Sexism Bot classifier on Twitch:
Moderating sexist comments on Twitch streamings.

David Gayete Ibáñez, Ramon Vallés Puig
david.gayete01@estudiant.upf.edu, ramon.valles02@estudiant.upf.edu
UPF (Modelatge de la Interacció Social 2.0)

Abstract*— No es una afirmación nueva que hoy, en la mitad del siglo XXI, la humanidad siga atascada en esos viejos prejuicios como el sexismo. Es indiscutible que para erradicar estos prejuicios debemos cortar de raíz, y cambiar todo el sistema tal como lo conocemos, empezando por el sistema educativo. Pero, hasta entonces, debemos tratar de erradicar, o al menos minimizar, cualquier comportamiento inapropiado que veamos a nuestro alrededor. Esta proyecto tiene como objetivo suprimir cualquier comportamiento sexista en una de las redes sociales más populares, Twitch, en la que el creciente interés de los adolescentes por convertirse en streamers, ha traído consigo un notable grupo de personas con malas intenciones, que molestan a las streamers, sólo por ser mujeres.

1. Introduccion

En este proyecto desarrollaremos mediante el conocido lenguaje de programación "python" un Bot para la aplicación Twitch que a través de un clasificador de texto, implementado con la librería de facebook "FastText", podrá decidir si un comentario sobre un determinado streaming corre el riesgo de ser sexista, y en caso afirmativo, tomar medidas al respecto. Los criterios utilizados para castigar estos comentarios han sido decididos por los participantes del proyecto, pero depende de cualquiera que utilice esta aplicación tomar unas medidas u otras.

2. Dataset

La primera y más tediosa tarea es recoger una cantidad suficiente de comentarios y clasificarlos para entrenar más tarde nuestro modelo de FastText. El conjunto de datos final ha sido subido al repositorio github [0], y está abierto a cualquiera que esté interesado.

2.1 Recolección de datos

Para recoger los comentarios que más tarde clasificaremos, se seleccionó una lista de streamers femeninas. Esta selección se basó en el número de seguidores que tenía el usuario (fama), y el tema que trataba en sus streamings (temática). Algunos de estos usuarios seleccionados son: 'Pokimane', 'Amouranth', 'AriGameplays', 'Shainni', 'LaChilenaBelu'.

El algoritmo para recolectar los comentarios ha sido desarrollado con el API de twitch [1] para python.
Para realizar las llamadas a la API primero nos registramos como desarrolladores (imprescindible para poder crear también un bot interactivo), y una vez registrados creamos una aplicación llamada "Sexism on twitch". A partir de ahí obtenemos nuestras credenciales, que no publicaremos por razones de seguridad.

El algoritmo ha estado funcionando durante 4 horas, recogiendo comentarios de entre alguns streameings elegidos al azar dentro de los usuarios seleccionados. Para cada flujo se creó un archivo ".txt" con el nombre "Chat_ususario_i.txt".

2.2 Limpieza de Datos

Como la gran mayoría de los comentarios no eran consistentes, o contenían palabras imposibles de analizar directamente, aplicamos un filtro de limpieza a todos los documentos.

Esta limpieza consistió en:

- Eliminar todos los comentarios compuestos por tan solo una palabra.
- Convertir a minúsculas todas las letras y eliminar caracteres no alfabéticos.

- Reducir a un mínimo de 2 el número de caracteres repetidos consecutivamente.
- Word steaming, to find the root of each word.

Y por razones que explicaremos más adelante, todos esos términos usados para saludar ('hola', 'holi', 'hola', 'hey', 'adiós', 'ciao'...) han sido eliminados.

2.3 Clasificación de Datos

Una vez que tuvimos nuestra lista de comentarios, empezamos a clasificarlos manualmente en 2 categorías diferentes.

Estas categorías discriminan entre los comentarios sexistas (1) y los no sexistas o inocuos (0). La segunda categoría, a su vez, consta de 2 subcategorías, que separarán los comentarios amorosos ("babosos") de los altamente sexistas:

- **0**: Comentario normal (inocuo)
- **1**: Comentario inapropiado:
  - **0**: "Baboso"
  - **1**: Claramente sexista

De todos los comentarios que teníamos en los diferentes documentos, se han clasificado alrededor de unos 3.500. De este conjunto de comentarios clasificados, aproximadamente 300 caen en la categoría de inapropiados mientras que los otros 3k eran simplemente inocuos.

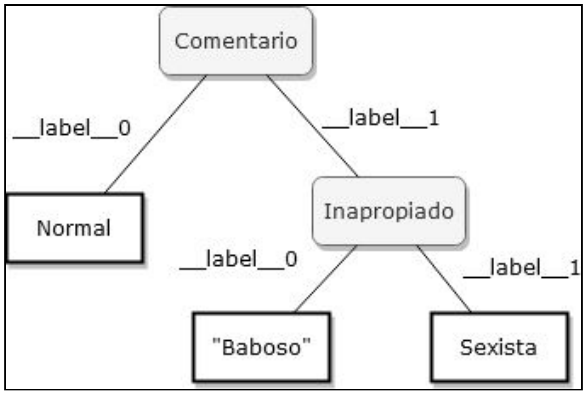

Fig. 1: Esquema básico del clasificador

3. Modelo

El clasificador ha sido construido con FastText [2], que es una biblioteca de código abierto para el aprendizaje de la clasificación de textos que permite entrenar representaciones supervisadas de palabras y frases.

Dado que clasificar el texto en diferentes categorías reduce la fiabilidad de un modelo, hemos decidido crear 2 modelos independientes que utilizaremos para hacer un árbol binario de profundidad 2, en el que las decisiones serán tomadas por cada uno de los clasificadores. De esta manera, el primer modelo descartará cualquier comentario inocuo, y todos los comentarios inapropiados serán enviados al modelo 2 para decidir su gravedad. La estructura se representa en la figura 1.

4. Entrenamiento

Los dos clasificadores FastText han sido entrenados con el 85% del conjunto de datos (el otro 15% ha sido reservado para la fase de pruebas).

La función o el criterio para medir el error de la predicción en cada iteración del entrenamiento és Cross Entropy Loss que recibe como input la probabilidad de cada clase dada por la función matemática Softmax [3]. Esta técnica es una de las más adecuadas para los clasificadores de texto, y la que se utiliza por defecto en FastText.

$$softmax(s)_i = \frac{e^{s_i}}{\sum_j^C e^{s_j}}$$

$$CrossE = -\sum_i^C t_i log(f(s)_i)$$

Ambos modelos tuvieron en cuenta el conjunto de unigrams, para representar palabras independientes, y bigrams, conjuntos de 2 palabras.

Los hiperparámetros utilizados se seleccionaron empíricamente como 5E-2 para el Learning Rate, y 200 iteraciones, y mostraron una precisión del 83,12 % y 66,67 % respectivamente.

5. Testeo

Un detalle que degradó considerablemente las predicciones del modelo, es que muchos de los comentarios sexistas que se habían clasificado, comenzaron en su gran mayoría con un cálido saludo, lo que confundió a nuestro modelo al identificar aquellos unigrams más utilizados por los usuarios sexistas. Este fenómeno hizo que nuestro modelo se alarmara con un simple "hola". Es por eso que, en la fase de limpieza, decidimos filtrar todos los saludos ya que no deberían tener ninguna influencia en el resultado de la predic- ción.

Para probar la calidad de las predicciones del modelo, se ha utilizado un pequeño conjunto de datos con aproximadamente 314 comentarios de usuarios. De estos 314 comentarios, 64 de ellos entran en la categoría de inapropiados. Con el, hemos podido

comprobar que el primer modelo tiene un 83,12% de precisión en la clasificación de los comentarios normales o no sexistas, mientras que el segundo modelo tiene, obviamente, más dificultad para clasificar el comentario entre “babosos” y fuertemente sexistas.

Para hacer un análisis más detallado de los resultados en ambos modelos, se realizó un análisis detallado.

	Normal	**Inapropiado**
Normal	228	22
Inapropiado	34	30

Fig. 2: Resultados del primer Modelo.

	“Baboso”	**Sexista**
“Baboso”	16	13
Sexista	10	25

Fig. 3: Resultados del segundo modelo.

Por un lado, como podemos ver en la Figura 2, de los 250 comentarios no sexistas, el primer modelo clasificó 22 como comentarios inapropiados y, de los 64 comentarios inapropiados, el modelo clasificó 34 como no sexistas.

Por otro lado, (Figura 3) sólo utilizamos los 64 comentarios inapropiados para probar el segundo modelo. De los 29 comentarios amorosos, 13 habían sido clasificados como comentarios sexistas, mientras que de los 25 comentarios sexistas, 10 habían sido clasificados como comentarios amorosos.

Lo que estos números nos dicen es que la mayoría de los comentarios inapropiados no se detectan, pero además, estos comentarios que el clasificador malinterpreta como normales, la mayoría de ellos resultan ser comentarios “babosos”, ya que son más difíciles de identificar.

6. Bot

Para construir el Bot en Twitch, usamos una librería de python llamada TwitchIO [4]. Para ello, hemos creado una clase ‘Bot’ a la cual mandamos nuestras credenciales de la API de twitch para inicializarlo en nuestro canal.

Una vez que el bot se ejecute, recibirá todos los mensajes que los usuarios envíen en el chat en vivo del streaming.

Por cada mensaje recibido por el bot, primero limpia el texto aplicando el mismo filtro que usamos con el dataset. Luego el mensaje se divide en un vector de palabras y se filtra con una lista de palabras prohibidas (creada de forma manual), que contiene palabras como "tetas", "escote", etc. Si una de las palabras del mensaje está contenida en la lista de palabras prohibidas, el mensaje será considerado como un comentario sexista, sin necesidad de utilizar el modelo entrenado.

En caso de que el primer filtro no haya detectado ninguna palabra crítica, pasamos al segundo filtro, es decir, a nuestros dos modelos clasificadores. Siguiendo la estructura explicada anteriormente (Figura 1), pasamos primero el mensaje al primer modelo. Si el mensaje es clasificado como inocuo, nos saltamos el segundo filtro y el Bot queda libre para recibir nuevos mensajes. En caso contrario, si el modelo clasificó el mensaje como inapropiado, pasamos el mensaje al segundo modelo. Si el modelo clasifica el mensaje como “baboso” o secista, el bot aplica los castigos correspondientes.

En referencia a los castigos por un comentario “baboso”, la primera y segunda vez que un usuario envía un mensaje de dicha categoría, el usuario recibirá un aviso en el chat. Si el mismo usuario ignora los avisos, la tercera vez que el usuario envíe el mismo tipo de mensaje, obtendrá un timeout de 60 segundos en el chat.

Sobre los castigos fuertemente sexistas, la primera vez que el usuario envíe un mensaje potencialmente sexista, obtendrá un timeout de 60 segundos en el chat. La segunda vez, el usuario obtendrá un tiempo de espera de 120 segundos en el chat. Finalmente, la tercera y última vez, obtendrá una prohibición permanente en el canal para que el usuario nunca pueda enviar más mensajes a ese streaming.

7. Simulación

Para hacer una simulación de nuestro Bot, primero enviamos el mensaje simple: "HOLAA", en este caso el primer modelo lo clasificó como un mensaje inocuo porque en la fase de limpieza lo considera un saludo y lo dejó correr sin necesidad de la interacción del clasificador. Como segunda simulación, el siguiente mensaje que enviamos fue "buenos días" y el primer

modelo lo clasificó también como un comentario inocuo.

Luego quisimos probar nuestro segundo modelo y para ello enviamos el comentario "hola hermosa". En este caso el primer modelo lo clasificó como un mensaje inapropiado y lo pasó al modelo 2 que finalmente lo clasificó como un mensaje de "hola hermosa", posteriormente se nos alertó de un comportamiento inadecuado.

Para la última simulación desafiamos a nuestro modelo con un mensaje sexista, así que enviamos "me suscribo si hace un twerk xd" y, como era de esperar, el primer modelo lo clasificó como un mensaje inapropiado y el segundo modelo como un mensaje sexista, así pues nos bloqueo del chat durante 60 segundos. Con esa simulación, podemos ver que, por simples comentarios que no dependen del contexto de la transmisión, el Bot dio resultados razonables.

8. Discusión y mejoras

Teniendo en cuenta los resultados a grossomodo, la mayoría de los errores se deben al contexto en el que está escrito el mensaje, sin ese contexto, se pierde alguna información pertinente y, por lo tanto, el mensaje adquiere un significado diferente. Muchos otros errores pueden ocurrir cuando el mensaje contiene palabras coloquiales o mal escritas y eso hace que nuestro modelo no las comprenda.

Una solución a estos problemas podría ser la utilización de diccionarios o conjuntos de datos de palabras coloquiales para evitar también los errores relativos a las abreviaturas de palabras o palabras mal escritas.

También sería una buena idea analizar la sintaxis de todos los mensajes. La razón es que la mayoría de los comentarios que están vinculados a un contexto determinado son también más largos y complejos, y la sintaxis en esos casos desempeña un papel importante. También porque se ha demostrado en numerosos estudios [5] que las personas que desean expresar frases más grandes suelen tener cuidado en la redacción, lo que hace que los detalles del comentario sean más significativos.

9. Conclusiones

En este proyecto hemos desarrollado un bot simple y funcional capaz de filtrar los comentarios inusuales que corren el riesgo de ser considerados como sexistas. Se han encontrado posteriormente, algunas debilidades que deben ser consideradas para mejorar la eficiencia del clasificador.

Algunas conclusiones generales que podemos extraer son que se necesitan datasets más grandes y variados para entrenar los diferentes modelos, lo que requerirá tiempo y dedicación, pero seguro que ayudará al clasificador a aumentar su conocimiento sobre cada categoría.

Como reflexión, pensamos que si Twitch se tomara en serio este tema y tuviera mayor interés en ayudar a las chicas streamers respecto a estos comportamientos inapropiados de personas sexistas, este tipo de problema ya no existiría.
Es hora de erradicar este estúpido prejuicio que es el sexismo, de nuestra sociedad.

10. Referencias

[0] R. Vallés, D. Gayete, Code implementation (opensource), https://github.com/VPRamon/SexismOnTwitch

[1] Twitch API, *Get Started with the Twitch API,* https://dev.twitch.tv/docs/api/

[2] FastText API: *FastText doccumentation*, https://fasttext.cc/docs/en/api.html

[3] F. Rashad, *Additive Margin Softmax Loss*, 24 June 2020, https://towardsdatascience.com/additive-margin-softmax-loss-am-softmax-912e11ce1c6b

[4] *Twitchio library for python, v1.1.0* https://pypi.org/project/twitchio/

[5] Janda, Harneet & Pawar, Atish & Du, Shan & Mago, Vijay. (2019). *Syntactic, Semantic and Sentiment Analysis: The Joint Effect on Automated Essay Evaluation.* IEEE Access. PP. 1-1. 10.1109/ACCESS.2019.2933354.