Sub-event Detection in Twitter Streams

Report on the Sub-event Detection Data Challenge
for CSC_51054 at École Polytechnique

Aziz Bacha and Wajdi Maatouk

December 2024

1 Introduction

This report documents our participation in the Kaggle challenge Sub-event Detection in Twitter Streams, which was part of the course **CSC_51054** Machine and Deep Learning at **École Polytechnique**. The challenge involved detecting sub-events during football matches from Twitter streams, with sub-events categorized as 'full time', 'goal', 'half time', 'kick off', 'other', 'owngoal', 'penalty', 'red card', and 'yellow card'. The challenge focused on a binary classification task: determining whether an event occurred in a given period.

It involved 258 entrants, 247 individual participants, and 115 teams, resulting in 2,469 total submissions. Our goal was to compete for the highest accuracy by developing effective models and leveraging advanced machine learning techniques.

Detecting events in social media streams like Twitter is a complex yet critical task with numerous real-world applications, including crisis management, market analysis, and public sentiment tracking. These platforms produce vast streams of noisy, unstructured, and real-time information, presenting unique challenges for machine learning models.

To tackle this problem, we adopted a two-stage approach. First, we utilized Twitter GloVe embeddings, feature engineering and worked extensively on model selection and hyperparameter tuning to achieve the highest accuracy possible. Subsequently, we focused on fine-tuning a large language model (LLM) for the task of event detection on a given text corpus. Then, we trained a meta-model using an LSTM architecture to leverage the LLM's outputs and produce predictions that took advantage of both the content of tweets during a period as well as the sequential nature of the data.

2 Preliminary Analysis of the Dataset

The dataset provided for this challenge contains detailed information about tweets and their associated metadata for a series of football matches. Each entry in the dataset includes the following columns: `ID`, `MatchID`, `PeriodID`, `EventType`, `Timestamp`, and `Tweet`. The `MatchID` identifies the match, while `PeriodID` represents specific periods within each match. `EventType` indicates whether a sub-event has occurred, and `Tweet` contains the text of the corresponding tweet. An example of the dataset's structure is shown below:

ID	MatchID	PeriodID	EventType	Timestamp	Tweet
11_0	11	0	0	1404575400000	RT @2014WorldCup: Argentina vs ...
11_0	11	0	0	1404575400000	@elijahman_ time to focus on ...
11_0	11	0	0	1404575400000	RT @FIFAWorldCup: GLOBAL ...
11_0	11	0	0	1404575400000	RT @CatholicNewsSvc: #PopeFrancis...
11_0	11	0	0	1404575400000	RT @soccerdotcom: If he scores ...

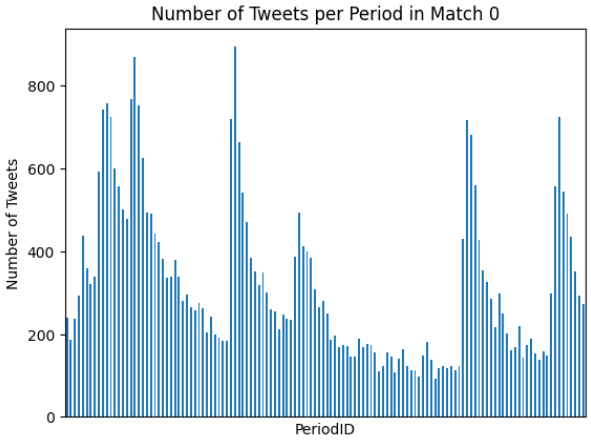
The training dataset consists of tweets from 16 matches, while the evaluation dataset includes tweets from 4 matches. Each match is divided into an average of 133 periods, with approximately 72 periods per match labeled as having a sub-event. On average, each period contains 2476 tweets, although the number of tweets varies significantly between periods and matches. For example, Match 0 contains 41,539 tweets, while Match 1 contains 973,985 tweets. Similarly, within a match, some periods contain significantly more tweets than others. This variability in tweet counts poses a challenge for modeling and requires careful consideration during feature engineering and preprocessing.

To illustrate the variability in tweet distribution, we include two graphs: one showing the number of tweets per period for Match 0 (Figure 1a) and another showing the distribution of tweets per match for all matches in the training dataset (Figure 1b).

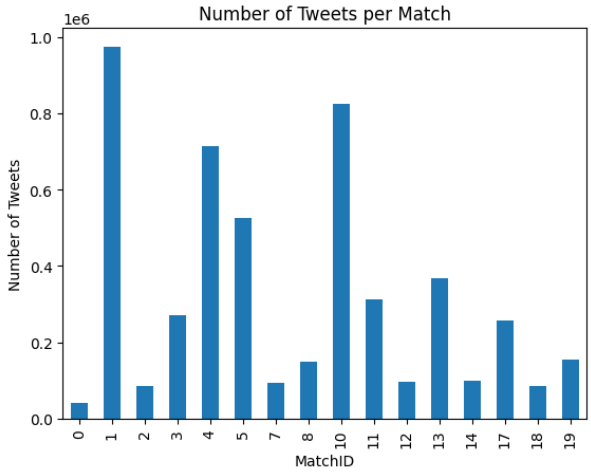
3 First approach using GloVe embeddings

3.1 Data Preprocessing

Preprocessing decisions were driven by an analysis of the dataset and the guidelines provided in [2]. We removed retweets and duplicate posts, as they replicate content without adding new information. This step reduced the dataset size by 54%, which improved computational efficiency and reduced noise. We also experimented with



(a) Number of tweets per period for Match 0.



(b) Number of tweets per match in the training dataset.

Figure 1: Tweet distribution across periods and matches.

including and excluding tweets containing URLs, as we hypothesized that the primary information in such tweets is often in the linked content rather than the tweet itself.

Further preprocessing involved removing mentions, punctuation, and special characters, followed by converting the text to lowercase to ensure uniformity. Tokenization was applied to split the text into individual tokens, which were then refined by removing stop-words. This was crucial for reducing the dimensionality of the data and focusing on words that contribute meaningfully to the classification task.

For normalization, we applied lemmatization using `WordNetLemmatizer`, which reduces words to their base forms while preserving their context. Although we experimented with stemming, lemmatization provided better results by maintaining the semantic integrity of the text.

3.2 Feature Engineering

Feature engineering aimed to provide meaningful inputs to help the model capture patterns in the data. Features were designed based on intuition and exploratory analysis, focusing on linguistic, temporal, and contextual aspects of the tweets.

We included `PeriodID` as a feature after observing its relevance in Figure 2, where some periods consistently showed the same label, particularly during events like kick-off, half-time, and full-time. This suggested that `PeriodID` captures important temporal patterns.

To provide semantic context, we created a dictionary of keywords relevant to specific sub-events and calculated the number of tweets containing these keywords for each period.

We also considered the number of tweets per period as a feature, assuming periods with sub-events, such as goals, would have higher activity. However, a boxplot (Figure 3) showed only minor differences between event types, and feature importance analysis indicated minimal contribution to predictions. As a result, we excluded this feature.

3.3 Model Choice and Comparison

We began by using TF-IDF to construct a vocabulary over the training dataset and then calculated scores for the concatenation of tweets within each period. Limiting ourselves to the 100 most common tokens, we used these scores to construct vector representations of each period. While this approach provided a basic representation of the textual content, the vectors contained limited semantic information, which led to mixed results when tested with various models.

To overcome this limitation, we utilized GloVe Twitter-200 embeddings, which provided dense, pre-trained word vectors capable of capturing richer semantic context. These embeddings were combined with the additional features described earlier, offering a more comprehensive representation of each period.

We experimented with a variety of models, including XGBoost, Random Forest, Multilayer Perceptron (MLP). Hyperparameter tuning was conducted using grid search with cross-validation.

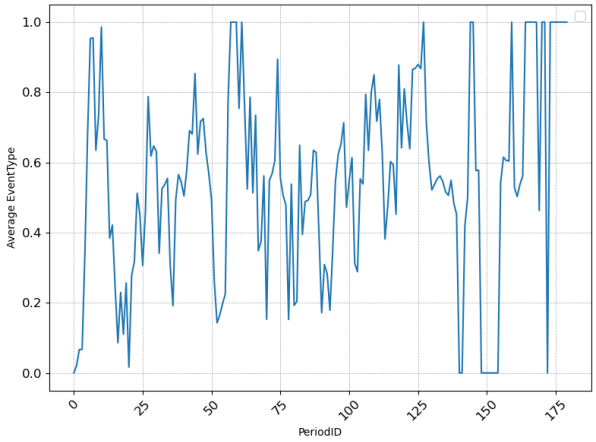


Figure 2: Average `EventType` per `PeriodID`.

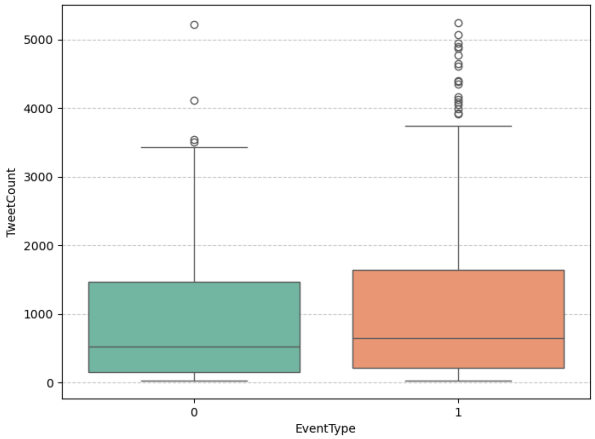


Figure 3: Boxplot of `TweetCount` by `EventType`.

The results of our experiments are summarized in the following table:

Classifier	Accuracy	Precision	F1-Score	ROC-AUC
XGBoost	0.777259	0.777083	0.776941	0.869474
Random Forest	0.788162	0.789415	0.787038	0.875097
MLP	0.758567	0.758538	0.757882	0.840760

Table 1: Performance metrics for various classifiers.

While the test set accuracy was promising, the Kaggle submission results did not meet our expectations. This discrepancy is likely due to overfitting on the training set. To address this, we experimented with techniques such as L1 and L2 regularization and dropout layers in the MLP. However, these adjustments did not result in significant improvements.

As a result, we decided to explore an alternative approach, which is detailed in the next section. This new method ultimately yielded the best results.

4 Using Large Language Models

4.1 Model Selection

After careful consideration, we settled on using Twitter RoBERTa-base for Sentiment Analysis as our base model. This model is pretrained on a large corpus of tweets and fine-tuned for sentiment analysis tasks. Its specialized training on Twitter data makes it well-suited for our problem, as it effectively captures the nuances and informal language typical of tweets.

4.2 Preprocessing

To retain the rich information in tweets while ensuring compatibility with the model's pretraining, we applied minimal preprocessing. Links and usernames were replaced with placeholders `http` and `@user`, respectively, in line with the preprocessing used during the model's pretraining phase. We also experimented with excluding retweets and tweets containing links, as the latter are often spam or advertisements. However, this approach led to a significant drop in accuracy, indicating that the loss of valuable information outweighed the benefits of reducing noise. We did however remove duplicate tweets, as those were very significant in number (around 1680118 tweets across the training dataset) and did not provide any additional information.

4.2.1 Fine Tuning

We followed a specific methodology for fine-tuning:

- Since the model has a token limit of 512 tokens, processing all tweets in a period simultaneously is infeasible. To address this, we created a fixed number n of tweet aggregates per period by concatenating random tweets until the token limit was reached. Each aggregate was assigned the `EventType` label of its associated period.
- We split the data into training and testing sets and used the Hugging Face Trainer with hyperparameters recommended by the model creators for fine-tuning.
- Notably, increasing n improves accuracy but significantly increases training time.

4.3 Calculating Predictions on the Evaluation Set

For the evaluation set, we created aggregates of tweets per period using the same n as during training. Additionally, we experimented with calculating predictions per individual tweet. And while the LLM appeared to be capable of identifying with significant accuracy relevant tweets that might indicate the presence of a sub-event, this approach, despite its higher granularity, did not yield improved accuracy. We attribute this to increased susceptibility to noise due to the random and irregular nature of tweet streams.

4.4 Calculating Predictions Per Period

We explored various methodologies for leveraging the LLM's results to calculate predictions per period:

- **Thresholding:** We defined a threshold for the proportion of aggregates labeled as 1 relative to those labeled as 0. Periods exceeding this threshold were labeled as 1. We optimized the threshold for accuracy on the training set and experimented with taking the mean threshold across periods with a true label of 1. This approach yielded mixed results and did not outperform the accuracy obtained using GloVe embeddings and a logistic regression model with regularization.
- **Meta-modeling:** We introduced a second model that used the LLM's predictions along with additional features to calculate predictions per period. The highest accuracy was achieved using the number of aggregates labeled as 1 and 0 by the LLM and the Period ID as features.

We experimented with various meta-model architectures, including logistic regression, XGBoost, and neural networks. Sequential neural networks, specifically LSTMs, yielded the best results due to the sequential nature of our data. The idea of leveraging LSTMs was inspired by [1], which applied LSTMs in a similar context and reported outstanding results. Hyperparameter tuning using grid search revealed that a window size of 1 was optimal. To avoid overfitting, and considering the limit number of features, we used a relatively simple model architecture, using only 16 hidden layers.

Annex: LLM Predictions on Tweets with True Label 0

Below are examples of tweets from periods with a true label of 0 where the model predicted tweets as 1. Each entry includes the tweet's ID and its text.

```
ID        Text
-------------------------------------------
19_12     rt  kick off
5_78      rt  half time germany 10 france
3_90      ghana scores again usa is out
3_116     rt  ronaldo just scored in portugalghana to give them a 21 leadwhich means us will advance even
7_117     sick goal rt  air  makes it 10
1_19      exciting end to end stuff
10_34     brazil were just that bad
5_123     germany made me proud
2_120     rt  the end now lets say goodbye to the team that brought football to another level  gracias
-------------------------------------------
```

Given the content of these tweets, it is unsurprising that the model may misclassify them. This underscores the challenge of the task, particularly given the highly noisy and unpredictable nature of Twitter streams.

References

[1] Giannis Bekoulis et al. "Sub-event detection from Twitter streams as a sequence labeling problem". In: *arXiv preprint arXiv:1903.05396* (2019). NAACL 2019. URL: https://doi.org/10.48550/arXiv.1903.05396.

[2] Polykarpos Meladianos et al. "An Optimization Approach for Sub-event Detection and Summarization in Twitter". In: *Advances in Information Retrieval*. Vol. 10772. Lecture Notes in Computer Science. Springer, 2018, pp. 481–493. DOI: 10.1007/978-3-319-76941-7_36.