Stop Bullying Me

Cahier des charges

Projet SP21 - WebMining

Laurent Chassot
laurent.chassot@master.hes-so.ch

Romain Capocasale
romain.capocasale@master.hes-so.ch

Vincent Moulin
vincent.moulin@master.hes-so.ch

18 juin 2021

Prof.

Laura Elena Raileanu

Assistants

Antoine Rochat et Elena Najdenovska

Table des matières

1 Contexte et objectifs du projet

Le projet **SBM** se déroule dans le cadre du cours de WebMining du semestre de printemps 2021 au sein du MSE. Le but est de réaliser un projet mêlant les différentes techniques de récupération, traitement et analyse de données apprises durant le cours.

Nous avons décidé de réaliser un projet en lien avec Twitter. En effet, le but est d'entraîner un modèle de Machine Learning sur un jeu de données de tweets labélisés. Le jeu de données contient des tweets neutres, des tweets injurieux ou haineux.

Au travers d'une interface web Flask, les utilisateurs pourront se connecter avec leur compte Twitter. Ils auront alors une vue d'ensemble des commentaires reçus sur leur compte. Les commentaires seront alors analysés par notre modèle qui annotera les tweets afin de déterminer si ceux-ci sont potentiellement haineux.

Cet outil a pour objectif de permettre à un "influenceur", une marque ou encore un journal, de traiter et récolter les commentaires haineux postés sur leurs comptes. L'idée est de pouvoir permettre à la personne ou l'entité utilisant l'outil de bloquer certains utilisateurs ou encore de les dénoncer pénalement.

2 Données

Le dataset utilisé provient de l'adresse suivante : https://data.mendeley.com/datasets/jf4pzyvnpj/1#__sid=js0. Il contient des tweets labelisés selon la catégorie de harcèlement (racisme, sexuel, attaques...). L'ensemble contient un dataset "twitter_parsed_dataset.csv". Celui-ci contient plusieurs sources de harcèlement mais labelise globalement (harcèlement ou pas).

L'ensemble du dataset se trouve sous une licence **Creative Common**. Elle permet donc de tout faire avec les données, tant que la source est créditée.

Le dataset choisit ci-dessus contient 16'850 enregistrements de tweets (2MB). Pour l'étape de Machine Learning, nous pensons cela suffisant en terme de quantité de données. Cependant, nous réévaluerons ce choix lors de nos tests de Deep Learning (concaténation de plusieurs datasets du pack).

Les datasets sont structurés en : text, annotation, label :

- text : Le text brut du tweet (contenant hastags, annotations, ...)
- annotation : précise plus exactement le label : si sexiste, raciste, ...
- label : 1 si sexiste, 0 sinon.

Le jeu de données est en anglais. le modèle sera donc entraîné pour travailler uniquement des tweets en anglais. L'utilisateur final aura donc la possibilité de faire analyser uniquement les commentaires en anglais.

Il est important de noter que les tweets labelisés comme harcelant proviennent de base de données recansant les tweets sexistes et racistes. Les tweets harcelant à proprement parler ne sont donc pas recensés dans le jeu de données.

Voici un example de 5 tweets non harcelant :

```
 'I would love all the teams breakfast while camping #mkr',
 '@jaredchase runes on the ground, you have to step in them, thing surrounds you to absorb dmg, but it does dmg. gotta dps it s
lowly..',
 "There seems to be 2 ways to succeed in #mkr cook terribly, whinge excessively about not getting a fair go and give 1's or u k
now, cook well",
 "@JRehling Here's your guide to understanding the difference between a loner and a cult, microbrain. http://t.co/oY2iJKbhKl",
 '"I honestly do not understand the attacks on this man." - @ninaburleigh on @wadhwa.',
```

FIGURE 2 – Tweets non-harcelant

Et 5 tweets harcelant :

```
"Ahh Manu well said! Just cook good food and you'll be OK. Simples! Back in ya box bitch Kat #MKR",
'RT @BlissTabitha: Australia: Gifted teen embraces Islam, becomes a jihad terrorist http://t.co/bpmb7EoaGv',
'@SumbelinaZ @IronmanL1 @Hatewatch Sorry bitch, you did your research in your own asshole.',
"@roisin_morgan And I don't care if the Jews do what the have to do to protect themselves from the genocidal Muslims.",
'OH MY GOD,  SOMEONE PUNCH KAT IN THE FACE.  #mkr',
```

FIGURE 3 – Tweets harcelant

2.1 Répartition des données

Le graphique ci dessous montre que la classe "No-bulling" est majoritaire par rapport à la classe "Bullying". Il est donc important de prendre cette information en compte dans l'entrainement des modèles. En effet, un surajustement pourrait facilement se produire lors de l'entraînement.

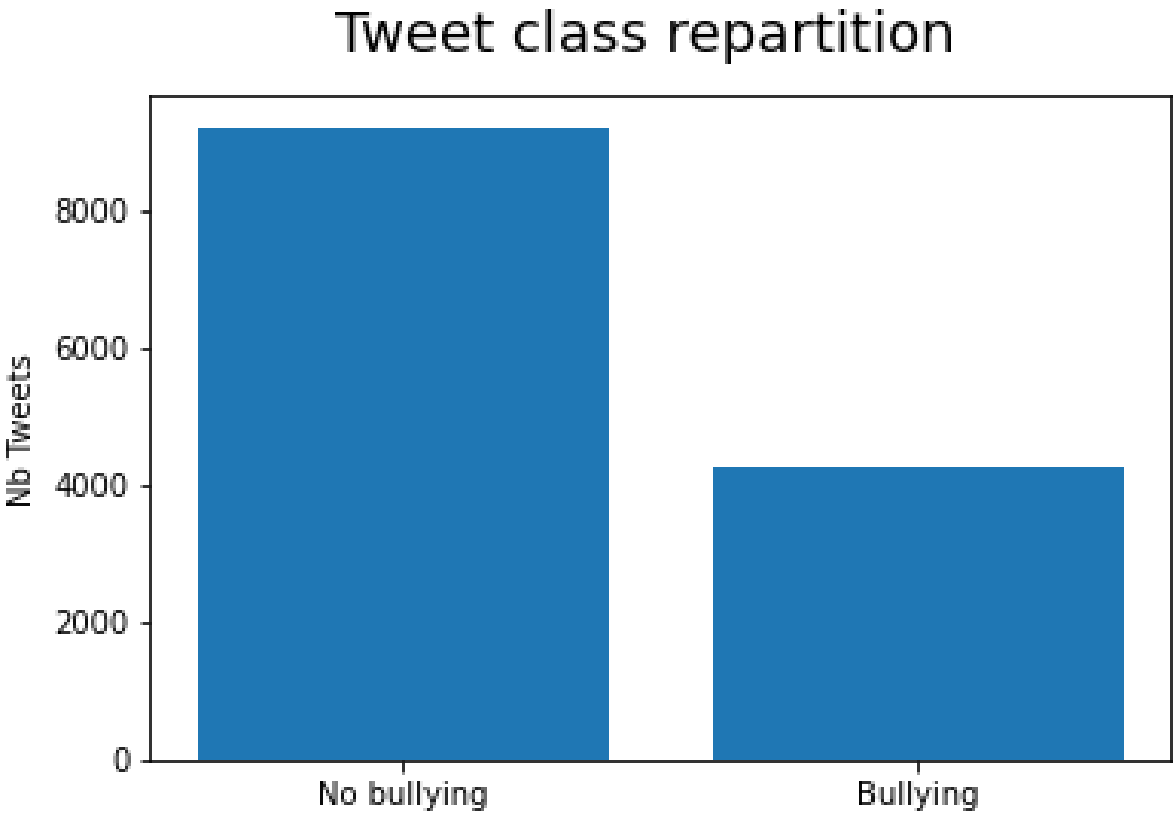

FIGURE 4 – Réparatition des classes

2.2 Nombre de mot par tweet

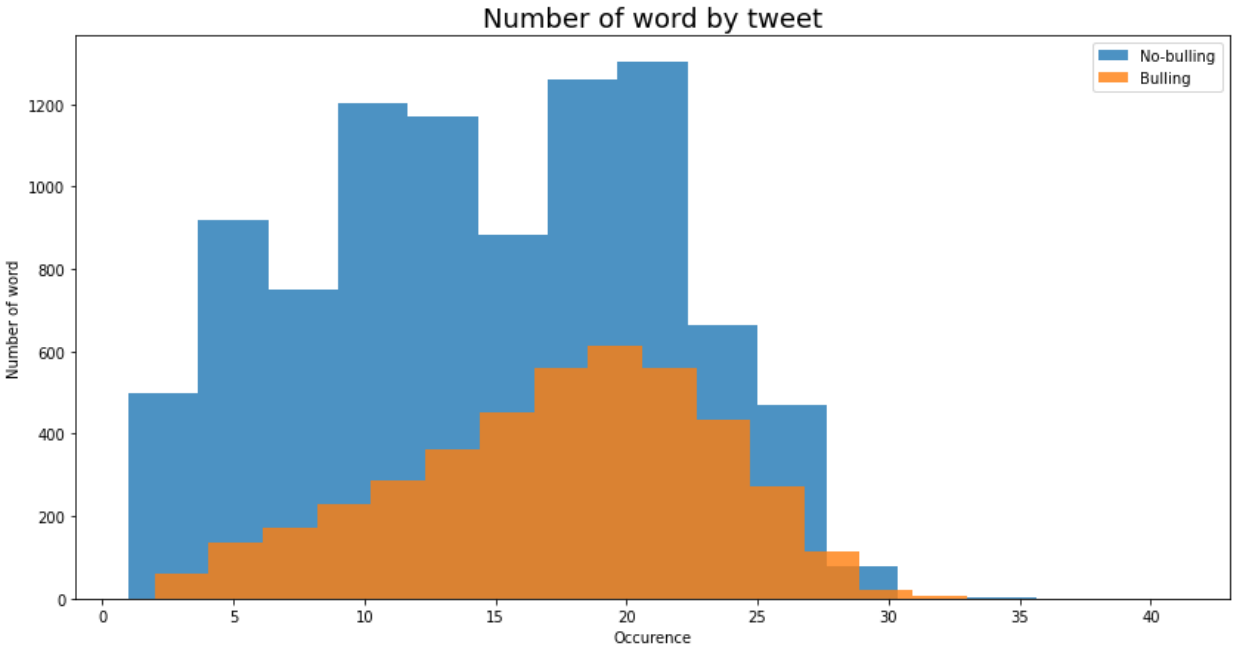

FIGURE 5 – Nombre de mot par tweet

On constate que le nombre moyen de mots par tweets est aux alentours de 20.

2.3 Preprocessing

Le prétraitement est implémenté via une classe nommée "TextProcessor" qui permet de prétraiter le texte selon différents paramètres :

- Suppression ou non des stops words
- Suppression de la ponctation ou non
- Taille minimum des mots à conserver
- Méthode de prétraitement des token Twitter (expliqué ci-dessous)
- Type de token Twitter à prendre en compte dans le prétraitement (expliqué ci-dessous)

2.3.1 Prétraitement des tokens Twitter

Les tweets sont un type de texte qui comporte différentes particularités comme :

— Des liens
— Des emojis
— Des hashtags
— Des mentions (@utilisateur)
— Des smileys

Il est donc important de traiter ces différents paramètres. Dans le cadre de ce projet, la libraire Python tweet-preprocessor est utilisée. Cette librairie permet soit de supprimer ces particularités ou de les prétraiter de la manière suivante. Un lien *https ://my.url.com/home* sera donc par exemple remplacé par URL et un hashtag *awesome* par $HASHTAG$. Ces tokens spéciaux vont pouvoir donner des informations suplémentaires aux modèles.

Dans la classe "TextProcessor", il est possible de préciser quelle opération on veut effectuer :
— Supprimer les tokens
— Prétraiter les tokens
— Ne rien faire
Il est également possible d'indiquer à la classe pour quel type de token (url, hashtag, ...) ces opérations s'appliquent.

L'idée est de tester différentes paramètres de cette classe "TextProcessor" pour maximiser les scores des modèles.

3 Etat de l'art

Actuellement, concernant le NLP, l'état de l'art s'intéresse aux "Transformers". En effet, il s'agit de l'évolution venant des RNN et LSTM. Un Transformer utlise le méchanisme d'attention. Celui-ci permet de définir contextuellement l'attention donnée à un mot. Cela permet par exemple de savoir à quoi se réfère un pronom dans une phrase. On peut citer quelques algorithmes dans la catégorie des Transformers.

— BERT
— Transformer-XL
— Compressive-Transformers

BERT est le moins récent des trois ci-dessus (2018). Beaucoup de sources existent déjà à son sujet. Il est sélectionné arbitrairement pour ce projet. Dans un but amélioratif, les autres modèles de Transformer pourraient être testés par la suite.

4 Conception

Cette section décrit les cas d'utilisation de l'application ainsi que le pipeline global suivi pour l'obtention des résultats.

4.1 Approches étudiées

Notre application se différentie des autres par le fait que les prédictions sur un tweet sont effectué avec 3 modèles et donc 3 approches différentes. Ces différentes apporches sont : les techniques de machines learning classique, les réseaux de neuronne récurrent et BERT. Ces méthodes peuvent alors être comparées les unes avec les autres autant en ce qui concerne la phase d'entrainement que lors de la prédiction sur un tweet donné.

4.2 Cas d'utilisation

Le diagramme 6 montre les différents cas d'utilisation de l'application pour un utilisateur.

Une fois connecté, il pourra parmis son flux de tweets en sélectionner certains afin de les analyser. Un score de harcèlement leur sera attribué. Des statistiques sur le nombre de tweets analysés et le nombre harcelant par exemple sont également observables.

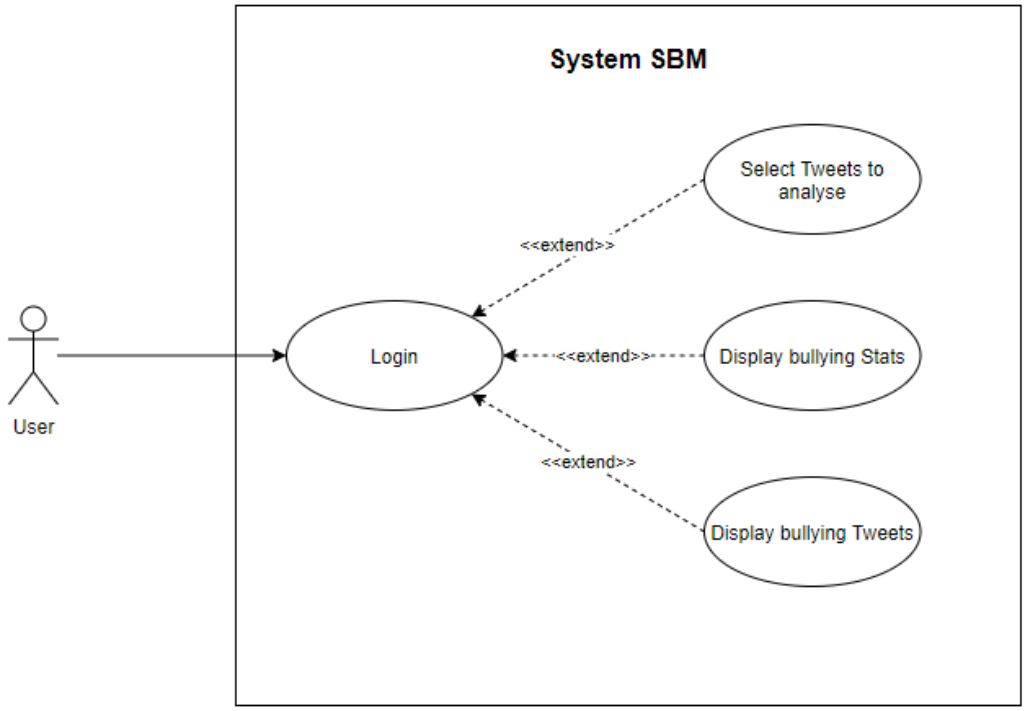

FIGURE 6 – Use Cases

4.3 Pipeline machine learning

L'objectif est de trouver le modèle donnant les meilleures performances pour prédire la classe "Bullying" ou "No-bulling" pour un tweet donné. La capture ci-dessous présente la pipeline proposée pour ce projet :

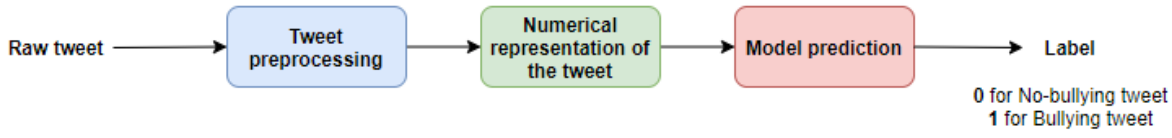

FIGURE 7 – Pipeline de machine learning

4.4 Architecture et technologies envisagées

La figure 15 présente l'architecture imaginée pour le projet.

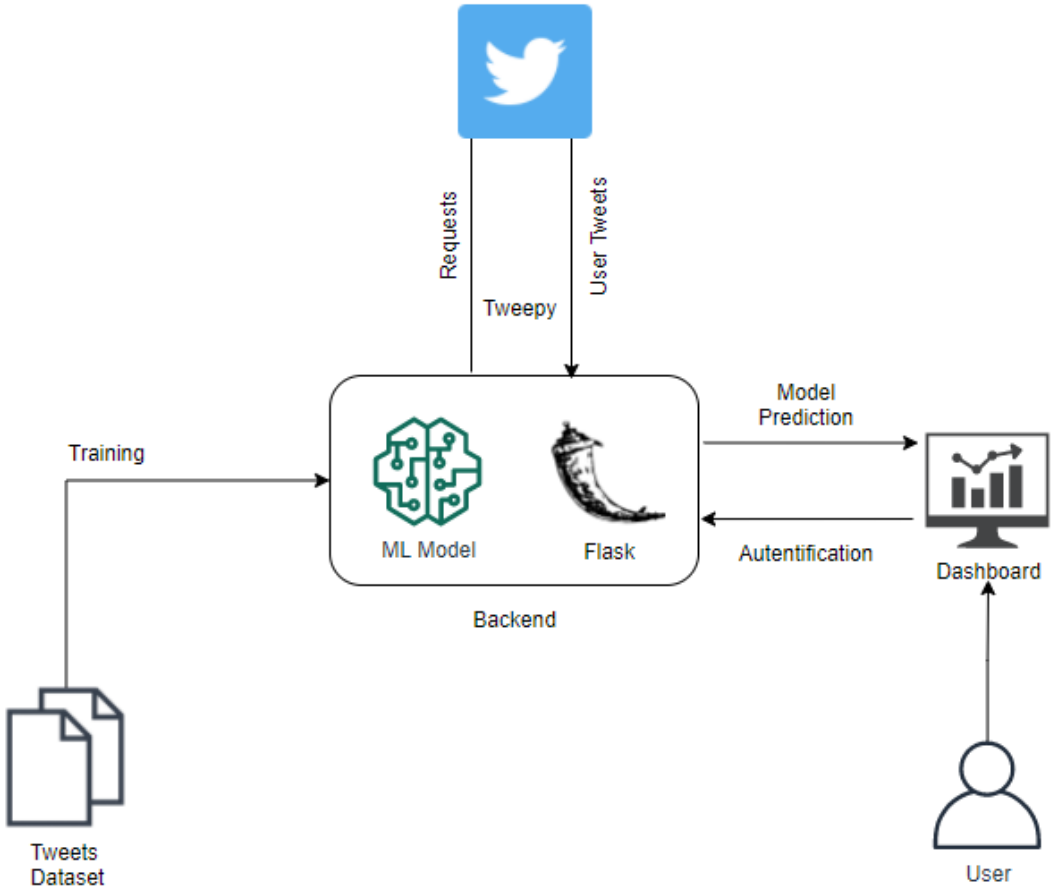

FIGURE 8 – Diagramme de l'architecture

5 Fonctionnalités

Cette section montre les fonctionnalités en fonction du système (frontend, backend).

5.1 Frontend

Le frontend permet d'intéragir avec le backend via une interface graphique. L'interface est réalisée en JavaScript sans l'aide de framework.

Premièrerement, il est demandé de se connecter en cliquant sur le lien ci-dessous :

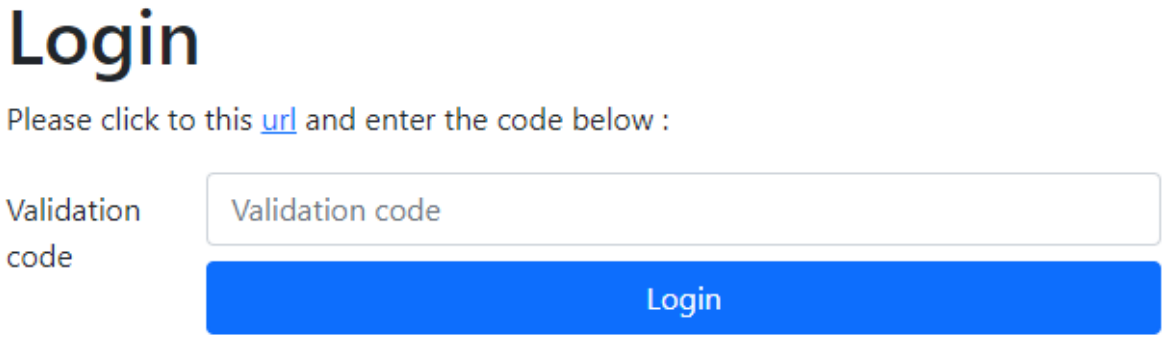

FIGURE 9 – Autentification

L'utilisateur est ensuite redirigé sur la page d'autentification de Twitter où il peut récupérer son code de validation à rentrer dans le champs présent sur la capture précédente :

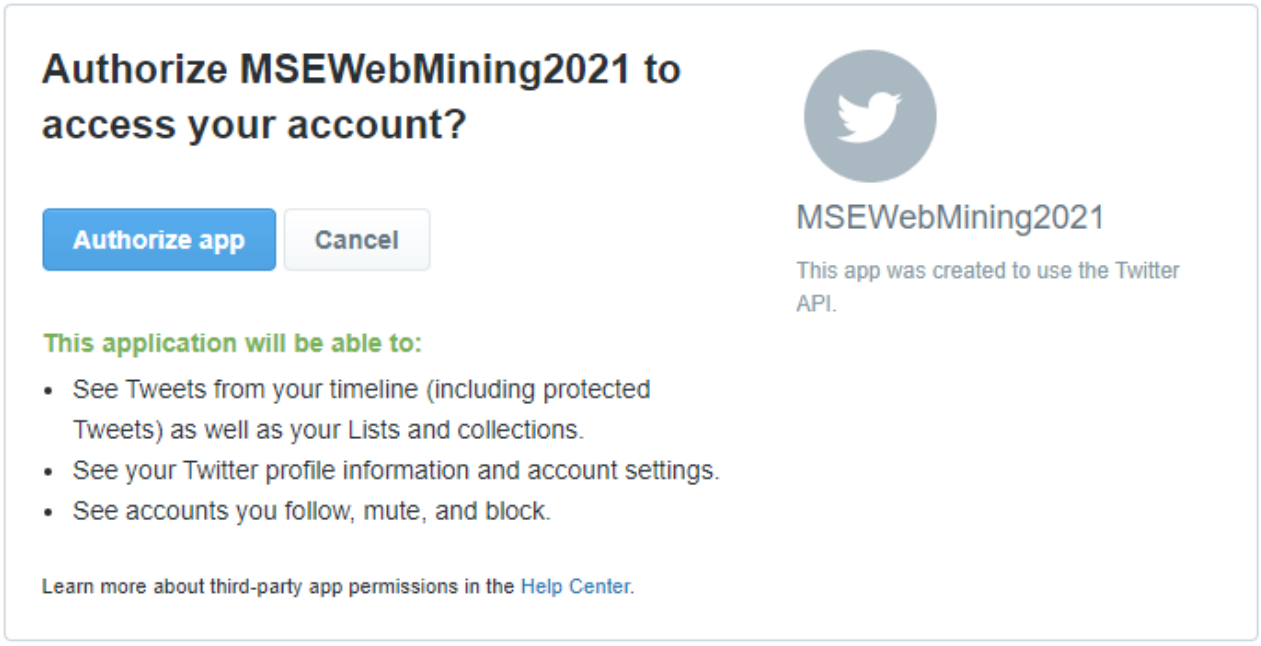

FIGURE 10 – Récupération code de validation

Une fois autentifié, le nom de l'utilisateur est indiqué et ses différents tweets sont affichés :

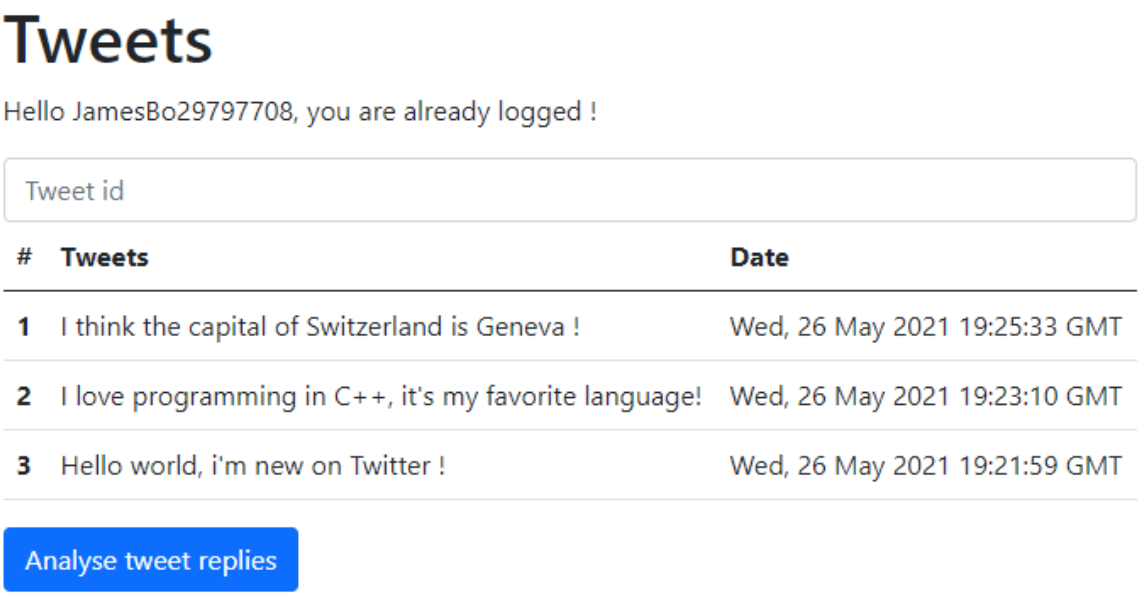

FIGURE 11 – Visualisation des tweets

L'utilisateur peut ensuite selectionner un de ces tweets et cliquer sur le bouton "Analyse tweets replies". Il est également possible de rentrer dans le champs en-dessus du tableau l'id d'un tweet de n'importe quel compte pour y analyser les réponses. Cette option a été insérée pour permettre aux utilisateurs ayant peu de réponses sur leurs tweets de quand même tester l'application.

La vue ci-dessous présente les réponses du tweet de l'utilisateur avec les différentes prédictions :

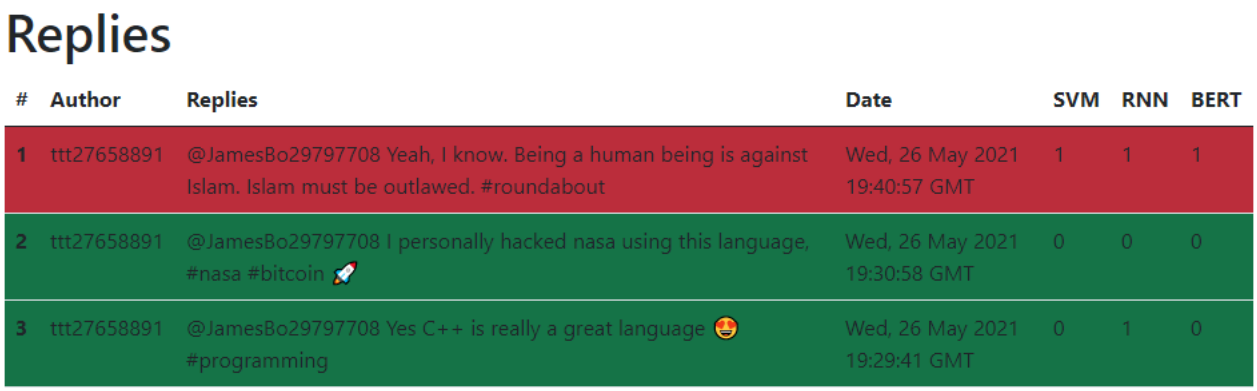

FIGURE 12 – Réponses des tweets

Les prédictions sont effectuées pour nos 3 modèles de machine learning developpés. Le label 0 indique que le tweet est considéré comme non harcelant et 1 comme harcelant. Pour chaque ligne le label dominant est retenu et la ligne est colorée en vert pour un tweet non harcelant et rouge pour harcelant.

5.2 Backend

Le backend a été developpé avec Flask en python. Ce framework a été choisi car les différents modèle de machine learning ont été developpés avec des librairies en Python.

L'intégration des modèle et du backend est alors simplifiée. Pour intéragir avec l'API de Twitter depuis le backend, la librairie Python Tweepy est utilisée. Le backend dispose des fonctionnalités suivantes :

Route	Autentification	Explications
/auth	non	Obtention de l'url permettant à l'utilisateur de s'autentifier
/set_verifier	non	Definition du token de connexion par l'utilisateur
/username	non	Récupération de l'utilisateur autentifié
/user_tweets	oui	Récupération des tweets de l'utilisateurs autentifié
/retrive_bullying_replies	oui	Récupération des réponses d'un tweet de l'utilisateur et effectue les prédictions

6 Techniques, algorithmes et outils utilisés

Cette section explique les différents algorithmes utilisés afin d'arriver à nos résultats attendu et comment ceux-ci sont configurés dans ce but.

6.1 Métriques

La métrique utilisée pour évaluer les modèles est le F1-Score. Cette métrique a été selectionnée car elle maximise le compromis entre la précision et le rappel. Elle donne également plus d'importance aux classes sous-représentées.

$$F1Score = \frac{2 * Precision * Recall}{Precision + Recall} \tag{1}$$

6.2 Méthodes de machine learning classique

Dans cette première phase, le but est d'explorer des algortihmes classiques de machines learning. Deux représentations des phrases ont été exploré dans cette phase :

- Tf-Idf
- Word2Vec

Les algorithmes de machine learning suivants sont utilisés :

- MultinomialNB
- Random Forest
- SVM

Les modèles suivants ont été entrainés avec les paramètres de prétraitement de texte suivants :

- Suppression de la ponctuation
- Suppression des stop words
- Taille minimum des mots de 2 carcatères
- Prétraitement des tokens spéciaux (tout types de tokens confondus)

Les différents modèles ont été entrainés à l'aide de la classe "RandomizedSearchCV" de Scikit-learn qui permet de chercher les meilleurs hyperparamètres pour un modèle tout en effectuant une validation croisée. Dans notre cas, une validation croisée de 5 fold est utilisée.

Le tableau suivant présente les scores obtenus pour les différentes expériences :

Modèle	**Représentation des phrases**	**F1-Score**
MultinomialNB	Tf-Idf	69%
Random Forest	Tf-Idf	84%
SVM	Tf-Idf	**85%**
Random Forest	Word2Vec	78%
SVM	Word2Vec	77%

On constate que le meilleur score obtenu pour ces différentes expériences est le SVM avec le Tf-Idf. Ce modèle sera donc utilisé pour effectuer les prédiction en production.

Il à noter qu'avec ces différentes techniques il est difficile de savoir si les modèlent surajuste ou non. Les résultats sont donc à prendre entre paranthèse.

6.3 Deep learning

Dans cette phase, différents types de réseaux de neuronnes ont été explorés, notamment :
- SimpleRNN
- LSTM
- GRU

Les mêmes paramètres de prétraitement du texte que l'étape précedente ont été utilisés.

Deux types d'embeddings ont été utilisés pour cette expérience :
- Embeddings sans poids
- Embeddings avec ceux de GloVe

GloVe est une représentaion vectorielles des mots entrainés de manière non-supervisé sur un grand corpus de documents. La matrice de GloVe peut être fournie en paramètre à la couche Embedding de Keras dans la définition du modèle. La matrice de GloVe est disponible en plusieurs dimensions. Dans le cadre de ce projet, une matrice de dimensions **100** est utilisée.

Les différents modèles ont été entrainés à partir d'une validation croisée sur 5 folds. Une fois l'hypothèse vérifiée, les modèles sont entrainés sur toutes les données pour maximiser les performances.

Les scores obtenus sur le jeu de test pour les différentes expériences sont les suivantes :

Modèles	**Embeddings pré-entrainés**	**Overfitting**	**F1-Score**
SimpleRNN	Non	Oui	79%
LSTM	Non	Oui	82%
GRU	Non	Oui	81%
SimpleRNN	Oui	Non	78%
LSTM	Oui	Non	82%
GRU	Oui	Non	**83%**

En comparant les graphiques de la loss et de l'accuracy, on constate que le réseau GRU sans GloVe surajuste beaucoup plus que le réseau avec GloVe.

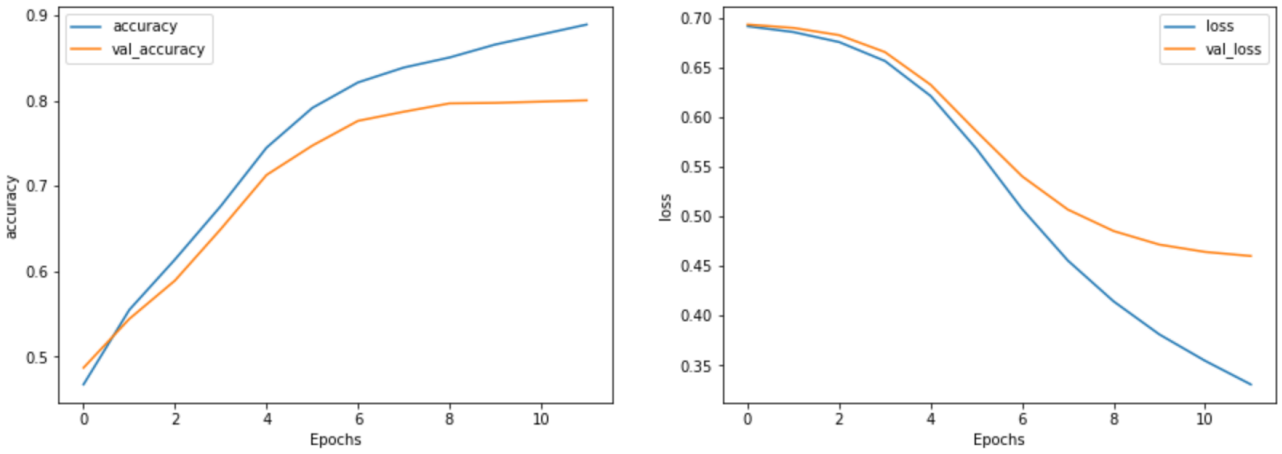

FIGURE 13 – GRU sans GloVe - surajustement

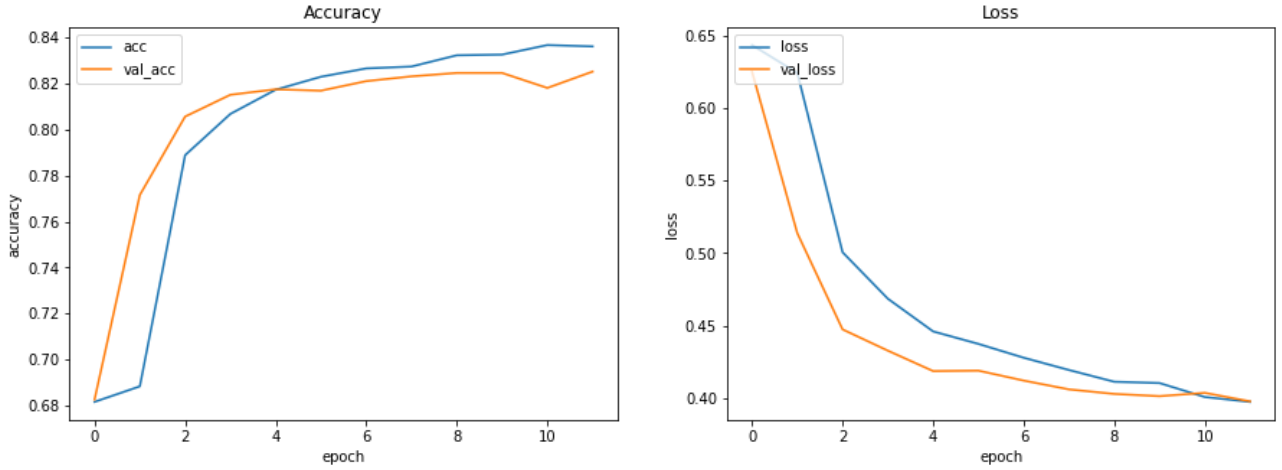

FIGURE 14 – GRU avec GloVe - pas de surajustement

Au vu des scores obtenus, le réseau GRU avec les embeddings GloVe est selectionné. L'architecture de ce réseau est la suivante :

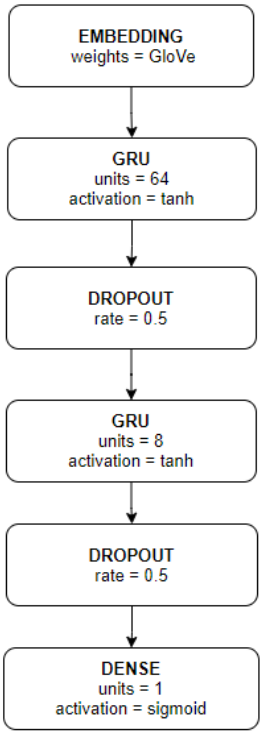

FIGURE 15 – Réseau

Les hyperparamètres suivants ont été utilisés :

- **Batch size :** 128
- **Epochs :** 10
- **Optimisateur :** Adam
- **Learning rate :** 0.001

6.4 BERT

BERT est un algorithme de l'état de l'art de la catégorie des "Transformers", proposé par HuggingFace. Aucun preprocessing supprimant des mots ou autre n'est fait. En effet, Bert a besoin des stop words pour comprendre le contexte du texte.

Pour la manipulation du modèle, un objet "Trainer" est utilisé. Celui-ci permet d'entrainer et d'évaluer le modèle facilement. Il prend en argument :

- Le modèle de base, préconfiguré : Ici, un AutoModelForSequenceClassification permet de l'obtenir
- Le tokenizer : Permet de vectoriser les phrases du dataset

— Le dataset d'entrainement : Objet Dataset, permettant de récupérer directement des tensors
— Le dataset d'évaluation : Egalement un objet Dataset
— Les hyperparamètres
— Les métriques à calculer : le F1, accuracy, precision, recall

Concernant les hyperparamètres donnés, il s'agit de :

— 10 époques
— un learning rate à 2e-5
— Une taille de batch de 16

Pour simplifier la création des objets Dataset, un objet BertDataModule est créé et contient les deux datasets. On passe ensuite ce module à notre classe SMBBertClassifier qui entraîne le modèle avec le Trainer expliqué ci-dessus.

Finalement, le modèle nous donne les résultats suivants (figure 16) pour notre dataset de test. Ceux-ci ne sont pas très bon, étonnement. Aucun moyen intuitif n'est trouvé pour améliorer ceci.

Epoch	Training Loss	Validation Loss	Accuracy	F1	Precision	Recall
1	0.231800	0.234172	0.682668	0.553925	0.466036	0.682668
2	0.135200	0.218098	0.682668	0.553925	0.466036	0.682668
3	0.104900	0.238066	0.682668	0.553925	0.466036	0.682668
4	0.087300	0.202137	0.682668	0.553925	0.466036	0.682668
5	0.074800	0.199315	0.682668	0.553925	0.466036	0.682668

FIGURE 16 – Résultats de notre modèle Bert

7 Planification, organisation et suivi répartition du travail

Cette section montre la répartition du travail pour le projet.

7.1 Work packages

La figure 17 montre les "Work Packages" pour notre application. Ils décrivent le découpage des activités et montrent les phases de ce projet.

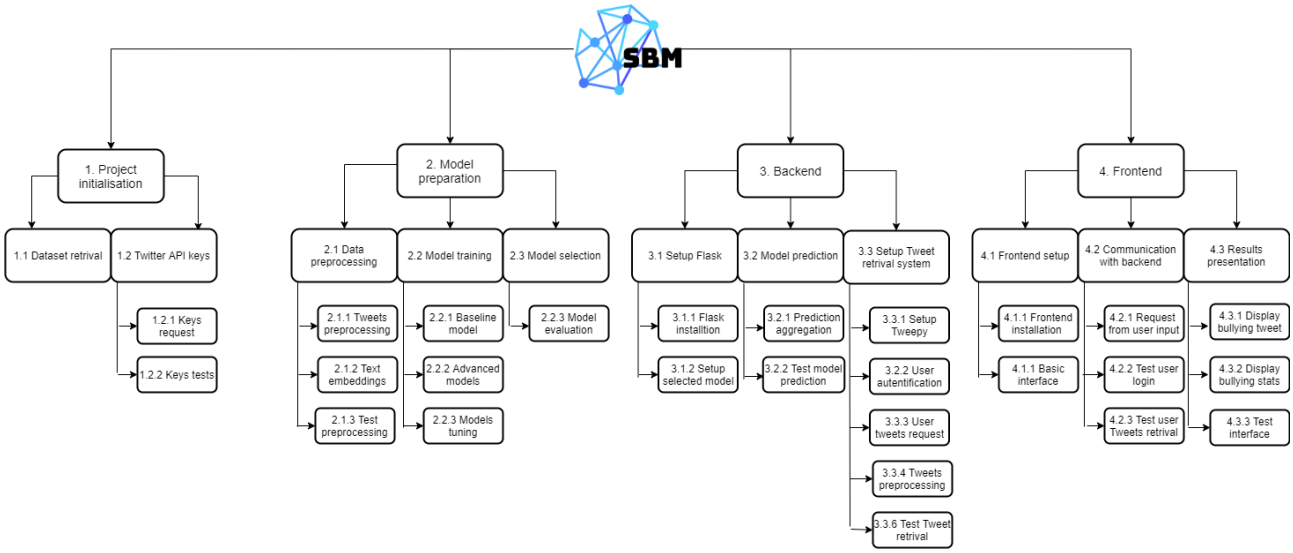

FIGURE 17 – Work packages

7.2 Gantt

La figure 18 présente le planning du projet.

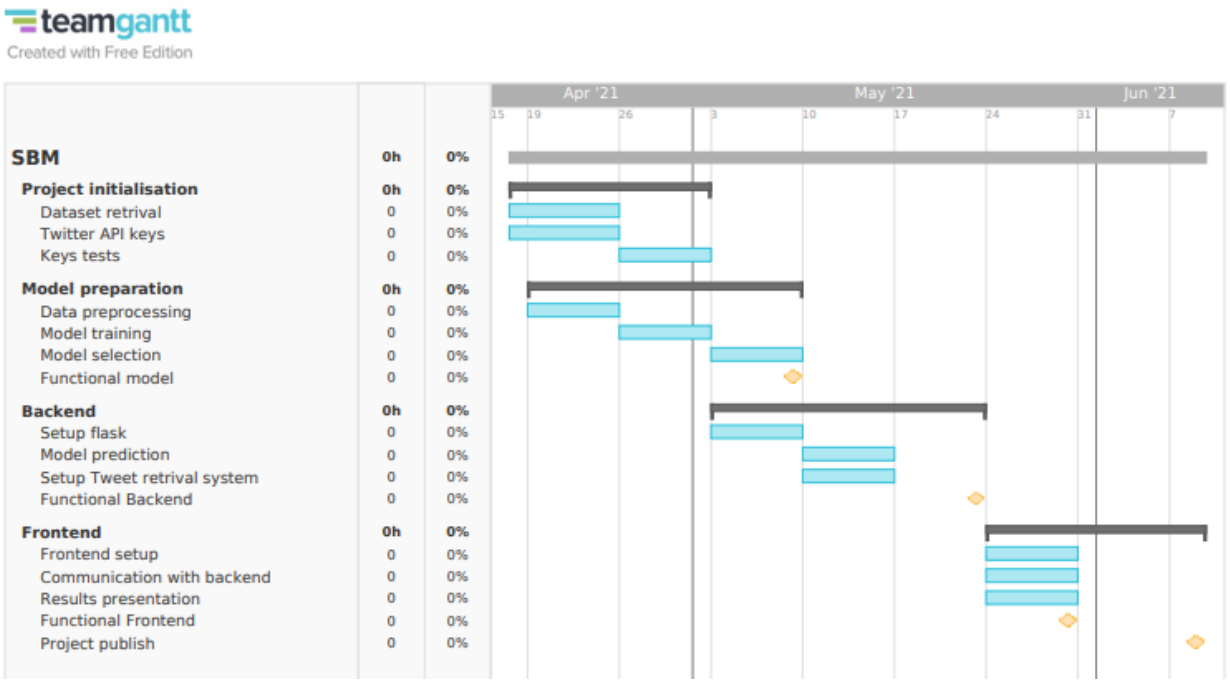

FIGURE 18 – Diagramme de Gantt

7.3 Répartition du travail

- **Machine learning**
  - **Machine learning classique :** Romain
  - **Deep learning :** Vincent
  - **BERT :** Laurent
- **Frontend :** Romain
- **Backend :** Romain

8 Conclusion / Travail futur

Pour conclure, nous pouvons dire que l'application dans son ensemble est fonctionelle. Le backend et le frontend permettent d'effectuer les différentes actions définies dans le cahier des charges. Cela concerne : l'autentifcation d'un utilisateur, la récupération de ces tweets ainsi que la prédiction de contenu harcelant ou non dans les commentaires de ce tweet.

En ce qui concerne la partie machine learning, différentes techniques ont été exploitées autant concernant le type de modèle (machine learning traditionnel, réseau de neuronnes et transfor-

mers), que le préproccesing du texte ainsi que la représentation vectorielle des phrases dans l'espace (tfidf, word2vec, GloVe, ...).

Cependant, on constate que les modèles obtiennent de bons résultats lors de l'entrainement / test. Malheureusement, en phase de production, certaines prédictions obtenues sont assez décevantes. Les modèles ont tendance à trop se concentrent sur certains mot, notamment le mot "girl" qui est souvent représenté dans les tweets et classifié comme "sexiste". Ceci implique que les modèles associent ce mot avec la classe "bullying". De plus, lorsque l'on inspecte le jeu de données en profondeur, on constate que certains tweets sont labelisés comme sexiste ou raciste, alors que sortie du context, ces tweets ne devraient pas forcément être labelisés comme tel. On constate également que le jeu de données utilisé pendant l'entraînement n'est globalement pas très représentatif des vrais tweets présents sur l'application. Le peu de données à dispostion ne permettent pas de couvrir l'ensemble des possibilités de tweets harcelants ou non qui pourraient être écrits sur Twitter.

Une première amélioration serait de trouver un nouveau jeu de données de plus grande taille, mieux labelisé et qui comprend plus de classes que "sexiste" et "raciste". L'architecture du projet est très modulable et un nouveau jeu de données pourrait être utilisé très facilement pour ré-entrainer les modèles et les déployer en production sur le backend.

En ce qui concerne BERT, il serait intéressant de creuser plus en profondeur les résultats obtenus. D'autres architectures de transformers, notamment celles cités dans le chapitre état de l'art pourraient être explorés.

Une autre expérience à réaliser pourrait être, sachant que dans certains cas les commentaires du tweet sont sortis de leur context, de développer un modèle qui aurait deux entrées. L'une serait le tweet principal et la deuxième, la réponse à ce tweet. Le modèle disposerait d'informations contextuelles pour effectuer une prédiction de meilleure qualité.

9 Packages python utilisés

Les différentes librairies utilisées se trouvent à la racine du projet dans le fichier requirements.txt

10 Sources

— https://stackabuse.com/python-for-nlp-movie-sentiment-analysis-using-deep-learning-in-keras
— https://www.analyticsvidhya.com/blog/2020/04/how-to-deploy-machine-learning-model-flask/
— https://www.kaggle.com/nitin194/twitter-sentiment-analysis-word2vec-doc2vec

— https://towardsdatascience.com/predicting-tweet-sentiment-with-word2vec-embeddings-67aace9b019d
— Etat de l'art : https://zappy.ai/ai-blogs/the-current-state-of-the-art-in-natural-language-processing-nlp
— Bert main source : https://towardsdatascience.com/fine-tuning-pretrained-nlp-models-with-huggingfaces-trainer-6326a4456e7b
— https://pypi.org/project/tweet-preprocessor/
— https://dev.to/paulkarikari/deep-learning-lstm-for-sentiment-analysis-in-tensorflow-with-keras-api-b7
— https://www.tensorflow.org/text/tutorials/text$_c lassification_r nn$