Toxic Behavior Detection and Prediction in Online Games

Comparing performances between text-based, sentiment-based, and LLM-based models

Jiajun Fang, Manish Kanuri, Sushma Ramesh

GitHub

Abstract

Toxic behavior in online multiplayer games, including verbal harassment, hate speech, and disruptive actions, poses a significant challenge to maintaining positive player experiences and safe gaming environments. With 81% of online gamers reporting harassment, and 68% encountering severe abuse, addressing this issue is crucial for both social and financial reasons. Traditional word-based models for detecting toxicity are often insufficient as they fail to account for context and nuance in player interactions. While human screening offers better accuracy due to its ability to understand context and intent, its scalability is limited by the overwhelming volume of reports in large-scale games. This project aims to leverage advanced language models (LLMs) to better understand the context, sentiment, and intent behind player messages, offering a more accurate and automated solution.

The approach involves three models with progressively sophisticated techniques for detecting toxic behavior. The traditional model, which identifies offenders based on specific harmful words, performs three times better than random guessing, providing a baseline for comparison. The sentiment analysis model, which assesses the overall tone of a player's interactions, surprisingly yields lower accuracy than the baseline model, even after balancing the skewed dataset. The most disappointing results come from the LLM-based model. Although it outperforms the other two approaches—achieving up to 20% better results than the baseline model—its accuracy falls significantly below our initial expectations. This highlights both the potential and limitations of context-aware language models in detecting toxic behavior.

A major limitation of this approach stems from the inherently complex nature of the task. Accurately identifying toxic behavior in multiplayer game chats involves understanding nuanced, context-dependent interactions, such as sarcasm and tone shifts, and distinguishing different types of negative behaviors (attacking vs defending), which makes achieving high accuracy challenging. Despite the improvements seen with LLMs, the unpredictability of human behavior in gaming contexts still presents a significant hurdle.

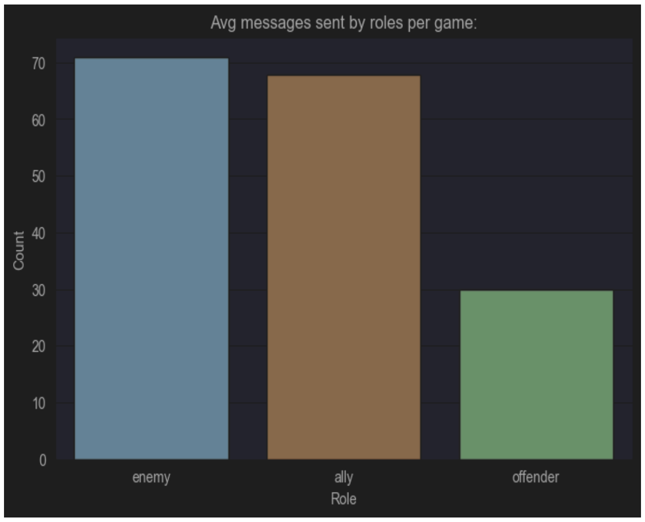
Experiment setup

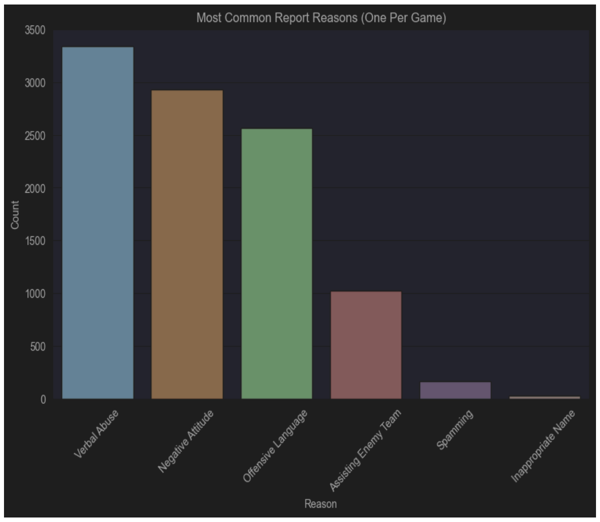
Dataset Description and EDA

This dataset contains cases from an online video game called *League of Legends*. A case is created when one or more players report another player for toxic behavior. The dataset consists of nine columns. **Offender** refers to the single player reported in the tribunal case. Each row represents a single message sent by a player, and each game contains only one offender. The description of each column is as follows:

- **Message:** The content of the message.
- **Association_to_offender:** Indicates whether the player is on the same team or the opposing team as the offender.
- **Time:** The timestamp when the message was sent.
- **Case_total_reports:** The number of reports received before the case is brought to the tribunal.
- **Allied_report_count:** The number of reports from allies on the same team as the offender.
- **Enemy_report_count:** The number of reports from players on the opposing team.
- **Most_common_report_reason:** The most common report reason among the five available categories.
- **Chatlog_id:** A unique identifier for each chat log.
- **Champion_name:** The in-game character used by the player who sent the message.

After performing exploratory data analysis (EDA), we gathered the following insights: There are a total of 10,058 cases and more than 1.7 million messages. On average, offenders sent 1.7 times more messages than their teammates and almost twice as many as players on the opposing team. Regarding report reasons, *verbal abuse*, followed by *negative attitude* and *offensive language*, accounts for 90% of all cases. Interestingly, the top 10 most reported champions are also the hardest to play.





Models

Traditional Model: Zero-Tolerance Word-Based

The Zero-Tolerance Word-Based Model is a rule-based approach that detects toxic players in multiplayer game chats by leveraging a predefined dictionary of harmful words and phrases (e.g., "noob," "trash," "idiot," "stupid"). The implementation starts with data cleaning, where rows with missing values in critical columns like message, chatlog_id, and player_id are removed to ensure data consistency. Messages are standardized to string format to prevent processing errors, enabling smooth downstream tokenization and analysis.

Next, the data is grouped by chatlog_id and player_id, allowing all messages sent by a specific player in a game to be aggregated into a single input. The aggregated messages are tokenized into individual words, which are then matched against the toxic word dictionary. The occurrences of harmful terms are counted for each player, and these counts are aggregated to calculate the total toxic word count for each player within a game.

The player with the highest toxic word count in each game is flagged as the predicted offender. In addition, the model generates a list of all flagged players who used any toxic words, along with their respective toxic word counts. This supplementary output provides further insights into overall game toxicity, even if the predicted offender is incorrect.

This implementation is efficient, requires minimal computational resources, and relies solely on simple text processing and aggregation. Its rule-based nature ensures that results are interpretable and directly traceable to specific toxic words, making it transparent and auditable. By isolating clear cases of toxic behavior, this model serves as an easy baseline for detecting offenders in multiplayer game chats while paving the way for more context-aware and sophisticated techniques in future iterations.

Model Architecture: The Zero-Tolerance Word-Based Model operates by analyzing aggregated chat messages from each player within a game, identified by chatlog_id and player_id, alongside a predefined dictionary of toxic words. Messages are first tokenized into individual words for analysis. These tokens are then filtered by matching them against the toxic word dictionary, and occurrences of harmful terms are counted. The counts are aggregated across all messages for each player, providing a comprehensive toxic word count. The player with the highest toxic word count in a game is flagged as the predicted offender. Additionally, the model outputs a list of all flagged players with their respective toxic word counts for further analysis.

Sentiment Analysis Model

Implementation: The process starts by importing necessary libraries and loading the chat logs dataset. Data cleaning involves checking for missing values in key columns and dropping rows with missing values in chatlog_id, champion_name, and association_to_offender. The dataset is filtered to include only games with offenders, ensuring relevant data is retained. The message column is converted to strings, and a unique player ID is assigned based on chatlog_id, champion_name, and association_to_offender.

The data is grouped by player, with messages aggregated for each player and a label assigned based on the player's association to the offense. Sentiment analysis is conducted using VADER SentimentIntensityAnalyzer to calculate a sentiment score for each player's messages. These scores are used as features, and the dataset is split into training and test sets using an 80:20 ratio.

A Random Forest Classifier is then trained on the sentiment scores, and predictions are made on the test set. The model's performance is evaluated using the F1 score, confusion matrix, and classification report, with the F1 score providing a balanced measure of effectiveness given the class imbalance.

Model Architecture: Random Forest is an ensemble learning algorithm, which means it builds multiple decision trees and aggregates their results for classification.

The model takes in a single feature, the sentiment score of each player's messages, computed using the VADER Sentiment Analysis tool. This tool provides a compound score representing the overall sentiment of the messages, which serves as the input for classification.

A Random Forest comprises an ensemble of decision trees. Each tree is trained on a random subset of training data (via bootstrapping) and a random subset of features, ensuring diversity among the trees. This structure reduces overfitting and improves generalization. Each tree splits the data based on the sentiment score to minimize impurity, such as Gini impurity or entropy, for predicting whether a player is an offender (label 1) or not (label 0).

Once trained, the Random Forest aggregates predictions using majority voting. For binary classification, it assigns the label (offender or non-offender) based on the majority vote. Using the default scikit-learn RandomForestClassifier with random_state=42, reproducibility is ensured. Hyperparameters like **n_estimators, max_depth, and min_samples_split** can be tuned to optimize the model’s performance.

LLM Model

Implementation: The model used for this task is the **T5 model** (Text-to-Text Transfer Transformer). The input to the model is a game chat log, and the task is to predict who is the offender within this game based on the context of the entire chat. T5 was chosen for its capability to handle both text classification and generation tasks, making it suitable for understanding complex relationships in the data.

The first step in preparing the dataset involved **stacking all the chats for each chatlog** into a single conversation, ensuring the chat sequence accurately represented player interactions. This was done by including each player's name alongside their messages, creating a conversational flow of dialogue. Additionally, the name of the offender for each game was added as a separate column to serve as the target label for the model.

To further enhance the model's performance in detecting toxicity, a **custom prompt** was added before each chatlog. This prompt helps the model understand the context better by signaling the task, for example: "Identify the offender in the following conversation." The prompt was carefully designed to guide the model's attention toward the core task of identifying the player responsible for toxic behavior.

The **T5Tokenizer** was then used to preprocess the text, ensuring it was in the correct format for the model. This involved tokenizing the chat logs and embedding the text to prepare it for input into the T5 model. After preprocessing, the model is trained using the **AdamW optimizer**, which is a popular choice for fine-tuning transformers. The learning rate was optimized for the model's convergence, and a reasonable batch size was used to ensure efficient training, depending on available GPU resources. The model was trained for 8 epochs with early stopping to avoid overfitting and underfitting, ensuring the model learned the necessary patterns without being too specific to the training data.

Model Architecture: The T5 model follows the encoder-decoder architecture typical of transformer models. In the context of this task:

- **Encoder**: The encoder processes the input text, encoding the chat logs into a dense representation.
- **Decoder**: The decoder takes this representation and generates predictions based on the context of the entire conversation, identifying which player is the likely offender.

This structure enables the model to understand the sequential nature of chat logs and the relationships between players, allowing for more accurate identification of toxic behavior in multiplayer game settings.

The model is executed in a **Google Colab** environment, leveraging GPUs (e.g., NVIDIA Tesla T4 or A100) to accelerate training. PyTorch is the primary framework for implementing and training the model, with support for distributed computing and GPU acceleration, ensuring faster convergence.

Experiment Result

Traditional Model

Main Results: The Traditional Zero-Tolerance Word-Based Model demonstrates the utility of a straightforward rule-based approach for detecting toxic players but highlights certain limitations. The model achieved an overall accuracy of 33%, correctly identifying the offender in approximately one-third of the games. It outputs the predicted offender (player with the highest toxic word count) and a list of flagged players for each game.

While it successfully identified offenders in cases like chatlog_id = 5 (player_id = 39), it failed in others such as chatlog_id = 1 (actual offender: player_id = 16). The model struggles with nuanced toxic behavior (e.g., sarcasm) and cannot adapt to dynamic language patterns, relying solely on a static toxic word dictionary. Despite its limitations, the flagged players list provides valuable insights into overall game toxicity.

Supplementary Results: The Traditional Zero-Tolerance Word-Based Model was designed with specific parameters to optimize performance. A comprehensive dictionary of harmful words (e.g., "noob," "trash," "idiot") was used to focus on explicit toxic behavior common in multiplayer games. Data cleaning ensured integrity by removing missing values and non-string messages, while message aggregation provided a complete view of each player's behavior within a game. The player with the highest toxic word count was flagged as the offender, aligning with the dataset’s structure. Evaluation metrics, including accuracy and flagged players, assessed performance and offered insights into overall toxicity. The model's strengths lie in its simplicity and ability to detect explicit toxicity, though it struggles with nuanced behavior like sarcasm and relies on a static dictionary. This approach serves as a robust baseline for future improvements.

Sentiment Analysis Model

Main Results: The Random Forest Classifier achieved an accuracy of 87%, an F1-score of 0.78, and detected 134 true positives (14%), ineffectively identifying offenders despite an attempt to fix the class imbalance. Adjusting the offender-to-non-offender ratio to 1:2 enhanced true positive detection a bit while avoiding excessive false positives, outperforming other models like logistic regression (112 true positives) and decision trees (121 true positives), but the model was mainly focusing on detecting who isn't an offender.

Key findings highlight the importance of managing class imbalance and incorporating sequential context for robust predictions. Sentiment analysis using VADER provided limited insight, suggesting the need for deeper linguistic processing. Future improvements could include transformer-based models like BERT for capturing semantic nuances, hyperparameter optimization for refining performance, and exploring alternative class ratios to further enhance offender detection.

Supplementary Results: The Random Forest model used 100 decision trees (n_estimators) as a baseline, balancing computational cost and performance. Default values for parameters like max_depth, min_samples_split, and min_samples_leaf were maintained initially, focusing on achieving baseline performance. A fixed random_state of 42 ensured reproducibility, particularly in train-test splits and tree-building processes. For evaluation, an 80:20 train-test split was adopted, providing a sufficient training set and robust test set for reliable evaluation.

To address class imbalance, a 1:2 ratio of offenders to non-offenders was maintained, avoiding the noise that could arise from over-sampling offenders in a 1:1 ratio. Chat messages were aggregated by player_id and sorted by timestamp to retain the sequence and context of interactions. Data cleaning involved removing rows with missing values and ensuring all messages were in string format for compatibility. True positives were prioritized for evaluating offender detection, supported by F1-score to balance precision and recall. Future work will focus on hyperparameter optimization, exploring advanced balancing techniques like SMOTE, new feature generation to expand sentiment dimsneions, and refining text preprocessing to enhance feature quality.

LLM Model

Main Results: The primary experimental result of the toxicity detection model using the T5 architecture was an accuracy of 36%. While this was below expectations and only slightly better than the baseline model, it highlights the inherent complexity of the task. The goal of the model is to identify the player exhibiting the most toxic behavior based on the entire conversation within a game, which requires deep context understanding and the ability to generate predictions based on that context. Despite experimenting with the larger **T5-large** model, the results were similar, leading to the decision to use **T5-small** due to its less computationally intensive nature. Additionally, experiments with other models such as **DistillBERT** and **DialoGPT** were unsuccessful, as they could not generate output or follow the prompt as accurately as T5. These findings suggest that, although T5 performed the best among the models tested, the task's complexity—requiring both context comprehension and accurate output generation—poses significant challenges.

Supplementary Results: Several parameters were experimented with in an effort to improve model performance. **Epochs** were increased, but early stopping ensured the model did not train past 3 epochs, indicating potential overfitting or an insufficient learning rate. The **batch size** and **token size** were also adjusted, but these changes resulted in only a marginal improvement of around 1%. Despite these adjustments, the model's performance did not significantly improve, suggesting that the issue may not solely be addressed through hyperparameter tuning. This points to the possibility that the fundamental nature of the task—processing dynamic, context-dependent player interactions—requires more advanced or different approaches beyond fine-tuning these parameters.

Discussion

Although the results are not ideal across all three models, we have learned a lot and identified ways to improve in the future:

Traditional Model: used a very basic rule to identify offenders in individual games: toxic word count. In the future, we could improve this by finding a dataset where toxic words have different weights (e.g., mildly toxic = 1, toxic = 2, extremely toxic = 3…) and using the final toxicity score to identify the offender in each game. This approach could prevent offenders who say only a few but very toxic words from getting away with it.

Sentiment Analysis Model: e weren't able to manipulate the dataset in a way that allowed the model to select the most likely offender from a list of players within each game. As a result, we asked the model to classify players based on their sentiment score and the total words spoken. The model's "accuracy" is higher than the others, but this does not reflect the true goal of the task: identifying the offender. Since only 10% of the players in the dataset are offenders, when we look at the true positives, the score is only 14%.

To improve this model, we should first figure out how to correctly manipulate the dataset so that the model can process it properly. Additionally, we should try to design more features using sentiment analysis by adding more dimensions to the tones, such as arousal (excited-calm) and dominance (powerful-submissive). With more dimensions, we might be able to capture more of the offender's sentiment type. This makes sense because those with the most negative sentiments are usually not the offenders themselves, but offenders can make others experience more negative emotions. Besides adding more dimensions, we could also consider the timing of sentiment changes. Since offenders are often the first aggressor, we could identify which player shows the first sudden negative sentiment change as a feature.

LLM Model: While the results aren't great, we learned that the task may be more difficult than anticipated. From the two previous models, we learned that simply choosing offenders based on negativity or the number of toxic words won't work. Offenders are adept at using techniques like sarcasm, taunting, spamming, and other tactics to bypass these traditional models (because everyone online knows that saying certain words will get them banned). Therefore, when the LLM model was able to use context and logic to output the player's name in each unique game based on reading conversational data with an average of 800 words, 36% accuracy is still quite impressive. Since there are a total of 10 players per game, 36% is significantly better than random guessing.

Although a human moderator would likely perform better than the LLM model as of now, the model was able to predict thousands of games in less than a minute. In the future, we could try a few things to boost the performance of this model. First, we could fine-tune the model on a game-specific toxic word/phrases dataset, so the model has better domain knowledge and can capture more game-specific terminology. Another approach would be to explore other models besides T5, such as BERT, which is an encoder-only model. If we adjust the dataset and labels so that BERT can produce classifications, we might obtain better results because BERT excels at understanding context.

Conclusion

In this project, we explored the use of different models for detecting toxic behavior in multiplayer game chats. We successfully developed and tested three distinct models: a traditional word-based approach, a sentiment analysis model, and a context-aware LLM model. Although the results were not as high as expected, we gained valuable insights into the challenges of identifying toxic behavior. The LLM model, in particular, demonstrated the potential to capture context and logic in player interactions, achieving 36% accuracy, which is notably better than random guessing. Moving forward, there are clear opportunities for refinement, including dataset enhancements, fine-tuning with game-specific terminology, and exploring alternative models like BERT. Despite the challenges, this project laid a strong foundation for future improvements in toxicity detection systems for online gaming communities.

Reference

Main Dataset:
https://www.kaggle.com/datasets/simshengxue/league-of-legends-tribunal-chatlogs

Toxic Dictionary Wordbank
https://github.com/Orthrus-Lexicon/Toxic/blob/main/Toxic%20words%20dictionary.txt

Toxicity Research
https://www.sciencedirect.com/science/article/abs/pii/S0747563220300972