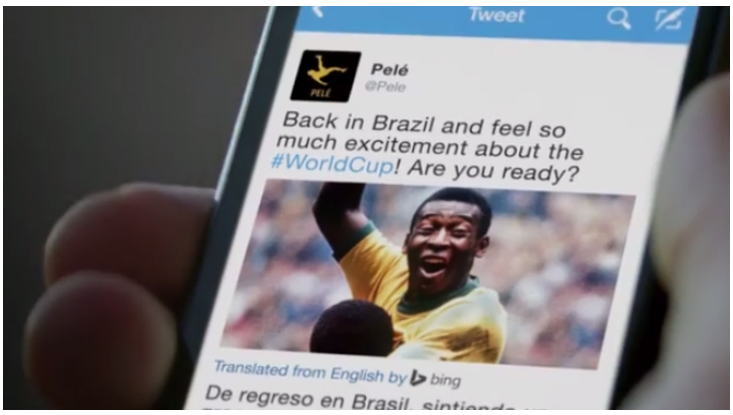
CSC Challenge : Sub-event Detection in Twitter Streams

Mohamed ALOULOU
mohamed.aloulou@polytechnique.edu

Fadi JEMMALI
fadi.jemmali@polytechnique.edu

Elie NICOLAS
elie.nicolas@polytechnique.edu

CHICHA LEARNING TEAM



Abstract

This report tackles the challenge of accurately detecting sub-events in Twitter streams during the FIFA World Cups. Tweets were organized into minute-long intervals, framing the task as a binary classification problem to identify, using numerous tweets per period, if an event occurred during the period. Preprocessing mainly included noise removal, lemmatization, and tokenization to enhance textual clarity. Various models were evaluated, including Mamba, LSTM, and transformer-based architectures. Our selected model involved concatenating tweets into quotas, enabling multiple samples to finetune `twitter-roberta`*, a pretrained model tailored for tweets. To enrich the input data and improve accuracy, feature engineering incorporated diverse textual and stylistic features to capture nuanced patterns in tweet content. Building on this enriched feature set, a meta-classifier based on decision trees was introduced. This meta-model leveraged contextual features and probabilities derived from BERT-based partitioned outputs, with each period represented by aggregated probabilities from multiple tweet quotas. This approach achieved notable accuracy gains and enhanced sub-event detection capabilities.*

1. Data preprocessing and feature extraction

1.1. Data preprocessing

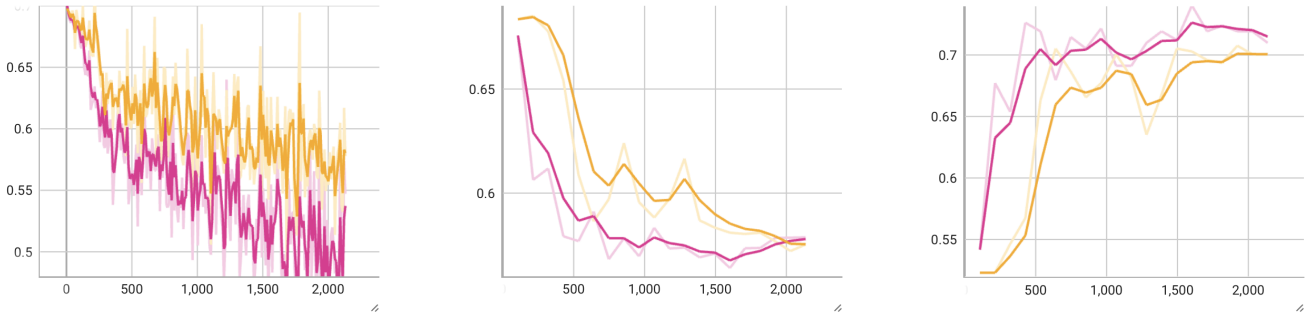
To ensure a clean foundation for the project, the raw tweet dataset underwent a systematic preprocessing pipeline. It was designed to standardize and clean the textual data through text transformation and noise removal. Firstly, we converted all text to lowercase, ensuring uniform text representation across the dataset. Secondly, we removed what we thought was noise, such as: URLs, mentions, hashtags, retweet indicator (*e.g.* "*rt*"), multiple whitespace characters, and english stop words.

First, we noticed that the dataset contains no `NaN` or missing values. If present, they would be challenging to impute meaningfully, so rows with `NaN`s would have been removed, provided their proportion was reasonable. Nevertheless, The dataset unfortunately contains a disproportionate amount of duplicates (around 30%). Had we realized it soon enough, we would have trained our model on the dataset of unique values. It should be noted that applying the same techniques to the cleaned dataset should improve results.

In the following sections we often train our models using Natural Language Processing (NLP) which requires a tokenization of texts before the embeddings. A basic approach is to take each word as a token, which is what we are applying when we are not using a pre-trained model. Once tokenized, we removed the english stopwords(*"the", "a", "and"* etc...) from the tweets. This removes a component of the noise, while also reducing total textual volume, which improves efficiency both in accuracy and runtime. Another important part of the process is the choice to lemmatize the words of the tweets. Lemmatization is a linguistic normalization technique that reduces words to their base form (lemma). Our data preprocessing resembles that of Meladianos *et al.* [4], however we introduced lemmatizing instead of stemming. Unlike basic stemming (word chopping), lemmatization looks at the whole word and its grammatical context to find its meaningful root, so you get more accurate and sensible base words. In the context of World Cup tweet analysis, lemmatization is particularly beneficial as it helps standardize varied word forms (such as *"playing", "played", "plays"* which all yield *"play"*) while maintaining the core meaning. This is a crucial step for capturing consistent sentiment and semantic content across very diverse tweets. Fig. 5 in Appendix describes the training loss, evaluation loss and evaluation accuracy of a BiLSTM_32 with and without both removing stopwords and lemmatization. Those additional processes generated smoother and more regular plots, while also yielding higher accuracy. These empirical findings are in accordance with our theretical understanding of these methods.

1.2. Feature extraction

Our feature extraction process began by a comprehensive exploration of features that were thought to be related to the dataset. We computed the average number of repeated letters, which seemed common for tweets related to goals (e.g. *"goaaallll"* in a tweet). We analyzed the average number of a symbol occurring, for example "!". Then we tried to create specific lists that are related to specific event types. For example, we generated a list of words that are football related. It turns out that for certain words in it, the more this word occurs, the higher is the probability of a type 1 event. Conversely, we accumulated a list of no-event terms, which indicates that there are no events. This list consists of words that describe a seemingly boring football where no action happens. It also contains the main words that are linked to advertisement and more generally spam tweets. On top of these we explored other features that seemed plausible, such as : tweet length, hashtag count, and top words by event type. One last important feature is sentiment analysis. Intuitively, if there is an important event happening, people will tweet with higher intensity. Conversely, if not much is going on during the game, then people will write more neutral tweets. To effectively operate the sentiment analysis, we use VADER, a sentiment analysis Rule-based model specifically trained for social media [3]. The first step was to see if the feature was significantly different for events of type 1 than for events of type 0 (Fig. 4).



(a) Training Loss (b) Evaluation Loss (c) Evaluation Accuracy

Figure 2. Comparison between training a BiLSTM_32 with GloVe-Twitter (in pink) versus with `twitter-roberta`. (in orange).

2. Model comparison

2.1. LSTM with attention score

One key limitation we identified in the baseline method was the approximation introduced by averaging all word embeddings to generate tweet embeddings, followed by averaging these tweet embeddings to represent an ID. To address this, we retained the *GloVe-Twitter* embeddings and maintained quite similar preprocessing steps but proposed a more sophisticated approach for ID representation.

We opted for deep learning models due to their capacity to capture complex interactions in data. Instead of averaging all predictions, we implemented a weighted averaging method based on attention scores and generated by a Single-Layer Perceptron (SLP), which reduces the 200-dimensional word embeddings to scalars normalized by the Softmax function. This enabled the model to learn the relative importance of words during embedding.

After obtaining learned tweet embeddings through the weighted mechanism, we utilized a sequential model. First, we experimented with Mamba [2], a promising model based on State Space Models (SSMs) and known for its selective nature. However, as it is hard to train, and due to overfitting issues (Fig. 6), we opted to use another model.

Subsequently, we implemented LSTMs that take sequences of tweets per ID as input, padded to ensure uniform lengths. This approach is similar to how a human might scroll through tweets sequentially to determine whether an event occurred. The LSTM model included a classification head with a single linear layer derived from the final hidden state. The hidden state size significantly impacted performance (Fig. 7).

We experimented with both LSTM and BiLSTM architectures and observed that BiLSTM performed better for our task (Fig. 7). The optimal model was a BiLSTM with a hidden state size of 32. To mitigate overfitting, we applied weight decay, penalizing large weights by adding a term proportional to their squared magnitude to the loss function. Additionally, we fine-tuned learning rates to balance convergence speed and model stability.

Evaluation loss was computed using the Binary Cross-Entropy (BCE) criterion:

$$\mathcal{L}_{BCE} = -\frac{1}{N}\sum_{i=1}^{N}\left[y_i \log(\hat{y}_i) + (1-y_i)\log(1-\hat{y}_i)\right] \quad (1)$$

BCE was chosen because it is effective for binary classification tasks with probabilistic outputs.

2.2. Bert

Another potential limitation we identified was the use of GloVe embeddings. To address this, we explored an alternative embedding model, `twitter-roberta-base`, which increased the embedding dimension from 200 to 768. We hypothesized that this change would enhance the performance of the LSTM model (Sec. 2.1); however, as shown in Fig. 2, the results did not meet our expectations. To better capture the relationships between words, we replaced the LSTM with weighted averaging mechanism and shifted our focus to a transformer-based approach with an attention mechanism. We employed a pretrained BERT model fine-tuned for tweets, `twitter-roberta-base` [1], using Hugging Face's `AutoModelForSequenceClassification` for adaptability and efficiency.

Initially, we trained a truncated model by concatenating tweets for each ID and using only the first 512 tokens. While computationally efficient, this approach resulted in a loss of information and achieved a validation accuracy of 0.69921. To address this, we explored `longformer-base`, which supports sequences up to 4096 tokens, resulting in a modest improvement, with validation accuracy increasing to *0.70703*.

The most effective strategy was to partition tweets into segments of up to 512 tokens, grouping approximately 50 tweets per segment based on the average tweet

length of 10 tokens. IDs were padded to 2000 tweets for consistency, and each segment was processed with `twitter-roberta`. This method retained more context and significantly improved performance. For aggregation, averaging probabilities outperformed a voting system, with PartitionBERT512 achieving val accuracies of *0.71875* using voting and *0.73437* with averaged probabilities.

Additionally, using evaluation loss (BCE), defined in Eq. (1), as the criterion for selecting the best model proved more effective than evaluation accuracy on the validation dataset. BCE is more reliable for probabilistic outputs, as it penalizes overconfident predictions and better aligns with the ultimate optimization objective of the model.

The architecture of the final model is illustrated in Fig. 3, and the training process is detailed in Fig. 8.

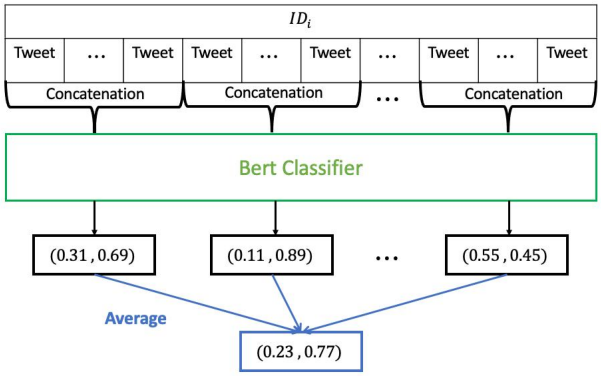


Figure 3. Partition BERT Classifier (by averaging).

2.3. Meta-Classifier

While the partitioned BERT model (Sec. 2.2) delivered promising results, its effectiveness could be further enhanced by integrating the features outlined in Section 1.2. Unfortunately, incorporating these features into the Hugging Face classifier would have necessitated recoding it, which would result in losing the efficiencies provided.

Indeed, feature generation process aimed to extract meaningful attributes from the raw tweet data aggregated at the `PeriodID` level rather than the tweet level. Key features included `PeriodID` and `PeriodID Ratio`, calculated as the ratio of the period to the total periods within a match. Features such as `TotalFootballTerms` (count of football-related keywords), `FootballTerms mean` and `NumTweets` (volume of tweets in a period) captured textual and contextual aspects of the data. Stylistic features, including `AvgPunctuation`, `AvgUppercase`, `!Ratio mean`, and `RepeatedLettersRatio mean`, provided insights into tweet emphasis and emotional expression.

To enhance the feature set, sentiment analysis was applied using the VADER model [3], generating attributes such as the mean and standard deviation of *positive*, *negative*, *neutral*, and *compound* sentiment scores.

The most important point is that in PartitionBERT, each period is characterized by different subgroups of tweets, each associated with classification probabilities. So, instead of averaging by ID, we aggregated all available data across periods and examined it holistically to better capture the nuances of the distributions. Consequently, features derived from BERT-generated probabilities provided a statistical perspective on sub-event likelihoods. For each period, we computed `Avg Prob`, `Var Prob` and `Weighted Avg Prob` (weighted by the squared probabilities). Furthermore, `Proportion High Probs` and `Proportion Low Probs` measured the prevalence of tweets with probabilities exceeding 0.9 and below 0.1, respectively.

By combining all the extracted features, we trained classifier models to predict the presence or absence of sub-events for each `PeriodID`. The first model utilized a `Random Forest Classifier` with a focus on performance optimization through hyper-parameter tuning. `RandomizedSearchCV` was employed to evaluate a predefined number of random combinations from a parameter grid to identify the best-performing hyperparameters. The parameters tuned included *n_estimators*, *max_depth*, *min_samples_split*, *min_samples_leaf*, *max_features*, and *bootstrap*. The model achieved a score of *0.74609* on the Kaggle leaderboard. The second model also employed a *Random Forest Classifier* but introduced enhancements to address class imbalance and improve generalization. *SMOTE* (Synthetic Minority Oversampling Technique) was applied to generate synthetic samples for the minority class, creating a balanced training dataset. Hyper-parameter tuning was again performed using *RandomizedSearchCV*, this time with a *Stratified K-Fold Cross-Validation* approach. This method preserved the class distribution across five splits, ensuring robust hyper-parameter evaluation. Additionally, feature importance was analyzed using the trained Random Forest model, with a bar chart illustrating the relative importance of features included in the appendix (Fig. 9). This model achieved a score of *0.74218*.

Despite the seemingly small difference in accuracy scores between the two models, a detailed comparison of their submission files revealed 58 mismatched rows. By setting all these mismatched rows to `EventType = 1` (the most frequent), the combined model achieved an improved accuracy score of *0.76171* on the Kaggle leaderboard. This result highlights the complementary strengths of the two models and demonstrates the potential benefits of strategically combining their outputs.

References

[1] Francesco Barbieri, Jose Camacho-Collados, Leonardo Neves, and Luis Espinosa-Anke. Tweeteval: Unified benchmark and comparative evaluation for tweet classification, 2020. 3

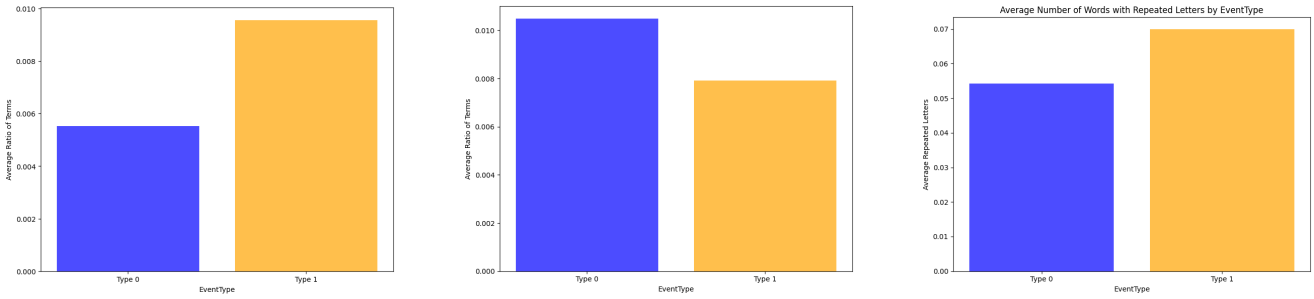
[2] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces, 2024. 3

[3] C.J. Hutto and Eric Gilbert. Vader: A parsimonious rule-based model for sentiment analysis of social media text. 01 2015. 2, 4

[4] Polykarpos Meladianos, Christos Xypolopoulos, Giannis Nikolentzos, and Michalis Vazirgiannis. *An Optimization Approach for Sub-event Detection and Summarization in Twitter*, pages 481–493. 03 2018. 2

Appendix

Some features motivations



(a) Average Ratio of football terms (b) Average Ratio of ”no event” terms (c) Average number of repeated letters

Figure 4. Features comparison between tweets when an event occurred (orange) versus tweets when it did not (in blue).

.

Football-related terms are crucial for analyzing tweets focused on events and gameplay dynamics. These terms include: ”goal,” ”penalty,” ”offside,” ”yellow,” ”red,” and so on. However, through analysis, we observe that spam tweets tend to be more strongly associated with ”no event” terms rather than football terms. To better capture the characteristics of spam content, we construct a set of ”no event” terms that reflect the lack of specific football-related information and often align with promotional or off-topic content. These terms include: ”fan,” ”boring,” ”possession,” ”streaming,” ”slow,” and so on.

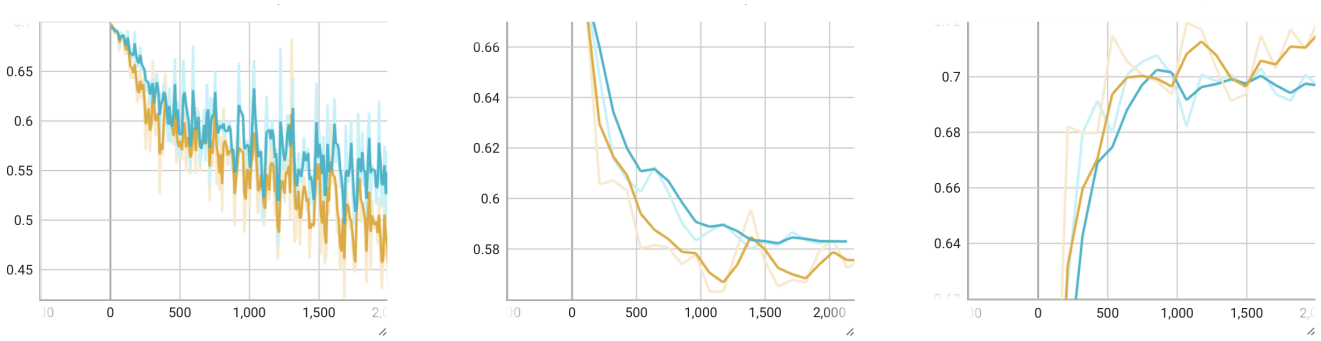
Training and Evaluation Data Splits

During training, we used a `train_test_split` with 20% of the training dataset allocated as the evaluation set. This evaluation set was distinct from the validation set provided by Kaggle, which we consistently referred to as the `validation_set`.

The `eval_set` allowed us to monitor performance during training and tune hyperparameters, while the `validation_set` served as an independent benchmark for model generalization.

To train our deep learning models, we utilized the GPU resources provided by l’X, accessed via SSH connections. For reference, training the PartitionBERT512 model took approximately 7 hours using an NVIDIA RTX A4000 GPU.

Lemmatizer and stop words



(a) Training Loss (b) Evaluation Loss (c) Evaluation Accuracy

Figure 5. Comparison between BiLSTM_32 with lemmatizer and removing stop words (in orange) versus without (in blue).

Training Mamba

(a) Training Loss (b) Evaluation Loss

Figure 6. Comparison of Training Performance of 3 different Mamba models with different depths.

Training LSTM

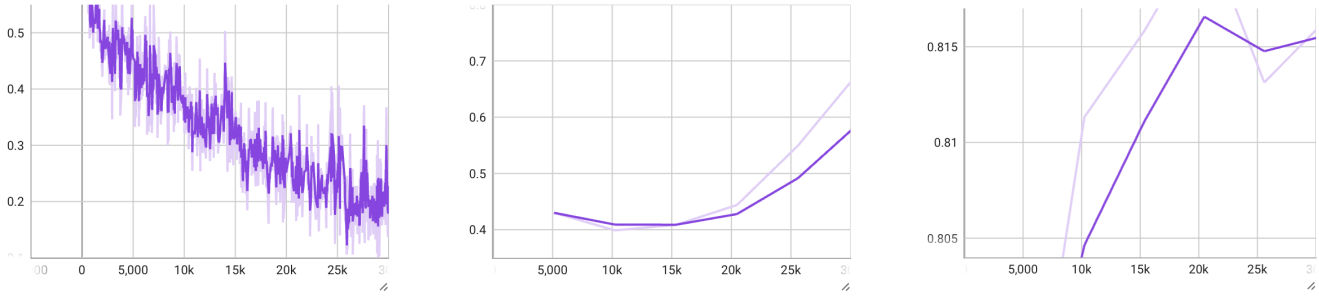
(a) Evaluation Loss (b) Evaluation Accuracy

Figure 7. **Comparison of Training Performance Between LSTM and BiLSTM Models:** These graphs compares the training performance of models with different configurations, including **LSTM with hidden state size 64**, **BiLSTM with hidden state size 32**, **BiLSTM with hidden state size 64**, and **LSTM with hidden state size 32**

Model	Accuracy on Validation Set
FrequencyClassifier	0.62890
LogisticRegression	0.65234
biLSTM32	0.70312
Finetuned Tweet RoBERTa 512 (with truncation)	0.69921
Finetuned Long-base Transformer 4092 (with truncation)	0.70703
PartitionBERT512 with vote	0.71875
PartitionBERT512 with avg_proba	0.73437
Metaclassifier Random Forest with PartitionBERT512	0.74609
CombinedMetaModels_sub_1	0.76171

Table 1. Model Accuracy on Validation Set

Bert with Partition Training



(a) Training Loss (b) Evaluation Loss (c) Evaluation Accuracy

Figure 8. Partition Bert based on pretrained `twitter-roberta-base` with groups of 50 tweets.

It is worth noting that the evaluation accuracy appeared higher because it was computed on 20% of the partitioned training data (into tweet quotas of 50 concatenated, with each ID represented across multiple quotas). This partitioning also increased the number of training steps per epoch due to the larger dataset size compared to previous configurations.

Feature importance

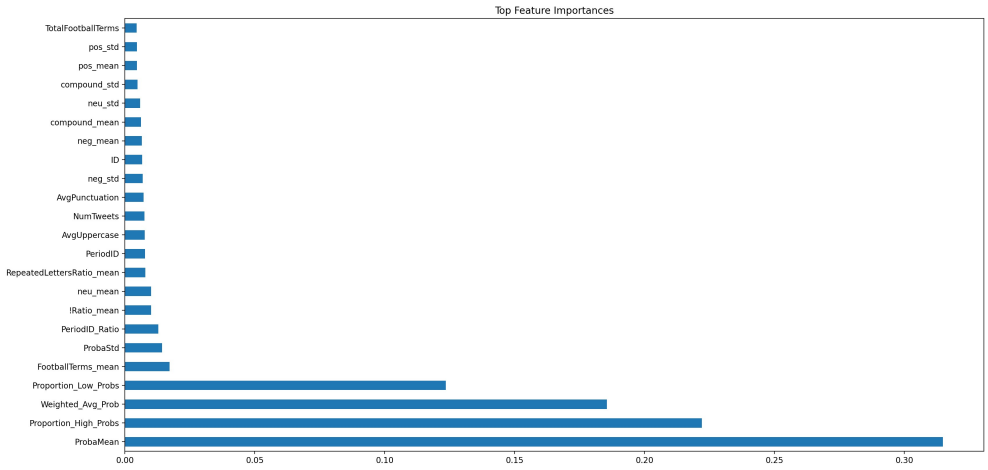


Figure 9. Feature importance in the Random Forest model.