Insult Detection in Hindi

Advisor: Prof. Amitabha Mukherjee

Chetan Dalal(11218) [*], Shivyansh Tandon(11690) [†]

Indian Institute of Technology, Kanpur

Abstract: The aim of our project is to detect comments in Hindi that may be considered as insulting to other participants of conversation. We use feature selection tools like skip grams, n-grams, negation feature and second person feature to build a vector model. We reduce the large number of featurers to an appropriate level by the Chi-Squared test. We employ supervised learning like logistic regression and SVM for training and testingthe data. We have also compared these techniques against the insults in English language. We create our own datset for insults in English by using source materials from various blogs and by translating available English insults to Hindi via Google Translate. Since the direct translation of source was not very efficient we had to manually change to get a meaningful translation. We have also presented a qualitative study on accuracy of Google Translate while translating from English to Hindi.

Keywords: Sentiment Analysis • Insult Detection• Hindi corpus• Supervised classification

1. Introduction

The growth of internet has been immense in the past few years. According to The Telecom Regulatory Authority of India (TRAI), the number of Internet subscribers in India are 164.81 million as of March 31, 2013, and is now worlds third largest internet user. Even social networking sites like Facebook have over 100 million active users in India (Times of India). Thus, detection of inappropriate use of language on internet which might harm the user is of utmost importance. With the advent of technology many devices and operating systems are now supporting Hindi, and hence there is a need for an insult detection system in Hindi. Insults act as a repelling force for new users and also prevent regular users to participate in discussions in future. It is also frustrating to find foul language when looking for something. It is also not possible to have a human moderator to monitor this enormous amount of data.

We focus on comments that may be insulting or harmful for other participant. Insults can be of many types like racial slurs, reference to handicap, foul language and provocative words. The indirect insults like sarcasm, disguise and crude words are not identified by the method. We aim at detecting the direct and extreme insults.

[*] *E-mail: chetand@iitk.ac.in, Department of Computer Science & Engineering*
[†] *E-mail: shivyans@iitk.ac.in, Department of Mathematics & Statistics*

2. Related Works

Various works on Sentiment Analysis have been done for the English language. The work by Xiang, G., Hong [2] dealt with offensive tweets with the help of Topical Feature Discovery over a Large Scale Twitter Corpus by using Latent Dirichlet Allocation model. The work by Razavi [3], Inkpen have used a Insulting and Abusing language dictionary along with features like frequency to run the three-level classification algorithm.

For Hindi, there has been a work on sentiment analysis in movie reviews by using a Semi-Supervised approaches to train a Deep Belief Network. A recent work on Hindi Subjective Lexicon was done by Bakliwal et al., where they worked on Hindi Polarity Classification. They used Hindi WordNet and used synonyms and antonyms of a given word in Hindi.

These previous research works have not been able to detect insults on third persons. The research have been directed on trivial and extreme second-person insults. However no research on Insult detection has been done for the Hindi language due to small corpora. We aim at creating a classifier that detects peer-to-peer insults by applying supervised learning techniques.

3. Datasets

One difficult part of our project was to obtain the dataset. Since there was no such database we manually created one. We tackled this problem in different ways.

- Collected comments from various Hindi blogs and forums(400 entries).
- Employed Google Translate to convert the available Kaggle's English dataset into Hindi(1000 entries) and then manually modified it to retain the context.
- Around 70% of the input is negative strings(non-insults).
- Created a list of bad words in Hindi, procured from various Hindi websites.
- Did a qualitative study on how Google Translate works for English to Hindi.

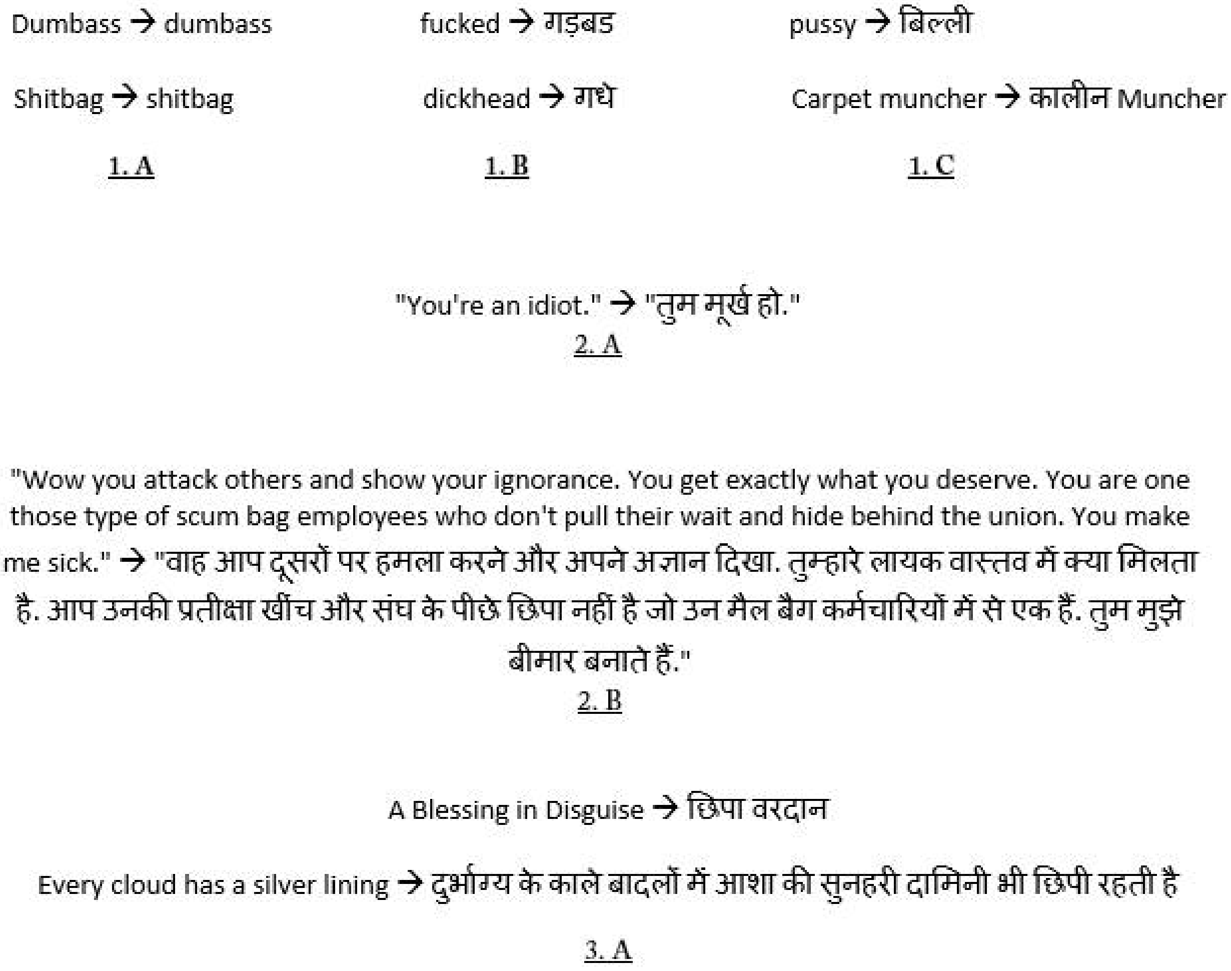
3.1. Google Translate: A Qualitative Study

We used Google Translate to convert English dataset into Hindi. As opposed to what one might expect with Google, it is not so awesome. It has its limitations but can at least help the user to understand the general meaning of a foreign text. Some languages produce better results than others, and works especially well when the target language is English and source language is one of the languages of European Union. The following were our conclusions from the study we did on English to Hindi translation according to the general results.

- Word Translation

  - Some of the English insults were not translated since the corpus avialabe was small.(Fig 1A)
  - Severity of several insults got reduced. Also, there is a similar many-to-one mapping of many insults to a single mid insult.(Fig 1B)
  - Meanings of many insults get lost in the literal translation of the statement. (Fig 1C)

- Sentence Translation: Short sentences are good(Fig 2A) while long ones lose meaning(Fig 2B).
- Idioms: Some are translated well(Fig 3A) while some are translated word to word and their meanings get lost(Fig 3B).



Dumbass → dumbass

Shitbag → shitbag

1. A

fucked → गड़बड

dickhead → गधे

1. B

pussy → बिल्ली

Carpet muncher → कालीन Muncher

1. C

"You're an idiot." → "तुम मूर्ख हो."

2. A

"Wow you attack others and show your ignorance. You get exactly what you deserve. You are one those type of scum bag employees who don't pull their wait and hide behind the union. You make me sick." → "वाह आप दूसरों पर हमला करने और अपने अज्ञान दिखा. तुम्हारे लायक वास्तव में क्या मिलता है. आप उनकी प्रतीक्षा खींच और संघ के पीछे छिपा नहीं है जो उन मैल बैग कर्मचारियों में से एक हैं. तुम मुझे बीमार बनाते हैं."

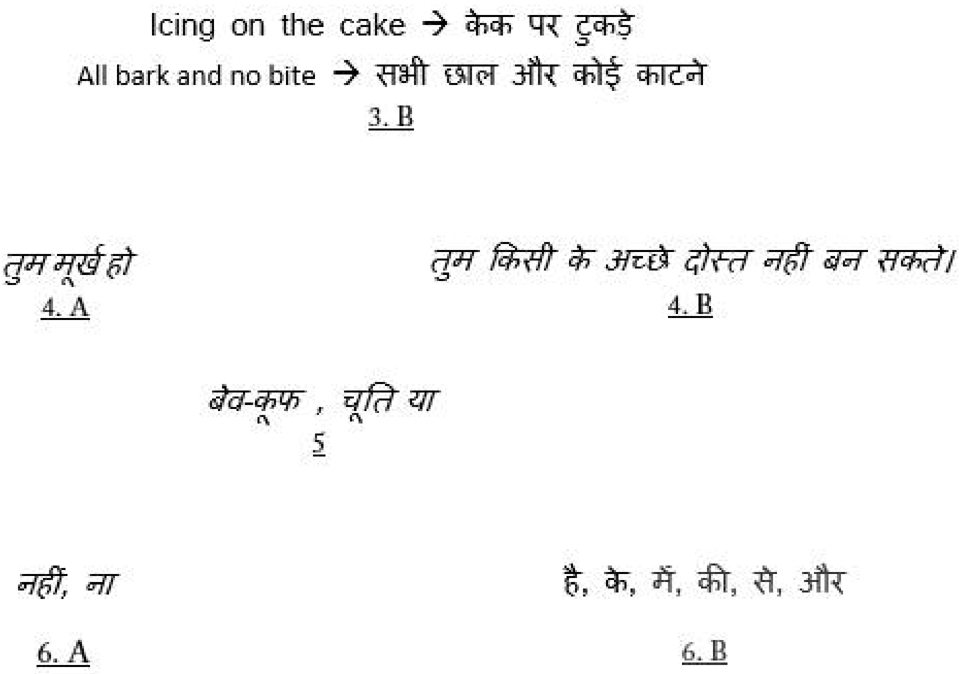
2. B

A Blessing in Disguise → छिपा वरदान

Every cloud has a silver lining → दुर्भाग्य के काले बादलों में आशा की सुनहरी दामिनी भी छिपी रहती है

3. A

Figure 1: Examples of insult detection in Hindi



Icing on the cake → केक पर टुकड़े

All bark and no bite → सभी छाल और कोई काटने

3. B

तुम मूर्ख हो

4. A

तुम किसी के अच्छे दोस्त नहीं बन सकते।

4. B

बेव-कूफ , चूति या

5

नहीं, ना

6. A

है, के, में, की, से, और

6. B

Figure 2: Examples of insult detection in Hindi

4. Implementation

Our method follows a 4-step process.

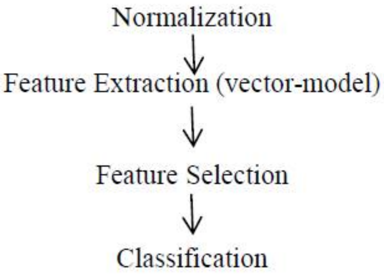


Figure 3: Implementation Model

4.1. Normalization

A raw source cannot be used directly as an input. The data from our dataset has to be modified before it can be used for Insult detection. It also helps to reduce unnecessary computation. Although, we must be careful not to lose useful information.

4.1.1. Removal of unwanted strings

First we remove the unwanted strings. The unwanted strings can be the encoding parts such as \\xa0, \\xc2, \\n or some HTML tags or some English words that were not translated. We also remove words that have single occurrence or come too frequently as they have no effect on insult but they are necessary for grammar.

4.1.2. Stemming

The second task that our code does is reducing words to their root. There are many words that have similar meanings but due to grammar or usage are modified. Since this results in unnecessary increase in number of features, we reduce them to their root. This helps to reduce the number of features and enables the code to produce good results on lesser data. (Fig 6A)

4.2. Feature Extraction

The words have no meaning for a computer and thus have to be translated into a vector form to perform operations on it. We convert the strings into vectors which are then used by Supervised Machine Learning algorithms.

4.2.1. Tokenizing

Split the data into tokens. The tokens can be characters, n-grams or words. The code uses words as tokens and builds 2,3,4,5 n-grams for feature vector.

4.2.2. Counting

Enumerate the tokens generated in previous step for each text string. This way a matrix (generally sparse) is created, representing our data (text strings) where the number of occurrence of each token is a feature for that string. The size of matrix is S x F, where S is the size of training data and F is size of the vocabulary.

4.2.3. Skip-Grams

We can have long distance related features in the input data. So in addition to n-grams we use skip grams and thus increase the size of our feature matrix. This feature has a parameter which determines the number of words to skip between two words.

4.2.4. Second Person Feature

We also add to our feature the set of words that occur after a second person words. We use this based on our observation of our dataset, which had insults based on similar structure. This feature is important as it improves accuracy.(Fig 4A)

4.2.5. Negation Feature

We also added a feature that considers the words with negative implications which inverts the meaning of a string. We then give extra weight to such sentences. This has significant improvement in results.(Fig 4B)

4.2.6. Normalization

Some words occur nearly in every comment and hence do not have any significance. So, we reduce its importance by removing all such words with high frequency in the beginning and for the rest of the features we use a measure of the relevance of the word by using TF-IDF.(Fig 6B)

4.2.7. TF-IDF

Some terms that appear frequently in a few statements but rarely in other comments tend to be more relevant and specific for those comments and therefore more useful for detecting insults. Hence we multiply each term with its corresponding inverse document frequency (IDF) and obtain a tf-idf vector for each string. This is also called weighing each term based on its inverse document frequency.

4.3. Feature Selection

Since the number of features generated are high it will be inefficient to compute directly on all of them. They might not be as important in deciding if a string is an insult or not. We use a feature selection algorithm Chi-Squared test, to select k best features. We chose this parameter equal to be 200 for the current dataset. We apply this statistical method to our feature matrix, constructed earlier.

4.3.1. Chi-Squared Test

This is a statistical test basically to find if a pair of variables on a data is statistically dependent or independent. Our method uses:

- Label of string, i.e. insult or not
- Occurrence or Non-occurrence of a feature

as the pair of variable. The feature which scores high in this test is selected for the training classifiers, and rest of the features are discarded. Thus we have optimized our model and selected only the relevant features and do not perform unnecessary computation.

4.4. Classification

This is the final step. We now apply machine learning algorithms to learn a classifier. We are using SVM and Logistic Regression to train our classifier and then combine the results of both algorithms to obtain a final classifier. This classifier is then used to classify whether a given string is an insult or not.

5. Results

Feature	**Accuracy**	**Recall**	**Precision**
Without second person and Without negation	86.84	0.60	0.84
With second person and Without negation	87.96	0.67	0.86
Without second person and With negation	86.96	0.63	0.86
With second person and With negation	87.71	0.68	0.84

The algorithm did not have any previous results to compare with, however, we have obtained good results in comparison with insult detection in English. We have such high results mainly because of many-to-one mapping while translating from English to Hindi. We have **improved our database to include more negative text strings** (non insults) to check the efficiency of our algorithm. However, the results follow because of similar and extreme nature of Hindi insults.

6. Conclusion and Future Work

In our attempt to detect inuslting comments, we have employed a supervised approach based on SVM and and Logistic Regression. We also took care of special cases like negation and second person. We have created a good dataset to further build our algorithms on.

We aim to increase the size of our dataset in future, and also look for alternative approaches that might be better and faster than the current supervised approach. To make use of the large amounts of online data available, we also plan on creating a model based on unsupervised approach. There is also a scope to identify insults that have been tampered(Fig 5).

7. Acknowledgement

We would like to thank our professor Dr. Amitabha Mukherjee. The project wold not have been possible without his encouragement and diligent efforts. We would also like to acknowledge the help of our TAs Ms. Sunaskhi Gupta and Mr. Mamidela Seetha Ramaiah.

References

[1] Sentiment Analysis For Hindi Language, MS Thesis IIIT-H 2013, Piyush Arora. Vestibulum turpis quam, tristique vel dapibus et, scelerisque luctus enim.

[2] Xiang G., Hong J., & Rose, C. P. 2012. Detecting Offensive Tweets via Topical Feature Discovery over a Large Scale Twitter Corpus, Proceedings of The 21st ACM Conference on Information and Knowledge Management, Sheraton, Maui Hawaii, Oct. 29- Nov. 2, 2012

[3] Amir H. Razavi, Diana Inkpen, Sasha Uritsky, and Stan Matwin. 2010 Offensive language detection using multi-level classification. In Proceedings of the 23rd Canadian Conference on Artificial Intelligence, pages 1627.

[4] D. Das and S. Bandyopadhyay. Labeling emotion in bengali blog corpus a fine grained tagging at sentence level. In Proceedings of the Eighth Workshop on Asian Language Resources, pages 4755, Beijing, China, August 2010. Coling 2010 Organizing Committee.

[5] Starter Code: http://www.kaggle.com/c/detectinginsults%25E2%2580%2593in-social-commentary/forums

[6] Datasets:
http://www.kaggle.com/c/detecting-insults-in-social-commentary/data
http://khabar.ndtv.com/news/zara-hatke/90-per-cent-indians-are-idiots-justice-katju-357932
http://ek-ziddi-dhun.blogspot.in/2008/08/blog-post_22.html
http://www.bbc.co.uk/hindi/
http://loksangharsha.blogspot.com/
http://www.noswearing.com/
http://www.youswear.com/