Profanity Detection in Online Club Game Names

By Ankit Gupta

WORD COUNT	7682	TIME SUBMITTED	22-JUL-2022 07:40AM
		PAPER ID	88681486

Profanity Detection in Online Gaming Club Names

By Ankit Gupta

Abstract

The increasing popularity of multiplayer online games along with the fact that active users comprise mostly of teenagers and adults, is making the gaming platforms vulnerable to toxicity, be it in the form of visuals: obscenity via video game streaming, vulgar images as profile pictures; audio: verbal abuse, bullying via voice messages; text: bullying during live chats, vulgarity or racist terms in user or club names. The highly competitive nature of games along with the age group of users make the usage of cuss words in the form of offensive, bullying, racist remarks, referring to pornography quite common. This makes it difficult for users at the receiving end to continue with their accounts and ultimately hampers the revenue of gaming companies. As a result, handling of toxicity has become a crucial problem to solve for such industries. Unlike the traditional profanity detection problems in which there exists a clear distinction between toxic and non-toxic datapoints, this problem has an additional difficulty due to the constraint that a certain degree of profanity needs to be allowed considering the discourteous nature of users' age group. This constraint makes the problem quite difficult even for powerful machine learning algorithms and leaves laborious manual reviewing as the only option available. In this research paper, we focused on handling profanity in text data and attempt to bring down the manual efforts by catching most of the very obvious and possibly some less obvious toxic club names using deep learning models while forwarding for manual review only those club names having reasonable doubt as per the model's score. Using LSTMS, we managed to get a recall of 80% at a false positive rate 0.04.

Contents

Chapter 1

Introduction

1.1 Profanity in online gaming

Based on the estimate of 5.6% annual growth forecast, the number of video game players worldwide will rise from 2.69 billion in 2020 to 3.07 billion in 2023 [21]. The multiplayer online games apart from the providing a common gaming platform, also provides a social environment to users from different parts of the world to interact, chat, discuss gaming tactics or about recent updates in any particular game etc. At the same time, the difference in the race, ethnicity, color, gender and opinions of users often triggers these interactions to turn hostile and become an arena of abusive and vulgar exchanges resulting in the mental harassment of either parties. While some of these unfortunate events may be accidental, there does exists a section of users who harass and bully others just for fun not realising that users at the receiving end may suffer from high anxiety and depression on continued exposure to such toxicity [45].

The highly competitive nature of multiplayer online games [31] along with the fact that majority of online gamers being adults and teenagers [28], results in toxicity creeping into the gaming environments in various forms such as videos, images, audio and text. Of these forms, the one that we studied in this research work, is the text where toxicity appears in chats among users during live gaming sessions and in club names. Toxicity as text is found in words or sentences that are offensive to a specific community, racist, hurting religious sentiments, indicating violence of some nature as in suicide and rape threats, containing pornographic terms, sexually abusive etc. Here our focus is on the problem of toxicity detection in club names. A club name is a name for a gaming team framed by a set of users forming the team.

Although, unlike chats where toxic comments are directly targeted to users on the other side, toxicity and abusiveness in club names, that are vulgar, racist, or offensive to a specific community, indirectly impact all other users who come across it. In other words, club names have higher blast radius (visibility) compared to toxicity in chats. Leaving these offensive club names undetected tarnishes the image of the corresponding gaming company, leading not only to existing users but also newcomers, quitting and thus impacting revenue. Thus, it is crucial for any reputed gaming organisation to keep their environment clean by detecting such offensive club names and taking appropriate actions against the offenders.

Given the enormous amount of data generated per day, labelling the data manually by hand is not possible. At the same time, if we consider the subjective nature of toxicity in texts, and the fact that a certain degree of toxicity needs to be allowed given the roughness in the language of adults, building a machine learning model that can completely replace manual intervention is not a trivial task. Hence, in this thesis we followed an intermediary approach of building a machine learning model that can take the load off the manual reviewers by correctly labelling the very obvious club names and passing on only those names with reasonable doubt for manual review stage.

The remainder of this thesis is structured as follows. Section 1.2 defines the problem statement; In Section 1.3 we discusses various challenges associated with the problem; Chapter 2 reviews related works existing in the field of hate speech detection; In Chapter 3 we talk about the associated dataset, preparing the data for model training and data preprocessing strategies.; Chapter 4 demonstrates the experiments conducted taking different models and the corresponding results; finally, Chapter 5 is a conclusion and discussion about future scope of this work.

1.2 Problem Statement

The club names data that is manually reviewed has a high proportion of non-toxic club names and thus in order to identify the extremely low percentage of toxic club names from the data, a major portion of time is unnecessarily spent in reviewing the non-toxic ones. Our problem is thus to make data for manually review, toxic rich by reducing the percentage of non-toxicity.

We used machine learning techniques to solve the problem which is to design a machine learning model that can be trained on labelled club names data and learn to give a score between 0 and 1 indicative of the probability of a given club name being toxic. These probability scores can then be used to classify club names as toxic and non-toxic by setting appropriate threshold. The performance metrics of the model are chosen to be recall and false positive rate (FPR). While recall is the percentage of toxic (positive) cases correctly detected by the model, FPR on the other hand is the fraction of non-toxic (negative) names incorrectly labelled as toxic. The reason for choosing recall as a performance metric is to detect as much toxic club names as possible and the purpose behind choosing FPR is to have very less number of false positives by the model, so that the time and efforts spent during manual review process could be significantly reduced by enriching the proportion of toxic club names in the data sent for manual reviewing. Thus, a high recall of toxic club names at a very low false positive rate is desired from the trained model.

1.3 Challenges

The problem as discussed in the Section 1.2 belongs to the category of binary text classification problem with the two classes, toxic and non-toxic. Although this is a very commonly encountered problem in machine learning field, the data of club names associated with the problem, entails several complications which turns it into a not-so-trivial problem. In this section, we discuss many challenges due to the data that makes it difficult to apply any of the well-known existing solutions onto this problem.

1.3.1 High imbalance in data

Although, size of data in terms of number of club names is large, the percentage of club names that are actually labelled as offensive and liable to be taken action against, after manually review is extremely less compared to the non-offensive names. The club names data we used to train our model had only 0.67% of club names labelled as toxic.

1.3.2 Nature of data

Firstly, unlike the chats data where each record consists of several sentences exchanged among the users as long as gaming session continues, the club names which are names that a group of users set as their team or club name, are quite short-length ranging between $1-5$ words or between $3-30$ characters as a whole. This leaves very less room for the deep learning models to develop contextual understanding from the data, which is one of the strengths of *BERT* (Bidirectional Encoder Representations from Transformers) [16] models.

Secondly, with the advent of increasing smartness among users in the sense of beating the automated models in detecting their toxic club names, these users come up with different tricks in framing club names that include: 1. usage of leetcodes, 2. digits as alphabets, 3. abbreviations, 4. offensive numeric codes, 5. special symbols, 6. punctuation symbols for alphabets or digits, 7. unnecessary repeated characters etc.

The roughness of texts in club names to the extent as described above makes it nearly impossible to design an effective tokenizer that can generalise well over the entire data.

1.3.3 Diversity in the data

Unlike, chats where the text is composed of sentences that are meaningful even though the individual words may be incorrectly spelled, in case of club names, users get the liberty to frame anything they like, be it digits between letters, multiple words without spaces, random casing, punctuation symbols, abbreviations and numeric codes with hidden meaning, digits and symbols resembling alphabets, repeated characters and even chunk of characters being placed in the beginning, middle and end, that does not have any sense of its own. This leads to umpteen different forms of appearance for one particular word and thus accounts for vast variety in the data.

1.3.4 Context

Context 7 plays a significant role in determining the toxicity of a club name. A club name cannot be termed as toxic by the mere presence of toxic words in it, unless, it is seriously insulting or posing a threat to others. For instance, a club name can contain self-humiliating remarks, e.g. "*WeAreMorons*". Apart from the meaning that is seemingly implied by the club name, one also needs to consider the nature of the respective game for which the club name has been set. A club name that depicts violence or some kind of a threat in the general sense may be considered completely normal and accepting when the concerned game is taken into context, reason being the violent terms present in the club name, are quite commonly used in the game. This factor sometimes makes it difficult even for human annotators to interpret and decide on the toxicity of some club names.

1.3.5 Subjective Labelling

Two 16 of the many factors that attribute to the success of any machine learning classifier are 1. size of data and 2. quality of data. For popular gaming organisations, scarcity of data is never a problem given the huge active participation of users from all over the world. The real issue is the later one i.e. quality of labelled data. Since, labelling the data is done manually, consistency in labelling becomes an important factor in determining the data's quality when multiple persons are involved in data labelling. In spite of a well framed policy on determining toxicity in club names, there can exist club names, for which it becomes quite subjective to decide if it fulfils the criteria of being labelled as toxic.

Chapter 2

Related Work

Besides gaming platforms, there exists abundant availability of online discussion forums where users virtually interact by sharing and exchanging their views, e.g. blog posts, comments on social media channels. Depending on the sensitivity of the topic, these online conversations often turns inappropriate and becomes the battleground of verbal abuses, personal attacks etc. resulting in degraded user experiences. Thus, toxicity detection and handling of hate speeches have been a wide area of research in order to safeguard the user experiences and thus the revenue from getting hampered.

In the last few years, a large number of researches has been conducted in the field of toxicity detection in online content. A combination of natural language processing (*NLP*) techniques and traditional machine learning (*ML*) models have been used to detect hate speeches in social media. These traditional ML techniques require features to be manually prepared along with statistical algorithms to build classifiers. Effective feature extraction and engineering are critical to the performance of any classification algorithm. Some of the common features that can be considered for hate speech detection are bag of words, n-grams [19, 25, 14, 41], TF-IDF, sentiment analysis, important linguistic features, knowledge-based features and meta-information of hate speech [8].

Along with the techniques mentioned above, different conventional ML algorithms can be employed to solve hate speech classification problems. Such ML algorithms can be Logistic Regression [3, 5, 15, 39, 40], Naïve Bayes [3], Support Vector Machines (*SVM*) [40, 3, 15] and Random Forest [24].

With the advent of deep learning, people are since shifting from using conventional ML models to deep learning architectures to exploit the automatic extraction of multi-layered features from the data, offered by the deep learning models. Two very popular deep learning models which have most commonly been used for toxicity classification are Convolutional Neural Network (*CNN*) [32] and Recurrent Neural Network (*RNN*) [37] models.

As one of the starting works on this field using deep learning models, Djuric et al.

[17] suggested a two-step approach which involves learning a low dimensional text embedding by using a *CBOW* model in order to extract paragraph2vec [26] embeddings and then using a binary classifier which is trained along with the embeddings to detect toxicity. Badjatiya et al. [7] used word embeddings for feature representations of tweets. They experimented with three deep learning models, *LSTM*, FastText and *CNN*, where for each of these models, either *GloVe* [35] or random word embeddings were used.

Park et al. [33] applied *CNN* models to detect abusive languages in English Twitter corpus of $20K$ tweets belonging to sexist and racist classes. They followed two approaches namely two-step approach which involves detecting abusive language first and then classifying it into its appropriate type; and one-step approach involving one multi-class classification. Based on the input features being characters, words or both, they designed three different types of *CNN*-based models: *CharCNN* [9], *WordCNN* [43], and *HybridCNN*. The third model i.e. *HybridCNN*, designed by them was found to give the best performance. Gambäck et al. [20] trained four different *CNN* models respectively on character 4-grams, word vectors using word2vec [29], word vectors randomly generated and combination of word vectors and character 4-grams.

Tundis et al. [38] applied optical character recognition(*OCR*) technology using *CNN* to detect hidden propaganda in mixed coded texts. The text is first split into words from which mixed coded words are identified and then for each such mixed coded word different subsets are created, and then for each such subset, the symbols are converted to images followed by replacing the symbols with the nearest alphabet character, the symbols resemble, using a pretrained *CNN* model. After all symbols are converted to alphabets, the resulting words are checked for its existence in dictionary and removed if not found. Finally, among the resulting words, the invalid words having no existence in dictionary are filtered out and the remaining valid words are combined to check if the resulting sentence is meaningful and of the multiple meaningful sentences in the end, the maximum toxicity score of all sentences is recorded as the final score for the original mixed coded input.

Besides *CNN* models, a number of solutions for toxicity detection have employed sequential models that is *RNN*. Pitsilis et al. [36] detected hate speech in tweets depending on the user's history of past tweets and the given tweet itself using an ensemble of Long short term memory networks *LSTM* [23], a special kind of *RNN* model.

There also exist deep learning models that are combination of *CNN* and *RNN* architectures as Zhang et al. [44] introduced *CNN* + *LSTM* model for toxic speech detection in tweets. Yenala et al. [42] applied deep learning models to detect and filter inappropriate query suggestions in search engines and toxic conversations by users in messengers. For detecting inappropriate query suggestions, they developed a novel deep learning architecture Convolutional Bi-Directional LSTM (*C-BiLSTM*) which is a combination of *CNN* and Bi-Directional LSTM (*BiLSTM*) [18]. For detecting toxic conversations, they used a combination of *LSTM* and *BiLSTM* networks.

Lees et al. [27] from Google Jigsaw gave deep learning solutions to the "Toxic Comment Classification Challenge" problem launched on Kaggle [2]. The problem was to predict toxicity of a given comment and dataset comprising comments from Wikipedia's talk page edits. They fine-tuned a *BERT* model pretrained on comment domain that achieved scores beyond 90%.

Mozafari et al. [30] applied transfer learning approach by using a pretrained *BERT* model trained on English Wikipedia and BookCorpus for toxicity detection in tweets data. They used many fine-tuning techniques including the *BERT*-based fine-tuning, inserting non-linear layers, inserting *BiLSTM* layer and inserting a *CNN* layer.

Although majority of the works in toxicity detection exists for English language, there exists abundant literature on hate speech problems on other languages as well. Alshalan et al. [4] applied deep learning approaches to detect toxicity in Arabic tweets. They evaluated four different models: gated recurrent units (*GRU*) [10], *CNN*, *CNN* + *GRU*, and *BERT*. Of the four models, *CNN* was found to give best performance. Vigna et al. [15] used *SVM* [13] and (*LSTM*) models to detect toxicity in manually annotated Italian comments on Facebook. They realised that *LSTM*s, capable of capturing long term dependencies in texts very well, outperform traditional classifiers like *SVM*. Ishmam1 et al. [24] evaluated Random Forest and *GRU* models for detecting toxic speeches in public Facebook pages in Bengali language.

Profanity detection methods are also tightly bound to the underlying the language of the corpus and thus different models are designed to solve the same problem of profanity detection in texts but over different languages. According to Corazza et al. [12], who evaluated and compared multiple *RNN* models across different languages which include English, Italian and German, although the *RNN* models generally performed quite well for all these languages but the best performing is found to be different for different languages. Further, Aluru et al. [5] evaluated various conventional ML models and deep learning models on nine languages: English, Spanish, Italian, French, German, Arabic, Indonesian, Polish and Portuguese. Their findings showed that it is the specific language that determines the best performing model.

Of all the challenges encountered in toxicity detection, the major challenge faced is the generalizability of the models. Models that are best performing when trained and tested for hate speech classification on one corpus may become an average performer for the same problem on other corpora [6, 22, 34]. Although the solutions discussed so far are powerful and well suited for their respective domains and datasets, none of them proved to be significantly helpful when trained and tested on the club names data, the reason being the corpus of club names differ significantly from that of texts, comments, tweets or search engines queries. The challenges as mentioned in Section 1.2 make the results even worse.

Chapter 3

The Dataset

3.1 Introduction

The data consisted of club names from a popular shooting game having users from all over the world. After every two or three months of time, the existing club names are collected and sent for review to a third party company (*TPC*). Based on the review results, some non-toxic club names are closed (labelled as non-toxic). The rest of the names are then sent for manual review process. This manual review process is carried out in two stages namely *T1* and *T2*. Names for which labels are not decided in stage *T1* are then finally decided in stage *T2*. The entire review process is termed as one wave. Till date, a total of 10 waves have been conducted.

Results of each wave were available in the form of a single spreadsheet containing the labels by both *TPC* and combined labels of *T1* and *T2*. Based on the labels, it is found that the labels by *TPC* were very oversensitive in the sense that more than 90% of the club names labelled by *TPC* as toxic were found to be non-toxic when reviewed manually by *T1* and *T2*. Thus, the labels by *TPC* could not be relied upon for training any model as the final decision will be of the manual reviewers only. *TPC* results were helpful only to minimise the number of non-toxic club names getting forwarded for manual review, thus relieving the burden of laborious manual review process but only by a small extent of up to 10%.

3.2 Data Preparation

Data from all the waves from wave 1 to 10 are merged into one single dataframe. Few rows with club names field blank, were detected and cleaned. Since subsequent waves will have many names from the previous waves, there were many duplicate club names found in the final combined data. These duplicates were removed by considering the label of the most recent wave, the corresponding club names belonged to. Since our focus was mainly on the English language club names, we filtered out all the foreign language names from the data which contributed only up to 3% of the whole data.

3.3 Data Description

The final merged data of ten waves with foreign language names removed contained around 5.5 million unique club names. Club names labelled as toxic contributed to mere 0.67% of the entire data. Thus, for every 10K non-toxic club names, there were only 67 names labelled as toxic, a case of extremely skewed class labels.

Although the data had a total of 15 columns, most of these columns were provided by *TPC* which were not of much importance and thus were not considered for training models. In fact, none of *TPC* results was used for training. These results were used only during model evaluation to gauge the effectiveness of the model over *TPC* and whether the model can replace *TPC* altogether.

3.4 Data Preprocessing

The data preprocessing strategy employed, was different for different models we experimented with. For *BERT* and Transformer models, no preprocessing was done on the club names. The club names were just fed to these model's respective tokenizer and the resulting tokenized data is used for training.

For pure sequential models, the club names were tokenized into bigrams. The idea behind using bigrams and not going for higher length grams was that, because club names were short lengthed and highly different from each other, higher lengthed grams will have less frequencies compared to bigrams, making the resulting trained models more prone to overfitting and thus having poor generalisation over unseen data, .

Initially we only used these tokenized names for model training and testing. But later, after going through the club names and analysing them, we designed a preprocessing pipeline with a series of stages based on patterns identified during the analysis. The stages of the pipeline discussed below are in order of its execution during preprocessing.

- removal of symbols: a large number of toxic club names were found to contain punctuation symbols with an intention to add noise in order to beat the models from getting detected. Such symbols were replaced by spaces.
- removal of extra spaces at boundaries: this is done to clean the club names of any extra spaces left out after removal of symbols in the previous stage.
- remove extra spaces in between: some toxic club names had the individual letters separated by multiple spaces just to avoid getting caught.

- removal of pin codes: around 15 K club names were found to contain pin codes. These pin codes do not serve any purpose in toxicity detection, hence removed.
- replacing 3 or more repetitions of alphabets to 2: unnecessary repetitions of characters is quite natural in club names like these. These repetitions will only serve as noise contaminating the resulting bigrams formed after tokenization. Such repetitions were reduced to 2 repetitions
- removal of stopwords: We curated a list of words that we believe would not serve any purpose in distinguishing between toxic and non-toxic club names. In this stage, such words if found in the club name will be removed.
- removal of digits with no meaningful context
- normalisation of mixed coded text: this is the most powerful stage of the entire preprocessing pipeline. This is an attempt to handle the mixed coded text in club names. We first implemented a spell correction module to correct all misspelled words to its nearest correct word by two edit distances. For this, we started with a small database of commonly used toxic terms created after analysing the club names. But later, we realised it was causing more damage than good by wrongly correcting many misspelled non-toxic club names to toxic terms. We later resorted to using regular expressions in order to handle not only misspelled words but also the digits, symbols, abbreviations etc. We finally got rid of the spell correction module and resorted completely to regular expressions through which we had more control over the raw club names data.

The club names are first passed through the preprocessing pipeline and then tokenized into bigrams. After preprocessing, 70% of club names are then used for model training while the remaining 30% is reserved for model evaluation.

Chapter 4

Experiments and Results

4.1 Training and Testing

We used Binary Cross Entropy Loss (*BCE*) as the loss function during training. But during the initial stages of our experiments with different models, we found that due to imbalance in the class labels being extreme, the models ignored the toxic club names completely during training by simply labelling all the club names as non-toxic to get high accuracy. In order to attract the model's attention towards the toxic club names, we used weighted *BCE* loss function in which for any mislabelled toxic club name by the model, we penalized the model by some weight greater than 1. Different values of weight ranging from 10 to 10000 were tried but to no effect.

We then defined a custom loss function which was probabilistic version of $\beta-$F1 Loss. Even with different values of β, the models continued to ignore the toxic cases.

Finally, we solved the problem, by taking random samples from both toxic and non-toxic classes to form a balanced data for every epoch, instead of training models on the entire data. That is, before each epoch, we took a randomly sampled subset from toxic club names and from non-toxic club names, we sampled k ($k > 1$) times the size of the toxic sample. The value of k once chosen remained fixed throughout the training process.

4.2 Models

4.2.1 chatBert: BERT model pretrained on chats data

To start with, we used *BERT* model pretrained on toxic comments of Wikipedia [27] and trained it on chats data of same game. The model was trained perfectly, giving acceptable results on test data. We then trained this pretrained model on the club names data. Even after rigorous model hyperparameter tuning, and trying both relu and sigmoid as the activation functions, the model gave poor results. (see Table 4.1)

One of the reasons possible for the model's inability to fit well to the club names data is the model's inbuilt tokenizer. This tokenizer, designed keeping the chats data in mind, may have proved ineffective for the club names data. The tokenization as mentioned in Section 1.3.2, is a major challenge for club names data.

The training time was very slow. It took more than 12 hours to train just 10 epochs of training on high powered GPU processor, Tesla V100. The slow training time left very less room for experimentation with the model.

4.2.2 Canine: pretrained transformer model

Canine [11] is a pretrained transformer model offered by Hugging Face [1] developers. The speciality of the model and the reason we went for this model is that there was no explicit tokenization step employed by this model. It just tokenizes the entire data at character level and converts each character to its Unicode representation. The unicode conversion was also a motivating factor for choosing this model as it can be used to fit to foreign language names as well, provided it fits to the English language names.

However, during training, the model turned out to be a poor fit to club names data. The model's architecture contains a deep transformer stack comprising of 12 Transformer layers stacked in series. Even after rigorous hyperparameter tuning with the model, including unlocking the model up to 5 transformer layers of the deep transformer stack component, the model gave poor results only.

One of the reasons we hypothesise for the model's inability to fit well to the club names data is the difference of the club names data from the corpus on which the model was pretrained. Because the texts are broken to characters, the dependence between the corpus and the initial embedding layers is huge, causing the pretrained embedding layers to poorly represent the input club names and transmitting almost noise to the subsequent layers.

The training time was very slow. It took more than 15 hours to train just 10 epochs of training. The slow training time left very less room for experimentation with the model.

4.2.3 Sequential Models

Getting failures with advanced *BERT* and Transformer models, we went down to try comparatively simpler models, the sequential models. We believed that the club names being very short lengthed (mostly ranging between one to five words), there is not much room for any contextual learning whereas the advanced models such as

BERT and transformer models are specially used for getting contextual understanding of the data. Further, if any pretrained model is used, then it will have its own tokenizer which may not be suitable for the club names. On the other hand, if we go for building our own tokenizer then we have to train such models from scratch which is inefficient with respect to both time and cost.

Henceforth, for each sequential models, the club names were first tokenized into bi-grams and then each club name was converted to a sequence of 3000 dimensional one-hot coded representation based on each individual bigram. Finally, using padding and truncation, these sequences of different lengths were each converted to the same length of 15 as 95% of the sequences were found to be less than 15.

Also, for all the sequential models, we used self-learning embedding by adding an embedding layer of dimension 300. The decision about the dimension is taken after experimenting with different embedding dimensions ranging from 150 to 600.

Recurrent Neural Networks (*RNN*)

We experimented with several architectures of *RNN*s using hyperparameter tuning during training and found the best performing *RNN* model to have embedding size 50, hidden layer dimension of 32 and followed by two feedforward layers of dimension 32 and 8 respectively. Dropout layers with dropout value 0.1 are added after each layer. Activation functions are chosen to be *relu* and *sigmoid* respectively after *RNN* and the final feedforward layer. Training was carried out for 600 epochs using Stochastic Gradient Descent *(SGD)* optimizer with a learning rate of 0.01. The purpose for choosing *SGD* as the optimizer during training was that all models were found to converge better compared to other optimizers we experimented with.

The results were much better than the previous models. For the first time, the underfitting problem we had been experiencing with previous models got resolved. But still, *RNN*s had overfitting issues. The test error though way lesser than the previous models was not up to the mark.

Long Short Term Memory Networks (*LSTM*s)

Although the club names were short lengthed, but when tokenized into bigrams, the length increases by a considerable extent. *LSTM*s are better than *RNN*s in processing longer length sequences.

We experimented with different versions of *LSTM*s and hypertuned each of them. The models along with their architecture for which we found best results are listed below.

- *LSTM*: embedding size 300, number of neurons in hidden layer 64, two feedforward layers of sizes 64 and 32 respectively.

- *BiLSTM*: embedding size 200, number of neurons in hidden layer 64, one feed-forward layer of size 64.
- Multi-layered *LSTM*s stacked in series
  - Layer 2 *LSTM* (*LSTM-L2*): embedding size 300, number of neurons in hidden layer 64, two feedforward layers of sizes 64 and 32 respectively.
  - Layer 3 *LSTM* (*LSTM-L3*): embedding size 300, number of neurons in hidden layer 64, two feedforward layers of sizes 64 and 32 respectively.
  - Layer 4 *LSTM* (*LSTM-L4*): embedding size 300, number of neurons in hidden layer 64, two feedforward layers of sizes 64 and 32 respectively.

For each of the above *LSTM*s, optimizer *SGD* is used with learning rate of 0.001 for the same reason as for *RNN*s.

The best part about sequential models was that the training process was very fast. It only took a maximum of 2 hours to training 1000 epochs of the data. This was a huge advantage for us as it gave us the freedom to experiment with different architectures such as the number of *LSTM* layers, feed forward layers, dropout values, choice of activation functions, optimizers, learning rate etc.

Sequential models on preprocessed data

The preprocessing pipeline (see Section 3.4) was developed only after getting satisfactory results with *LSTM* models. The idea behind developing preprocessing pipeline was to further improve the results of these models by having cleaner bigrams after tokenization.

So we preprocessed the data through the pipeline, and retrained the best performing *LSTM*s on it. The results though improved only by $1 - 2\,\%$, but the preprocessing pipeline is not the final one, will continue to be part of the model's maintenance in the long run and will adjust the model dynamics by accounting for the changing trends in the newer club names.

4.3 Results

Results of all models we experimented with, are compiled into two tables as shown below. In Table 4.1, we list the results without the preprocessing pipeline (see Section 3.4) being applied in the data preprocessing stage. Table 4.2 shows the results after the preprocessing pipeline was designed and applied on the data before model training. Since we did not see any significant results with *BERT* and Transformer models as shown in Table 4.1, we did not apply any preprocessing with these models and thus only the sequential models results are listed in Table 4.2. Although, the performances are shown in terms of recall, *FPR* and precision, precision is not a major concern given the huge proportion of non-toxic club names compared to that of toxic club names.

Model Type	**Model**	**Recall**	**FPR**	**Precision**
BERT	chatBert	30%	0.17	11%
	Canine	41%	0.13	7%
Sequential	RNN	64%	0.05	24%
	LSTM	78%	0.05	27%
	BiLSTM	80%	0.05	27%
	LSTM-L2	82%	0.05	33%
	LSTM-L3	81%	0.04	31%
	LSTM-L4	82%	0.03	38%

Table 4.1: Results without preprocessing pipeline

Model	**Recall**	**FPR**	**Precision**
RNN	69%	0.05	29%
LSTM	80%	0.05	31%
BiLSTM	81%	0.05	32%
LSTM-L2	82%	0.04	38%
LSTM-L3	83%	0.04	35%
LSTM-L4	84%	0.035	42%

Table 4.2: Results of sequential models with preprocessing pipeline

In Table 4.1, it can be seen that the sequential models fitted to the club names much better compared to the pretrained *BERT* and transformer models. Both types of *BERT* models, chatBERT and Canine have recall only near 40% with high *FPR* of 17% and 13% respectively, whereas each of the sequential models have *FPR* lower than 5% with high recall percentage reaching beyond 80s. Of all the sequential models tried, the best performing model is found to be *LSTM-L4* having 82% recall with *FPR* as low as 3%.

Further, in Table 4.2, it can be seen that the preprocessing pipeline improved the performance of sequential models. For each sequential model, the recall went up by $1-2$ % at nearly the same *FPR*. The recall of the best performing model *LSTM-L4* went up from 82% to 84% at the cost of just 0.5% increase in *FPR*. Although the performance improvement due to the preprocessing pipeline may not seem high at this stage but with more and more data from the subsequent waves in future, along with the respective upgradation of the pipeline, we expect these margins of improvement to go higher.

Chapter 5

Future Work and Conclusion

We experimented with deep learning models of different architectures ranging from *BERT* to deep transformers to finally the sequential models i.e. the *RNN*s, *LSTM*s and *BiLSTM*s and realized that the sequential models outperformed the former ones, possibly because of the nature of the club names data.

We managed to tackle the extreme imbalancedness of the class labels in the data, by training the models on taking balanced number random samples from both class labels at every epoch. Finally, we handled the roughness of text in club names by designing a preprocessing pipeline in which we defined various methods to clean the club names of noisy texts that are of no help in classification.

We continue to work with *LSTM* architecture models and work on the preprocessing pipeline to further improve it. The impact of the pipeline though seems low at this stage but we believe that it is going to come handy in the long run in the sense that it can be adjusted to cater to the changing trends in the club names in future. Further, considering the fact that different games though differing in its respective gaming vocabularies, but don't differ significantly enough when it comes to the usage of toxicity; we aim to analyse the generalizability of these models on club names of other games. Lastly, after having developed these models on English language names to an acceptable performance, we will proceed towards handling toxicity in foreign language names as well.

Bibliography

[1] Hugging face. Available online: `https://huggingface.co/`

[2] kaggle/toxic comment classification challenge. Available online: `https://www.kaggle.com/c/jigsaw-toxic-comment-classificationchallenge`

[3] Alfina, I., Mulia, R., Fanany, M.I., Ekanata, Y.: Hate speech detection in the indonesian language: A dataset and preliminary study. In: 2017 International Conference on Advanced Computer Science and Information Systems (ICACSIS). pp. 233–238. IEEE (2017)

[4] Alshalan, R., Al-Khalifa, H.: A deep learning approach for automatic hate speech detection in the saudi twittersphere. Applied Sciences 10(23), 8614 (2020)

[5] Aluru, S.S., Mathew, B., Saha, P., Mukherjee, A.: Deep learning models for multilingual hate speech detection. arXiv preprint arXiv:2004.06465 (2020)

[6] Arango, A., Pérez, J., Poblete, B.: Hate speech detection is not as easy as you may think: A closer look at model validation (extended version). Information Systems p. 101584 (2020)

[7] Badjatiya, P., Gupta, S., Gupta, M., Varma, V.: Deep learning for hate speech detection in tweets. In: Proceedings of the 26th international conference on World Wide Web companion. pp. 759–760 (2017)

[8] Biere, S., Bhulai, S., Analytics, M.B.: Hate speech detection using natural language processing techniques. Master Business AnalyticsDepartment of Mathematics Faculty of Science (2018)

[9] Chen, Y.: Convolutional neural network for sentence classification. Master's thesis, University of Waterloo (2015)

[10] Cho, K., Van Merriënboer, B., Bahdanau, D., Bengio, Y.: On the properties of neural machine translation: Encoder-decoder approaches. arXiv preprint arXiv:1409.1259 (2014)

[11] Clark, J.H., Garrette, D., Turc, I., Wieting, J.: Canine: Pre-training an efficient tokenization-free encoder for language representation. Transactions of the Association for Computational Linguistics 10, 73–91 (2022)

[12] Corazza, M., Menini, S., Cabrio, E., Tonelli, S., Villata, S.: A multilingual evaluation for online hate speech detection. ACM Transactions on Internet Technology (TOIT) 20(2), 1–22 (2020)

[13] Cristianini, N., Ricci, E.: Support Vector Machines, pp. 928–932. Springer US, Boston, MA (2008), `https://doi.org/10.1007/978-0-387-30162-4_415`

[14] Davidson, T., Warmsley, D., Macy, M., Weber, I.: Automated hate speech detection and the problem of offensive language. In: Proceedings of the international AAAI conference on web and social media. vol. 11, pp. 512–515 (2017)

[15] Del Vigna12, F., Cimino23, A., Dell'Orletta, F., Petrocchi, M., Tesconi, M.: Hate me, hate me not: Hate speech detection on facebook. In: Proceedings of the First Italian Conference on Cybersecurity (ITASEC17). pp. 86–95 (2017)

[16] Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018)

[17] Djuric, N., Zhou, J., Morris, R., Grbovic, M., Radosavljevic, V., Bhamidipati, N.: Hate speech detection with comment embeddings. In: Proceedings of the 24th international conference on world wide web. pp. 29–30 (2015)

[18] Fei, H., Tan, F.: Bidirectional grid long short-term memory (bigridlstm): A method to address context-sensitivity and vanishing gradient. Algorithms 11(11), 172 (2018)

[19] Fortuna, P., Nunes, S.: A survey on automatic detection of hate speech in text. ACM Computing Surveys (CSUR) 51(4), 1–30 (2018)

[20] Gambäck, B., Sikdar, U.K.: Using convolutional neural networks to classify hate-speech. In: Proceedings of the first workshop on abusive language online. pp. 85–90 (2017)

[21] Gilbert, N.: Number of Gamers Worldwide 2022/2023: Demographics, Statistics, and Predictions. Available online: `https://financesonline.com/number-of-gamers-worldwide/`

[22] Gröndahl, T., Pajola, L., Juuti, M., Conti, M., Asokan, N.: All you need is" love" evading hate speech detection. In: Proceedings of the 11th ACM workshop on artificial intelligence and security. pp. 2–12 (2018)

[23] Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation 9(8), 1735–1780 (1997)

[24] Ishmam, A.M., Sharmin, S.: Hateful speech detection in public facebook pages for the bengali language. In: 2019 18th IEEE international conference on machine learning and applications (ICMLA). pp. 555–560. IEEE (2019)

[25] Jaki, S., De Smedt, T.: Right-wing german hate speech on twitter: Analysis and automatic detection. arXiv preprint arXiv:1910.07518 (2019)

[26] Le, Q., Mikolov, T.: Distributed representations of sentences and documents. In: International conference on machine learning. pp. 1188–1196. PMLR (2014)

[27] Lees, A., Sorensen, J., Kivlichan, I.: Jigsaw@ ami and haspeede2: Fine-tuning a pre-trained comment-domain bert model. In: EVALITA (2020)

[28] Lenhart, A., Jones, S., Macgill, A.: Adults and video games (2008)

[29] Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781 (2013)

[30] Mozafari, M., Farahbakhsh, R., Crespi, N.: A bert-based transfer learning approach for hate speech detection in online social media. In: International Conference on Complex Networks and Their Applications. pp. 928–940. Springer (2019)

[31] of Nerds, T.G.: Increasing competition in online multiplaye games. Available online: `https://thegameofnerds.com/2022/02/22/increasing-competition-in-online-multplayer-games/` (2022)

[32] O'Shea, K., Nash, R.: An introduction to convolutional neural networks. arXiv preprint arXiv:1511.08458 (2015)

[33] Park, J.H., Fung, P.: One-step and two-step classification for abusive language detection on twitter. arXiv preprint arXiv:1706.01206 (2017)

[34] Park, J.H., Shin, J., Fung, P.: Reducing gender bias in abusive language detection. arXiv preprint arXiv:1808.07231 (2018)

[35] Pennington, J., Socher, R., Manning, C.D.: Glove: Global vectors for word representation. In: Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP). pp. 1532–1543 (2014)

[36] Pitsilis, G.K., Ramampiaro, H., Langseth, H.: Effective hate-speech detection in twitter data using recurrent neural networks. Applied Intelligence 48(12), 4730–4742 (2018)

[37] Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning internal representations by error propagation. Tech. rep., California Univ San Diego La Jolla Inst for Cognitive Science (1985)

[38] Tundis, A., Mukherjee, G., Mühlhäuser, M.: An algorithm for the detection of hidden propaganda in mixed-code text over the internet. Applied Sciences 11(5), 2196 (2021)

[39] Unsvåg, E.F., Gambäck, B.: The effects of user features on twitter hate speech detection. In: Proceedings of the 2nd workshop on abusive language online (ALW2). pp. 75–85 (2018)

[40] Vijayaraghavan, P., Larochelle, H., Roy, D.: Interpretable multi-modal hate speech detection. arXiv preprint arXiv:2103.01616 (2021)

[41] Waseem, Z., Hovy, D.: Hateful symbols or hateful people? predictive features for hate speech detection on twitter. In: Proceedings of the NAACL student research workshop. pp. 88–93 (2016)

[42] Yenala, H., Jhanwar, A., Chinnakotla, M.K., Goyal, J.: Deep learning for detecting inappropriate content in text. International Journal of Data Science and Analytics 6(4), 273–286 (2018)

[43] Zhang, X., Zhao, J., LeCun, Y.: Character-level convolutional networks for text classification. Advances in neural information processing systems 28 (2015)

[44] Zhang, Z., Robinson, D., Tepper, J.: Detecting hate speech on twitter using a convolution-gru based deep neural network. In: European semantic web conference. pp. 745–760. Springer (2018)

[45] Zsila, Á., Shabahang, R., Aruguete, M.S., Orosz, G.: Toxic behaviors in online multiplayer games: Prevalence, perception, risk factors of victimization, and psychological consequences. Aggressive Behavior 48(3), 356–364 (2022)

Profanity Detection in Online Club Game Names

ORIGINALITY REPORT

5%
SIMILARITY INDEX

PRIMARY SOURCES

1	"Advances in Knowledge Discovery and Data Mining", Springer Science and Business Media LLC, 2017 Crossref	36 words — 1%
2	www.mdpi.com Internet	34 words — 1%
3	deepai.org Internet	26 words — < 1%
4	link.springer.com Internet	21 words — < 1%
5	www.slideshare.net Internet	18 words — < 1%
6	dokumen.pub Internet	17 words — < 1%
7	"Deep Learning-Based Approaches for Sentiment Analysis", Springer Science and Business Media LLC, 2020 Crossref	15 words — < 1%
8	www.webology.org Internet	12 words — < 1%

9	Lecture Notes in Computer Science, 2015. Crossref	11 words — < 1%
10	"Neural Information Processing", Springer Science and Business Media LLC, 2017 Crossref	10 words — < 1%
11	Boyu Qiu, Yanrong Chen, Xu He, Ting Liu, Sixian Wang, Wei Zhang. "Short-Term Touch-Screen Video Game Playing Improves the Inhibition Ability", International Journal of Environmental Research and Public Health, 2021 Crossref	10 words — < 1%
12	Prashant Kapil, Asif Ekbal. "A deep neural network based multi-task learning approach to hate speech detection", Knowledge-Based Systems, 2020 Crossref	10 words — < 1%
13	ebin.pub Internet	10 words — < 1%
14	www.stat.ucla.edu Internet	10 words — < 1%
15	"AI*IA 2018 – Advances in Artificial Intelligence", Springer Nature America, Inc, 2018 Crossref	9 words — < 1%
16	"Neural Information Processing", Springer Nature, 2016 Crossref	9 words — < 1%
17	"Proceedings of the Future Technologies Conference (FTC) 2020, Volume 1", Springer Science and Business Media LLC, 2021 Crossref	9 words — < 1%

18	Harish Yenala, Ashish Jhanwar, Manoj K. Chinnakotla, Jay Goyal. "Deep learning for detecting inappropriate content in text", International Journal of Data Science and Analytics, 2017 Crossref	9 words — < 1%
19	"Advances in Computational Intelligence", Springer Science and Business Media LLC, 2019 Crossref	8 words — < 1%
20	"Advances in Information Retrieval", Springer Science and Business Media LLC, 2019 Crossref	8 words — < 1%
21	"Advances in Information Retrieval", Springer Science and Business Media LLC, 2020 Crossref	8 words — < 1%
22	"Data Management, Analytics and Innovation", Springer Science and Business Media LLC, 2020 Crossref	8 words — < 1%
23	Jan Kocoń, Alicja Figas, Marcin Gruza, Daria Puchalska, Tomasz Kajdanowicz, Przemysław Kazienko. "Offensive, aggressive, and hate speech analysis: From data-centric to human-centered approach", Information Processing & Management, 2021 Crossref	8 words — < 1%
24	Michele Corazza, Stefano Menini, Elena Cabrio, Sara Tonelli, Serena Villata. "A Multilingual Evaluation for Online Hate Speech Detection", ACM Transactions on Internet Technology, 2020 Crossref	8 words — < 1%
25	ucilnica2021.fri.uni-lj.si Internet	8 words — < 1%

26	"Digital Libraries at the Crossroads of Digital Information for the Future", Springer Science and Business Media LLC, 2019 Crossref	7 words — < 1%
27	"Intelligent Systems and Applications", Springer Science and Business Media LLC, 2019 Crossref	6 words — < 1%
28	"Machine Learning and Knowledge Discovery in Databases. Applied Data Science Track", Springer Science and Business Media LLC, 2021 Crossref	6 words — < 1%
29	"Natural Language Understanding and Intelligent Applications", Springer Nature, 2016 Crossref	6 words — < 1%
30	"Statistical Language and Speech Processing", Springer Science and Business Media LLC, 2019 Crossref	6 words — < 1%

EXCLUDE QUOTES	ON	EXCLUDE SOURCES	OFF
EXCLUDE BIBLIOGRAPHY	ON	EXCLUDE MATCHES	OFF