Data Analytics

# Analysis of Toxic Comments on Social Media Platforms
# And Predict the Level of Toxicity

CAPELLI Guillaume

Nov, 2023

# Table of content

# Introduction:

## Understanding Online Toxicity

In a world where online interactions are increasingly shaping our perceptions and everyday experiences, the presence of toxic comments on social media has emerged as a critical concern. For me, this analysis is not just a scholarly pursuit but a necessary step towards fostering a safe digital environment. Toxic comments can deter meaningful discourse, perpetuate negativity, and even cause significant psychological harm. By analyzing these comments, I aim to unveil the patterns and triggers of online hostility, providing insights that can help platforms and communities curtail this pervasive issue.

**Use Case Overview:**

The digital landscape offers an unparalleled platform for free expression. Yet, the anonymity and detachment provided by screens often lead to an increase in hostile and aggressive communication. My focus is on identifying and understanding these patterns of toxicity to mitigate their impact and support the creation of more respectful online communities.

**Project Goal:**

The primary goal of my project is to dissect the complex dynamics of toxic comments across various social media platforms. The end objective is to collate these findings to inform and train a machine learning model that can detect and categorize toxicity autonomously, thereby aiding platform moderators and community managers.

**Objective:**

I have set specific objectives to guide my analysis:

.To analyze trends in comments over time and assess how user engagement with content evolves.

.To investigate the relationship between comment attributes and user interaction, such as likes and replies.

.To categorize comments into different levels of toxicity, which will serve as foundational data for training my predictive model.

# Data and data sources

**YouTube**: Utilizing Selenium, I scraped comments from select videos that were known for polarizing content. I chose YouTube because it's a widely used platform where video content often elicits strong opinions and reactions, which are reflected in the comments section. This dataset offers a window into user reactions and interactions, which is pivotal for understanding the public's sentiment on widely viewed content from Police control and a Trump discour.

**Link** : https://www.youtube.com/watch?v=kuhhT_cBtFU&t=2s
https://www.youtube.com/watch?v=v3abZ4aAGUU

**Reddit**: I harnessed the Reddit API to extract comments from specific threads that sparked debate and controversy. Reddit is a forum known for its community engagement and candid discussions, which can sometimes escalate into toxicity. By focusing on threads with divisive content, I aimed to capture a diverse range of opinions and sentiments, which is crucial for analyzing discourse patterns in an environment that fosters open discussion.

**Link** :
https://www.reddit.com/r/funny/comments/17r7lh2/was_he_impatient_or_does_he_have_a_point/?onetap_auto=true

**Kaggle**: The dataset from the "Jigsaw Toxic Comment Classification Challenge" offered a pre-labeled and structured array of user comments. I selected this particular dataset because it provides a substantial volume of labeled data, which is invaluable for training and benchmarking my machine learning models. The labels cover a range of toxic behaviors, which are essential for a nuanced analysis of online interactions.

**BigQuery**: Here, I tapped into expansive datasets to process and analyze comments from various online forums, leveraging the computational power of BigQuery to manage the data's scale. Big data systems like BigQuery allow for the processing of massive datasets that would be otherwise challenging to handle. By utilizing this resource, I could include a more comprehensive set of data in my analysis, ensuring that the models I develop are well-informed by a broad spectrum of user interactions.

**Link :** https://console.cloud.google.com/bigquery?p=bigquery-public-data&d=hacker_news&page=dataset&project=da-bootcamp-2023

# Data collection

- **YouTube**: In the "Youtube_WebScraping.ipynb" Jupyter notebook, I've employed a sophisticated technique for web scraping data from YouTube comments. The process begins with setting up the Selenium WebDriver, which is crucial for automated web browsing and interaction with web pages. This setup enables me to programmatically control a web browser, simulating user actions like clicking and scrolling.
  **Shape** : 2238 rows × 4 columns

- **Reddit**: I install PRAW, the Python Reddit API Wrapper, which simplifies the process of accessing Reddit's API. This installation also includes dependencies like update-checker and prawcore. begins by prompting for Reddit API credentials (client_id and client_secret). These credentials are essential for accessing Reddit's API and are securely entered using Python's getpass module, which hides the client_secret input for security.
  **Shape** : 2219 rows × 4 columns

- **Kaggle**: Like as mention on the data and data source section, itw as a pretty easy, but one the most important data for my project. But to get, that, itw as just a registration and a simple download.
  **Shape** : 159571 rows × 9 columns

- **BigQuery**: I made sure to have all the necessary libraries by installing google-cloud-bigquery and pandas. I imported necessary modules like os from Python's standard library and bigquery from Google Cloud. My focus here was to write and execute SQL queries to extract data from BigQuery.
  **Shape** :  3000 rows × 4 columns

# GDPR

In conducting this analysis, I am acutely aware of the importance of respecting user privacy and adhering to GDPR regulations. To ensure compliance, I've taken several steps:

1. **Data Anonymization**: Any identifiable information extracted during data collection has been anonymized. Usernames and other potential identifiers have been removed or obscured to prevent the possibility of re-identification.

2. **Data Minimization**: I've only collected the data necessary for the analysis. Superfluous details that do not contribute to the project's objectives have been excluded from the datasets.

3. **Consent and Legality**: Where applicable, I've made sure that the data collected is either publicly available or consent for its use in analysis has been obtained, aligning with the legal bases prescribed by the GDPR.

4. **Security Measures**: All datasets are stored securely, with access limited to authorized personnel only, thereby safeguarding the data against unauthorized access or breaches.

By implementing these practices, I ensure that the project not only aligns with legal frameworks but also upholds the principles of ethical data usage.

# Data Cleaning and Exploratory Data Analysis

**Data Cleaning:**

Most of the time there was no need to clean the data in the sense that there were 0 nulls and 0 duplicates, I instead renamed certain columns to harmonize the dataframes with each other. I also removed all special characters from the comments. I added a "comment_length" column to all dataframes. And change the time format of the data columns.

**YouTube after :** 2212 rows × 5 columns
**Reddit after :** 2219 rows × 5 columns
**Kaggle after :** 159571 rows × 9 columns
**Big Query after :** 3000 rows × 5 columns

**EDA:**

1. **Descriptive Statistics:** I calculated descriptive statistics for the 'like' and ‘comment_length’ columns. It revealed a wide variation in Likes, with a significant standard deviation. Generated a summary: Identified the number of unique authors. Analyzed the time range covered by the dataset, spanning multiple years. Investigated posting frequency to discern patterns over time and analyzed the length of text entries, providing insights into the verbosity and engagement in posts.

2. **Like Distribution:** Using a histogram, I visualized the distribution of comment likes, which displayed a right-skewed pattern indicating that most comments had likes close to zero.

3. **Activity Over Time:** A line graph depicting the number of comments over time showed a decreasing trend in comment activity.

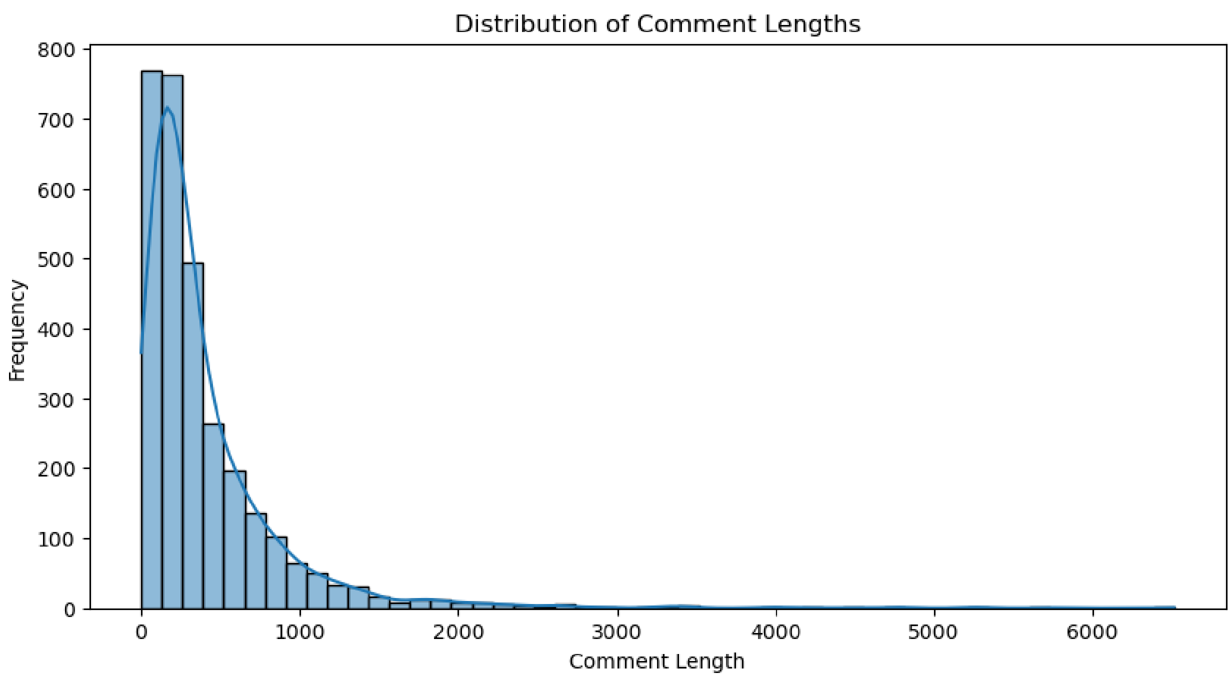
4. **Active Users Analysis:** I identified the most active users by comment count and average like, revealing which users were most engaged in the dataset.

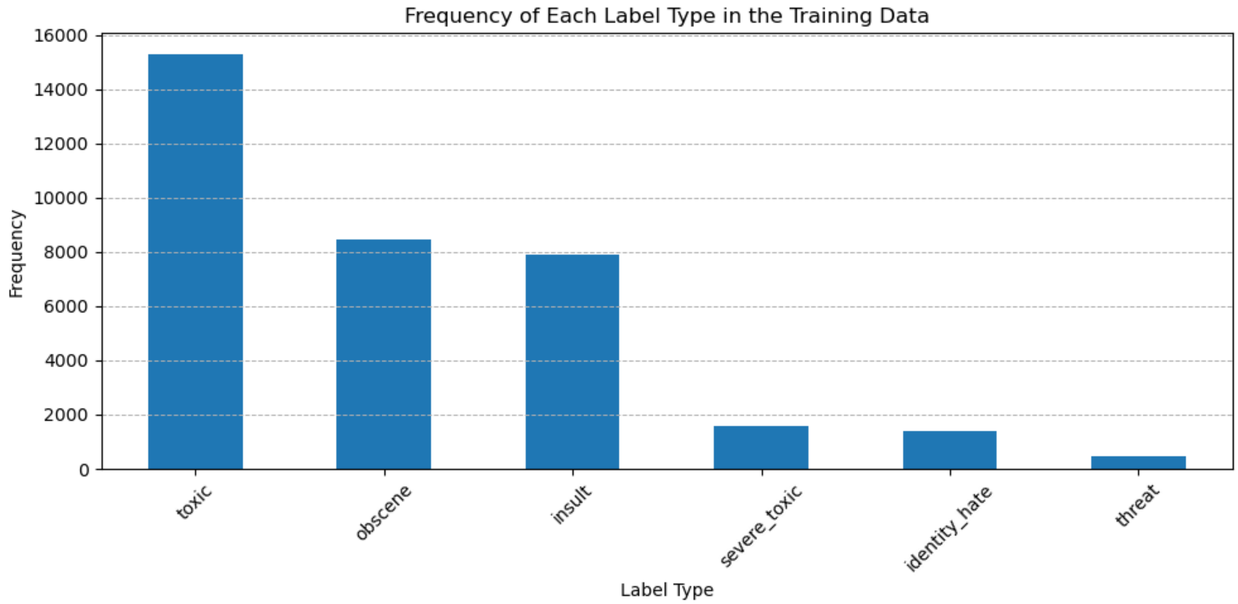
5. **Comment Length Correlation:** By analyzing comment length in relation to likes, I found no strong correlation between the length of a comment and its like.

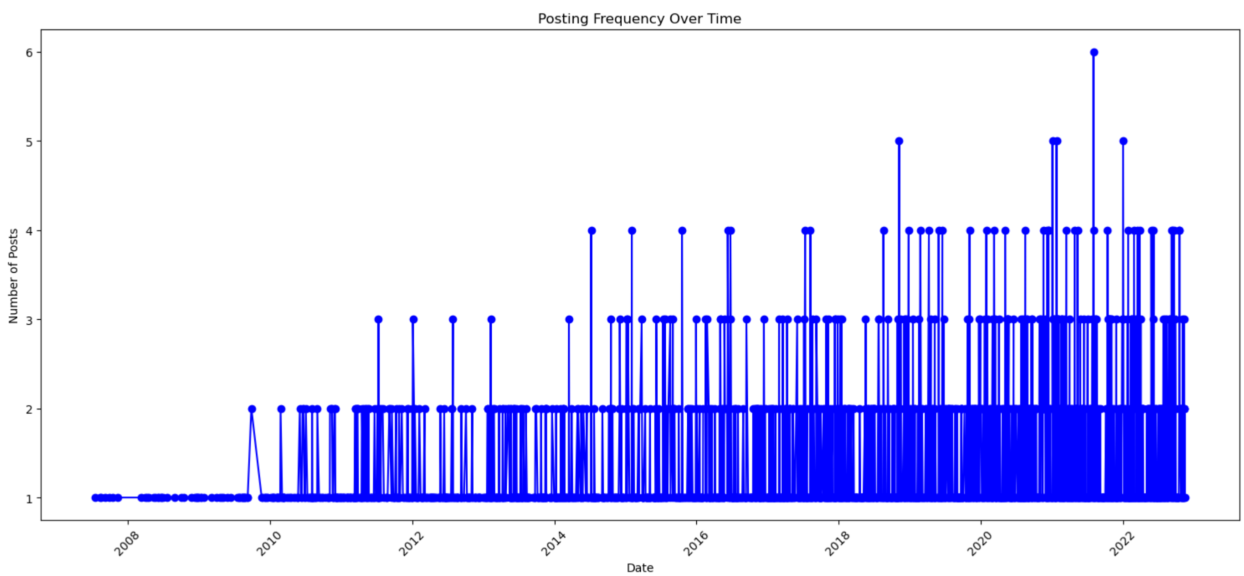
6. **Word Frequency Analysis:** I generated word clouds for specific users, providing a visual representation of the most frequently used words in their comments.

7. **Special Character Removal:** For UTF-8 compatibility with MySQL, I removed all special characters from the 'comment' column, retaining only alphanumeric characters and spaces. This step was crucial to prevent encoding issues during database import.

## Here’s some visualisation examples:

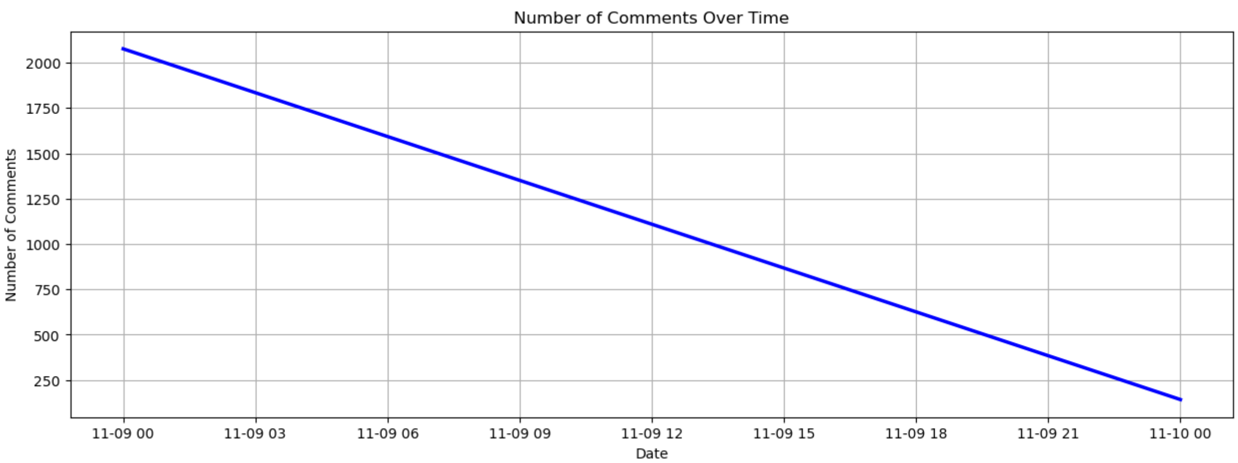



Frequency of Each Label Type in the Training Data
16000
14000
12000
10000
8000
6000
4000
2000
0
Frequency
toxic
obscene
insult
severe_toxic
identity_hate
threat
Label Type

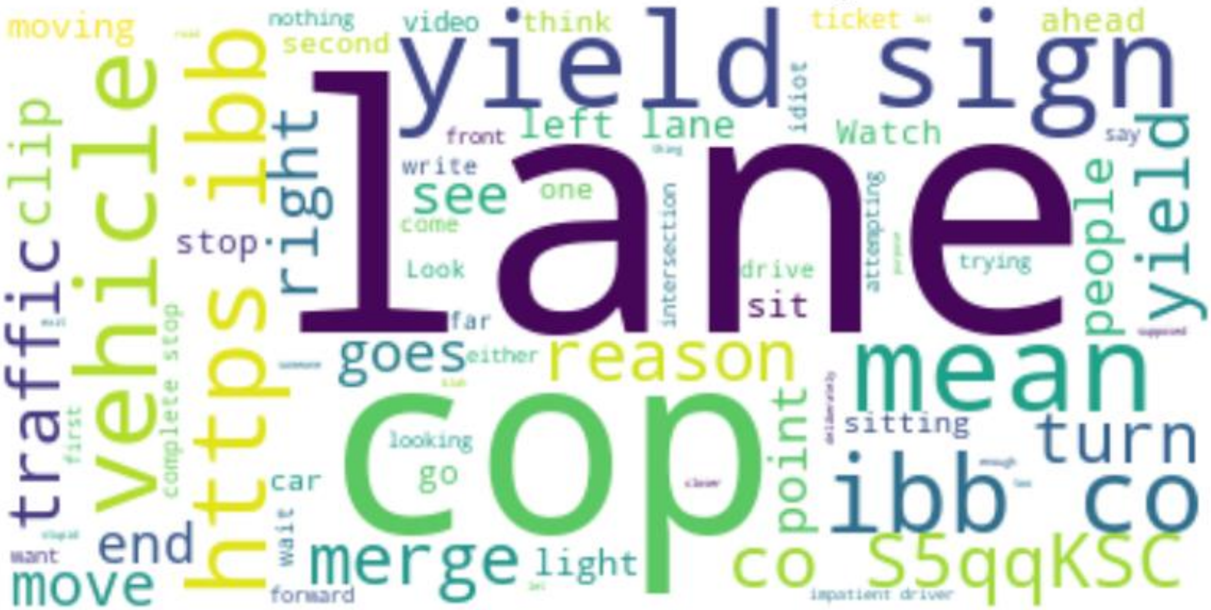

Posting Frequency Over Time
6
5
4
3
2
1
Number of Posts
2008
2010
2012
2014
2016
2018
2020
2022
Date

## Distribution of toxicity depending of the comment length (%)

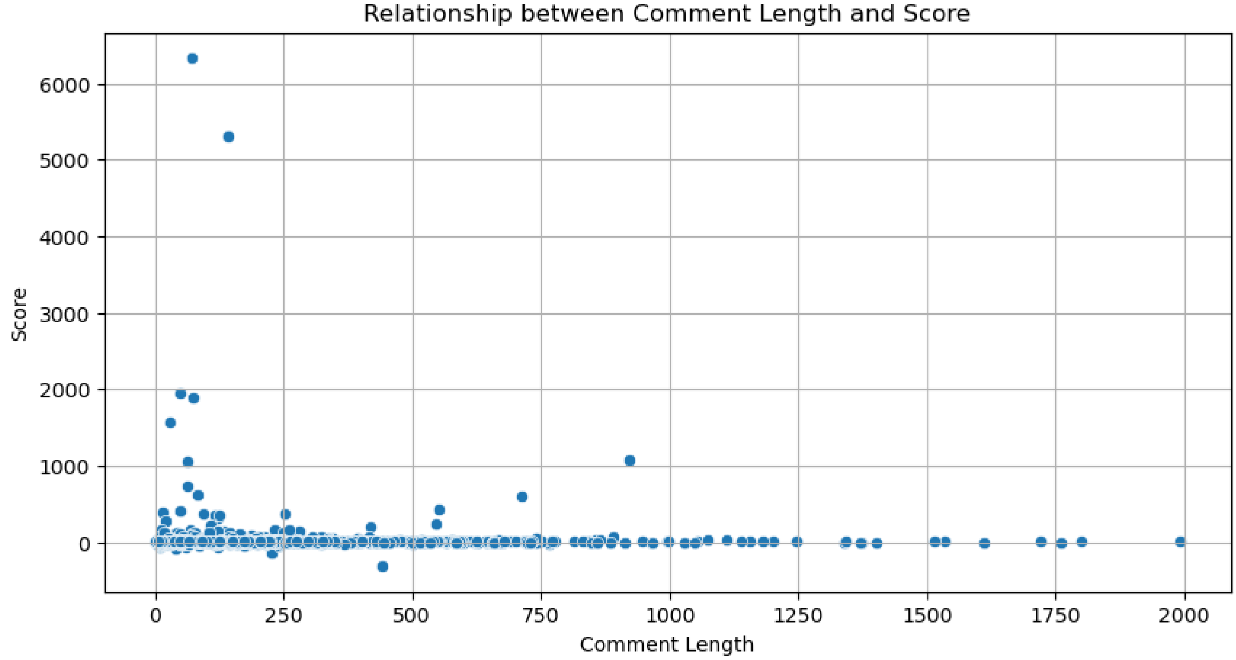
| length_category | toxic | severe_toxic | obscene | threat | insult | identity_hate |
|---|---|---|---|---|---|---|
| **(0, 50]** | 16.854900 | 2.324639 | 10.863510 | 0.425424 | 9.577108 | 1.610534 |
| **(50, 100]** | 14.062247 | 1.629113 | 7.869995 | 0.518723 | 7.533636 | 1.406225 |
| **(100, 200]** | 9.738177 | 0.880055 | 5.121316 | 0.356408 | 4.970528 | 0.841672 |
| **(200, 500]** | 6.921928 | 0.452371 | 3.471786 | 0.203459 | 3.313781 | 0.577910 |
| **(500, 1000]** | 5.265800 | 0.321269 | 2.610311 | 0.110436 | 2.389438 | 0.496963 |
| **(1000, 2000]** | 4.198430 | 0.364091 | 1.934236 | 0.068267 | 1.627034 | 0.352714 |
| **(2000, 5000]** | 8.714134 | 2.869288 | 5.313496 | 0.371945 | 4.357067 | 0.956429 |



Word Cloud for User BonnieMcMurray




Relationship between Comment Length and Score



Word Cloud for User Global_Lock_2049

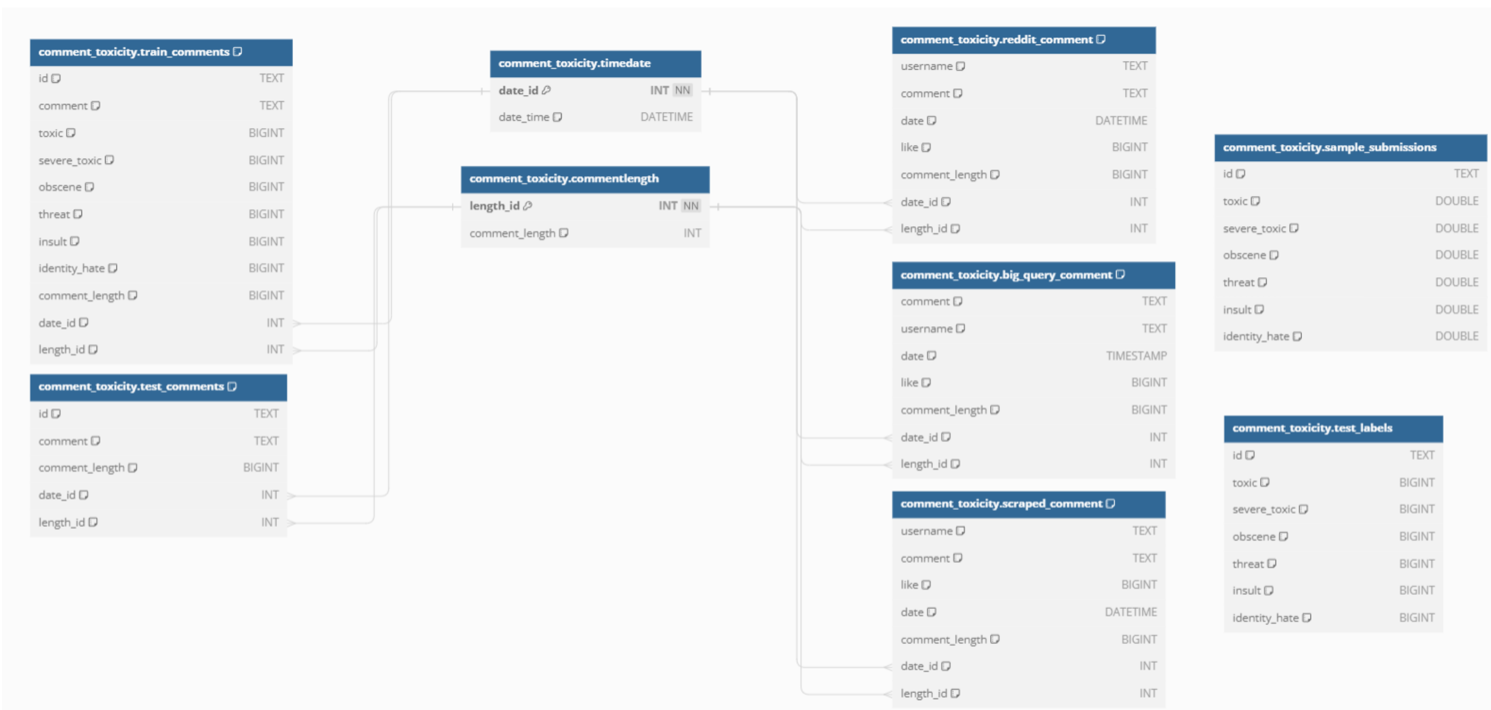
# Data base type selection

For this project, MySQL was the database of choice due to its excellent capability to handle structured data, which is abundant in social media commentary. Its relational nature allows me to define clear data schemas, making the organization and retrieval of data more straightforward. Additionally, MySQL's scalability means that as my data grows, the database can efficiently grow with it, which is crucial given the vast amount of comments collected.

MySQL also guarantees the integrity of my data with its transactional support. This is particularly important when integrating multiple data sources, as it ensures consistency throughout the datasets. The security features are top-notch, a non-negotiable aspect when dealing with potentially sensitive user data.

Lastly, the widespread community support and integration with numerous tools and languages streamline the process from data storage to analysis. MySQL's robustness, coupled with its ability to integrate seamlessly into the analytics pipeline, makes it an indispensable tool for my analysis.

# Entities. ERD



# Database's Queries

```sql
SELECT td.date_time AS Date, COUNT(*) AS NumberOfComments
FROM TimeDate td
JOIN Scraped_Comment sc ON td.date_id = sc.date_id
GROUP BY td.date_time
ORDER BY td.date_time;
```

```sql
SELECT AVG(cl.comment_length) AS AverageCommentLength
FROM CommentLength cl
JOIN Scraped_Comment sc ON cl.length_id = sc.length_id;
```

```sql
SELECT sc.username, COUNT(*) AS NumberOfComments
FROM Scraped_Comment sc
GROUP BY sc.username
ORDER BY NumberOfComments DESC
LIMIT 10;
```

```sql
SELECT tc.comment, tc.toxic, tc.severe_toxic, tc.obscene, tc.threat, tc.insult, tc.identity_hate
FROM Train_comments tc
WHERE tc.toxic = 1
  AND tc.severe_toxic = 1
  AND tc.obscene = 1
  AND tc.threat = 1
  AND tc.insult = 1
  AND tc.identity_hate = 1
ORDER BY tc.comment
LIMIT 10;
```

```sql
SELECT AVG(comment_length) AS Average_Length_Of_Highly_Toxic_Comments
FROM train_comments
WHERE toxic = 1
  AND severe_toxic = 1
  AND obscene = 1
  AND threat = 1
  AND insult = 1
  AND identity_hate = 1;
```

# Database's Queries – Outputs

| Date | NumberOfComments |
|---|---|
| 2022-11-13 10:40:59 | 380 |
| 2022-12-18 10:40:59 | 95 |
| 2023-01-17 10:40:59 | 18 |
| 2023-02-16 10:40:59 | 14 |
| 2023-03-18 10:40:59 | 26 |
| 2023-04-17 10:40:59 | 16 |
| 2023-05-17 10:40:59 | 26 |
| 2023-06-16 10:40:59 | 21 |
| 2023-07-16 10:40:59 | 27 |
| 2023-08-15 10:40:59 | 20 |
| 2023-09-14 10:40:59 | 41 |
| 2023-10-14 10:40:59 | 47 |
| 2023-11-01 10:40:59 | 4 |
| 2023-11-02 10:40:59 | 1 |
| 2023-11-03 10:40:59 | 1 |
| 2023-11-04 10:40:59 | 2 |
| 2023-11-05 10:40:59 | 1 |
| 2023-11-06 10:40:59 | 5 |
| 2023-11-07 10:40:59 | 2 |
| 2023-11-08 10:40:59 | 8 |
| 2023-11-09 10:40:59 | 3 |
| 2023-11-11 10:40:59 | 1 |
| 2023-11-12 14:40:59 | 1 |
| 2023-11-12 20:40:59 | 1 |

| AverageCommentLength |
|---|
| 128.7211 |

| username | NumberOfComments |
|---|---|
| arifcoco | 4 |
| joshuasJR | 4 |
| exactceamzis6525 | 4 |
| gutz1981 | 3 |
| zmaxpro5681 | 3 |
| daeneydirusso4069 | 3 |
| bri9146 | 3 |
| sfl6307 | 3 |
| carmenkoening7728 | 3 |
| stephaniedorene | 2 |

| comment | toxic | severe_toxic | obscene | threat | insult | identity_hate |
|---|---|---|---|---|---|---|
| I hope your retarded kids get anal rape… | 1 | 1 | 1 | 1 | 1 | 1 |
| AM GOING TO RAPE YOU IN THE ASS Y… | 1 | 1 | 1 | 1 | 1 | 1 |
| and your little faggot boy Propol pray pr… | 1 | 1 | 1 | 1 | 1 | 1 |
| ANYONE WHO SUPPORTS THIS IS FUCK… | 1 | 1 | 1 | 1 | 1 | 1 |
| Bitch You are a little bitch I fuckin spen… | 1 | 1 | 1 | 1 | 1 | 1 |
| Eat shit you fucking arse rapping jew fu… | 1 | 1 | 1 | 1 | 1 | 1 |
| faggot You lil piece of shit I havent van… | 1 | 1 | 1 | 1 | 1 | 1 |
| FAGGOTS YO FUCKER IT WAS FUCKIN… | 1 | 1 | 1 | 1 | 1 | 1 |
| Fuck All Asyriac Nation Qamishli belong … | 1 | 1 | 1 | 1 | 1 | 1 |
| fuck you honkey why you hatin on black… | 1 | 1 | 1 | 1 | 1 | 1 |

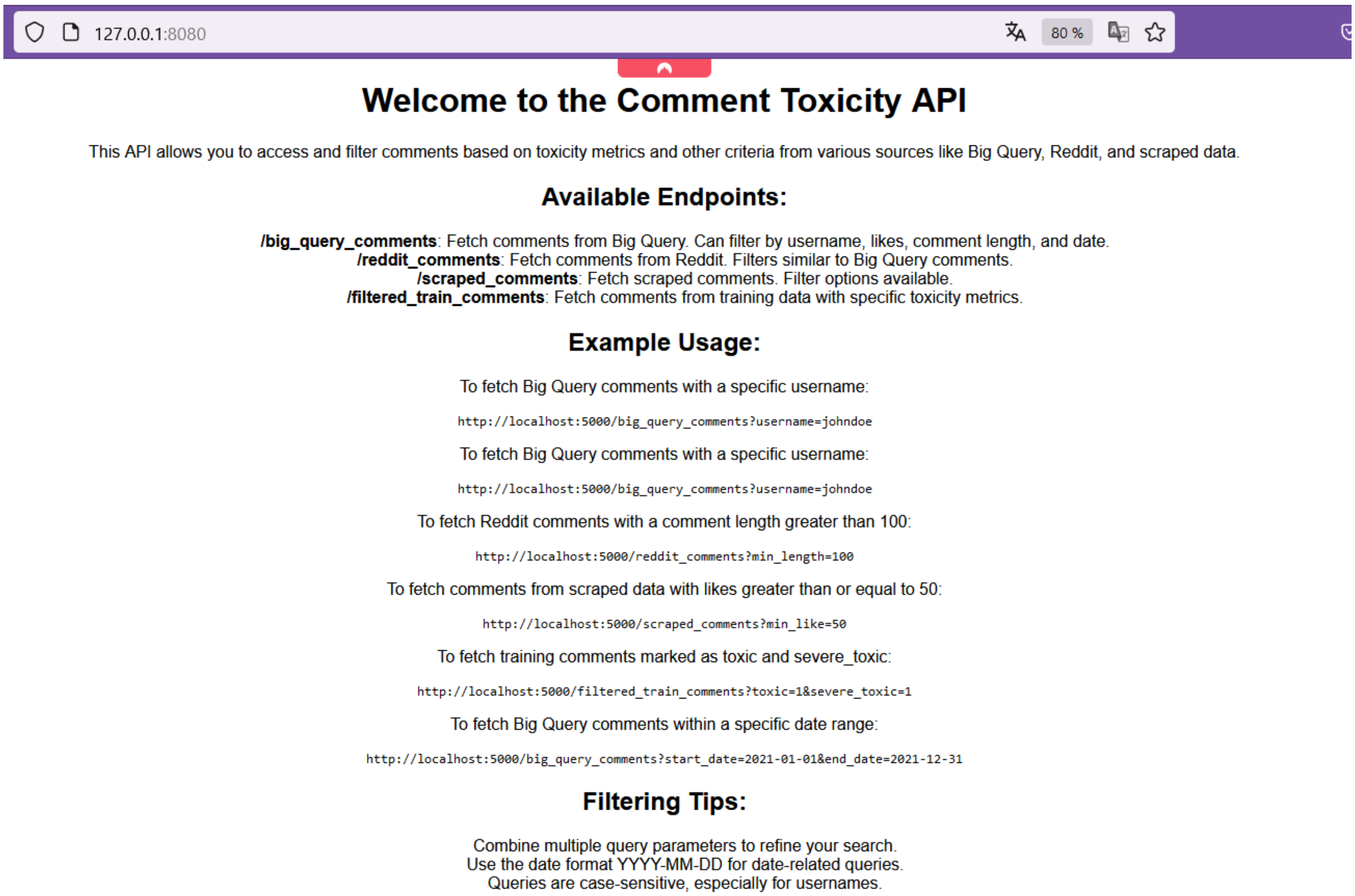
| Average_Length_Of_Highly_Toxic_Comments |
|---|
| 462.5806 |

# Exposing Data via API

```python
from flask import Flask, request, jsonify, abort
import pymysql
import os
import math

app = Flask(__name__)

# Database configuration
db_config = {
    'host': 'localhost',
    'user': 'root',
    'password': os.getenv("MySQL_password"),
    'db': 'comment_toxicity',
    'cursorclass': pymysql.cursors.DictCursor
}

# Database connection function
def get_db_connection():
    return pymysql.connect(**db_config)

@app.route('/')
def homepage():
```

```python
@app.route('/big_query_comments', methods=['GET'])
def get_big_query_comments():
    username = request.args.get('username')
    min_like = request.args.get('min_like', type=int)
    max_like = request.args.get('max_like', type=int)
    min_length = request.args.get('min_length', type=int)
    max_length = request.args.get('max_length', type=int)
    start_date = request.args.get('start_date')
    end_date = request.args.get('end_date')

    query = "SELECT username, comment, date, `like`, comment_length FROM big_query_comment WHERE 1=1 "
    params = []

    if username:
        query += "AND username = %s "
        params.append(username)

    if min_like is not None:
        query += "AND `like` >= %s "
        params.append(min_like)

    if max_like is not None:
        query += "AND `like` <= %s "
        params.append(max_like)

    if min_length is not None:
        query += "AND comment_length >= %s "
        params.append(min_length)

    if max_length is not None:
        query += "AND comment_length <= %s "
        params.append(max_length)

    if start_date:
        query += "AND date >= %s "
        params.append(start_date)

    if end_date:
        query += "AND date <= %s "
        params.append(end_date)

    conn = get_db_connection()
    cursor = conn.cursor()
    cursor.execute(query, tuple(params))
    comments = cursor.fetchall()
    conn.close()

    return jsonify(comments)
```

# API – Homepage




## Welcome to the Comment Toxicity API

This API allows you to access and filter comments based on toxicity metrics and other criteria from various sources like Big Query, Reddit, and scraped data.

### Available Endpoints:

**/big_query_comments**: Fetch comments from Big Query. Can filter by username, likes, comment length, and date.
**/reddit_comments**: Fetch comments from Reddit. Filters similar to Big Query comments.
**/scraped_comments**: Fetch scraped comments. Filter options available.
**/filtered_train_comments**: Fetch comments from training data with specific toxicity metrics.

### Example Usage:

To fetch Big Query comments with a specific username:

`http://localhost:5000/big_query_comments?username=johndoe`

To fetch Big Query comments with a specific username:

`http://localhost:5000/big_query_comments?username=johndoe`

To fetch Reddit comments with a comment length greater than 100:

`http://localhost:5000/reddit_comments?min_length=100`

To fetch comments from scraped data with likes greater than or equal to 50:

`http://localhost:5000/scraped_comments?min_like=50`

To fetch training comments marked as toxic and severe_toxic:

`http://localhost:5000/filtered_train_comments?toxic=1&severe_toxic=1`

To fetch Big Query comments within a specific date range:

`http://localhost:5000/big_query_comments?start_date=2021-01-01&end_date=2021-12-31`

### Filtering Tips:

Combine multiple query parameters to refine your search.
Use the date format YYYY-MM-DD for date-related queries.
Queries are case-sensitive, especially for usernames.

# Machine Learning

**Introduction to Machine Learning :**

Machine Learning (ML) is a pivotal branch of artificial intelligence that equips computer systems with the ability to learn and enhance their performance from data without explicit programming. In the context of my project, ML is deployed to automatically classify online comments into various toxicity categories, exemplifying the quintessential power of Natural Language Processing (NLP) in sentiment analysis.

**Preprocessing and Vectorization :**

Before training ML models, it's essential to transform textual comments into a structured format. I employed the TF-IDF (Term Frequency-Inverse Document Frequency) technique to convert the text into numerical vectors. This method weighs terms based on their relative importance within a comment against the corpus of documents. Vectorization using TF-IDF is a critical step in handling text for NLP tasks.

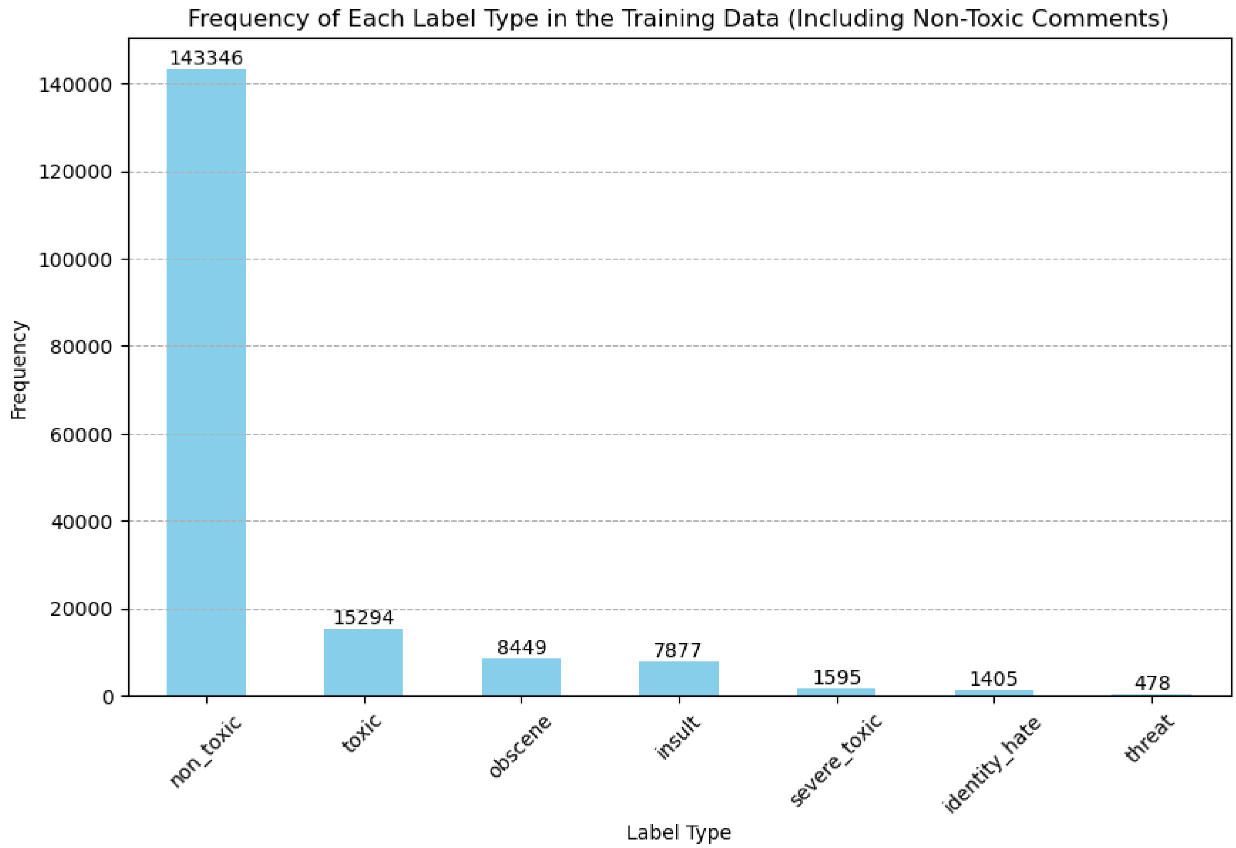
```
# Preprocessing: Vectorizing the text data using TF-IDF
tfidf_vectorizer = TfidfVectorizer(max_features=5000)
X = tfidf_vectorizer.fit_transform(train_df['comment'])
y = train_df[['toxic', 'severe_toxic', 'obscene', 'threat', 'insult', 'identity_hate']]
```

```
X
```

```
<159571x5000 sparse matrix of type '<class 'numpy.float64'>'
	with 5926788 stored elements in Compressed Sparse Row format>
```

### Model Selection and OneVsRestClassifier

I chose two robust and well-established classification models: logistic regression and Support Vector Machine (SVM). These models are widely used in text classification due to their efficacy with high-dimensional datasets. To adapt these models to a multi-label classification problem, where each comment might be tagged with multiple toxicity categories, I utilized OneVsRestClassifier. This meta-classifier strategy trains a separate binary classifier for each label, making it ideal for my use case where comments could belong to various categories of toxicity.



**Model Training :**

The models were trained using OneVsRestClassifier, allowing each label of toxicity to be treated independently. This means that a separate model was trained for each toxicity category, enabling each to specialize in recognizing a specific form of toxicity.

**Hyperparameter Optimization :**

The subsequent step was hyperparameter optimization via GridSearchCV, which tested various hyperparameter combinations for each model. Parameters C, max_iter, and solver were adjusted for logistic regression, while SVM saw adjustments in C and max_iter.

```
# Define the grid parameters for logistic regression
param_grid_logreg = {
    'estimator__C': [0.1, 1],  # Start with a smaller range
    'estimator__max_iter': [1000, 3000],  # Start with a lower max_iter
    |
}
```

**And <u>PICKLE</u> the trained models in case of <u>DEMO</u> :**

```
import pickle

# Sérialisation du modèle de régression logistique
with open('logreg_model.pkl', 'wb') as file:
    pickle.dump(logreg_model, file)

# Sérialisation du modèle SVM
with open('svm_model.pkl', 'wb') as file:
    pickle.dump(svm_model, file)

# Sérialisation du vectoriseur TF-IDF
with open('tfidf_vectorizer.pkl', 'wb') as file:
    pickle.dump(tfidf_vectorizer, file)
```

## Model Evaluation :

The models were evaluated based on several key metrics. Precision indicates the proportion of correct predictions among all positive predictions, recall assesses the model's ability to identify all actual positive instances, and the F1 score provides a balance between precision and recall, especially useful for imbalanced data.

```
Rapport de classification pour le modèle de régression logistique:
              precision    recall  f1-score  support
toxic          0.899759  0.628831  0.740285   2969.0
severe_toxic   0.454545  0.207006  0.284464    314.0
obscene        0.901361  0.638939  0.747795   1659.0
threat         0.636364  0.160920  0.256881     87.0
insult         0.823359  0.550678  0.659961   1549.0
identity_hate  0.670886  0.191336  0.297753    277.0
micro avg      0.863386  0.570678  0.687160   6855.0
macro avg      0.731046  0.396285  0.497857   6855.0
weighted avg   0.849898  0.570678  0.679056   6855.0
samples avg    0.054650  0.048156  0.049082   6855.0
Précision pour le modèle de régression logistique: 0.9215415948613505
```

```
Rapport de classification pour le modèle SVM:
              precision    recall  f1-score  support
toxic          0.878973  0.680027  0.766806   2969.0
severe_toxic   0.496503  0.226115  0.310722    314.0
obscene        0.884674  0.702833  0.783339   1659.0
threat         0.666667  0.252874  0.366667     87.0
insult         0.794195  0.582957  0.672375   1549.0
identity_hate  0.663934  0.292419  0.406015    277.0
micro avg      0.843960  0.621736  0.716002   6855.0
macro avg      0.730824  0.456204  0.550987   6855.0
weighted avg   0.832292  0.621736  0.708920   6855.0
samples avg    0.059020  0.054040  0.054244   6855.0
Précision pour le modèle SVM: 0.92279492401692
```

```
Rapport de classification pour le modèle SVM optimal:
              precision    recall  f1-score  support
toxic          0.878973  0.680027  0.766806   2969.0
severe_toxic   0.496503  0.226115  0.310722    314.0
obscene        0.884674  0.702833  0.783339   1659.0
threat         0.666667  0.252874  0.366667     87.0
insult         0.794195  0.582957  0.672375   1549.0
identity_hate  0.663934  0.292419  0.406015    277.0
micro avg      0.843960  0.621736  0.716002   6855.0
macro avg      0.730824  0.456204  0.550987   6855.0
weighted avg   0.832292  0.621736  0.708920   6855.0
samples avg    0.059020  0.054040  0.054244   6855.0
Précision pour le modèle SVM optimal: 0.92279492401692
```

```
Rapport de classification pour le modèle de régression logistique optimal:
              precision    recall  f1-score  support
toxic          0.899759  0.628831  0.740285   2969.0
severe_toxic   0.454545  0.207006  0.284464    314.0
obscene        0.901361  0.638939  0.747795   1659.0
threat         0.636364  0.160920  0.256881     87.0
insult         0.823359  0.550678  0.659961   1549.0
identity_hate  0.670886  0.191336  0.297753    277.0
micro avg      0.863386  0.570678  0.687160   6855.0
macro avg      0.731046  0.396285  0.497857   6855.0
weighted avg   0.849898  0.570678  0.679056   6855.0
samples avg    0.054650  0.048156  0.049082   6855.0
Précision pour le modèle de régression logistique optimal: 0.9215415948613505
```

## Applying Models to New Datasets

After training and optimization, the models were applied to new datasets to predict the toxicity of comments. This tested the models' ability to generalize to unseen data.

| | comment | username | date | like | comment_length | toxic | severe_toxic | obscene | threat | insult | identity_hate |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | gt That isnx27t really the point The point is ... | ardy42 | 2020-07-23 20:05:04+00:00 | 0 | 552 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | Why would you put garbage in your production d... | eru | 2015-09-07 03:27:57+00:00 | 0 | 55 | 1 | 0 | 0 | 0 | 0 | 0 |
| 2 | iwho the fuck are they designing these things ... | userbinator | 2018-02-04 07:15:32+00:00 | 0 | 71 | 1 | 0 | 1 | 0 | 0 | 0 |

| | comment |
|---|---|
| 0 | gt That isnx27t really the point The point is that the US armed forces already use space for military purposes so extensively that they even have a whole branch dedicated to space militarizationpMilitary use of space quotmilitarization of spacequot My understanding is that the US Space Force pretty much just controls space assets iusedi by the military that are comparable to civilian ones ie sensor communication and navigation satellites I donx27t think theyx27ve stationed any actual iweaponsi in space |
| 1 | Why would you put garbage in your production databases |
| 2 | iwho the fuck are they designing these things foripThe Germans |

**The model working :**

You can look at the distribution of toxicity labels for each dataset, in order, Hacker News, Reddit and Youtube

```
Big Query Logistic Regretion Distribution:
 toxic            0.500000
severe_toxic      0.000000
obscene           0.033333
threat            0.000000
insult            0.000000
identity_hate     0.000000
dtype: float64

Big Query SVM Distribution:
 toxic            0.833333
severe_toxic      0.000000
obscene           0.100000
threat            0.000000
insult            0.033333
identity_hate     0.000000
dtype: float64
```

```
Reddit Logistic Regretion Distribution:
 toxic            8.472285
severe_toxic      0.000000
obscene           4.416404
threat            0.000000
insult            2.478594
identity_hate     0.000000
dtype: float64

Reddit SVM Distribution:
 toxic            10.950879
severe_toxic      0.000000
obscene           6.173952
threat            0.000000
insult            3.289770
identity_hate     0.000000
dtype: float64
```

```
Scrapped Logistic Regretion Distribution:
 toxic            3.887884
severe_toxic      0.000000
obscene           0.316456
threat            0.000000
insult            0.135624
identity_hate     0.000000
dtype: float64

Scrapped SVM Distribution:
 toxic            5.244123
severe_toxic      0.000000
obscene           0.632911
threat            0.361664
insult            0.135624
identity_hate     0.045208
dtype: float64
```

Big Query

This result is not surprising given the subject of the big query dataset, Hacker News is known to be quite serious and the topic is very tech-centric

Reddit Comments Dataset:

This result is not surprising given the site in which the dataset was taken. Reddit is known for being pretty wild.

Scraped Comments Dataset:

This result is not surprising given the site in which the dataset was taken. Youtube is known for being wild at times but is pretty well moderated.

# Challenges

Throughout this project, I encountered a number of challenges that tested my skills and determination:

1. **Web Scraping with Selenium**: Grasping the inner workings of the ChromeDriver alongside the Selenium library for web scraping was initially daunting. It took considerable effort to understand the nuances of interacting with web elements and automating the extraction process efficiently.

2. **API Authentication Issues**: I faced a frustrating hurdle with the Reddit API when my environment variables for the Client_ID and Secret_ID didn't function as expected. It was a lesson in debugging and securing sensitive information properly within my development environment.

3. **Date Conversion**: One particularly tricky aspect was dealing with relative dates in the data, such as "3 years ago" or "9 months ago." I had to devise a method to convert these into absolute dates to create a usable timeline for my analysis.

4. **Entity-Relationship Model Creation**: Designing the ER model to accurately represent the data was intricate, especially when adding tables like DateID and CommentLengthID. It was essential to establish meaningful relationships between tables to support comprehensive queries and insights.

5. **Refreshing HTML and CSS Skills**: Finally, building the homepage for the Flask API required me to revisit and refresh my knowledge of HTML and CSS. It was a challenge to get back up to speed and ensure the front-end design was both functional and aesthetically pleasing.

6. **Model and Hyperparameters :** finding the right model and the right process was quite complicated to manage given the nature of my data train, I had to manage 6 labels and not just one, it was more than a simple sentiment analysis. And another thing, it was quite complicated for my computer to make the hyperparameter with the logistic reg, I still don't know to this day why it had so much difficulty pushing the hyperparameters further but I found a solution.

These challenges were demanding, but overcoming them has significantly improved my technical abilities and problem-solving acumen. Each obstacle was an opportunity for growth, and mastering these areas was immensely satisfying and added depth to my expertise.

# References

**Links:**

https://www.youtube.com/watch?v=kuhhT_cBtFU&t=2s
https://www.youtube.com/watch?v=v3abZ4aAGUU
https://www.reddit.com/r/funny/comments/17r7lh2/was_he_impatient_or_does_he_have_a_point/?onetap_auto=true
Jigsaw Toxic Comment Classification Challenge
https://console.cloud.google.com/bigquery?p=bigquery-public-data&d=hacker_news&page=dataset&project=da-bootcamp-2023

**TRELLO:**
https://trello.com/b/eUwdijbo/final-project

**GITHUB:**
https://github.com/GuillaumeCapelli/Final_Project

# THANK YOU