, T., Sutskever, I., Chen, K., Corrado, G., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. arXiv preprint arXiv:1310.4546.

[48] 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.

[49] 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.

[50] Raunak, V., Gupta, V., & Metze, F. (2019, August). Effective dimensionality reduction for word embeddings. In Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019) (pp. 235-243).

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

[52] Bharti, S. K., Babu, K. S., and Raman, R. (2017). Context-based Sarcasm Detection in Hindi Tweets. In Ninth International Conference on Advances in Pattern Recognition (ICAPR), 2017.

[53] Katyayan, P., & Joshi, N. (2021). Performance evaluation of machine learning algorithms for detecting Hindi sarcasm. In 2021 Bharatiya Vaigyanik

evam Audyogik Anusandhan Patrika 29(1) (pp. 43-48).