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*— It is not a new statement that today, in the middle of the 21st century, humanity is still stuck in those old prejudices such as sexism. It is indisputable that to eradicate these prejudices we must cut at the root, and change the entire system as we know it, starting from the educational system. But, until then, we must try to block, or at least to minimize, any inappropriate behavior we see around us. This research aims to suppress any sexist behavior in one of the most popular social networks, Twitch, in which the growing interest of teenagers in becoming streamers, has brought with it a noticeable group of people with bad intentions that bother the female streamers, just for being female.

1. Introduction

In this project we will develop in the well-known programming language "python" a Twitch Bot that through a text classifier, implemented with the facebook library "FastText", will be able to decide whether a comment on a given streaming runs the risk of being sexist, and if so, take action on it. The criteria used to punish these comments have been decided by the participants of the project, but is up to anyone using this application.

2. Dataset

The first, and most tedious task is to collect a sufficient amount of comments and sort them for later training our FastText model. The final dataset has been uploaded at the github repository [0], and is open to anyone.

2.1 Data scraping

To collect the comments which we will later classify, a list of female streamers was selected. This list was based on the number of followers the user had (fame), and the topic he/she was dealing with in his streams (theme). Some of these selected users are: 'Pokimane', 'Amouranth', 'AriGameplays', 'Shainni', 'LaChilenaBelu'.

The data scraping algorithm has been developed with the twitch API [1] for python.
To make calls to the API we first registered as developers (essential to be able to create an interactive bot as well), and once registered we created an app called "Sexism on twitch". From there we get our credentials, which we will not publish for security reasons.

The algorithm has been running for 4 hours, collecting comments from randomly chosen streams within the selected users. For each stream a ".txt" file was created with the name "Chat_user_i.txt".

2.2 Data cleaning

Since the vast majority of comments were not consistent, or contained words impossible to analyze directly, we applied a cleaning filter to all documents.

This cleaning consists of:

- Dropping all those comments composed of a single word.
- Lowering down and removing non-alphabetic characters.
- Reducing to a maximum of 2 the consecutively repeated characters
- Word steaming, to find the root of each word.

And for reasons that we will later explain, all those terms used to greet ('hola', 'holi', 'hi', 'hey', 'adios', 'ciao' ...) have been dropped.

2.3 Data classification

Once we had our list of comments, we started to manually sort them into 2 different categories.

These categories discriminate between sexist comments (1) and non-sexist or innocuous comments (0). The second category, in turn, consists of 2 subcategories, which will separate those love-struck ("babosos") comments from the highly sexist ones:

- **0**: Normal comment (innocuous)
- **1**: Inappropriate comment
  - **0**: Love-struck ("babosos")
  - **1**: Strongly sexist

From all the comments we had in the different documents, about 3,500 have been classified. From this set of classified comments, approximately 300 fall into the category Inappropriate while the other 3k were simply innocuous.

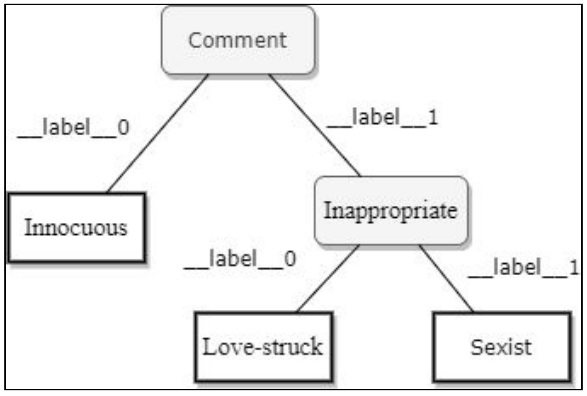

Fig. 1: Basic Scheme of the category classifier

3. Model

The classifier has been built with FastText [2], which is an open source library for text classification learning that allows you to train supervised representations of words and sentences.

As classifying text in different categories reduces the reliability of a model, we have decided to create 2 independent models in order to make a binary tree with depth 2, in which the decisions will be taken by each of the classifiers. In this way, the first model will discard any Innocuous comment, and all Inappropriate comments will be sent to model 2 to decide its gravity. The structure is represented in figure 1.

4. Training

The two FastText classifiers have been trained with 85% of the dataset (the other 15% has been reserved for the testing phase).

The loss function, or criterion used in each epoch is Cross Entropy loss, which receives as input the probability of each class given by the output of applying a softmax function [3]. This technique is one of the most suitable for text classifiers, and the default in FastText.

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

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

Both models took into account the set of unigrams, to represent single words, and bigrams, sets of 2 words.

The hyperparameters used were selected empirically as 5E-2 for the learning rate, and 200 epochs, and showed an accuracy of 83.12 % and 66.67% respectively.

5. Testing

A detail that degraded considerably the accuracy of the model, is that many of the sexist comments that had been classified, began in its vast majority with a warm greeting, which confused our model when identifying those unigrams most used by sexists. This phenomenon made our model alarm with a simple "hello". For that reason, in the cleaning phase, we decided to filter all greetings since they should not have any influence on the result of the prediction.

For testing the model, a small dataset with approximately 314 user comments have been used. From these 314 comments, 64 of them fall into the inappropriate category. Furthermore, we had proven that the first model has 83.12 % of accuracy on classifying normal or non-sexist comments, whereas the second model has, obviously, more difficulty classifying the comment between love-struck and strongly sexist.

To make a more detailed analysis of the results on both models, a detailed analysis was carried out.

	Normal	**Inappropriate**
Normal	228	22
Inappropriate	34	30

Fig. 2: Classifier Results of the First Model

	Love-struck	Sexist
Love-struck	16	13
Sexist	10	25

Fig. 3: Classifier Results of the Second Model

In one side, as we can see in **Fig. 2**, from the 250 non-sexist comments, the first model classified 22 as inappropriate comments and, from the 64 inappropriate comments, the model classified 34 as non-sexist.

On the other side, (**Fig. 3)** we only used the 64 inappropriate comments to test the second model. From the 29 love-struck comments, 13 had been classified as sexist comments whereas on the 25 sexist comments, 10 had been classified as love-struck comments.

What these numbers tell us is that most inappropriate comments are not detected, but in addition, these comments that the classifier misinterprets as normal, most of them result to be love-struck comments, as they are more difficult to identify.

6. Bot

For building the Twitch Bot, we used a python library named TwitchIO [4]. For that, we have created a Bot class where we passed our twitch api credentials to initialize it in our twitch channel.

Once the bot is running, it will get all the messages that users send in the twitch live chat.

For each message received by the bot, it first cleans the text by applying the same filter as we did with the dataset. Then the message is splitted into a vector of words and filtered with a forbidden words list, containing words like “boobs”, “escote”, etc. If one of the words in the message is contained in the forbidden words list, the message will be considered as a sexist comment, without the need of using the trained model.

In case the first filter had not detected any critical word, we pass to the second filter, that is, our two classifier models. Following the structure previously explained (**Fig. 1**), we first pass the message into the first model. If the message is classified as innocuous, we skip the second filter and the bot is ready to get new messages. In the other case, if the model classified the message as inappropriate, we pass the message into the second model. If the model classifies the message as love-struck, the Bot applies the love-struck punishments. If the bot classifies the message as strongly sexist, the bot will apply the sexist punishments.

In reference to the love-struck punishments, the first and second time a user sends a love-struck message, the user will get an advertisement in the chat. If the same user ignores the adverts, the third time the user sends the same type of message, will get a 60 seconds timeout in the chat.

About the strongly sexist punishments, the first a user sends a strongly sexist message, it will get a 60 seconds timeout in the chat. The second time, the user will get a 60 seconds timeout in the chat. Finally, the third and last time, he/she will get a ban permanently on the channel so the user will never be able to send more messages to that streaming.

7. Simulation

For doing the simulation of our Bot, we first sent the message “HOLAA” , the first model classified it as an innocuous message because the cleaning phase considers it a greet and dropped it without needing the interaction of the classifier. As a second simulation, the next message we sent was “buenos dias” and the first model classified it also as an innocuous comment.

Then we wanted to try out our second model and for that we sent “hola hermosa”. In this case the first model classified it as an inappropriate message and passed it to the model 2 which finally classified it as a Love-struck message.

For the last simulation we challenged our model with a sexist message so we sent “me suscribo si hace un twerk xd” and, as expected, the first model classified it as Inappropriate message and the second model as sexist message. With that simulation, we can see that, for simple comments that do not depend on the context of the streaming, the Bot gave reasonable results.

8. Discussion and further work

Considering the overall results, most of the errors are because of the context in which the message is written, without this context, some relevant information is lost and therefore the message acquires different meaning. Many other errors may occur when the message contains colloquial or mispelled words and that makes our model not understand them.

A solution for these problems could be to use dictionaries or datasets of colloquial words to avoid errors relating to abbreviations.

Also a good idea would be to analyze the syntax of all the messages. The reason is that most comments that are linked to a given context are also longer and com-

plex, and the syntax in those cases take an important role. Also because it has been proved by numerous studies [5] that people who want to express larger sentences usually take care in the writing, which makes the details of the comment more meaningful.

9. Conclusions

In this project we build a simple and functional bot capable of filtering unusual comments which run the risk of being considered as sexist. Some weaknesses have been found and must be considered when improving the efficiency of the classifier.

Some general conclusions that we can take are that bigger datasets are needed to train the different models, which will require time and dedication, but for sure will help the classifier to increase its knowledge on each category.

As a reflection, we think that if Twitch takes this topic seriously and wants to help all the girl streamers with these inappropriate behaviors from sexist people, this kind of problem would not exist anymore.

It is time to eradicate this stupid prejudice that is sexism, from our society.

10. References

[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.