Toxicitat al chat de Twitch

MIS 2.0

Report

#Talaiòtics

Sergi Olives	**NIA: 193196**
Lluís Hernández	**NIA: 193200**
Eloi Yerpes	**NIA: 192781**

ÍNDEX

1. Context i Objectiu del Projecte

En la societat en la que vivim, on cada vegada hi ha més anonimat a les xarxes socials i una comunicació més fàcil, és necessari controlar en certa manera la toxicitat a la que es pot

arribar en certes plataformes. Nosaltres ens hem centrat en la toxicitat a Twitch, una plataforma que està molt popular, i on hi sol haver comunitats molt tòxiques depenent de l'streamer o joc que s'està emetent en directe.

El nostre objectiu principal es reduir la toxicitat del chat d'aquesta plataforma, analitzant la toxicitat de diferents streams, crear una aplicació que sigui capaç de reduir aquesta toxicitat, i mostrar-la d'una manera gràfica per a que els streamers puguin veure quin nivell de toxicitat té el seu chat, i puguin prendre mesures a consciència.

2. Desenvolupament

L'aplicació s'ha desenvolupat en 5 grans passes, una primera on extreim dades i les marquem com a tòxiques o no, després crearem un model de ML on prediem la toxicitat d'un missatge i posteriorment analitzar l'eficàcia d'aquest model per a finalment calcular la toxicitat general del xat de twitch en un interval de temps, mostrar-ho de forma gràfica i desenvolupar un bot per a reduir la toxicitat del stream.

2.1. Extracció de dades i labeling:

Primer hem passat per un procés d'obtenció de dades, per al qual hem utilitzat un bot creat per nosaltres (**reader.py**) que es connectava a un chat de twitch en específic (o varis) i ens anava mostrant els missatges en temps real i a la vegada guardant-los en un arxiu txt en local. Nosaltres, una vegada recopilats els missatges, els *marquem (labelling)*. Això vol dir que davant el missatge posem l'string "__label__toxic" o "__label__notoxic" (aquesta sintaxis és necessària per el bon funcionament de la llibreria FastText explicada en els següents apartats), dependent de si considerem el missatge escrit toxic o no. En aquest cas hem considerat els missatges tòxics si tenen paraules obscenes, insults, amenaces, odi contra una identitat o tòxiques en general. El fitxer final amb els missatges labeletjats es l'input que agafarem per tal d'entrenar el nostre model de ML.

```
__label__toxic i hope you die
__label__toxic fuck
__label__toxic show bob
__label__toxic stoopid
__label__toxic can u just suck it
__label__toxic you're a furry pog
__label__toxic n*gg
__label__toxic shroudw fuck ya
__label__toxic trash game
__label__toxic u suck
__label__toxic i have aids
__label__toxic fuckin dog
__label__toxic u dumb
```

```
__label__notoxic heyyyy
__label__notoxic lets go boys
__label__notoxic ben trolling
__label__notoxic i believe in you
__label__notoxic it's over
__label__notoxic ofc he hits it
__label__notoxic bennnyyy
__label__notoxic bruh crit is lagging
__label__notoxic get your shirt back off
__label__notoxic motivate him
```

Per a entrenar el nostre model hem decidit agafar un corpus labeletjat manualment de 975 msg, amb un 70% de no tòxics i un 30% de tòxics.

Prèviament a l'inici del desenvolupament del projecte, es va comentar la possibilitat d'obtenir dades de Twitch (50%), Kaggle toxic dataset (25%) i Twitter (25%) però després de repensar-ho, vam veure que realment el llenguatge utilitzat a Twitch és molt diferent al dataset de Kaggle i Twitter i que labelejar dades d'aquests dos datasets tampoc ens estalviava molt temps, així que vàrem optar per obtenir tot el nostre dataset de Twitch.

2.1.1. Normalització dels missatges

Per a millorar el rendiment del model de Machine Learning, tots els missatges d'entrenament varen ser normalitzats passant totes les lletres a minúscules, eliminant signes de puntuació i suprimint tabs i salts de línia.

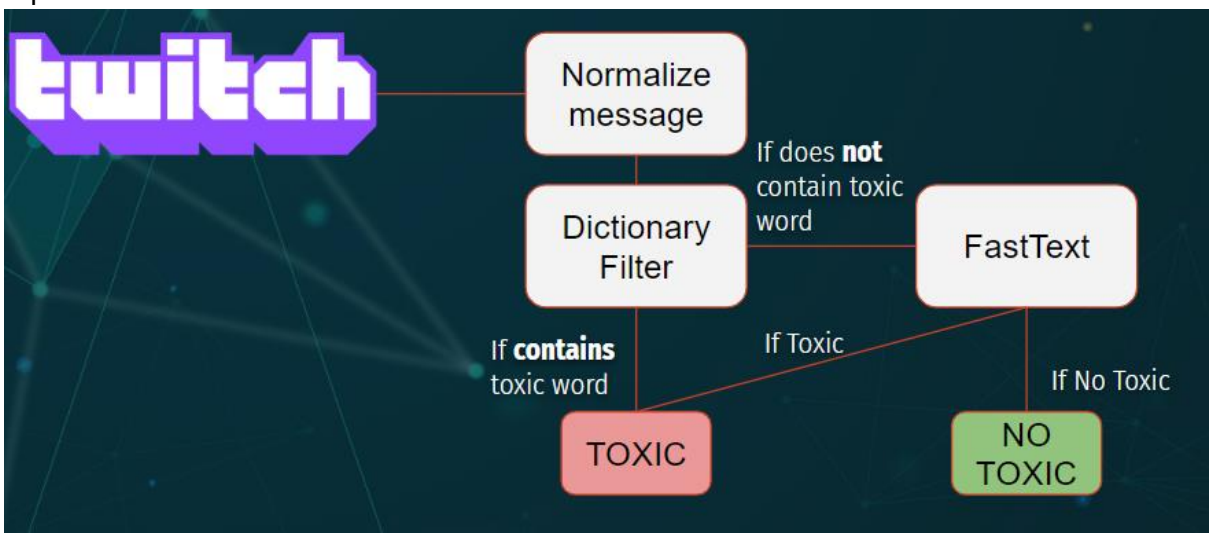
2.2. Classificació de missatges

Seguidament, amb el bot que ens va extraient les dades *in real time* dels chats, hem implementat un mètode de classificació binària (tòxic o no) per tal de poder calcular posteriorment la toxicitat del chat. A continuació, hem normalitzat tots els missatges d'entrenament tal i com s'ha explicat prèviament.

El primer filtre es basa en utilitzar un diccionari amb paraules de *hate speech*. Aquest diccionari el vam crear a partir de dos diccionaris cercats [1] [2] . Per aplicar-lo, recorrem tots els missatges paraula per paraula i anem comprovant si pertanyen al diccionari. Segons el resultat, els classifiquem com a tòxics o no tòxics i passem només els missatges no tòxics al segon filtre.

Aquest segon filtre es un model que utilitza la llibreria Fasttext i que ens classifica els missatges marcats com a no tòxics pel model. El model inicialment l'hem entrenat amb les dades etiquetades que hem extret, com expliquem en la secció anterior, i a part hem definit uns certs paràmetres d'entrada: el nombre d'epochs (defineix el nombre de vegades que l'algorisme d'aprendisatge treballa amb el dataset d'entrenament) amb un valor de 25, el wordNgram (defineix seqüències de N lletres de la paraula donada que ens donen els prefixos i sufixos d'aquesta i ens ajuden a distingir amb les paraules originals) que en el nostre cas es 2 i el lose rate de 1.

Hem decidit utilitzar diccionari ja que es una manera ràpida de fer un primer filtre per tal de que la quantitat de missatges que introduïm en el model sigui menor i per tant vagi més ràpid.

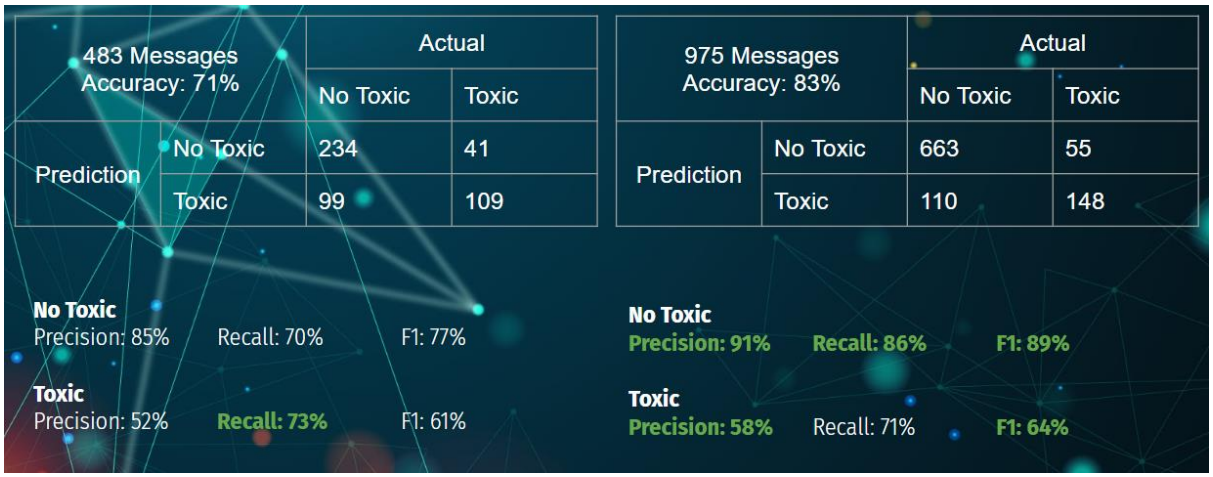


Toxic Message Detection Pipeline

2.3. Anàlisi del model de FastText

Un dels primers dubtes que ens van sorgir al inici del projecte era quants missatges necessitavem. Per això, vam decidir fer una primera recollida de dades i utilitzar el mètode K-Fold Testing per a veure les estadístiques del nostre model (exclusivament FastText, en aquest cas no s'analitza el diccionari). El testing es va fer amb una K = 10 i amb el mètode *stratified*, és a dir, cada fold conté el mateix percentatge de missatges tòxics. Es va fer amb un script nostre anomenat **kfoldtesting.py** que utilitzava uns folds creats a partir d'un arxiu de text que era separat en 10 folds gràcies a l'script **createFolds.py**.

El primer que vàrem fer va ser recollir 483 missatges únics **sense normalitzar** (40% tòxics) i vàrem aconseguir una accuracy del 68%. Llavors vam decidir provar de normalitzar els missatges del model per a veure si millorava (aplicant el mètode de normalització explicat a l'inici del report) i l'accuracy va arribar a 71%. Finalment, vam decidir seguir recopilant dades fins arribar a 975 missatges que van ser normalitzats (30% tòxics) arribant a una accuracy del 83%.



Aquestes són les *confusion matrices* dels models amb instàncies normalitzades de 483 missatges i 975. Clarament es veu una millora substancial utilitzant el segon dataset, que va és el que al final vam decidir que utilitzem pel nostre projecte.

A partir d'aquestes matrius, veim que es prediu més missatges no tòxics que realment són tòxics que no a l'inrevés, resultat que ens ha agrada ja que preferim un bot que deixi passar alguns missatges tòxics enlloc de banejar usuaris que no envien missatges tòxics de forma incorrecte.

Analitzant els errors en el training set, ens trobem amb alguns casos curiosos com el fet de que qualsevol missatge que contingui la paraula **ur** es marca com a tòxic probablement per la quantitat de missatges tòxics que contenen aquesta paraula com podria ser **ur gay** o **ur bad**. A més, normalment els missatges marcats incorrectament com a tòxics solen ser per errors en el context i el significat en funció d'aquest mentres que els missatges incorrectament marcats com a no tòxics són a causa d'errors ortogràfics i falta de dades per a que el model entengui que fuckk == fuck.

FastText ens retorna també un valor anomenat **certesa de la predicció** que va del 0 a l'1 i indica amb quina certesa s'ha fet la predicció. És curiós com tot i haver prediccions errònies,

el model sol tenir una certesa molt alta, normalment per damunt del 75% i mai baixa del 50% llavors realment per el model, els missatges aquests estan certament ben predits. No arribem a cap conclusió amb aquesta informació però descarta l'hipòtesi de que els missatges predits erròniament tindrien una certesa baixa.

2.4. Computació de la Toxicitat

Una vegada hem tingut un model que classifica suficientment bé, amb la accuracy mostrada en la secció anterior, ens hem centrat en avaluar els streamings. Per fer-ho, hem calculat la toxicitat del chat en un cert interval de temps determinat, que pot ser modificat en cas de fer estudis en una longitud de temps més llarga (l'script és **dic_filter.py**). Si ens centrem en el càlcul en si, hem dissenyat una funció, en la qual li passem una llista de 1 i 0 (tòxic o no tòxic) i ens conta la quantitat de uns que té.
La llista que passem com a input a la funció es buida una vegada s'arriba al temps determinat per tal de que només hi hagin les etiquetes dels missatges d'aquell interval de temps.

La toxicitat en un temps determinat d'un xat de twitch la mesurem en missatges tòxics entre total de missatges en l'interval de temps.

3. Aplicacions

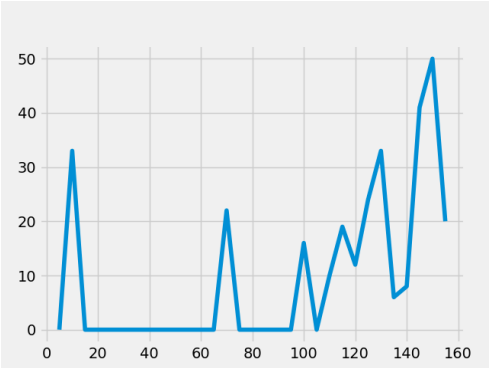
Amb l'objectiu de reduir la toxicitat al xat de Twitch, vam decidir crear un chat bot moderador i una eina d'anàlisi per entendre millor com funcionava la toxicitat als xats. Primerament també vam pensar en intentar entendre la toxicitat d'un xat relacionant-lo amb el Twitter del streamer però era una aplicació massa ambiciosa que per motius de temps no hem pogut implementar.

3.1. ChatBot

Hem desenvolupat un bot que assignat com a moderador a un chat de twitch permet modificar el mode del xat entre 4 opcions: tothom pot parlar, només seguidors, només subscriptors i només emojis (**dic_filter.py** sempre i quan es tinguin permisos de moderació). La toxicitat del xat es calcula amb la unificació del diccionari i el model de fasttext tal i com s'ha explicat abans, amb la peculiaritat de que ara, quan es supera el 20% de toxicitat en el xat en un interval de temps predeterminat (10 segons en el nostre cas per a fer la demo) puja un nivell de restricció, és a dir, si el xat estava en només followers i supera el 20% de toxicitat, el mode canviarà a només subscriptors. De la mateixa forma, si en un interval de temps la toxicitat baixa del 20%, es baixarà un nivell de restricció, per exemple de només followers a que tothom pugui parlar. Aquesta és una idea per a comprovar que funcionés bé i es pogués mostrar clarament, però tant els intervals de temps, com el percentatge per a canviar de mode com el tipus de canvi de mode són totalment customitzables per l'streamer, que per exemple podria fer que si la toxicitat puja un 40%, salti dos nivells alhora.

3.2. Livegraph

També a partir de la pipeline de computació de toxicitat, hem desenvolupat una altra versió d'un bot que llegeix missatges de twitch, els classifica com a tòxics o no i computant la toxicitat total del xat en un interval de temps determinat (30 segons en el nostre cas, modificable a través de l'script **dic_filter.py**, sense permisos de moderació o comentant la part dels canvis de mode de xat). Una vegada computada aquesta toxicitat, es guarda en un arxiu de text juntament amb l'interval de temps. Després, s'ha de correr un script per separat (pot ser al mateix temps, però no es poden ajuntar en dos scripts diferents per problemes de sincronització d'escriptura/lectura) anomanat **livegraph.py** que mostra el gràfic l'evolució en temps real de la toxicitat del xat de twitch de forma visual i senzilla per a que l'streamer tingui informació sobre el que està passant al seu canal.

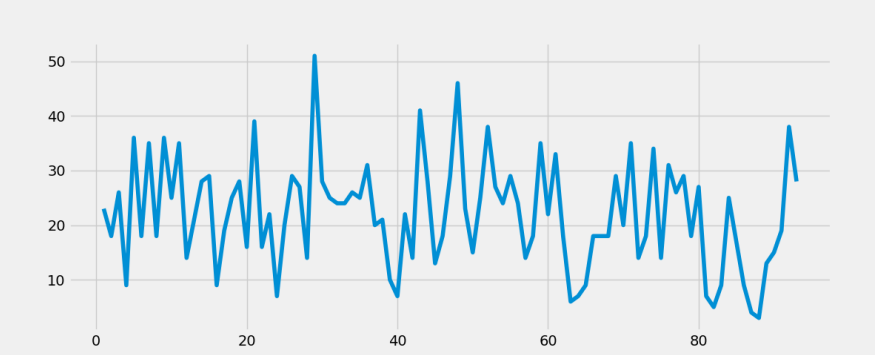


3.3. Anàlisi de retransmissions

Aquest bot que computa la toxicitat, l'hem provat de correr en diferents streamings per tal d'obtenir i analitzar els resultats. Hem fet una recopilació dels més interessants, els quals son un streamer toxic de lol, dues competicions i un stream benèfic. Els resultats de cada un els podem veure a continuació:

Loltyler1, jugant a LOL amb el chat amb only-followers:

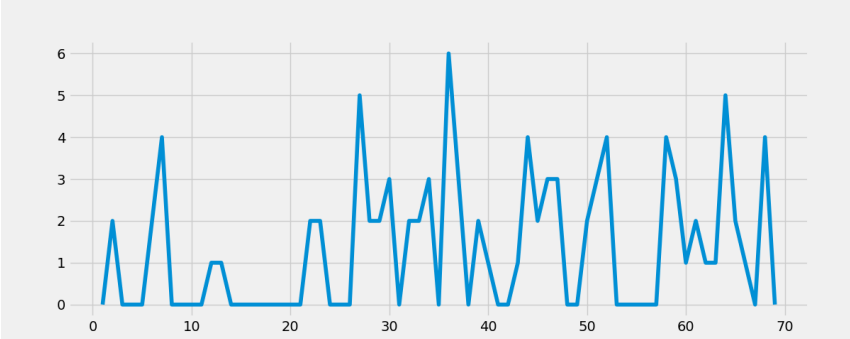
%Toxicity



#Intervals of 30 seconds

Stream benèfic, jugant a Minecraft amb el chat amb only-followers:

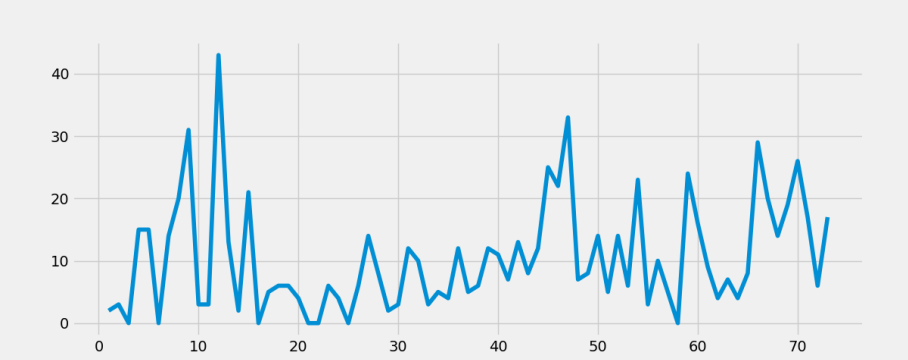
%Toxicity



#Intervals of 30 seconds

Competició Dota, chat amb only-followers:

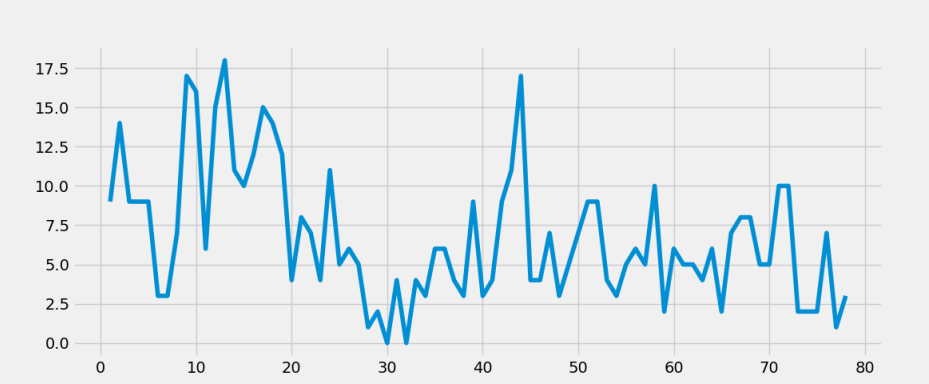
%Toxicity



#Intervals of 30 seconds

Competició CSGO, chat amb slow-mode i onlfy followers (els seguidors poden escriure cada 5s):

%Toxicity



#Intervals of 30 seconds

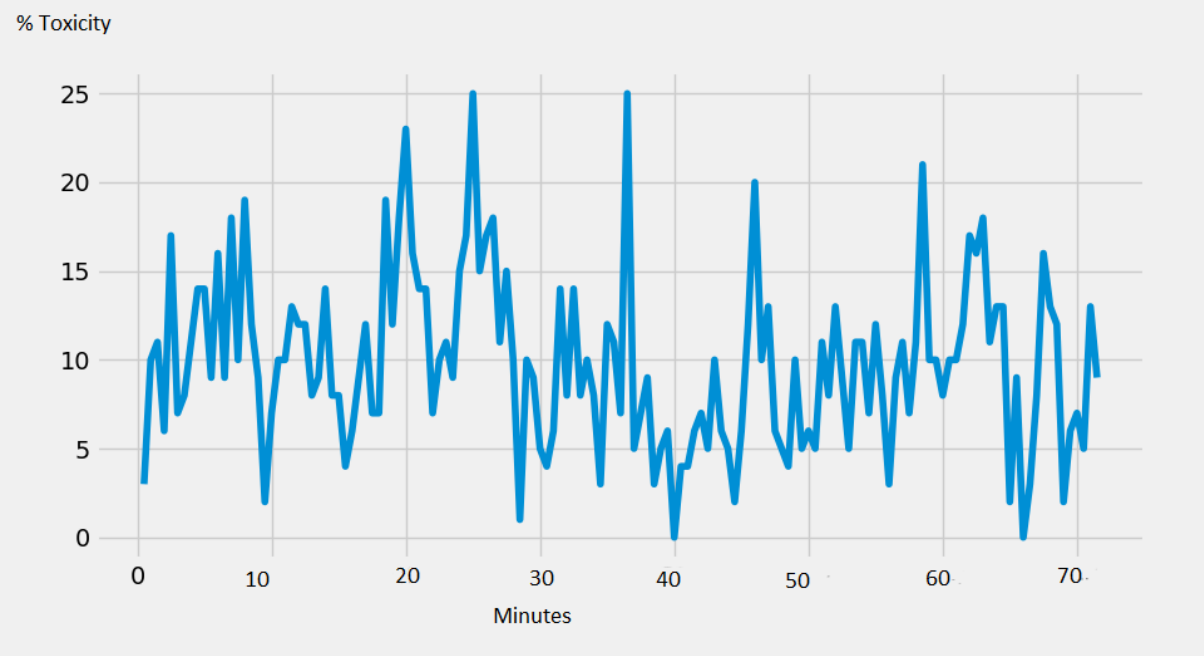
Com podem veure, l'streamer loltyler1 és amb diferència el que té un chat més tòxic, obtenint de vegades un % de toxicitat de més del 50% cosa que s'entén per el background tòxic de Tyler. A les dues competicions es pot veure un % de toxicitat moderat, on Dota en té més però s'ha de tenir en compte que l'slow-mode del chat de CSGO pot influir amb que els índex de toxicitat siguin més baixos. En aquestes dues també es poden observar pics de toxicitat, que poden ser donats per jugades puntuals en un punt de l'stream que causen que el chat es torni boig.

Finalment, hem decidit dur els resultats una mica més enllà i observar què passa en els moments en els que tenim pics de toxicitat. Per això hem agafat 3 streams de mitja hora i hem analitzat que ha passat en els seus pics de toxicitat. Els resultats han estat els següents:

ESL CSGO - Quarts de Final Fnatic vs Team Liquid - ESL Pro League Finals Odense

En aquest cas, es va analitzar el comportament del chat de Twitch en uns playoffs d'un dels tornejos més importants de CSGO, les finals de la Pro League retransmès per ESL_CSGO (https://www.twitch.tv/esl_csgo). La part analitzada és el primer mapa del partit a millor de 3 mapes de quarts de final entre l'equip suec Fnatic i el nord americà Team Liquid. L'stream va arribar a més de 90.000 viewers i el xat estava en **only followers, slow mode 5 seconds**.
L'anàlisi comença a l'inici del primer mapa a les 16:15 aproximadament i acaba en acabar el primer mapa 70 minuts després. En general, el nivell de toxicitat està de mitja entre 7% i 14% durant tot el mapa. L'anàlisi cada 30 segons no permet distingir si hi ha alguna disminució o augment de toxicitat entre rondes (15 segons d'espera) o timeouts (30 segons d'espera) però per exemple entre els minuts 37 i 45 es veu una clara disminució de la toxicitat (arribant a 0% en el minut 40) ja que es troben a la mitja part del primer mapa on hi ha anuncis/temps d'espera activat.
Com a altra dada curiosa que es pot extreure analitzant el gràfic i el xat en temps real, podem veure com els màxims i mínims es donen a causa de grans jugades i moments en la partida però el pic és màxim o mínim en funció de com sorgeix la jugada. Si la jugada ocorre

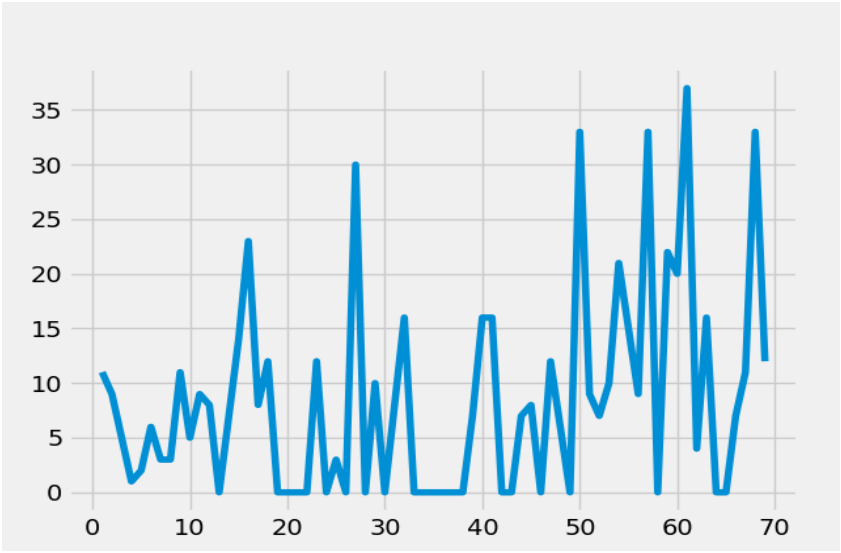
gràcies a uns grans dispars i moviment d'un jugador, el xat s'omple de paraules com "VAC" (Valve Anti Cheat) o el nom del jugador en qüestió provocant que la toxicitat arribi a mínims com als minuts 9 o 66. L'altre mínim destacat al minut 28 és a causa de que el públic a l'estadi comença a aplaudir i tot el xat s'omple de "Clap" simulant aquestes palmades via text.
Per altra banda, si ens anem a analitzar els màxims en la gràfica, tots són a causa de males jugades per part d'un equip o un jugador en concret, causant una reacció tòxica al xat. Per exemple al minut 20, hi ha un màxim tòxic ja que el xat es comença a enriure de l'equip de Team Liquid per anar perdent 1-7 i altres pics com al minut 25 on un jugador de Team Liquid intenta matar a ganivetades al rival i aquest l'acaba matant amb el seu rifle de francotirador sense utilitzar la mira, provocant aquest riure generalitzat. El mateix passa al minut 35 i 45 on dos jugadors fallen moltes bales fàcils. Les paraules més utilitzades pels missatges tòxics en aquest cas són frases que contenen la paraula NA(North America) com "NA CS" o "NA Grenade" ja que un dels memes interns del CS:GO és que els americans no saben jugar a Counter-Strike i ho utilitzen com a frase despectiva, fet que pot causar que la toxicitat en aquest partit en concret fos més gran ja que Fnatic es va acabar imposant a l'equip nord americà de Team Liquid.



League of Legends - Broxah
El segon anàlisi l'hem realitzat sobre l'streamer Broxah, jugador professional del LoL, jugant una partida. L'estudi, abarca tot una partida d'aquest joc, de duració aproximada de 30 minuts, amb més de 8000 persones veient-lo en directe i amb el mode de **chat de només subscriptors**.

Durant la partida es pot apreciar com els pics de toxicitat es deuen tant a jugades que succeeixen dins del joc, com a aspectes externs al joc que involucren a l'streamer i que porten a que hi hagi una discussió en el chat.
A més a més podem veure com el chat es poc toxic en termes generals on fins i tot hi han trams en que la toxicitat es 0 durant un període de temps, però quan hi ha jugades intenses, la toxicitat puja molt més que en el cas de l'streaming anterior.

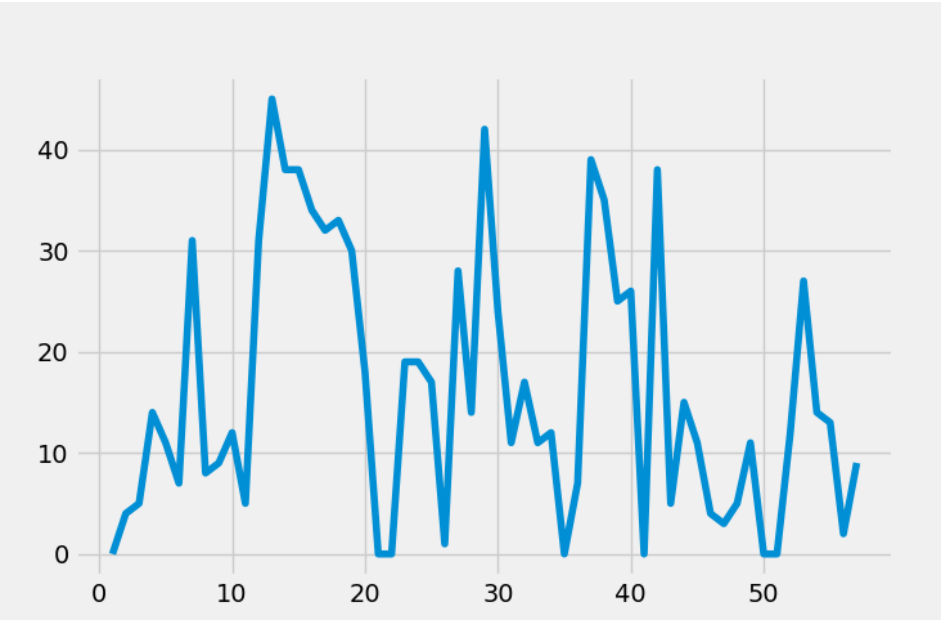
Centrant-nos més en els punts determinants del gràfic, el pic del punt 15 es deu a una jugada de 4vs4 que acaben guanyant. El punt 26-27 es una discussió que hi ha en el chat entre les lligues europea i nordamericana que tot i que es frena en els minuts següents, s'allarga fins al punt 50 on hi ha un altre pic de toxicitat alta. Finalment trobem diversos pics en el tram final de la partida que es deuen a jugades d'importancia que succeeixen dins del joc (com la pèrdua del baron, jugadors que no acaben de rematar, ...) i que acaben propician que l'equip de l'streamer acabi perdent la partida.



Overwatch - Dafran

Per finalitzar, hem recollit dades d'un stream de Overwatch, en concret d'un streamer anomenat Dafran, que ara presumia de no ser tòxic i ser un streamer "reformed pma". En el seu streaming hi havia 6000 persones connectades i s'han recollit dades durant mitja hora, anotant el que anava passant per veure si es poden extreure conclusions del resultat:

%Toxicity



#Intervals of 30 seconds

Podem veure que es l'stream més tòxic de tots (quan streamer en si a l'stream no es va comportar de manera tòxica). Podem veure 3 pics de toxicitat, en els minuts 10, 15 i 20. El primer pic ve donat quan maten a l'streamer després de que empri la seva ulti sense matar a ningu, l'equip perd la primera ronda, i l'streamer reporta a un personatge del seu equip. El segon pic ve donat quan l'equip de l'streamer està atacant, l'streamer fa una bona jugada (finalment mor), però inclus així el seu equip no és capaç de guanyar el segon punt. Finalment el tercer pic (que realment es pot dividir en 2, un en el minut 17 i un altre en el 22), ve donat quan l'streamer tira la seva ulti, va morint el seu equip i finalment perden per c9.

4. Problemes i feina futura

Ens hem trobat alguns problemes durant el desenvolupament del projecte. Just a l'inici ens hem plantejat quin criteri seguir per a marcar els missatges com a tòxics o no i quants de missatges necessitem per al nostre model. També era important saber si les dades funcionarien, si els missatges de twitch es poden prediure d'una forma més o menys fiable. El procés de normalització també va ser crític en l'aspecte de saber si agafavem emojis o no de cara a l'entrenament del model. Finalment, el gran problema que ens trobem en un xat de twitch i analitzant només els missatges de text, és que el context és realment important i ens estem perdent informació al només analitzar missatges però raons logístiques, de cara a aquest projecte ha estat impossible analitzar el context de cada missatge.

Llavors, a nivell de feina futura pel seguiment del projecte hem pensat en algunes solucions a aquests problemes com tenir en compte el context a l'hora de labelejar, mirant el vídeo mentres es labeleja per exemple, però aquí apareix una problemàtica, el model també hauria d'entendre el context. A nivell de data labelling, una de les problemàtiques que ens

hem trobat és que cada instància ha estat labelejada per una sola persona per motius de temps tot i que idealment hauria haver estat labelejada per més d'una i ja que el criteri d'un missatge tòxic o no pot variar una mica, tot i haver algunes premises bàsiques. També una de les possibles millores és augmentar la quantitat de missatges d'entrenament, que gràcies a les dades del testeig s'ha demostrat que realment val la pena i finalment l'anàlisi d'emojis i si es poden entendre com a tòxics seria un estudi interessant.

5. Conclusions

Per acabar, estem molt contents amb el projecte i els prometedors results que hem aconseguit amb un dataset curt, demostrant que això amb moltes dades pot funcionar molt bé. Ens hem trobat amb certes dificultats per aconseguir les dades de Twitch culpa de la seva API tot i que aquesta mateixa ens ha permès desenvolupar un chatbot de forma bastant ràpida. I finalment, a partir de les dades que hem aconseguit es pot veure que realment hi ha comportaments molt diferents en funció del joc i el tipus d'stream i streamer que causen que un jugador tòxic tingui un xat tòxic mentres que un stream benèfic quasi no tingui missatges tòxics.

6. Referències

- [1] www.noswearing.com/dictionary
- [2] https://hatebase.org/
- https://fasttext.cc/docs/en/supervised-models.html
- https://pythonprogramming.net/live-graphs-matplotlib-tutorial/
- https://dev.twitch.tv/docs/irc