PROJECT - NATURAL LANGUAGE AND INFORMATION RETRIEVAL

Hurtful HUmour (HUHU) Detection of humor spreading prejudice on Twitter

Authors:
Adam Steciuk
Sara Echary

INTRODUCTION

The purpose of this report is to delve into the use of humor as a means of conveying prejudice against marginalized communities, with a particular focus on Spanish tweets. Our goal was to examine tweets that show prejudice against women and feminists, the LGBTIQ community, immigrants, racially discriminated people and overweight people. The project consisted of 3 subtasks in which we were able to focus on in turn:

SUBTASK 1: HURTFULHUMOUR DETECTION

The primary objective of this subtask was to determine whether a given tweet that contains preemptive content is intended to be humorous. Participants must distinguish between tweets that use humor to convey bias and those that express biases devoid of humor.

SUBTASK 2A: PREJUDICE TARGET DETECTION

In this subtask, the task was to identify specific target groups within the aforementioned minority categories (women and feminists, LGBTIQ community, immigrants, racially discriminated and overweight people) for each tweet.

SUBTASK 2B: DEGREE OF PREJUDICE PREDICTION

The third subtask involved predicting, on a continuous scale of 1 to 5, the average level of prejudice conveyed in messages targeting various minority groups.

PREPARATION

PREPROCESSING

The initial step in building a machine learning system involves loading the training dataset. Next spacy tokenizer model was loaded. We used "es_core_news_lg". By using the default pipeline of the spacy model, we obtained tokens with information about their part of speech, lemma, whether they are a stop word, etc.

We created an utility function that easily allows us to filter the tokens based on their attributes. The consists functionalities such as: filtering out all the punctuation marks, stopwords (based on nltk spanish stopwords corpora), emojis, numbers, newline characters, whitespaces, URLs, and twitter-specific mentions and hashtags. This utility allows as to build and experiment with different representations of the data in the next steps of the project.

TF-IDF

To implement TF-IDF, the Scikit-learn library provides the classes TfidfTransformer and TfidfVectorizer. These classes offer efficient mechanisms for transforming raw text data into TF-IDF representations, enabling further analysis and modeling tasks. To calculate TF-IDF, we remove stopwords, punctuations, emojis, numbers, newlines and spaces. We didn't use this approach for the final models as it wasn't providing any satisfactory results.

FASTTEXT

FastText was used as an algorithm for building word embedding. Pre-trained word embeddings were downloaded from dccuchile/spanish-word-embeddings. According to josecannete/spanish-corpora, the corpus on which the FastText embeddings were trained, was processed in the following way: lowercase, removed url, replaced multiple spaces with single one, so in order to get the best results we did the same. Urls in the training set were replaced with the string `URL` but we removed them anyway. The same was done for hashtags and mentions.

After obtaining the words within a text and constructing embedding vectors, it becomes possible to represent a sentence using the word vectors contained within it. The approach we used to achieve sentence representation was averaging of the vectors of all the words present in the sentence.

DESCRIPTION OF THE MODELS

Hyperparameters play a crucial role in machine learning classifiers, as they define the behavior and configuration of the model. To fine-tune hyperparameters of the models, we decided to use GridSearchCV with 5-fold cross validation. This technique involves exhaustively considering all possible combinations of hyperparameter values within a predefined range or set of options. By systematically evaluating each combination, GridSearchCV helped us identify the best hyperparameter configuration that maximizes the model's performance.

The utility for loading and saving models was created, as like utility for dataset balancing.

In order to prevent the possibility of overfitting we also divided the training data into training-validation and test sets according to 80-20 ratio.

During the project we tested three different approaches:

- FastText embeddings with classic ML models,
- BETO representation of the tweets with classic ML models,
- RoBERTa fine-tuned for specific downstream tasks.

As BETO representation we decided to truncate and optionally add padding to 35 tokens for each tweet as during analysis we discovered that 75% of tweets don't exceed 31 tokens.

In this report we will not dive into what hyperparameters were tuned and why, to not make it make the report to lengthy, but all the details could be found in the project notebook: LNR-hurftul-humor-detector/project.ipynb.

SUBTASK 1: HURTFUL HUMOUR DETECTION

For this subtask we evaluated the performance of the following scikit-learn ML classifiers using FastText embeddings

1. LogisticRegression,
2. SVC,
3. DecisionTreeClassifier,
4. KNeighborsClassifier,
5. RandomForestClassifier,
6. VotingClassifier of the models above.

We did the same for tweet representations obtained with BETO (with exclusion of voting classifier). Those models were chosen as they can be considered as a good representation of all the most popular ML models families.

We also try fine-tuned RoBERTa model for this specific task.

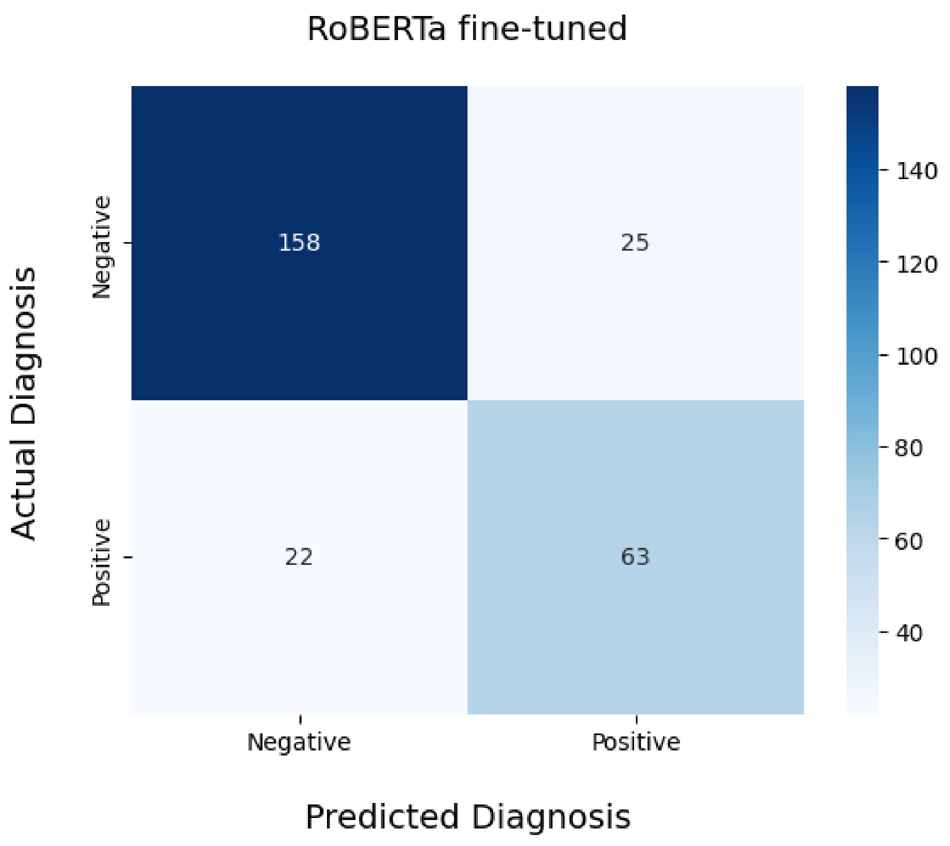
When it comes to the problem of imbalanced data, we found out that the best results are obtained by using class weighting approach.

Results obtained on the separated test set:

Model	Accuracy	Precision	Recall	F1	ROC AUC
FastText + SVC	0.776	0.610	0.708	0.655	0.756
FastText + LogisticRegression	0.770	0.594	0.745	0.661	0.763
FastText + DecisionTreeClassifier	0.639	0.420	0.522	0.465	0.606
FastText + KNeighborsClassifier	0.811	0.763	0.540	0.633	0.734
FastText + RandomForestClassifier	0.755	0.582	0.665	0.620	0.729
FastText + VotingClassifier	0.791	0.669	0.602	0.634	0.737
BETO + SVC	0.809	0.667	0.733	0.698	0.788
BETO + LogisticRegression	0.759	0.578	0.733	0.647	0.751
BETO + DecisionTreeClassifier	0.621	0.423	0.714	0.531	0.647
BETO + KNeighborsClassifier	0.779	0.652	0.571	0.609	0.720
RoBERTa fine-tuned	0.825	0.716	0.741	0.728	0.802

The best f1-score was obtained by fine-tuned RoBERTa model. This is not surprising as it is considered as state-of-the-art model for NLP tasks. The fine-tuned RoBERTa model was used for the final predictions.

Confusion matrix obtained on the test set by fine-tuned RoBERTa:



Confusion matrices for other model can be found in the project notebook: LNR-hurftul-humor-detector/project.ipynb.

Best score during laboratories: **0. 728 f1**

Official results for subtask 1: **0.744 f1**

Subtask 2A: Prejudice Target Detection

For this task we decided to use two methods

1. ExtraTreesClassifier
2. RandomForestClassifier

because they are both ensemble methods that are inherently suitable for multilabel classification. Similarly to the first subtask, both were trained and evaluated on FastText and BETO representation.

The difference between both models is that ExtraTreesClassifier is choosing random split among the features resulting in being less sensitive to noise in the data and usually faster training time. The downside is that it may require more trees to achieve the same performance as RandomForestClassifier.

Unfortunately, we didn't manage to try a fine-tuned transformers pipeline for this subtask because we lack the time to do so.

Model	**Accuracy**	**Weighted F1**	**F1 Micro**	**F1 Macro**
FastText + ExtraTreesClassifier	0.413	0.469	0.554	0.351
FastText + RandomForestClassifier	0.456	0.512	0.591	0.392
BETO + ExtraTreesClassifier	0.310	0.354	0.442	0.231
BETO + RandomForestClassifier	0.338	0.373	0.467	0.246

The best weighted f1-score was obtained by RandomForestClassifier with FastText embeddigs, so this combination was used to generate the final predictions.

Unfortunately in this subtask it was necessary to decrease/delete the number of tuned hyperparameters due to restricted time for the delivery. It is evident that ExtraTreesClassifier performed worse than RandomForestClassifier. The reason for that may be the mentioned above fact that ExtraTreesClassifier may require more trees to achieve same performance as RandomForestClassifier.

When it comes to models trained from BETO representation, the results are worse than the ones obtained from FastText embeddings. We suspect that the reason for that is that we didn't tune the hyperparmeters which resulted in models trained from higher dimensional data to perform worse.

Best score during laboratories: **0.512 weighted f1**

Official results for subtask 2A: **0.325 weighted f1**

Subtask 2B: Degree of Prejudice Prediction

For this subtask we decided to try the following models:

1. SVR,
2. ElasticNet,
3. SGDRegressor,
4. GradientBoostingRegressor,

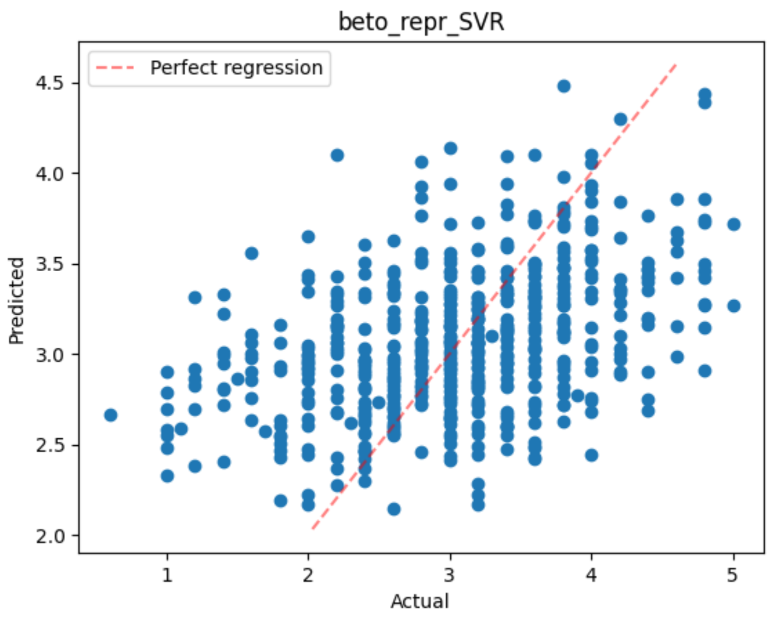
both for FastText embeddings and BETO representation. These models were chosen because they offer a diverse range of the most popular techniques used for regression modeling. SVR handles high-dimensional and non-linear data with robustness to outliers. ElasticNet combines Ridge and Lasso regression for feature selection and prevention of overfitting. SGDRegressor is computationally efficient and adaptable to different loss functions. GradientBoostingRegressor captures non-linear relationships and handles missing data very well.

For this task we didn't manage to try any fine-tuned transformers models because of the time constraints.

Model	R2	RMSE
FastText + SVR	0.165	0.761
FastText + ElasticNet	0.010	0.829
FastText + SGDRegressor	0.097	0.791
FastText + GradientBoostingRegressor	0.133	0.775
BETO + SVR	0.178	0.755
BETO + ElasticNet	0.144	0.770
BETO + SGDRegressor	0.091	0.794

On the test set the best performing model was SVR with BETO representation. It outperformed the best performing (on the K-Fold validation) model - SVR with FastText embeddigs. It indicates that the BETO representation is more capable of capturing the degree of prejudice in the text than FastText embeddings without overfitting.

Below we can see predicted results for each sample from the test dataset plotted against the real labels.



Trend plots for other models can be found in the project notebook: [LNR-hurftul-humor-detector/project.ipynb.](LNR-hurftul-humor-detector/project.ipynb)

Best score during laboratories: **0.755 RMSE**

Official results for subtask 2B: **0.920 RMSE**

SUMMARY

In our opinion, the biggest challenge posed by the competition was the time constraints, limiting our exploration of alternative approaches. Nonetheless, we found the diversity of the tasks involved to be enlightening, exposing us to various methodologies for solving NLP problems. This experience has broadened our understanding of the topic and provided valuable insights for future projects in this field.