SubEvent Detection on Twitter Streams

Pedro Silva, Sabrina Lomelino Sartori, Mamadou Lamarana Diallo

{pedro.silva, sabrina.lomelino-sartori,mamadou-lamarana.diallo}
@polytechnique.edu

Kaggle usernames: peulsilva, sabrinasartori10, lamrana

December 12, 2024

1 Related Work

Transformers Transformers [1] are a deep learning architecture designed for processing sequential data and have proven highly effective for tasks in natural language processing (NLP). They rely on the *attention mechanism*, which allows the model to focus on relevant parts of the input sequence by computing relationships between tokens.

Intuitively, attention assigns scores that quantify how much one token influences another within the context. As the model processes multiple layers, the embeddings of each token are linearly combined based on these scores, allowing the model to effectively capture and represent semantic relationships within the text. This mechanism can be formalized in equation 1.

$$\mathbf{E}^{(\ell)} = g\left(\mathbf{E}^{(\ell-1)} + \text{Attention}^{(\ell)}(\mathbf{E}^{(\ell-1)})\right), \tag{1}$$

where g represents normalization and linear layers.

Model Quantization Model quantization reduces model size and speeds up inference by representing weights and activations with lower precision, like INT8 instead of FP32. This introduces quantization errors, where the difference between original and quantized values ($e = w_{\text{original}} - w_{\text{quantized}}$) can propagate through layers and affect predictions. Techniques like quantization-aware training (QAT) simulate quantization during training to minimize these errors, ensuring better accuracy compared to post-training quantization. Quantization is ideal for efficient inference on resource-constrained devices, which was the case on this challenge.

2 Preprocessing

Given the volume of data, the first step involved thorough preprocessing to enhance the dataset's utility for modeling. Figure 1 present the pipeline of preprocessing on the tweet dataset. The key steps were:

1. **Removing Duplicate Tweets and Retweets** Duplicate tweets and Retweets (marked as "`RT`") were removed as they contributed minimal informational value. These redundant entries accounted for slightly more than 50% of the total dataset.

2. **Excluding Tweets with Mentions** Tweets containing mentions (e.g., `@username`) were excluded after verifying that they added negligible value to identifying events.

3. **Prioritizing Informative Short Tweets** Short tweets (fewer than 10

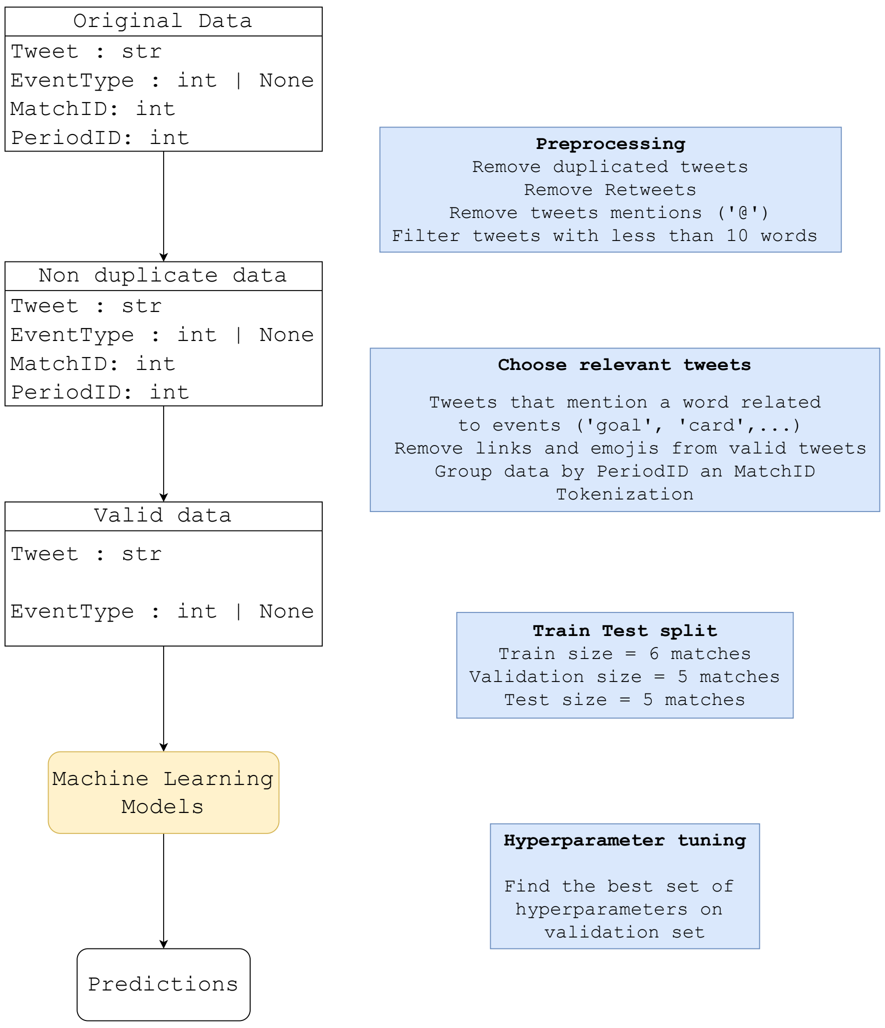

Figure 1. Pipeline schema illustrating the overall solution architecture.

words) often provided clearer signals about events. For example, goal-related tweets typically consisted of concise text such as “`Goooal!!!`” or “`A nice score by [player]`” These were retained for their high relevance.

4. **Removing Emojis and Links** Emojis and links were stripped from the text. Although such elements can occasionally provide context, it was assumed that the pre-trained models utilized were not optimized for processing text containing emojis, potentially reducing their semantic understanding of these tokens.

5. **Concatenating Tweets** Finally, all tweets within a specific time period were concatenated to form a single text corpus. This combined text was used as input for training machine learning models (detailed in Section 3).

To ensure both the fitting and generalization capabilities of our models, we split the dataset of 16 matches into three subsets: 6 matches for training, 5 matches for validation, and 5 matches for testing. The validation set was specifically used to identify the optimal hyperparameters for each task.

3 Approaches

3.1 Bag of Words

We used a Bag-of-Words (BoW) model with TF-IDF (Term Frequency-Inverse Document Frequency) for feature extraction. TF-IDF prioritizes relevant terms while down-weighting common words like articles and prepositions.

The vocabulary included football-specific terms (e.g., `"goal,"` `"penalty"`) and the 30 most frequent tokens from the training data. It is important to note that, since we are counting the occurrence of each token, specifically in this approach, we removed country names by placeholders such as `"County A"`, `"Country B"` as shown below.

Original text: Great match between Brazil and Cameroon

Processed text: Great match between country A and country B

We trained a classifier (Linear Regression, Random Forest or a MLP) to predict the target value. Key hyperparameters were optimized using 10-fold cross-validation for robust performance.

3.2 Encoder-Only Transformers

Encoder-only transformer models, such as BERT [2], are pre-trained on the task of next-token prediction by considering all tokens within a given context length, rather than just those preceding the current token. This bidirectional approach enables these models to excel in various text classification tasks and has established them as state-of-the-art solutions in the field.

We finetuned an encoder only model, minimizing the binary cross entropy between the target and the predictions. As there were many relevant tweets on each period and the original BERT model support only 512 tokens in it’s context, we chose a model that was trained on long context, using sliding window attention, which scales linearly with the context length. This model was the LongFormer [3], which supports up to 4096 token in a single input.

To ensure stability during the fine-tuning process, we adopted a selective freezing strategy. All model coefficients were frozen except for the final two layers and the classifier layer [4]. This method effectively reduced the risk of common issues like gradient explosion or vanishing gradients, which are often encountered in deep models.

Additionally, we employed an optimization approach with a learning rate scheduler. The scheduler dynamically adjusted and decayed the learning rate during training, improving stability and minimizing the chances of overfitting or converging to suboptimal solutions. Details about the selected hyperparameters used for fine-tuning are provided in appendix C. [1]

During training, the best model was determined based on its ability to maximize the minimum accuracy across all validation matches. This approach was chosen to ensure consistent performance across a diverse set of matches. Traditional validation metrics, such as selecting the model with the highest overall validation accuracy, often resulted in overfitting to validation matches that were similar to the training set. Consequently, this led to significantly poorer performance on validation matches that were less similar to the training data. Our method aimed to address this issue by focusing on the worst-performing validation match, thereby improving the model's robustness and generalization. Given the set of validation match indexes $\mathcal{V}$, this would be formalized as :

$$f^\star = \arg\max_f \left\{ \min_{i \in \mathcal{V}} (\text{accuracy}(Y_i, f(X_i)) \right\} \quad (2)$$

3.3 Decoder-Only Transformers

Unlike encoder-only architectures, decoders employ attention masks that conceal tokens following the current one. This approach makes decoder-based models highly effective for tasks such as in-context learning and sequence generation. In this section, we evaluated whether an event occurred by generating summaries from short tweets (less than 10 words) posted during the relevant time periods.

For this task, we utilized the Llama 3.1 8B Instruct model, a widely adopted open-source model. Its efficiency and compatibility allowed us to run it on standard school GPUs using quantization techniques .

We instructed the model to generate summaries based on the text data from those periods. The primary methodology involved encouraging the model to reason through the problem, rather than simply producing an answer. Previous work [5, 6] has shown the effectiveness of reasoning for those kind of models.

The key advantage of this approach is its interpretability from a human perspective. The summaries provided contained all the essential information relevant to the given period, making the outputs straightforward to explain. However, a significant disadvantage was the model's complexity. This complexity limited our ability to fine-tune it for our specific task, even when employing techniques like low-rank adapters. It is important to highlight that **we used the model only for inference**.

[1]The final model is available on `https://huggingface.co/peulsilva/best_model_tweet_stream`

4 Results

The overall results are presented in Table 1. As anticipated, the Bag of Words (BoW) model demonstrated insufficient complexity for this task, resulting in significantly poorer performance compared to the other approaches. Specifically, it achieved an accuracy of 0.56 on the validation set and 0.72 on the training set. These results indicate that the extracted features were not effective in explaining the target variable. Consequently, we did not evaluate the BoW model on any further datasets.

We also see that the decoder model does not perform as well as the encoder only model (69 % agains 61 % on validation set). For us, this is mainly explained by inconsistent labeling that increases a lot the number of false positive predictions by the LLM (discussed with details on appendix B). The interpretability side of this approach show us that the labeling process was not adequate and is the main reason why the decoder struggles on this task.

Table 1. Comparison of accuracy of models across splits

Split	BoW	Encoder	Decoder
Train	0.72	95 %	67 %
Validation	0.56	69 %	61 %
Test	-	68 %	61 %
Hidden	-	74 %	60 %

References

[1] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones,

Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023.

[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[3] Iz Beltagy, Matthew E. Peters, and Arman Cohan. Longformer: The long-document transformer, 2020.

[4] Jaejun Lee, Raphael Tang, and Jimmy Lin. What would elsa do? freezing layers during transformer fine-tuning, 2019.

[5] Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step, 2023.

[6] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models, 2023.

A Description of original dataset

Table 2. Statistics of Matches

Match ID	Number of Tweets	% of Positive Labels
0	42k	59%
1	974k	54%
2	87k	60%
3	272k	45%
4	713k	44%
5	526k	57%
6	285k	-
7	95k	51%
8	148k	75%
9	113k	-
10	824k	36%
11	314k	60%
12	97k	57%
13	368k	53%
14	99k	68%
15	628k	-
16	45k	-
17	256k	43%
18	86k	56%
19	156k	55

B Inconsistencies found in the labels

The objective of this challenge was not clear from the beginning: one can think that we want to predict **the exact moment an event happened** based on tweets on a time window. However this could not be the case, since there are in average 100 positively labeled periods per match, but a match has almost never more than 15 events.

The other possibility is that the task is to predict wether if at least one of the tweets on a given period mentions a football event. A priori this makes sense with the data we were provided. However, during the exploration, and specially with the aid of LLM-generated summaries (as described on section 3.3) we noted **lots** of samples that should be positively labeled and wetr not. Some examples are shown below.

Germany vs Algeria PeriodID 52

Tweet: `YELLOW TO HALLICHE`
Label: 0
Real event: Rafik Halliche received a yellow card on minute 42

Germany vs France PeriodID 36

Tweet: `Goal Hummels`
Label: 0
Real event: Mats Hummels scored for Germany on minute 13

Argentine vs Germany PeriodID 162

Tweet: `Its Over!!! YASSSSS GOT THISS!!!`
Label: 0
Real event: Germany won the World Cup on Overtime, justifying the enthusiasm of the fan

C Hyperparameters

C.1 Bag of Words

The vocabulary was composed of 35 domain-specific words :

```
"goal", "penalty", "halftime",
"full-time", "yellow", "red",
"kickoff", "extra time",
"stoppage time", "foul",
"offside", "handball", "save",
"tackle", "dribble", "corner",
"substitution", "header",
"free kick", "throw-in",
"assist", "hat-trick", "own
goal", "victory", "defeat",
"draw", "win", "loss", "tie",
"comeback", "goalkeeper",
"striker", "midfielder",
"defender", "referee", "fans",
"var", "gooal"
```

And more 30 most frequent tokens on the train set. We tested Logistic Regression, Random Forest, Decision Tree and MultiLayerPerceptron models. The best results were obtained with the Random Forest model.

C.2 Encoder-only

The LongFormer model was trained on a single NVIDIA RTX A5000 over 20 epochs with a learning rate of 10^{-5}, with decay factor of 0.1, with batch size 8.

C.3 Decoder-only

We used Llama 3 - 8B Instruct model for inference with 4bits quantization.

D Prompts used

In the approach described on section 3.3, we used the following prompt to obtain the responses.

```
 You are a helpful AI tasked
with analyzing a collection of
tweets posted during a single
minute of a football match.
Your goal is to generate a
concise summary of the key
events that occurred during this
time and to specifically answer
whether any of the following
events occurred:

 1.  A goal (including who
scored, if mentioned).
2.  A yellow or red card
(including the player or team,
if mentioned).
3.  A kickoff (start of a half
or after a goal).
4.  Halftime or fulltime
whistle.

 Here are the tweets:
{tweets}

 Instructions:
1.  Analyze the tweets for clear
indications of the above events
using common football-related
terminology, phrases, or
hashtags.
2.  If the event is ambiguous
or not explicitly stated in
the tweets, mark it as "Not
mentioned."
3.  Summarize any additional
significant match events or fan
reactions from the tweets that
are relevant to understanding
the minute.
4.  Return the response in the
following structured format:

{format}

 Try to reason as much as
possible and avoid simple and
quick answers
```