**A Novel Approach for Sentiment Analysis of Hinglish**

**Text**

Himanshu Singh Rao1, Jagdish Chandra Menaria1, Satyendra Singh Chouhan2

1 College of Technology and Engineering,

Udaipur 313001, INDIA

{Singhraoh1, Jagdishcm41297}@gmail.com

2 Malviya National Institute of Technology (MNIT),

Jaipur, 302017, INDIA

[sschouhan.cse@mnit.ac.in](mailto:sschouhan.cse@mnit.ac.in)

**Abstract.** *Hinglish* is an informal language, which is written entirely in Latin script but contains informal words, phrases or even slang in a piece of writing, written both in Hindi and English. The field of sentiment analysis is vast and a lot of work has been done in the past couple of decades yet sentiment analysis of *Hinglish* texts remains unexplored. Therefore, in this paper, we present a novel approach for sentiment analysis of *Hinglish* text. The proposed approach uses algorithms such as Stemming, *Levenshtein* distance and *Soundex* index for preprocessing the data. Thereafter, it applies various classification models to identify the polarity of *Hinglish* text. The experimental results show the effectiveness of the proposed approach.

**Keywords.—***Hinglish* text**,** Sentiment analysis, *Soundex* index, *Levenshtein*

distance.

# Introduction

Sentiment analysis is a type of data mining that measures the inclination of people’s opinions through natural language processing (NLP), computational linguistics and text analysis, which are used to extract and analyze subjective information from the Web - mostly social media and similar sources. The analysed data quantifies the general public's sentiments or reactions toward certain products, people or ideas and reveal the contextual polarity of the information.

Sentiment analysis helps data analysts within large enterprises gauge public opinion, conduct nuanced market research, monitor brand and product reputation, and understand customer experiences. As per comprehension review of Ravi and Ravi [1], most of the work in this field has been done on text written in pure English language and a few works is done in other languages like Arabic, Spanish, Chinese etc. The amount work done on Indian languages further decreases.

In this paper, we did the sentiment analysis of one of the most used languages in Indian context i.e. *Hinglish*, which is written entirely in Latin script but contains informal words, phrases or even slang in a piece of writing, written both in Hindi and English. For instance, consider a movie review written in latin text but in Hindi language:

“*gaane film mein achhae lagae gae hain*”

To accompany sentiment classification (whether positive or negative) with such text, we applied bag of words approach. For pure English text, work has already been done but for *Hinglish* everything is needed to be done from scratch. Like while creating sparse matrix, the main problem is to match the same Hindi words written with different spellings, e.g., “paisa” and “pesa” both have the same meaning as “money” in English, but can be written in different ways by different users. To counter this problem we use technique of *Levenshtein* method to find most similar words and finally used *Soundex* function to give the words a score a score based on their pronunciation. If the difference between the new word and to be matched word is less than a threshold value then both words are supposed to be same else different column is made in sparse matrix. A dictionary of *Hinglish* words is also used in the model to improve the accuracy of the approach.

For classification purposes, we used various classification techniques like SVM, logistic regression, Random Forest, Naive Bayes and Decision Tree, and calculate their performance metrics and evaluate the best model suited for *Hinglish* text classification. The contribution of this work is significant for many reasons. This work can be useful for e-commerce companies, movie productions and social media content analysis as it covers analyzing precious customer’s feedback written in different languages which cannot be analyzed using traditional methods.

The rest of the paper is organized as follows. Section 2 presents the related work. In Section 2, we present the proposed approach. Section 4 discusses the experiments and results. Conclusions are given in Section 5.

# Related Work

In this section, we present the literature work in the field of sentiment analysis using machine learning techniques.

Richa et al [2] applied dictionary-based approach to analyze the polarity of review. In this work, the result is the collection of reviews among positive, negative and neutral reviews of the sentiment of the sentence. Kumar et al [3] experimented with different combinations of feature selection methods and a host of classifiers using term frequency-inverse document frequency feature representation. They carried out in total 840 experiments in order to determine the best classifiers for sentiment expressed in the news and Facebook comments written in *Hinglish*. The experiment concluded that group of term frequency-inverse document frequency-based feature representation,

Radial Basis Function Neural Network, and gain ratio selection as the best combination to classify sentiment expressed in the *Hinglish* text.

Pooja et al [4] applied Hindi SentiWordnet (HSWN) to find the polarity of reviews made on Hindi movie. HSWN is used to find the polarity of review words and then find overall polarity of the review. The main aim of the work was to improve the word net and to analyse the polarity of the review. For the words those do not exist in wordnet, were first translated into English and then added to wordnet based on the result and hence wordnet will also be improved. To find the aggregate polarity of the review, each word is given score when found in HSWN and overall score can be calculated to find the polarity of the review.

Ramanathan et al [5] made a lightweight stemmer for Hindi, this stemmer worked just on some basic ground rules and is valid only for Devnagri script. We used his work in Devnagri script to be applied in Latin script. Thus providing us with a lightweight stemmer that works for *Hinglish* language.

Shanshank et al [6] worked on analysing sentiments of text containing both Hindi and English words. They used statistical method to find the polarity. In this work, if the frequency of positive words in the statement is more, then the statement is positive and vice versa. In this work, each word is tagged as E if word is English or H if the word is Hindi. Then correct spelling of each word is found. Sounds like “haw”, “boo‟, “oopps” are also considered by the model. Then, Roman Hindi is transliterated to Devanagari Hindi and wordnet is used to find review of the statement.

Kaur et al [7] used a dictionary based approach to classify *Hinglish* text based, they used simple pre -processing model and spell corrections, to correctly identify the incorrect spellings in the *Hinglish* text. Their approach is based on tf/idf model in which unigram bigram and trigram feature selection models are used. Negation handling is also performed, for sentimental analysis they used popular classification algorithms that are used in the industry to classify *Hinglish* text.

Mathur et al [8] worked on multilingual sentiment analysis. This model used Multi Input – Multi -Channel Transfer Learning with multiple feature inputs to classify offensive tweets. Their model of multi -channel CNN-LSTM has been pre trained on English tweets through transfer learning. However, spelling variations and the lack of grammar rules introduce ambiguity in the model and overall making model harder to train and therefore results were altering. Their proposed MIMCT model outperforms other baseline supervised classification models.

The literature survey shows that most of the work on *Hinglish* text has been done entirely using machine learning models and no proper implementation of preprocessing is done regarding *Hinglish* language. These approaches are not very effective and they didn’t focus on the preprocessing part i.e. to say that that the dataset cleaning was not done efficiently solely based on *Hinglish* texts.

In the light of the above works, we present a novel approach that classifies the *Hinglish* text. The proposed approach will not only work properly for *Hinglish* language but also for other languages whose dataset is limited. The detailed discussion of the proposed approach is given in Section 3.

# Proposed Work

In the proposed approach, we used the dictionary of stopwords containing 1036 words and IIT KGP dataset of 29437 words for our *Hinglish* dictionaries to check the consistency of words and then added various methods like Stemming(on *Hinglish* words specifically) , *Levenshtein* distance and *Soundex* index for the words in *Hinglish* text to match as closely as possible to our dictionaries of stopwords and IIT KGP , if they don’t meet the required criterion (Selected using Stemming , Levenshtein Distance, *Soundex* Index) then the words is declared new and passed on to the feature extraction phase, else the word is removed if found in stopwords, or the various operations are performed on the word using Stemming ,*Levenshtein* or *Soundex* index. The overall approach is shown in the Figure 1. The description of each step of process is as follows.

## Dataset Acquisition

Due to the lack of dataset in *Hinglish* language, a new dataset was built from scratch using Google Transliterator 1 from the Hindi movie review dataset created by IIT Patna2,later converted into *Hinglish* movie review dataset. Total 1100 reviews were converted into *Hinglish* in which 550 were positive reviews and 550 were negative reviews. Stopwords dictionary 3 was used, which contains 1036 stopwords in *Hinglish* language. The dataset was created manually by transliterating all 29374 words into Hindlish language using Google Transliterator.

## Data Preprocessing

Text preprocessing involves the following tasks.

## Tokenization*:*

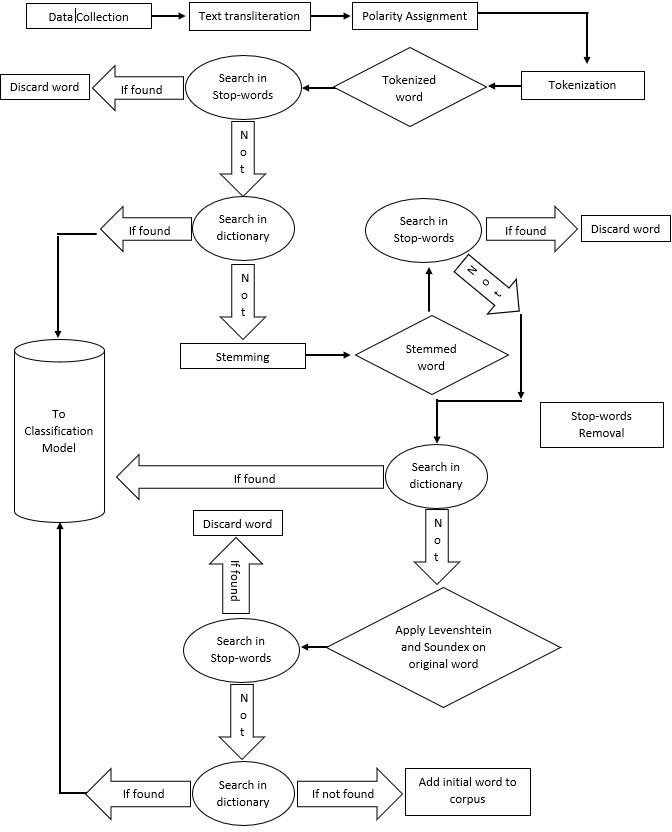
It is the process of breaking up a sequence of strings into pieces such as words, keywords, phrases, symbols and other elements called tokens.

In this some characters like punctuation marks, tags are also discarded and all letters are converted into lower case.

1 <https://www.google.com/inputtools/services/features/transliteration.html>

2 [https://www.iitp.ac.in/~ai-nlp-ml/resources.html](https://www.iitp.ac.in/%7Eai-nlp-ml/resources.html)

3 [https://github.com/TrigonaMinima/*Hinglish*NLP/blob/master/data/assets/stop\_*Hinglish*](https://github.com/TrigonaMinima/HinglishNLP/blob/master/data/assets/stop_hinglish)



**Fig 1**. Model for *Hinglish* text polarity classification

* + 1. **Preprocessing model***:* This model is used for removing stopwords and finding close relationships between the words in the movie review and the words in the dictionary e.g., the word “achaaa” and “acha” have close relationship as to say they sound similar and have exact meaning. But the word “achaaa” is not in the dictionary but the word “acha” is, so to remove this ambiguity we perform various operations on the word which are 1) Stemming 2) *Levenshtein* distance 3) *Soundex* Index.

1. **Stemming.** Usually stemming is done on English or Hindi text only. We have various libraries such as NLTK that performs stemming on English text and

Lucenes HindiStemmer4 for stemming in Hindi. The problem is that the Hindi stemmers are for Devnagri script not Roman. Therefore, implementing the same logic, we found a regex which does the same work as Hindi stemmer but in Roman script.

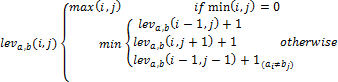
*Regex : re.sub(r'(.{2,}?)([aeiougyn]+$)',r'\1', word)* (1)

**Table 1.** Stemming table

|  |  |
| --- | --- |
| **Input Word** | **Stemmed word** |
| dosti | Dost |
| doston | dost |
| boliye | bol |
| bola | bol |
| jana | ja |
| jaenge | ja |

The above regex, deletes all vowels along with g, y, n from the end of the word, but leaves at least a 2 characters long stem, so that the words like “aayenga” do not completely vanish. Although this Stemmer does not work perfectly for all the *Hinglish* words as *Hinglish* language is too informal and versatile, still it covers majority of *Hinglish* words.

1. ***Levenshtein* distance.** The *Levenshtein* distance is a string metric for measuring the difference between two sequences. Or in other words the *Levenshtein* distance between two words is the minimum number of singlecharacter edits (insertions, deletions or substitutions) required to change one word into the other. The *Levenshtein* distance between two strings a,b (of length

|a| and |b| respectively) given by leva,b(|a|,|b|) where

(2)

4 <http://hitesh.in/2012/stemming-transliterated-hindi/>

Table 2 shows the *Levenshtein* Distance between two *Hinglish* words.

**Table 2.** *Levenshtein* Distance between two words.

|  |  |  |
| --- | --- | --- |
| **Word 1** | **Word 2** | ***Levenshtein* Distance** |
| kahaanee | Kahani | 3 |
| paisa | Pesa | 2 |
| ishq | Ishk | 1 |
| paani | Paani | 0 |

1. ***Soundex* Index.** *Soundex* is a phonetic algorithm for indexing names by sound, as pronounced in Latin script. The goal is for homophones to be encoded to the same representation so that they can be matched despite minor differences in spelling. Since similar sounding words are mostly used in *Hinglish* we have taken a review and converted each word into its *Soundex* index and then the *Soundex* index was compared to the *Soundex* index of the words in the dictionaries. Thus, *Soundex* becomes a very powerful tool while comparing words that have similar sounds in the dictionary. Table 3 shows the *Soundex* index of various *Hinglish* words and their absolute difference.

**Table 3.** *Soundex* index of various words and their absolute difference.

|  |  |  |
| --- | --- | --- |
| ***Soundex* index of word in dictionary** | ***Soundex* index of word in movie review** | **Absolute *Soundex***  **difference** |
| Kahani – K500 | Kahaanee – K500 | 0 |
| Anupam – A515 | Anupaa – A510 | 5 |
| Kiska – K200 | Kisne – K250 | 50 |
| Drishya – D620 | Drishti – D623 | 3 |
| Himanshu – H552 | Himanshi – H551 | 1 |
| Ameer – A560 | Ameen – A550 | 10 |
| Jaa – J000 | Jaunga – J520 | 520 |
| Abhi – A100 | Aaabhi – A100 | 0 |

|  |  |  |
| --- | --- | --- |
| Kahan – K500 | Kahn - K500 | 0 |
| Toofan – T150 | Tufan – T150 | 0 |
| Yahi – Y000 | Yehi – Y000 | 0 |
| Acha – A200 | Aachaa – A200 | 0 |
| Toh – T000 | Tw – T000 | 0 |

Combining all these operations on *Hinglish* words it becomes a very efficient approach while comparing *Hinglish* words both in their meaning and in their phonetic transcription. Using these operations each word is modified accordingly using Stemming, Levenshtein or *Soundex* Index and is thus matched more precisely with the dictionaries.

## Example (Data preprocessing)

Consider the sentence in the movie review is:

“*Kahaanee ke abhaav me unake vyaktigat prayaas kaa prabhaav kam ho jaataa ha*i.”

* Tokenization.

The above sentence will be tokenized into

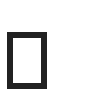
[*'kahaanee', 'ke', 'abhaav’, 'me', 'unake', 'vyaktigat', 'prayaas', 'kaa', 'prabhaav', 'kam', 'ho', 'jaataa', 'hai'*]

Special characters will also be removed and all letters are converted into lower case. Thereafter, as the per the preprocessing shown in the Figure 1, the stopwords found in the Stopwords dictionary were *ke , me ,*

*kaa – after applying* ***stemming*** *the word changed to ‘ka’ ho, hai*

after applying ***Levenshtein*** and ***Soundex*** the nearest word for “kahaanee‟ was found to be “kahani‟ ,therefore the word “kahaanee‟ is replaced with “kahani‟.

Last corpus created is [*kahani abhaav unake vyaktigat prayaas prabhaav kam jaataa*]

 Feature Extraction

Each word is then fed into Bag of words model and sparse matrix of corpus is created for each review. The 2000 words with max frequencies were stored in sparse matrix and then fed to classification model.

## Classification

After the preprocessing, bag of word is used. It is the count of how many times a word occurs in the collection of sentences.

In this model occurrence of each word or the word count is used as feature. Bag of Words model feature selection plays a crucial role in classifying. It tells us the how the words will be used as the features while classifying a review or text.

There are three feature selection model models: *unigram, bi gram and N gram*; in this paper unigram feature selection model is used

* *Unigram Model*: It takes individual words present in a sentence as features, the whole sentence is divided into individual words.

Example: *mujhe toh film boht acchi lagi*

Feature Set:{*mujhe,toh,film,boht,acchi,lagi*}(pre-processing not done)

* These words are then fed into vectorizer that count the occurrences of each word, and transforms words into features based on their frequencies

To determine the polarity of the sentence classification algorithms such as *Naive Bayes, Decision Tree, Support Vector Machine, Logistic Regression* and *Random Forest* classifiers were used.

# Result and Discussion

The proposed approach is implemented using Python 3.7 using machine learning and natural language processing libraries such as NLTK, scikit-learn, etc. After Pre- processing and feature extraction the dataset containing 1100 samples of positive and negative polarity were divided into training and testing test. The classifiers used on the following samples were Naive Bayes, Decision Tree, Support Vector Machine, Logistic Regression and Random Forest.

The classification methods were used with their default parameters that are configured in scikit-learn library, except the number of features, trees and depth – these were customized according to the size of the data and limitations associated with the use of computing resources. These classifiers were evaluated based on the accuracy and the following performance metrics:

* **Precision** - Precision is the ratio of correctly predicted positive observations to the total predicted positive observations. High precision relates to the low false positive rate.

*Precision = True Positives/True Positives + False Positives* (3)

* **Recall** (Sensitivity) - Recall is the ratio of correctly predicted positive observations to the all observations in actual class. It is different from precision in the fact that it is a quantity measure to check how many relevant results are being returned by the algorithm while precision checks from the selected items, how many of them were relevant

*Recall = True Positives /True Positives + False Negatives* (4)

* **F1 score** - F1 Score is the weighted average of Precision and Recall. Therefore, this score takes both false positives and false negatives into account.

(5)

Table 4 shows the results obtained of the proposed approach on the above mentioned performance metrics.



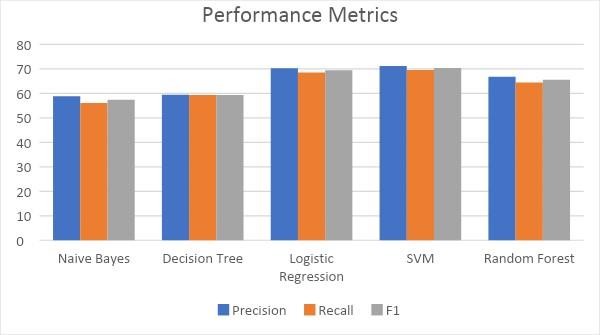
From the results, following observation made:

* + Support Vector Machine classification method with the given data of movie reviews is the best (accuracy – 70.90%, precision – 71.79%, recall – 69.55% and F1 score – 70.31%) classification method based on the accuracy, precision, recall and F1 score in comparison to that of other classifiers.
  + Naïve Bayes classification method is less stable method as the values of average classification accuracy, precision and F1-score are spaciously distributed in comparison to other methods.

Figure 2 graphically illustrates that the average values of classification accuracy

,precision, recall and F1 -score of Logistic Regression and Support Vector Machine are similar and Support Vector Machines has achieved higher average classification

accuracy, precision ,recall and F1 – score results in comparison to Naïve Bayes, Decision Tree ,Logistic Regression and Random Forest but the difference is not statistically significant, except in the case of Naïve Bayes and Random Forest.



**Fig. 2.** Graphical Representation of various performance metrics

# 4 Conclusion and Future Directions

In this paper, we proposed an approach for the sentimental analysis of Hindi text written in English, i.e., *Hinglish*. We also build the dataset and dictionaries from scratch. The dataset is open sourced for further research work in the *Hinglish* language. We implemented various algorithms like Stemming, *Levenshtein* and *Soundex* that efficiently preprocess data unlike used in the past approaches. The processed data was fed into various classifiers using unigram Bag of Words model.

The comparison of various classifiers including Naive Bayes, Decision Tree, Support Vector Machine, Logistic Regression and Random Forest are also presented in the paper. Out of these classifiers Support Vector Machine did the best job while classifying movie reviews and pointing out the polarity of reviews with accuracy – 70.90%, precision – 71.79%, recall – 69.55% and F1 score – 70.31%.

In Future work, the proposed approach can be improved by using different feature selection models like Bi gram and N gram and also by using hybrid classification models.

# References

1. K. Ravi and V. Ravi, "Sentiment classification of *Hinglish* text," *2016 3rd International Conference on Recent Advances in Information Technology (RAIT)*, Dhanbad, 2016, pp. 641-645.
2. Sharma, R., Nigam, S., and Jain, R., “Polarity Detection of Movie Reviews in Hindi Language”. *International Journal on Computational Science & Applications* 4, 4 (2014), pp. 49-57.
3. Ravi, K. and Ravi, V. “Sentiment classification of *Hinglish* text”. *3rd IEEE International Conference on Recent Advances in Information Technology (RAIT)* (2016) pp. 641-645.
4. Pandey, P. and Govilkar, S. “A Framework for Sentiment Analysis in Hindi using HSWN”. *International Journal of Computer Applications* 119, 19 (2015).
5. Ramanathan, Ananthakrishnan, and Durgesh D. Rao. "A lightweight stemmer for Hindi." *the Proceedings of EACL*. 2003.
6. Sharma, S., P. Y. K. L., S. and Rakesh Chandra, B. “Sentiment analysis of codemix script”. *IEEE International Conference on Computing and Network Communications (2015)*
7. Kaur, Harpreet, Veenu Mangat, and Nidhi Krail. "Dictionary based sentiment analysis of *Hinglish* text." *International Journal of Advanced Research in Computer Science* 8.5 (2017).
8. Mathur, Puneet, et al. "Did you offend me? classification of offensive tweets in *Hinglish* language." *Proceedings of the 2nd Workshop on Abusive Language Online (ALW2)*. 2018.