Toxicity Detection in Multiplayer Online Games

Märtens, Marcus; Shen, Siqi; Iosup, Alex; Kuipers, Fernando

DOI
10.1109/NetGames.2015.7382991

Publication date
2015

Document Version
Accepted author manuscript

Published in
2015 International Workshop on Network and Systems Support for Games (NetGames)

Citation (APA)

Important note
To cite this publication, please use the final published version (if applicable). Please check the document version above.
Toxicity Detection in Multiplayer Online Games

Marcus Mårtens*, Siqi Shen†, Alexandru Iosup‡ and Fernando Kuipers*

*Network Architectures and Services Group, Delft University of Technology, Delft, The Netherlands
†Parallel and Distributed Processing Laboratory, National University of Defense Technology, China
‡College of Computer, National University of Defense Technology, China

Email: {m.maertens, f.a.kuipers, a.iosup}@tudelft.nl, shensiqi@nudt.edu.cn

Abstract—Social interactions in multiplayer online games are an essential feature for a growing number of players world-wide. However, this interaction between the players might lead to the emergence of undesired and unintended behavior, particularly if the game is designed to be highly competitive. Communication channels might be abused to harass and verbally assault other players, which negates the very purpose of entertainment games by creating a toxic player-community. By using a novel natural language processing framework, we detect profanity in chat-logs of a popular Multiplayer Online Battle Arena (MOBA) game and develop a method to classify toxic remarks. We show how toxicity is non-trivially linked to game success.

I. INTRODUCTION

Multiplayer Online Battle Arena (MOBA) games have been growing increasingly popular and captivate their player base in virtue of complex game mechanics and competitive nature. Riot’s League of Legends claims to have over 67M monthly active players1 and grosses over 1 billion US dollars of revenue yearly.2 With 18M US dollars, one of the largest price pools in the history of eSports for a single tournament was crowdfunded almost entirely by the player base of Valve’s Dota 2.3

MOBAs are played in independent n vs n matches, typically with n = 5, in which the players of each team need to closely cooperate to penetrate the other team’s defences and obtain victory. Players who refuse to cooperate and act without considering their own team are easy targets and get killed more frequently, which diminishes the team’s chances. Together with the intricate and sometimes counter-intuitive strategic nature of MOBAs, this gives rise to conflict within the teams. Triggered by game events like kills or just simple mistakes, players begin to turn sour. The communication channels that were meant to coordinate the team effort can then be used to verbally assault other players, often by using profane terms and heavy insults.

Possible consequences are resigned players, whom might no longer be interested in competing for the win. But even if the match is won eventually, players could still feel offended, abused and might regret their decision to play the game in general. In this way, the mood of a communication could qualify as a social Quality of Experience (QoE) metric [1].

Collecting bad game experiences like this is harmful for the community, as it can bias a player’s attitude towards engaging in cooperation even when confronted with fresh opponents and new teammates in later matches. The perceived hostility in a player community is frequently referred to as toxicity. Toxicity imposes a serious challenge for game designers, as it may chase active regular players away. It might also prevent new players from joining the game, because a toxic base appears as unfriendly and hostile to newcomers.

The main contribution of this work is to devise an annotation system for chats of multiplayer online games that can be used for detecting toxicity (Section III). We apply the system to a large dataset (Section II) collected from a representative game of the MOBA genre and propose a method based on machine learning that uses the annotation system to predict the outcome of ongoing matches (Section IV). We end with related work (Section V) and conclusions (Section VI).

II. DATA

A. Data sources

All data used in this work are based on one of the ancestors of all MOBA games: Defense of the Ancients (DotA).4 This game started as a custom map for the real-time strategy game Warcraft III but soon became so popular that community platforms emerged that allowed for players to register, get profiled and being matched up against each other based on their skill. One of these platforms was DotAlicious, from which we crawled our data.

The website of DotAlicious is no longer available online, as DotA has been substituted by newer MOBAs like League of Legends, Heroes of the Storm or Dota II. The core game principles have not been changed by much by DotA’s successors, but the accessibility of replays, chat-logs and player-related information for them is more limited due to several privacy concerns of the developing companies. Also, alternative means of information exchange, like protected voice-chats, make it more difficult to obtain a record of comprehensive inter-team communication. Hence, we believe that our data from DotA are suitable for our purpose while still being representative for the game genre in general. Additionally, it allows us to study toxicity without harming a live community.

2http://goo.gl/bBKggU (gamasutra.com, Sep. 2015)
3http://goo.gl/FuFK6u (www.theguardian.com, Sep. 2015)
B. Data cleansing and match outcome

Our DotAlicious dataset consists of replays from 12923 matches, spanning the time between the 2nd and the 6th of February 2012.

The duration of matches in the dataset is distributed bi-modally, indicating that a small fraction of the matches ended prematurely. We used information from the hosting-bot of DotAlicious to determine matches that resulted in a draw or were canceled by the players early on. In total, out of 12923 matches, 1653 were aborted before game start, 706 were canceled after game start and 241 resulted in a draw by mutual player agreement (see Figure 1).

For the remaining matches, there are two possible outcomes: either one team destroys the other team’s main structure (victory condition) or all players of one team forfeit, which results in a collective surrender (loss condition). We have identified 10305 matches with a well-defined winning team, of which 6082 matches ended by the victory condition and 4223 matches resulted in a draw by mutual player agreement (see Figure 1).

For all matches, we extracted all chat-lines used by the players and applied a tokenization based on simple whitespace splitting. Symbols like “!” or “?” remained part of the words, as long as they were not separated by whitespaces. The case of the letters was unchanged to analyse the use of capitalization as a stylistic figure (shouting).

Overall, the language used is extremely abbreviated, elliptical, full of spelling-errors and barely following grammatical structures. Consequently, standard techniques from Natural Language Processing (NLP) like part-of-speech recognition, spelling-correction and language detection were either not applicable or performed poorly. On the other hand, we observed little variety in the topic of the chat, resulting in a rather restricted and repetitive set of vocabulary. We thus devised a novel annotation system to classify the most frequent words together with their miss-spelled variants.

The most dominant language in the corpus is English, which is used as a pidgin language for non-native speakers to communicate with each other. To classify the most frequently used words in this work, we do not consider words from any other language. Consequently, non English words will be either “unannotated” or classified as “non-latin” (for example in the case of Chinese, which is easy to detect).

To classify the semantics of a word, we apply sets of simple rules to them. There are three different classes of rules that we use:

1) **pattern**: the word includes or starts with certain symbols,
2) **list**: the word is member of a pre-defined list, and
3) **letterset**: the set of letters of the word equals the set of letters of a word from a pre-defined list.

The letterset class is useful to capture unintentionally or intentionally misspelled words, if no meaningful recombinations of their letters (like anagrams) exist in the corpus. For example, the set of letters used to spell the word “noob” is \{“n”, “o”, “b”\}, which is the same set as used for words like “NOOOOOOOOb”, “boon”, “noobbbbb” or “noonb” which were actually used in the chats. In total, the letterset method allowed to capture 224 (case-sensitive) different ways of writing “noob” that were used in the dataset. On the other hand, no other meaningful English word that could be built using this set (for example “bonobo” or “bonbon”) was found in the corpus. Also for other words than “noob”, the amount of introduced false positives due to the letterset-method was negligible for our dataset.

The precise word-lists and patterns that we used are provided as supplements to this work online as they are technical details. Nevertheless, Table I shows the rule classes used for each annotation category together with a short description, some examples, their precedence and their absolute prevalence in the text corpus.

The text-corpus consists of 7042112 words in total, of which 286654 are distinct. Each distinct word is checked against our rules and annotated accordingly. If no rules apply, the word is “unannotated”. If multiple rules apply, we break the tie by choosing the category with the highest precedence. Considering the set of all distinct words in the corpus, our annotation system covers around 16% of them. However, many of the most-frequent words are annotated, so that on average over 60% of all (non-distinct) words used per match have an annotation.

B. Different chat-modes

Our data allows us to investigate two fundamentally different chat-modes for each match: in the all-chat, a player can broadcast a message to each other player that participates in the match. In the ally-chat, the message is only sent to
TABLE I
ANNOTATION CATEGORIES

<table>
<thead>
<tr>
<th>category</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nonlatin</td>
<td>special character, foreign language</td>
</tr>
<tr>
<td>praise</td>
<td>acts of courtesy, kindness, sport spirit or gratitude</td>
</tr>
<tr>
<td>bad</td>
<td>profanity, swear words, inappropriate language</td>
</tr>
<tr>
<td>laughter</td>
<td>acronyms expressing laughter</td>
</tr>
<tr>
<td>smiley</td>
<td>emoticons, symbols resembling faces or emotions</td>
</tr>
<tr>
<td>symbol</td>
<td>symbols or numbers</td>
</tr>
<tr>
<td>slang</td>
<td>DotA-specific game-technical terms, used to coordinate with team</td>
</tr>
<tr>
<td>command</td>
<td>in-game commands, control words to trigger certain effects</td>
</tr>
<tr>
<td>stop</td>
<td>English stop words</td>
</tr>
<tr>
<td>timemark</td>
<td>automatically generated time-stamps, prepended in pause-mode</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>rules</th>
<th>examples</th>
<th>precedence</th>
<th>unique count</th>
</tr>
</thead>
<tbody>
<tr>
<td>pattern</td>
<td>文章</td>
<td>500</td>
<td>20133</td>
</tr>
<tr>
<td>list</td>
<td>gj, gg, thx, hf</td>
<td>100</td>
<td>295</td>
</tr>
<tr>
<td>list, letterset</td>
<td>noob, idiot, *</td>
<td>90</td>
<td>4881</td>
</tr>
<tr>
<td>letterset</td>
<td>HAHAHAHA, lol, ROFL</td>
<td>60</td>
<td>2158</td>
</tr>
<tr>
<td>pattern, list</td>
<td>:D, :, oO, _ _ _ _</td>
<td>50</td>
<td>1110</td>
</tr>
<tr>
<td>pattern</td>
<td>?, 1, ..., 1, 2, 3, 4</td>
<td>40</td>
<td>3181</td>
</tr>
<tr>
<td>list</td>
<td>ursa, mid, back, farm, bkb</td>
<td>30</td>
<td>10046</td>
</tr>
<tr>
<td>pattern</td>
<td>'ff, 'pause, -swap</td>
<td>20</td>
<td>2513</td>
</tr>
<tr>
<td>list</td>
<td>was, i, it, can, you</td>
<td>10</td>
<td>1322</td>
</tr>
<tr>
<td>pattern</td>
<td>[00:05], [01:23]</td>
<td>5</td>
<td>223</td>
</tr>
</tbody>
</table>

Fig. 2. Average use of annotated words per chat-mode. Chat-mode depicted as solid bars (all-chat) and as transparent bars (ally-chat). Error-markers show one standard deviation. Category “unannotated” was omitted.

C. Toxicity detection

For the purpose of our investigation, we define toxicity as the use of profane language by one player to insult or humiliate a different player in his own team. As such, the use of “bad” words is a necessary, but not sufficient condition for toxicity. For example, bad words can also be used just to curse without the intent to actually insult someone else. Profanity is also used in ironic or humoristic ways. For example, some players use self-deprecating remarks to admit in-game mistakes: “sry, I am such a noob - lol”. Thus, detection of toxicity can not be based on words alone but needs to take the current context into account.

We are using n-grams to distinguish toxicity from ordinary profane language. An n-gram is a contiguous sequence of n words that appears in a context. The context in our case consists of all words in the chat-line that contained the “bad” word plus all words from all chat-lines that were sent by the same player to the ally-chat not more than 1 second before or after.

For all players who participated in at least 10 matches, we search for all “bad” words they use, construct their contexts and count each n-gram that contains at least one “bad” word for n = 1, 2, 3, 4. Afterwards, we look at the 100 most frequently used n-grams for n = 1, 2, 3, 4 and manually determine which of them are toxic and which are not. Our criterion for toxicity is the following: for unigrams (n = 1) we consider them toxic if they could be understood as an insult. For example “crap” is no insult, but “moron” is. For n-grams with n = 2, 3, 4, we consider every context toxic that includes an insult directed towards a person. Examples include “*f*ing idiot”, “shut the *f*” and “i hope u die”. On the contrary, profane language that we do not classify as toxic includes n-grams like “*f* this”, “cant do s**” and “dont give a *f*”. In total, we deem 45 unigrams, 21 bigrams, 32 trigrams and 36 quadgrams as toxic. The list of these n-grams is provided as supplement to this work online (see Footnote 6).

IV. ANALYSIS OF GAME TOXICITY AND SUCCESS

A. Triggers of toxicity

Our method detects at least one toxic remark in 6528 out of the 10305 matches. In 90% of all toxic remarks, there are at most 5 toxic remarks detected. Several outliers exist in the data, the strongest contains 22 toxic remarks in a single match. The total number of toxic remarks was 16950. We expect that certain game events trigger players to act toxic.

Some authors did not want to include explicit quotes of profanity, which is why we decided to apply self-censorship in this work.
One possible game event is a kill where one player (killer) temporarily eliminates the character of another player in the opposing team (victim). There is a reaction time $\Delta$ involved between the actual kill-event and the time a player needs to submit a response to the chat. We look for each toxic remark if there was a kill-event taking place not earlier than $\Delta$ seconds before. For comparison, we also choose 16950 random chatlines (distributed over all matches) and look for a kill-event in their recent past as well. It turns out, that toxic remarks are more frequently preceded by kill-events than random remarks. Table II reports the absolute number of kill-events and how many of them were submitted by the killer or the victim. Especially victims of kill-events tend to become toxic, potentially blaming their teammates for their own fate.

### B. Game success and profanity

We have the hypothesis that with diminishing chances to succeed in the game, the level of profanity raises. To test our hypothesis, we compute the winrate for each player as the amount of matches won divided by the amount of matches played in total. We restrict the analysis to players who participated in at least 10 matches, which leaves 4009 distinct players in our dataset. Next, we count how many words the players participated in at least 10 matches, which leaves 4009 distinct players in our dataset. We restrict the analysis to players who participated in at least 10 matches. For comparison, we also choose 16950 random chatlines (distributed over all matches) and look for a kill-event in their recent past as well. It turns out, that toxic remarks are more frequently preceded by kill-events than random remarks. Table II reports the absolute number of kill-events and how many of them were submitted by the killer or the victim. Especially victims of kill-events tend to become toxic, potentially blaming their teammates for their own fate.

<table>
<thead>
<tr>
<th>kill-events</th>
<th>from killer</th>
<th>from victim</th>
</tr>
</thead>
<tbody>
<tr>
<td>toxicity ($\Delta = 5$s)</td>
<td>2219</td>
<td>23</td>
</tr>
<tr>
<td>random ($\Delta = 5$s)</td>
<td>1488</td>
<td>74</td>
</tr>
<tr>
<td>toxicity ($\Delta = 10$s)</td>
<td>5285</td>
<td>124</td>
</tr>
<tr>
<td>random ($\Delta = 10$s)</td>
<td>3176</td>
<td>200</td>
</tr>
</tbody>
</table>

As we can see from the top part of Figure 4, after a short initial period, in which it is uncertain to the players whether they might be losing or winning, we observe that teams that will lose the match in the end tend to use relatively more bad words than teams that will win the match. This difference is
The classifier itself has no notion of time: the TF-IDF features (term frequency inverse document frequency) of each word, a standard weighting technique frequently used in information retrieval [2]. For all computations, we use Scikit-learn [3] with its default parameters for all algorithms and do not undertake any effort to optimize them. The idea is not to create the most accurate classifier possible but rather to use the accuracy of the classifier to measure the importance of words with respect to match outcome.

The outcome should become more certain with the progression of the match, which should be reflected in the words used by the players. We introduce the parameter $t$ to control the amount of chat history that is given to the classifier. For example, for $t = 1.0$ the classifier is trained (and evaluated) on the complete ally-chats of each match, whereas for $t = 0.5$ it only knows what was written until the middle of the matches. The classifier itself has no notion of time: the TF-IDF features are purely based on frequencies and reflect neither order of words nor the specific time they were submitted to the chat.

As each word corresponds to one feature, we can partition all features by using our annotation system. We use the classifier 1) for all words regardless of their annotations, 2) for all words but words from the “command” category, 3) for no words except from the “bad” category and 4) for no words except from the “slang” category. The reason for excluding words from “command” is to avoid to provide the classifier with information if a player forfeited, which is announced by typing the command word “!ff” in the chat.

Figure 5 shows the average accuracy and the 95% confidence interval of the classifier for these scenarios under a 10-fold cross-validation. The number of used features and the accuracy scores for $t = 0.5$, $t = 0.75$ and $t = 1.0$ are presented in Table IV.

While words from the “bad” category (which constitute a precondition for toxicity) have some predictive power, it is significantly lower than using just all words or words from “slang” alone. We find it also remarkable that “slang” uses the least amount of features but gives still fairly good predictions. This might be due to the importance of team coordination which is covered mostly by key words from this category. It seems reasonable that their usage shows not only the game expertise of players, but also engagement and an increased interest to improve the team-play, which could result in a better chance to win the match. The occurrence of “bad” words however seems to be much less indicative for either winning or losing, suggesting only a weak link to game success. Consequently, also toxicity might not be the best indicator to determine if a game is going well for a team or not. Profanity will appear either way.

V. RELATED WORK

Antisocial behavior in virtual environments has been investigated in the field of computer sociology, most commonly under the term cyberbullying [4]. The impact of profane language on video games [5] and in a wider sense also on social media [6] is a vital area of research. Suler [7] shows psychological factors explaining the online disinhibition effect, giving toxic disinhibition as a negative example. This effect is a possible explanation why we observe such high levels of bad behavior online in general.

Similar to toxicity is the concept of griefing, the act of disrupting the game experience of other players by performing unacceptable actions. This has been investigated for virtual
worlds like Second Life [8] and MMORPGs like World of Warcraft [9].

An excellent case study for toxicity in MOBAs is given by the works of Blackburn and Kwak [10], [11]. The authors use crowd-sourced decisions from the tribunal, a player-based court that passes judgement on reported incidents in matches from League of Legends. While our definition of toxicity is tied to profane language only, the authors additionally consider certain in-game actions (i.e. “intentional feeding”) as toxic. They develop a classifier to assist players or even substitute the crowd-sourced decisions of the tribunal, which are whether an accused player is guilty of toxic behavior or not. As only cases submitted to the tribunal are considered, the authors have access to a ground truth for toxicity which is not present for our data. However, this might also create a selection bias, as typical matches will not end up on the tribunal.

Shim et al. [12] describe a different system based on the Pagerank to filter out “bad players” in MOBAs. Our approach is orthogonal, as it uses natural language processing on the player chats instead of relying on player’s complaints submitted via a report function. Institutions like the tribunal would not work without players reporting others, while our approach does not need any explicit player feedback to detect and monitor toxicity.

VI. DISCUSSION AND CONCLUSION

We developed a methodology to annotate frequently used expressions in written chat communication of Multiplayer Online Games. While our method is tested in this work only with data from DotA, we believe that it can be adapted to other MOBAs and possibly even games of different genres. To use the full system, one would need to update the predefined lists and patterns of game-specific terms to match their equivalents from the new game. Although this requires some degree of game-knowledge, the detection of toxicity itself is largely independent from any game-specifics, as it is based on profanity used in English language, enriched by a few terms commonly used in computer games.

The developed toxicity detection is based on contextual information to distinguish simple swearing from deliberate insults. It can be a building block for a monitoring system that can be used together with player reports to identify toxic players. Moreover, a well-trained classifier could be used to design a live-system that displays the odds of winning for each team to observers based on their communication, while the match is still ongoing.

Our analysis shows that toxicity is fueled by the inherent competitiveness (i.e., killing each other) of MOBA games but is only weakly linked to success. If players can be successful despite being toxic, they need a different incentive to cease insulting and behave more pleasant. On the other hand, the matchmaking systems that ensemble the teams could be altered to take toxicity into account to avoid creating a social powder keg. Although we might not be able to prevent toxicity entirely, controlling it would ensure a much more positive game experience for newcomers and experienced players alike.

REFERENCES