CSC_51054_EP - Report on Data Challenge

Sub-event Detection in Twitter Streams

Kaggle Name Team : `Tristan Enzo Hari`

Tristan Donzé
Institut Polytechnique de Paris
`tristan.donze@outlook.com`

Enzo Pinchon
Institut Polytechnique de Paris
`enzopinchon9@gmail.com`

Hari Praanesh
Institut Polytechnique de Paris
`mc.haripraanesh@studentmails.com`

Abstract

This document presents the work carried out as part of the final course project Machine & Deep Learning Introduction. The main objective of this project was the classification of tweets from matches during the 2014 World Cup. The document details the steps undertaken to successfully complete this project: data preprocessing, visualization, feature extraction and selection, model selection, and the comparison of model performances.

Background

The provided dataset consists of multiple CSV files, each corresponding to a specific match from the 2014 World Cup and containing tweets published during the match, capturing users' real-time reactions and comments. Each CSV file includes the following columns: `ID, MatchID, PeriodID, EventType, Timestamp, Tweet`. The `ID` column is constructed by concatenating `MatchID` and `PeriodID`, separated by an underscore (`_`). The `EventType` column represents the label for each entry. By definition, this label remains constant for the same period (`PeriodID`) and does not vary between different identifiers (`ID`). Entries are labeled with `EventType = 1` when one of the following events occurs during the match: `full time, goal, half time, kick off, other, owngoal, penalty, red card, yellow card`.

Contents

1 Data Preprocessing and Feature Selection/Extraction

1.1 Preprocessing Data

Our dataset consists of textual tweets, which we cleaned to retain only the most relevant and original content. Following common practices in tweet preprocessing [2], we used regular expressions (regex) to remove retweets (duplicates), tweets with mentions (directed messages that could add noise), and those containing external links (typically ads or content shares, offering little analytical value). This process enabled us to create a dataset that is both qualitative and homogeneous, while preserving variations such as tweet frequency over time. 1

1.2 Feature Selection & Extraction

1.2.1 Basic features

Initially, our analysis focused on data visualization to identify potential patterns or significant trends and thus, create relevant features. Using regular expressions and bar charts, we graphically represented the distribution of "spotted" tweets by PeriodID, differentiating tweets according to their EventType (1 in red, 0 in blue). We define a tweet as "spotted" if it contains any word from a predefined list of terms related to the following: `full time, goal, half time, kick off, other, owngoal, penalty, red card, yellow card`. We applied the same approach to analyze the number of Retweets in the raw dataset, as well as the average tweet length in the cleaned dataset. However, these last two visualizations did not reveal any particularly relevant trends.

1.2.2 NLP features

To effectively represent the content of tweets numerically, we used vector embeddings. This representation requires an initial preprocessing and tokenization step for the texts.

The tokenization process was carried out in several stages. First, we standardized the tweets by converting all text to lowercase and removing punctuation. Next, we applied tokenization to break each tweet down into lexical units (tokens). A lemmatization step then reduced words to their root forms (for example, "running," "ran," and "runs" become "run"). Finally, we filtered out stop words such as "and," "the," and "is," keeping only meaningful terms.

After experimenting with different embedding models, we chose "glove-twitter-200," which generates 200-dimensional vectors. This choice was made because performance remained relatively stable across the various models tested. We also tried the compact BERT model "all-MiniLM-L6-v2," but its results did not surpass those obtained with "glove-twitter-200." Additionally, we created embeddings using TF-IDF to explore alternative representations. These embeddings were constructed using the same tokenization process as described above, ensuring consistency in the preprocessing pipeline.

The embedding generation process proceeded as follows: for each user identified by their ID, we first collected and tokenized all of their tweets. These tokens were then aggregated into a single flattened list per ID. Finally, this list was vectorized to obtain a unique embedding representing all of the user's tweets.

1.2.3 Temporal causality

Events are influenced by temporal factors, for instance, after a goal, there is often a brief reduction in activity before a spike in tweets. Similarly, as the match nears its end, the likelihood of events like goals and fouls increases. Factors like yellow cards can also temporarily alter player behavior, leading to more cautious play.

While tweets per minute are a useful indicator, incorporating temporal patterns enhances event detection. For example, tweets tend to decrease before a major event, as users pause to focus on the game. However, due to lack of precise timestamps, this behavior cannot be detected with complete accuracy.

The match's progression is also crucial; early on, events are less frequent, while towards the end, teams take more risks, increasing event likelihood. Additionally, Twitter activity varies across the match, with fewer reactions early on and more towards the end. This allows us to roughly detect events like halftime breaks.

2 Model Choice, Tuning and Comparison

2.1 Our best model

Our first performant approach was based on a Random Forest model using only the basic features mentioned earlier. Following these results, we decided to explore the use of XGBoost in combination with our precomputed embeddings.

XGBoost, like Random Forest, is an ensemble model based on decision trees. However, it stands out due to its "gradient boosting" approach, which builds trees sequentially, with each new tree correcting the errors of the previous ones.

To identify the optimal hyperparameters for XGBoost, we employed a Grid Search Cross-Validation (GridSearchCV) strategy. This process systematically explored combinations of parameters, such as learning rate, maximum depth, and number of estimators, to achieve the best performance on the validation set.

In an attempt to optimize, we tried incorporating all available features (basic which has been normalized and embeddings) into our XGBoost model. However, this approach proved less effective, likely due to information redundancy and noise introduced by certain variables.

To avoid overfitting, we fixed the maximum depth of the trees to 10, ensuring a balance between model complexity and generalization. Furthermore, XGBoost's default parameters are specifically designed to mitigate overfitting. This include parameters such as subsample, lambda and alpha for respectively L2 and L1 regularization.

2.2 Temporal based model

In football match event detection, tweets form a temporal sequence reflecting real-time reactions. Events like goals or fouls generate bursts of activity and evolving tweet patterns. A temporal model is ideal for capturing these dynamics, distinguishing meaningful patterns from background noise.

By leveraging the temporal structure, the model identifies correlations between tweets over time, detects event buildups, and considers the temporal proximity of reactions. This enables event detection based on the broader context of tweet sequences.

We explored both LSTMs and TCNs, as detailed in the below section, but focus on TCNs here, as they outperformed LSTMs in our experiments and in other time-series applications [1]. TCNs use dilated convolutions to capture long-range dependencies and avoid vanishing gradients, are computationally efficient, and preserve sequence order integrity, making them ideal for large datasets like football match tweets.

Given these advantages, we selected TCNs as our primary model. Data preprocessing involved organizing tweets into time periods, creating contexts similar to NLP techniques like word2vec, but adjusted for the temporal nature of events. Context before and after events is used to predict future events, with time-dependent significance considered. In TCNs, consistent context size is crucial. When context is missing (e.g., before the first row), padding is added. A 'mask' feature indicates whether data is real or padded, helping the model distinguish artificial inputs.

We initially implemented a basic TCN without attention mechanisms or dropout, testing 487 parameter combinations by varying time-step, kernel size, its evolution across layers, layer count, and output channels. From the top 10 configurations, we developed a parallel TCN architecture to improve flexibility, enabling convolutional blocks to focus on different input aspects. Dropout was added to reduce over-fitting, while attention mechanism was use to assign varying importance to each convolutional block. Additionally, we add at the start of each convolutional block another attention mechanism to be able to focus on distinct features, further enhancing the specialization of the convolutions blocks. Finally we add a residual connection that sends the focused row (the one for which we want to predict the event type) alongside the convolution output into the attention system and MLP.

The TCN model performed well overall, achieving 75% accuracy on the full evaluation set. However, we did not test it with the feature from our best approach, which dynamically extracts the top N most important words for classification.

The model's performance could likely improve with additional features capturing temporal and behavioral patterns, such as precise timestamps or user IDs. Residual tweet noise, even after preprocessing, may have reduced the quality of extracted features. Furthermore, the limited dataset of 16 files restricted the model's ability to generalize effectively, highlighting the need for more diverse and extensive training data.

2.3 LSTM Model with K-Fold Cross-Validation

We developed an LSTM (Long Short-Term Memory) neural network. This model can process sequential data, using TF-IDF embeddings passing through a dense layer with ReLU activation. The network includes a bidirectional LSTM layer with 128 units. Dropout mechanisms at 0.3 were applied to limit overfitting. A 5-fold cross-validation strategy was used. This approach divides the dataset into five portions, allowing the model to learn from different data combinations and validate its performance on unseen data. Training used the Adam optimizer and binary cross-entropy loss function, saving the best-performing model configuration at each iteration. After cross-validation, the model was trained on the complete dataset. However, the results on Kaggle were not conclusive. These poor performances are likely due to overfitting, the lack of a real temporal connection between embeddings for each ID. Moreover, the K-fold approach can potentially disrupt the temporal continuity of the data, breaking the inherent sequential relationships.

3 Conclusion

In conclusion, this project provided a comprehensive pipeline, from data collection and preprocessing to model selection and performance comparison (XGBoost, TCN, LSTM). Despite efforts in feature engineering, temporal modeling, and the exploration of advanced techniques (embeddings, TCN with attention), the results remained mixed. Challenges such as noisy data, limited temporal granularity, and the complexity of event prediction constrained the models' performance. These limitations highlight potential areas for improvement, including access to richer data, better temporal contextualization, and the exploration of novel approaches combining semantic and temporal contexts.

A Additional Images

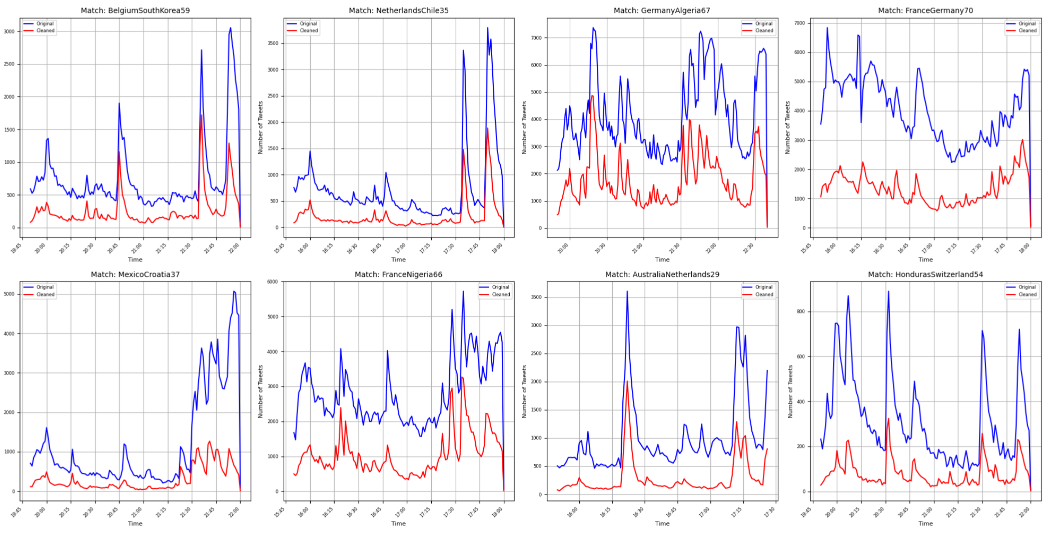

Figure 1: For each match (non-exhaustive), evolution of the number of tweet over time: blue for raw data, red for cleaned data.

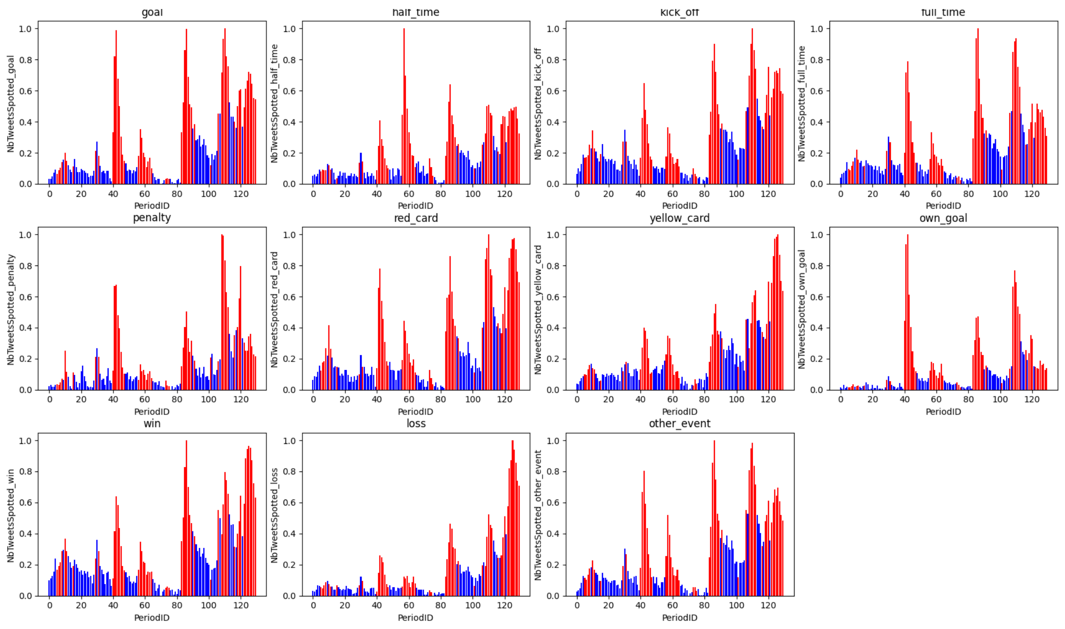

Figure 2: Relevant word frequency per period in the same text corpus.

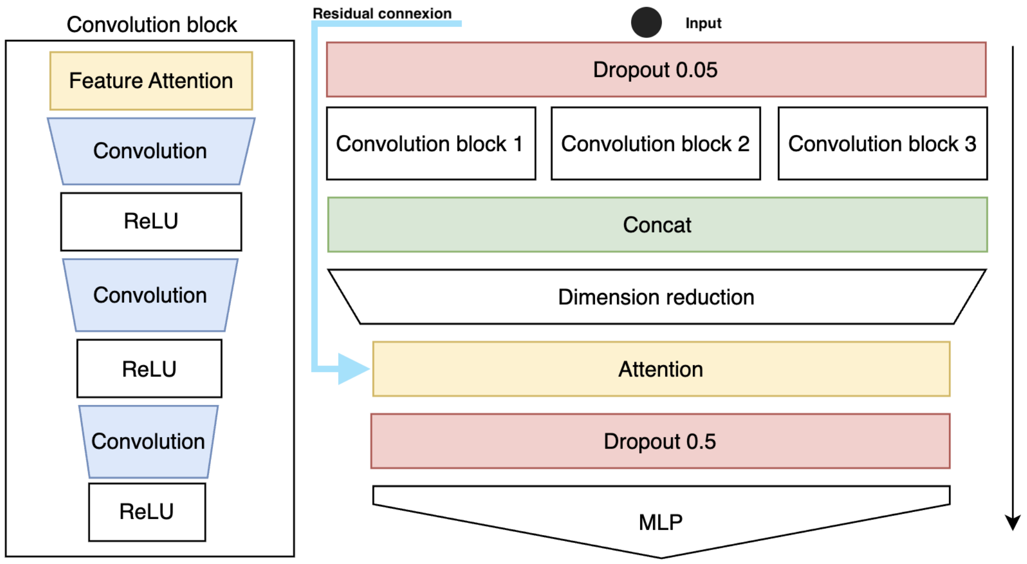

Figure 3: A representation of the TCN we developped.

References

[1] Shaojie Bai, J Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*, 2018.

[2] Polykarpos Meladianos, Christos Xypolopoulos, Giannis Nikolentzos, and Michalis Vazirgiannis. An optimization approach for sub-event detection and summarization in twitter. In *Advances in Information Retrieval: 40th European Conference on IR Research, ECIR 2018, Grenoble, France, March 26-29, 2018, Proceedings 40*, pages 481–493. Springer, 2018.