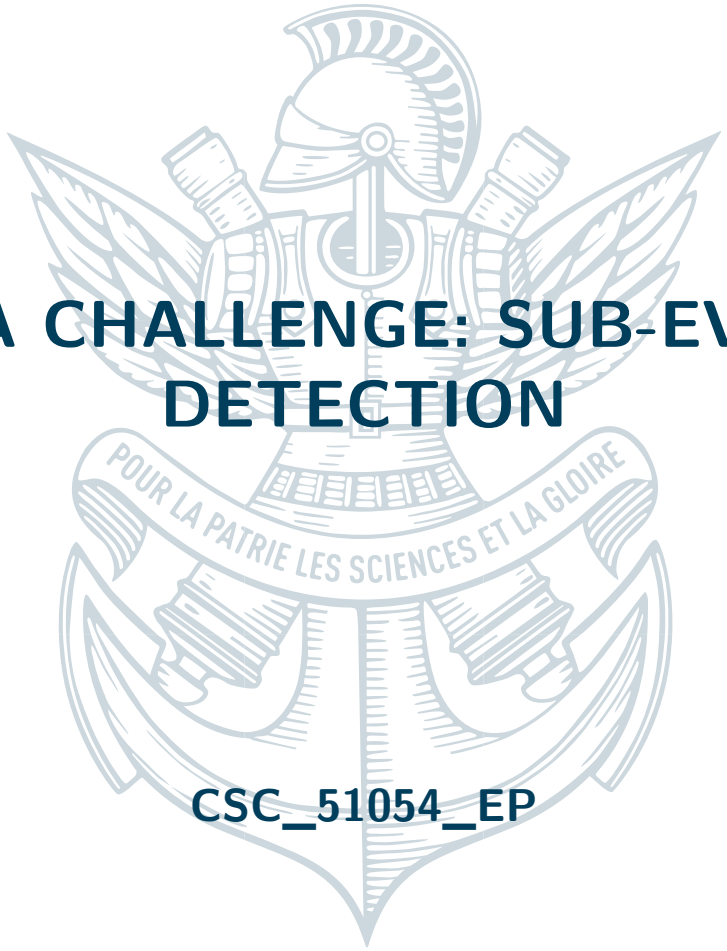

DATA CHALLENGE: SUB-EVENT DETECTION

CSC_51054_EP

doomsday

Franck BAMBOU - Harold NGOUPEYOU - Loïc TABUEU

TABLE DES MATIÈRES

1
DATA PREPROCESSING AND FEATURE ENGINEERING

1.1 Data Cleaning and Preprocessing

To ensure a high-quality dataset, we began by conducting a manual exploration followed by visualizations using **words clouds**. This revealed the presence of numerous spam tweets common to both classes. Consequently, our first task was to clean the dataset. We introduced a **spam score** for each tweet, based on patterns such as phrases like *"give away free"* and the presence of excessive hashtags. Tweets with a high spam score were removed. Additionally, retweets containing *"RT @"* were excluded to reduce noise further and ensure a more focused dataset.

The **preprocessing** steps were tailored to the expected requirements of our feature extractors. These steps globally included the removal of URLs, numbers, mentions, punctuation, stopwords, and emojis. For word vector models, lemmatization and lower casing were applied. In contrast, for sentence transformer models, there were skipped to preserve the original context.Moreover, for word vectors, text was tokenized into individual words.

Furthermore, we removed words that appeared very infrequently in the vocabulary and eliminated the 10 most frequent words, as these were not discriminative for the classification task. After these steps, the dataset size was reduced by more than half, providing a cleaner and more compact dataset for subsequent analysis and modeling.

1.2 Feature Extraction

During the challenge, we focused on two types of embeddings : **word embeddings** and **sentence embeddings**. These embeddings were chosen to capture different levels of semantic information from the tweet. Each type was selected based on its suitability for representing textual data in a meaningful way.

- ### Word Embeddings

For this, we selected first **GloVe Twitter (200d)**. Research has shown the effectiveness of GloVe embeddings in tweet classification tasks, as highlighted in studies such as [1], which demonstrates their superior performance in representing short, informal text.
However, upon evaluating GloVe's coverage on our dataset, we observed that it covered roughly 72% of the vocabulary, the unknown words include mainly acronyms and compound words. This limitation prompted us to include **FastText** as an additional embedding. FastText was chosen for its ability to generate embeddings for out-of-vocabulary (OOV) words by leveraging subword information[2], a crucial advantage for noisy text like tweets. This combination allowed us to handle misspellings, variations, and unseen words more effectively.

To construct the final word embedding for each token :
— If the word exists in both GloVe and FastText, we concatenated their respective embeddings ;
— If the word is not available in GloVe, only the FastText embedding was used ;
— For OOV words, we replaced them with embedding for the word *something* as a fallback, ensuring no information was lost.

- Sentence Embeddings

For sentence embeddings, we utilized the **all-MiniLM-L6-v2** model, a state-of-the-art transformer-based model from the sentence-transformers library. This model was chosen for its balance between computational efficiency and performance in semantic similarity tasks[3]. Unlike traditional word embeddings, which represent words independently, the sentence-transformer generates contextual embeddings that encode the semantic meaning of the entire tweet, capturing relationships between words in their broader context.

The decision to use this specific model rather than just averaging BERT embeddings is supported by [4], which demonstrates the effectiveness of transformer-based sentence embeddings in improving downstream NLP tasks. The all-MiniLM-L6-v2 variant further optimizes this approach by distilling a larger model, enabling faster inference while maintaining competitive performance.

1.3 Experimental Analysis of Features

We conducted experiments on the performance of different feature sets, including the hybrid combination of GloVe + FastText embeddings, the sentence embeddings and the concatenation of both embedding types. The evaluation was performed using an **80-20 train-test split**, with **accuracy** on the test set as the primary metric.

For this task, we selected **XGBoost** as our primary model due to its strong performance in text classification tasks, particularly for unbalanced and high-dimensional datasets.

Without feature selection, we observed the following results :
— Hybrid embeddings : **75%**
— Sentence embeddings : **79%**
— Concatenation of both : **76%**
Next, we applied three feature selection techniques to further refine the feature set : **PCA**, **feature importance-based** selection (relies on XGBoost's inherent ability to compute feature importance by evaluating each feature's contribution to reducing impurity and improving predictive accuracy) and a **hybrid method** combining SelectFromModel (scikit-learn) with Lasso regularization.

The results of these experiments were as follows :
— PCA : **80.14%**
— Feature importance-based selection : **80.84%**
— Hybrid Method : **78%** (performance likely suboptimal due to non-optimized regularization strength)

2
MODEL CHOICE, TUNING AND COMPARISON

2.1 Overview of Models Evaluated

To identify the most effective model for our task, we tested several algorithms commonly used in the literature for similar task and compared their performance to the previously evaluated XGBoost model.

— **Logistic Regression :** To address the risk of overfitting, we applied feature selection combined with L1 regularization. this approach achieved a maximum accuracy of **76%**

— **Random Forest :** Leveraging its inherent capability to rank features by importance, we applied feature importance-based selection, which produced a maximum test accuracy of **78%**
— **Deep Neural Network :** For a more complex model like this one, we introduced a dropout layer with a rate of 0.3 and added L2 regularization with a weight decay of 0.01 to prevent overfitting. This approach resulted in a maximum accuracy of 75%, indicating that the increased complexity did not significantly improve performance, likely due to the dataset size.
— **Fine tuned twitter-roberta-base-sentiment-latest :** We utilized the RoBERTa-base model from Cardiff NLP, which was trained on approximately 124 million tweets. This model was chosen for its inherent capability to analyze tweet structures and extract essential information. To adapt it for binary classification, we modified its classification head. To prevent overfitting, we closely monitored the training and validation losses throughout the process. Additionally, we dynamically fine-tuned key hyperparameters, including the number of epochs, learning rate, and weight decay. Despite these efforts, our model achieved a public score of approximately 67% on Kaggle.

From these experiments, we concluded that XGBoost was the most effective model for our task, achieving the highest accuracy. Consequently, we selected it as our primary approach for subsequent analysis.

One of the key challenges in our dataset was the class imbalance, with class 1 being significantly more prevalent than class 0. This imbalance led to a consistently low F1 score for class 0, indicating the model's difficulty in effectively identifying minority class instances. To mitigate this, we employed **SMOTE** (Synthetic Minority Oversampling Technique), a method that generates synthetic samples for the minority class (class 0) by interpolating between existing data points using a **k-nearest neighbors** algorithm. By applying SMOTE, we aimed to balance the class distribution and improve the model's ability to correctly classify minority class instances. This effectively ted to a balanced F1 score between the two classes.

2.2 Hyperparameter Tuning

Using XGBoost with the sentence embeddings, we initially achieved a public score of **0.703** on Kaggle, which was lower than the test accuracies observed during evaluation. Upon analyzing the loss curves, we identified signs of overfitting. To address this, we implemented a combination of feature selection based on SHAP values and hyperparameter tuning.

For hyperparameter tuning, we utilized the **Optuna** library, which employs advanced Bayesian optimization techniques. This approach was chosen for its ability to efficiently explore the parameter space while dynamically balancing exploration and exploitation, leading to more effective and faster optimization. We conducted the tuning by optimizing the **AUC** metric instead of the accuracy which may not reflect our model's ability to distinguish between class effectively. Among the hyperparameters tuned, we have **max_depth** which controls the complexity of the tress, **gamma** which serves as a regularization mechanism, determining the minimum loss reduction required to make a split and **subsample** which specifies the fraction of training samples used for each tree. The list of all tuned parameters, along with their ranges, is provided in the appendix.

Additionally, during the **9-fold cross-validation** for the final model, we introduced **early stopping** with a patience of 5 epochs to prevent overfitting during training. This mechanism allowed the model to terminate training early if the validation loss did not improve, ensuring better generalization. The final cross-validation results showed an average **train log loss of 0.49** and a **test log loss of 0.54**, demonstrating a significant improvement in the model's robustness ; however, reducing overfitting also led to a slight reduction in overall performance.. This led to a public score of roughly **0.72**

RÉFÉRENCES

[1] R. ALRashdi and S. O'Keefe, "Deep learning and word embeddings for tweet classification for crisis response," *arXiv preprint arXiv :1903.11024*, 2019.

[2] T. Mikolov, E. Grave, P. Bojanowski, C. Puhrsch, and A. Joulin, "Advances in pre-training distributed word representations," *arXiv preprint arXiv :1712.09405*, 2017.

[3] A. Abdaoui and S. Dutta, "Attention over pre-trained sentence embeddings for long document classification," *arXiv preprint arXiv :2307.09084*, 2023.

[4] N. Reimers and I. Gurevych, "Sentence-bert : Sentence embeddings using siamese bert-networks," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Association for Computational Linguistics, 2019, pp. 3982–3992. [Online]. Available : https://aclanthology.org/D19-1410

APPENDIX

Figure 1. The word clouds for both classes reveal significant overlap in keywords, with frequent appearances of terms such as "Follow", "RT," "give," and "away," highlighting the prevalence of spam and retweets in the dataset prior to cleaning.

```python
def objective(trial):
    params = {
        'objective': 'binary:logistic',
        'eval_metric': 'auc',
        'max_depth': trial.suggest_int('max_depth', 3, 10),
        'gamma': trial.suggest_float('gamma', 0.1, 5.0),
        'reg_alpha': trial.suggest_float('reg_alpha', 1, 50),
        'reg_lambda': trial.suggest_float('reg_lambda', 0.5, 10.0),
        'colsample_bytree': trial.suggest_float('colsample_bytree', 0.6, 0.9),
        'min_child_weight': trial.suggest_int('min_child_weight', 5, 20),
        'learning_rate': trial.suggest_float('learning_rate', 0.01, 0.15),
        'subsample': trial.suggest_float('subsample', 0.6, 0.9),
        'scale_pos_weight': trial.suggest_float('scale_pos_weight', 0.8, 1.5),
        'seed': 42,
    }
```

Figure 2. The hyperparameters tuned during the Optuna optimization. The ranges were chosen to balance exploration and exploitation while optimizing the AUC.

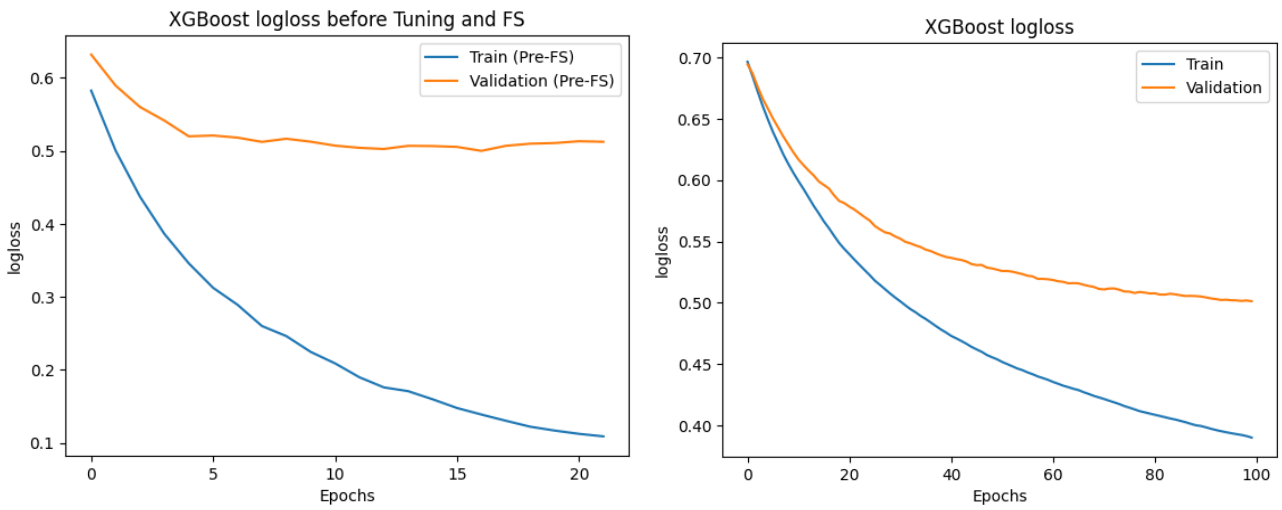

Figure 3. After tuning and feature selection, the validation logloss stabilizes and quite closely follows the training logloss, compared to the untuned model.

```
Best threshold : 0.53
Metrics with Selected Features:
Accuracy: 0.7819
ROC-AUC: 0.8433
Classification Report:
              precision    recall  f1-score   support

           0       0.81      0.72      0.76       156
           1       0.76      0.84      0.80       165
```

Figure 4. The table displays the performance metrics of the XGBoost model after applying feature selection. The model achieved an accuracy of 78.19% and a ROC-AUC score of 0.8433. The classification report indicates balanced performance across both classes, with precision, recall, and F1-scores of 0.76 to 0.81 for class 0 and 0.76 to 0.84 for class 1. The best threshold for classification was determined to be 0.53.