

Master in Computational Social Science  
2022-2023 Academic Year

*Final Master’s Thesis*

“Fair play? Detecting and monitoring hate speech during football events”

Álvaro Sanz Castañeda

Tutor

Iñaki Úcar Marqués

Madrid, XX/XX/XXXX



Esta obra se encuentra sujeta a la licencia Creative Commons

**Reconocimiento – No Comercial – Sin Obra Derivada**

**ABSTRACT**

Social Media Listening techniques have gained relevance during the last decade due to their advantages for brands and institutions in terms of informative gathering and strategic decision making. In this work, we combine Data Mining, Machine Learning and Data Visualization techniques to answer our central question: does the celebration of football matches, correlate with an uptick in the proportion of online hate speech?

By taking the Spanish National Team’s participation in the 2022 Qatar World Cup as our case study, we retrieve more than 4 million tweets sent during its participation and build a Supervised Machine Learning binary classifier to categorize each one as hateful or not. With this information we construct meaningful visualizations and perform statistical analyses to address this potential correlation, also ‘zooming in’ on Spain’s last game to observe what events may trigger discriminatory messages.

**Key words:** Hate speech, Machine Learning, social media, football, Spanish National Team

**RESUMEN**

Las técnicas de Escucha en las Redes Sociales han ganado relevancia durante las últimas décadas dadas sus ventajas para empresas e instituciones en términos de recopilación de información y toma de decisiones estratégicas. En el presente trabajo, combinaremos técnicas de Minería de Datos, Machine Learning y Visualización para responder a nuestra pregunta central: ¿Correlaciona la celebración de partidos de fútbol con un aumento en la proporción del discurso de odio en Internet?

Tomando como caso de estudio la participación de la Selección Española en el Mundial de Qatar 2022, recuperamos más de 4 millones de tweets enviados durante su participación y construimos un Clasificador binario a través de Supervised Machine Learning para categorizar cada uno de ellos como relativo al discurso de odio o no. Con esta información, construimos visualizaciones y realizamos análisis estadísticos, haciendo también ‘zoom’ sobre el último partido jugado por España para observar qué tipo de eventos pueden desencadenar mensajes discriminatorios.

**Palabras clave:** Discurso de odio, Machine Learning, Redes Sociales, fútbol, Selección Española

**DEDICATORIA**

-

CONTENIDOS

1. Introduction
2. State of the art
3. Methodology
4. Results and discussion
5. Conclusions & Future Works
6. Bibliography
7. Annex
8. INTRODUCTION

a

1. STATE OF THE ART

a

1. METHODOLOGY
   1. Operationalization and hypotheses

Before describing our set of hypotheses, it’s necessary to adequately define what is hate speech and what characteristics define it, as it will later be a central part of stages like Tweet annotation. From now on, we’ll be using the definition given by Waseem and Hovy, as it gives a clear set of possible orientations of the messages classified as hateful:

“A tweet is offensive if it

1. uses a sexist or racial slur.

2. attacks a minority.

3. seeks to silence a minority.

4. criticizes a minority (without a well founded argument).

5. promotes, but does not directly use, hate speech or violent crime.

6. criticizes a minority and uses a straw man argument.

7. blatantly misrepresents truth or seeks to distort views on a minority with unfounded claims.

8. shows support of problematic hash tags. E.g.”#BanIslam”, “#whoriental”, “#whitegenocide”

9. negatively stereotypes a minority.

10. defends xenophobia or sexism.

11. contains a screen name that is offensive, asper the previous criteria, the tweet is ambiguous

(at best), and the tweet is on a topic that satisfies any of the above criteria.” (2016, p.XXXX).

As previously mentioned, we’ll take as case study the Spanish National Team participation in the World Cup: this includes the 3 matches played in the Groups Stage and the first Knockout Round. In order to have our data contextualized, we’ll set our temporal frame from 3 days before the first match until 3 days after the last one, which leaves 20th of November of 2022 and 9th of December of the same year as the limits. The objective with this approach is to have a ‘basal’ level of hate the days before with which to compare to the time after them and to be able to capture the potential lasting effect days after. In that period, Spain played 4 matches:

• 23/11 vs. *Costa Rica*

• 27/11 vs. *Germany*

• 01/12 vs. *Japan*

• 06/12 vs. *Morocco*

For each of them, for each match we’ll consider the tweets 2 hours prior to its start, 2 hours for the match itself and 2 hours after the end (3 hours in the case of Spain vs. Morocco, as it went to overtime and penalties) as our objective of study. Only using the tweets sent during the game itself would be reductionist, as the diffusion of hateful messages may not only be limited to the time between the first and last whistles.

Our dependent variable will be the percentage of hateful tweets relative to all the tweets sent during that span of time, and we’ll address two main hypotheses:

H1. Moments before, during and after the celebration of football matches, the proportion of tweets categorized as hateful increases significantly in comparison to time spans when no matches are being played.

H2. This effect increases significantly when the supporter’s team does not get a win and when the rival team comes from a different continent.

* 1. Data Collection & Annotation

The first logic step to start our work is to construct the dataset with which we’ll later try to extract conclusions from. To achieve this we performed a Data Mining process interacting with the Twitter Academic API which resulted in 4.5M tweets, each one with information regarding their author, the time when the tweet was sent, its language, the users mentioned… Among many others.

The extraction process followed a two-way schema to ensure capturing as much football conversation as possible, which consisted of:

1. Query search: searching by the names, surnames, and Twitter handles of all the football players involved, by the countries played, their demonyms and national teams accounts and by some general football terms, applying both geographical and language filters.
2. Semi-random users from @SEFutbol: we retrieved information for the Spanish National Team 2.7M followers, to later apply some filters regarding number of tweets sent and creation date in order to avoid inactive users and bots. With the resulting users who met the criteria, we ran a random sampling process of 30.000 of them and we retrieved all their tweets between the time span delimited.

All queries and code are available in Annex X. We compiled all files into a single .csv file resulting in our 4.5M tweets dataset.

(Sample of some tweets)

As our objective is to run a Supervised Machine Learning task, it’s compulsory to have some tweets already tagged as hateful or not in order to build the training and testing splits for the model selection. As other authors have already stated (CITAR), the percentage of hateful tweets in regular conversational topics, such as the one we’re working with, is not usually very high (this doesn’t mean that it’s not important), so we face an unbalanced classes problem. To address this challenge we again combined randomization (2000 random tweets) with specific terms searching, such as common racial, homophobic or sexist slurs, sampling some of the tweets containing them (3000 in total) and binding them all in a single dataframe.

We generated a ‘hate’ variable and manually annotated them as hateful (1) or not hateful (0) according to the criteria expressed in Section 3.1. The final distribution of the 5000 tweets ended up like this:

(FRECUENCIAS)

With this annotation process we tried to ensure that there were sufficient hateful tweets for further algorithms to be trained with but not too many of them, as our objective of study is essentially an unbalanced distribution and it could bias these models in the next steps. Some examples of the tweets annotated are:

(EJEMPLOS)

* 1. Data Preprocessing & Feature engineering

Once we already have our set of tweets with the desired variables, we must adapt the data to our specific needs for Text Classification. This includes transforming the texts from the tweets using Data Wrangling techniques and creating some variables when they meet some criteria.

In first place, we remove duplicated tweets (we may have captured the exact same tweet two or more times, when applying different queries, if they met two different conditions, or the same text repeated over time by an automatic account) and generate binary variables for Mentions, URLs and Hate terms presence.

We perform additional text transformations, starting from lower-casing all tweets, removing symbols and special characters like “\n”, “@” or “#” and transforming certain emojis to text, so that the algorithms then capture all of them depending on their emotions. Twitter users were left as text without its handle as we thought they could be a nice predictor for cases of famous football players when being massively attacked in social media for a determinate reason.

|  |  |
| --- | --- |
| **Emojis** | **Transformation** |
| 🤣😂 | “tokenrisa” |
| 🐀 | “tokenrata” |
| 😡🔪🤬👊🪓🤮💩 | “tokenenfado” |
| 👏💪✊💪❤️🥰❤🤍 | “tokenpositivo” |

Other emojis or non UTF-8 characters have been removed, as they offered little value to our analytic purposes. The next step in this process was to add to “tokenrisa” all possible Spanish laughs combinations, for which we employed an online consulted Regular Expression (CITAR).

* 1. **Document Features Matrix & Natural Language Processing**

Text Classification tasks differ substantially from other predictive approaches in the input given to the algorithm: while for traditional classification or regression predictions variables tend to be numbers and factors, in this case the features are all the words used in the set of tweets. Manually building a dataframe with all words used in our 4,5M tweets isn’t feasible due to computational constraints maybe, but also because it’s not practical nor efficient.

Instead, we converted our set of tweets to a ‘quanteda’ (a R library) corpus object, taking as input each tweet as a document and the already generated features as document variables. This allows us not only to have the ‘bag-of-words’, but also those additional factors created in the previous Section.

However, the corpus object can’t be introduced as input to the algorithms, so we need to transform it to a ‘Document Features Matrix’ (‘DFM’, from now on), a kind of intermediate step between the dataframe and the quanteda default object. When constructing it, we’ll split each tweet in tokens (minimal analysis units), removing punctuation, numbers and separators while doing it. In our case, we will be using unigrams (words) as our tokens’ unit: character n-grams were computationally expensive for such a large set of texts and word n-grams (two, three or more words altogether) gave worse results in terms of predictions.

After splitting all tweets in single words, we removed stopwords from the “ntlk” package list for the Spanish language, to reduce the inputs given to the models which offer low or none information and that just make processes longer. We also performed an ‘stemming’ process to the words remaining: ‘stemming’ a word means reducing it to its lexical root, so that words coming from the same base are computed as equal features. This will improve modeling performance and reduce unnecessary sparsity in the ‘DFM’.

Once all these previous steps are done, we transform the corpus into ‘DFM’, computing the TF-IDF for each feature appearing: The TF-IDF goes beyond mere word frequencies and also measures the term’s importance relative to the other documents. Even if we previously removed stopwords, we apply some trimming to only leave words present in at least 5 tweets (2 in the training and testing splits, as the total amount is much lower), to avoid misspellings or other strange terms from biasing the analysis.

* 1. Model building & Selection

Once the previous operations have been applied to both the full dataset and the manual annotated subset, we search for the model that will later classify all tweets as hateful or not. To do so, we firstly use the ‘set.seed()’ function followed by a random split of 4500 tweets to train and 500 to test so that we obtain the same divisions in any execution of the code.

Giving to the ‘caret’ package a quanteda object as input has a particularity: we need to create a common ‘DFM’ with matched features between the train and test splits. The TF-IDF values will be 0 for those terms not present at all in the testing dataframe, but this is compulsory to give the model the same input (this step will also be necessary in Section 3.5.). Additionally, we create an object with the actual class of each document/tweet, so that we can later check the model’s performance.

We performed a 5-fold cross-validation process with successive models, setting each one’s specific parameters and setting “ROC” as the metric to tune the hyperparameters to. Algorithms like ‘Random Forest’, ‘Logistic regression’ or ‘Neural Networks’ were very computationally expensive in this case and were discarded because of the need to later classify millions of cases.

The algorithm with best performance, both in terms of Area Under the Curve and time, was ‘XGBoost’. We predicted with the trained model setting the output as probabilities, not the class directly, so we could set the threshold that better fit for our purposes and that correctly balanced False Positives and False Negatives, with a value of 0.3. This means that if the predictive task assigned a tweet a probability of more than 0.3 of being hateful, it would be classified as so.

The Confusion Matrix of the model is shown on Figure X.X. and reflects the following metrics:

|  |  |
| --- | --- |
| **Metric** | **Value** |
| Accuracy | 0.8598 |
| Balanced Accuracy | 0.7727 |
| Sensitivity | 0.6168 |
| Specificity | 0.9286 |
| Precision | 0.7097 |
| Area Under the Curve (threshold independent) | 0.8865 |

In addition, we constructed another corpus with two additional dataframes of 4196 and 4500 already annotated tweets, offered by XXXX and XXXX, respectively, coming from previous works of online hate speech detection using Supervised Machine Learning techniques. However, the performances of the model didn’t improve and the measured metrics were all worse, which gives us a valuable clue: the annotation part is critic and very context-dependent, as hateful tweets captured from other temporal frames and from different conversations may bias the algorithms and make the performance decrease.

* 1. Full dataset classification

Once the model has been tested and its performance is proved, we advance to the next phase: applying it to our whole set of tweets to give to each one its hateful or not label. As specified in previous sections, the pre-processing phase was exactly the same except for the ‘trimming’ function, in which we only left features present in 5 or more tweets to avoid misspellings or uninformative words.

When trying to run the model for the more than 4 million tweets, we faced some obvious computational constraints regarding the available RAM memory of our domestic laptop: the vector that R Studio was trying to allocate was 161.2Gb big. To avoid this problem, we constructed a loop in which successive iterations of 50000 tweets were classified using the model, with the following sequence:

1. Before setting the loop, keep open the training DFM, establish a batch size of 50000 tweets and create an empty dataframe in which we’ll store all predictions, with two variables (“No” and “Yes”).
2. We extract the 50000 corresponding tweets for each iteration from the full DFM, with all the tweets, into another DFM (from now on, ‘dfmat\_matched’).
3. We match the features between all the tweets contained in ‘dfmat\_matched’ and the training DFM, so that the model has that input too (if not, it would fail).
4. We predict using our pre-loaded model (‘XGBoost’) with the DFM resulting from step 3 as new data, specifying that the output should be probabilities too.
5. We store this predictions into the empty dataframe from step 1 and repeat successively binding all rows. As each iteration of 50000 does not apply random sampling, but instead takes tweets in order from the beginning to the end, there’s no problem identifying each prediction with its tweet.

Once this loop is over, we have an ‘all\_predictions’ dataframe with the same number of rows as the initial dataframe. To associate each tweet with its final label, it’s necessary to generate a ‘hate’ factor variable with value “Yes” when the “Yes” probability is higher than the set threshold, 0.3, and we bind this column with the initial set of tweets.

* 1. Visualization & Analysis

The final step from our Working Pipeline is the Visualization of the data obtained and its analysis through statistical methods to address our initial hypotheses. We’ve employed the ‘ggplot2’ R package to construct two graphics:

* A general one, in which the whole evolution relative to the whole temporal period is shown.
* An specific one showing the evolution in the Moroccan’s case, in order to distinguish if there are some types of event that triggered upsides in hate speech or if there’s no clear pattern.

To answer our hypotheses we made use of Inferential Statistics through successive t-tests

1. RESULTS AND DISCUSSION

a

1. CONCLUSIONS & FUTURE WORKS

a

1. BIBLIOGRAPHY

a

1. ANNEX

a