# **Natural Language Processing**

# **Detecting Profanity and Blasphemy from Movie SRT files**

**Team Members:**

Manideep Chakilam -11509362

Sreekar Veeravelly - 11452217

Raghupathi Thorlikonda - 11553766

Pavan Kalyan Komaravelli -11598258

**Introduction:**

**Problem Statement:**

To Train an NLP/NER model to detect Profanity and Blasphemy from movie dialogue SRT files and detoxify the contents in the SRT files.

**Abstract:**

Toxic comments are one of today's most pressing concerns for social media platforms. It is critical to identify poisonous comments and the persons who submit them in order to offer a positive experience for content providers and their audiences. However, because there are various representations of the same text in low-resource Indic languages, determining toxicity is difficult.

Furthermore, social media posts/comments or movie dialogues do not follow a certain format, syntax, or sentence structure, making the process of detecting abuse much more difficult for multilingual social media platforms.

In our project we have trained regression models as to get the intensity of toxicity in the comment. We have used models like SGD, Decision tree with vectorization techniques like BOW and TF-IDF. And also used some Transformer based models to detect and detoxify the dialogues.

Project aims to extract and detect the profane and blasphemy words or dialogues from the movie subtitle files (srt files). Here we have used the combination of NLP packages and models to train the final model. Various cloud storage and technologies were used to finalize the project requirements.

**Goals and Objectives:**

**Motivation:**

As most of the movie ratings are dependent on the language used in the movie. We are going to train an NLP model based on the Machine Learning and NER technique to detect the Profanity and Blasphemy from line-by-line analysis of the SRT files. We will also apply Detoxification of the sentences using transformer based deep learning models. So that benign contents can be consumed by the audience.

**Significance:**

As we know now a days due to social media impact each controversy are easily spread. And due to human validation, it might be possible that we can miss some scenes which may have created some controversy.

And most of the movie ratings are dependent on the language used in the movie. Some countries have strict ban on bad words or blasphemy words based on Gods.

To avoid these things, we are going to train an NLP model based on the NER technique to detect the Profanity and Blasphemy from line-by-line analysis of the SRT files and detoxify the dialogues in the sentence like bully or racial words.

**Objective:**

Movies have different Rating systems based which differs country by country. Like if a movie is R rated due to language in Gulf countries, but it might not be the same case in Western countries. These processes are manually checked by humans.

Predict the toxicity of a comment made by the user. (0 -> not toxic, 1 -> highest toxicity level)

We will predict the toxicity level (target attribute). The values range from 0 to 1 inclusive. This is a regression problem. It can also be treated as a classification problem if we take every value below 0.5 to be non-toxic and above it to be toxic, we will then get a binary classification problem.

We have converted our outcome to Classification by adding threshold to the probabilities

Language model research are going on to detect different language aspects which may be context aware or may not be context aware.

Many systems are available which are working based the exact match scenarios and are not context aware.

We are working on the model which is context aware and can connect the other word embeddings to get the sentence overview with the sentiment.

Transformer Models will later be used to get the detoxification of the sentence.

**Features:**

* In this project we are detecting the profanity words from the sentences and then by paragraphs.
* We will collect word file (movie sub-titles) and detect the words and highlights the words and provide the probability it to be a positive or negative meaning in the given sentence like finding the opinion of the text.
* We are trying to change the detected profanities to other words like example Fuck to F\*\*\*.
* We are trying to update the bad-words to positive words without changing the meaning of the sentence**.**

**Deliverables:**

* The Program will accept the Subtitles files and Output the CSV file containing the Profane and Blasphemy words from the movie dialogues And also add a text column with detoxicated sentences.

**Inference:**

Input for Inference: Path of SRT file

Output: CSV file with columns such as *sentence, Profanity or Blasphemy detection, start time and end time of dialogue, and Detoxication sentence. etc if any.*

**Background**

**Related work:**

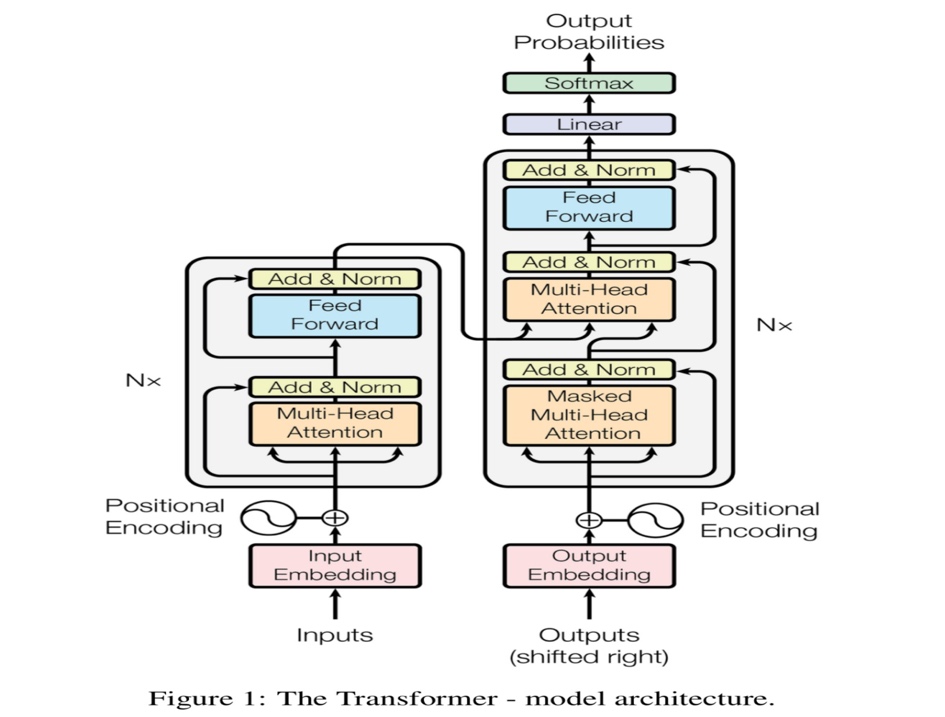
An ensemble model with various techniques was presented **by Van Aken et al**. to solve the difficulties in toxic comment classification. This ensemble model works best in classes with low instance counts and substantial data variance. Deep neural network architectures were compared for overlapping harmful sentiment data by Saeed et al. The authors demonstrate that in the task of overlapping multilabel harmful sentiment categorization, the Bi-GRU model outperforms.

Google's viewpoint API for toxicity identification was the target of a suggested attack by **Hossein et al.** The authors provided specific examples to demonstrate how a highly toxic phrase gives benign phrases high toxic scores while giving upsetting abusive statements low toxic scores.

To address the issue of class imbalance, **Ibrahim et al.** offered three data augmentation techniques: unique words augmentation, random mask, and synonyms substitution. Additionally, in order to identify toxicity in user-generated content, the authors developed an ensemble model (CNN, Bidirectional LSTM, and GRU).

**Model:**

**Architecture Model:**

****

On the left is the Encoder, and on the right is the Decoder. Encoder and Decoder are both made up of modules that can be stacked on top of each other multiple times, as indicated by Nx in the diagram. We can see that the modules are primarily made up of Multi-Head Attention and Feed Forward layers. Because we cannot use strings directly, the inputs and outputs (target sentences) are first embedded in an n-dimensional space.

The positional encryption process of the various words is a small, but significant, component of the model. As  a sequence rely on the sequence in which its elements are introduced into a model, we have to find a way to assign each word or component in our pattern a relative position. Unfortunately, we lack recurrent networks which can do this. Each word's embedded representation (an n-dimensional vector) is expanded by these positions.

**Workflow Diagram:**

**Diagram

Description automatically generated**

As the above diagram describes we our initial stage is to train the model by using the training dataset that we are going to use and then we perform the upcoming stages as above.

**Dataset:**

**A yellow sign with black text

Description automatically generated with low confidence**

We have collected SRT data from <https://www.imdb.com/> of more than 100’s of movies and each movie having about 1000+ lines.

Main aim here is to label the raw data for the training.

**Data Characteristics:**

There is currently no data available as .srt files to detect profanity and blasphemy. Therefore, we will be manually collecting the data and labelling it.

We will be using 100 .srt files of which 50 of Profanity and 50 of Blasphemy for the training. Each file consists of an average 1500 lines of dialogues.

Therefore, total we will be labelling and training total around 150000 lines of data.

Each SRT has following columns. Number of lines, start time, end time, and dialogue.

**Other Data Source**: [**https://www.kaggle.com/c/jigsaw-unintended-bias-in-toxicity-classification/data**](https://www.kaggle.com/c/jigsaw-unintended-bias-in-toxicity-classification/data)

It might be tough to talk about topics that are important to you. Because of the possibility of online abuse and harassment, many individuals have stopped expressing themselves and have given up searching out various viewpoints. Many communities have limited or altogether shut off user comments as platforms fail to successfully support dialogues.

We have taken large number of Wikipedia comments which have been labelled by human ratters for toxic behavior. The types of toxicity are:

* toxic
* severe\_toxic
* obscene
* threat
* insult
* identity\_hate

**Columns in train data:**

* + **Comment\_text**: This is the data in string format which we have to use to find the toxicity.
  + **target**: Target values which are to be predicted (has values between 0 and 1)
  + Data also has additional toxicity subtype attributes:
    - severe\_toxicity
    - obscene
    - threat
    - insult
    - identity\_attack
    - sexual\_explicit

**Comment\_text data also has identity attributes carved out from it, some of which are:**

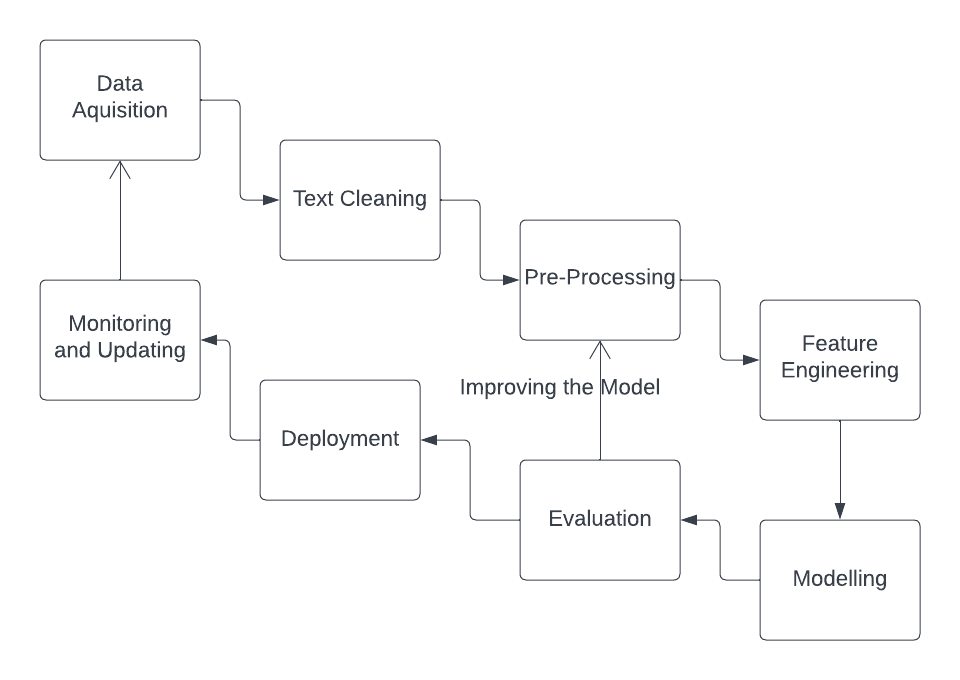
* + male
  + female
  + homosexual\_gay\_or\_lesbian
  + christian
  + jewish
  + muslim
  + black
  + white
  + asian
  + latino
  + psychiatric\_or\_mental\_illness

**Glimpse of the Data:**

A picture containing website

Description automatically generated

**Detail Design of features:**

****

**Analysis Of Data:**

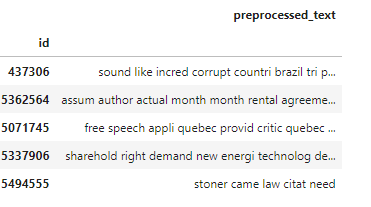
For Data Analysis and Processing have performed various analysis:

* Target value Distributions
* Distribution of additional toxicity features
* Percentage of toxicity nature
* Word Cloud
* Stemming
* Lemmatization
* Stop word removal
* BoW
* TFIDF

**Data Preprocessing**: As we are working on Text data So there was not much of preprocessing apart from cleaning text and basic text preparation.

* Removing of special characters and punctuations.
* Converting everything to lower case.
* **Stemming**: For stemming we have used Snowball Algorithm. Snowball is a small string processing programming language designed for creating stemming algorithms for use in information retrieval.

**Glimpse of Preprocessed data:**



We can see that the data was Highly skewed which affected the outcome as well.

Chart, histogram

Description automatically generated

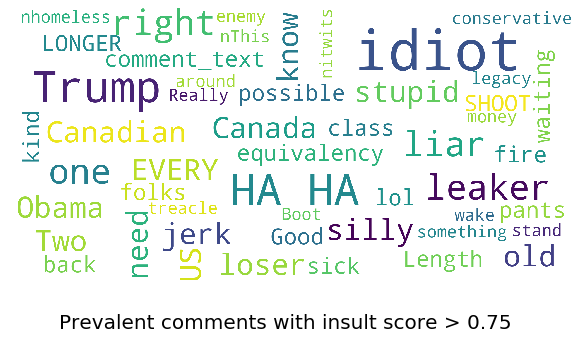
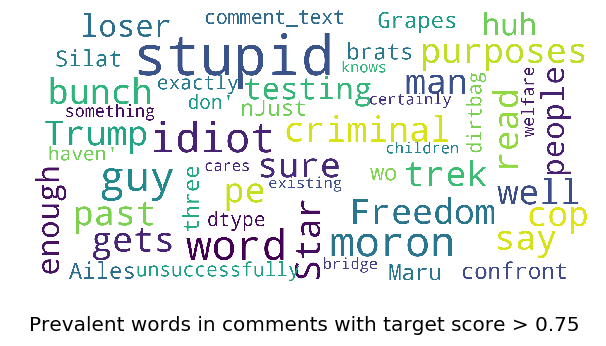
In train dataset only 8% of the data was toxic.

Out of that 8%, 81% of the toxic comments made are insults, 8.37% are identity attacks, 7.20% are obscene, 3.35% are threat.

Chart, bar chart

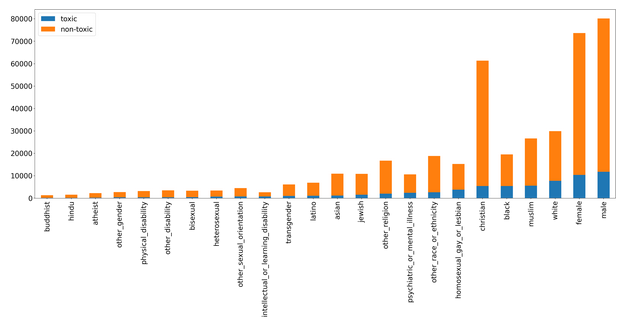
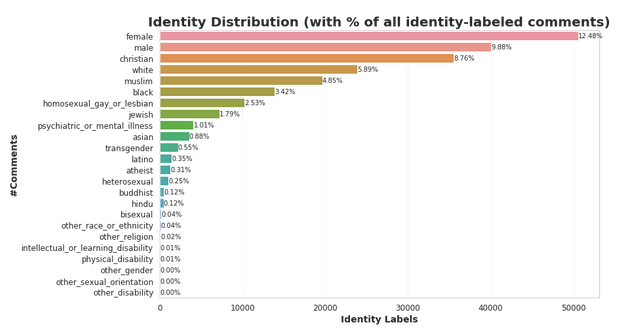
Description automatically generated

Some extracted words used for training

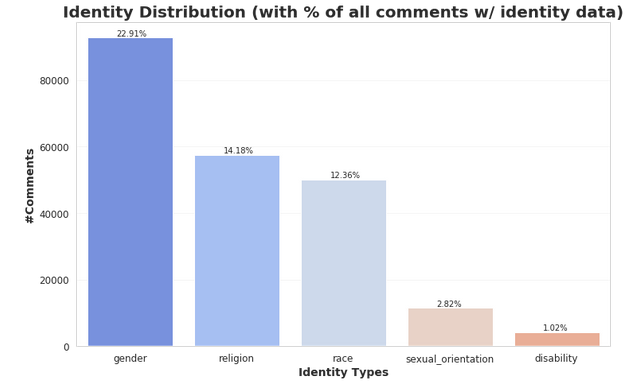
**** ****

**Graph models with explaination:**

**Identity words distribution Use of Words in Toxic and Non-Toxic categories**

****

**Here we can see the summing up of Identity words.**

****

**These are the important takeaways from this EDA that could prove useful for preprocessing and model construction:**

* We're working with an unbalanced data collection (92 percent vs 8 percent).
* The number of annotators and the toxicity value have a correlation.
* Insult is the most common harmful subtype, and it dominates all others. Even more fascinating, it has a (very) strong association with target value, raising the question of whether the toxic definition in this dataset is "insulting."
* Due to the lack of label availability and association, identity attributes are ineffective in predicting target value.
* Some comments have a dubious goal value, implying that cleaning the (noisy) dataset before to training the model is critical.

When we look at several examples of false positives, we can see that some of the comments have a debatable target value. We are not sure why comments like "Effin idiot" and "Crooked Jerk!" aren't considered toxic. "You're a stone-cold MO-Ron," for example, with a target value of 0. These could indicate that we need to take extra precautions when cleaning the dataset.

We can also see that comments with target values that are dangerously close to the toxic threshold (0.5) have been rounded up to "safe" remarks. Despite the fact that "overpaid, infantile, morons" is clearly an insult, model thought it was safe.

**As real-world scenarios will be Skewed but for training purposes we can train model with less skewed or up sampled data.**

**Implementation :**

**Algorithms:**

* It would be a mix of Algorithms and Models.
* We would be training NER model using spacy and other Deep Learning frameworks like Keras, TensorFlow.
* With also some additional string sorting using Regex.
* Transformer Models are used for extracting embedding.

**Methods:**

**What is Zero-shot text classification?**

As we all are aware, text classification is a problem that falls under the umbrella of natural language processing, and the model's job is to predict the classes of the text documents. The traditional method requires us to train the model on a sizable quantity of labelled data, and it also prevents them from making predictions based on hidden data. Natural language processing has been advanced to its farthest level by combining zero-shot learning with text categorization.

Any model used in conjunction with the zero-shot text classification approach has as its primary objective the categorization of text documents without the use of a single labelled piece of data or having seen any labelled text. Transformers are where we mostly locate zero-shot classification implementations.

**Diagram

Description automatically generated**

**Vectorization**: Along with training we have applied various vectorization techniques to extract the most of information from the text data.

We have extracted the vectors Using below methods and trained SGD and Decision Tree models to test the Regression outcome.

For Vectorization we have used:

**BOW:** A bag-of-words is a portrayal of text that depicts the event of words inside a document. It includes two things: A vocabulary of known words. A proportion of the presence of known words.

**Term Frequency - Inverse Document Frequency (TFIDF):** In data recovery, tf-idf, is a mathematical measurement that is expected to reflect how significant a word is to a report in an assortment or corpus. It is frequently utilized as a weighting factor in searches of data recovery, text mining, and client displaying.

**Training Algorithms we used:**

**Decision Tree Regressor:** A Decision tree is a quite certain sort of likelihood tree that empowers you to settle on a choice regarding some sort of cycle.

A huge benefit of a Decision tree is that it powers the thought of all potential results of a choice and follows every way to an end. It makes an extensive examination of the outcomes along each branch and recognizes choice hubs that need further investigation.

**SGD Regressor:** Stochastic Gradient Descent (SGD) is a simple yet efficient optimization algorithm used to find the values of parameters/coefficients of functions that minimize a cost function. In other words, it is used for discriminative learning of linear classifiers under convex loss functions such as SVM and Logistic regression.

SGDrandomly picks one informative element from the entire informational collection at every cycle to decrease the calculations tremendously. It is additionally normal to test few informative elements rather than only one point at each progression and that is designated “mini-batch” gradient descent.

**LSTM networks** are appropriate to characterizing, handling and making expectations in light of time series information, since there can be slacks of obscure length between significant occasions in a period series.

LSTMs were created to manage the evaporating angle issue that can be experienced while preparing conventional RNNs.

**The Structure of Training pattern looks like:**

BagOfWords:

SGDRegressor

Decision Tree

TFIDF:

SGDRegressor

Decision Tree

LSTM

We have extracted the vectors using BOW and TFIDF and trained the SGD and Decision tree models using both the vectors. And at last we have also trained the LSTM model using Tensorflow-Keras. Based on the Regression value, if the value is near to 0 then less toxic and if near to 1 then more toxic comment.

After Getting the probabilities from the trained model we have added threshold to get the Classification as Toxic or Not.

Greater than 0.5 is 1.

Less than 0.5 is 0.

**Model Outcome:**

**Analysis:**

We have trained the above-mentioned model with multiple hyper parameters and iterations using cross- validation.

We have got Decreasing MSE scores:

1. **BagOfWords:**
   1. *SGDRegressor:*
      1. Hyperparameters Tuned Values: learning\_rate(alpha): 1e-05 and penalty: l2
      2. Train MSE Loss: 0.02272
      3. CV MSE Loss: 0.02412
   2. *Decision Tree:*
      1. Hyperparameters Tuned Values: max\_depth: 7 and min\_samples\_leaf: 100
      2. Train MSE Loss: 0.032
      3. CV MSE Loss: 0.03098

2.**TFIDF**:

* 1. *SGDRegressor:*
     1. Hyperparameters Tuned Values: learning\_rate(alpha): 1e-05 and penalty: l2
     2. Train MSE Loss: 0.02476
     3. CV MSE Loss: 0.02614
  2. *Decision Tree:*
     1. Hyperparameters Tuned Values: max\_depth: 7 and min\_samples\_leaf: 100
     2. Train MSE Loss: 0.0309
     3. CV MSE Loss: 0.03141

3. **LSTM**:

* 1. Train MSE Loss: 0.01564
  2. CV MSE Loss: 0.017

**After Training Again with more Epochs and changing parameters we have seen Slight but not that significant increment in the Results.**

1. **BagOfWords:**
   1. *SGDRegressor:*
      1. Hyperparameters Tuned Values: learning\_rate(alpha): 0.001 and penalty: l2
      2. Train MSE Loss: 0.02144
      3. CV MSE Loss: 0.02414
   2. *Decision Tree:*
      1. Hyperparameters Tuned Values: max\_depth: 7 and min\_samples\_leaf: 150
      2. Train MSE Loss: 0.030
      3. CV MSE Loss: 0.03025

**2.TFIDF:**

* 1. *SGDRegressor:*
     1. Hyperparameters Tuned Values: learning\_rate(alpha): 0.01 and penalty: l2
     2. Train MSE Loss: 0.02356
     3. CV MSE Loss: 0.02412
  2. *Decision Tree:*
     1. Hyperparameters Tuned Values: max\_depth: 5 and min\_samples\_leaf: 100
     2. Train MSE Loss: 0.0302
     3. CV MSE Loss: 0.0303

**3.LSTM:**

* 1. Train MSE Loss: 0.01434
  2. CV MSE Loss: 0.015

**The loss curve clearly explains that LSTM is giving the better reduction in loss.**

**Diagram, application

Description automatically generated**

**LSTM Model**

**Table

Description automatically generated**

**For ML Models we can see the Important features below:**

**Table

Description automatically generated with medium confidence**

Below we can see the output of when converted to classification.

If we add the threshold bar we can even convert it to Classification problem

**Chart, treemap chart

Description automatically generated** **Table

Description automatically generated**

**AUC and ROC:**

After converting the probabilities to the absolute 0 and 1 we have plotted ROC curve for LSTM, SGD and Decision Tree.

LSTM has highest AUC Score= 0.91

**Chart

Description automatically generated**

**Project Management:**

**Work Completed:**

* **Description:** In this project we are detecting the profanity words from the sentences and then by paragraphs. We will collect word file (movie sub-titles) and detect the words and highlights the words and provide the probability it to be a positive or negative meaning in the given sentence like finding the opinion of the text.
* **Responsibilty:** 
  + Manideep Chakilam -NER model and Collection of srt and

processing them.

* + Raghupathi Thorlikonda -Dataset and Collection of srt and

processing them.

* + Sreekar Veeravelly -Model for detecting curses and

Creation of corpus.

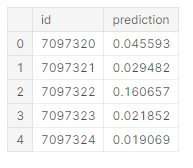
* + Pavan Kalyan Komaravelli -Model on Text Classification and NER

Training.

* **Contribution:**
  + Manideep Chakilam -25%
  + Sreekar Veeravelly -25%
  + Raghupathi Thorlikonda -25%
  + Pavan Kalyan Komaravelli -25%

**Issues/Concerns:** Collection of srt and processing them. Also we have some issues in cleaning the data. Creation of corpus because the daily used words which varies from word to word. Also, NER training which vanishes the old sentences when we update with new sentences.

**Results:**



**Here we can see that the Third comment got the high score as it looks slight toxic compared to other sentences**

7097320: Integrity means that you pay your debts. Does this apply to President Trump too?“

7097321: This is malfeasance by the Administrator and the Board. They are wasting our money!

7097322: @Rmiller101 - Spoken like a true elitist. But look out bud. The re-awakening in Europe, Brexit and now the Trump victory are just the beginning. The unwashed masses are coming for you and your ilkâ€¦.

Lets see the result of one sentence:

**"Hey you idiot"**

Chart, bar chart

Description automatically generated

Below we can observe that Model is able to detect the intensity of the sentence:

Text, table

Description automatically generated

Text

Description automatically generated

**For the Final Outcome we can see that the sentences are detoxified**

Graphical user interface, text, application

Description automatically generated

**Conclusion:**

Social media has recently evolved into a center for information sharing and enjoyment. The findings of this study can be utilized to develop systems that can detect toxicity and provide a healthier experience for users.

Finally, we have Successfully trained and worked on optimization of various models with diff approach.

LSTM Model gave the best MSE score and even after converting the scores to Classification we got good results.

We Were able to Detoxify the Dialogues using Transformer Models.

The internet has given individuals unprecedented freedom of expression.

**Uncle Ben:**

With great internet connection, comes great responsibility

**References**:

* Practical Text Classification with Large Pre-Trained Language Models by Neel Kant et al.
* Mahmud, A., Ahmed, K.Z. and Khan, M., 2008. Detecting flames and insults in text.
* Modelling Profanity and Hate Speech in Social Media with Semantic Subspaces by Vanessa Hahn, Dana Ruiter, Thomas Kleinbauer, Dietrich Klakow
* https://www.analyticsvidhya.com/blog/2019/09/demystifying-bert-groundbreaking-nlp-framework/
* <https://www.analyticsvidhya.com/blog/2020/03/6-pretrained-models-text-classification/>
* <https://towardsdatascience.com/building-a-better-profanity-detection-library-with-scikit-learn-3638b2f2c4c2>
* <https://lubna2004.medium.com/profanity-to-be-or-not-to-be-dd32d53648f7>

**GITHUB LINK:**

<https://github.com/ManideepChakilam/Detecting-Profanities>