**ITCS-665 Natural Language Processing - Midterm Project**

**Sentiment Analysis and Toxicity Detection for Text Messages**

**Thant Zin Myo (6837929)**

**Contents**

[1. Introduction 2](#_Toc210268951)

[2. Goal & Objectives 2](#_Toc210268952)

[3. Learning Outcomes 3](#_Toc210268953)

[4. Dataset Description 3](#_Toc210268954)

[5. Exploratory Data Analysis 4](#_Toc210268955)

[6. Preprocessing 5](#_Toc210268956)

[6A. Cleaning 5](#_Toc210268957)

[6B. Tokenization & Lemmatization 5](#_Toc210268958)

[6C. Feature Engineering 5](#_Toc210268959)

[7. Model Selection 5](#_Toc210268960)

[8. Modeling 6](#_Toc210268961)

[8.1 Full Process 6](#_Toc210268962)

[9. Evaluation 7](#_Toc210268963)

[10. Comparative Analysis 7](#_Toc210268964)

[11. Ethical Considerations 8](#_Toc210268965)

[12. Conclusion and Future Work 8](#_Toc210268966)

[13. References 9](#_Toc210268967)

[14. Acknowledgment 9](#_Toc210268968)

**Table of Figures**

[Figure 1 Class Distribution 3](#_Toc210269294)

[Figure 2 Data analysis with numbers 4](#_Toc210269295)

[Figure 3 Word distribution by class 4](#_Toc210269296)

[Figure 4 Word count percentiles and outlier 4](#_Toc210269297)

[Figure 5 Text Pattern Analysis for Cleaning 4](#_Toc210269298)

[Figure 6 Full Process 6](#_Toc210269299)

[Figure 7 Comparison Summary 7](#_Toc210269300)

[Figure 8 Confusion Matrices 7](#_Toc210269301)

[Figure 9 Model Performance Comparison 8](#_Toc210269302)

# 1. Introduction

My current company is developing a live streaming application where viewers can interact in real time by sending chat messages and stickers, similar to popular platforms like Twitch or TikTok. These interactive features make streams more lively and enjoyable, as the audience can share thoughts, reactions, and support instantly.

However, when many people join a live session, the chat often includes toxic or hateful comments. Such comments can harm the streamer, create a negative environment, and even push other viewers away. To maintain a positive community, there is a need for a system that can automatically detect and filter harmful messages.

This project focuses on building a toxic comment classification model. The model will analyze each message and decide whether it is safe (non-toxic) or harmful (toxic). While the main use case is live streaming, such a system is also valuable for online games, forums, and social media where open chat is common.

The overall goal is to create a safer and more inclusive environment that protects users while still allowing them to freely express themselves with messages and stickers.

# 2. Goal & Objectives

* Apply knowledge of text preprocessing and classification techniques.
* Design and implement a text classification.
* Evaluate and compare models with and without Lexicons helps.

# 3. Learning Outcomes

By the end of this project, we can:

* Preprocess and prepare raw text data for classification.
* Apply and compare classification approaches.
* Analyze performance metrics and interpret model behavior.
* Communicate findings effectively through documentation and visualization.

# 4. Dataset Description

For this project, we are using the dataset from **Kaggle’s Jigsaw Toxic Comment Classification Challenge**. This dataset has comments collected from Wikipedia pages. Each comment is labeled into six toxic categories: **toxic, severe toxic, obscene, threat, insult, and identity hate**. To make the problem easier we combined all of them into one label:

* **toxic (1)** -> if the comment belongs to any of the six categories
* **non-toxic (0)** -> if the comment does not belong to any category

**Dataset details:**

* **Size:** 159,571 comments
* **Class balance:** about **90% non-toxic** and **10% toxic**
* **Format:** CSV file with comment text and multiple label columns

A graph with a bar and a red and blue bar

AI-generated content may be incorrect.

Figure 1Class Distribution

In addition to the dataset, we will also use **Five lexicon hate word lists** [**(luis von ahn-badwords.txt**](https://www.cs.cmu.edu/~biglou/resources/bad-words.txt)**,** [**ldnoobw-bad-words**](https://github.com/LDNOOBW/List-of-Dirty-Naughty-Obscene-and-Otherwise-Bad-Words/blob/master/en)**,** [**pmathur5k10 -hinglish-offensive-texts**](https://github.com/pmathur5k10/Hinglish-Offensive-Text-Classification/blob/main/Hinglish_Profanity_List.csv)**,** [**orthrus-lexicon**](https://github.com/Orthrus-Lexicon/Toxic/blob/main/Toxic%20words%20dictionary.txt)[**toxic**](https://github.com/Orthrus-Lexicon/Toxic)**,** [**multilingual\_toxic\_lexicon**](https://huggingface.co/datasets/textdetox/multilingual_toxic_lexicon) **)**. These lexicons are collections of words that are often used in toxic or hateful comments. By combining the dataset with the lexicon approach, we can compare how well machine learning models work alone, with lexicons only, and with both methods together.

# 5. Exploratory Data Analysis

* **Label Distribution**: As we can see in Fig-1 Distribution is quite imbalanced (Non-toxic: 143k, Toxic: 16k). and average length with 394 characters and around 67 words per comments. We also detect some text with HTML, Quoted Text, Number and URL which are necessary for cleaning.

A black screen with white text

AI-generated content may be incorrect.

A graph of a number of numbers

AI-generated content may be incorrect.Figure 2 Data analysis with numbers

A graph of a graph showing a number of numbers

AI-generated content may be incorrect.

Figure 3 Word distribution by class Figure 4 Word count percentiles and outlier

* **Patterns detected**: 5,151 comments with URLs, 49,996 with quotes, 51,212 with numbers.

A graph with different colored bars

AI-generated content may be incorrect.

Figure 5 Text Pattern Analysis for Cleaning

# 6. Preprocessing

## 6A. Cleaning

The cleaning pipeline included:

* Lowercasing text
* Removing numbers, quotes, URLs, and HTML tags
* Expanding contractions (e.g., *don’t* -> *do not*) with the help of [andrewbury contractions sources](https://github.com/andrewbury/contractions/blob/master/contractions.txt)
* Removing punctuation (keeping only alphabetic tokens)
* Stripping extra whitespace

## 6B. Tokenization & Lemmatization

* **Tokenization**: Splitting text into words using NLTK.
* **Stop word removal**: Common English words removed (e.g., “the”, “and”).
* **Lemmatization**: Words reduced to their base form (e.g., *running -> run*).

## 6C. Feature Engineering

We experimented with:

* **TF-IDF vectorization** with unigrams and bigrams (max features=10,000).
* **Count Vectorizer** with unigrams and bigrams (max features=10,000 )+ (4760 words but remove duplicate = 3836 lexicon words).

# 7. Model Selection

We focused on simple but effective classifiers called **Multinomial Naive Bayes (MNB) and Logistic Regression (LR)**. It is often used in text classification tasks.

We tested 4 different models with different setting:

* + **Model 1: Multinomial Naive Bayes**
    - **Setting: TF-IDF, 3A-Cleaning, 3B- Tokenization & Lemmatization**
  + **Model 2: Multinomial Naive Bayes**
    - **Setting: Lexicon, Bag of words, 3A-Cleaning, 3B- Tokenization & Lemmatization**
  + **Model 3: Logistic Regression**
    - **Setting: TF-IDF, 3A-Cleaning, 3B- Tokenization & Lemmatization**
  + **Model 4: Logistic Regression**
    - **Setting: Lexicon, Bag of words, 3A-Cleaning, 3B- Tokenization & Lemmatization**

# 8. Modeling

**Train/Test Split**

We divided the dataset into two parts:

For without lexicons Dataset we split into

* 80% for training (127,656 samples)
* 20% for testing (31,915 samples)

For with lexicons Dataset we split into

* 80% for training (127,656 samples)+ lexicon word = 132, 416 sample
* 20% for testing (31,915 samples)

## 8.1 Full Process

Figure 6 Full Process

# 9. Evaluation

We measured accuracy, precision, recall, F1-score, and confusion matrix values. Naive Bayes with TF-IDF reached 94.32% accuracy, 93.82% precision, 47.27% recall, and an F1-score of 62.87%. Adding lexicons and Changing to Bag of words improved recall to 54.33% and F1 to 67.17%. Logistic Regression with TF-IDF gave 93.22% accuracy, 64.25% precision, 75.10% recall, and F1 of 69.25%. With lexicons and changing to Bag of words improved slightly, reaching 93.75% accuracy, 67.66% precision, 73.84% recall, and F1 of 70.62%.

Overall, Naive Bayes was precise but missed many toxic comments, while Logistic Regression detected more toxic cases and offered a better balance. Lexicons + bag of words models gave small improvements in both models.

A close-up of a chart

AI-generated content may be incorrect.

Figure 7 Comparison Summary

A blue and white squares with black text

AI-generated content may be incorrect.

Figure 8 Confusion Matrices

# 10. Comparative Analysis

Naive Bayes worked best for precision, meaning it rarely misclassified safe comments as toxic, but it missed many toxic ones. Logistic Regression was more balanced and achieved higher recall, which is more useful for real-time moderation. Adding lexicons helped both models by catching more toxic words, though at the cost of some extra false positives. The best overall model was Logistic Regression with bag of words and lexicons.

A graph of different colored bars

AI-generated content may be incorrect.

Figure 9 Model Performance Comparison

# 11. Ethical Considerations

Toxic comment detection raises fairness and bias concerns. Since the dataset and lexicons are mainly in English, the models may fail on other languages or unfairly flag dialects and slang. Overly strict filtering can also block harmless jokes or expressions. Because of this, such systems should support multiple languages and always include report or reclaim function so that we can moderate to avoid unfair censorship.

# 12. Conclusion and Future Work

For future work, more advanced models or deep learning models like BERT should be tested, cross-validation should be used for stronger evaluation, and error analysis should be done to see which comments are missed. Expanding to multiple languages and combining with human moderators will make the system fairer and more reliable.

Currently, the best model achieved an F1-score of around **0.7**, which shows good progress but also highlights that further improvement is still needed. Additional training with larger and more diverse datasets, along with fine-tuning on real-world chat data, will help improve overall performance.

# 13. References

* Kaggle. (2018). *Jigsaw Toxic Comment Classification Challenge*.  
  <https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge>
* von Ahn, L. (n.d.). *CMU Offensive/Profanity Word List*. Carnegie Mellon University.  
  <https://www.cs.cmu.edu/~biglou/resources/>
* LDNOOBW. (n.d.). *List of Dirty, Naughty, Obscene, and Otherwise Bad Words*. GitHub.  
  <https://github.com/LDNOOBW/List-of-Dirty-Naughty-Obscene-and-Otherwise-Bad-Words>
* Mathur, P. (n.d.). *Hinglish Offensive Text Classification – Profanity Lexicon*. GitHub.  
  <https://github.com/pmathur5k10/Hinglish-Offensive-Text-Classification>
* Orthrus Lexicon Project. (n.d.). *Toxic Words Dictionary*. GitHub.  
  <https://github.com/Orthrus-Lexicon/Toxic>
* Hugging Face. (n.d.). *Multilingual Toxic Lexicon – TextDetox Dataset*.  
  <https://huggingface.co/datasets/textdetox/multilingual_toxic_lexicon>
* Bury, A. (n.d.). *English Contractions List for NLP Tasks*. GitHub.  
  <https://github.com/andrewbury/contractions/blob/master/contractions.txt>

# 14. Acknowledgment

All modeling and analysis were carried out by the author, while ChatGPT (current version GPT-5) was only used to improve grammar and readability.

The Jupyter Notebook and training dataset can be downloaded from GitHub: