International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

Extracting Contextual Feature Form Hinglish Short Text by Handling Spelling Variation at Character and Word Level

[1]Rajshree Singh, [2]Dr. Reena Srivastava

Submitted: 10/02/2023 **Revised**: 13/04/2023 **Accepted**: 06/05/2023

Abstract: The major purpose of sarcasm detection has been to comprehend the text's evaluation. Sarcasm detection is regarded as one of the most provocations in it, and it has been subject to significant provocation. Irony is a unique manner of expressing information that contradicts a notion and creates confusion. Data pre-processing is one of the main duties performed by most developers. Numerous studies on irony detection use a variety of feature extraction techniques. These studies used a variety of machine learning classification models that includes Naive Bayes, Logistic Regression, etc. Precision, recall, and F-score are among the research project results that can be utilized to forecast the most appropriate model. This study discusses numerous methods for detecting sarcasm and irony in text. Extra Tree Classifier and gradient boosting classifier gives the best result having F-Score 95.43 and 95.29 respectively with Wn = 4,Cn = 3 and CWn =4 to detect sarcasm in Hinglish Language.

Keywords: *Pre-Processing; short text; Auto-Encoder; Hinglish*

Introduction

Since the invention of social media, most human interaction takes place online. As a result, textual data produced in enormous amounts is utilized to make useful inferences. People from different cultures utilize social media platforms like Facebook, Twitter, Reddit, and others to communicate with one another and express their thoughts. The study of various language expressions including sarcasm, irony, hostility, humour, hate, etc. has become a keen area of research, thanks to the abundance of data available from social media. Automatic detection of these expressions is currently being extensively researched, particularly in the field of Natural Language Processing (Joshi et al., 2017).

The current trend in sentimental analysis and transliteration is code-mixing. Difficult techniques are required to extract the correct sentiment from code-mixed data. Joshi et al., (2017) conducted research on code-mixed Hindi-English social text data. Code-mixed data requires more pre-processing steps than monolingual data. Using the Support Vector Machine (SVM) classifier, the author proposes numerous pre-processing techniques for mixed scripts and achieves 58.2% accuracy.

Shalini et al. (2018) proposed a distributed representation for extracting sentiments from code-mixed text in multiple languages, including Kannada-English, Hindi-English, and Bengali-English. The author has employed numerous techniques to Sentimental Analysis on Indian Languages (SAIL), including SVM, Fast-Text, Bi-directional Long Short-Term Memory (Bi-LSTM), and Convolutional Neural Network (CNN). CNN is applied with various filter sizes and achieves an accuracy of 71.5 percent for the Kannada-English dataset, whereas Bi-LSTM achieves an accuracy of 60.2 percent for the Hindi-English dataset and 72.5 percent for the Bengali-English dataset.

Choudhary et al. (2018) proposed a novel approach that outperforms the highest existing result in Sentimental Analysis of Code-Mixed

Research Scholar, School of Computer Application, Babu Banarasi Das University, Lucknow
Email - rajshree.singh07@bbdu.ac.in
Dean, School of Computer Application, Babu Banarasi Das University, Lucknow
Email - dean.soca@bbdu.ac.in

Text (SACMT) by 7.6% in terms of accuracy and 10.1% in terms of F-Score. Mishra et al. (2018) proposed a variety of Machine Learning and Deep Learning techniques for processing sentimental analysis of Indian languages. The author achieved a 69% F1-Score for the Bengali-English dataset and a 58% F1-Score for the Hindi-English dataset.

Research Gap: Being a challenging problem, automatic sarcasm detection has been a prominent research topic. Although a significant amount of research has been conducted on sarcasm detection in English, the detection of sarcasm in code-mixed languages such as Hinglish (Hindi-English) remains largely unexplored. Goal of this study is to categorize Hinglish(a blend of Hindi and English) into sarcastic categories.

Material & Method

1. Dataset Creation

5250 tweets make up the current dataset presented in the research (Swami et al., 2018), of which 504 tweets have been classified as sarcastic while the rest 4746 have not been. Given that this dataset is both insufficient and extremely skewed, all deep learning models appeared to incorrectly anticipate that all tweets would not be sarcastic. Performance measures that are sensitive to imbalanced datasets were employed for this: Consider using measurements like precision, recall, or the F-Score instead of accuracy, which can be deceptive in datasets with a class imbalance.

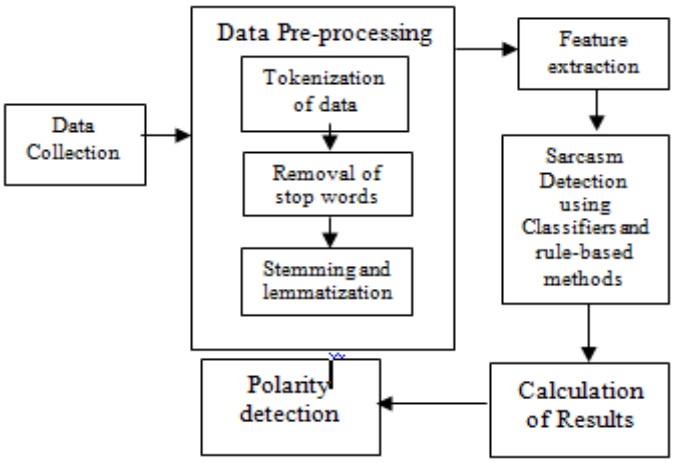

Fig 1: Sarcasm Detection using Machine Learning

Roughly 60% of the samples were down-sampled from the majority classes at random since the degree of imbalance was moderate, bringing it into the mild category. After down-sampling, the minority class contribution increased to about 20–21%. The frequency of terms in a document or group of documents is a prominent focus of natural language processing (NLP) research. It is extremely difficult to adhere to the occurrence rule of the words when speaking Hinglish, nevertheless. because the Hindi words in the dataset are translated into English. It becomes challenging to follow the grammatical norms of a language when it is written into another language. Because Hindi is being transcribed into English in this instance, it is impossible to follow the proper writing convention.

A greater comprehension and formulation of the issues cannot be achieved by merely relying on word frequency. Therefore, this research suggests a better pre-processing strategy.

2. Pre-Processing

In natural language processing(NLP), the most crucial aspect of any machine learning model is data pre-processing(Swami et.al, 2018). Good results are contingent on the quality of the data pre-processing. In the proposed model, #sarcasm is removed from the tweet before data

is pre-processed. Specifically in the following steps:

2.1 Tokenization

In a nutshell, tokenization is the process of dividing text sequences into "tokens" or smaller units(Banerjee et. al, 2020). The tokenization of paragraphs into sentences and sentences into words. Python's Tokenizer() function is used to generate identifiers. Following tokenization, the procedure returns a list of tokens.

2.2. Noise Removal

Eliminate punctuation marks such as a comma, an exclamation point, a quotation mark, and a question mark. This meaningless term is never used in sentences. After noise removal, a catalogue of clean tokens is obtained that can be further processed. Additionally, all characters and words were converted to lowercase. The python translate() function was used to remove punctuation, while the lower() function was used to lower every character in the dataset.

2.3 The elimination of stop words

A stop word is a commonly used word (such as "the", "a", "an", "in", and "i"); these words are been disregarded as they have no bearing on the polarity of the tweet and are therefore of no use to us. For a Python script to remove stop words.

2.4. Normalisation

Normalisation concludes the pre-processing phase. This attempts to level the playing field for all text, for instance by converting all characters to lowercase. The process of normalisation improves text compatibility.

- **Why N-Gram?**

  Chance of misspelling a word is very high(when written in Hinglish), therefore, capturing the characteristics of a sentence can only boost the accuracy(Mandal et. al, 2018). For this, a char n-grams followed by word n-grams method is proposed to boost the accuracy. The proposed approach can capture the dynamics of the sentence.

  For instance, consider Hindi variants of the word profit **"fayada", "fayda", "faayada", "faayda", "faayadaa", "faaydaa"**.

  At word level all are different words, so the algorithm will consider it differently. However, if the word is broken using char n-grams then the algorithm can find some similarity between the words. For instance, the char similarity matrix when char n-grams is taken as 2 is shown in Table I. While, when char n-grams is taken as 3 is Shown in Table II. Table III shows when word N grams employed on char n grams.

Table 1: Char Similarity Matrix for char n Grams=2

Words	**Tokens**						
	Fa	**ay**	**ya**	**ad**	**Da**	**yd**	**aa**
fayada	1	1	1	1	1	0	0
fayda	1	1	0	0	1	1	0
faayada	1	1	1	1	1	0	1
faayda	1	1	0	0	0	0	1
faayadaa	1	1	1	1	1	0	1
faaydaa	1	1	0	0	1	1	1

Table 2: Char Similarity Matrix for char n Grams=3

Words	**Tokens**								
	Fay	**Aya**	**Yad**	**ada**	**ayd**	**Yda**	**Faa**	**Aay**	**daa**
fayada	1	1	1	1	0	0	0	0	0
Fayda	1	0	0	0	1	1	0	0	0
faayada	0	1	1	1	0	0	1	1	0
faayda	0	0	0	0	0	0	1	1	0
faayadaa	0	1	1	1	0	0	1	1	1
faaydaa	0	0	0	0	1	1	1	1	1

Table 3: Word N Grams Employed on Char N Grams

Words	Tokens										
	faay	**ayya**	**yaad**	**adda**	**dayd**	**ydaa**	**ayyd**	**ydda**	**faaa**	**aaay**	**daaa**
fayada	1	1	1	1	0	0	0	0	0	0	0
fayda	1	0	0	0	0	0	1	1	0	0	0
faayada	0	1	1	1	0	0	0	0	1	1	0
faayda	0	0	0	0	0	0	1	1	1	1	0
faayadaa	0	1	1	1	0	0	0	0	1	1	1
faaydaa	0	0	0	0	0	0	1	1	1	1	1

As the value of N in N-Char grams increases, the chances of identifying similar words increase. Similarly, the probability becomes stronger, moving towards W-grams at char levels and Wgrams at the word level. Emoticons belonging to any one of the following are preserved:
':-)', ':)', '(:', '(-:',':-D', ':D', 'X-D', 'XD', 'xD','

3. TF-IDF (Term Frequency-Inverse Document Frequency)

Words are transformed into vectors by TF-IDF(Term Frequency -Inverse Document Frequency). The words that appear in the fewest sentences in the supplied document are found by IDF and TF, respectively. TF determines how frequently a word is present in the given document. The TF-IDF scoring system determines the frequency of each word used in a certain document(Patwa et. al, 2020). The limitation of TF-IDF is that semantic information cannot be provided.

4. Sarcasm Detection Classifiers

Classification models like Naïve Bayes, Decision Tree, Random forest, AdaBoost, Gradient Boost will be implemented for detecting sarcasm. Their performance will be evaluated on the basis of Precision, Recall and F1-Score.

Result

Precision, Recall and F1-Score are calculated to evaluate performance of each classification model(Bhargava et. al, 2016).

$$\text{Precision (P)} = \frac{\text{TP}}{\text{TP+FP}} \quad (1)$$

$$\text{Recall (R)} = \frac{\text{TP}}{\text{TP+FN}} \quad (2)$$

$$\text{Accuracy} = \frac{\text{TP+TN}}{\text{TP+TN+FP+FN}} \quad (3)$$

$$\text{FScore} = \frac{2*(\text{Precision} * \text{Recall})}{(\text{Precision} + \text{Recall})} \quad (4)$$

All the models are performed for the following cases:
1. Word N Grams (Wn): 2,3,4
2. Char N Grams (Cn):2,3,4
3. Char Word N Grams (CWn): 2,3,4

In this research, autoencoders have been utilised for dimensionality reduction.

Autoencoders are a type of neural network architecture that is used for dimensionality reduction and feature learning(Pradhan & Sharma, 2022). It consists of an encoder and a decoder, which are trained to reconstruct the original input data from a lower-dimensional representation. Autoencoders can be trained in an unsupervised manner, using only the input data without any labels. This makes them useful for learning useful features from the data without the need for labelled examples.

The proposed autoencoders consist of 11 layers where the first 5 are the compression layers, the 6th is the central layer and the last 5 are the expansion layers. The reduced features are extracted from the central layer and then re-train all the existing machine learning models. At each layer, features are reduced by a factor of 1.45 and rounded to the nearest integer if the outcome is in decimal. The reduction factor of 1.45 is considered in such a way that it will reduce the features by around 10 times. . The

autoencoders model is trained by minimizing the root mean squared error between the final output after expansion and the initial input. The model is run for 6,000 epochs for the selected combination as shown in the figure below.

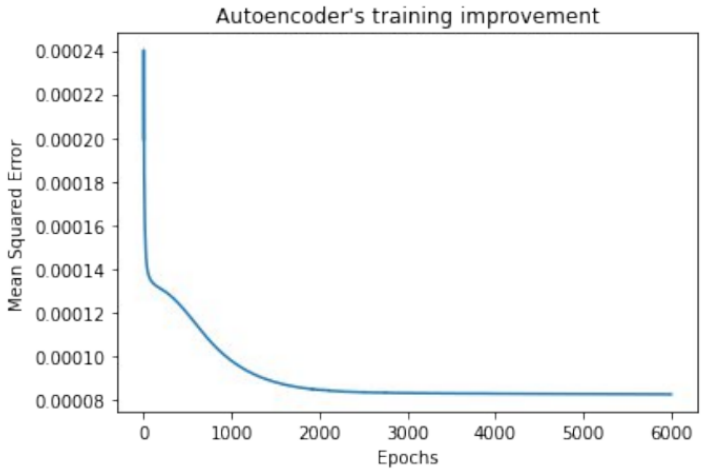

Fig 2: Training Improvement of Autoencoder

The following cases for dimensionality reduction because accuracy are almost the same for lower values:

Table 4: Initial features for the selected combinations

Combination	Original Features	Reduced Features
Wn= 3; Cn=3 ; CWn=3	30673	3300
Wn= 3; Cn=3 ; CWn=4	43425	4672
Wn=3 ; Cn=4 ; CWn=3	42761	4600
Wn=3 ; Cn=4 ; CWn=4	56613	6091
Wn=4 ; Cn=3 ; CWn=3	30932	3328
Wn=4 ; Cn=3 ; CWn=4	43684	4700
Wn=4; Cn=4 ; CWn=3	43020	4628
Wn=4 ; Cn=4 ; CWn=4	56872	6119

After Feature Reduction for example:

Table 5: Reduced Features Result (Wn=4,Cn=3,CWn=4)

	Precision	**Recall**	**F Score**
Logistic Regression	97.33	87.5	92.15
Naïve Bayes	23.74	59.74	33.97
Random Forest	99.92	91.09	95.30
Decision Tree	88.04	79.90	83.77
Extra Tree Classifier	99.44	91.74	95.43
Gradient Boosting Classifier	99.07	91.79	95.29
Ada Classifier	88.57	90.85	89.69

Results on the reduced features are as follows:

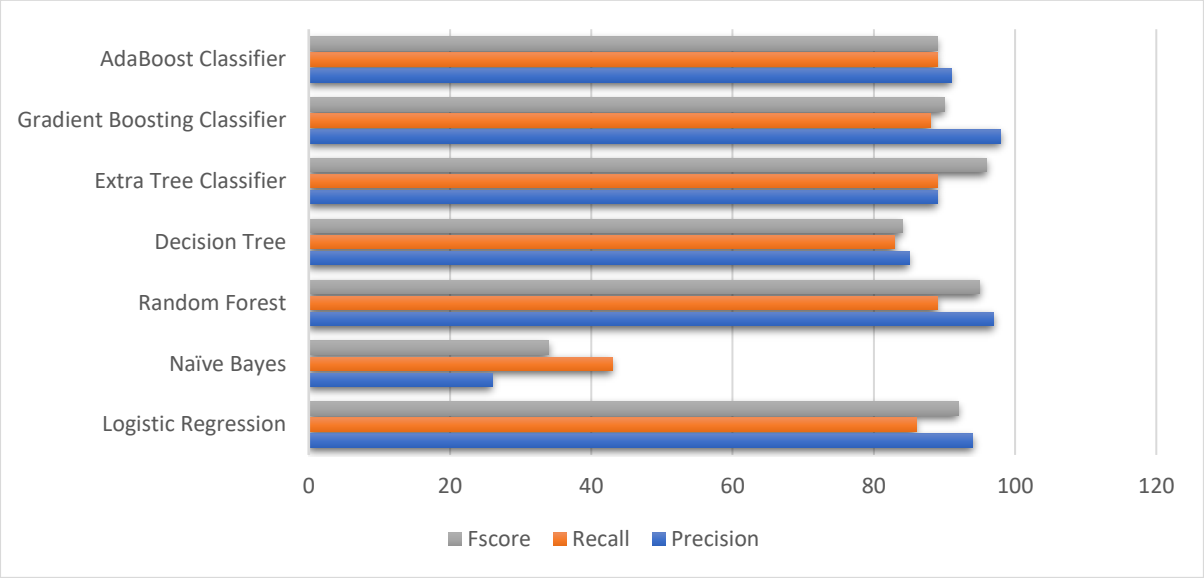

Fig 3: Performance Measure after Dimensionality Reduction

Conclusion

Twitter is a microblogging tool that many people use to share their ideas and opinions. sarcasm is to speak in ways which are the opposite of actual meaning in intent to be uncomfortable or to make a mockery of another person. Sarcasm can be found everywhere, especially on social media. Identifying sarcasm is one of the most important and difficult skills. Classifier models: Naive Bayes, Logistic Regression, Decision Tree, Random Forest, Extra Tree Classifier, Gradient Boosting Classifier, and Adaboost Classifier are used in this paper to identify sarcasm in Twitter. Various types of datasets are discussed in Hinglish Language. The sarcasm identification task is a categorization issue in which sarcastic utterances are distinguished from their non-sarcastic counterparts. After feature reduction, it can be well observed from figure3 and Table 5, that Extra Tree Classifier and gradient boosting classifier gives the best result having F-Score 95.43 and 95.29 respectively with Wn = 4,Cn = 3 and CWn =4 to detect sarcasm in Hinglish Language.

References

[1] Aditya Joshi, Pushpak Bhattacharyya, and Mark J. Carman. 2017. Automatic sarcasm detection: A survey. ACM Comput. Surv., 50(5).

[2] Sahil Swami, Ankush Khandelwal, Vinay Singh, Syed Sarfaraz Akhtar, and Manish Shrivastava. A corpus of english-hindi code-mixed tweets for sarcasm detection, (2018).

[3] Shalini, K., Hb, B.G., Kumar, M., Soman, K.P.: Sentiment analysis for code-mixed Indian social media text with distributed representation. In: 2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI), pp. 1126–1131 (2018). https://doi.org/10.1109/ICACCI.2018.8554835.

[4] Mishra, P., Danda, P., Dhakras, P.: Code-mixed sentiment analysis using machine learning and neural network approaches (2018)

[5] Choudhary, N., Singh, R., Bindlish, I., Shrivastava, M.: Sentiment analysis of code-mixed languages leveraging resource rich languages. In: 19th International Conference on Computational Linguistics and Intelligent Text Processing, 2018, Hanoi, Vietnam (2018).

[6] Pradhan, R., Sharma, D.K. An ensemble deep learning classifier for sentiment

analysis on code-mix Hindi–English data. *Soft Computing*(2022).

[7] R. Bhargava, Y. Sharma and S. Sharma,: Sentiment analysis for mixed script Indic sentences. In: *2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, Jaipur, India, pp. 524-529(2016).

[8] Patwa, P., Aguilar, G., Kar, S., Pandey, S.J., Pykl, S., Gambäck, B., Chakraborty, T., Solorio, T., & Das, A. SemEval-2020 Task 9: Overview of Sentiment Analysis of Code-Mixed Tweets, (2020).

[9] Mandal, S., Mahata, S.K., & Das, D. Preparing Bengali-English Code-Mixed Corpus for Sentiment Analysis of Indian Languages.(2018)

[10] Banerjee, S., Jayapal, A.K., & Thavareesan, S. NUIG-Shubhanker@Dravidian-CodeMix-FIRE2020: Sentiment Analysis of Code-Mixed Dravidian text using XLNet. *Fire*(2020).