Final version of the accepted paper. Cite as: *"M. Abulaish and A. Kamal, Self-Deprecating Sarcasm Detection: An Amalgamation of Rule-Based and Machine Learning Approach, In Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence (WI), Santiago, Chile, pp. 1-6, Dec. 3-6, 2018."*

Self-Deprecating Sarcasm Detection: An Amalgamation of Rule-Based and Machine Learning Approach

Muhammad Abulaish, *SMIEEE*
Department of Computer Science
South Asian University, New Delhi, India
Email: abulaish@ieee.org

Ashraf Kamal
Department of Computer Science
Jamia Millia Islamia, New Delhi, India
Email: ashrafkamal.mca@gmail.com

***Abstract*—Sarcasm is a special category of figurative language, which is mainly used in online social media to convey messages with implicit semantics and criticism. Such messages are used for sarcastic remarks using contemptuous, ridicule, bitter, taunt, and mock related words or phrases. Though sarcasm detection is a well-considered problem by the researchers, to the best of our knowledge, none of them has considered the problem of *self-deprecating sarcasm*, which is a special category of sarcasm, mainly used by the users to deprecate or criticize themselves using sarcastic phrases. In this paper, we propose a novel self-deprecating sarcasm detection approach using an amalgamation of rule-based and machine learning techniques. The rule-based techniques aim to identify candidate self-around tweets, whereas machine learning techniques are used for feature extraction and classification. A total number of 11 features, including six self-deprecating features and five hyperbolic features are identified to train three different classifiers – *decision tree*, *naïve Bayes*, and *bagging*. The proposed approach is evaluated over a Twitter dataset containing 107536 tweets, and compared with some state-of-the-art methods for sarcasm detection.**

***Index Terms*—Social network analysis, Sarcasm detection, Self-deprecating sarcasm, Machine learning.**

I. Introduction

Nowadays, Twitter has become one of the most popular microblogging platforms with a huge active user-base. In 2006, Twitter was established as a microblogging venue to express ideas, opinions, emotions, views on day-to-day events occurring around the globe. By the second quarter of 2018, the monthly active users on Twitter are around 335 millions. Data generated by such a huge number of users are very useful for various applications, such as election results prediction, market trend analysis, product endorsement and recommendation, e-governance, sentiment analysis, and open-source intelligence.

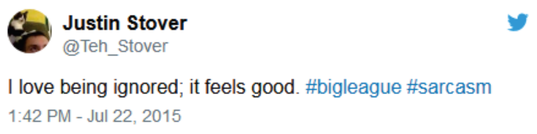
One of the most appealing attributes of Twitter is its character limitation to allow maximum 280 characters in a tweet, due to which tweets are generally precise and rich in semantic contents. However, tweets also contain high amount of informal data containing slangs, acronyms, non-literal words, emoticons, bashes, misspelled words, and unstructured phrases. Due to presence of informal contents, automated tweets analysis, especially sentiment analysis, has turned out to be a challenging task. The sentiment analysis problem becomes more challenging when opinion bearing words are used by the users to make sarcastic remarks using contemptuous, ridicule, amuse, bitter, taunt, and mock related words or phrases. Moreover, it is challenging to classifying tweets containing non-literal texts, which express different figurative language categories in which polarity of literal meaning may get affected significantly [1].

Sarcasm, which is one of the known categories of the figurative language, is very much common in human communication, especially in online social networks. In Collins dictionary, sarcasm[1] is defined as "*mocking, contemptuous, or ironic language intended to convey scorn or insult*". The implicit meaning hidden behind the sarcasm text is represented with an aim to insult or scorn a person or group through words or phrases in which bitter, contemptuous, taunt, ridicule remarks are involved[2].

Sarcasm can be classified into seven categories[3] – *self-deprecating*, *brooding*, *deadpan*, *polite*, *obnoxious*, *manic*, and *raging*. Self-deprecating sarcasm is a special category of sarcasm which is mainly used by the users to deprecate, criticise, disparage, deride or undervalue themselves using the sarcastic words or phrases. On analysis of a large number of tweets, it is found that sometimes users comment on themselves in their tweets. We have considered such tweets as *self-around* or *self-deprecating* tweets, and all those self-around tweets as *self-deprecating sarcasm* in which disparage, belittle, deprecate, undervalue, or criticism related words or phrases are used by the users to deprecate themselves in a humorous manner. Accordingly, *self-deprecating*[4] *sarcasm* is defined as a "*sarcasm that plays off of an exaggerated sense of worthlessness and inferiority*". Figure 1 presents an exemplar tweet with self-

[1]https://www.collinsdictionary.com/dictionary/english/sarcasm
[2]https://www.thefreedictionary.com/Sarkasm
[3]https://literarydevices.net/sarcasm/
[4]http://edtimes.in/seven-types-sarcasm/

deprecating sarcasm using the sarcastic phrase *love being ignored*.


Justin Stover
@Teh_Stover

I love being ignored; it feels good. #bigleague #sarcasm

1:42 PM - Jul 22, 2015

Fig. 1: A tweet expressing self-deprecating sarcasm

In this paper, we propose a self-deprecating sarcasm detection approach using rule-based and machine learning techniques. The proposed approach is implemented as a two-layer approach in which rule-based techniques are employed at the first layer to identify self-around tweets and to filter out all those tweets that are not self-around, increasing the efficiency of the self-deprecating sarcasm detection process. The second layer considers self-around tweets identified by the first layer as an input and applies feature extraction and classification techniques for self-deprecating sarcasm detection. A total number of 11 features including six *self-deprecating* features and five *hyperbolic* features are identified and three different classifiers – decision tree, naïve Bayes, and bagging are used for self-deprecating sarcasm detection.

II. Related Works

Researchers have proposed various supervised learning techniques such as Support Vector Machine (SVM), Maximum Entropy (ME), Decision Tree (DT), and Logistic Regression (LR) for sarcasm detection in Twitter [2], [3], [4], [5], [6]. In addition, there are few articles in which semi-supervised [7], [8], [9], rule-based [10], [11], pattern-based [5], and deep learning techniques [12] have been proposed for sarcasm detection. Though most of the authors [7], [8], [2], [4], [5], [11], [12] have mainly considered English language texts, some of them have also considered other languages for sarcasm detection [3]. Similarly, most of the existing literatures have considered Twitter datasets for sarcasm detection, and a very few have considered other data sources, such as Amazon merchant site [7], [8]. The features considered for sarcasm detection are mainly seen as lexical, pragmatics, phrases, punctuations, sentiments, syntactic, patterns, interjections, intensifiers, semantic, and context incongruity.

Tsur et al. [8] proposed a semi-supervised algorithm for sarcasm detection using syntactic, patterns- and punctuations-based features in Amazon's product reviews dataset. González-Ibáñez et al. [2] applied SVM approach along with sequential minimal optimization and LR techniques to identify sarcasm. Riloff et al. [10] proposed a rule-based approach for sarcasm identification in Twitter, where positive sentiment contradicts the negative situation. Bouazizi and Ohtsuki [5] proposed a pattern-based approach for sarcasm detection in Twitter. Lukin and Walker [9] applied a semi-supervised technique using boot-strapping approach. Ptácek et al. [3] applied SVM and ME-based machine learning techniques for sarcasm detection in English and Czech languages. Gupta and Yang [6] proposed an affect-cognition-sociolinguistics features for sarcasm detection using SVM. Rajadesingan et al. [4] proposed an approach for sarcasm classification using behavioral modeling and applied SVM, DT, LR techniques. Ghosh et al. [1] applied SVM and regression techniques to detect sarcasm. Bharti et al. [11] proposed two pattern-based algorithms to detect sarcasm. Amir et al. [12] proposed a deep learning based convolutional neural network model to learn user embedding along with utterance based embedding for sarcasm detection.

It can be seen from the above discussions that a number of researchers have proposed different techniques for sarcasm detection in Twitter and other data sources. However, none of them have considered the detection of self-deprecating sarcasm, which is a special case of sarcasm mainly used by the users to deprecate, criticize, disparage, deride or undervalue themselves using the sarcastic words or phrases.

III. Proposed Approach

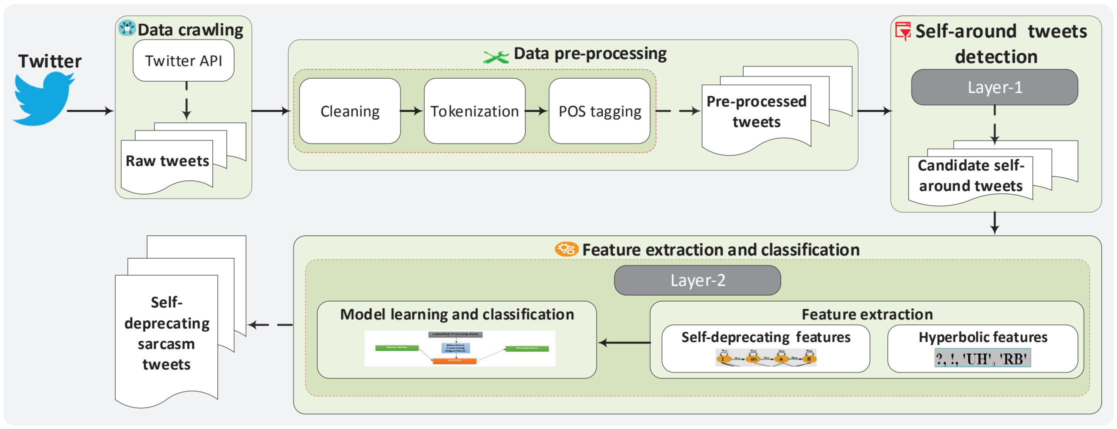
Figure 2 presents the work-flow of the proposed self-deprecating sarcasm detection approach. A brief description of the various modules is presented in the following sub-sections.

A. Data Crawling and Data Pre-processing

We have implemented a data crawler in Python using `REST API` to curate Twitter data for the English language tweets ids provided by Ptácek et al. [3]. The data pre-processing module applies a series of processing tasks, such as cleaning, tokenization, and Parts-Of-Speech (POS) tagging over the curated tweets to generate ready-to-analyze data for self-deprecating sarcasm detection. Data cleaning aims to remove URL's, @mention, retweets, hashtags, ampersands, extra white spaces, and to convert upper-case letters into lower-cases. It also removes hexa-characters, double quotes, emoticons, numbers, and dots. Thereafter, tweets are tokenized and POS tags are generated through `spacy` tagger.

B. Self-Around Tweets Detection (Layer-1)

The self-around tweets detection module is mainly used as a filter, and its main task is to identify self-around tweets that are candidates for self-deprecating sarcasm. This module immensely improves the efficiency of the self-deprecating sarcasm detection process, as all those tweets that are not self-around are filtered out in this layer. This module is implemented as a rule-based system, as given in algorithm 1. Instead of string matching, it uses regular expressions to implement the filtering rules. Steps 3-7 of the algorithm 1 identify self-around tweets based on POS tags. If POS tag of the first word of a tweet is either `UH` (interjection), or `NN` (noun), or `RB` (adverb), or `VB`



Twitter
Data crawling
Twitter API
Raw tweets
Data pre-processing
Cleaning
Tokenization
POS tagging
Pre-processed tweets
Self-around tweets detection
Layer-1
Candidate self-around tweets
Feature extraction and classification
Layer-2
Model learning and classification
Feature extraction
Self-deprecating features
Hyperbolic features
?, !, 'UH', 'RB'
Self-deprecating sarcasm tweets

Fig. 2: Work-flow of the proposed self-deprecating sarcasm detection approach

(verb), or `VBP` (verb, non third person singular present), or `JJ` (adjective), or `MD` (model verb), then the tweet is considered as self-around and it is stored for further processing. Similarly, steps 8-14 of the algorithm 1 traverse the whole tweet and check whether it contains either of the tokens 'i', 'me' or 'my'. If a tweet contains any of these tokens, then it is considered as self-around and stored for further processing.

Algorithm 1: Self-Around Tweets Detection

```
Input  : A file $D$ containing pre-processed tweets
Output: A file $D_{sa}$ containing only self-around tweets
foreach $tweet$ in $D$ do
    $t$ ←spacy(tweet);      /* tokenize & assign POS tags */
    $tag$ ←t[1].tag ;        /* POS tag of the first word */
    if $tag \in \{UH, NN, RB, VB, VBP, JJ, MD\}$ then
        append($D_{sa}$, $t$);       /* $t$ is self-around tweet */
        goto step 1;
    end
    $n$ ← no-of-token(t)
    for $i$ ← 1 to $n$ do
        if $t[i].token \in \{i, me, my\}$ then
            append($D_{sa}$, $t$) ; /* $t$ is self-around tweet */
            goto step 1;
        end
    end
end
```

C. Feature Extraction and Classification (Layer-2)

The feature extraction and classification module aims to identify various features from self-around tweets for learning classification models. A detailed discussion about these functionalities is presented in the following sub-sections.

1) Feature Extraction: Feature extraction is one of the core steps for the development of any classification system. We have identified a total number of 11 features and classified them into two categories – *self-deprecating* features and *hyperbolic* features. For feature identification, we have considered the sequential order of the tags and tokens in a tweet as an important criteria.

Self-Deprecating Features:
The features in this category aim to capture deprecating factors in self-around tweets, based on the patterns of the relative orders of tokens and tags in a tweet.

$\mathcal{F}_1$ *(interjection followed by 'i'):* If an interjection (`UH`-tagged word) is followed by the token 'i', then the value of this feature is considered as 1, otherwise 0. For example, consider the tweet "*Wow I do enjoy insomnia!!!!*", which is a self-deprecating sarcasm following this pattern.

$\mathcal{F}_2$ *('i' followed by verb and question word):* If a token 'i' is followed by the main verb (`VBP`) and then by a question word (`WRB`), then the value of this feature is considered as 1, otherwise 0. For example, consider the tweet "*I love when apartment complex office treats me like I'm the idiot when I tell them their crappy anything is broken.*", which is a self-deprecating sarcasm following this pattern.

$\mathcal{F}_3$ *(Common deprecating pattern):* This feature aims to capture different self-deprecating patterns in a tweet. The value of this feature is considered as 1 if either of the following conditions ($C1$ to $C5$) is satisfied, otherwise 0.

- *C1*: If the token '*it*' is followed by a question word (`WRP`). For example, consider the tweet "*I love it when you try to help me with my flaws. No really, I had no idea they were there.*", which is a self-deprecating sarcasm following this pattern.

- *C2*: If a question word (`WRP`) is followed by a pronoun (`PRP`). For example, consider the tweet "*Just looooove when my hair appointment has to be cancelled for a last minute #Brussels trip.*", which is a self-deprecating sarcasm following this pattern.

- *C3*: If the token 'i' is followed by the word 'love'. On analysis, it is found that the word 'love' is most frequently used in self-deprecating tweets. For example, consider the tweet "*I love when you call me at two in the morning*", which is a self-deprecating sarcasm following this pattern.

- *C4*: If an adjective (`JJ`) is followed by an adverb (`RB`). For example, consider the tweet "*Awesome really love everything about polynomial and rational functions a lot.*", which is a self-deprecating sarcasm following this pattern.
- *C5*: This is the reverse scenario of *C4*, and it represents a condition in which an adverb (`RB`) is followed by an adjective (`JJ`). For example, consider the tweet "*I am gonna be stuck in my house all day again tomorrow really excited.*", which is a self-deprecating sarcasm following this pattern.

$\mathcal{F}_4$ *('i' followed by verb and adverb or adjective):* This is also a strong indicator of self-deprecating sarcasm, as intensifiers (adverbs/adjectives) play an important role in exaggeration or overstatement [11]. If a self-around tweet contains the token '*i*', which is followed by the main verb (`VBP`) and then by *adverb* (`RB`) or *adjective* (`JJ`), then the value of this feature is considered as 1, otherwise 0. For example, consider the tweets "*I love really being ignored and completely iced out.*" and "*I love happy to sat through a meeting about nothing. But there were cupcakes afterwards so it was worth it.*" that are self-deprecating sarcasms following these patterns.

$\mathcal{F}_5$ *('am' followed by adjective or adverb):* If a self-around tweet contains the auxiliary verb '*am*', which is followed by either *adjective* (`JJ`) or *adverb* (`RB`), then the value of this feature is considered as 1, otherwise 0. For example, consider the tweets "*Working till 8 in Christmas Eve. I am single here so excited*" and "*I am really glad my iPhone shuts itself down it really makes happy*", that are self-deprecating sarcasms following these patterns.

$\mathcal{F}_6$ *('i' followed by negative modal verb):* If a self-around tweet contains the token '*i*', which is followed by a modal verb (`MD`) and then followed by the token '*not*' (e.g., I may not, I can not, etc.), then the value of this feature is considered as 1, otherwise 0. For example, consider the tweet "*I can not wait to go home and study some more!*", which is a self-deprecating sarcasm following this pattern.

Hyperbolic Features:
The presence of hyperbole in a tweet plays an important role in sarcasm detection. The hyperbolic tweets generally employ exaggeration, over emphasis, or over-statement. In order to detect self-deprecating sarcasm, the frequency count of interjections, exclamation marks, and question marks are considered as hyperbolic features. In addition, two intensifier-related hyperbolic features such as frequency count of adverbs and adjectives are also considered, because presence of these intensifiers adds an over-exaggeration component in a tweet.

$\mathcal{F}_7$ *(interjection count):* If the frequency count of interjections (`UH`) in a tweet is at least two, then the value of this feature is considered as 1, otherwise 0. For example, consider the tweet "*unfollower you and u came know this via unfollower! oh wow! my day is just not complete without tweets of yours.*", which expresses self-deprecating sarcasm and contains two interjections – *oh* and *wow*.

$\mathcal{F}_8$ *(exclamation mark count):* If the frequency count of exclamation mark in a tweet is at least two, then the value of this feature is considered as 1, otherwise 0. For example, consider the tweet "*I feel special i live in the coldest place in the world today !!!*", which expresses self-deprecating sarcasm and contains three exclamation marks.

$\mathcal{F}_9$ *(question mark count):* If the frequency count of question mark in a tweet is at least two, then the value of this feature is considered as 1, otherwise 0. For example, consider the tweet "*I need a break from my own thoughts really ???*", which expresses self-deprecating sarcasm and contains three question marks.

$\mathcal{F}_{10}$ *(adverb count):* If the frequency count of adverb in a tweet is at least two, then the value of this feature is considered as 1, otherwise 0. For example, consider the tweet "*textbook shopping online yay no really its totally fun for me not.*", which expresses self-deprecating sarcasm and contains two adverbs – *really* and *totally*.

$\mathcal{F}_{11}$ *(adjective count):* If the frequency count of adjective in a tweet is at least two, then the value of this feature is considered as 1, otherwise 0. For example, consider the tweet "*wow today has been so great I just want to add it to my already perfect week*", which expresses self-deprecating sarcasm and contains two adjectives – *great* and *perfect*.

2) Model Learning and Classification: We have considered three classifiers – *decision tree*, *naïve Bayes*, and *bagging* that are implemented as a component of `WEKA` tool kit 3.8. We have used `C4.5 (J48)` implementation of *decision tree* with its default settings of `WEKA`. *Naïve Bayes* is based on Bayes' theorem, and it supports predictive modeling and assumptions of strong independence across the features. Like decision tree, the default settings of `WEKA` are also used for this classifier. *Bagging* is a classifier ensemble technique, which provides better predictive results using multiple learning algorithms. It reduces variance and avoids overfitting. Like *decision tree* and *naïve Bayes*, we have

TABLE I: Crawled datasets statistics

Tweets category	Balanced #ids (#tweets)	Unbalanced #ids (#tweets)	Total #ids (#tweets)
Sarcasm	50000 (35451)	25000 (17637)	75000 (53088)
Non-sarcasm	50000 (39216)	75000 (58979)	125000 (98195)
Total #ids (#tweets)	100000 (74667)	100000 (76616)	200000 (151283)

TABLE II: Balanced and unbalanced datasets generated from Table I

Tweets category	Dataset-1 (#tweets)	Dataset-2 (#tweets)	Total (#tweets)
Sarcasm	35000	17500	52500
Non-sarcasm	35000	57500	92500
Total (#tweets)	70000	75000	145000

also used default `WEKA` settings for the bagging method.

IV. Experimental Setup and Results

This section presents the experimental evaluation of the proposed approach for self-deprecating sarcasm detection.

A. Dataset

We have considered the tweets ids provided by Ptácek et al. [3] and curated the respective tweets using a crawler developed in `Python 2.7`. The authors provided balanced and unbalanced distribution of tweets ids. In balanced distribution, 50000 tweets ids are available for each sarcasm and non-sarcasm categories, whereas the unbalanced distribution contains 25000 sarcasm tweets ids and 75000 non-sarcasm tweets ids. Out of total 200000 tweets ids (balanced and unbalanced) provided by Ptácek et al. [3], only 151283 tweets were successfully crawled, as few tweets were protected or deleted and accordingly unable to be downloaded using the API. Table I presents a summary of the crawled dataset from Twitter. From this dataset, we have created two datasets namely *dataset-1* (balanced) and *dataset-2* (unbalanced) consisting of 70000 and 75000 tweets, respectively, as given in Table II. Rest of the experiments are based on these datasets.

B. Evaluation Metrics

In this section, we present a formal description of the performance evaluation metrics – *Precision*, *Recall*, and *F-score* that are defined using the concept of *True Positives* (number of self-deprecating sarcasm tweets identified correctly), *False Positives* (number of normal tweets identified as self-deprecating sarcasm), and *False Negatives* (number of self-deprecating sarcasm tweets identified as normal). *Precision* and *Recall* measure the correctness and completeness, respectively of a classifier, whereas *F-score* is the harmonic mean of the *Precision* and *Recall* values. The value of *F-score* is higher only when the values of both *Precision* and *Recall* are high. Equations 1, 2, and 3 define these metrics formally.

$$Precision = \frac{True\ Positives}{True\ Positives + False\ Positives} \tag{1}$$

TABLE III: Retained instnaces after filtering through algorithm 1

Tweets category	Dataset-1 (#tweets)	Dataset-2 (#tweets)	Total (#tweets)
Sarcasm	26428	13219	39647
Non-sarcasm	25498	42391	67889
Total (#tweets)	51926	55610	107536

$$Recall = \frac{True\ Positives}{True\ Positives + False\ Negatives} \tag{2}$$

$$F\text{-}score = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{3}$$

C. Evaluation Results

In this section, we present the evaluation results of the proposed self-deprecating sarcasm detection approach. We have applied algorithm 1 on the tweets of each datasets to identify self-around tweets for further processing (i.e., feature extraction and classification). Table III presents a summary of the retained tweets after filtering out the tweets using algorithm 1 that are not self-around. Thereafter, we convert each self-around tweet into an 11-dimensional numeric feature vector using a software module implemented in `Python 2.7`, where each dimension corresponds to a particular feature defined in the previous section. Thereafter, we have trained three classification algorithms *decision tree*, *naïve Bayes*, and *bagging* over each dataset separately, and calculated the values of the performance evaluation metrics using 10-fold cross validation.

Table IV presents the performance evaluation of the proposed approach in terms of *Precision*, *Recall*, and *F-score* using all three classification algorithms. It can be observed from this table that *decision tree* achieved highest *Recall* and *F-score* values, whereas *bagging* achieved highest *Precision* values for both datasets. On the other hand, *naïve Bayes* achieved low *Precision* and *F-score* but high *Recall* for both datasets.

As stated earlier, to the best of our knowledge, there is no literature on self-deprecating sarcasm detection. Therefore, it is no possible to provide a comparative evaluation of our proposed approach. However, since self-deprecating sarcasm is a special category of sarcasm, we have considered three state-of-the-art techniques on sarcasm detection that have used rule- and pattern-based approaches. For comparative analysis, we have considered the highest *Precision*, *Recall*, and *F-score* values reported by the authors in their respective literatures. We have also considered out highest *Precision*, *Recall*, and *F-score* values across all classifiers and datasets. Figure 3 presents a visualization of the comparative analysis of our proposed approach with three state-of-the-art techniques. It can be observed from this figure that our proposed approach performs significantly better than the existing approaches.

TABLE IV: Performance evaluation results using 10-fold cross validation

Dataset	Decision tree			Naïve Bayes			Bagging		
	Precision	Recall	F-score	Precision	Recall	F-score	Precision	Recall	F-score
Dataset-1	0.984	0.914	0.948	0.543	0.913	0.681	0.992	0.891	0.939
Dataset-2	0.969	0.923	0.945	0.550	0.905	0.684	0.980	0.885	0.930

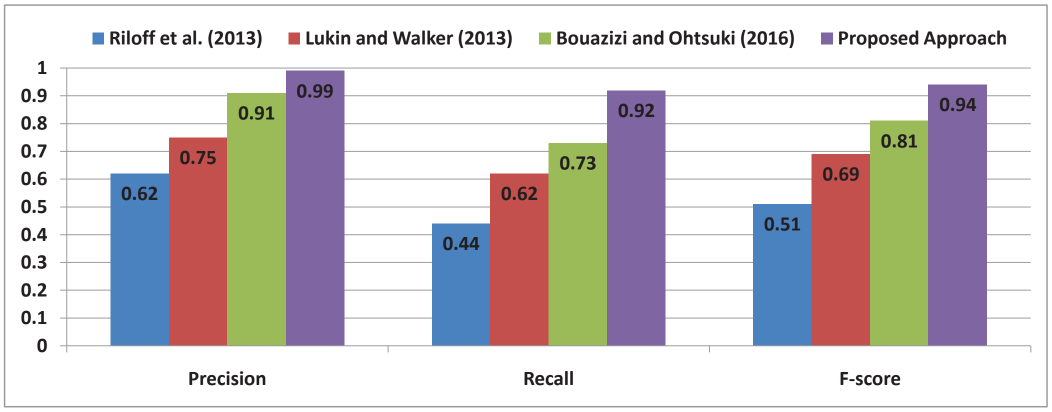


Fig. 3: Performance comparison results over the datasets given in Table III

V. Conclusion and Future Research

In this paper, we have introduced the problem of self-deprecating sarcasm detection, which is a special case of sarcasm detection. To this end, we have proposed an amalgamation of rule-based and machine learning approaches for detecting self-deprecating sarcasm. Though a number of researchers have targeted different categories of figurative language, such as sarcasm, irony, satire, etc. in Twitter, none of them has considered the problem of self-deprecating sarcasm detection. Like self-deprecating sarcasm, detection of other categories of sarcasm, such as *brooding*, *polite*, *raging*, etc. in Twitter seems one of the future directions of research.

Acknowledgment

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

[1] A. Ghosh, G. Li, T. Veale, P. Rosso, E. Shutova, J. Barnden, and A. Reyes, "Semeval-2015 task 11: Sentiment analysis of figurative language in twitter," in *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval), Denver, Colorado*, pp. 470–478, ACL, June 4-5, 2015.

[2] R. G.-I. nez, S. Muresan, and N. Wacholder, "Identifying sarcasm in twitter: A closer look," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL), Portland, Oregon*, pp. 581–586, ACL, June 19-24,2011.

[3] T. Ptácek, I. Habernal, and J. Hong, "Sarcasm detection on czech and english twitter," in *Proceedings of the 25th International Conference on Computational Linguistics (COLING), Dublin, Ireland*, pp. 213–223, ACL, August 23-29, 2014.

[4] A. Rajadesingan, R. Zafarani, and H. Liu, "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*, pp. 97–106, ACM, February 2-6, 2015.

[5] M. Bouazizi and T. Ohtsuki, "A pattern-based approach for sarcasm detection on twitter," *IEEE Access*, vol. 4, pp. 5477–5488, September 2016.

[6] R. K. Gupta and Y. Yang, "Crystalnest at semeval-2017 task 4: Using sarcasm detection for enhancing sentiment classification and quantification," in *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval), Vancouver, Canada*, pp. 626–633, ACL, August 3–4, 2017.

[7] D. Davidov, O. Tsur, and A. Rappoport, "Semi-supervised recognition of sarcastic sentences in twitter and amazon," in *Proceedings of the 14th Conference on Computational Natural Language Learning (CONLL), Uppsala, Sweden*, pp. 107–116, ACL, July 15-16, 2010.

[8] O. Tsur, D. Davidov, and A. Rappoport, "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*, pp. 162–169, AAAI, May 23-26, 2010.

[9] S. Lukin and M. Walker, "Really? well. apparently bootstrapping improves the performance of sarcasm and nastiness classifiers for online dialogue," in *Proceedings of the Workshop on Language in Social Media (LASM), Atlanta, Georgia*, pp. 30–40, ACL, June 13, 2013.

[10] E. Riloff, A. Qadir, P. Surve, L. D. Silva, N. Gilbert, and R. Huang, "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*, pp. 704–714, ACL, October 18-21, 2013.

[11] S. K. Bharti, K. S. Babu, and S. K. J. Jena, "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*, pp. 1373–1380, IEEE, August 25–28, 2015.

[12] S. Amir, B. C. Wallace, H. Lyu, P. Carvalho, and M. J. Silva, "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 (CoNLL), Berlin, Germany*, pp. 167–177, 2016.