| Sub-event Detection in Twitter Streams  INF554 Data challenge    **Team Monad**  Boyuan Zhang  Yanxu Zhao  Côme Vincent |
| --- |

[**Context 1**](#_qrwgdbtpvuzg)

[**1. Data Preprocessing and Feature Selection/Extraction 1**](#_e3n3pqb4287)

[1. Preprocessing 1](#_crg3ddjua4qp)

[2. Feature Motivation and Intuition 1](#_34u1hwsg32td)

[**2. Model Choice, Tuning and Comparison 2**](#_q5oe7s4iq8de)

[2.1 Background and Motivation 2](#_e4eo74t0cl30)

[2.2 Logistic Regression & Random Forest Model 2](#_oudbwf6fghbp)

[2.3 LSTM Model 2](#_9tdsv9ssmy1y)

[2.4 Transformer 3](#_laignuynas4b)

[2.5 Model Comparison 3](#_ov2aeexsilfz)

[3. Conclusion and improvements 3](#_sxzlz6io4lwn)

[**Appendix 4**](#_jfgahiln8dqx)

[1. Tables 4](#_4ydqsd50jpu8)

[fig 1.a : Representation of the hyperparameter search conducted on the LSTM. Each dot is a trained model. 4](#_8ejyzf6ttt3z)

[fig 1.b : Representation of the hyperparameter search conducted on the LSTM. Each dot is a trained model. 4](#_n6rn9woa0oqi)

[2. References 5](#_euehxdv7562a)

[3. Notes on the codebase 6](#_qputx1ggf0xz)

[Efficiency 6](#_wv5mv71m14gb)

[Clean code 6](#_2onccttvij1a)

[Working with git 6](#_hvicovuhpplv)

# Context

We describe in this report the construction of a machine learning pipeline designed to train a model capable of accurately detecting sub-events (meaningful happenings) during football matches by classifying tweets. Training data consists of 12 labeled matches, with 4 additional unlabelled matches used for evaluation. We take inspiration from the work of G. Bekoulis *et al.* [1]

# 1. Data Preprocessing and Feature Selection/Extraction

## 1. Preprocessing

Here we normalize the content of the tweets to keep only meaningful data :

* Lowercasing and Lemmatization: standardized text to lowercase and reduced words to their base forms (e.g., `running` to `run`) for consistency ;
* Duplicate Removal: dropped duplicate tweets to ensure data uniqueness : keeping duplicates not only multiplies the amount of rows we have to process, but also reduces our models accuracy (~10% for the LSTM) ;
* Stop Word Removal: filtered out common stop words (e.g., `the`, `and`) to reduce noise (non informative data) ;
* Transforming non textual content: URLs, user mentions and emojis are all changed according to the embedding model used (i.e. URLs -> ‘HTTPURL’ for bertweet).

This preprocessing has been carefully chosen and adapted to the next step (*e.g.* changing URL formatting to fit expected input for BERT encoding) because it has been shown to provide significant improvement on classification accuracy [2].

## 2. Feature Motivation and Intuition

Feature engineering, or the choosing the right data to use during our predictions, is essential to accurate classification [3].

The first obvious information to give to the model is the content of the tweet that has been normalized in the first step. However, using raw text is proven to be less efficient than tokenizing and embedding it [9], which lets us represent the meaning of words through vectors.

We tried two encoders before finding the right one :

* GloVe : good at capturing context using word to word occurrences, but cannot embed out of vocabulary words [9].
* Paraphrase MPNet and BERT base : pre-trained language model based on the BERT architecture proposed by J. Devlin *et al.* [4] that are often used in the state of the art of NLP [5].

We finally decided to use BERTweet, a model using the same BERT-base architecture but has been trained specifically for tweet encodings. It has been shown to outperform regular BERT models for tweet based NLP tasks [6].

# 2. Model Choice, Tuning and Comparison

## 2.1 Background and Motivation

The goal of this task is to classify events based on textual information from tweets. To achieve this, we explored four different classification models: Logistic Regression and Random Forest, a Long Short Term Memory (LSTM) model and a transformer (Self-Attention Mechanism). This section outlines the performance comparison and discusses the strengths and weaknesses of each model.

## 2.2 Logistic Regression & Random Forest Model

The baseline algorithm used was a logistic regression (LR). This algorithm is one of the earliest methods of text classification [7] but expects data points to be independent [8], which is why we decided to try a random forest. They have the advantage of being fast to train compared to other deep learning methods [8]. However the lack of accuracy (same results as LR) and temporal component led us to search for a new model.

## 2.3 LSTM Model

LSTM (Long Short-Term Memory) is a type of recurrent neural network (RNN) designed to handle sequential data. It is particularly effective in capturing temporal dependencies and contextual relationships.

Selecting the right set of Hyper Parameters for a ML model are essential to obtain best results [10]. There are multiple ways of fine tuning hyper parameters. Amongst them are :

* Grid search : methodical search of a whole hyper space of parameters;
* Manual search : tweaking of parameters by hand;
* Random search : exploring the subspace randomly.

Given the same amount of computational power, random search gives better results than grid search [11].

During training we have used a combination of random and manual search, which is an efficient way of reaching best parameters [12]. We first test a lot of combinations using random search, then manually explore the subspace that brought the best results. For this model we have 5 dimensions of hyper parameters (with search range in brackets):

* Model architecture : number of layers [1, 3] and neuron per layers [100 - 300];
* Training parameters : learning rate [0.001, 0.01], number of epochs [500, 3000], batch size [32, 128].

Results (available in fig 1.a and 1.b) helped us steer the manual search in the right direction.

## 2.4 Transformer

The core innovation of the Transformer architecture lies in the introduction of the self-attention mechanism, which enables the model to directly capture dependencies between different positions in the input sequence without relying on traditional recurrent neural networks (RNNs) or convolutional neural networks (CNNs) [13].

We have tried training a BERT model [4] to classify tweets but the algorithm being very computationally heavy led to unwanted interruptions of the training overnight. In the restricted time available to use we only succeeded in training the model on 10% of the available data, after train/test split. These limitations led to the model only achieving 62% accuracy.

## 2.5 Model Comparison

Our model comparison highlighted trade-offs between efficiency and accuracy:

* Logistic Regression & Random Forest: These baseline models were quick to train but failed to capture temporal dependencies, yielding similar, suboptimal accuracy.
* LSTM Model: With strong temporal modeling capabilities, the LSTM outperformed other models, achieving the highest accuracy. A combination of random and manual hyperparameter tuning maximized its performance, making it the most effective within our constraints.
* Transformer-Based Model: While promising due to its self-attention mechanism, the high computational cost limited its training to a fraction of the dataset, resulting in only 62% accuracy.

# 3. Conclusion and improvements

This project highlights the practicality of LSTM models for classifying sub-events in football matches using tweets. Their ability to model temporal dependencies, combined with reasonable computational demands, makes them highly effective for this type of task.

Our findings suggest that transformer models, while potentially more powerful, require substantially more resources to fully realize their potential. Future research should explore strategies to mitigate these constraints, such as leveraging more efficient transformer variants or distributed training setups.

Further work could be done to improve the LSTM classification. We believe using Beginning, Inside, Outside (BIO) tags like G. Bekoulis *et al.* [1] instead of simple binary classification could better leverage the temporal capabilities of LSTMs. With enough time we believe a full transformer model might bring equivalent or better results than the LSTM. We would also like to explore weight decay to prevent overfitting in our LSTM model, which would let us run more epochs and potentially have better overall results [14]. We believe that more feature engineering can be done, for example taking into account the number of tweets per minute when predicting. We have implemented the necessary code but haven’t had time to properly test the effectiveness of it.

# Appendix

## 1. Tables

#### fig 1.a : Representation of the hyperparameter search conducted on the LSTM. Each dot is a trained model.

#### fig 1.b : Representation of the hyperparameter search conducted on the LSTM. Each dot is a trained model.

## 2. References

[1] G. Bekoulis, J. Deleu, T. Demeester, and C. Develder, “Sub-event detection from twitter streams as a sequence labeling problem,” in *Proceedings of the 2019 Conference of the North*, Stroudsburg, PA, USA: Association for Computational Linguistics, 2019. Accessed: Dec. 12, 2024. [Online]. Available: <https://doi.org/10.18653/v1/n19-1081>

[2] A. K. Uysal and S. Gunal, “The impact of preprocessing on text classification,” *Information Processing &amp; Management*, vol. 50, no. 1, pp. 104–112, Jan. 2014, doi: 10.1016/j.ipm.2013.08.006.

[3] C. Wang, P. Nulty, and D. Lillis, “A Comparative Study on Word Embeddings in Deep Learning for Text Classification,” in *Proceedings of the 4th International Conference on Natural Language Processing and Information Retrieval*, New York, NY, USA: ACM, Dec. 2020. Accessed: Dec. 12, 2024. [Online]. Available: <https://doi.org/10.1145/3443279.3443304>

[4] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding,” arXiv.org. [Online]. Available: <https://arxiv.org/abs/1810.04805>

[5] M. Mars, “From Word Embeddings to Pre-Trained Language Models: A State-of-the-Art Walkthrough,” *Applied Sciences*, vol. 12, no. 17, p. 8805, Sep. 2022, doi: 10.3390/app12178805.

[6] D. Q. Nguyen, T. Vu, and A. T. Nguyen, “BERTweet: A pre-trained language model for English Tweets,” arXiv.org. [Online]. Available: https://arxiv.org/abs/2005.10200

[7] *Analysis of Binary Data, Second Edition*. Routledge, 2018. Accessed: Dec. 12, 2024. [Online]. Available: <https://doi.org/10.1201/9781315137391>

[8] K. Kowsari, K. Jafari Meimandi, M. Heidarysafa, S. Mendu, L. Barnes, and D. Brown, “Text Classification Algorithms: A Survey,” *Information*, vol. 10, no. 4, p. 150, Apr. 2019, doi: 10.3390/info10040150.

[9] J. Pennington, R. Socher, and C. Manning, “Glove: Global Vectors for Word Representation,” in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Stroudsburg, PA, USA: Association for Computational Linguistics, 2014. Accessed: Dec. 12, 2024. [Online]. Available: <https://doi.org/10.3115/v1/d14-1162>

[10] L. Yang and A. Shami, “On hyperparameter optimization of machine learning algorithms: Theory and practice,” *Neurocomputing*, vol. 415, pp. 295–316, Nov. 2020, doi: 10.1016/j.neucom.2020.07.061.

[11] A. Nugroho and H. Suhartanto, “Hyper-Parameter Tuning based on Random Search for DenseNet Optimization,” in *2020 7th International Conference on Information Technology, Computer, and Electrical Engineering (ICITACEE)*, IEEE, Sep. 2020, pp. 96–99. Accessed: Dec. 12, 2024. [Online]. Available: <https://doi.org/10.1109/icitacee50144.2020.9239164>

[12] F. Hutter, H. H. Hoos, and K. Leyton-Brown, “Sequential Model-Based Optimization for General Algorithm Configuration,” in *Lecture Notes in Computer Science*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 507–523. Accessed: Dec. 12, 2024. [Online]. Available: <https://doi.org/10.1007/978-3-642-25566-3_40>

[13] Y. Liu *et al.*, “RoBERTa: A Robustly Optimized BERT Pretraining Approach,” arXiv.org. [Online]. Available: <https://arxiv.org/abs/1907.11692>

[14] SrivastavaNitish, “Dropout: a simple way to prevent neural networks from overfitting: The Journal of Machine Learning Research: Vol 15, No 1,” The Journal of Machine Learning Research, Jan. 2014.

## 3. Notes on the codebase

### Efficiency

Because machine learning algorithms are very expensive computation wise, using all the computing power available to us was paramount to this project. That is why very early we ensured that all processes that could be parallelized would be. For example the preprocessing steps can be carried out on each file separately, so we used the multiprocessing library of python to start a new process per core that would each process a different file.

But parallelizing isn't the only answer to limited time and compute power. When possible we also saved the results of expensive computation to binary files on the disk, using the pickling abilities of the pandas library.

### Clean code

The provided baseline code was a relatively concise, one file python script. However with complexification of the prediction pipeline the code became less readable. We decided to rewrite the whole codebase into multiple files and classes that could be instantiated with different parameters. In combination with the pickling mentioned above, this has proven to be very useful to try new tools in the pipeline, like for example modifying the embedding stage. We would only have to save the pickled data somewhere, change parameters in the targeted class, and observe the results.

### Working with git

As all non trivial codebases need an efficient version control system. We have decided to use git in conjunction with github to host our project and let us work as a team more efficiently. The repository can be publicly accessed here :

<https://github.com/FBI-openup/Data-challenge-Sub-event-Detection-in-Twitter-streams/>