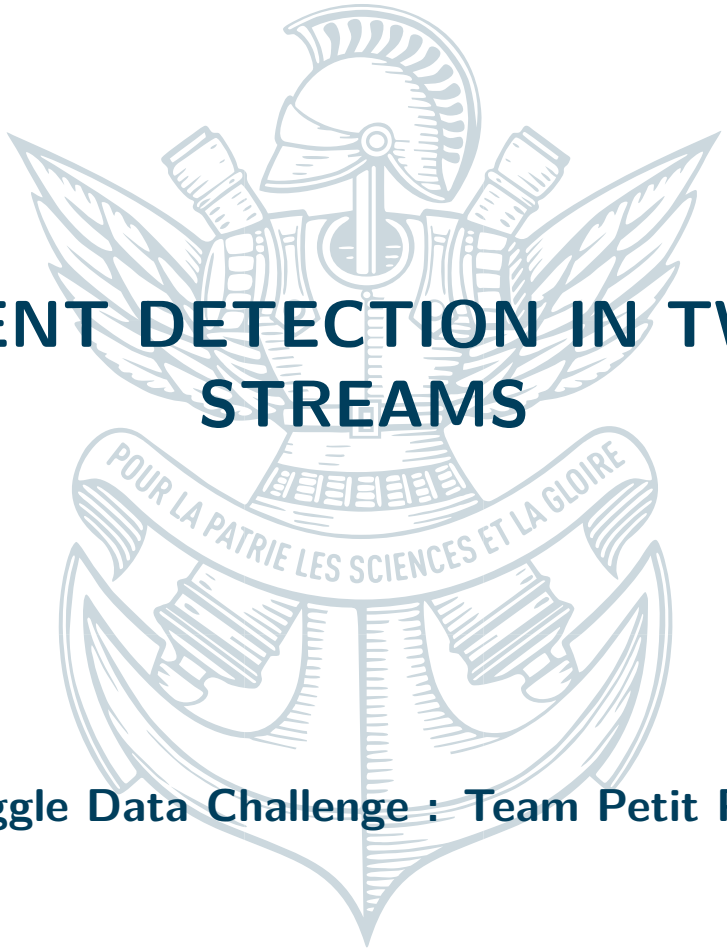

SUBEVENT DETECTION IN TWITTER STREAMS

Kaggle Data Challenge : Team Petit Pois

December 12, 2024

Arthur Compoint, Adrien Goldszal, Théodore Halley
Authors can be reached at first.lastname@polytechnique.edu

ÉCOLE POLYTECHNIQUE

IP PARIS

CONTENTS

1 DATA PREPROCESSING AND FEATURE SELECTION/EXTRACTION

1.1 Respository

Please find the repository associated to this work here : Sub-event detection in twitter streams

As it needs an OpenAI API Key, please find one attached in the apikey.txt.file

1.2 Data preprocessing

Social media data, such as tweets, are inherently noisy and heterogeneous, often including unwanted content like spam, advertisements and malicious material. Knowing this, we decided to follow the method suggested by Meladianos et al (2018) [1] to preprocess the raw tweet data.

Filtering out tweets From both the training and evaluation data, we removed all retweets and duplicate posts as these often replicate content without adding value, thereby reducing noise. Additionally, tweets containing "@"-mentions were excluded, assuming these are generally irrelevant to the specific event of interest.

Tweet preprocessing The remaining tweets were subjected to standard text preprocessing steps, to reduce variation and complexity as much as possible without loosing too much information, in order :

- lowercasing. there is no specific need to detect or learn uppercase letters in this setting.
- text cleaning : numbers, special characters and URL's don't add to the sentence meaning and won't help for sub-event detection.

- tokenization : we decided to use the NLTK tokenizer instead of just splitting with whitespace, as it handles contractions better (and negations, which could be useful)
- Stopword removal and lemmatization : here as well, we use the NLTK WordNetLemmatizer, which takes context better into account and returns valid words.

Testing the new preprocessing on the simple logistic regression from the baseline already showed promising results and a better classification.

Model	Without preprocessing	With preprocessing
Logistic Regression	0.65234	**0.67968**

Table 1: Preprocessing impact on evaluation

As a disclaimer, this method was used in every model tested except the BERT embeddings (see further down for details), where we we simply removed URLs as Bert has got its own tokenizer, is case-sensitive and handles punctuation.

1.3 Dataset analysis

A first analysis of the dataset shows us that, in fact, the events detected by the tweet summaries of the graph-of-words method from [1] we were given can be very erratic. An actual event can be tweeted about before or multiple times after its real timestamp and be considered a positive sub-event. The Argentina-Belgium match is a good example for this, where, from the 7 real events in the match [1], the actual positive sub event distribution is as shown in **Figure 2** (Appendix). We can see that in reality, more than 20 peaks are detected, sometimes in close succession.

With the idea that important events would possibly be correlated with the content of the tweets and the number of tweets, we decided to analyze the tweet count per match minute, as well as the number of tweets mentioning the 8 subevents to see if these can be pertinent features to analyze and to what extent. **Figure 3** (Appendix) shows this trend confirmed and illustrates it for two matches. Large events are associated with clear peaks of tweet numbers, and the percentage of tweets with important keywords spikes as well. Moreover, Match 1 enables us to validate the importance of the preprocessing. We can see that for some matches with large amounts of events, the non preprocessed tweets create a large amount of noise that makes the peaks indistinguishable.

1.4 Feature selection : 2 main approaches

1.4.1 • Word counts & Volume based features

The first approach experimented focused on leveraging the link between tweet count per minute as well as the number of tweets containing the subevents, to classify the tweets. The timestamp is also considered as a feature. We tried renormalizing the features as well to see if this would have an impact on classification, as doing so would prevent feature domination and avoiding anomalies, but this didn't seem to have much of an impact. In general, the performance of the classifiers were not up to par with what seemed possible through a direct consideration of the tweet content.

1.4.2 • Tweet embedding based features

Another possibility we explored consisted in using the tweets themselves. For this, we followed standard Natural Language Processing pipelines by computing embeddings based on every tweet. We decided to compare different

Model	Test with renormalization	Test without	Eval
Logistic Regression	0.623	0.629	
Random Forest	0.6945	0.711	**0.67187**
SVM	0.648	0.644	
XGBoost	0.696	0.697	

Table 2: Volume-based features : classifier results

types of embedding models, namely : GloVe200, GloVe50, and BERT and compare their effects on classification.

We then proceed with the embedding of the tweets using all three models and group tweets by period for each match by computing the average embedded vector for each period. We therefore end up with 3 different types of embeddings for both the training and evaluation sets.

Model	Dummy	Logistic Regression	Random Forest	SVM	XGBoost	Bagged SVM
GloVe-50	0.53271	0.72118	0.75234	0.7134	0.74766	0.65732
GloVe-200	0.53271	0.73832	0.7648	0.72897	0.76636	0.67913
BERT	0.53271	0.77259	0.77103	0.77882	0.78505	0.67601

Table 3: Results of the 3 different embeddings on a testing set (not evaluation data).

Model	Dummy	Logistic Regression	Random Forest	SVM	XGBoost	Bagged SVM
GloVe-200	0.62890	0.68750	0.67187	0.67187	0.64843	0.66796
BERT		0.52343		0.52734		

Table 4: Results on the evaluation set

Analyzing the results, we can see that despite very good training accuracy on BERT embeddings, there is a strong overfitting compared to GloVe-200. This could be due to a noticeable distribution shift between the train/test data and the evaluation matches.

2 MODEL CHOICE, TUNING AND COMPARISON

2.1 Machine learning based approaches (non Deep Learning)

As the BERT embeddings seemed to make the classifiers strongly overfit, we decided to focus on using the GloVe200 embeddings. We tested, as seen before, multiple machine learning methods and compared them : Logistic Regression, SVM, RandomForest and XGBoost. Although the overfitting was much smaller than with BERT, the results in the evaluation set were still systematically worse than in the testing set. Furthermore, the results using Natural Language Processing pipelines were quite disappointing as they did offer only slightly better results than a basic tweet count and word count.

Random Forest : The Random Forest method gave us good results, despite it being less than SVM. We tried varying the different parameters, notably the number of estimators, the maximal depth and maximum amount of features, mainly by testing a wide array of such parameters. Nevertheless, we were not able to improve our score of **0.67187**, it mainly came down to optimizing computation time.

XGBoost : We tried varying learning rate, as well as the model's alpha and lambda parameters to reduce the overfitting in XGBoost which was quite large especially compared to the other methods.

SVM : As seen in 1.3.2, the SVM was the best overall classifier, on a linear kernel. A nonlinear rbf kernel also seemed to greatly reduce accuracy. We varied the regularization parameter by performing a basic grid search to get the best performance.

2.2 Neural network based approaches

LSTM's : Taking inspiration from Bekoulis et. al [2], who tried tackling the same problem, we tried using a neural network as a classifier for our data. In his work, Bekoulis shows how an added chronological LSTM to tweet-average embeddings can improve performance at what he calls "bin-level", ie minute level precision. A chronological LTSM takes into account temporal dependencies, which can be crucial in a soccer match. A simple chronological LSTM with a sequence length of 5 returned an accuracy around the one from the baseline. But taking longer sequence lengths did not increase accuracy at all. There was a clear overfitting problem after more than 30 epochs of training. A more thorough implementation with a BIO tagging scheme [3] as was detailed in the paper, seemed necessary to capture time dependencies better, and was tried. However, it overfit even more quickly and efforts to limit it were unsuccessful. A more thorough analysis would have been conducted had their been more time.

Model	Eval
Tweet-AVG LSTM	0.65625
Tweet-AVG LSTM with BIO tagging	0.53515

Table 5: LSTM Results on the evaluation

2.3 LLMs

DSPy We wanted to try using an LLM framework to compare its results with our machine learning method. The framework we decided to adopt is called DSPy [3]. It is a recent (2023) framework that focuses on "programming with structured and declarative natural-language modules" rather than trying to work with often complex prompt engineering. Furthermore, it does not necessarily require training (or "optimizing" as DSPy developpers call it) and if you do decide to optimize it, you do not need a lot of data to do so. We decided to experiment with it using the language model GPT-4o-mini.

Unsuccessful training Our first attempt consisted in using a trained version of our DSPy module trained on a few (15) tweets from the training set selected at random to read one by one each tweet in the evaluation set and say whether the tweet references an event occurring or not. We then grouped the tweets by match and period and said that the period corresponds to an event if at least 1 tweet references an event happening. Unfortunately, we were not able to proceed with this method as 1) it took way too long to individually process each tweet and 2) the training is flawed as we select a random sample of tweets but a tweet on the training set is labeled 1 if the period in which it was posted contained an event and not if the tweet itself made reference to an event occurring.
We tried to fix problem 1) by processing only n randomly selected samples per match period for $n = 5$ and $n = 10$ however this method only yielded a relatively poor result on the evaluation data of 0.66015 for $n = 10$ and 0.66406 for $n = 5$. We wanted to try to run the same process on the whole training set or at least part of if but we were confronted to huge run times (circa 40h just for 5 tweets per period) and were not able to do so,

therefore preventing us from trying to figure out where exactly the algorithm needed improvement.

Leveraging graph-based summaries However, later on in the project we found a solution by leveraging the summarization module of paper [1], leveraging a graph based method to return a relevent summary for every minute of every match of all the tweets per period. Our new algorithm therefore consisted in summarizing one match picked at random for training purposes, training the DSPy module on it (note this fixes problem 2), summarizing the evaluation tweets per period and going through each period asking the module to output a 1 if an event is detected happening in the period and a 0 otherwise (note that this also fixes problem 1). This led to a much better result : 0.71484. Seeing this was yielding better results, we decided to train our DSPy module on more than one match, namely 4. This once again led to significant a improvement : 0.73046 on the evaluation set, our best result overall.

Model	Eval
Randomized training	0.66406
Training with 1 summary	0.71484
Training with 4 summaries	**0.73046**

Table 6: LLM Results

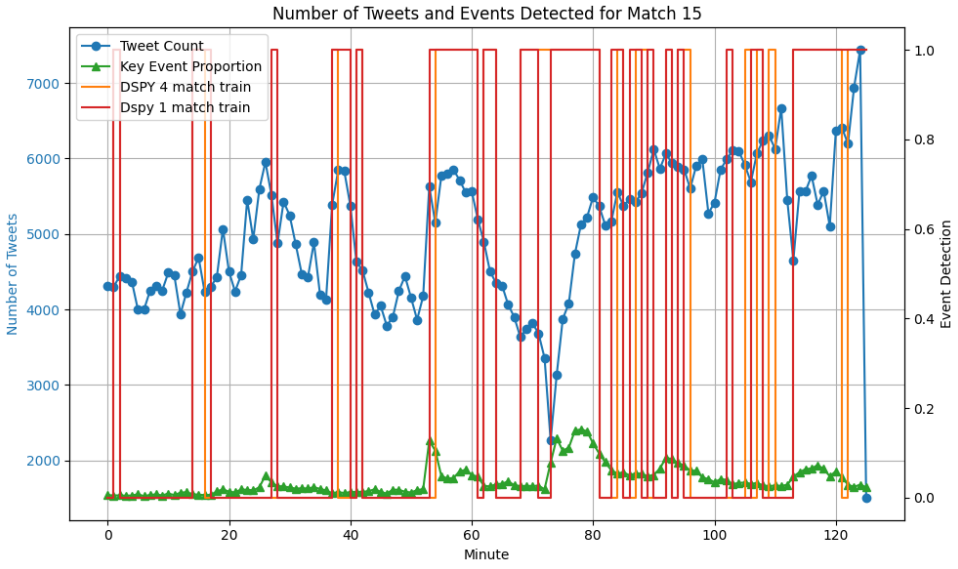

Figure 1: Results of our two best LLM predictions on a match

3 APPENDIX

3.1 Dataset Analysis

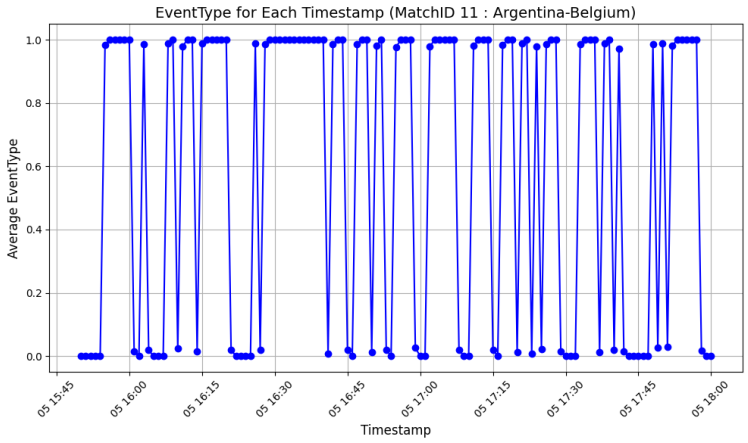

Figure 2: Events and their labels for the Argentina Belgium match

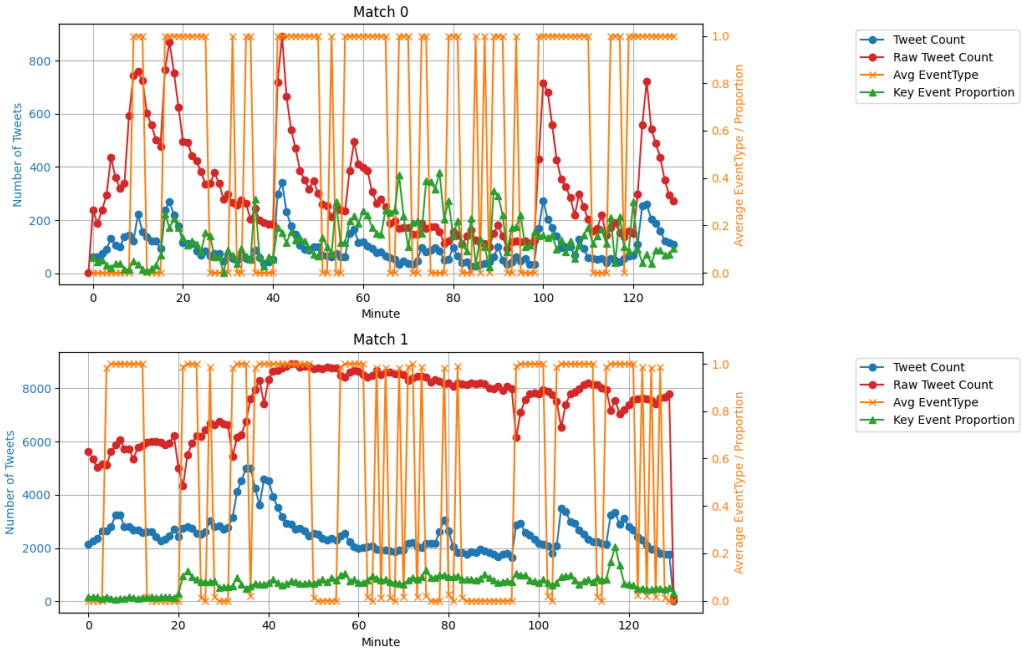

Figure 3: Unprocessed vs processed tweets vs processed with keywords

REFERENCES

[1] Meladianos, P., Xypolopoulos, C., Nikolentzos, G., & Vazirgiannis, M. (2018). 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 (pp.481-493). Springer International Publishing.

[2] Giannis Bekoulis, Johannes Deleu, Thomas Demeester, and Chris Develder. 2019. Sub-event detection from twitter streams as a sequence labeling problem. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pages 745–750, Minneapolis, Minnesota. Association for Computational Linguistics.

[3] Lance Ramshaw, Mitch Marcus. 1995. Text chunking using transformation-based learning. In Third Workshop on Very Large Corpora.

[4] DSPy: Compiling Declarative Language Model Calls into Self-Improving Pipelines Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan, Saiful Haq, Ashutosh Sharma, Thomas T. Joshi, Hanna Moazam, Heather Miller, Matei Zaharia, Christopher Potts