



Irony Detection in Bengali Tweets: A New Dataset, Experimentation and Results

Adhiraj Ghosh¹ and Kamal Sarkar²(✉)

¹ Manipal Institute of Technology, Manipal, India

adhiraj.ghosh@learner.manipal.edu

² Jadavpur University, Kolkata, India

jukamal2001@yahoo.com

Abstract. Irony detection is a difficult task because the intended meaning of a sentence differs from the literal meaning or sentiment of that sentence. Most existing work on this subject has focused on irony detection in the English language. Since no public dataset is available for this task in the Bengali domain, we have created a Bengali irony detection dataset that contains a total of 1500 labeled Bengali tweets. This paper presents the description of the Bengali irony detection dataset developed by us and reports some results obtained on our Bengali irony dataset using several widely used machine learning algorithms such as Naïve Bayes, Support Vector Machine, K-Nearest Neighbor and Random Forest.

Keywords: Irony detection · Machine learning · Bengali · Tweets · Naïve Bayes · Support Vector Machine · K-Nearest Neighbor · Random Forest

1 Introduction

There has been a tremendous surge of data on the internet after the social media boom. Since the advent of the Social Web in 2006, the amount of textual content on the internet has exponentially grown and provides a great deal of potential for analysis. Since millions of tweets are generated every day, machine learning provides the necessary platform for adequate data analytics to better understand the activities of the average social media user as well as the users' feedback on various social and political issues. So, to understand the users' activities and comments on social media, understanding of varied features of language is needed. When the automatic system is used to manipulate and process a large amount of social media texts, the system should have the ability to understand the intricate features of language. Ironic texts are more difficult to understand than the general non-ironic texts.

Irony is popularly defined as a literary device wherein the literal meaning of a sentence differs from the figurative meaning the sentence is trying to portray [1]. This paper focuses on comprehending the most common forms of irony by enlisting certain characteristic features of each type of irony to develop a system to classify tweets

correctly. Automatic irony detection provides a more fine-grained understanding of sarcasm detection [2] and sentiment analysis [3] and the multiclass analysis streamlines the Bengali user's propensity for the usage of the language. Irony detection and analysis is important for the development of sociolinguistics and analysis the uses of language constructs by specific communities. Automatic irony detection also has many other applications, for example, online harassment detection, ironic speech understanding and more importantly, sentiment analysis. It was observed that sentiment analysis system shows relatively poor performance on ironic texts compared to non-ironic texts. Hence, an added analysis of irony in the tweets propels such tasks towards more effectiveness.

The most common approaches to Irony detection in English Twitter use machine learning algorithms [4, 5]. But a huge stride in automatic irony detection in English language was made by Hee et al. [6–8] as a part of a shared task, SemEval-2018 Task 3 on Irony detection in English tweets [9]. Some of the most notable results of the irony detection tasks for English tweets [10–14] have been obtained using various techniques that are based on machine learning algorithms.

The Bengali language is the 7th most spoken language in the world, with approximately 215 million speakers all over the world. Bengali is a primary language in Bangladesh, and in India - West Bengal, Tripura and Assam. In the past few years, South Asian nations like India and Bangladesh have witnessed a data revolution, which was prompted by greatly diminished costs of mobile data in both countries, resulting in a massive boom in the number of active internet users in the countries. The latest reports on the number of active internet users in India 604.21 million¹ and that of Bangladesh being 91.421 million². These staggering numbers provide a parameter to apprehend the activity of most of the Bengali speaking population in the countries.

Twitter has been the platform for posting opinions and comments on various social and political issues. With the evolution of a script-based keyboard and the advent of the smartphone, Twitter sees plenty of users' opinions written in several languages and scripts the people from various corners of the world are using this platform and posting opinions and comments in their own languages.

The focus of this work is on developing a corpus for Bengali irony detection and developing baseline systems for irony detection in Bengali. Section 2 describes the Bengali irony dataset created by us. In Sect. 3, we have provided a detailed description of the various types of irony we have considered for our experiments and the distinction factors among the four types of irony considered in implementing our classification model. Section 4 is where we have discussed the feature extraction process. The four widely used baseline classification algorithms have been used to implement our models- Naïve Bayes, Support Vector Machine, K-Nearest Neighbors and Random Forest which have been described in Sect. 5. Section 6 provides the results of our models on our dataset and finally, in conclusion section, we describe future scope and the importance of study in the field of Indian languages.

¹ https://main.trai.gov.in/sites/default/files/PIR_04042019_0.pdf.

² <http://www.btrc.gov.bd/content/internet-subscribers-bangladesh-january-2019>.




2 Corpus Development

It is to be noted that there is a difference between sentiment, satire and irony when it comes to a linguistic definition and scope. While there is research in sentiment analysis in Bengali in [3], research is lacking in a fine-grained model for irony in Bengali. In that regard, a major obstacle faced in the development of irony analysis system for Bengali tweets was the lack of any publicly available dataset for Bengali Irony Detection. To that end, a total of 1500 tweets were collected from Twitter and manually annotated by us for corpus development. Due to the low number of tweets available in the Bengali language as compared to English, the tweets have been collected within the time period spanning from 17/10/2010 to 4/1/2019. The tweets that have been collected and annotated contain the bare textual content and does not contain traces of metadata, which include twitter handle, display name, timestamps, user ids, locations, etc. All mentions, tags, URLs and punctuation were removed too as they are not significant for irony classification. Finally, all English letters were converted to lowercase to prevent duplication in the vocabulary list.

The entire corpus was created in a text file with UTF-8 (Unicode Transformation Format) encoding. In this way, the.txt file could support all the Bengali script characters. In our case, the comma separates the tweet from its type of irony label.

For irony detection, there are some stark differences from standard sentiment analysis. Firstly, for irony classification, we did not consider stop word removal as this may tamper the overall meaning of the tweets. Next, every emoji was replaced by a universal code as determined by the Unicode Common Locale Data Repository (CLDR), which supports all Unicode characters. Every emoji also has a CLDR short name³, which is a suitable substitute for the emoji itself for Natural Language Processing applications. Table 1 shows some examples of the representation of emojis in Unicode.

Table 1. Code and the CLDR short names for common emojis.

Emoji	Code	CLDR short name
	U + 1F600	grinning_face
	U + 1F602	face_with_tears_of_joy
	U + 1F47B	ghost

3 Irony Types

We have annotated the tweets at the two level: (1) coarse level and (2) fine-grained level. At coarse level, a tweet is annotated as ironic and non-ironic whereas at the fine-grained

³ <https://unicode.org/emoji/charts/full-emoji-list.html>.

level, ironic tweets are further classified into four types of irony. We have implemented document classification as a method of manual annotation, where based on a given set of rules that are used to define a class or label and a defined set of parameters to cause a distinction among the four classes, we have managed to label each input tweet based on the given task (coarse-grained annotation or fine-grained annotation).

Four types of irony were considered for our multi-class irony classification- (1) irony-clash, (2) verbal irony, (3) situational irony and (4) non-irony. The distribution of each type of label in our developed irony corpus is shown in Table 2.

Table 2. Distribution of irony types in the corpus.

Irony type	Number of tweets
Verbal_Irony	281
Irony_Clash	258
Situational_Irony	254
No_Irony	707

Our developed Bengali irony dataset contains 707 non-ironic tweets and 793 ironic tweets. Since our research primarily focuses with fine-grained irony analysis, the dataset was limited to 1500 tweets due to the lack of adequate number of ironic tweets present at the aforementioned timeframe and to prevent classification bias for the non-ironic tweets.

Since irony analysis is extremely subjective, the annotation process has been made as objective as possible. Based on objective criteria, at first, a fundamental distinction was made between an ironic statement and an ironic narrative. Then, the ironic statement is divided into three types-Verbal_Irony and Irony_Clash and Situational_Irony. The detailed descriptions of the different types of irony with sample examples are given in the subsequent subsections.

3.1 Verbal Irony

Verbal irony is widely used and well-defined form of irony [15]. According to the core premise of irony, verbal irony describes a statement that has an implicit sentiment of irony. Burgers [16] describes the four aspects of verbal irony: implicit, evaluative, differentiable from a non-ironic statement, and the comprehension of an opposing sentiment. Keeping these aspects in mind, a statement that contains verbal irony should:

- have a detectable sentiment in the literal composition of the tweet
- have the opposite sentiment in the intended meaning of the tweet
- have only one sentiment in the words used in the tweet and the opposite in its intention. For example, the words of a tweet should have a positive sentiment literally and a negative sentiment intentionally and vice versa
- not have neutral sentiment attached to either the literal composition or the intended meaning of the tweet

- have emojis of the same sentiment as that of the overall literal sentiment of the tweet, if present in the tweet.

A few examples of verbal irony are given below:

1. “আপনার হৃদয় শিরিষ-কাগজ এর মতন মসৃণ” (Your heart is as smooth as sandpaper). Here, the overall sentiment of the tweet is positive with the use of the word শিরিষ- (smooth). But the intended meaning uses the কাগজ (sandpaper) and infers a contradiction, since sandpaper is rough, thereby providing an opposite, negative sentiment to the meaning behind the tweet. The tweet does not have a neutral sentiment. Therefore, since the tweet fulfils 4 out of 5 criteria for verbal irony, we annotate the above tweet as Verbal_Irony.
2. খেলা দেখে অনেক মজা পাইতাছি। খেলা তো নয় মনে হয় কমেডি হইতাছে|||: face_with_tears_of_joy: #LOL #BANvAFG” (Watching the game, I am having a lot of fun. The match feels like a comedy show, not a cricket game: face with tears of joy: #LOL #BANvAFG). This tweet has positive sentiment, including emojis. The tweet has only one sentiment and an opposing intention and hence fulfils almost all the criteria for verbal irony.

3.2 Irony by Clash

This category describes an expressive statement where there exists a literal and opposite intended meaning of the tweet and where the positive polarity and the negative polarity that defines the inversion between the literal and intended meanings are present in the tweet itself, as opposed to verbal irony, where the intended meaning has to be inferred by the reader. Therefore, tweets with the Irony_Clash label have some distinctive features. These kinds of tweets should:

- have both sentiments of positive and negative polarities in the literal meaning of the tweet, providing a literal distinction from verbal irony
- have either of the two sentiments in the intended meaning of the tweet
- not have an overall neutral sentiment just because words of opposite polarities are clashing
- may or may not depend on emojis to provide the opposing sentiment.

A few examples of irony-clash are given below.

1. “৬০,০০০ টাকার মোবাইল ফোন Bluetooth নাই!! তাহলে সেই আপেল হাতে না রেখে খাওয়াটাই ভালো।” (A 60,000 rupees phone doesn’t even have Bluetooth in it!! Then, it’s better if I ate that apple rather than keep it in my hand.) In this tweet, the sentiment clash occurs between ‘doesn’t’ and ‘better’. The tweet is ironic due to the similarities made between a costly phone and an everyday fruit and the intended meaning of the tweet has a negative sentiment as well, without having a neutral overall sentiment. Hence, it is classified as Irony_Clash.

3.3 Situational Irony

While verbal irony and irony by clash described irony existent in statements, situational irony is evident in the narratives of situations in text. The irony is evident when the outcome of the narrative defies the standard expectation of the same. In literary texts, situational irony arises when the readers were aware of the outcome of the actions of the characters when the characters themselves did not. According to Shelley [17], situational irony is derived from the schema-recognition system of human cognizance where we script a narrative for a situation, failing to comply with which is deemed ironic. The situational ironic tweets have the following characteristics. The tweets should:

- be in the form of a narrative of situation, with an identifiable flow of events
- exist in a particular time frame, from start to finish
- refer to a subject of the narrative, living or non-living, whose actions are deemed as ironic
- must operate on a distinction between what is being claimed in the text and what is being inferred by the reader
- invoke a sense of higher understanding that is not implicit to the tweet, which is the burden of the reader to comprehend
- may or may not depend on emojis to add to the narrative.

An example of Situational irony is given below.

1. “একটি অ্যাম্বুলেন্স রাস্তায় একটি লোককে চাপা দিল ।” (The ambulance ran over a man on the street). This tweet depicts the narrative of an ambulance on the street, which had run over a man, where we see the beginning and end of the narrative. The irony is denoted by the contrast between ambulance, the vehicles which are associated with hospitals and, by extension, life and “লোক কে চাপা দিল” (ran over the man) which signifies death. Hence, the tweet can be classified as Situational_Irony.

3.4 No Irony

The easiest to understand, a tweet is non-ironic if the statement or narrative does not have an inversion between its literal meaning and its intended meaning. Such tweets require no added analysis and are much easier to create and annotate. Most of the tweets on Twitter are non-ironic, in all languages. For example:

1. “মানুষের জীবনে শৈশব হল সবচেয়ে গুরুত্বপূর্ণ ।” (Childhood is the most important part of human life.) This tweet has no implicit meaning, is not a narrative or situation, does not have a clash in the statement and does not have an inverted intended sentiment. Hence the tweet is non-ironic.

4 Feature Extraction

For any machine learning algorithm to be applied on irony detection, it is important to design features that can discriminate among tweets. Since it is preferred not to use text

as input for normal machine learning models. Hence it is necessary to implement word embedding, which is a methodology where words, after tokenization, are defined by real-valued vectors in a vector space. For our classification task, we have used a Term Frequency-Inverse Document Frequency (TFIDF) based model, which is a frequency-based word embedding technique, that involves representing each tweet in a vector space where a feature corresponds to a distinct term of the corpus. This was implemented after the tokenisation of the entire corpus and the subsequent development of a vocabulary list for the same. Here the feature is a distinct term and feature value is the TFIDF weight of the term calculated as the product of term frequency (number of times a term occurs in a tweet) and inverse document frequency calculated based on corpus statistics [18].

The TFIDF model is a mechanism for information retrieval that highlights the importance of a word in a corpus [19]. By using this model, we give less importance to terms that are present throughout the corpus since they have poor discriminating power. Using TFIDF model, the entire corpus is converted to tweet-term matrix wherein each row of the matrix corresponds to the vector representation of a tweet. The formula which calculates the TFIDF of a term in a tweet d is:

$$\text{tfidf}(d, t) = \text{tf}(d, t) * \text{idf}(d, t) \quad (1)$$

where $\text{tf}(d, t)$ indicates the frequency of the term t in the tweet d .

The inverse document frequency, for n documents in the corpus is defined as:

$$\text{idf}(d, t) = \log \left[\frac{n}{\text{df}(d, t) + 1} \right] + 1 \quad (2)$$

where the document frequency function $\text{df}(d, t)$ returns the value of how many tweets of the corpus contain the term t at least once.

There may exist many terms (for example, articles and preposition), in an extreme case, which exist in all the documents in a corpus. Mathematically, the idf of these terms should be 0. To prevent the TFIDF to be 0, there is a 1 added to the end of the formula. A smoothing factor has been incorporated into our system to prevent zero divisions. This is done by adding 1 to the numerator and denominator of the idf formula, which is now defined as follows:

$$\text{idf}(d, t) = \log \left[\frac{n + 1}{\text{df}(d, t) + 1} \right] + 1 \quad (3)$$

We have considered the above-mentioned smoothing factor while computing the tweet-term matrix.

Finally, we obtain the tweet-term matrix where each row corresponds to a tweet and each column corresponds to each distinct term in the corpus and each value in the row is the TFIDF weight of the corresponding column term if the term is present in the corresponding tweet. This value is set to 0 if the term is not present in the tweet. Each row is labeled with the label of corresponding tweet.

Using the TFIDF weight criterion in the irony classification task provides valuable insight into how effective the irony classification corpus is to identify patterns and distribution of features across the corpus, which effectively helps us determine how these features are more likely to express which type of irony.

5 Model Development

After developing the tweet-term matrix for our entire corpus, we have divided the dataset into training and testing set. Each model is developed using the machine learning algorithm trained with the training set. For the irony classification, we have considered several machine learning algorithms, such as Naïve Bayes, Support Vector Machine, K-Nearest Neighbor and Random Forest. Each of the four algorithms and the rationale behind using them to test our corpus have been described below.

5.1 Naïve Bayes

Naïve Bayes (NB) Classifier finds probability of a tweet t being in the class C based on the following equation:

$$P(t|C) = P(w_1 w_2 \dots w_k | C) = P(C) \prod_{i=1}^k P(w_i | C) \quad (4)$$

where:

- Tweet t is represented as vector, $[w_1 \ w_2 \dots w_k]$ and w_i is the TFIDF weight of the term corresponding to the i -th column of tweet-term matrix described in the earlier sections,
- $P(w_i | C)$, the probability that i -th feature of the tweet t with value w_i belongs to class C , is calculated based on the assumption that the values of the feature are normally distributed in the class C . So, the observation value w_i of the i -th feature of the tweet, its expected value in class C and the variance of the values of the feature in class C are plugged into the equation of normal (Gaussian) distribution to compute the probability $P(w_i | C)$.
- $P(t|C)$ is called posterior distribution [20] and $P(C)$ is called prior probability calculated as ratio of size of C and the sum of sizes of all classes considered in developing the model.

Given a test tweet, the probability of the tweet being in each class is computed and its label is decided by comparing those probabilities.

Smoothing is applied to deal with data sparseness problem. In our model, we have used a smoothing parameter α , for which we have used Laplace smoothing ($\alpha = 1$) [21]. The Naïve Bayes classifier will converge more quickly than discriminative statistical models and algorithms like logistic regression, which is a significant rationale for using this algorithm for testing on the dataset.

5.2 Support Vector Machine

Vapnik introduced the utility of the Support Vector Machine (SVM) for classification in 1995 [22] and why this method of supervised learning is so robust to overfitting. The

SVM algorithm finds the maximum margin hyperplane in the feature space separating one class from another, defined as

$$wx + b = 0 \quad (5)$$

x being the input vector used for classification and w and b are vectors are learned through the process of implementing the SVM algorithm.

To ensure a globally optimal solution, SVM is used to solve linearly constrained problems like Eq. (6), defined as

$$\min_w \frac{1}{2} ||w||^2 + C \sum_i \xi_i \quad (6)$$

C being the penalty parameter of error terms or the parameter used to control tolerance of outliers of the feature vector set and ξ is the slack variable used to relax linear separability.

Since we have implemented irony analysis with 4 labels, we have used multi-class SVM that uses one vs. all strategy to develop a group of binary classifiers whose predictions are combined to find the label of the test instance though the alternative method of solving multiclass SVM problems in one step by solving a much larger optimization problem is also used in [23]. In text classification problems, high dimensional spaces and feature vectors are the norm and the SVM algorithm is popular in handling such spaces.

5.3 K-Nearest Neighbor

Fix and Hodges [24] describes the K-Nearest Neighbor (KNN) method as an algorithm that takes an unclassified sample as input and determines the class of the sample by finding the mode of the labels of the K nearest neighbors of the input sample. The K nearest neighbors of the input sample is selected from the training set by computing Euclidean distance between the input sample and the training samples. Based on the value of K , the vector space of inputs is divided accordingly [25]. Euclidian distance between two points x and y in the Euclidian space is calculated as follows:

$$E(x, y) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2} \quad (7)$$

This method employs lazy learning, by not learning feature association with the classes during the training phase, rather makes use of the abstraction of data samples during testing. Since there is no training phase, this classifier parses through the entire training set for each prediction. The KNN algorithm was implemented especially to analyse the model's performance on lazy learning.

5.4 Random Forest

Random forest [26] is a kind of ensemble classifier which combines the predictions of many decision trees using majority voting to determine the class for a test instance. Each

decision tree participated in ensembling process is built based on a subset of features chosen randomly from the feature set. The method integrates the idea of “bagging” [27] and the random selection of features.

This algorithm implements multiple decision trees to create subsamples of the data and uses averaging to improve accuracy and stop the occurrence of over-fitting. We use this algorithm for our irony classification task for several reasons - for many datasets, it is proven to be a highly accurate classifier, it runs efficiently on large and high dimensional datasets as the algorithm works by creating subsets of the input data and the process can be split to multiple processors or machines to run parallelly.

6 Experiments and Results

For implementing and testing our models, we have used a manually annotated corpus of 1500 tweets, which resulted in the generation of 1500*20721 feature vector (that is, each tweet is represented as a vector of 20721 dimensions). After splitting the dataset into the training set and the testing set, the machine learning algorithms were trained on the training set to develop the models. Due to the dependence of each term in a document with its neighboring terms, a major feature of the corpus prioritized for the irony classification task was the distinctive manner of expression. This feature was captured and fed into each machine learning model by implementing the word n-gram sequence model the given samples of text used in training.

We have judged the performances of these models for our defined two tasks-(1) irony detection at the coarse-grained level (classification of tweets as ironic or non-ironic) and (2) irony detection at the fine-grained (classification of tweets into one of four classes: Verbal_Irony, Irony_Clash, Situational_Irony and No_Irony).

We have used the standard 10-fold cross-validation [28] which divides the data into equal 10 parts and considers one part consisting of 150 unique tweets as the test set and the remaining 9 parts consisting of 1350 tweets as the training set for each fold. The accuracy on the test set for each fold is recorded and the overall accuracy is computed by averaging the results obtained for 10 folds. To obtain optimal results, we have tuned the parameters of each algorithm to best fit our dataset. The details of parameter tuning have been described in the following section.

For implementation of our models, we have used Google Colaboratory, a Google Cloud Service variant of the Jupyter Notebook, which supports Python 2 and Python 3. The Pandas library was used to simplify and organize the structure of the dataset into dataframes, which is a useful data structure for our text classification task. The library used for the implementation of the machine learning algorithms is scikit-learn. The metric used to assess the quality of the dataset and models for the corpus is accuracy as it is a good metric for the comparison of the four algorithms used and is a relevant metric to check how the model works on unseen data.

6.1 Naïve Bayes

For this model, we did appropriate parameter tuning by setting smoothing parameter, alpha to 1 for Laplace smoothing. The parameter fit_prior [29] was also set. This model

achieves the 10-fold cross validated accuracy of 46.53% for fine-grained task and 56.67% for coarse-grained task.

6.2 Support Vector Machine

Due to its inherent ability to deal with high dimensional data, the Support Vector Machine is one of the best models for text classification. We have used multi-class SVM for the multiclass irony analysis of Bengali tweets.

We have implemented a Support Vector Classification (SVC) model with several kernels [30] for obtaining the optimized results. We have also varied the value of penalty parameter C for obtaining the best results. We have shown in Table 3 and Table 4 how the performance of Support Vector Classifier on our Bengali irony dataset is affected when the choices of kernel and the cost parameter are varied.

Table 3. Accuracy of Support Vector Machine Classifier for fine-grained irony detection task when the choices of kernel and the cost parameter are varied.

SVC Kernels	C = 1	C = 10	C = 100	C = 1000
Linear	47.20	45.93	45.86	44.27
Polynomial	47.13	46.93	43.40	43.27
RBF	47.33	47.13	44.33	43.06

Table 4. Results of Support Vector Classifier algorithm for coarse-grained irony detection task when the cost parameter C is varied.

SVC Kernels	C = 1	C = 10	C = 100	C = 1000
Linear	64.53	65.67	65.07	63.67
Polynomial	67.47	66.47	63.87	62.60
RBF	66.67	66.27	63.80	63.20

As we can see from Table 3 and Table 4, the SVM algorithm achieves the best performance for the fine-grained irony classification task using the RBF kernel and C being set to 1 whereas, for coarse-grained irony classification, SVM achieves the best performance when the polynomial kernel is chosen, and C is set to 1.

6.3 K-Nearest Neighbor

Since the value of K affects the performance of K-nearest neighbor algorithm [31], we have varied the values of k from 1 to 99 and the ten-fold cross validation accuracy

is calculated for each case. Only odd values of K were considered to avoid the ties among the classes. The optimum value of accuracy was found at $k = 79$ for fine-grained classification and at $k = 7$ and 15 for the coarse-grained classification task. Weights of vote cast by each nearest neighbor is set to the inverse of their Euclidian distance.

Since the Euclidian distance method was considered, the power parameter for the Minkowski metric [32] was set as 2. The performances of the KNN model for both the coarse-grained and the fine-grained Bengali irony tweet classification tasks with varying values of K are shown in Table 5 and Table 6.

Table 5. Fine-grained Bengali irony classification performance of K-Nearest Neighbor Algorithm when the value of K is varied.

KNN model	Accuracy (%)
$K = 79$	47.93
$K = 85$	47.67
$K = 81, 99$	47.60
$K = 73$	47.53
$K = 71, 77, 97$	47.47

Table 6. Coarse-grained Bengali irony classification performance of K-Nearest Neighbor Algorithm when the value of K is varied.

KNN model	Accuracy (%)
$K = 7, 15$	62.87
$K = 17$	62.73
$K = 9$	62.67
$K = 27$	62.53
$K = 5, 13, 25$	62.40

6.4 Random Forest

To obtain the best performance with Random forest, we have varied the number of trees from 2 to 256, at intervals of 2^i , where i is taken from 1 to 8. Values after 256 are not taken as, [33] the model accuracy value stagnates as more trees are used. For the Random Forest Classifier, we have not specified a maximum depth of a tree and kept the minimum value of leaves fixed to 1. The accuracy of the Random Forest model for both the Bengali irony tweet classification tasks with varying number of trees in Table 7 and Table 8. As we can see from Table 7 and Table 8, Random Forest with number of trees set to 128 performs the best for the fine-grained Bengali irony tweet classification task and Random Forest with number of trees set to 256 performs the best for the coarse-grained Bengali irony tweet classification task.

Table 7. Fine-grained Bengali irony classification performance of the Random Forest algorithm when the number of trees is varied.

Random Forest model	Number of trees							
	2	4	8	16	32	64	128	256
Accuracy (%)	36.33	39.80	43.53	45.80	46.67	47.67	48.13	47.60

Table 8. Coarse-grained Bengali irony classification performance of the Random Forest algorithm when the number of trees is varied.

Random Forest model	Number of trees							
	2	4	8	16	32	64	128	256
Accuracy (%)	55.27	57.93	59.99	61.73	63.13	64.53	65.93	66.99

6.5 Comparisons of Models and Discussion

By comparing the results of all the machine learning algorithms, we can observe that the Random Forest algorithm with 128 trees performed the best for our fine-grained Bengali irony tweet classification task and showed 48.13% accuracy though the KNN algorithm also achieves a very close results with the value of k set to 79, which is surprising considering the fact that K-Nearest Neighbor algorithm is relatively simple and is implemented by lazy learning principle.

But, for coarse-grained Bengali irony tweet classification task, SVM with polynomial kernel and the cost parameter C set to 1 performed the best among all of our developed machine learning models. For this task, we also observe that Random Forest model with 256 trees achieved the performance very close to the SVM based model. The SVM model achieves 67.47% accuracy whereas the Random Forest model achieves 66.99% accuracy. Since the coarse-grained Bengali irony tweet classification task is basically a binary classification task, SVM performs the best for this task. Our experimental results reveal that the Random Forest model performs consistently well for both the tasks.

We conclude that the correct classification and detection of irony is linguistically difficult, by the low accuracy results of the fine-grained approach. This may be boiled down to the low number of ironic tweets available right now, which caused a class imbalance problem in the fine-grained corpus. This resulted in the predictions ranging in the 40-50% cross-validated accuracy bracket, while the coarse-grained binary classification for irony yielded better results, ranging in the 60-70% cross-validated accuracy bracket. This is because the corpus was far more balanced for tweets that were classified as simply ironic or not ironic. We discuss solutions for the above problem in Sect. 7.

7 Conclusion

In our research, we created a new dataset for irony classification in the Bengali language. This was done due to the discernible lack of any relevant corpus for our specific research,

even though such corpuses are well developed in several other languages. For any new corpus created for natural language processing task, it needs to be tested on several machine learning algorithms to report the baseline results on the dataset. Our paper provides valuable insight into the linguistic analysis required to identify and annotate irony as well as gives an understanding of methods and problems in implementing multi-class irony analysis of Bengali tweets. Though we have used four widely used machine learning techniques for implementing our models, our best models achieve 47.93% accuracy for fine-grained Bengali irony tweet classification task and 67.47% for coarse-grained Bengali irony tweet classification tasks. It shows that classification of Bengali irony tweets is not an easy task.

The major problem we have faced while completing this work is the class imbalance problem. The number of tweets that were not ironic are far greater in number than the rest in the fine-grained dataset. A consequence of the above may be the low separability between the majority class, non-ironic tweets, with the minority classes, which is something which has resulted in the classification accuracy reported in this paper.

The amount of analysis done for this corpus, being the first Bengali corpus for irony classification, should set precedence for other researchers in the field of irony classification of Bengali tweets. On a positive note, the recent surge in activity in the usage of the Bengali script on Twitter will make possible in future to develop the larger corpus and increase it beyond the current 1500 tweet dataset, thus enabling the creation of better feature vectors. We hope that the current corpus will be improved with time with more ironic tweets to handle the class imbalance problem. Further work on irony classification in Bengali will involve word-embedding with Long Short-Term Memory (LSTM) to establish semantic relationships in the corpus and latent semantic analysis for further dimensionality reduction, which may improve the accuracy as well as other performance metrics of the irony classification task.

Acknowledgements. This research work has received partial support from the project entitled “Indian Social Media Sensor: an Indian Social Media Text Mining System for Topic Detection, Topic Sentiment Analysis and Opinion Summarization” funded by the Department of Science and Technology, Government of India under the SERB scheme.

References

1. Sperber, D., Wilson, D.: Irony and the use-mention distinction. *Philosophy*, **3**, 143–184 (1981)
2. Bouazizi, M., Ohtsuki, T.O.: A pattern-based approach for sarcasm detection on Twitter. *IEEE Access*, **4**, 5477–5488 (2016)
3. Sarkar, K., Chakraborty, S.: A sentiment analysis system for Indian language Tweets. In: Prasath, R., Vuppala, A.K., Kathirvalavakumar, T. (eds.) *MIKE 2015. LNCS (LNAI)*, vol. 9468, pp. 694–702. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-26832-3_66
4. Barbieri, F., Saggion, H.: Modelling irony in Twitter. In: *Proceedings of the Student Research Workshop at the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pp. 56–64 (2014)
5. Karoui, J., Zitoune, F.B., Moriceau, V., Aussenac-Gilles, N., Belguith, L.H.: Towards a contextual pragmatic model to detect irony in tweets. In: *The 53rd Annual Meeting of the Association for Computational Linguistics and The 7th International Joint Conference of the Asian Federation of Natural Language Processing*, pp. 644–650 (2015)

6. Van Hee, C.: Can machines sense irony?: exploring automatic irony detection on social media (Doctoral dissertation, Ghent University) (2017)
7. Van Hee, C., Lefever, E., Hoste, V.: Monday mornings are my fave:)# not exploring the automatic recognition of irony in english tweets. In: Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pp. 2730–2739 (2016)
8. Van Hee, C., Lefever, E., Hoste, V.: Guidelines for annotating irony in social media text. version 2.0. Technical Report 16-01, LT3, Language and Translation Technology Team (2016)
9. Van Hee, C., Lefever, E., Hoste, V.: SemEval-2018 task 3: irony detection in English Tweets. In: Proceedings of the 12th International Workshop on Semantic Evaluation, pp. 39–50 (2018)
10. Ghosh, A., Veale, T.: IronyMagnet at SemEval-2018 task 3: a siamese network for irony detection in social media. In: Proceedings of the 12th International Workshop on Semantic Evaluation, pp. 570–575 (2018)
11. Baziotis, C., et al.: NTUA-SLP at SemEval-2018 task 3: tracking ironic tweets using ensembles of word and character level Attentive RNNs. arXiv preprint [arXiv:1804.06659](https://arxiv.org/abs/1804.06659) (2018)
12. Wu, C., Wu, F., Wu, S., Liu, J., Yuan, Z., Huang, Y.: THU_NGN at semeval-2018 task 3: Tweet irony detection with densely connected LSTM and multi-task learning. In: Proceedings of the 12th International Workshop on Semantic Evaluation, pp. 51–56 (2018)
13. Rangwani, H., Kulshreshtha, D., Singh, A.K.: NLPRL-IITBHU at SemEval-2018 Task 3: combining linguistic features and emoji pre-trained CNN for irony detection in tweets. In: Proceedings of the 12th International Workshop on Semantic Evaluation, pp. 638–642 (2018)
14. Rohanian, O., Taslimipour, S., Evans, R., Mitkov, R.: WLV at SemEval-2018 task 3: dissecting tweets in search of irony. In: Proceedings of the 12th International Workshop on Semantic Evaluation, pp. 553–559 (2018)
15. Wilson, D., Sperber, D.: On verbal irony. *Lingua*. **87**(1), 53–76 (1992)
16. Burgers, C., van Mulken, M., Schellens, P.: Finding irony: an introduction of the verbal irony procedure (VIP). *Metaphor Symbol*. **26**, 186–205 (2011)
17. Shelley, C.: The bicoherence theory of situational irony. *Cogn. Sci.* **25**(5), 775–818 (2001)
18. Ramos, J.: Using TF-IDF to determine word relevance in document queries. In: Proceedings of the first instructional conference on machine learning, vol. 242, pp. 133–142 (2003)
19. Yun-tao, Z., Ling, G., Yong-cheng, W.: An improved TF-IDF approach for text classification. *J. Zhejiang Univ. Sci. A* **6**(1), 49–55 (2005). <https://doi.org/10.1007/BF02842477>
20. McCallum, A., Nigam, K.: A comparison of event models for naive bayes text classification. In: AAAI-98 Workshop on Learning for Text Categorization, vol. 752(1), pp. 41–48 (1998)
21. Yuan, Q., Cong, G., Thalmann, N.M.: Enhancing naive bayes with various smoothing methods for short text classification. In: Proceedings of the 21st International Conference on World Wide Web, pp. 645–646. ACM, (2012)
22. Vapnik, V.: *The Nature of Statistical Learning Theory*. Springer, New York (1995)
23. Hsu, C.W., Lin, C.J.: A comparison of methods for multiclass support vector machines. *IEEE Trans. Neural Networks* **13**(2), 415–425 (2002)
24. Fix, E., Hodges, J.: Discriminatory analysis: nonparametric discrimination, consistency properties. USAF School of Aviation Medicine, Randolph Field, Texas (1951)
25. D’Agostino, M., Dardanoni, V.: What’s so special about Euclidean distance? *Soc. Choice Welfare* **33**(2), 211–233 (2009)
26. Breiman, L.: Random forests. *Mach. Learn.* **45**(1), 5–32 (2001)
27. Breiman, L.: Bagging predictors. *Mach. Learn.* **24**(2), 123–140 (1996)
28. Blum, A., Kalai, A., Langford, J.: Beating the hold-out: bounds for k-fold and progressive cross-validation. In: COLT, vol. 99, pp. 203–208 (1999)
29. Albert, J.: Teaching Inference about Proportions Using Bayes and Discrete Models. *J. Stat. Educ.* **3**(3) (1995)

30. Cristianini, N., Scholkopf, B.: Support vector machines and kernel methods: the new generation of learning machines. *AI Mag.* **23**(3), 31 (2002)
31. Yang, Y.: An evaluation of statistical approaches to text categorization. *Inf. Retrieval* **1**(1–2), 69–90 (1999)
32. Lu, B., Charlton, M., Brunsdon, C., Harris, P.: The Minkowski approach for choosing the distance metric in geographically weighted regression. *Int. J. Geogr. Inf. Sci.* **30**, 351–368 (2016)
33. Oshiro, T.M., Perez, P.S., Baranauskas, J.A.: How many trees in a random forest? In: Perner, P. (ed.) *MLDM 2012. LNCS (LNAI)*, vol. 7376, pp. 154–168. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-31537-4_13