Final version of the accepted paper. Cite as: *Ashraf Kamal and Muhammad Abulaish, An LSTM-Based Deep Learning Approach for Detecting Self-Deprecating Sarcasm in Textual Data, In Proceedings of the 16th International Conference on Natural Language Processing (ICON), Hyderabad, India, pp. 1-10, December 18-21, 2019.*

An LSTM-Based Deep Learning Approach for Detecting Self-Deprecating Sarcasm in Textual Data

Ashraf Kamal
Department of Computer Science
Jamia Millia Islamia, (A Central University)
New Delhi, India
ashrafkamal.mca@gmail.com

Muhammad Abulaish
Department of Computer Science
South Asian University
New Delhi, India
abulaish@sau.ac.in

Abstract

Self-deprecating sarcasm is a special category of sarcasm, which is nowadays popular and useful for many real-life applications, such as brand endorsement, product campaign, digital marketing, and advertisement. The self-deprecating style of campaign and marketing strategy is mainly adopted to excel brand endorsement and product sales value. In this paper, we propose an LSTM-based deep learning approach for detecting self-deprecating sarcasm in textual data. To the best of our knowledge, there is no prior work related to self-deprecating sarcasm detection using deep learning techniques. Starting with a filtering step to identify self-referential tweets, the proposed approach adopts a deep learning model using LSTM for detecting self-deprecating sarcasm. The proposed approach is evaluated over three Twitter datasets and performs significantly better in terms of *precision*, *recall*, and *f-score*.

1 Introduction

Over a decade, the popularity of the micro-blogging platform, Twitter, has significantly increased for analyzing its content for varied real-world applications. The information extracted from Twitter can shed light on numerous applications, such as text categorization, sentiment analysis, election campaign and result prediction, open-source intelligence, and event detection. However, the contents available on Twitter in the form of tweets are short and limited to maximum 280 characters. Moreover, tweets are informal and mainly consist of misspelled words, slangs, bashes, acronyms, shortened words, non-literal unstructured phrases, and emoticons. Due to existence of such volumunious informal texts in the form of tweets text information processing is a challenging task. Moreover, analysis of the tweets has become more challenging due to presence of figurative language, especially sarcasm. The main role of a sarcastic tweet is to reverse the actual polarity and alter the literal semantics. However, the computational detection of sarcasm benefits many applications, especially opinion mining and sentiment analysis systems (Bouazizi and Ohtsuki, 2015).

The online Macmillan dictionary defines sarcasm[1] as "*the activity of saying or writing the opposite of what you mean, or of speaking in a way intended to make someone else feel stupid or show them that you are angry*". Sarcasm is the most seen figurative language category over online social media platforms. The presence of sarcasm in tweets is dramatically rising and computational detection of sarcasm is a challenging and interesting task. It is widely covered by researchers in recent years, but the study on different categories[2] of sarcasm, such as self-deprecating sarcasm, is very limited. Self-deprecating sarcasm[3] is a special category of sarcasm in which users mainly apply sarcasm over themselves using disparage, ridicule, and contemptuous remarks in a sarcastic style using humor. It is defined as a "sarcasm that plays off of an exaggerated sense of worthlessness and inferiority". For example, the phrase *love going to the office on Sunday* in the text "*Really, I always love going to the office on Sunday*" represents a self-deprecating sarcasm.

Nowadays, self-deprecating sarcasm has become a new style of product marketing and campaign strategy. It is mainly used for product endorsement purposes. This new marketing and campaign strategy is mainly used to excel the busi-

[1] https://bit.ly/2WsUkUk (last accessed on 15-Nov-19)

[2] https://literarydevices.net/sarcasm/ (last accessed on 15-Nov-19)

[3] https://bit.ly/2vwjtid (last accessed on 15-Nov-19)

ness growth, but without losing the brand value (Kamal and Abulaish, 2019). The main aim of this strategy is to draw the attention of the customer towards the brand. As per the American marketing association[4], "self-deprecating advertising means consumers can see a different side to brands, making them more relatable and down-to-earth". Interestingly, after an in-depth analysis of tweets, we found that there are many tweets in which users refer themselves. We consider such tweets as *self-referential* or *self-deprecating*. For example, "*Really, I just love it*" is a self-referential tweet. Our analysis further reveals that some of the self-referential tweets are self-deprecating using sarcasm, i.e., in these tweets users undervalue, criticize, insult, and disparage themselves using sarcastic phrases. We consider all such self-referential tweets as self-deprecating sarcasm.

In this paper, we propose a deep learning approach using `Long Short-Term Memory` (LSTM) to detect self-deprecating sarcasm in textual data like tweets. Initially, after preprocessing, we first identify self-referential tweets from the dataset based on a set of patterns, and rest of tweets are filtered out. The main motivation behind the filtration of the non-self-referential tweets is to increases the overall efficiency of the self-deprecating sarcasm detection process. In brief, the main role of the self-referential tweets identification module can be summarized as follows:

- Identification of explicit self-referential tweets: After an in-depth analysis across all the datasets, we identify a set of patterns followed by the self-referential tweets. Table 1 presents a set of regular expression based patterns and it is categorized as *specific patterns* and *generic patterns*. The specific patterns are based on tags and tokens present in the tweet which indicate self-referential nature of the tweet. On the other hand, *generic patterns* are based on the presence of first person singular/plural personal pronoun. These patterns are found as strong indicator of self-referential tweets. We consider such self-referential tweets as explicit, otherwise implicit.

- Identification of clusters from explicit self-referential tweets: We identify explicit self-referential tweets clusters based on overlapping contents (i.e., tri-grams) and using Jaccard similarity between the explicit self-referential tweets.

- Pattern-mining from clusters: Once the explicit self-referential tweets clusters are identified, we fetch the most frequent substring (i.e, tri-gram) as a referential pattern from each cluster.

- Identification of implicit self-referential tweets: If an implicit tweet matches with the referential pattern of any cluster, then it is considered as a self-referential tweet.

- Merge with explicit tweets: Finally, all identified implicit self-referential tweets are merged with explicit tweets to generate a list of the self-referential tweets.

Once the list of self-referential tweets is generated, it is passed to the model learning and classification module for self-deprecating sarcasm detection. To this end, each self-referential tweet is converted into an input vector, it is fed to pre-trained `GloVe` word embedding, and model learning and classification task is accomplished using LSTM for detecting self-deprecating sarcasm.

This remainder of this paper is organized as follows. Section 2 presents a brief review of the state-of-the-art techniques and approaches for computational sarcasm detection. It also highlights the uniqueness of our proposed approach over the existing state-of-the-art techniques. Section 3 presents the functional details of the proposed approach, including model learning and classification using LSTM. Section 4 presents the experimental and evaluation results. Finally, section 5 concludes the paper and discusses future research directions.

2 Related Work

Automatic sarcasm detection is considered as a classification task (Zhang et al., 2016), and the main task is to classify any piece of texts as sarcasm or non-sarcasm. Tsur et al. (2010) applied semi-supervised approach to detect sarcasm in Amazon product reviews. Davidov et al. (2010) applied the same approach to detect sarcasm in tweets and product reviews. González-Ibánez et al. (2011) considered lexical and pragmatics features to detect sarcasm on Twitter datasets.

[4] `https://bit.ly/2EEuQGQ` (last accessed on 15-Nov-19)

Riloff et al. (2013) identified sarcastic contrast-based patterns and considered words with positive sentiment and negative phrases in a tweet containing sarcasm. Liebrecht et al. (2013) discussed the role of hyperrbole in sarcasm detection. Ptácek et al. (2014) detected sarcasm in English and Czech tweets. Bharti et al. (2015) proposed rule-based algorithms based on some patterns for sarcasm detection. They also highlighted the importance of hyperbole in sarcastic texts. Bamman and Smith (2015) extracted extra-linguistic information based on the context of the instances for sarcasm detection.

Rajadesingan et al. (2015) applied three machine learning classifiers – Support Vector Machine (SVM), logistic regression, and decision tree for sarcasm detection, considering the behavioral modeling-based approach. Ghosh et al. (2015) proposed SemEval-2015 (task-11) and considered sarcasm, irony, and metaphor for sentiment analysis in Twitter data. Joshi et al. (2015) discussed the role of incongruity for sarcasm detection. Bouazizi and Ohtsuki (2016) considered a pattern-based approach. Mishra et al. (2016) considered lexical- and contextual-based features. Joshi et al. (2016) proposed word-embedding related features using `Word2Vec`[5].

Recently, deep learning models have been used as a popular technique for sarcasm detection problem. Zhang et al. (2016) applied a bi-directional gated recurrent neural network for sarcasm detection. They considered syntactic and semantic information and extracted contextual features. Amir et al. (2016) applied content- and user embedding-based Convolutional Neural Network (CNN) model. Ghosh and Veale (2016) considered CNN, LSTM, and Deep Neural Network (DNN) for sarcasm detection. Poria et al. (2016) considered features, such as sentiment, emotion, and personality and applied SVM and CNN classifiers. Tay et al. (2018) considered attention-based neural model for sarcasm detection. Hazarika et al. (2018) proposed a contextual sarcasm detector using CNN-based textual model in which context and content related information are used for sarcasm detection. Recently, Dubey et al. (2019a) converted sarcastic texts into non-sarcastic interpretation using encoder-decoder, attention, and pointer generator architectures. Dubey et al. (2019b) detected sarcasm in numerical portion of tweets using CNN and attention network.

Though sarcasm detection is widely covered by the researchers, studies related to the varied categories of sarcasm are still not explored. Recently, Abulaish and Kamal (2018) noticed the use of self-deprecating sarcasm in Twitter, mainly for the purpose of brand endorsement and sales campaign. They considered self-deprecating sarcasm as a special category of sarcasm in which users express sarcasm over themselves. They also proposed a rule-based and machine learning-based approach for detecting self-deprecating sarcasm detection in Twitter. The proposed work in this paper is new LSTM-based deep learning approach for self-deprecating sarcasm detection in textual data.

3 Proposed Approach

In this section, we discuss the proposed LSTM-based deep learning approach for self-deprecating sarcasm detection. Figure 1 presents the workflow of the proposed approach. It can be seen from this figure that besides data crawling and data pre-processing, the main functionalities of the proposed approach are self-referential tweets detection, and self-deprecating sarcasm detection using deep learning technique. Further details about all functional modules are presented in the following sub-sections.

3.1 Data Crawling

The data crawling module aims to retrieve English tweets using Twitter's `REST` API and it is implemented in `Python 2.7`. We have considered tweet ids provided as a part of two benchmark datasets – Ptácek et al. (2014) and SemEval-2015[6] to curate tweets using our data crawling module. In addition, we have also created our own Twitter dataset containing tweets crawled for the period 1st April 2019 to 19th May 2019.

3.2 Data Pre-Processing

The data pre-processing module aims to apply various pre-processing tasks on the curated tweets to produce fine-grained data for self-deprecating sarcasm detection. The pre-processing consists of data cleaning (removal of dots, retweets, numbers, hashtags, emoticons, @mention, URL's, am-

[5] `https://code.google.com/archive/p/word2vec/` (last accessed on 15-Nov-19)

[6] `https://bit.ly/34OnGgB` (last accessed on 15-Nov-19)

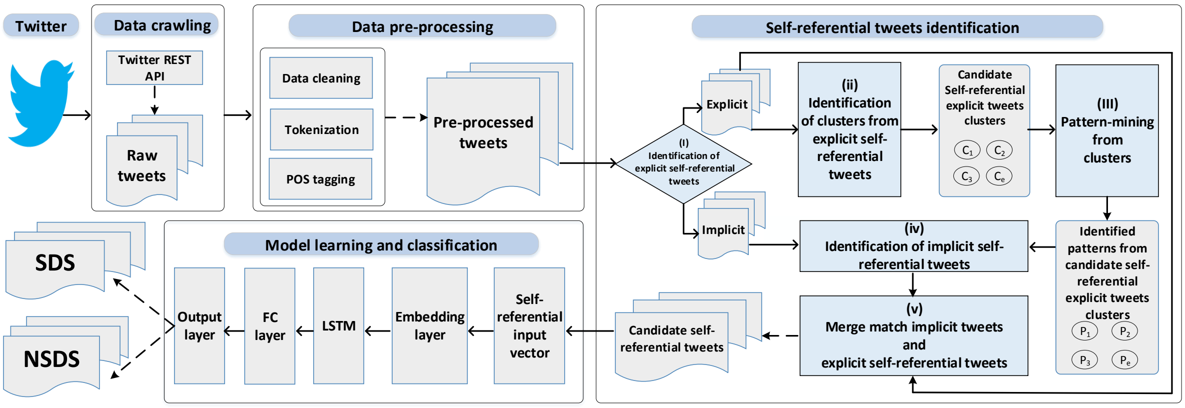

Twitter
Data crawling
Twitter REST API
Raw tweets
Data pre-processing
Data cleaning
Tokenization
POS tagging
Pre-processed tweets
Self-referential tweets identification
Explicit
(I) Identification of explicit self-referential tweets
Implicit
(ii) Identification of clusters from explicit self-referential tweets
Candidate Self-referential explicit tweets clusters
C1 C2 C3 Ce
(III) Pattern-mining from clusters
Identified patterns from candidate self-referential explicit tweets clusters
P1 P2 P3 Pe
(iv) Identification of implicit self-referential tweets
(v) Merge match implicit tweets and explicit self-referential tweets
Candidate self-referential tweets
SDS
NSDS
Model learning and classification
Output layer
FC layer
LSTM
Embedding layer
Self-referential input vector

Figure 1: Work-flow of the proposed approach

persands, double quotes, and extra white spaces) and lower-case conversion. Thereafter, spacy[7] is used to tokenize the tweets and generate POS tags for each token.

3.3 Self-Referential Tweets Identification

After an in-depth analysis of the datasets it is observed that all tweets are not self-referential or self-deprecating in nature. To this end, this module presents a filtration mechanism to generate a corpus of self-referential tweets. The non-self-referential tweets are filtered from further consideration because they rarely contain a self-deprecating sarcasm. Motivated by Zhao et al. (2015), identification of self-referential tweets is performed using the following sequence of steps.

(i) ***Identification of Explicit Self-Referential Tweets:***
In this step, we identify the self-referential tweets that have explicit pattern in the text and these tweets are considered for further processing to mine implicit patterns (signals) of self-referential behavior in tweets. The explicit self-referential tweets have certain patterns, which can be defined using the regular expressions given in Table 1. The tweets from the pre-processed corpus are matched using these regular expressions to identify the explicit self-referential tweets. The pattern for explicit nature of self-referential tweets are of two types – *specific* and *generic*.

Patterns	Category
$UH\ (i \mid my)$	Specific
$(we \mid i)$ [love] $(it \mid when)$	Specific
$when\ (my \mid our)$	Specific
$(am \mid are)\ [still]$	Specific
$(i \mid my \mid me \mid mine \mid myself)$	Generic
$(we \mid are \mid us \mid our \mid ourselves)$	Generic

Table 1: Regular expressions to identify explicit self-referential tweets

The *specific patterns* are based on either sequential order of tokens and tags, or sequential order of tokens. If any of the specific pattern from table 1 founds in the pre-processed tweets, then it is added to the explicit set, otherwise it is checked further from *generic patterns*. The *generic patterns* are based on the first person singular/plural personal pronoun, such as 'i', 'we', and their objective and possessive cases, such as 'my', 'me', '$mine$', '$myself$', 'are', 'our', 'us', and '$ourselves$'. The first person singular/plural personal pronoun and its grammatical variants are strong indicator for a tweet to be referred as self-referential.

If any of the token from the pre-processed tweet matches with any *generic patterns*, then such tweet is considered as explicit self-referential tweet, and added to the explicit set of self-referential tweets, E_s. Otherwise, the tweet is added to the set of implicit tweets, I_t. Further, the identified explicit tweets are modeled as a undirected weighted graph and given to a clustering algorithm for further processing, which is defined in the next step.

(ii) ***Identification of Clusters from Explicit Self-Referential Tweets:***
This step clusters the tweets in E_s to identify the near-duplicate (similar) explicit self-referential tweets. To this end, first E_s tweets are modeled

[7] https://spacy.io/ (last accessed on 15-Nov-19)

as an undirected graph, where each node of the graph represents a tweet and edge represents the similarity between the underlying pair of nodes. The similarity between two tweets (nodes), say t_i and t_j, is calculated using Jaccard coefficient to observe the overlapping set of tri-grams between the tweets, as defined in equation 1, where T_i and T_j represents the set of tri-grams for tweets t_i and t_j, respectively. We choose tri-grams in our experiment because self-deprecating phrases in a tweet generally contain at least three words. We create an edge between a pair of near-duplicate tweets if the Jaccard similarity based on set of tri-grams is greater than a threshold 0.6 as defined in (Zhao et al., 2015). Thereafter, `depth first search` algorithm is applied on the constructed graph to extract clusters (connected components), where each cluster represents the set of identical explicit self-referential tweets. The extraction process only extract clusters having atleast three tweets.

$$J(t_i, t_j) = \frac{|T_i \cap T_j|}{|T_i \cup T_j|} \tag{1}$$

(iii) ***Pattern-Mining from Clusters:***
Following the cluster identification process in the previous step, this step mines frequent patterns from the extracted clusters. To this end, the occurrence probability of every pattern of each cluster is computed and patterns having probability greater than 0.8 are regarded as patterns. For example, if a cluster has 5 tweets and a tri-gram “great way start” occurs in four out of 5 tweets, then it can be regarded as a frequent pattern (tri-gram). This procedure is repeated for every pattern in each cluster to extract the list of frequent patterns. Thereafter, the duplicate frequent patterns identified from two or more clusters are filtered to generate unique set of frequent patterns P.

(iv) ***Identification of Implicit Self-Referential Tweets:***
The first step of this whole procedure held tweets which have no explicit pattern as self-referential tweets, called *implicit tweets*. This step will improve the recall of the self-referential tweets identification process. This step matches the identified patterns from previous step in *implicit tweets* to extract implicit self-referential tweets. To this end, first an implicit tweet is tokenized in to tri-grams and thereafter these set of tri-grams are matched with the set of frequent patterns P using Jaccard similarity. Finally, a tweet that has Jaccard similarity greater than a threshold 0.6 is considered as a implicit self-referential tweets. This procedure is repeated for every tweets of I_t to generate a set of implicit self-referential tweets, I_s. For example, table 2 presents 3 example implicit self-referential tweets identified from I_t.

Pattern matched implicit self-referential tweets
1. great way start nothing.
2. waking with stomach pains best way start day.
3. battling cousin always great way end day.

Table 2: Implicit self-referential tweets identified from I_t

(v) ***Merging of Implicit and Explicit Tweets:***
Finally, in this step, the identified implicit self-referential tweets are added to the set of explicit self-referential tweets to generate a final set of self-referential tweets i.e. $S = E_s \cup I_s$. In the remaining paper, this curated corpus of self-referential tweets is used for experimental evaluation.

3.4 Model Learning and Classification

In recent years, deep learning has become an emerging trend in the field of text mining and natural language processing. The semantic modeling of textual data using deep learning approaches has drawn significant attention among the research community. Various neural network-based models including CNN, LSTM, and DNN are used for diverse text modeling applications such as document classification, machine translation, speech recognition, and so on. Detection of self-deprecating sarcasm is one such application that is largely unexplored. To this end, we modeled the self-deprecating sarcasm detection as a deep-learning problem.

On analysis, it is found that the long sequence of words or phrases plays an important role to construct a self-deprecating sarcastic patterns, such as *love being ignored*, *office on sunday*, and *happy to be late* in a tweet. Therefore, to model the long-sequences based self-deprecating sarcastic pattern, LSTM seems a perfect fit. Using model learning through LSTM, a self-referential tweet is classified as a Self-Deprecating Sarcasm (SDS) or Non Self-Deprecating Sarcasm (NSDS).

A detailed discussion about the model learning and classification is presented in following subsection.

Input Layer:
In this layer, a self-referential tweet, containing n words, is given as an input. In this manner, each self-referential tweet is converted to a self-referential input vector where every word is replaced with its index value of the dictionary, i.e., $S\epsilon R^{1\times n}$. Further, each self-referential input vector is padded and converted in the matrix form. The padding is used to make every input of same length. Thereafter, padded input vector is passed to the next layer (i.e., embedding layer).

Embedding Layer:
In the padded vector from the input layer, all the words are replaced with their corresponding representation vector or embeddings. In this paper, we have used pre-trained `GloVe` 200-dimensional embeddings trained on a Twitter corpus of 27 billion tokens. As a result of this procedure, the self-referential input tweet matrix is converted to $S\epsilon R^{L\times D}$, where L is the maximum tweets length and D represents embedding dimension.Thereafter, the embedding layer output is passed to the LSTM layer.

LSTM:
Hochreiter and Schmidhuber (1997) proposed LSTM architecture, which is a type of RNN. It is easier to train an LSTM model in comparison to an RNN model. Moreover, it also overcomes the vanishing gradient problem while back propagation through time. In LSTM, the long term temporal dependencies can be easily captured between two time steps using the memory cell. Figure 2 presents the architecture of LSTM, where each memory cell consists of *input gate* i_t, *forget gate* f_t, and *output gate* o_t. These digital gates are responsible for memory update mechanism, and it acts as a function for the current input x_t and previous hidden state h_{t-1}.

An LSTM model is trained using equations 2, 3, 4, 5, 6, and 7. Equations 2 and 3 present *input* and *forget* gates, whereas equations 5, 6, and 7 present *output gate*, *new cell state*, and *hidden state*, respectively.

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i) \tag{2}$$

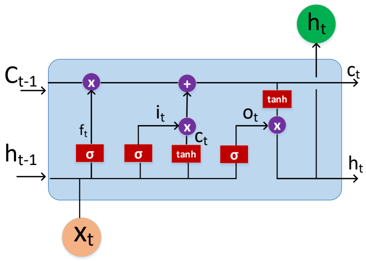


Figure 2: The architecture of LSTM

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f) \tag{3}$$

$$q_t = \tanh(W_q[h_{t-1}, x_t] + b_q) \tag{4}$$

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_0) \tag{5}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot q_t \tag{6}$$

$$h_t = o_t \odot \tanh(c_t) \tag{7}$$

In equation 4, the non-linear activation function – tanh is used to squash the value between -1 and 1, and it plays a role for cell state to forget the memory. On the other hand, non-linear activation function, sigmoid (σ) generates an output in the interval [0, 1]. LSTM works as the *gating* function for the three gates, which are discussed in the previous paragraph. Since it has a value in interval [0, 1], the information across the gates are either passed completely or not.

FC and Output Layers:
The output from the LSTM layer is passed to the fully connected dense layer followed by a sigmoid activation function. We have used binary cross-entropy as the loss function, used 40 epochs for training the model, batch-size of 256, verbose is 2, and `adam` as an optimizer. The dataset is divided into training and testing parts for experimental evaluations wherein 80% of the data is used for training and remaining 20% is used for testing procedure.

4 Experiments Setup and Results

In this section, we discuss the experimental evaluation of the proposed approach.

4.1 Experimental Settings

We have implemented the experimental setup for data crawling, data pre-processing, and self-referential tweets identification tasks in `Python 2.7`, model training and classification tasks in `Python 3.5`, and used `Keras` neural network API for LSTM model. Table 3 presents the hyper-parameters values of LSTM model used in the proposed approach.

Hyper-parameters	Value
Embedding dimension	200
Padding sequences	20
Spatial dropout (after embedding layer)	0.4
Number of neurons	256
Dropout (after LSTM layer)	0.4

Table 3: Hyper-parameters values for LSTM model used in our proposed approach

4.2 Datasets

The proposed approach is evaluated over three Twitter datasets including two benchmark datasets by Ptácek et al. (2014) and SemEval-2015. The authors released only tweet-ids for these benchmark datasets due to privacy concerns. Therefore, a crawler is developed in `Python 2.7` to curate tweets corresponding to provided tweet-ids using Twitter `REST` API. However, few tweets were deleted or protected and, as a result, we were unable to crawl all the tweets. A brief statistics about these two datasets is given in the first two rows of table 4. Apart from the two benchmark datasets, we curated a Twitter dataset from 1st April to 19th May 2019 using "#sarcasm" hashtag. We refer this dataset as `Twitter-280` and its statistical summary is given in the third row of table 4. Similarly, we crawled non-sarcastic tweets using two #not, #hate hastags. Table 5 presents the statistics of identified self-referential tweets after the *self-referential tweets identification* module. Table 6 presents the final statistics of the balanced and unbalanced datasets generated from table 5.

Datasets	#Sarcasm	#Non-sarcasm	Total (#tweets)
Ptácek et al. (2014)	53088	98195	151283
SemEval-2015	1526	2366	3892
Twitter-280	13786	14949	28735
Total (#tweets)	68400	115510	183910

Table 4: Statistics of the crawled datasets

Datasets	#Sarcasm	#Non-sarcasm	Total (#tweets)
Ptácek et al. (2014)	29580	37767	67347
SemEval-2015	761	1609	2370
Twitter-280	6971	7017	13988
Total (#tweets)	37312	46393	83705

Table 5: Statistics of identified self-referential tweets by the *self-referential tweets identification* module

4.3 Evaluation Metrics

This section discusses the standard data mining metrics – *precision*, *recall*, and *f-score*, which are used to evaluate the proposed approach. Formally, these metrics in terms of True Positives (*TP*), False Positives (*FP*), and False Negatives (*FN*) are define in equations 8, 9, and 10, where *TP* is defined as the number of correctly classified as SDS tweets, *FP* is defined as number of NSDS tweets misclassified as SDS tweets, and *FN* is defined as number of SDS tweets misclassified as NSDS tweets.

$$Precision\ (\pi) = \frac{TP}{FP + TP} \tag{8}$$

$$Recall\ (\rho) = \frac{TP}{FN + TP} \tag{9}$$

$$F\text{-}score\ (F1) = \frac{2 \times \pi \times \rho}{\pi + \rho} \tag{10}$$

4.4 Evaluation Results

This section presents the experimental evaluation results over the three datasets discussed in subsection 4.2. All the experimental evaluations are performed using an LSTM model trained on 40 epoch. Table 7 presents the performance evaluation results of our proposed approach using the LSTM model on balanced and unbalanced datasets in terms of three evaluation metrics. On analysis, it can be observed from this table that in terms of all the three evaluation metrics, the proposed approach performs comparatively better on balanced datasets and shows slightly lower performance on unbalanced datasets. Another interesting observation from this table is that, in terms of all the three evaluation metrics, proposed approach performs best on Ptácek et al. (2014) dataset. Further, table 7 shows that the proposed approach performs comparatively better on the balanced version of our created dataset.

Datasets		#Sarcasm	#Non-sarcasm	Total (#tweets)
Ptácek et al. (2014)	Balanced	14500	14500	29000
	Unbalanced	5750	23000	28750
SemEval-2015	Balanced	500	500	1000
	Unbalanced	250	1100	1350
Twitter-280	Balanced	5000	5000	10000
	Unbalanced	500	2000	2500

Table 6: Statistics of the balanced and unbalanced datasets generated from table 5

Datasets		Evaluation results		
		π	ρ	F1
Ptácek et al. (2014)	Balanced	0.93	0.94	0.93
	Unbalanced	0.92	0.89	0.90
SemEval-2015	Balanced	0.86	0.84	0.85
	Unbalanced	0.93	0.75	0.83
Twitter-280	Balanced	0.90	0.92	0.93
	Unbalanced	0.89	0.86	0.88

Table 7: Performance evaluation of our proposed approach using LSTM on balanced and unbalanced datasets presented in table 6

4.5 Comparative Analysis

To be the best of authors knowledge there is no prior work on self-deprecating sarcasm detection using deep learning approach. However, a rule and machine learning-based approach was presented by the authors in Abulaish and Kamal (2018) and proposed approach is compared with that one. In Abulaish and Kamal (2018), authors considered Ptácek et al. (2014) dataset to detect self-deprecating sarcasm in tweets. We implemented Abulaish and Kamal (2018) to evaluated its efficacy over the three datasets. Figures 3 and 4 present the comparative performance evaluation of the proposed approach with Abulaish and Kamal (2018) in terms of *precision*, *recall*, and *f-score* over balanced and unbalanced version of all the three datasets, respectively.

It can be observed from figures 3 and 4 that the proposed LSTM-based deep learning approach outperforms Abulaish and Kamal (2018) in terms of *precision*, *recall*, and *f-score* on both balanced and unbalanced datasets. However, Abulaish and Kamal (2018) reported slightly better performance in terms of *precision* and *f-score* results on Ptácek et al. (2014) dataset.

5 Conclusion and Future Work

In this paper, we have proposed a new approach using LSTM-based deep learning for detecting self-deprecating sarcasm in textual data. The self-deprecating sarcasm is a special category of sarcasm in which users apply sarcasm on themselves. One of the major applications of this work is to promote self-deprecating marketing strategies. The proposed approach is evaluated over three Twitter datasets, including two benchmark datasets, and the experimental results are promising. It also performs significantly better than one of the state-of-the-art methods, which used rule-based and machine learning techniques for self-deprecating sarcasm detection. Exploring new patterns and consideration of multimedia contents for self-deprecating sarcasm detection seems one of the promising directions of future research.

Acknowledgments

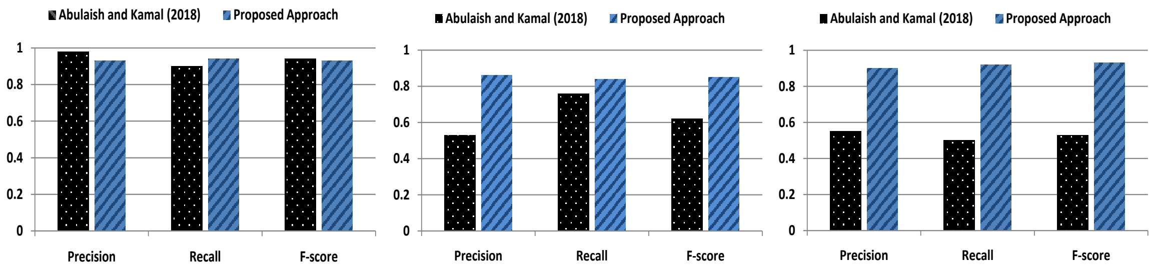
This publication is an outcome of the R&D work undertaken project under the Visvesvaraya PhD Scheme of Ministry of Electronics & Information Technology, Government of India, being implemented by Digital India Corporation.

References

Muhammad Abulaish and Ashraf Kamal. 2018. Self-deprecating sarcasm detection: An amalgamation of rule-based and machine learning approach. In *Proceedings of the International Conference on Web Intelligence (IEEE/WIC/ACM), Santiago, Chile*, pages 574–579. IEEE.

Silvio Amir, Byron C. Wallace, Hao Lyu, Paula Carvalho, and Mário J. Silva. 2016. Modelling context with user embeddings for sarcasm detection in social media. In *Proceedings of the 20th Special Interest Group on Natural Language Learning Conference on Computational Natural Language Learning((SIGNLL-CoNLL), Berlin, Germany*, pages 167–177. Association for Computational Linguistics.

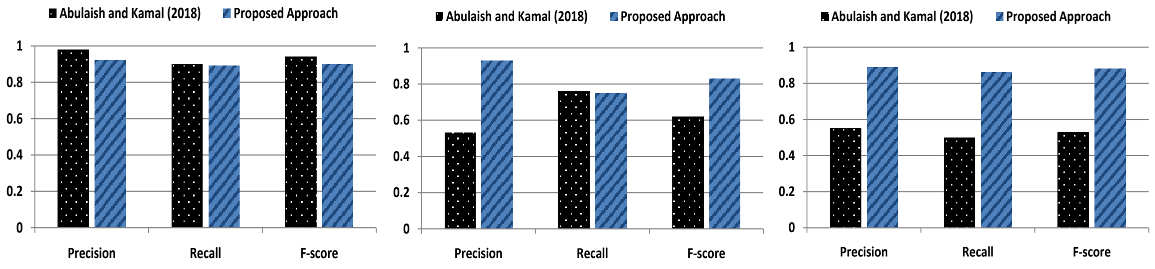
David Bamman and Noah A. Smith. 2015. Contextualized sarcasm detection on twitter. In *Proceedings of the 9th International Association for the Advancement of Artificial Intelligence Conference on Web and Social Media (ICWSM), Oxford, UK*, pages 574–577. Citeseer.



(a) (b) (c)

Figure 3: Visualization of the performance comparison results presented in table 7 over the balanced datasets (a) Ptácek et al. (2014) (b) SemEval-2015 (c) Twitter-280

.



(a) (b) (c)

Figure 4: Visualization of the performance comparison results presented in table 7 over the unbalanced datasets (a) Ptácek et al. (2014) (b) SemEval-2015 (c) Twitter-280

.

Santosh Kumar Bharti, Korra Sathya Babu, and Sanjay Kumar Jena Jena. 2015. Parsing-based sarcasm sentiment recognition in twitter data. In *Proceedings of the IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM), Paris, France*, pages 1373–1380. IEEE.

Mondher Bouazizi and Tomoaki Ohtsuki. 2015. Opinion mining in twitter how to make use of sarcasm to enhance sentiment analysis. In *Proceedings of the IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM), Paris, France*, pages 1594–1597. IEEE.

Mondher Bouazizi and Tomoaki Ohtsuki. 2016. A pattern-based approach for sarcasm detection on twitter. *IEEE Access*, 4:5477–5488.

Dmitry Davidov, Oren Tsur, and Ari Rappoport. 2010. Semi-supervised recognition of sarcastic sentences in twitter and amazon. In *Proceedings of the 14th Conference on Computational Natural Language Learning (CoNLL), Uppsala, Sweden*, pages 107–116. Association for Computational Linguistics.

Abhijeet Dubey, Aditya Joshi, and Pushpak Bhattacharyya. 2019a. Deep models for converting sarcastic utterances into their non sarcastic interpretation. In *Proceedings of the ACM India Joint International Conference on Data Science and Management of Data (CoDS-COMAD), Kolkata India*, pages 289–292. ACM.

Abhijeet Dubey, Lakshya Kumar, Arpan Somani, Aditya Joshi, and Pushpak Bhattacharyya. 2019b. “when numbers matter!”: Detecting sarcasm in numerical portions of text. In *Proceedings of the 10th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Medi Analysis (NAACL), Minneapolis, USA*, pages 72–80. Association for Computational Linguistics.

Aniruddha Ghosh, Guofu Li, Tony Veale, Paolo Rosso, Ekaterina Shutova, John Barnden, and Antonio Reyes. 2015. Semeval-2015 task 11: Sentiment analysis of figurative language in twitter. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval), Denver, Colorado*, pages 470–478. Association for Computational Linguistics.

Aniruddha Ghosh and Tony Veale. 2016. Fracking sarcasm using neural network. In *Proceedings of the 15th North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT), San Diego, California, USA*, pages 161–169. Association for Computational Linguistics.

Roberto González-Ibánez, Smaranda Muresan, and Nina Wacholder. 2011. Identifying sarcasm in twitter: A closer look. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL), Portland, Oregon*, pages 581–586. Association for Computational Linguistics.

Devamanyu Hazarika, Soujanya Poria, Sruthi Gorantla, Erik Cambria, Roger Zimmermann, and Rada Mihalcea. 2018. Cascade: Contextual sarcasm detection in online discussion forums. In *Proceedings of the 27th International Conference on Computational Linguistics (COLING), Santa Fe, New Mexico, USA*, pages 1837–1848. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Aditya Joshi, Vinita Sharma, and Pushpak Bhattacharyya. 2015. Harnessing context incongruity for sarcasm detection. In *Proceedings of the 53rd nnual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL-IJCNLP), Beijing, China*, page 757–762. Association for Computational Linguistics.

Aditya Joshi, Vaibhav Tripathi, Kevin Patel, Pushpak Bhattacharyya, and Mark Carman. 2016. Are word embedding-based features useful for sarcasm detection? In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), Austin, Texas*, pages 1006–1011. Association for Computational Linguistics.

Ashraf Kamal and Muhammad Abulaish. 2019. Self-deprecating humor detection: A machine learning approach. In *Proceedings of the 16th International Conference of the Pacific Association for Computational Linguistics (PACLING), Hanoi, Vietnam*, pages 1–13. Springer.

Christine Liebrecht, Florian Kunneman, and Antal V. D. Bosch. 2013. The perfect solution for detecting sarcasm in tweets #not. In *Proceedings of the 4th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis (WASSA), Atlanta, Georgia*, pages 29–37. ACL.

Abhijit Mishra, Diptesh Kanojia, and Pushpak Bhattacharyya. 2016. Predicting readers' sarcasm understandability by modeling gaze behavior. In *Proceedings of the 13th Association for the Advancement of Artificial Intelligence Conference on Artificial Intelligence, Phoenix, Arizona, USA*. AAAI.

Soujanya Poria, Erik Cambria, Devamanyu Hazarika, and Prateek Vij. 2016. A deeper look into sarcastic tweets using deep convolutional neural networks. In *Proceedings of the 25th International Conference on Computational Linguistics (COLING) Osaka, Japan*, pages 1601–1612.

Tomás Ptácek, Ivan Habernal, and Jun Hong. 2014. Sarcasm detection on czech and english twitter. In *Proceedings of the 25th International Conference on Computational Linguistics (COLING), Dublin, Ireland*, pages 213–223.

Ashwin Rajadesingan, Reza Zafarani, and Huan Liu. 2015. Sarcasm detection on twitter: A behavioral modeling approach. In *Proceedings of the 8th Association for Computing Machinery International Conference on Web Search and Data Mining (WSDM), Shanghai, China*, pages 97–106. ACM.

Ellen Riloff, Ashequl Qadir, Prafulla Surve, Lalindra D. Silva, Nathan Gilbert, and Ruihong Huang. 2013. Sarcasm as contrast between a positive sentiment and negative situation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), Seattle, Washington, USA*, pages 704–714. Association for Computational Linguistics.

Yi Tay, Anh Tuan Luu, Siu Cheung Hui, and Jian Su. 2018. Reasoning with sarcasm by reading inbetween. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL), Melbourne, Australia*, pages 1010–1020. Association for Computational Linguistics.

Oren Tsur, Dmitry Davidov, and Ari Rappoport. 2010. Icwsm-a great catchy name: Semi-supervised recognition of sarcastic sentences in online product reviews. In *Proceedings of the 4th International Association for the Advancement of Artificial Intelligence Conference on Weblogs and Social Media (ICWSM), Washington, DC, USA*, pages 162–169. AAAI.

Meishan Zhang, Yue Zhang, and Guohong Fu. 2016. Tweet sarcasm detection using deep neural network. In *Proceedings of the 26th International Conference on Computational Linguistics (COLING), Osaka, Japan*, pages 2449–2460.

Zhe Zhao, Paul Resnick, and Qiaozhu Mei. 2015. Enquiring minds: Early detection of rumors in social media from enquiry posts. In *Proceedings of the 24th International Conference on World Wide Web (WWW), Florence, Italy*, pages 1395–1405.