Troll Detection System

A PROJECT REPORT

Submitted by

Darshal Thakkar(18BECE30030)

Darshil Vora(18BECE30035)

Bhavya Pandya(18BECE30026)

Dhairya Patel(219SBECE30004)

In fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

in

Computer Engineering

LDRP Institute of Technology and Research, Gandhinagar

Kadi Sarva Vishwavidyalaya

April, 2022

LDRP INSTITUTE OF TECHNOLOGY AND RESEARCH GANDHINAGAR

CE-IT Department

CERTIFICATE

This is to certify that the Project Work entitled **"Troll Detection System"** has been carried out by **18BECE30030** under my guidance in fulfilment of the degree of Bachelor of Engineering in Computer Engineering Semester-8 of Kadi Sarva Vishwavidyalaya University during the academic year 2021-22.

Prof. Jayana Kaneria	**Dr. Shivangi Surati**
Internal Guide	**Head of the Department**
LDRP ITR	**LDRP ITR**

LDRP INSTITUTE OF TECHNOLOGY AND RESEARCH GANDHINAGAR

CE-IT Department

CERTIFICATE

This is to certify that the Project Work entitled **"Troll Detection System"** has been carried out by **18BECE30035** under my guidance in fulfilment of the degree of Bachelor of Engineering in Computer Engineering Semester-8 of Kadi Sarva Vishwavidyalaya University during the academic year 2021-22.

Prof. Jayana Kaneria	**Dr. Shivangi Surati**
Internal Guide	**Head of the Department**
LDRP ITR	**LDRP ITR**

LDRP INSTITUTE OF TECHNOLOGY AND RESEARCH GANDHINAGAR

CE-IT Department

CERTIFICATE

This is to certify that the Project Work entitled **"Troll Detection System"** has been carried out by **18BECE30026** under my guidance in fulfilment of the degree of Bachelor of Engineering in Computer Engineering Semester-8 of Kadi Sarva Vishwavidyalaya University during the academic year 2021-22.

Prof. Jayana Kaneria	**Dr. Shivangi Surati**
Internal Guide	**Head of the Department**
LDRP ITR	**LDRP ITR**

LDRP INSTITUTE OF TECHNOLOGY AND RESEARCH GANDHINAGAR

CE- Department

CERTIFICATE

This is to certify that the Project Work entitled **"Troll Detection System"** has been carried out by **219SBECE30004** under my guidance in fulfilment of the degree of Bachelor of Engineering in Computer Engineering Semester-8 of Kadi Sarva Vishwavidyalaya University during the academic year 2021-22.

Prof. Jayana Kaneria
Internal Guide
LDRP_ITR

Dr. Shivangi Surati
Head of the Department
LDRP_ITR

TABLE OF CONTENTS:

❑INTRODUCTION:

❑ Introduction:

- Troll is usually defined as somebody who provokes and offends people to make them angry, who wants to dominate any discussion or who tries to manipulate people's opinions and thoughts. The problems caused by such persons have increased with the diffusion of social media. Therefore, on the one hand, press bodies and magazines have begun to address the issue and to write articles about the phenomenon and its related problems while, on the other hand, universities and research centers have begun to study the features characterizing trolls and to look for solutions for their identification.
- This problem generates situations, repeated daily, in which users with fake accounts, or at least not related to their real identity, publish news, reviews or multimedia material trying to discredit or attack other people who may or may not be aware of the attack.
- These acts can have great impact on the affected victims' environment generating situations in which virtual attacks escalate into fatal consequences in real life.
- In this paper, we present a methodology to detect and associate fake or harmful troll on Twitter social network which are employed for defamatory activities to a real profile within the same network by analyzing the content of comments generated by both profiles.
- With the rise of social media, people are exposed to large amounts of information on social media platforms, which creates the opportunity for some organizations to distribute rumors, misinformation, and speculation, in an attempt to manipulate the opinion of the public.
- Daily Twitter has been flooded with false news, and propaganda-spreading trolls. This dedicates the importance of detecting troll tweets to protect the public from the inappropriate content in the social network.
- The aim of this work is to determine whether a given tweet is a troll tweet, based solely on its content. The task of computational detection of troll tweets in a multilingual corpus is a linguistic and machine learning problem. It can be considered as an authorship verification task, based on the assumption that the troll tweets are generated by some sort of "troll farm." Some reports about a "troll farm" suggest that work at the "troll farm" is strictly regulated.

- This problem generates situations, repeated daily, in which users with fake accounts, or at least not related to their real identity, publish news, reviews or multimedia material trying to discredit or attack other people who may or may not be aware of the attack.
- These acts can have great impact on the affected victims' environment generating situations in which virtual attacks escalate into fatal consequences in real life.
- In this project, we present a methodology to detect and associate fake or harmful troll on social networks which are employed for defamatory activities to a real profile within the same network by analyzing the content of comments.

❑PURPOSE:

- We have developed the Troll Detection to detect the harmful comment that affect human temperament and Troll Detection will automatically detect the harmful comment. This portal can detect the comments on Twitter or on any other Application that provoke the affect people.

- This portal will be helpful to stop this kind of acts that can have great impact on the affected victims' environment generating situations in which virtual attacks escalate into fatal consequences in real life.

- The classic Functionality of this Application focuses on fake and harmful troll on Twitter and on any other Application.

❑SCOPE:

Troll Activity Detection:

Research works have been conducted on troll detection. Most of the focuses on the troll accounts that create unusual comments on any post. We have used Machine Learning Algorithms and Natural Language Processing to precisely identify the trolls. And since the use of social media is increasing everyday, the need for an efficient troll detection system is also increasing. So this troll detection system can be very useful even with the future perspective.

❑TECHNOLOGY:

In developing the Job portal website, we have used NLP, ML and Python.

Natural language processing

Natural language processing is a subfield of linguistics, computer science, and artificial intelligence concerned with the interactions between computers and human language, in particular how to program computers to process and analyze large amounts of natural language data.

The goal is a computer capable of "understanding" the contents of documents, including the contextual nuances of the language within them. The technology can then accurately extract information and insights contained in the documents as well as categorize and organize the documents themselves.

Machine learning:

Machine learning is the study of computer algorithms that can improve automatically through experience and by the use of data. It is seen as a part of artificial intelligence.

Machine learning algorithms build a model based on sample data, known as training data, in order to make predictions or decisions without being explicitly programmed to do so. Machine learning algorithms are used in a wide variety of applications, such as in medicine, email filtering, speech recognition, and computer vision, where it is difficult or unfeasible to develop conventional algorithms to perform the needed tasks.

A subset of machine learning is closely related to computational statistics, which focuses on making predictions using computers; but not all machine learning is statistical learning. The study of mathematical optimization delivers methods, theory and application domains to the field of machine learning. Data mining is a related field of study, focusing on exploratory data analysis through unsupervised learning. Some implementations of machine learning use data and neural networks in a way that mimics the working of a biological brain. In its application across business problems, machine learning is also referred to as predictive analytics.

Python:

Python is an interpreted high-level general-purpose programming language. Its design philosophy emphasizes code readability with its use of significant indentation. Its language constructs as well as its object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects

API:

We have use Python Flask framework for this project.

Flask is a micro web framework written in Python. It is classified as a microframework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions. However, Flask supports extensions that can add application features as if they were implemented in Flask itself. Extensions exist for object-relational mappers, form validation, upload handling, various open authentication technologies and several common framework related tools.

❑ALGORITHMS:

Support Vector Machines

Support Vector Machines (or SVM) is one of the most used algorithms in Troll classification today due to the nature of the data, such as the fact that most of the works in the area have binary labels (Troll/Non Troll). This algorithm works by finding a hyperplane in a higher dimensional space, which the features that describe the data points are mapped to. In the algorithm's domain, the data points belong to one of two categories. An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a margin that is as wide as possible. One example could be the division of a group of accounts into which ones were trolls and which were not. This algorithm was designed for robustness to overfitting through the margin reduction, and this robustness comes from projecting the data to higher dimensionalities, having also the advantage of being memory-efficient. It does have disadvantages, however, such as its unsuitability for large datasets due to the resources required to store each result requiring memory that scales quadratically with the number of data points. Another disadvantage is the bad handling of noisy.

Logistic Regression

Logistic regression is a statistical algorithm used to predict and/or classify effects and interactions with binary response data. It is one of the simplest tools for prediction, generating a discrete outcome from a set of variables, which may be continuous, discrete, dichotomous, or a mix of any of these. Logistic regression gives the conditional probability that an event will occur given the values of the independent variables as well as providing information on the causality of the different values. This approach is popular among researchers in the analytical fields either as a solution or a baseline - a minimum or starting point to be used for comparisons with other, more complex algorithms.

This data mining method has the advantage that it is one of the simplest methods for binary classification, which is observable when analyzing its computational efficiency. It has also no necessity for the input features to be scaled or altered, as well as the fact that the results outputted are easy to understand. It does have disadvantages, however: it performs better by removing similar and unrelated features (requires data reduction efforts), and is unable to solve non-linear problems. More complex algorithms may produce better results for the same problem.

Decision Tree

Decision tree is a classification procedure that recursively partitions a data set into smaller subdivisions on the basis of a set of tests defined at each branch (or node) in the tree. The tree is composed of a root node (formed from all of the data), a set of internal nodes (branches), and a set of terminal nodes (leaves). Each branch in a decision tree has only one parent branch and two or more descendant ones, while leaves have none. In binary classification, the tree's leaves are only of two kinds.

Decision trees are proven to be efficient in terms of computation, as they do not require normalization of data (scaling variables to have values between 0 and 1) and they have inherent mechanisms which makes them resistant to missing values. These characteristics make decision trees one of the most popular algorithms, whether as a single solution by themselves, or by grouping several together, constituting a Random Forest. They are, however, especially sensitive to the training data as their error has a high variance value. This can lead to minor changes in the data originating highly different models, which makes this algorithm sensitive to overfitting.

K-Nearest Neighbors

K-nearest neighbors (or KNN) is a supervised learning algorithm which estimates how likely a data point is to be a member of the existing classes, depending on what class the data points nearest to it are in. The algorithm determines the membership of an observation, based on the states of the points that are near it. The range (k), represents a sample of the data on which the algorithm will base its decision when assigning a new observation to a class: If the majority of the points are in group A, then it is likely that the data point in question will be A rather than B, and vice versa.

Applying this algorithm to troll detection, if a membership of an account was unknown, but his behavior was, his nearest neighbors would be the accounts that displayed similar behavior. If the K accounts in the group (his neighbors) closest to him were trolls, the KNN algorithm states that it is most probable that the account is a troll, and vice versa. KNN has relevant advantages, such as adapting as new training data is collected, allowing the algorithm to respond quickly to changes in the input during real-time use. Another advantage is the fact that it does not build a model and therefore needs no training step. It has significant disadvantages, however. A common disadvantage is that for large datasets, it tends to be slow and computationally heavy.

Random Forest

The Random Forest algorithm is an ensemble algorithm— a set of individually trained classifiers, in this case, of decision trees, whose predictions are combined when classifying novel instances. They operate by constructing a series of decision trees (Figure 2.10) when training the model. Once the creation of trees is finished, the algorithm outputs the class that is the most frequent (mode) verdict among the individual decision trees.

This algorithm has mechanisms that allow it to reduce the risk of overfitting that comes from the utilization of decision trees. Unlike the latter, they tend to be harder to interpret. Despite that inconvenience, studies exist that explicitly claim that random forests are the most likely to give accurate results for data mining problems and are frequent in many classification studies.

Algorithm Evaluation

To check whether the algorithm is producing interesting and overall correct results (in the case of this subject, if it is correctly identifying trolls), we have checked their overall accuracy score. We found out that **Random Forest** Algorithm gives the best results and works good with our dataset.

❑ NATURAL LANGUAGE PROCESSING TECHNIQUES:

Tokenization

Tokenization is described by Trim as "the process of segmenting running text into words and sentences". Text is a linear sequence of symbols (characters, words or phrases), so before performing an NLP task, a pre-processing must be done. The requires a tokenization process – to be segmented into linguistic units such as words, punctuation, numbers, alphanumeric characters, etc. The identification of units that do not need to be further decomposed for subsequent processing is of great relevance. Errors made at this stage are likely to induce more errors at later stages of text processing (e.g.: lower accuracy), so tokenization must be done carefully.

Stopword Removal

Stop-words are usually irrelevant to the text analysis, as they carry no information themselves (e.g. conjunctions). After removing stop-words in a document, the remaining text contains only the relevant words. In the normal data preparation process, stop word removal is a common step, although for more recent representations it is not required.

Stemming

Another phase in the data preparation process is Stemming, described by Pande et al as "the process for reducing inflected (or sometimes derived) words to their stem, base or root form". This process is done as a way to approach words of the same family or of similar meanings mainly by truncating prefixes or suffixes (for example, the words Doggy, Doglike and Dogs all get stemmed to the word Dog). This process may reduce redundancy in the corpus, which can result in performance and computational advantages. Its performance varies in the language and the type of words. This process, along with stopword removal, has had fewer usage with the appearance of newer approaches such as the ones based on subword tokenization, character embeddings or language models. For approaches that use the bag-of-words representation, however, stemming is an important step, and is relevant for this work, as several troll detection works use the BoW representation.

Lemmatization

When performing stemming, lemmatization usually accompanies it. It is the algorithmic process of determining the lemma (the base form under which the word is entered in a dictionary). Unlike stemming, lemmatization depends on correctly identifying the intended part of speech and meaning of a word in a sentence, as well as within the larger context surrounding that sentence, such as neighboring sentences or even an entire document. As a result, developing efficient lemmatization algorithms is an open area of research. If done correctly, it also provides a productive way to generate generic keywords for searching documents.

Text Representations

Text representation in NLP is seen as a broad term, referred by Chen et al. as "methods, systems, and apparatus, including computer programs encoded on computer storage media, for computing numeric representations of words". In practice, text representation is a fundamental problem in text mining, that aims at numerically representing unstructured text documents to make them mathematically computable.

Bag of Words

Bag of Words (BoW) is one of the most popular models for text representation. In this model, a text (such as a sentence or a document) is represented as the bag (multiset) of its words, disregarding grammar and even word order but keeping multiplicity. This approach uses the tokenized words for each observation, and calculates the frequency of each token. In the BoW model, each sentence is treated distinctively. After that, a processed list of words is made from the four documents: The next step is to create vectors for each document, where the frequency of the 10 distinct words is counted. With the vectors created, feature engineering is now possible. This representation has well known flaws, such as sparsity and sensitivity to overly common words, making a word with low significance have a high value of relevance attributed to it, simply because it appears commonly across a document or documents. One other limitation is that it doesn’t respect the semantics of a word, leading to the words "car" and "automobile" to be treated as two completely separate terms.

Word Embeddings

An embedding is a mapping of a discrete –categorical – variable to a vector of continuous numbers. In the context of text, this means that words, sentences and whole documents are represented as mathematical values. Word embeddings rose as an answer to a recurring problem in text - words with similar meanings were treated by the models as unique symbols. In this new approach to represent document vocabulary, words were represented as dense vectors, derived using various training methods inspired from neural-network language modeling. This approach proved useful in several studies of different areas.

❑TROLL CLASSIFICATION:

Data Source

The source of the data varies across the studies, with the target platform varying with the problem itself: studies whose objective is to find Trolls focus on local news forums, or in the case of social networks, the researchers tend to follow accounts that are linked to or write primarily about provoking and offending text, for example. In other types of troll detection studies, like the more population specific efforts, the platform is more specific to the country, with the large majority of its user-base limited to that location.

Data Collection

The collection process of the data is a fundamental step in the data mining process, since a faulty collection may compromise the learning performed by the algorithms, or may miss important attributes in the domain. The data collection dimension has three key concepts subject to it: Technology, sampling and annotation process. When there is a need for a collection technology method, the usage of APIs and crawlers is the most frequent. Here we have used Kaggle's dataset troll detection dataset.

Dataset

The constitution of the dataset and its classes varies across the various approaches. The majority of cases use datasets on which half of all messages/posts/comments come from trolls, and the other half from normal users. This technique is the most common in classification works, since the majority of algorithms used for binary classification (like random forest) display better results when using a balanced dataset, and if it is too unbalanced, these can have good accuracy on the majority class but poor accuracy on the minority class(es). This happens due to the influence that the larger class has on traditional training criteria.

Label

A label, in machine learning, can be described as the output of the model, in this case, the result of the classification. In troll classifiers, the labels are predominantly binary: Troll or Not Troll. We have created an API which gives the output Troll or Not Troll upon giving the input text. It can be integrated with social media platform's comments to determine trolls.

Text Features

Text features play a fundamental role in the majority of studies gathered for troll detection. As opposed to non-textual ones, text features are all types of features that are extracted from text. Of the features gathered, they can be divided into three types of subfeatures: Superficial Text Features, Learned Semantic Features and Engineered Semantic Features.

Superficial Text Features:

This type of features comes from the extraction of knowledge from the text, but the type of information gathered is mostly about the attributes of the text, such as counters of certain types of words and ratios of those values. This type of features is the most prevalent type across the studies due to its range of possibilities. Such features include: Word count; Swear word count; Bad word to normal word ratio.

Learned Semantic Features:

A type of features that are obtained when using NLP, these are extracted when applying functions specific for text categorization, such as creating embeddings from documents or obtaining sentiment scores. In these cases, new information is learned about the text and/or the relationships from the words in it.

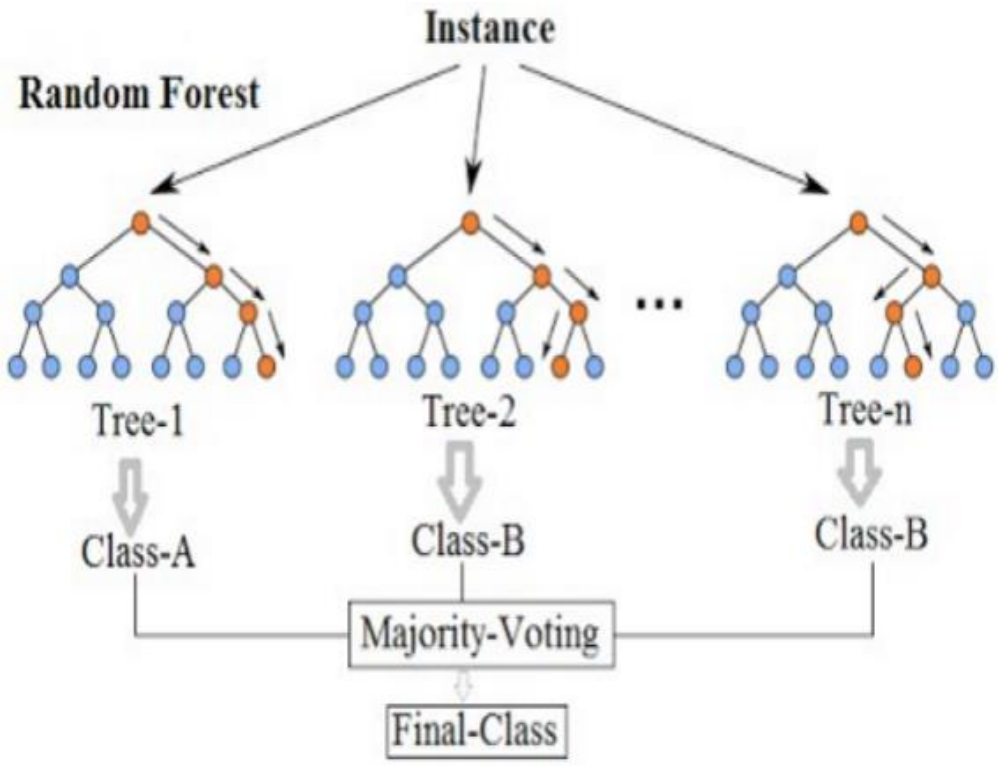
❑WORKING:

We have used Random Forest Machine Learning algorithm to classify the trolls and non-trolls. We have also used NLP features such as lemmatization and Bag of Words to precisely identify the words which can be trolls. In addition, we have created an API which can be integrated with social media sites to detect the trolls in comment section. This API for now just detect the trolls by giving the comment as input manually.

Machine Learning Algorithms Used

Of all the studies found, the vast majority experiments with several algorithms, and chooses the one best suited for their specific problem, or the one that displays the best results. Since troll detection is viewed generally as a binary classification problem, algorithms suited best for this kind of task tend to be favored. Aside from the algorithms detailed previously, the solutions using Random Forest algorithm, gives the best classification result.

Random Forest



Exploratory Data Analysis

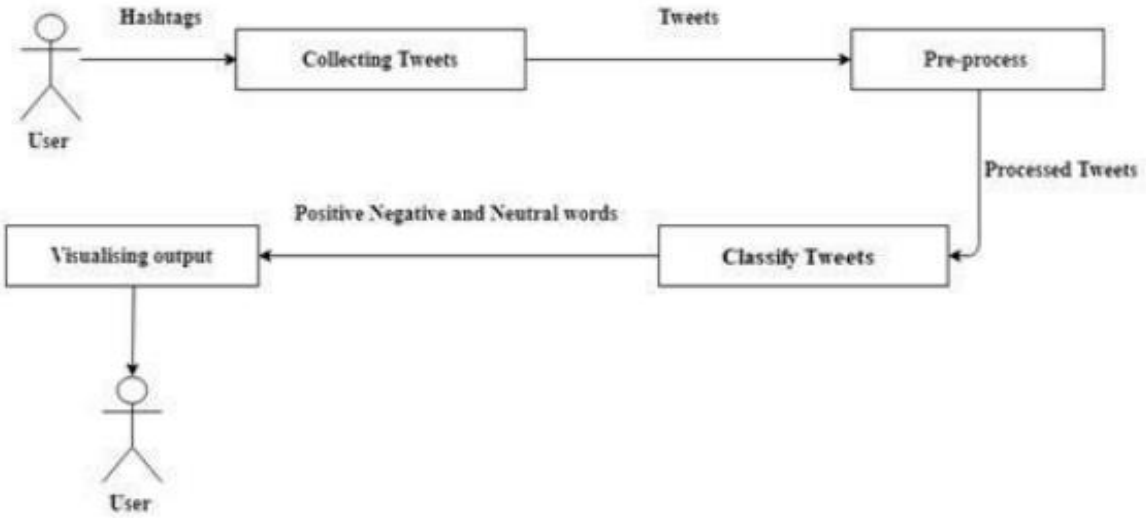
This section refers another experiment performed on the data, where simple statistical and visualization methods were applied, to better understand the data and the problem. An assessment of the feature correlations of the various groups was also performed.

Statistical Analysis of Features

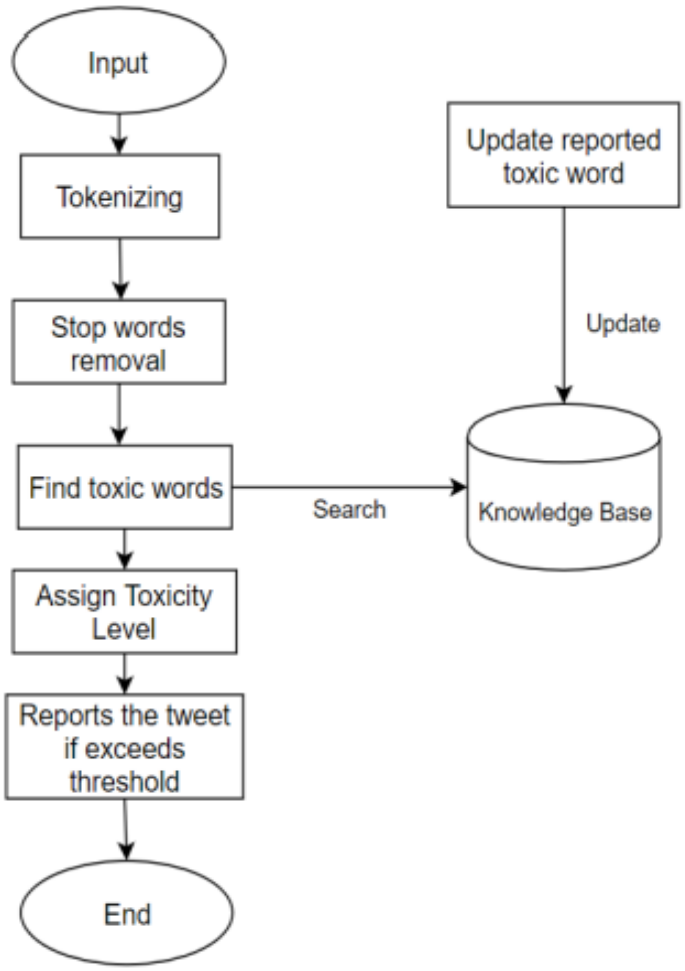
The statistical analysis was done by splitting the created feature sets, creating sets consisting of just trolls and just non trolls. From the analysis of trolls that write in English contain in their speech more swearwords, words in all caps and stop-words.

❑DIAGRAMS:

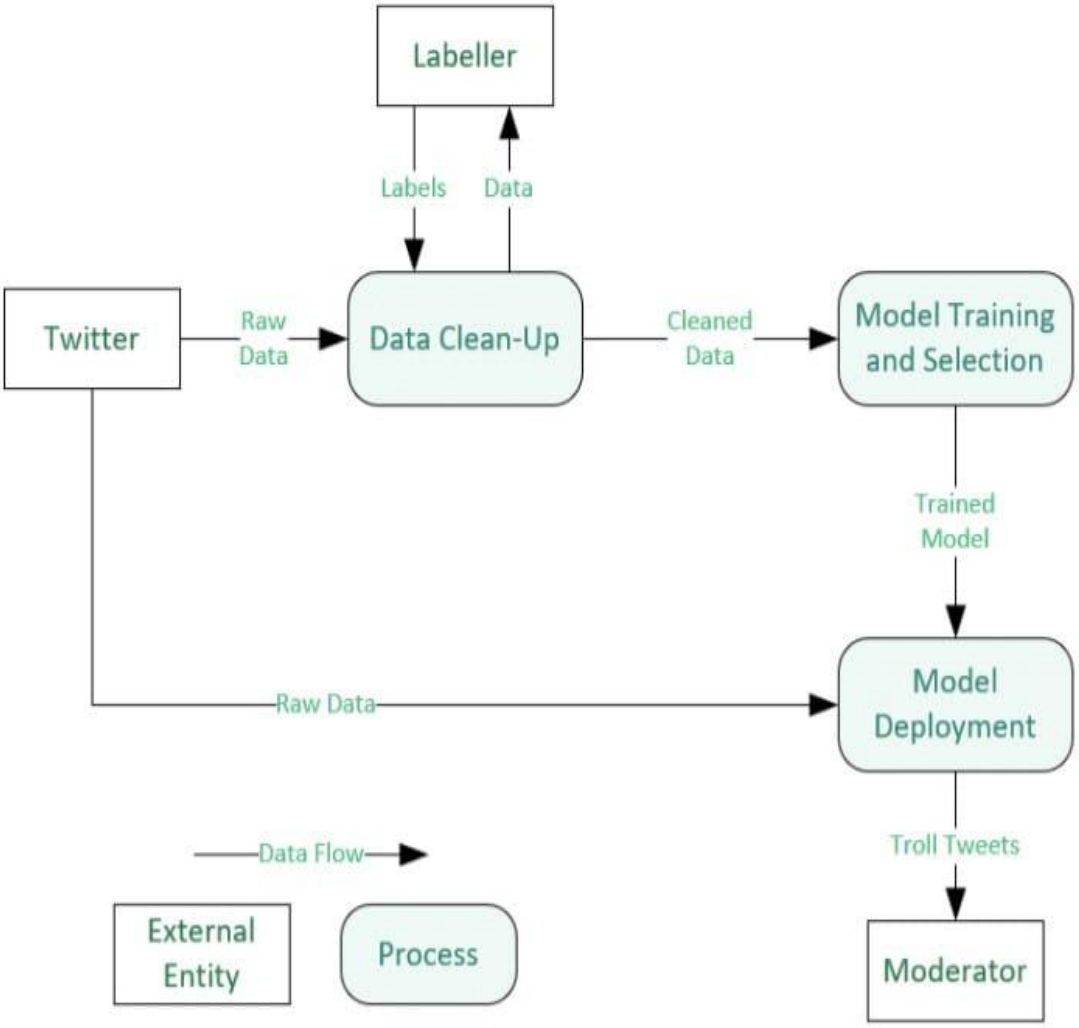
Framework of sentiment analysis



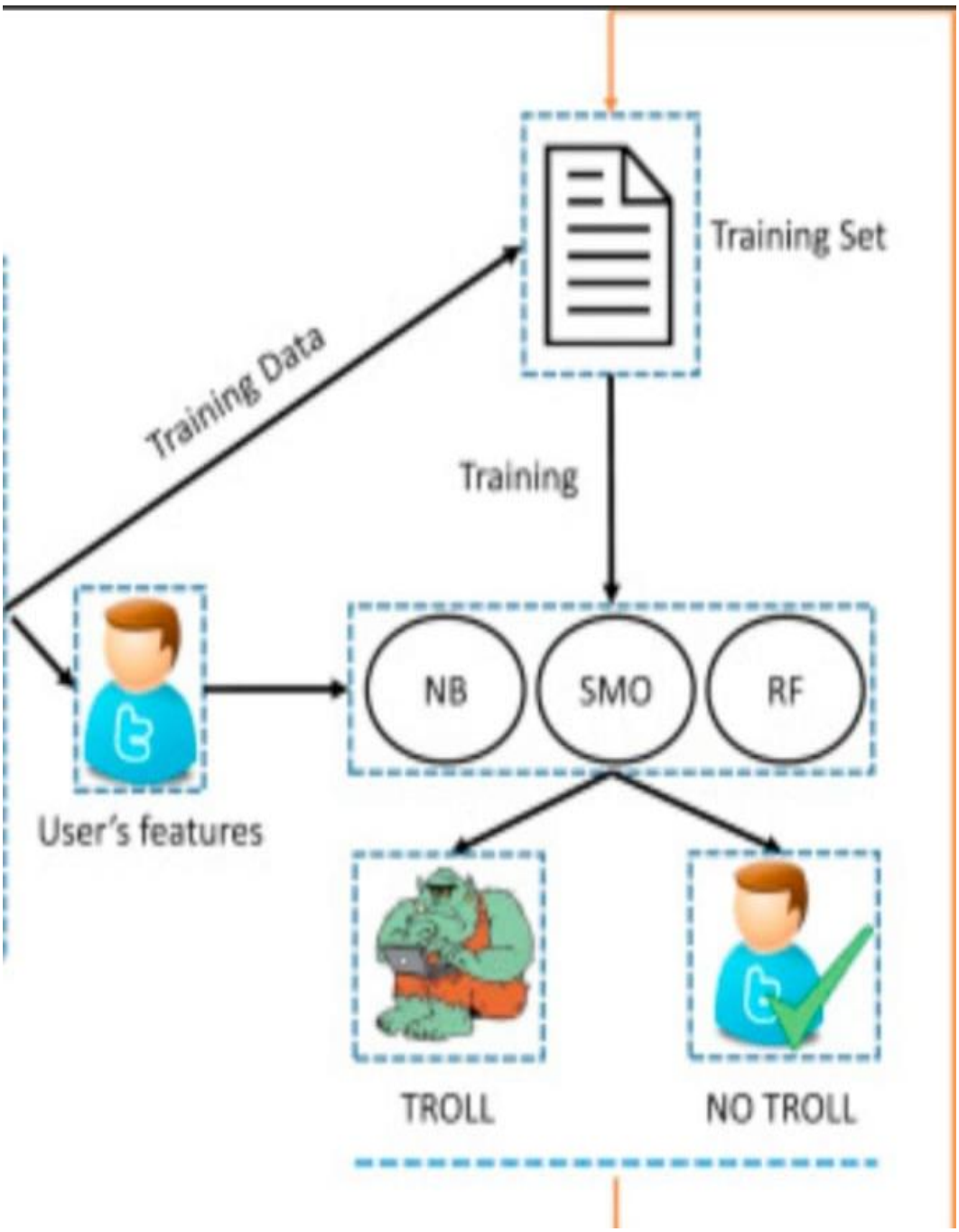
Working Flow Diagram



Activity Diagram



Work Diagram



❑RESULT

In this section, the results of a validation are analyzed and its pertinency for this work is discussed. The validation of trolls is considered as the most important step and different algorithm's accuracies are compared.

Experiment Results and Analysis

This section details the result of several algorithms performed during the duration of the project, compares them and discusses their results. Below a comparison on the result of all the models is shown.

Performance Estimation Methodology

To measure and evaluate the results of the classification experiments, different metrics are calculated and analyzed. It is important to measure the models' performance on a multitude of metrics, as well as evaluating which are the best metrics both for evaluation and for comparison with other troll detection approaches.

All models result

Here the best obtained results for all the different algorithms are detailed. All results were obtained and accuracy score is presented along with the results.

Random Forest

```
[14]: from sklearn.ensemble import RandomForestClassifier

      #Create a Gaussian Classifier
      clf=RandomForestClassifier(n_estimators=100)

      #Train the model using the training sets y_pred=clf.predict(X_test)
      clf.fit(X_train,y_train)

      y_pred=clf.predict(X_test)

[15]: from sklearn.metrics import accuracy_score
      from sklearn import metrics

[16]: print('accuracy for random forest: ',accuracy_score(y_test, y_pred))
      print(metrics.classification_report(y_test, y_pred))

      accuracy for random forest:  0.9485128717820545
                   precision    recall  f1-score   support

                0       0.97      0.94      0.96      2436
                1       0.91      0.96      0.94      1565

         accuracy                           0.95      4001
        macro avg       0.94      0.95      0.95      4001
     weighted avg       0.95      0.95      0.95      4001
```

Logistic Regression

[17]:
```
from sklearn.linear_model import LogisticRegression
LR = LogisticRegression(max_iter=10000)
LR.fit(X_train,y_train)
y_pred_LR = LR.predict(X_test)
print("Logistic Regression")
print("Accuracy score =", accuracy_score(y_test, y_pred_LR))
print(metrics.classification_report(y_test, y_pred_LR))
```

```
Logistic Regression
Accuracy score = 0.5968507873031742
              precision    recall  f1-score   support

           0       0.61      0.92      0.73      2436
           1       0.43      0.10      0.16      1565

    accuracy                           0.60      4001
   macro avg       0.52      0.51      0.45      4001
weighted avg       0.54      0.60      0.51      4001
```

KNN

[18]:
```
from sklearn.neighbors import KNeighborsClassifier
neigh = KNeighborsClassifier(n_neighbors = 5)
neigh.fit(X_train,y_train)
y_pred_KNN = neigh.predict(X_test)
print("KNN")
print("Accuracy score =", accuracy_score(y_test, y_pred_KNN))
print(metrics.classification_report(y_test, y_pred_KNN ))
```

```
KNN
Accuracy score = 0.8120469882529368
              precision    recall  f1-score   support

           0       0.86      0.82      0.84      2436
           1       0.74      0.80      0.77      1565

    accuracy                           0.81      4001
   macro avg       0.80      0.81      0.81      4001
weighted avg       0.82      0.81      0.81      4001
```

Naïve Bayes

[19]:
```
from sklearn.naive_bayes import GaussianNB
naive = GaussianNB()
naive.fit(X_train,y_train)
y_pred_naive = naive.predict(X_test)
print("Naive Bayes")
print("Accuracy score =", accuracy_score(y_test, y_pred_naive))
print(metrics.classification_report(y_test, y_pred_naive ))
```

```
Naive Bayes
Accuracy score = 0.3979005248687828
              precision    recall  f1-score   support

           0       0.91      0.01      0.02      2436
           1       0.39      1.00      0.56      1565

    accuracy                           0.40      4001
   macro avg       0.65      0.51      0.29      4001
weighted avg       0.71      0.40      0.24      4001
```

Boosting

[20]:
```
from sklearn.ensemble import GradientBoostingClassifier
gradient = GradientBoostingClassifier(n_estimators=100,max_depth=None,min_samples_split=2, random_state=0)
gradient.fit(X_train,y_train)
y_pred_gradient = gradient.predict(X_test)
print("Gradient Boosting")
print("Accuracy score =", accuracy_score(y_test, y_pred_gradient))
print(metrics.classification_report(y_test, y_pred_gradient ))
```

```
Gradient Boosting
Accuracy score = 0.9425143714071482
              precision    recall  f1-score   support

           0       0.98      0.92      0.95      2436
           1       0.89      0.97      0.93      1565

    accuracy                           0.94      4001
   macro avg       0.94      0.95      0.94      4001
weighted avg       0.95      0.94      0.94      4001
```

Decision Tree

```
[21]: from sklearn.tree import DecisionTreeClassifier
decision = DecisionTreeClassifier()
decision.fit(X_train,y_train)
y_pred_decision = decision.predict(X_test)
print("Decision Tree")
print("Accuracy score =", accuracy_score(y_test, y_pred_decision))
print(metrics.classification_report(y_test, y_pred_decision ))
```

```
Decision Tree
Accuracy score = 0.9370157460634841
              precision    recall  f1-score   support

           0       0.98      0.92      0.95      2436
           1       0.88      0.97      0.92      1565

    accuracy                           0.94      4001
   macro avg       0.93      0.94      0.93      4001
weighted avg       0.94      0.94      0.94      4001
```

So here we can see from the results that Random Forest algorithm gives the best results.

❑CONCLUSION & DISCUSSION:

- Interacting with trolls has become an almost unavoidable parts of being online.

- Staying on the offense is important, along with knowing how to a troll's comments and criticism.

- Trolling is the new generation cybercrime and trolls are the new generation of criminals on the internet who derive sadistic pleasure in spreading abuse and hate.

- We know to handle them but if it's serious we shouldn't ignore it and take strict action against it.

Limitations:

- This Troll Detection System is only for English Language so it cannot detect trolls of any other language.
- Because of the limited data available, sometime our model cannot predict the troll with different grammatical structure

❑FUTURE ENHANCEMENT:

- We can use Neural Network and Deep Learning Concepts to further train the model and detect the trolls more accurately.
- We can use Reinforced Learning approach so that our model can learn from itself and it can even detect the trolls which were not present in the dataset at the time of training