Building an NLP Model to Detect Offensive Language in Football Social Media Comments

Seminar Thesis

in the context of the Seminar "*Natural Language Processing for Business Analytics*"

at the Faculty of Business, Economics, and Law

Friedrich-Alexander-Universität Erlangen-Nürnberg

Intelligent Information Systems

Prof. Dr. Patrick Zschech

Supervisor:	Prof. Dr. Mathias Kraus
	Prof. Dr. Patrick Zschech
Presented by:	Paul Roth (22688773)
	Jan Baumgart (22541155)
	Akshun Dogra (23134385)
	Ali Kirac (23359350)
	Burak Cinar (23359365)
Submission:	2024-03-01

Table of Contents

Figures

Tables

Abbreviations

AI	Artificial Intelligence
BERT	Bidirectional Encoder Representations from Transformers
DF	Data Frame
NLP	Natural Language Processing
OLR	Ordinary Logistic Regression
PRAW	Python Reddit API Wrapper
SVMs	Support Vector Machines
TOS	Terms of Services

Symbols

v3	3rd generation (version)

1 Introduction

Natural language processing (NLP) is a machine-learning technology that allows computers to interpret, manipulate, and comprehend human language. Nowadays, there is an increasing number of larger organizations that are using Artificial Intelligence (AI) and handling large amounts of data of voice and text data from various communication channels like emails, text messages, social media newsfeeds, video, audio, and more. To tackle this situation, various organizations use NLP to automatically process this data, analyze the intent or sentiment in the message, and respond in real-time to human communication (Oracle 2022).

Football is the most popular sport globally and holds a significant value in their fans' hearts. Usually many people follow a sport so closely, they tend to express their feelings when their teams win or lose. In addition, social media has allowed people to express their feelings and connect with others to discuss football. However, football is a sport where one team will win and the other loses and few people do not take it well when their team loses. As a result, offensive language in online forums, particularly in the context of sports comments, has become a significant concern due to its potential to incite violence, discrimination, and hostility.

The increase in offensive language on online platforms poses a serious challenge to all companies in maintaining a inclusive and respectful community on their social media platforms. Sometimes it has caused outright violence and riots due to some fans fighting among themselves on these platforms, which raises the question, how can social media platforms fix this issue?

1.1 Project Goal

This project aims to develop an offensive language detection model in the football media comment section. We used NLP techniques to analyze a diverse dataset of football-related comments which are marketed as offensive language. In addition to that, we will implement various NLP models, such as classification and BERT or an Ordinary Logistics Regression Model to improve our performance. We also compared our models by having datasets from different leagues and national football matches, so that we can understand how offensive language differs during different football events such as two clubs competing in a league match or a match between two countries competing in world cup.

1.2 Project Managment

Our goal is to create an NLP project that would detect offensive words. However, we all worked in a group for the first time and all five team members worked together to reach our goal. We divided our tasks into various sub-tasks. We also had a constant weekly meeting where we discussed our progress and challenges that we faced during the project. We used various tools to maintain seamless and transparent communication such as Microsoft Teams as our main project management and communication tool and using WhatsApp for quick conversation. There were various other software's that we used to make our model such as anaconda, visual studio, GitHub, so that all our team members could work simultaneously in real time. Moreover, the consultancy lecture and our preliminary presentation helped us to constantly improve our model. We received valuable insight from the professor for example, we made a specific guideline regarding labelling our data, after receiving feedback in our preliminary presentation.

1.3 Business Understanding and Research

Before implementing our model, understanding the existing information on how offensive language is in context with football is crucial. There are various methods for detecting offensive language on social media platforms. One approach involves keyword-based filtering, where predefined lists of offensive words are used to flag potential offensive language. However, this method struggles to capture the intent of language and context, often missing implicit offensive language and not understanding harmless phrases.

Machine learning offers a better approach than keyword-based filtering, training algorithms on labeled data sets to identify patterns associated with offensive language. We used various models like Support Vector Machines (SVMs) and Naive Bayes Classifiers to classify new content.

Despite these approaches there remain several challenges. For example: Data scarcity in the form of large and diverse labeled datasets for training models is difficult to generate with less resources. Additionally, context-dependency can be an issue, as offensive language often relies heavily on the underlying intent and surrounding context, which can be difficult for NLP models to grasp until and unless the data is correctly labelled. Finally, the constant evolution of language, with new forms of offensive language and symbols emerging continuously, necessitates ongoing adaptation and training of these detection models.
In the next chapter we will discuss more about our data and how we scrape the data and what were the various limitation and challenges we faced during web scrapping.

2 Data Understanding / Web Scraping

2.1 Data Limitations

First, we narrowed down the scope of the data that needs to be scraped. Our goal is to only scrape comments or (comment) posts that are limited to football topics or their environment. This restriction is aimed at ensuring the quality and relevance of the data, which will help us to achieve our project goal of identifying offensive language specifically in football.

We initially expected a comment quantity / Dataset size of approx. 2000, as these also had to be labeled by us in the later course of the project before the model training.

An attempt was made to diversify the data sources and distribute them across as many platforms as possible. However, this plan was partially thwarted by restrictions on the part of the platforms or timewise, as explained in more detail in the following sections.

2.2 Data Sources

The Dataset on which this project is based was scraped by us over the course of the project. As a source, we searched various online platforms, preferably social media platforms known for their user-generated content. Among the platforms we looked at more closely are YouTube, Instagram, Facebook, and Reddit. These were chosen for the reason mentioned above, but especially because they are platforms that are known for reflecting many different opinions in discourse and comment. X (formerly Twitter) was not considered further at the beginning of the project due to time scheduling for the project.

On top of this, we have also included a small Dataset from Kaggle, but this does not make up a large part of our entire Dataset. This mainly serves the purpose of giving our Dataset something like a guideline and diversifying the vocabulary and word choice a little. This should lead to even better prediction results (Samoshyn 2020).

When it comes to expressing opinions and collecting data, the META group's platform network cannot be ignored. Its social media platforms Facebook and Instagram are among the largest in the world and generate significant amounts of data traffic. When trying to get the comments under posts via the Graph API of Facebook and Instagram, we faced several problems. On the one hand, Meta's Graph API is a complex topic, which a part of the group tried to understand and spent the first weeks with. This also threw us behind schedule without providing us with any added value in the form of data. On the other hand, we found out that we can only load comments and posts from our own pages or posts from the meta platforms legally as a normal user. We suspect that there are ways to get special tokens for the API and also get other comments, but this is only a guess and not our knowledge at the end of the project.

It also came to our mind to get the comments from posts and pages via simple web scraping. However, this turned out to be a much more complex task, as Instagram and Facebook are very dynamic in the way they build their pages, and it is difficult to load the comments there by automated scraping of posts. But above all, it is also against Meta's Terms of Services (TOS), which strictly prohibit web scraping. From this point onwards, we no longer dealt with the meta platforms, as we were already able to load comments from the other platforms at this point. We therefore decided to continue with these.

The web video social media platform YouTube, which has enabled people to share almost any kind of video with the world since 2005, has also been included in our list of sources. In particular, we used commentaries from videos of football match summaries. In addition, attention was paid to a mix of international matches and national matches in the respective leagues, with the majority of the league matches being from the English Premier League. Technically we used the YouTube Data API v3 here, which uses the unique identifiers (videoID) of a video to retrieve and store the comments of the video. The more detailed technical process of the scraping procedure is described in the next chapter.

Reddit has been one of the world's largest platforms for the exchange of information between users on all kinds of topics since 2005. There are also areas that specialize solely in football (e.g. r/football or r/soccer). In February 2024, Reddit was the 16th most visited website in the world, making it the perfect place to find comments of all kinds (Similarweb 2024). For collecting comments from soccer game threads on reddit we have developed a script that uses the Python Reddit API Wrapper (PRAW). A Reddit account with its login data is also required for identification in order to use PRAW. To access the desired threads on Reddit, you also need a unique ID which can be found in the URL.

In the following, we will take a closer look at the Data Collection and the underlying scripts.

2.3 Data Collection

Now that we've narrowed down the sources, we'll go into more detail about the code for scraping data from YouTube and Reddit. In the following We will split up both sources and go through each of the codes descriptively.

2.3.1 YouTube

First, we are going to analyze and go through the Python Script we developed with the YouTube Data API v3, that has been used to access and retrieve comments on Videos from soccer matches.

```
import googleapiclient.discovery
import googleapiclient.errors
import pandas as pd
```

First, we had to import the “googleapiclient” Package for Python, which is used to interact with e.g. YouTube and videos on the platform. We also imported pandas to further interact with the data that we receive.

```
youtube = googleapiclient.discovery.build("youtube", "v3", developerKey="[redacted]")
```

Next, we initialize the Google v3 API client. We adjusted it, so that can use it for scrapping from YouTube, for that a developer key is needed as well (blurred here for Data Privacy).

```
videoID = ""


with open(r"videos_used.txt", "a") as file:
    file.write(videoID + "\n")
```

In this part we use the videoID of a specific soccer match review from YouTube and save it into a string variable. Then the Script opens a text file named ‘videos_used.txt’ in append mode, there it writes the VideoID of the used Video, so that we can keep track of which videos we already used so far.

```
request = youtube.commentThreads().list(
    part = "snippet",
    videoId= videoID,
    maxResults = 100
)

response = request.execute()
```

Next, we have the main part of the script: Here we construct a request to the API to retrieve comments from the Video we defined earlier. The ‘part’ parameter specifies the data that should be retrieved, ‘snippet’ means that basic information (incl. the content) of the comment is loaded. The ‘maxResults’ does what its name indicates, it defines how many comments should be scrapped. In this case we load the content from the first 100 comments from the videoID. After that we execute the request and store response data in the variable ‘response’.

```
comments = []

for item in response['items']:
    comments.append(item['snippet']['topLevelComment']['snippet']['textDisplay'])
```

In the next part we define a list with the name ‘comments’, which has the purpose to store all the comments that we stored earlier in the ‘response’ variable.

After that we access each comment snippet we scraped earlier and within this snippet we access the ‘topLevelComment’ which basically gives some order to the comments. We also access the ‘textDisplay’ which is the actual content / text of the comment. Lastly, we append the content of each comment to the ‘comments’ list.

```
df = pd.DataFrame(comments)
```

In the next step we just convert the comments list into a Pandas Data Frame (DF) to further process it with Pandas functions.

```
df['source'] = 'youtube'
df['label'] = ''
df['type'] = 'league'
```

At the end we add 3 new columns to the DF:

- Source: adds a column that contains the note that the comment comes from YouTube.
- Label: adds an empty column that will later be used for the labeling process.
- Type: adds the note whether the comment is from an international or a league match. (needs to be manually adjusted)

```
df.to_csv('yt_comments_new.csv', index=False, mode="a")
```

To end the script, we append the comments that we scrapped into a CSV file that stores our Dataset.

2.3.2 Reddit

```python
import praw
from praw.models import MoreComments
import pandas as pd
import os
```

This code imports the PRAW library, a Python wrapper for the Reddit API, allowing access to Reddit data such like comments from a thread. It also imports the Morecomments model from PRAW which helps with identifying what content is an actual comment and thus reduces the load on the server accessed by the API. The Pandas library is also imported for the same purpose as before.

```python
reddit = praw.Reddit(client_id='[redacted]', client_secret='[redacted], user_agent='NLP')
```

This Part of the script initializes a connection to the Reddit API using the provided client ID, client secret (blurred for Data Privacy) and user agent. This enables us to get access to Reddit content such as comments.

```python
redditIds = [""]
comment_data = []
```

In this part of the code we enter one or more unique IDs from Reddit threads (comment page for a topic like a specific football game). We also initialize an empty list ‘comment_data’ to store comments that we collected from Reddit.

```python
comment_limit = 300

for redditId in redditIds:
    post = reddit.submission(id=redditId)
    comments = post.comments
    comment_count = 0

    for comment in comments:
        if isinstance(comment, MoreComments):
            continue
        if len(comment.body.split()) < 80:
            comment_data.append({'Comment': comment.body})
            comment_count += 1

            if comment_count >= comment_limit:
                break
```

This loop iterates over each Reddit ID we entered in the last step. It then retrieves the called Thread and its comments, there it collects comments that have less than 80 words, to avoid very long comments and keep the dataset slim and easier to label. We also make sure with the comment_limit that we scrape exactly 300 comments per Thread.

```
df = pd.DataFrame(comment_data)
df['label'] = ''
df["source"] = 'reddit'
```

Next, we create a Pandas DF from the comment_data list, which contains all the Reddit Comments. We also add two new columns that serve the same purpose as the ones from the YouTube data.

```
with open("Reddit_IDs.txt", "a") as file:
    file.write(redditId + "\n")
```

This block of code opens a file named "Reddit_IDs.txt" and adds the current Reddit ID to it. This logs the Reddit ID for tracking purposes.

```
if not os.path.exists("redditComments.csv"):
    df.to_csv("redditComments.csv", sep=';', index=False, mode='w', header=True)
else:
    df.to_csv("redditComments.csv", sep=';', index=False, mode='a', header=False)
```

As the last step we first check if the ‘redditComments.csv’ already exists.

- If the file does not exist, it writes the DF to the CSV file, including the header.
- If the file already exists, it appends the DF to the existing CSV file without adding the header again.

This ensures that the header is added only once, preventing duplication when the code is executed several times.

We have now explained the scripts for recording the comment data and in the next step we will discuss which and how much data we have loaded from which sources into our data set.

2.4 Data Volume and Attributes

In total our data set, which we use to train the models, consists of **3862** comments after the cleaning process, that will be discussed later in next chapter. The Count of Comments per source can be seen in the chart below.

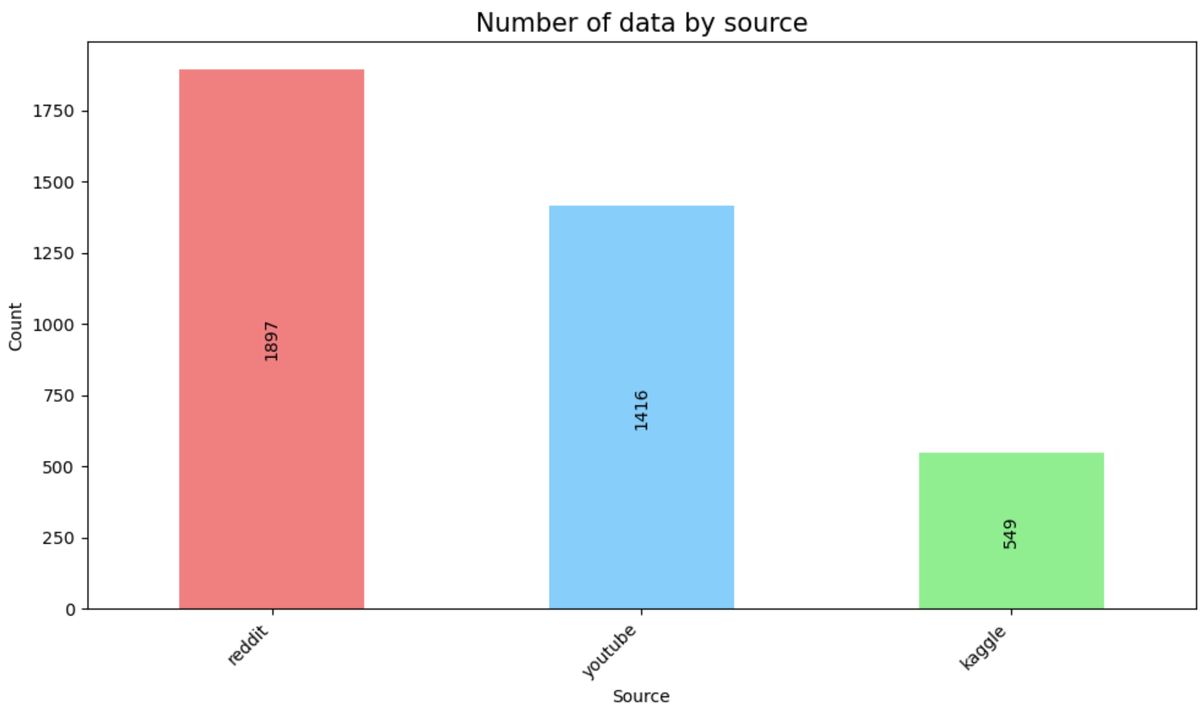


Figure 1: Bar Chart for total Amount of Data per Source

Now below we can also see the distribution of the dataset shares from the 3 sources:

We can see that Reddit makes up to roughly half of the data, that is because we discovered during the scraping that we can find more samples for offensive language there. This is a consequence of the automatic filters that YouTube is already increasingly using to delete offensive language.

YouTube also makes up the second largest share with 36.7% of all comments in the dataset.

And as intended the proportion of Kaggle data is very limited. Since our goal was mainly to collect our own data.

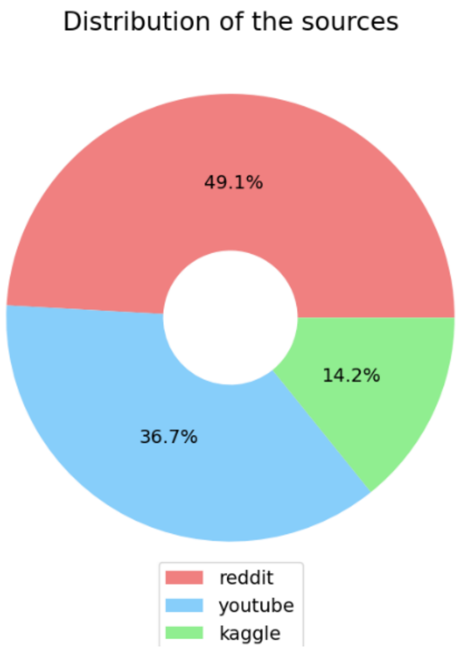


Figure 2: Pie Chart for the Distribution of Sources

After the labeling process (which will be explained in more detail later) was completed, you can now see that a large part of the comments in the dataset cannot be categorized as offensive language. This was to be expected, as the comment sections are already kept clean by filters and of course most people don't just use offensive language in their comments.

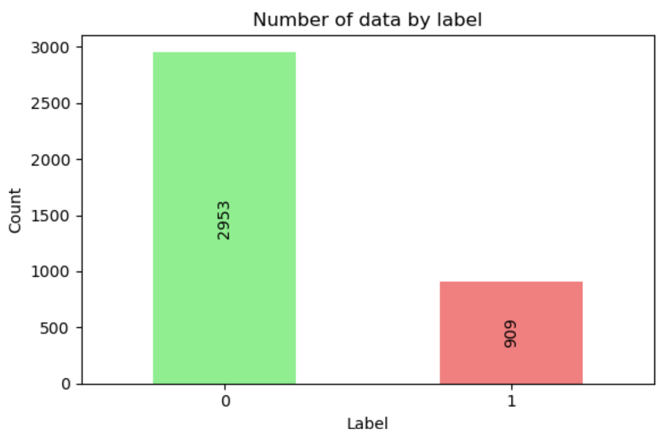


Figure 3: Bar Chart for the Amount of Data per Label

Additionally below can also be seen the percentage of comments from the respective sources that are considered offensive. There you can see that over half of our offensive comments (481 / ~ 53%) come from the designated Kaggle dataset. This underlines Kaggles relevance, as the vocabulary of offensive language could otherwise be too small in the model later on to detect such comments.

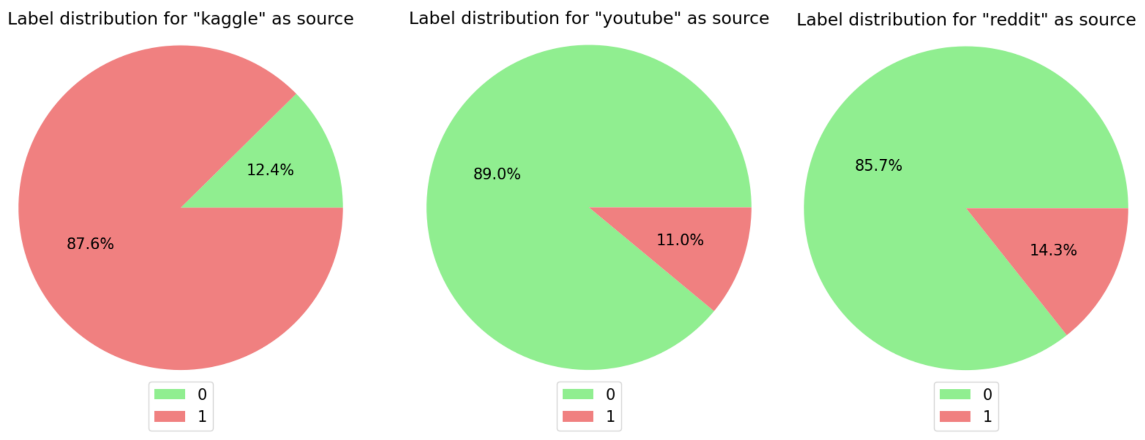


Figure 4: Pie Charts for the Distribution of Labels per Source

The following chapter discusses the topic of data preparation further and examines how the data was cleaned and processed before it was used by the model.

3 Data preparation

3.1 Labelling Guidelines

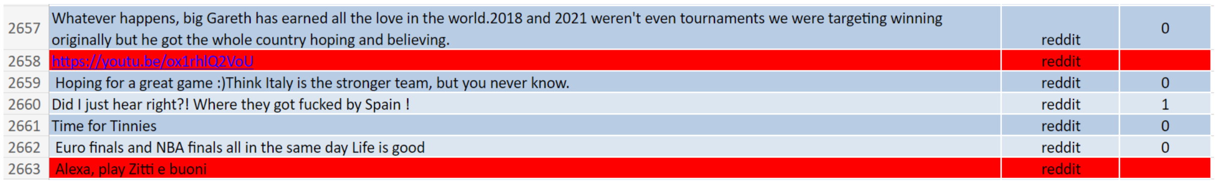
Data preparation is an essential step in natural language processing projects, particularly when using models like the Bidirectional Encoder Representations from Transformers (BERT). This step involves cleaning and preprocessing raw data in order to make it ready for analysis and training. Ensuring the quality and effectiveness of the analysis is ensured by adhering to explicit labeling guidelines. Process ambiguity and inconsistencies can be caused by imprecise instructions or a subjective interpretation of the data that degrades the performance of the model. Our first effort had many problems because it didn't have clear labeling instructions.

We revised our strategy after the initial presentation, taking into account further feedback from our professors and peers, realizing the significance of precise labeling guidelines. This led to the creation and implementation of precise offensive language guidelines for labeling that have characteristics specific toward football. Following these instructions helped us reduce biases, strengthen the analysis's resilience, and provide the machine learning model a more comprehensible learning process. The Appendix contains these explicit guidelines along with examples of how they can be applied.

3.2 Labelling Process

A crucial was merging the data from YouTube, Kaggle, and Reddit into one Excel file after exporting the scraped comments. The process of preparing the data was streamlined by this consolidation, enabling comprehensive analysis and labeling. We opted for using the Excel format due to its easy readability and accessibility and to perform some data cleaning steps simultaneously while labelling the data. Considering that we had to read the comments during the labeling process regardless, this method worked well for efficient data handling.

We distributed tasks evenly among team members, utilizing our combined knowledge as data scientists, in order to efficiently manage the labeling workload. This task delegation promoted accountability and teamwork, guaranteeing that every comment was carefully considered and given the proper label. Furthermore, we understood the importance of removing undesired comments from the dataset. We improved the quality and relevance of the dataset by removing comments that were written in non-English and that had no meaningful content (such as one-word entries, single symbols, or HTML code). In order to preserve the data's integrity and prevent noise in the later analysis these steps were highly necessary.



2657	Whatever happens, big Gareth has earned all the love in the world.2018 and 2021 weren't even tournaments we were targeting winning originally but he got the whole country hoping and believing.	reddit	0
2658	https://youtu.be/ox1rhlQ2VoU	reddit	
2659	Hoping for a great game :)Think Italy is the stronger team, but you never know.	reddit	0
2660	Did I just hear right?! Where they got fucked by Spain !	reddit	1
2661	Time for Tinnies	reddit	0
2662	Euro finals and NBA finals all in the same day Life is good	reddit	0
2663	Alexa, play Zitti e buoni	reddit	

Using an Excel file that was saved in the MS-Teams OneDrive space was an excellent method to increase productivity and collaborate. The simultaneous viewing and editing of the dataset by several team members allowed for real-time updates and seamless collaboration. Additionally, the Pandas module made it easier to import and export CSV and Excel files, which was useful for data manipulation tasks.

```
df.to_csv('yt_comments_new.csv', index=False, mode="a")
```

3.3 Data Cleaning for BERT

As part of the cleaning process tailored for BERT, we recognized the significance of emojis in social media comments as they often convey sentiments and emotions. For instance, offensive language directed towards people of color is frequently expressed on social media using the monkey emoji. We transformed emoji code into alphabetical expressions using the python emoji library to guarantee consistent emoji treatment. Furthermore, we checked the length of each comment since the limitation imposed by BERT of a maximum 512 token input. By ensuring that the input data complies with BERT's specifications, this preprocessing step facilitates effective model training and inference.

Finally, the BertTokenizer.from_pretrained function was used to carry out the tokenization process, which was an essential step in preparing the comments for further analysis and smooth model integration. Transforming textl data into a format compatible with BERT models, the BERT tokenizer must be used. The main benefits of the tokenizer are its large vocabulary, context-sensitive representations, and subword tokenization, all of which have been painstakingly tailored for BERT's architecture. Initializing: divides the input text into subword tokens.

```
model_name = "bert-base-uncased"
tokenizer = BertTokenizer.from_pretrained(model_name)
model = BertForSequenceClassification.from_pretrained(model_name, num_labels=2)
```

This allows to map the tokens to their corresponding IDs within the model's vocabulary. This meticulous process eventually improves the model's overall performance, guaranteeing reliable and accurate results in downstream tasks.

3.4 Data Cleaning for the Ordinary Logistics Regression Model

To guarantee the reliability and precision of our dataset, we undertook rigorous text cleaning procedures during the data preparation phase. Once more making use of the emoji library, we started translating emojis into text. After doing this, we looked into finding patterns in the comments. We found several patterns using regular expressions, including user handles, hashtags, URLs, punctuations, and particular numbers. After these patterns were found, they were extracted for examination. Then, in order to prevent these patterns from interfering with further processing stages, we undertook the systematic removal of them.

```
# Remove punctuations
df.replace(r'[^\w\s]', '', regex=True, inplace=True)

# Remove digits
df.replace(r'\d+', '', regex=True, inplace=True)

# Replace underscores with spaces
df.replace(r'_', ' ', regex=True, inplace=True)
```

Moreover, standard text preprocessing techniques were applied to refine the comments further, including decapitalization to standardize case, tokenization to break down comments into individual tokens, and removal of stopwords to focus on meaningful content. Additionally, special characters, including non-alphanumeric symbols, were eliminated to streamline textual data, while digits were extracted to maintain semantic integrity.

```
# Tokenize using Tokenizer
tokenizer = WordPunctTokenizer()
df['comment'] = df['comment'].apply(tokenizer.tokenize)
```

4 Modelling

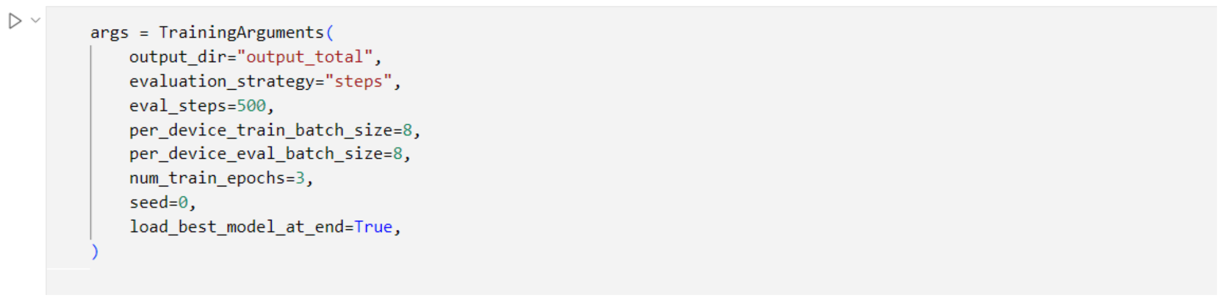
4.1 BERT Model

4.1.1 Selection and Architecture

The main reason we chose BERT base model for our offensive language detection project is its state-of-the-art performance. BERT, a transformer-based model, is known with its remarkable capabilities in capturing contextual information, which makes it appropriate for understanding nuances in comments on social media.

4.1.2 Training Strategy

The success for an NLP project is mostly dependent on well-defined training strategy. In our case, we adopted a strategy with specific training arguments tailored to optimize the BERT model for offensive language detection. The key training arguments include:



```
args = TrainingArguments(
    output_dir="output_total",
    evaluation_strategy="steps",
    eval_steps=500,
    per_device_train_batch_size=8,
    per_device_eval_batch_size=8,
    num_train_epochs=3,
    seed=0,
    load_best_model_at_end=True,
)
```

Evaluation Strategy and Steps: For frequent assessment of the model's performance during training, we set the evaluation strategy to occur every 500 steps. This approach helps in monitoring the model's progress and produces early identification of potential issues.

Batch Size: Both training and evaluation were performed with a per-device batch size of 8. This parameter affects the amount of data processed in each iteration and has implications for memory usage and training speed.

Number of Training Epochs: The model was trained for 3 epochs, striking a balance between capturing sufficient patterns in the data and avoiding overfitting. The number of training epochs is a critical parameter that influences the convergence and generalization of the model.

Random Seed: Setting a random seed to 0 ensures reproducibility, allowing others to replicate the experiments and results. Reproducibility is vital for establishing the credibility and reliability of the developed model.

Loading the best Model at the End: Enabling this option ensures that the model saved at the end of training corresponds to the point where the evaluation metric is at its best. This helps in preventing overfitting and obtaining a more robust model.

4.1.3 Cross-Check-Validation

Overfitting is caused by the model memorizing the training data instead of learning the more-general mapping from features to labels. While there are some techniques to prevent overfitting, it is almost impossible to detect before testing the data. In order to rigorously assess and validate the performance of the developed offensive language detection model, a robust evaluation methodology was employed known as k-fold-cross-check validation where the dataset is divided into k subsets (folds), and the model is trained and evaluated k times, each time using a different fold as the test set and the remaining folds for training.

In our project, a 3-fold cross-check validation strategy was chosen to strike a balance between computational efficiency and obtaining reliable performance estimates. This strategy involves dividing the dataset into three equal-sized folds, training the model on two folds, and evaluating it on the remaining fold. This process is repeated three times, ensuring that each fold serves as the test set exactly once.

4.1.4 Motivation For Cross-Check Validation

Reduced Variance: With the results from different datasets Cross-check validation helps reduce the variance in performance estimates by averaging results over multiple models. This provides more robust and reliable numbers.

Identifying Overfitting: One of the most popular ways to detect overfitting is cross-check validation. The use of cross-check validation helps, ensuring that the model does not memorize specific patterns in the training data but instead learns the underlying linguistic structures.

4.1.5 Implementation

```
X_train_1 = X_1 + X_2
y_train_1 = y_1 + y_2
X_test_1 = X_3
y_test_1 = y_3
```

```
X_train_2 = X_1 + X_3
y_train_2 = y_1 + y_3
X_test_2 = X_2
y_test_2 = y_2
```

```
X_train_3 = X_3 + X_2
y_train_3 = y_3 + y_2
X_test_3 = X_1
y_test_3 = y_1
```

After randomly dividing dataset into three subsets, we assigned different subsets as test and train datasets. The evaluation metrics (including accuracy, precision, recall, and F1 score) were computed for each model and then we computed average scores to see a comprehensive performance. This approach allowed us to assess the model's consistency across different subsets of the data.

4.2 Ordinary Logistics Regression Model

4.2.1 Benefits

The model employed for offensive language detection in football comments is a logistic regression model, specifically designed for binary classification tasks. When predicting a binary outcome based on one or more predictor variables, a common machine learning application is logistic regression, a straightforward but effective statistical model. Our code was inspired by a contribution on Kaggle (prakharprasad 2021).

Logistic regression makes sense in the context of detecting offensive language in football comments for a number of reasons. First of all, logistic regression can analyze textual data represented as feature vectors because it is skilled at handling high-dimensional data. Using methods such as TF-IDF (Term Frequency-Inverse Document Frequency) vectorization, the raw text comments can be transformed into a feature matrix, which then allows logistic regression to capture the underlying patterns and associations between words and their corresponding labels.

Furthermore, situations where interpretability and explainability are critical are a good fit for logistic regression. Understanding the factors impacting the model's predictions and identifying the key features that influence the classification choice are crucial in the context of offensive language detection. Interpretability and insights into the underlying mechanisms of offensive language detection are facilitated by the coefficients that are produced by logistic regression. These coefficients indicate the strength and direction of the relationship between each feature and the target variable.

4.2.2 Building

There are several crucial steps involved in developing the logistic regression model for detecting offensive language in football comments. The dataset is first divided into input features (X) and the target variable (y), which stand for the labels assigned to the comments, respectively. A train-test split is then used to further split the data into training and testing sets, enabling evaluation and validation of the model. We decided to split the data according to the industry standard 80% train and 20% test split.

```
# split the data into training and testing sets
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=1)
original_X_test = X_test
print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)
```

After that, TF-IDF vectorization is used to convert the textual data into a numerical format appropriate for modeling. Through this process, the importance of each word in the corpus in relation to the entire dataset is captured, converting the raw text comments into a sparse matrix of TF-IDF features.

```
# use TfidfVectorizer to convert the raw documents into feature matrix
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer(max_features=5000)
X_train = vectorizer.fit_transform(X_train).todense()
X_test = vectorizer.transform(X_test).todense()
print(X_train.shape, X_test.shape)
```

Lastly, the training data is used to instantiate and train the logistic regression model. Based on the features that are extracted, the model learns to predict the likelihood that a comment will belong to a specific class (such as offensive language or non-offensive language). The performance and capacity for generalization of the trained model are then assessed using the testing data.

```
# instantiate the model (using the default parameters)
logreg = LogisticRegression()
logreg.fit(np.asarray(X_train), np.asarray(y_train))
train_predictions = logreg.predict(np.asarray(X_train))
test_predictions = logreg.predict(np.asarray(X_test))
```

Regularization is a critical aspect of model training that helps prevent overfitting by penalizing overly complex models, thus encouraging simpler and more generalizable solutions. Hyperparameter tuning, on the other hand, involves optimizing the parameters of the model to achieve the best performance on unseen data.

Our logistic regression model was subjected to regularization, namely L1 and L2 regularization, in order to reduce overfitting and enhance its capacity to generalize to previously unseen data. By introducing a penalty term based on the absolute value of the coefficients, L1 regularization—also referred to as Lasso regularization—encourages sparsity in the model by shrinking some coefficients to zero. Conversely, L2 regularization, also known as Ridge regularization, reduces the impact of outliers by penalizing the squared magnitude of the coefficients, resulting in more robust and stable models.

```
# regularization and hyperparameter tuning
from sklearn.model_selection import GridSearchCV
# define the grid
param_grid = {'C': [0.001, 0.01, 0.1, 1, 10, 100, 1000]}
# instantiate the model
logreg = LogisticRegression(class_weight=weights)
# instantiate the grid search model
grid_search = GridSearchCV(logreg, param_grid, cv=5)
# fit the grid search to the data
grid_search.fit(np.asarray(X_train), np.asarray(y_train))
# print the best parameters
print(f"Best Parameters: {grid_search.best_params_}")
```

Grid search, an approach that methodically investigates a variety of hyperparameters to find the best combination for the model, was used to tune the hyperparameters. In our instance, we changed the logistic regression regularization parameter C, which controls the degree of regularization. We were able to determine the ideal regularization strength that maximizes performance on the validation data by evaluating the model's performance at various values of C. The model with the best performance, as determined by a predefined evaluation metric (e.g., accuracy, F1 score), was selected as the optimal model.

```
#instantiate the best model
logreg = LogisticRegression(C=10, class_weight={0: 1.0, 1: 13}, penalty='l1', solver='liblinear')
```

4.3 Results and Evaluation

4.3.1 Results of BERT Model

In this part we are going to mention two different results of cross-check validation. The reason that we have two different results is after we first trained our model, as we are going to mention below, we tried to improve our model especially in terms of precision. The way we have chosen to do so is scraping and adding more data to our dataset.

Results of First BERT Model

	Accuracy	Precision	Recall	F1
Training 1	0,910	0,861	0,788	0,823
Training 2	0,958	0,922	0,934	0,928
Training 3	0,904	0,857	0,822	0,839
Average	0,924	0,880	0,848	0,863

Table 1: Results of first BERT model on different test dataset

Even though the results seem promising, when we used our model to make predictions on data, which was not used as test or train data in the training process, we found an interesting pattern. When we check the comments that are wrongly predicted as non-offensive (False negatives), some of them are examples of vivid offensive language. It is necessary to note that test data has some comments that are written by us, and they are explicit offensive language. The comments below are false-negative and written by us comments:

- why dont you just go back to where you came from
- the more i watch the more i want to kill
- man this was a shitty team
- did you see those untalented bitches
- i gotta say that I hope your career crashes and burns. You don't deserve success

We wanted to improve our model in terms of detecting explicit offensive comments. We thought that the reason is that our data we used to train our model has much more non-offensive comments than offensive ones. (1679 non-offensive, 707 offensive comments) So we concluded that we need more offensive comments. After we scraped some data our new dataset had 2928 non-offensive and 907 offensive.

Results of Second BERT Model

	Accuracy	Precision	Recall	F1
Training 1	0,898	0,877	0,671	0,760
Training 2	0,929	0,813	0,912	0,860
Training 3	0,987	0,966	0,980	0,973
Average	0,938	0,885	0,854	0,864

Table 2: Results of second BERT model on different test dataset

As we can see, training model with more data gave almost the same results. After we used second model on same data we got similar results. False negatives of the two models are similar.

At first glance, our models failed to detect offensive language even if it is explicit.

	Accuracy	Precision	Recall	F1
Average	0,849	0,667	0,500	0,571

Table 3: Results of second BERT model on new data set which includes comments other than social media sources

However, when we try predicting social media comments we face the following results.

	Accuracy	Precision	Recall	F1
Average	0,982	0,875	0,875	0,874

Table 4: Results of second BERT model on a new dataset which includes only social media comments

We concluded that the BERT model gives meaningful results when it predicts comments that are already filtered by social media platforms’ filters.

4.3.2 Takeaways From Comparing the two BERT Models

Importance of Data

When we first trained our model, even though the scores are satisfying, we were not happy when we found out that our model cannot detect offensive language in the comments we have written. We decided to use more data to train model in order to eliminate this. We increased our offensive comment from 700s to 900s. Still, we had similar results. Detecting was still a problem. Then, we recognized that we missed a point. The false negatives were not real social media comments, so they were not subject to filters social media platforms are using. We concluded that our model predicts best for the comments that were already filtered by social media platforms.

Place For Improvement

Our model has still some issues with detecting unexplicit and content dependent offensive languages. And if derogatory words are used in a non-offensive way, our model has a hard time predicting it right.

Example: “*Africa vs africa”*

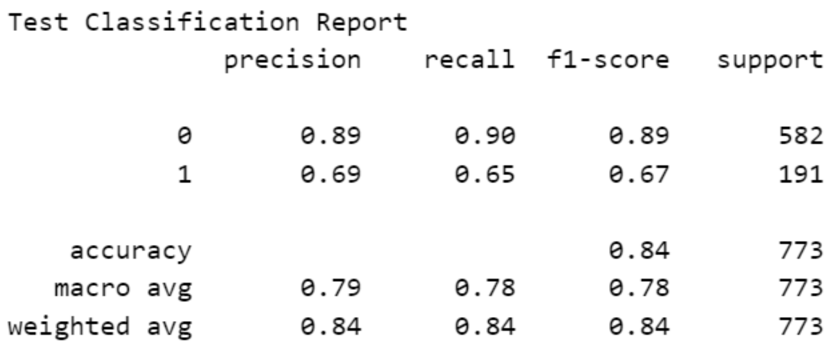
For this comment we labeled it as an offensive comment since it is written under an international game post between England and one African country. Content-wise it has racial implications. Our model predicted it as non-offensive.

Example: *“Jude f#%^*ng Bellingham…. What a player!!!* ❤*”*

Even the word “fucking” used in this comment, we can easily understand that the comment has no offensive meaning. Our model predicted this as offensive.

4.3.3 Results of Ordinary Logistic Regression (OLR) Model

Classification reports and confusion matrices, which are used to assess the performance of our offensive language detection model, offer important insights into how well it distinguishes between offensive and non-offensive comments.

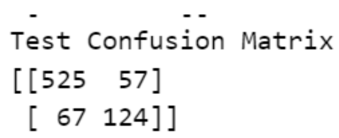


```
Test Classification Report
              precision    recall  f1-score   support

           0       0.89      0.90      0.89       582
           1       0.69      0.65      0.67       191

    accuracy                           0.84       773
   macro avg       0.79      0.78      0.78       773
weighted avg       0.84      0.84      0.84       773
```

As we assess the model's performance on the test set, we find that the F1-score, precision, and recall for both classes are acceptable. The recall is at 0.90, suggesting that some non-offensive comments were mistakenly classified as offensive, even though the model maintained a high precision of 0.89 for non-offensive comments, indicating that the majority of comments classified as non-offensive were indeed non-offensive. Similarly, for offensive comments, the recall has a value of 0.65, meaning that some offensive comments were not correctly identified by the model, while the precision is at 0.69, meaning that some comments that were classified as offensive were actually not. The model does an adequate job of identifying instances of offensive language in football comments, according to the confusion matrix. Relatively high numbers of true positives (124) and true negatives (525) demonstrate the model's success in correctly identifying instances of non-offensive language as well as in detecting offensive language. Nevertheless, false positives (57) and false negatives (67) indicate that misclassification still occurs occasionally.



```
  -        --
Test Confusion Matrix
[[525  57]
 [ 67 124]]
```

Overall, there is space for improvement, particularly in decreasing false positives and false negatives, even though our offensive language detection model performs well in accurately identifying offensive comments. The model's performance and robustness in practical applications may be improved with more tweaking and optimization, which may involve incorporating more sophisticated methods like ensemble learning and further adjusting the hyperparameters. However, the model's ability to identify offensive language in football-related comments is a promising first step toward creating a more welcoming and safe online community.

4.4 Evaluation of BERT and OLR-Model

When we compare the results from both models, we can say that the results are satisfying. Even though the BERT model has better results, with 84% the OLR Model can be considered as a trustworthy Model. Especially, when we take the straightforward deployment and simple model structure of the OLR into account, its less susceptibility of overfitting makes it an appropriate Model. On the other hand, the BERT model has a better understanding of textual nuances, which can be improved in our case, and with its pre-trained representations it can handle complex tasks.

As Conclusion, both models can be used according to needs. In the case of Logistic Regression, with its interpretability, efficiency and scalability it can provide valuable insights and perform in scenarios where the complexity of more advanced models may not be warranted or feasible. BERTS’s deep architecture, bidirectional context understanding, and pre-trained representations make it particularly well-suited for tasks like offensive language detection, where context, nuance, and adaptability to complex language patterns are crucial for achieving high performance.

5 Model Application

As we wanted to analyze the amount of offensive language in football comments, we decided to scrape new data from YouTube and labeled the data according to comments on international games and league games. From the analysis conducted on the new data using our models, we gained valuable insights into the distribution of offensive language across different types of comments, particularly distinguishing between international and league-related comments.

5.1 BERT-Model Results

Our analysis of offensive comments in league and international football matches provides valuable insights into the prevalence of toxicity within different contexts. According to the predictions made by our BERT model, there were 5 offensive comments out of 851 international match comments and 44 offensive comments out of 545 league matches. While international matches have less than 1% offensive comments, league matches have 8%.

5.2 OLR Results

Looking at the overall distribution of labels, we observe that the majority of comments (approximately 92%) are labeled as non-offensive (label 0), while a smaller proportion (approximately 8%) are classified as containing offensive language (label 1). This indicates that offensive language is relatively infrequent in the dataset.

Numerically, league-related comments have a higher count of offensive language, with 81 comments labeled as offensive, compared to 59 offensive comments in international comments.

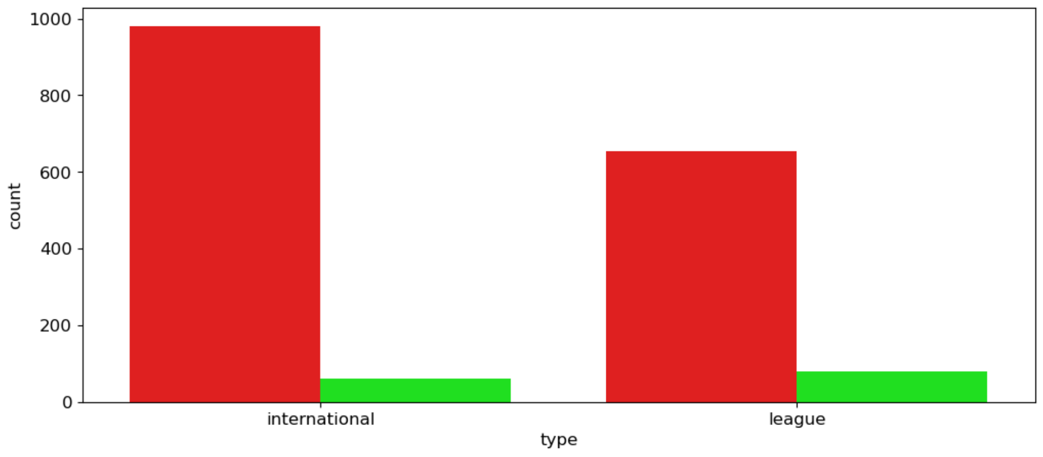


Figure 5: Bar Chart of the total Number of Comments and the offensive Comments (per type)

When examining the breakdown of offensive comments by type, we find that league-related comments tend to have a higher proportion of offensive language compared to international comments. Specifically, approximately 11.05% of comments related to league comments are labeled as offensive, whereas only about 5.67% of comments related to international comments contain offensive language. This suggests that league-related topics may elicit more contentious or emotionally charged responses compared to international topics.

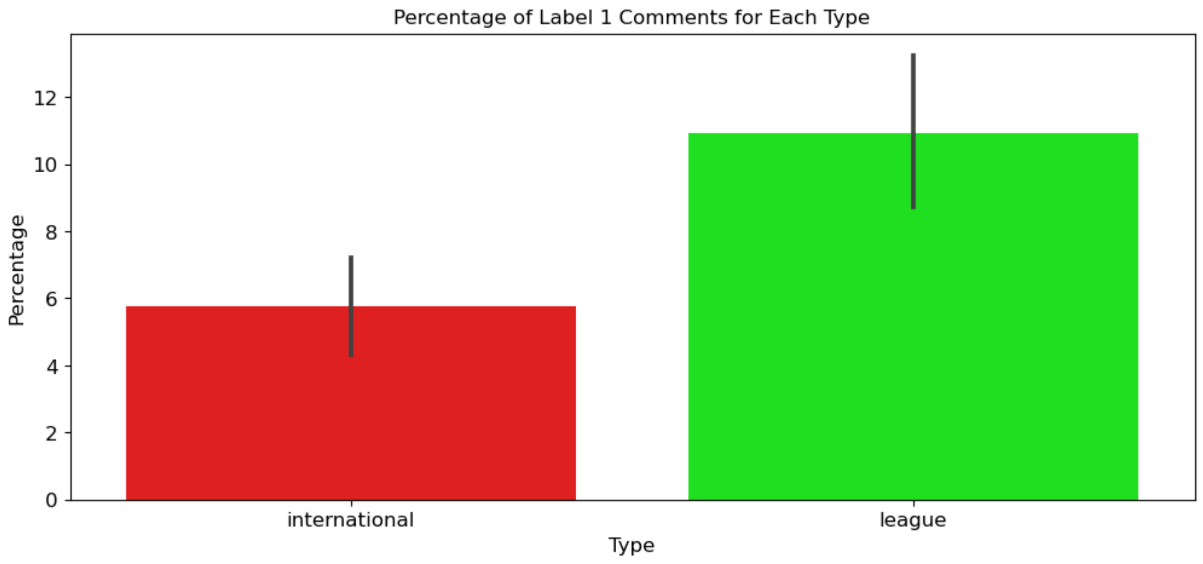


Figure 6: Bar Chart Distribution of offensive comments given each Type

To sum up, the analysis we conducted indicates that YouTube comments for league games typically contain more offensive language than comments involving international games.

5.3 Comparison

From the results of both models, we understand that league matches attract a higher number of offensive comments compared to international matches. The difference between the predictions made by our BERT model and the OLR model raises questions about the factors influencing the prevalence of offensive comments in league and international football matches. However, the magnitude of the difference varies between the two models, highlighting the complexities involved in offensive language detection and the need for further investigation.

The analysis of the newly collected data offers additional evidence that our work is in line with the project's business objectives. We make knowledge accessible that can help to create a more welcoming and constructive online football community by providing insightful knowledge about international and league games. By accurately detecting offensive language, our system could make online forums safer and friendlier for football fans to participate.

6 Project Evaluation

The Evaluation of our NLP Models, developed for football-related comments, is important for determining the achievement of project objectives. In this evaluation, we summarize the project and give an Outlook.

6.1 Summary

The goal of our this project was to create a reliable model for detecting offensive language in football-related media comments. It makes use of sophisticated machine learning methods like BERT and ordinary logistic regression. The principal aim of the project is to improve the quality of virtual exchanges among football enthusiasts by efficiently detecting and addressing instances of offensive language.

One of the main components of our project was data scraping, which involved gathering football-related conversations from different websites like YouTube, Instagram, Reddit, and Kaggle. This all-encompassing strategy guaranteed the collection diverse data for our models' training and assessment.

Extensive preprocessing steps and well-defined labeling guidelines were used in the data preparation process that followed data collection to guarantee compatibility with the selected machine learning algorithms. This also made it easier to combine data from various sources, which allowed for a comprehensive analysis of offensive language patterns on various platforms.

Two main approaches were used in the development of the offensive language detection models: logistic regression and BERT. It was decided that BERT, which is well known for its contextual understanding abilities, would perform better when handling subtle language use, and that logistic regression would offer a more comprehensible method of detecting offensive language, which would be especially useful for this binary classification task.

To ensure the reliability capabilities of our models, evaluation strategies were employed, including cross-check validation techniques and comprehensive performance assessments using evaluation metrics such as accuracy, precision, recall, and F1 score.

Despite the promising results obtained from the initial model training, further refinements were deemed necessary to address specific challenges encountered during the evaluation phase.

These challenges included the models' difficulty in detecting explicit offensive language, particularly in comments authored by the researchers themselves.

Iteratively improving the models in response to these difficulties involved adding more offensive comments to the training dataset and investigating different evaluation strategies in an effort to increase recall and precision rates.

Moreover, the utilization of the models for detecting offensive language in practical situations, like examining YouTube comments to determine the distribution of offensive language, yielded understanding of the frequency and type of offensive language in the football community.

Even though our models were very successful at spotting instances of offensive language, there were still a few areas that needed to be improved. These included improving the models' ability to adapt to various linguistic patterns and cultural contexts, as well as addressing content-dependent offensive language nuances.

In conclusion, our university project is a deliberate attempt to use machine learning methods to improve online conversation among football fans. Our offensive language detection models strive to create a more welcoming and courteous online community for football fans everywhere by means of ongoing improvement and optimization.

6.2 Outlook

While the offensive language detection system demonstrates promising performance, there is room for refinement to further enhance its effectiveness and address any limitations. Future efforts may focus on fine-tuning the Models and Parameters, such as adjusting the Learning Rates, Batch Sizes, and Regularization Techniques to optimize the Model Performance and Generalization Capabilities.

Additionally, augmenting the training dataset with additional annotated comments and expanding its diversity can help improving the model robustness and adaptability to diverse linguistic patterns and contexts. Continuous monitoring and iterative improvement are essential to ensure that our NLP models remains effective and relevant in addressing the evolving challenges of online discourse within the online football community.

6.3 Limitation

In this Section, internal and external validity are discussed. First, there are some limitations about the internal validity. The size of the dataset is nearly 4000 Comments. This would mean that our dataquantity may not be sufficient enough to train an NLP Model. To increase the internal validity of our model and overcome the limitation caused by insufficient data, one would need to increase the size of the dataset even further. This limitation in data size could have implications for the model's training and performance, not only in internal validity but also in terms of external validity.

A limitation to our external validity is also that our model only identifies hate speech written in English. Restricting the identification of hate speech to the English language. This limits the external validity of our natural language model. This restriction falls into account for the presence of hate speech in languages other than English, particularly in regions where football is immensely popular, such as Spain, Brazil, and Italy. Ignoring these linguistic variations undermines the model's effectiveness in comprehensively capturing hate speech across different cultural and linguistic contexts. Because hate speech manifests differently across languages and cultures, a more inclusive approach to language detection is essential to ensuring the model's applicability and effectiveness on a global scale.

Ultimately, the assessment of the NLP Model emphasizes how important it is to foster a welcoming and constructive online community among Football Fans. The Models are essential in promoting civil discourse and improving football fans' online experience by effectively detecting offensive language. Sustained improvement and optimization are necessary to guarantee the Model's continuous efficacy and conformity with the Project's main goals.

References

Oracle (2022): What Is Natural Language Processing (NLP)? Online verfügbar unter https://www.oracle.com/eg/artificial-intelligence/what-is-natural-language-processing/

prakharprasad (2021): Twitter Classification. In: *Kaggle*, 24.08.2021. Online verfügbar unter https://www.kaggle.com/code/prakharprasad/twitter-hate-speech-classification, zuletzt geprüft am 01.03.2024.

Samoshyn, Andrii (2020): and Offensive Language Dataset. Online verfügbar unter https://www.kaggle.com/datasets/mrmorj/hate-speech-and-offensive-language-dataset, zuletzt aktualisiert am 17.06.2020, zuletzt geprüft am 28.02.2024.

Similarweb (2024): Top Websites Ranking - Most Visited Websites in 2024. Online verfügbar unter https://www.similarweb.com/top-websites/, zuletzt aktualisiert am 28.02.2024, zuletzt geprüft am 28.02.2024.

Tan, Vincent (2021) Fine-tuning pretrained NLP models with Huggingface’s Trainer Fine-tuning pretrained NLP models with Huggingface’s Trainer | by Vincent Tan | Towards Data Science

Appendix

GITHUB REPOSITORY:

https://github.com/prizyou/NLP4B_Football

ONEDRIVE LINK:

https://msfau.sharepoint.com/:f:/t/NLPGroup259/Eux9yMk35y5Gow-IJh-FvTEB0uSkrJqXhapGFAgZfjjrEQ

Table of labelling guidelines with examples:

Type of offensive language	Example
Racial Slurs	*"You [racial slur] [nigga] are ruining the game for everyone else. Get out of here!"*
Discriminatory Language	*"Football isn't for [derogatory term] [monkey] like you. Go back to where you came from!"*
Personal Threats	*"I hope someone injures you in the next match. You deserve it."*
Ethnic or National Origin Insults	*"You [derogatory term] [ape] from [specific country] [niggaland] are a disgrace to the sport."*
Sexist Remarks	*"Women have no place in football. Stick to the kitchen."*
Homophobic Language	*"You play like a [derogatory term for LGBTQ+ individuals] [cocksucker]. No wonder your team is losing."*
Ableism	*Example: "Having a player with a disability is dragging the team down. They should be benched."*
Religious Insults	*"Your religious beliefs are backward. Keep them out of football comments."*
Xenophobic Comments	*"Foreign players are taking spots from our own talent. They don't belong here."*
Player-Centric Hate	"You're the worst striker in the league; go retire!"
Fan Community Hate	"Our rival fans are a bunch of [derogatory term] [assholes]."
Incitement to Violence	"Someone needs to break your legs on the field. It's the only way you'll learn."
Explicit Offensive Language	"Your team is [explicit profanity][bunch of assholes] and so are you. No one wants you here."
Intentional Harm	"I hope that your career crashes and burns. You don't deserve success."

Declaration of Academic Integrity

Ich versichere, dass die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen hat und von dieser als Teil einer Prüfungsleistung angenommen wurde. Alle Ausführungen, die wörtlich oder sinngemäß übernommen wurden, sind als solche gekennzeichnet.

[illegible]

Jan Baumgart Nuremberg, 2024-03-01

[illegible]

Paul Roth

[illegible]

Ali Kirac

[illegible]

Akshun Dogra

[illegible]

Burak Cinar