HUrtful HUmour (HUHU): Detection of humor spreading prejudice on Twitter

José Fco. Olivert
José Baixauli
Kexin Jiang

May 2023

Contents

1 Introduction

In the ever-evolving landscape of natural language processing (NLP), one of the most critical challenges is the detection of hurtful humor, an issue that has gained significant attention in recent years. With the proliferation of social media and online platforms, it is crucial to develop robust NLP models that can accurately identify and mitigate harmful or offensive content. In this article, we proudly present our remarkable achievements in a recent competition focused on hurtful humor detection. Through rigorous research, innovation, and the application of cutting-edge NLP techniques, our team has excelled in tackling this complex task. Our primary goal was to develop an advanced NLP model capable of discerning between humor that is genuinely light-hearted and that which crosses the line into hurtful territory. By effectively harnessing the power of machine learning and linguistic analysis, we aimed to create a tool that can promote a safer and more inclusive digital environment for all users. Throughout the competition, our team meticulously constructed a state-of-the-art NLP pipeline, incorporating various stages such as data preprocessing, model selection, and fine-tuning. Leveraging large-scale datasets and leveraging pre-trained language models, we focused on developing a robust and scalable solution that could generalize well to real-world scenarios. The backbone of our approach revolved around the creation of a comprehensive and diverse dataset specifically curated for hurtful humor detection. This ensured that our model was trained on a wide range of instances, allowing it to capture intricate patterns and variations in offensive language usage. We are thrilled to report that our efforts bore fruit, as our NLP model achieved good performance in two of the tasks of the competition. By leveraging advanced deep learning architectures and innovative feature representations, our solution exhibited superior capabilities in distinguishing between harmless jokes and those that could potentially cause harm or offense. In this article, we will delve into the technical details of our winning approach, shedding light on the unique methodologies and techniques we employed. Furthermore, we will discuss the implications of our research, exploring how our breakthroughs in hurtful humor detection can contribute to building a safer, more inclusive digital space.

2 Data preprocessing

The first step in every NLP task is to understand and preprocess the data you have. We have a dataset containing tweets in spanish. We removed all the emojis and special characters of the tweet but then we noticed that most of the tweets had some dialogues that could include a joke inside the tweet. Due to this we decided to use all the text of each tweet in order to not to lose any useful information.

In order to preprocess the data we have used different tools:

- First we used a tokenizer to transform the raw data into input ids, tokens and attention mask. This part was useful for the first subtask we will talk about later.
- Secondly we precomputed the embeddings with a pretrained model in order to feed other models and train with this data instead of the raw text.
- Thirdly we did a sentiment analysis of the data. We used a module named "pysentimiento" to subtract the scores for different sentiments in the text such as joy, anger or disgust.

With all this additional data and features for each tweet we will try different approaches in the different subtasks that we will comment later on.

In order to train and test our models we have divided our train dataset into train and test doing a 80-20% split. We have also used a k-fold cross validation in order to see the training process of our models.

3 Subtask 1. HUrtful HUmour Detection

The first task of this competition was a simple binary classification task in order to classify the tweets in two labels: Hurtful humor or just humor. This was the task that we spent more time on in order to compare different approaches and get the best results. We tried to tackle the problem using different techniques:

3.1 Embeddings + Sentiment analysis + Classical ML techniques

This approach was the first one we did. We used a huggin face model in order to compute the embeddings for each tweet. Later we subtract the sentiments of each sentence using what we explained in the Data preprocessing Section. When we had all of this we trained some basic classifiers like SVM and MLP, with the embeddings plus the scores for each sentiment. This approach did not get bad results but we thought we could do it better.

3.2 Recurrent Neural Network approach

In this approach we tried to build a kind of a pipeline using Recurrent Neural Network as the main layer in our model. The input of this model is the text tokenized by "PlanTL-GOB-ES/roberta-base-bne" tokenizer. The first layer is a LSTM. This first layer returns an output of the shape 60 (max length of the tweet) * 128 (number of hidden layers). Then we use linear layers and dropouts in order to condense the information and get a final output of 2 dimensions of a Softmax. It was very interesting to try to implement this approach because it was something new for us. On the other hand, it was a limitation too due to the lack of knowledge using this technique.

3.3 Fine tuning BETO and RoBERTa

This approach was based on using the BETO and RoBERTa transformers in order to predict the label. The input to this model was the tokenized text and the attention mask. Just like we explained in the last approach we implemented linear layers and dropouts in order to condensed the information. The last layer was a Softmax 2-dimensional to get the probabilities for both of the labels.

3.4 Final Model Selection and HUHU results

The last approach was the one that got the best results. We compared using the two models in order to choose one but they were very similar. We sent the two runs of this task using both of the models of the last approach and we got the best results with RoBERTa, achieving a 0.779 of f1 score and a 10th place in the competition.

4 Subtask 2a. Prejudice Target Detection

In this subtask, our objective was to correctly classify if the tweet was committing prejudice against women and feminists, the LGBTIQ community, immigrants and racially discriminated people or overweight people . So we had a multilabel problem of 4 classes. We implemented a One vs All strategy using both bagging and boosting techniques. We used the F1 macro to assess the model's overall performance as it calculates the F1 score for each class independently and then takes the average of these scores, giving equal importance to each class. Furthermore, we utilized the embeddings and sentiment analysis techniques, which were previously described, as our input data (X_data)

4.1 BaggingClassifier

Bagging classifier for multilabel classification is an ensemble learning method that combines multiple base classifiers to make predictions on multiple labels simultaneously. Preprocessing the data, we tried different base classifiers and found that the best outcome was provided using:

- $baseclassifier : RandomForestClassifier$
- $nestimators : 10$

Once this was evaluated with the validation data the F1 macro was 0.55.

4.2 XGBClassifier

The second technique we tried was a XGBoost (eXtreme Gradient Boosting) Classifier which is a powerful algorithm for multilabel classification. It builds an ensemble of decision trees using gradient boosting, sequentially improving the model by correcting the mistakes of previous trees. We used the same preprocessing steps as before and obtained an F1 macro of 0.78 in the validation data.

4.3 Final Model Selection and HUHU results

We evaluated different classification models for the multilabel task, and XGBoost Classifier outperformed the others, yielding significantly higher F1 macro scores. As a result, we chose XGBoost Classifier as our final model for this particular subtask. We did not consider using Bagging Classifier in subsequent runs because the difference in F1 macro scores between Bagging and XGBoost was substantial, indicating that XGBoost provided superior performance for our specific problem. Our final model achieved the 31st position in the rankings, with a f1-score of 0.569, indicating that there is still considerable room for improvement.

5 Subtask 2b. Degree of Prejudice Prediction

In this subtask, our objective was to predict the degree of prejudice in a given tweet on a continuous scale from 1 to 5, where 1 represents the least prejudicial and 5 represents the most prejudicial. To tackle this problem, we employed two different regression algorithms: XGBRegressor and AdaBoostRegressor.

5.1 XGBRegressor

XGBoost (eXtreme Gradient Boosting) is an advanced implementation of the gradient boosting algorithm. It is known for its efficiency and effectiveness in solving various machine learning problems, including regression tasks. We chose the XGBRegressor for its ability to handle a large number of features and its robustness against overfitting. The preprocessing steps are the ones explained above in section 2. We performed hyperparameter tuning using GridSearchCV to find the optimal values for the XGBoostRegressor. The best hyperparameters obtained were:

- $nestimators$: 1000
- $maxdepth$: 7
- $learningrate$: 0.1
- $subsample$: 0.8

With these optimal parameters, we trained the XGBoostRegressor on the training set and evaluated its performance on the validation dataset. The model achieved a Root Mean Squared Error (RMSE) of 0.50.

5.2 AdaBoostRegressor

AdaBoost (Adaptive Boosting) is an ensemble learning algorithm that combines multiple weak learners to create a strong learner. It works by iteratively adjusting the weights of the training instances and the base learners to minimize the error. We chose the AdaBoostRegressor for its ability to improve the performance of the base estimator and its adaptability to different problems. Similar to the XGBoostRegressor, we used the same preprocessing steps, which is explained in more detail in section 2.

For the AdaBoostRegressor, we used the DecisionTreeRegressor as the base estimator. We performed hyperparameter tuning using GridSearchCV to find the optimal values for the AdaBoostRegressor. The best parameters obtained were:

- $nestimators$: 100
- $learningrate$: 0.05

With these optimal parameters, we trained the AdaBoostRegressor on the training set and evaluated its performance on the validation set. The model achieved a Root Mean Squared Error of 0.517.

5.3 Unused model: Voting Regressor

In addition to the XGBoostRegressor and AdaBoostRegressor, we also experimented with a Voting Regressor model. The Voting Regressor is an ensemble model that combines the predictions of multiple base estimators to produce a final prediction. The idea behind this approach is to leverage the strengths of different models to improve overall performance. However, in our case, the Voting Regressor did not yield satisfactory results, so we decided not to use it for the final submission. The base estimators we used for the Voting Regressor were:

- Linear Regressor
- Decision Tree Regressor
- Random Forest Regressor

We used the default parameters for the base estimators and set the 'weights' parameter to 'unifroma' to give equal importance to each estimator's prediction. We trained the Voting Regressor on the training set and evaluated its performance on the validation set. The voting Regressor achieved a Root Mean Squared Error (RMSE) of 0.55.

5.4 Final Model Selection and HUHU results

After evaluating the performance of the XGBoostRegressor, AdaBoostRegressor and Voting Regressor, we compared their RMSE values on the validation set. And we selected the two models with the lowest RMSE value as our final models for the Degree of Prejudice Prediction subtask, which are XGBoostRegressor and AdaBoostRegressor.

In the HUHU competition, we achieved a RMSE of 0.909 using the AdaBoostRegressor, placing us 12th in the ranking. This result indicates that our approach was successful in predicting the degree of prejudice in tweets, although there is still room from improvement. We were quite surprised with this result since we thought that XGBoostRegressor would have achieved better results than AdaBoostRegressor.

6 Conclusions

Overall, our participation in the HUHU competition allowed us to apply and consolidate the knowledge acquired during the course. We are very happy with what we have accomplished and our performance in the competition demonstrates the potential of our approaches in addressing the challenges of detecting humor, prejudice targets, and the degree of prejudice in tweets. However, there is still room for improvement, and future work could focus on refining our models and exploring additional techniques to enhance their performance. For the first task we could have added the sentiment analysis into the model in order to get better results. The lack of time and experience working with this complex model has made the process really tricky. For the second task, XGBRegressor and AdaBoostRegressor are powerful models, but they can be sensitive to hyperparameters and prone to overfitting. A more thorough hyperparameter tuning process or the use of regularization techniques could potentially improve the models' performance.

You can check our code in the link below.[1]

[1]https://github.com/pepe-olivert/beto-roberta-rnn