# Offensive Query Detection

**Akash** **Atishay** **Gaurav** **Roopak**

## Abstract

The offensive query detection model is a machine learning system that identifies hate speech in online comments and tweets through the use of a bag-of-words model. The model is trained on a labeled dataset, which enables it to recognize patterns and features associated with offensive language. The system has been augmented with a language-translation model to enable it to operate across multiple languages.

The model is an extension of the Naive Bayes model and has undergone several experimental enhancements, including multilingual capabilities. The system is designed to detect and flag offensive language in online forums, social media, and other digital communication platforms. The model's ability to recognize offensive language makes it a useful tool for social media platforms and other online communities focused on preventing hate speech and promoting responsible digital discourse.

## 1 Problem Statement And Motivation

The problem is about detecting offensive content in social media posts and classifying them into different categories such as safe or hateful or offensive and clean. The task requires developing an approach to detect and classify the content with high precision and recall and comparing various models for hate-speech detection.

Additionally, the approach should be able to generalize to different languages without the use of large-scale Universal Language Models. The overall objective is to develop a method that can effectively filter out offensive content from social media platforms, ensuring a safe and positive user environment.

Offensive query detection has become a critical issue in social media, particularly on platforms like Twitter, where the rise of hate speech and online harassment has caused significant harm to individuals and communities. Offensive queries and hate comments on Twitter have been found to create a toxic and hostile environment that can lead to real-world harm, including harassment, discrimination, and violence.

The problem of offensive query detection is complex, as it requires identifying and understanding the context and intent of user-generated content, which can be highly nuanced and difficult to interpret accurately. For instance, a seemingly innocuous comment can be interpreted as offensive or harmful depending on the cultural, social, or historical context in which it is made. Additionally, the sheer volume of content on social media platforms makes it challenging for human moderators to review and flag potentially offensive queries or comments effectively.

Several machine learning models have been developed for offensive query detection to address this issue, each with its own strengths and weaknesses. One of the most widely used models is the rule-based model, which employs a set of predefined rules to identify offensive queries based on keywords, patterns, and regular expressions. This model is relatively straightforward and can be trained quickly, but it is limited by its inability to detect subtle forms of offensive language or understand the underlying intent behind a user's query or comment.

Another model that has gained popularity in recent years is the supervised learning model, which uses labelled datasets to train a machine learning algorithm to accurately classify offensive queries or comments. This approach is more effective at detecting nuanced forms of offensive language and can adapt to language use and context changes. However, it requires a large and diverse dataset of labelled examples, which can be difficult to obtain, and it may struggle with detecting novel or unfamiliar forms of offensive language.

Big companies like Twitter, Facebook, and Google have implemented various offensive query detection models to moderate their platforms effectively. For instance, Twitter uses a combination of rule-based and machine-learning models to detect and flag potentially harmful or offensive content on its platform. On the other hand, Facebook uses a machine learning-based model called DeepText, which can understand the meaning and context of user-generated content in multiple languages. Google uses a combination of machine learning models and human moderators to review and flag potentially offensive content on its platforms, including YouTube, Gmail, and Google Search.

In conclusion, offensive query detection is a critical issue in social media, requiring sophisticated machine learning models to accurately identify and flag potentially harmful or offensive content. While there is no one-size-fits-all solution, a combination of rule-based and machine learning-based models and human moderation can help create a safe and inclusive online environment for all users.

## 2 Related Work

### 2.1 Convolutional Neural Networks (CNNs)

CNNs use a series of convolutional layers to extract features from the textual input, which can be words, phrases, or sentences. These features are then passed through a series of pooling layers to reduce the spatial dimensionality of the feature maps while retaining important information. Finally, the features are passed through one or more fully connected layers that perform the final classification of the query as offensive or non-offensive.

One advantage of using CNNs for offensive-query detection is their ability to capture the local context of the input query, allowing them to detect subtle nuances in language use and identify potentially offensive content. Additionally, CNNs can learn to generalize to new and unseen forms of offensive language, making them effective at detecting novel forms of offensive content.

However, a limitation of CNNs is their inability to capture the full semantic meaning of language. They may also require a large amount of labeled data for training and struggle to detect more subtle forms of offensive language.

Several companies, including Twitter and Facebook, have used CNNs for offensive-query detection on their platforms. Twitter, for instance, uses a combination of rule-based and machine learning-based models, including CNNs, to detect and flag potentially harmful or offensive content on its platform.

### 2.2 Google's BERT

Google's Bidirectional Encoder Representations from Transformers (BERT) is a state-of-the-art natural language processing model that has been used for offensive query detection. BERT is trained on large amounts of text data, allowing it to learn complex relationships between words and phrases and generate more accurate predictions.

BERT works by pre-training a transformer-based neural network on large amounts of unlabelled text data, followed by fine-tuning labeled offensive and non-offensive query data for the specific task of offensive-query detection. During the fine-tuning process, BERT learns to classify new queries based on their context, considering the full meaning of the language used.

One advantage of using BERT for offensive-query detection is its ability to capture the full meaning of language, including more subtle forms of offensive language like sarcasm and irony. Additionally, BERT requires less labeled data than CNNs, making it more scalable.

However, a disadvantage of BERT is its high computational cost, making it slower and more resource-intensive to train and use than other models. Additionally, BERT's performance may be affected by the quality and diversity of the training data and may require fine-tuning on additional datasets to improve its accuracy.

### 2.3 Recurrent Neural Networks (RNNs)

Recurrent Neural Networks (RNNs) have also been used for offensive query detection. Unlike CNNs, RNNs are designed to capture sequential dependencies in the input text, making them effective at detecting offensive language that involves longer-term dependencies.

RNNs work by processing input text sequentially, one word or character at a time, and updating a hidden state that encodes information from all previous inputs. This allows RNNs to capture context and long-term dependencies that other models may miss.

One advantage of using RNNs for offensive-query detection is their ability to handle variable-length inputs, making them suitable for detecting offensive language in social media posts or other forms of unstructured text. Additionally, RNNs can learn to generalize to new and unseen forms of offensive language, helping them effectively detect novel forms of offensive content.

However, a limitation of RNNs is that they can suffer from the vanishing gradient problem, where the gradients used to update the network weights can become too small to make meaningful updates. This can make it difficult to train deep RNNs, which can limit their performance.

## 3 Method

### 3.1 Translation

For the purpose of detecting hate speech, we employed two distinct machine-learning approaches for translating non-English languages into English. The first method utilized Google Translate, which operates as a Python module (GoogleTrans) and translates data online from Hindi to English. However, this approach was limited by the requirement of an active internet connection, resulting in potential time delays.

The second method we utilized involved building an extensive dictionary and using it to replace all Hindi words with their corresponding English translations. However, this approach was limited by the dictionary's offline nature and potential omission of certain Hindi words.

Despite the limitations of these approaches, they allowed us to translate other languages into English, which was a crucial step in our project for detecting hate speech. Moving forward, additional machine-learning methods and tools can be further explored to enhance the accuracy and efficiency of our translations.

### 3.2 Pre-Processing

Our preprocessing approach began by collecting and cleaning the dataset, which involved removing any irrelevant or redundant data. We then performed several standard text preprocessing techniques, including stop word removal, stemming, and other cleaning methods.

Stop words are commonly used words that add little to no meaning to a sentence, such as ”and” and ”the”. Removing these stop words helps reduce the dataset's dimensionality, making it easier to analyze and process. Stemming involves reducing words to their root form, allowing for a more effective analysis of related words.

In addition to these techniques, we also performed additional cleaning methods such as removing special characters, emojis, links, and account mentions and converting all text to lowercase to ensure consistency in the dataset. The resulting preprocessed dataset was a foundation for developing our hate speech detection model.

### 3.3 Base Model

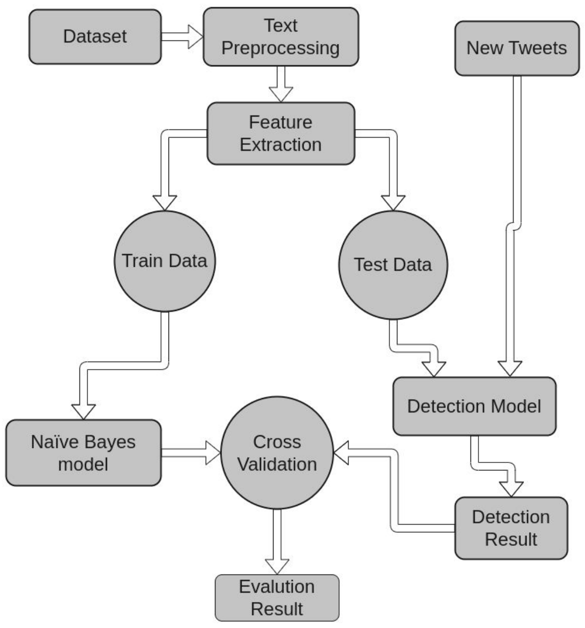

Figure 1: Pipeline for our model

Naive Bayes is a commonly used classification algorithm that can also be applied for hate speech detection. The algorithm works by analyzing the probability of an input text being classified as hate speech or not based on the occurrence of certain words or features within the text.

To implement Naive Bayes for hate speech detection, the algorithm first needs to be trained on a labeled dataset of hate speech and non-hate speech texts. During the training phase, the algorithm calculates the probability of the occurrence of certain words or features in hate speech and non-hate speech texts. It uses this information to build a classification model.

Once the model is built, it can be applied to new, unlabelled texts to determine if they are likely to be hate speech or not. The algorithm does this by analyzing the probability of each word or feature within the text and then using Bayes' Theorem to calculate the probability of the entire text being hate speech.

## 4 Experiments

### 4.1 Bag-of-Words

In the context of hate-speech detection, the bag-of-words method is a commonly used approach to represent textual data as a set of features. Our first model used this method to train a machine learning model on a labeled dataset, in which the data set was treated as a collection of individual words or n-grams, and each token was counted and represented as a feature vector.

The bag-of-words model used in our first model involves several steps. First, the dataset is tokenized, and each token is counted to represent a feature vector. Then, based on the frequency of each word in the labeled dataset, each word is assigned a score if it frequently appears on a particular label.

When the model is given a query text, it calculates the score of each word based on the assigned scores and adds up the total score for each label. The label with the highest total score is then predicted as the label for the query text.

$$\text{score(c,priority,wc)} = \text{priority} * (\text{c/wc})$$

This method allows our model to classify text based on the frequency of words in the labeled dataset. However, it has some limitations, such as not capturing the order or context of words. To improve accuracy, more advanced NLP methods can be used in conjunction with the bag-of-words approach.

### 4.2 TF-IDF

The term frequency-inverse document frequency (TF-IDF) method is used in natural language processing to represent textual data as a set of features. Our second model used this method to detect hate speech by calculating the probability of a query being offensive, given a label.

The TF-IDF method involves several steps. First, the dataset is tokenized, and each token is counted to represent a feature vector. Then, the term frequency (TF) of each token is calculated by dividing the number of times a token appears in the text by the total number of tokens. The inverse document frequency (IDF) of each token is calculated by dividing the total number of documents in the dataset by the number of documents (here labels) that contain the token. Finally, the TF-IDF score of each token is calculated by multiplying the TF by the IDF.

When the model is given a query text, it calculates the TF-IDF score for each token in the text and uses these scores to calculate the probability of the text being offensive, given a label. The label with the highest probability is then predicted for the query text.

$$\text{tf-idf(t,d,D)} = (1 + \log_{10} \mathrm{tf}(t,d)) \cdot \log_{10} \frac{|D|}{\mathrm{df}(t,D)}$$

This method allows our model to take into account both the frequency of words in the text and the frequency of words in the dataset as a whole, thus providing a more nuanced representation of the text. Although relatively less expensive computationally, it has its limitations, such as not capturing the context of words. To overcome these limitations, more advanced NLP methods can be used in conjunction with the TF-IDF approach.

| Method | Accuracy | Precision | F1 Score |
|---|---|---|---|
| TF | 0.836 | 0.835 | 0.91 |
| TF-IDF | 0.841 | 0.839 | 0.912 |
| Bag-of-Words | 0.913 | 0.932 | 0.948 |
| Variation 4 | 0.907 | 0.917 | 0.946 |

Table 1: **Performance metrics for different methods on English dataset**

### 4.3 Translation Methods

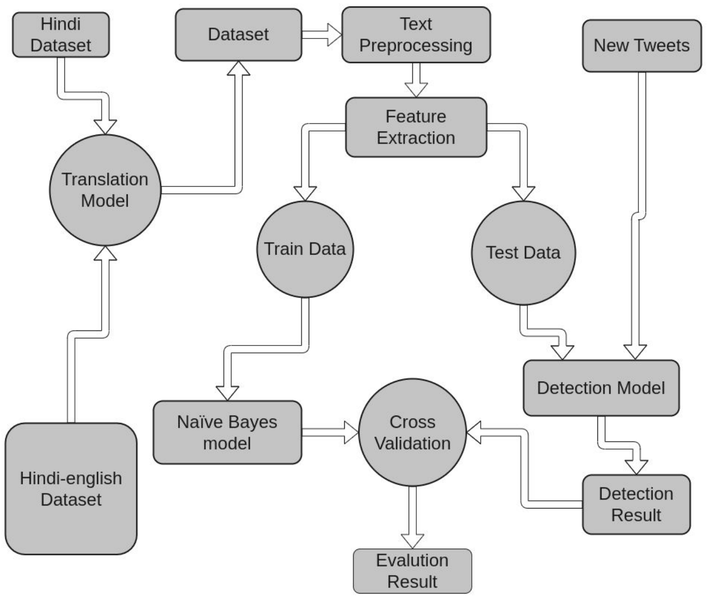

Figure 2: Pipeline for multi-lingual model

Initially, Google Translate was used as the translation tool for the project to translate Hindi text into English. However, due to its online nature and slow speed, an alternative solution was sought. To optimize the process, a CSV file was obtained containing nearly 70,000 Hindi words and their corresponding English translations.

Using this file, a dictionary was created, which allowed the project team to translate the Hindi text into English word by word. A bag-of-words approach was employed, making the process faster and more accurate. As a result, the team was able to achieve accurate translations (token mapping) from Hindi to English in a timely and efficient manner, without relying on an online tool.

| Method | Accuracy | Precision | F1 score |
|---|---|---|---|
| TF | 0.789 | 0.974 | 0.234 |
| +Bad Word | 0.826 | 0.781 | 0.390 |
| +high threshold | 0.827 | 0.594 | 0.606 |

Table 2: **Performance metrics for different methods on Hindi dataset**

## 5 Analysis

The offensive query detection model uses machine learning to detect hate speech in online comments and tweets. It has been trained on a labeled dataset and extended to be multilingual with a language-translation model. Heuristics such as pre-classification based on curse words and higher thresholds for classification have improved the system's performance, allowing it to learn offensive language patterns effectively. The system has shown promising results in detecting hate speech and can aid in social media content moderation.

Despite the system's success in detecting hate speech, there are still some limitations and scope for improvement. Firstly, the system may not be effective in detecting new and evolving forms of hate speech. This is because the model is based on a labeled dataset and may not be able to identify new patterns and features of offensive language that may arise. Secondly, the model may struggle to detect sarcasm or irony, which can lead to false positives or false negatives. Additionally, the model's accuracy may vary depending on the language, dialect, or cultural context, which can affect the system's multilingual capabilities. To address these limitations, the system could be further improved by incorporating more extensive datasets, using more advanced natural language processing techniques, and considering the cultural context of the language used in the dataset.

## 6 Appendix

### 6.1 References

Davidson, T., Warmsley, D., Macy, M., Weber, I. (2017). Automated Hate Speech Detection and the Problem of Offensive Language. In *Proceedings of the 11th International AAAI Conference on Web and Social Media (ICWSM).*

Fortuna, P., Nunes, S., Bizarro, P. (2012). Automated detection of offensive language behavior on social networking sites. In *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12).*

Zampieri, M., Malmasi, S., Nakov, P., Rosenthal, S., Farra, N., Kumar, R., ... Mubarak, H. (2020). SemEval-2020 Task 12: Multilingual Offensive Language Identification in Social Media (OffensEval 2020 Shared Task). *arXiv preprint arXiv:2006.07235.*

Park, S., Fung, P. (2017). Detecting aggressive tweets with recurrent neural networks. In *Social Media in State Politics: Mining Policy Agendas Topics.*

Wiegand, M., Ruppenhofer, J., Klein, E. (2018). AllenNLP: A Deep Semantic Natural Language Processing Platform.

Wulczyn, E., Thain, N., Dixon, L. (2017). Ex Machina: Personal Attacks Seen at Scale. In *Proceedings of the 26th International Conference on World Wide Web.*

Billard, B. D. R. (2021). English-Hindi Dictionary. https://github.com/bdrillard/english-hindi-dictionary/

| Name | Experiments | Ideation | Comments |
|---|---|---|---|
| Atishay Jain (20CS30008) | Analysed translation schemes<br>Implemented translation with synonym expansion<br>Modified basmodel to get F1 score of 0.6 | Extended the model to multiple languages using translation with modifications. | |
| Roopak Priydarshi (20CS30042) | Analysed OffensEval submissions.<br>Implemented preprocessing pipeline.<br>Collected English - Hindi datasets. | Layer sentiment analysis over Naive Bayes.<br>Add synonym matching to augment base model. | |
| Akash Das (20CS10006) | Analyzed Naïve Bayes model.<br>Implemented Preprocessing.<br>Implemented TF and TF-IDF. | Extended Naïve Bayes with term frequency and document/label frequency. | |
| Gaurav Malakar (20CS10029) | Explored CNN-RNN-based available models.<br>Implemented Sentiment analysis over Naive Bayes.<br>Collected English datasets. | Use of Naive Bayes as base model.<br>Augmentations to the basic BOW models. | |

Table 3: Work Distribution of Group Members (Group 12 - Information Retrieval, Spring 2023)