

RAPPORT TECHNIQUE

Projet technique



Erwan Tinen-Toulac
Adrien Gimberg
Lisa Giordani
Oscar Jozon

Tuteur : M. Lefrançois

2020



SOMMAIRE

Remerciements.....	p. 3
Introduction.....	p. 3
I) Recherche documentaire et analyse du sujet.....	p. 4
1) Etude d'articles scientifiques	
2) Le cyberharcèlement et la loi	
3) Définition des critères du cyber-harcèlement	
4) Précision du cadre du sujet	
II) Gestion de l'équipe et du travail.....	p. 8
1) Pistes de travail envisagées	
2) Détermination des tâches à effectuer	
3) Planning	
4) Répartition des tâches	
III) Algorithmes de détection du cyber-harcèlement.....	p. 12
1) Algorithme naïf	
2) Algorithme de deep learning	
3) Statistiques	
IV) Intégration des algorithmes dans un système de détection du cyber-harcèlement sur Twitter.....	p. 16
1) API Twitter	
2) Base de données MySQL	
3) Serveur web local et interface utilisateur	
4) Mise en relation des acteurs	
V) Résultats et validation du système.....	p. 22
1) Résultats et validation de l'algorithme naïf	
2) Résultats et validation de l'algorithme de deep learning	
3) Comparaison des modèles et benchmarking	
Conclusion.....	p. 26
Bibliographie.....	p. 27
Annexes.....	p. 28

Remerciements

A M. Lefrançois, enseignant chercheur à l'école des Mines de Saint-Etienne, pour avoir accepté de suivre notre projet en tant que tuteur et pour nous avoir guidé lors de nos réflexions. –

A Mme Villata et Mme Cabrio, chercheuses au CNRS, pour nous avoir transmis l'article scientifiques sur leur projet de détection d'harcèlement sur les réseaux sociaux CREEP, et pour nous avoir accordé un interview afin de répondre à nos multiples interrogations sur le sujet.

Introduction

Selon un sondage IFOP réalisé pour France Info sur un échantillon de 1003 personnes, 22% des français entre 18 et 24 ans déclarent avoir été victime de cyberharcèlement au cours de leur vie. Cette pratique consiste à porter atteinte à une personne via les réseaux sociaux de manière répétée ou concertée si elle est le fait de plusieurs personnes.

Ainsi, comme beaucoup d'autres réseaux sociaux, Twitter est le siège de nombreux cas de cyberharcèlement. Les messages échangés sur cette plateforme prennent la forme de courts textes de 280 caractères, et sont archivés dans une base de données actualisée en temps réel. Nous avons donc décidé de nous intéresser au cyberharcèlement ayant lieu sur la plateforme Twitter.

Le projet a donc pour but de concevoir un algorithme permettant d'identifier et d'enregistrer les cas de cyberharcèlement ayant lieu sur la plateforme Twitter.

L'approche choisie consiste en :

- la détermination les critères représentatifs du cyberharcèlement (fréquence, durée, contenu)
- l'identification des cas de cyberharcèlement en utilisant les critères précédemment déterminés ainsi que du machine learning pour déterminer le caractère répréhensible d'un message
- la récupération en premier lieu les données mises à disposition gratuitement par Twitter, suivi d'un test de l'algorithme obtenu
- la mise à disposition de ces données via site web pour les aider à prouver les cas d'harcèlement et ainsi y mettre fin

I) Recherche documentaire et analyse du sujet

1) Sources documentaires utilisées

Lors de nos recherches, nous avons pu réaliser un interview de Serena Villata, une chercheuse au CNRS ayant travaillé sur un projet de détection du cyber-harcèlement (le projet CREEP). Nous avons pu alors avoir un avis technique sur la possibilité de réaliser le projet, ainsi qu'une direction à prendre pour assurer le fonctionnement de notre code. Suite à l'interview, Mme. Villata nous a transmis son article scientifique, qui nous servira par la suite (ainsi que d'autres articles trouvés sur *IEEE* et *Technique de l'Ingénieur*) de modèle pour nos benchmarks.

Durant notre travail, notre tuteur M. Lefrançois nous a donné les pistes nécessaires afin de donner une direction à notre étude. Il nous a également donné des conseils concernant la gestion du travail en équipe.

Enfin, le site du gouvernement nous a permis de déterminer les éléments pertinents que notre algorithme doit être capable de relever. La lecture des articles du code pénal concernant le cyber-harcèlement a mené vers une ouverture d'esprit, qui nous a aidé à centrer le problème social et juridique à résoudre par la technique.

2) Le cyberharcèlement et la loi française

Le but premier de l'algorithme est d'aider les personnes victime de cyber-harcèlement sur Twitter à récupérer facilement des preuves de la situation dont ils sont victimes. Ces preuves permettent par la suite de supporter un signalement d'abus auprès de Twitter, ou encore une action juridique à l'encontre des individus malveillants.

Pour cela, il est donc important que l'algorithme se base sur des textes de loi et sur la jurisprudence afin que les données récoltées soient les plus pertinentes possible. Dans le cadre de notre étude, nous adapterons l'algorithme aux lois et au système juridique français.

Depuis août 2014, le cyberharcèlement est considéré comme un délit passible d'amende et/ou d'une peine d'emprisonnement inférieure à 10 ans. L'article 222-33-2-2 du code pénal créé par la loi 2014-873 stipule alors :

« Le fait de harceler une personne par des propos ou comportements répétés ayant pour objet ou pour effet une dégradation de ses conditions de vie se traduisant par une altération de sa santé physique ou mentale est puni d'un an d'emprisonnement et de 15 000 € d'amende lorsque ces faits ont causé une incapacité totale de travail inférieure ou égale à huit jours ou n'ont entraîné aucune incapacité de travail. »

Le code pénal nous permet alors d'identifier les éléments que notre algorithme doit mettre en valeur afin de pouvoir être considéré dans le cadre d'une plainte ou d'une action judiciaire. En effet, notre programme doit être capable de mettre en avant **l'existence d'actes répétés** visant à intimider ou insulter la victime. Dans notre cas, il s'agirait d'étudier le nombre de messages Twitter malveillants adressés à l'utilisateur. Cependant, les insultes à répétition mises en valeur par l'algorithme ne se limitent pas à une seule personne. Il est fréquent qu'une victime de cyber-harcèlement subisse des brimades de plusieurs agresseurs, formant un groupe. Ce concept dit de « harcèlement par raid numérique », introduit par la loi du 3 août 2018 contre les violences sexuelles et sexistes (loi Schiappa), permet alors de montrer la culpabilité d'une personne n'ayant publié qu'un seul tweet injurieux, mais faisant parti d'un groupe de harcèlement en meute.

L'enjeu de l'algorithme ne se limite pas alors aux messages insultants proliférés par un individu seul, mais inclut aussi le **repérage des nœuds de harcèlements** : les groupes d'utilisateurs organisés publient en masse des messages injurieux envers une personne. En mettant en relation plusieurs messages et en prouvant la présence d'une action de groupe, l'algorithme serait capable de fournir des preuves utiles pour appuyer une plainte.

En ce qui est de l'échelle de gravité du harcèlement, suite à la loi Schiappa, le cyberharcèlement est puni de deux ans d'emprisonnement et de 30 000 € d'amende :

1. Lorsqu'ils ont causé une incapacité totale de travail supérieure à huit jours ;
2. Lorsqu'ils ont été commis sur un mineur de quinze ans ;
3. Lorsqu'ils ont été commis sur une personne dont la particulière vulnérabilité, due à son âge, à une maladie, à une infirmité, à une déficience physique ou psychique ou à un état de grossesse, est apparente ou connue de leur auteur ;
4. Lorsqu'ils ont été commis par l'utilisation d'un **service de communication au public en ligne**.

L'amende s'élève à 45 000 € si au moins deux critères sont remplis.

Nous pouvons observer que le type de harcèlement (sexisme, homophobie...), les précédents messages de la victime ou les informations sur l'individu perpétrant le délit n'a aucune influence sur la sentence. **Seules la plateforme sur laquelle le cyber-harcèlement a lieu (service de communication au public en ligne), les conséquences sur la victime et son âge contribue à l'aggravation de la peine**. Ces informations faisant partie de la vie privée de la victime, notre algorithme n'en tiendra pas compte.

Le programme ne créera alors aucune forme d'échelle jugeant l'importance de certaine forme de cyberharcèlement. Le rôle de l'algorithme se limitera à la détection et à la mise en relation des divers messages insultants adressés au sujet. Pour assurer l'objectivité et la pertinence des données récoltées, il sera alors envisageable de renvoyer l'intégralité des messages insultants visant l'utilisateur sous forme de liste, accompagné de statistiques précisant la fréquence et le nombre de messages insultants en fonction du temps ainsi que le nombre d'utilisateurs proliférant ces messages.

Pour cela, il est nécessaire de définir précisément les éléments que recherche notre algorithme afin de détecter les cas de cyberharcèlement au sein des tweets.

3) Définition des critères du cyberharcèlement

Les informations rassemblées précédemment nous permettent de déterminer les critères du cyber-harcèlement sur les réseaux sociaux. En effet, un individu est considéré comme un cyber-harceleur **s'il effectue de manière répétée des posts offensants ou insultants à destination d'un autre utilisateur.**

Ces critères nous permettront par la suite de déterminer les données que nos algorithmes doivent détecter et afficher.

4) Précision du cadre du sujet

Afin de mieux cerner le cadre de notre projet, nous avons conçu un diagramme de cas d'utilisation.

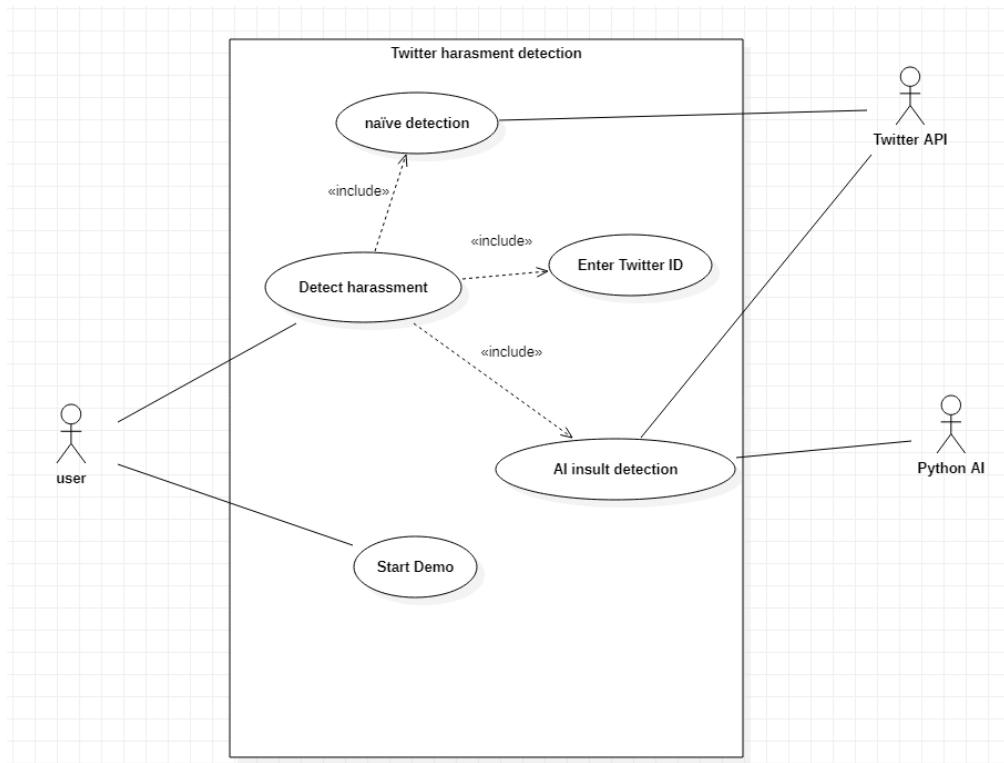


Diagramme des cas d'utilisation

On considérera deux algorithmes. Un algorithme naïf qui utilisera une méthode de détection d'insulte basique (mot par mot) et un algorithmique qui utilisera une intelligence artificielle codée en python. L'intelligence artificielle aura déjà été entraînée pour détecter les messages insultants, ainsi le système donnera le contenu du tweet à l'IA qui déterminera si le tweet est insultant ou non.

Nous avons également déterminé les exigences fonctionnelles et non-fonctionnelles que notre système devra satisfaire.

Exigences fonctionnelles :

1. L'utilisateur doit pouvoir choisir la méthode de détection des tweets insultants (IA ou algorithme naïf).
2. L'utilisateur doit pouvoir utiliser l'algorithme naïf
3. L'utilisateur doit pouvoir utiliser l'algorithme de deep learning
4. L'utilisateur doit pouvoir récupérer les tweets insultants adressés à un compte Twitter
5. L'utilisateur doit pouvoir s'informer au sujet du cyber-harcèlement

Exigences non fonctionnelles :

- Utilisabilité : l'interface utilisateur doit être ergonomique et facile d'utilisation
- Efficacité : le temps de calcul des algorithmes doit être raisonnable
- Confidentialité : le système ne doit pas stocker les tweets insultants détectés par les algorithmes et les statistiques calculées à partir de ces résultats

II) Gestion de l'équipe et du travail

1) Pistes de travail envisagées

À la suite d'un premier rendez-vous avec notre tuteur de projet, nous avons envisagé plusieurs approches afin d'établir une ligne directrice pour notre travail.

Choix de la plateforme :

Nous avions premièrement suggéré de rendre l'algorithme compatible avec tous les réseaux sociaux. En effet, tout comme les sites tels que Instagram ou TikTok sont sujets à de nombreux cas de cyberharcèlement.

Cependant, le traitement de données s'avérait difficile pour ces deux réseaux. En effet, deux problèmes majeurs se sont posés. Le premier concerne le type de contenu : les échanges publics sur Instagram sont majoritairement des photos, et sur TikTok, exclusivement par vidéo, rendant le décryptage des données difficile.

Le second problème provient de l'accès aux données. Instagram et TikTok ne permettent pas au public d'avoir accès à l'API (livrant les données du réseau en temps réel). Il était alors impossible en tant qu'étudiant d'implémenter l'algorithme sur le réseau.

Nous avons alors restreint notre projet à la recherche de cas de cyberharcèlement sur Twitter, réseau majoritairement textuel avec des posts courts (280 caractères) donnant accès aux données en temps réel via API.

Choix du langage de programmation :

Pour la conception du système, nous avons choisi d'utiliser **Symfony**, qui est un framework MVC écrit en PHP. Donc, nous utiliserons le **langage PHP** pour implémenter l'algorithme naïf

L'algorithme utilisant des bases de données, le **langage SQL** a été choisi pour récupérer des données de l'API Twitter.

La partie « machine learning » de l'algorithme sera écrite en **python** grâce à la bibliothèque Tensorflow, une bibliothèque pour le deep learning (et l'apprentissage machine en général) open-source gratuite développée par Google. Un **code Python** (rendu compatible avec le code PHP) permettra l'analyse de la base de données de Twitter par IA depuis un script PHP.

L'interface utilisateur, sous forme de page web, sera écrite en **HTML** et **CSS**.

Types de données traitées :

Twitter propose de communiquer par plusieurs types de média : texte, emojis, images statiques, images animées, vidéos, url...

Nous avions premièrement envisagé de prendre en compte l'intégralité des médias dans l'algorithme. Une classification des emojis permettrait de déterminer les émotions positives et négatives, donc les intentions des tweets. D'autre part, les images animées provenant pour la plupart du site Giphy, classifiant les images par des tags, il serait potentiellement possible de détecter si un gif est malveillant ou non.

Cependant, à cause de la difficulté de mise en place de ces systèmes de détection (emojis à double sens pouvant corrompre la détection, données de Giphy inaccessibles ...), **nous avons choisi de nous concentrer sur l'analyse textuelle des tweets**.

Recherche des critères du cyber-harcèlement :

D'autre part, il a été proposé d'attribuer un score aux détections afin de déterminer la probabilité qu'un tweet soit offensant ou non. Le score serait établi en fonction du nombre d'insultes détectées. Cette idée a été abandonnée selon les conseils de Mme. Villata durant son interview.

Nous avions également pensé à questionner une personne travaillant dans le **domaine juridique** afin de vérifier si les critères du cyber-harcèlement que nous établiront lui paraissent pertinents. Cependant, un travail de recherche sur les articles du code pénal concernant le cyberharcèlement s'est avéré suffisant pour déterminer les critères pertinents du cyberharcèlement selon la loi française, et la précision toute relative des algorithmes n'était pas suffisante pour que l'on soumette les résultats à l'analyse d'un professionnel. En effet une telle intervention n'aurait été utile que si l'on atteignait des seuils de précision très élevés.

2) Détermination des tâches à effectuer

L'analyse des différentes pistes de travail nous a permis de classer différentes tâches à effectuer :

Récolte des données de Twitter :

Afin de récupérer les données de Twitter, nous devrons trouver un moyen d'avoir accès à l'API (interface de programmation d'application) et d'y extraire les tweets publiés.

Création des algorithmes de détection du harcèlement :

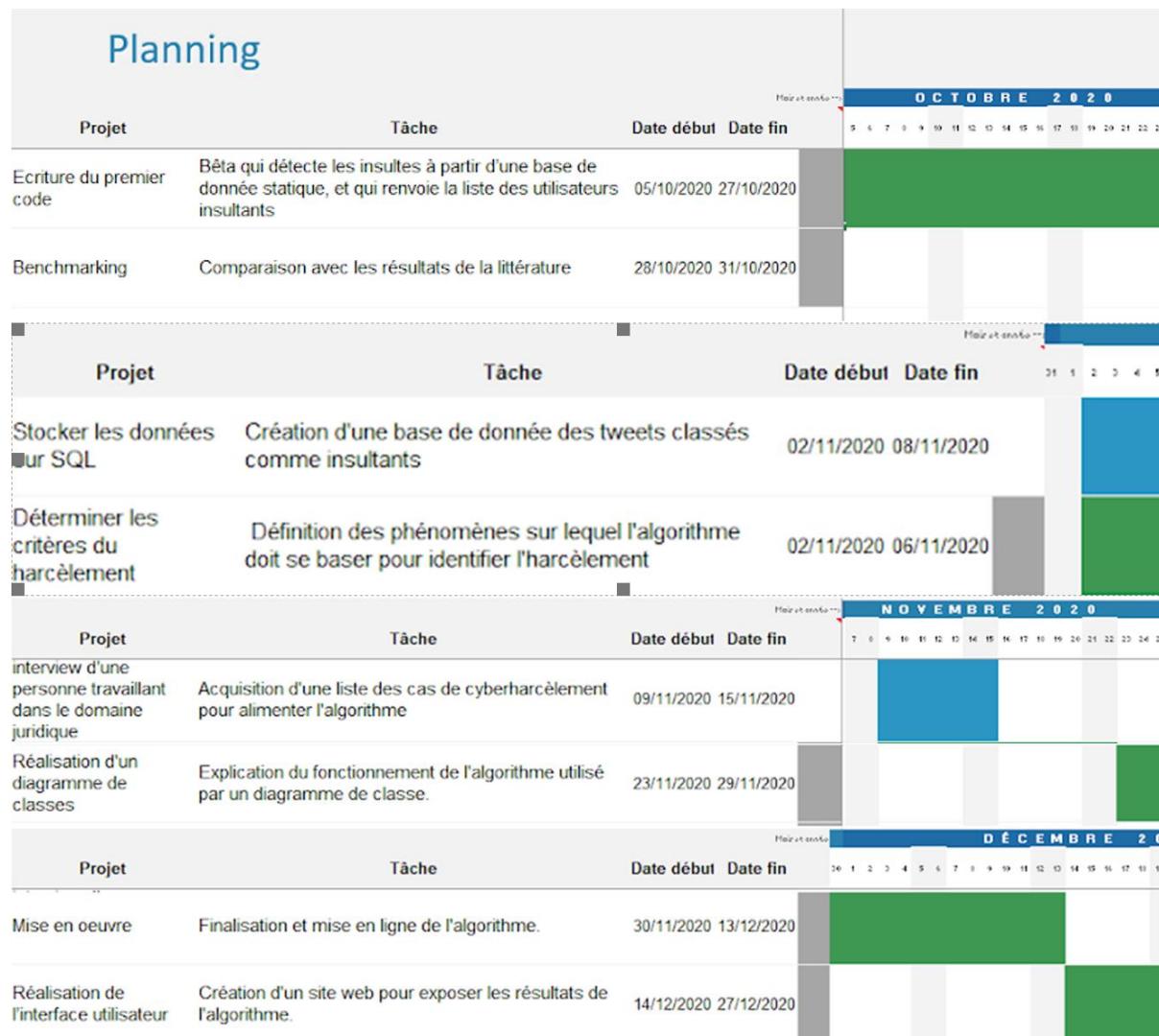
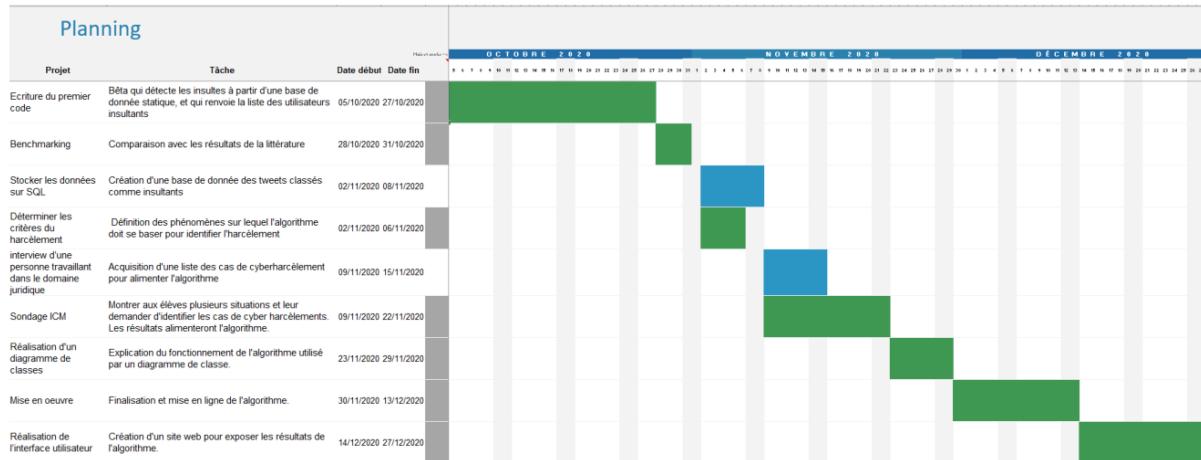
Il est nécessaire de créer un programme relié à l'API capable de détecter le cyber-harcèlement pour détecter les potentiels cas de cyber-harcèlement. Pour cela, nous devons récupérer une base de données d'insultes, puis concevoir l'algorithme qui détecte les messages offensants. Un benchmark sera ensuite réalisé à partir des résultats de la littérature^[1]. Pour obtenir de meilleurs résultats, nous pourrons ensuite y introduire une intelligence artificielle.

Création de l'interface utilisateur :

Pour pouvoir être utilisé, le système devra avoir une interface utilisateur sous forme de site web. En plus des algorithmes, un onglet "le cyber-harcèlement et la loi" présentera les textes de lois concernant le cyberharcèlement, un onglet "FAQ" permettra à l'utilisateur de comprendre le fonctionnement du site et un onglet "contact" présentera les membres de l'équipe et redirigera l'utilisateur vers des liens utiles (signalement sur Twitter, signalement à la police...).

3) Planning

Après avoir déterminé les tâches que nous aurons à effectuer, nous avons élaboré le planning suivant afin de répartir la réalisation de ces tâches dans le temps.



Planning

4) Répartition des tâches

La répartition des tâches s'est effectuée grâce au site Trello. L'image de l'annexe 1 montre la répartition des tâches selon les membres du projet. Celle-ci est également résumée dans le tableau suivant.

Membres du projet	Tâches attribuées
Adrien	<ul style="list-style-type: none">- Loi sur le cyber-harcèlement- Critères caractérisant le cyber-harcèlement- Interview de Serena Villata
Lisa	<ul style="list-style-type: none">- Interview de Serena Villata- Conception de l'algorithme naïf- Validation de l'algorithme naïf- Création des bases de données MySQL- Mise en relation des différents acteurs sur le projet Symfony- Connexion du projet au server web local- Conception de l'interface utilisateur en HTML et CSS- Ecriture de la documentation- Déploiement du projet sous Windows
Erwan	<ul style="list-style-type: none">- Prise en main de l'API Twitter- Réalisation du diagramme des cas d'utilisations- Conception de l'algorithme naïf- Ecriture de la documentation- Lien entre le code en PHP et l'IA en Python- Déploiement du projet sous Linux
Oscar	<ul style="list-style-type: none">- Conception de l'intelligence artificielle- Validation de l'intelligence artificielle- Benchmarking- Lien entre le code en PHP et l'IA en Python

III) Algorithmes de détection du cyber-harcèlement

Pour chacune des sous-parties de cette partie, nous commencerons par donner un tableau dans lequel figurent les noms des contrôleurs et des entités dont le fonctionnement et le rôle seront expliqués dans la sous-partie. Une **documentation** permettant de détailler brièvement le rôle et l'utilité de chaque fichier de code du projet a également été écrite et est disponible dans le fichier nommé *documentation*.

1) Algorithme naïf

Pour pouvoir mettre à disposition nos algorithmes à de potentielles victimes de cyberharcèlement sur un serveur web, nous avons utilisé un framework MVC (modèle-vue-contrôleur) écrit en php, nommé **Symfony** (version 4). C'est pourquoi nous avons écrit notre algorithme naïf en php qui est un langage impératif orienté objet.

Controller	Entity
- BullyingController - InsultController	- Insult

Nous avons souhaité créer un algorithme naïf étant le plus simple possible et remplissant l'objectif initial afin de pouvoir comparer les performances de notre algorithme de deep learning avec des données de type "**benchmark**". Ainsi, le rôle de l'algorithme naïf est de détecter les tweets qui possèdent au moins un mot insultant et de les rendre pour qu'ils soient ensuite afficher sur l'interface utilisateur. On définit un mot comme étant insultant s'il appartient à une base de données d'insultes.

L'algorithme est principalement composé du contrôleur **BullyingController** et ses méthodes *getBullying()* et *insultIn()*. Grâce à la méthode *getBullying()*, le contrôleur parcourt chaque tweets donnés en entrée à l'algorithme. Puis, on parcourt chaques mots qui composent un tweet et on vérifie s'ils appartiennent à un tableau d'insultes donné en entrée à l'aide de la méthode *insultIn()*. On prend soin de transformer le message du tweet un minuscule avant de le parcourir pour s'affranchir des différentes manières (majuscule ou minuscule) d'écrire un mot.

Afin de trouver une **base de données d'insultes** à fournir à notre algorithme naïf, nous avons premièrement cherché des bases de données d'insultes en français sur Internet. La première base de données a été trouvée sur *Wikipédia* et la deuxième sur *GitHub*. La deuxième base de données comportait en plus une annotation des insultes permettant de les classer par catégories (homophobie, stéréotypes ethniques...). Cependant, une grande partie des insultes contenues dans ces bases de données étaient composées de plus d'un mot et étaient donc incompatibles avec notre algorithme naïf qui ne compare qu'un seul mot après l'autre à la base de données d'insultes. Il nous aurait fallu complexifier l'algorithme naïf et ce n'était pas l'objectif que nous avions prévu pour cet algorithme. Néanmoins, nous avons testé notre algorithme naïf en utilisant uniquement les insultes composées d'un seul terme. Nous avons obtenu des résultats insatisfaisants, car les insultes contenues dans ces deux bases

de données étaient plutôt farfelues ou démodées, donc très peu de tweets fournis par l'API Twitter étaient détectés comme insultants.

Par conséquent, nous avons décidé de créer nous-même notre propre base de données d'insultes en questionnant notre entourage et en étudiant des messages insultants présents sur Twitter. Ainsi, nous avons recueilli une centaine d'insultes couramment utilisées. En utilisant cette base de données plus simple mais plus proche des insultes que nous pouvons trouver actuellement sur Twitter, nous avons obtenu de meilleurs résultats. Cette base de données est associée à la classe *Insult* qui est elle-même gérée par le contrôleur *InsultController*.

2) Algorithme de deep learning

Étant donné que les modules de deep learning les plus développés sont disponibles en langage python, nous avons choisi de l'utiliser pour ce script. Comme il est possible d'exécuter des scripts python depuis un script PHP s'ils sont présents sur le serveur, il a suffit de rendre les scripts pythons lancables depuis la console pour qu'il puissent être intégrés au projet (à quelques soucis d'architecture et de disponibilités des packages près).

Pourquoi le deep learning, et comment ? Le Natural Language Processing, ou traitement de la langue naturelle, est un domaine aussi complexe qu'étendu, et malgré les percées toujours plus impressionnantes des nombreux chercheurs du domaine, il reste pratiquement impossible de trouver (ou de concevoir) un algorithme impératif efficace de classification de phrases. Cela est d'autant plus vrai en Français, qui est un langage relativement délaissé par la communauté scientifique, notamment à cause de sa complexité de construction. Ainsi, il paraissait logique de se tourner vers le deep learning pour espérer obtenir des résultats. Pour cela nous avons utilisé la bibliothèque Tensorflow, développée par Google, pour gérer la création et l'entraînement du réseau de neurones sur un dataset relativement petit (environ 100 000 tweets).

Le réseau de neurones utilisé ici est un réseau à une seule couche de 512 neurones prenant en entrée une matrice de dimension 1000x2 représentant la phrase (obtenue via le module de preprocessing de texte du package keras) et renvoyant une valeur comprise entre 0 (non offensant) et 1 (offensant) qui caractérise le message. Il est important de noter que de nombreux paramètres du réseau (taille, profondeur, fonctions d'activation ...) pourraient être ajustés, mais qu'il faudrait pour cela progresser "à tâtons", et que ce n'est malheureusement pas possible pour nous, chaque session d'entraînement d'un réseau prenant plusieurs heures (on ne peut donc pas vraiment envisager de méta-analyse du problème pour l'optimiser plus); c'est également à cause de ce manque de puissance de calcul que la forme du réseau est si simple, pour garder des temps d'entraînement et de prédiction raisonnables. Le squelette du code est simple, car pratiquement identique à tous les projets de deep learning : création du réseau de neurones, organisation et transformation des données puis ajustement/entraînement du réseau de neurones.

Deux modèles sont créés : un modèle dit “simple” dans lequel un réseau est entraîné à déterminer le caractère offensant d'un message, et un modèle “dual” qui compare la prédiction d'un réseau entraîné à caractériser les caractères offensants du message à la prédiction d'un réseau entraîné à caractériser le caractère non-offensant d'un message, pour essayer d'augmenter la finesse de la détection.

Les résultats obtenus sont moyennement satisfaisants : même si les modèles sont capables de repérer assez bien les cas de harcèlement (en ayant un taux tout de même élevé de faux positifs), les valeurs obtenues sont souvent très proches de 0.5, ce qui traduit une forme d’indécision des modèles, probablement dû à un dataset d’entraînement trop peu étendu.

3) Statistiques

Controller	Entity
- StatisticsController	- Statistics

Afin de permettre à l'utilisateur de pouvoir quantifier le cyberharcèlement dont il se considère victime, nos algorithmes calculent quelques statistiques sur les résultats rendus. Ainsi, l'utilisateur peut connaître le nombre de tweets sur lesquels la recherche a été faite (ce nombre étant limité par l'API Twitter). L'utilisateur peut également savoir combien de tweets ont été détectés comme pouvant témoigner d'un cyberharcèlement. Un pourcentage de tweets potentiellement offensants parmi les tweets étudiés par le système est également affiché sur l'interface utilisateur.

Pour chaque nouvelle requête une instance de la classe *Statistics* est créée. À sa création, l'objet comporte la date à laquelle la demande de l'utilisateur est faite ainsi que l'identifiant Twitter de l'utilisateur (vers lesquels sont adressés les tweets offensants). Les compteurs du nombre de tweets étudiés total et du nombre de tweets détectés comme offensants sont initialisés à zéro. Puis, l'objet de la classe *Statistics* est complété au fur et mesure des itérations de l'algorithme. C'est la méthode *getBullying()* du contrôleur *BullyingController* qui se charge d'incrémenter les compteurs. Enfin, la méthode *update()* du contrôleur *StatisticsController* crée le pourcentage qui correspond aux deux compteurs et stocke l'objet de la classe *Statistics* dans la table associée *statistics*.

Pour effectuer une analyse plus complète de l'algorithme naïf dans le but de **valider** ou non ce modèle, on calcule des statistiques supplémentaires. Pour pouvoir accéder à ce mode particulier appelé “validationNaif”, l'utilisateur doit saisir l'URL [/main/validateNaif](#) qui n'est pas accessible directement via le site web puisque ce mode n'est utile qu'aux développeurs des algorithmes. Dans ce mode qui fait appel à la méthode *validateNaif()* du *MainController*, la recherche des tweets insultants se fait dans une base de données de 50000 tweets annotés (1 si le tweet est insultant, 0 sinon) qui est constituée de 50% de tweets insultants (table *tweets_demo*). Des statistiques sont alors réalisées sur les résultats fournis par l'algorithme. Premièrement, la méthode *getBullying()* du contrôleur *BullyingController* calcule la matrice de confusion suivante :

	1	0
true	VP	FP
false	FN	VN

où VP : vrai positif, FP : faux positif, FN : faux négatif et VN : vrai négatif

Par la suite, la méthode *update()* du contrôleur *StatisticsController* crée le FP rate, FN rate, recall (sensibilité), precision (vpp : valeur prédictive positive) et F-score à partir des valeurs de la matrice de confusion (voir les formules ci-dessous). Le contrôleur *StatisticsController* *fini par stocker l'objet de la classe Statistics dans la table associée statistics.*

$$\begin{aligned}
 \text{FP rate} &= \text{FP}/(\text{VP}+\text{FP}) \\
 \text{FN rate} &= \text{FN}/(\text{VP}+\text{FN}) \\
 \text{recall} &= \text{VP}/(\text{VP}+\text{FN}) \\
 \text{precision} &= \text{VP}/(\text{VP}+\text{FP}) \\
 \text{F-score} &= 2*\text{precision}*\text{recall}/(\text{precision}+\text{recall})
 \end{aligned}$$

Formules utilisées pour calculer les statistiques

Enfin, après avoir réalisé plusieurs appels à l'algorithme sur différentes données de la base de données *tweet_demo*, on crée un dernier objet de la classe *Statistics* qui permettra de stocker la moyenne de chacune des statistiques précédemment calculées. C'est la méthode *average()* du contrôleur *StatisticsController* calcule et stocke cet objet qui correspond à la moyenne des statistiques. C'est donc cet objet qui sera le plus intéressant à interpréter.

IV) Intégration des algorithmes dans un système de détection du cyber-harcèlement sur Twitter

Les deux algorithmes que nous venons de décrire ne peuvent pas fonctionner seuls : ils doivent être mis en relation avec différents acteurs. Ainsi, dans cette partie nous étudierons les acteurs nécessaires au fonctionnement de notre système qui sont l'API Twitter, la base de données MySQL ainsi que le serveur web local où l'interface utilisateur permettra à l'utilisateur de pouvoir utiliser notre système.

1) API Twitter

Controller	Entity
- TwitterAPIController - TwitterAPIRequestController	- TwitterAPIConnection - TwitterAPIRequest - TwitterAPIExchange

Twitter étant la plateforme sur laquelle nous voulons identifier le cyber harcèlement, il a fallu trouver un moyen de relever les tweets depuis la base de données de Twitter. Heureusement Twitter a développé une API qui permet, selon le niveau d'accès et la formule d'abonnement choisi, d'accéder à une certaine quantité d'informations. Nous avons donc créé un **compte développeur** qui a été validé par Twitter afin d'avoir accès à cette base de données de tweets. Notre accès nous permet de relever 100 tweets par requête. De plus, on peut uniquement relever des tweets datant d'il y a 8 jours à compter du jour où la requête est effectuée.

Dans un premier temps, nous nous sommes posé la question de comment utiliser l'API afin de développer un algorithme naïf. Nous avons vu deux possibilités :

- 1) Pour chaque message adressé à une personne (insultant ou non), vérifier pour chaque mot du message s'il appartient à la BDD d'insultes
- 2) Pour chaque insulte de la BDD d'insultes, faire une requête dans l'API qui renvoie les messages contenant cette insulte.

Pour des raisons de **complexité en temps**, et de simplicité, nous avons décidé d'opter pour la première option. Qui plus est, cela permet de mieux juger l'efficacité de l'algorithme de détection.

Le premier souci que nous avons rencontré fut l'accès à l'API Twitter depuis le code en PHP puisque nous ne savions pas du tout comment s'effectue la connexion. Pour pallier ce problème nous avons utilisé **TwitterAPIExchange.php**, un code php récupéré sur GitHub qui permet d'accéder à la version 1.0 de l'API en s'identifiant avec les tokens du compte développeur.

Ensuite, le second souci rencontré concernant l'API est le **faible nombre de tweet fourni par requête**. En effet, il est difficile de travailler avec uniquement 100 tweets. Nous avons donc utilisé une astuce qui consiste à faire 8 requêtes : une requête par jour à partir de la date à laquelle est effectuée la requête. On commence alors par récupérer les tweets destinés à la présumée victime jusqu'au jour de la requête, puis on refait la même

manipulation pour le jour d'avant tout en vérifiant qu'on n'ajoute pas un doublon à la liste de tweets récupérés. On s'arrête lorsqu'on a atteint le jour le plus ancien jusque auquel on pouvait remonter (8ième jour à partir de la date de la requête). Cela permet d'obtenir de la part de l'API Twitter jusqu'à 800 tweets à traiter. Ainsi, si l'utilisateur a été mentionné 600 fois en une semaine, il sera possible de récupérer jusqu'à 600 tweets.

Un dernier problème subsiste, c'est la performance de la détection d'insultes déjà intégrée à Twitter. En effet, Twitter possède de nombreux bots et modérateurs qui détectent si un tweet viole les conditions d'utilisation ou non, et forcent les utilisateurs à supprimer leurs tweets. Ainsi, même en récoltant 800 tweets mentionnant l'utilisateur, on trouve en général peu de tweets offensants dans ce qui est relevé, même en se focalisant sur les comptes de personnalités politiques.

Ainsi, pour pallier le problème du trop **faible nombre de tweets insultants** fournis par l'API Twitter, nous avons voulu proposer un **mode “démo”** à l'utilisateur pour qu'il puisse tester les performances des deux algorithmes. Dans ce mode, les algorithmes fonctionnent de la même manière qu'en temps normal, la seule différence provient des tweets sur lesquels ils effectuent leur recherche. En effet, les tweets fournis normalement par l'API Twitter sont remplacés par 100 tweets provenant du dataset d'apprentissage de l'algorithme de deep learning. En sachant que ce dataset est composé de 50% de tweets offensants et de 50% de tweets non offensants, on peut tester la performance de nos algorithmes. Ce mode démo permet également de voir l'algorithme naïf fonctionner dans des conditions plus "réelles", c'est-à-dire dans le cas où l'API Twitter nous permettrait d'accéder à la totalité des tweets dirigés envers une personne et donc où le flux de messages offensants à traiter par l'algorithme serait potentiellement plus élevé.

2) Base de données MySQL

Controller	Entity
- InsultController - StatisticsController - TweetController - TweetDemoController - WebRequestController	- Insult - Statistics - Tweet - TweetDemo - WebRequest

Nous avons utilisé une base de données MySQL composée de **tables** suivantes :

- insults
- statistics
- tweets
- tweets_demo
- web_requests

Chacune des tables est associée à une entité du projet. On utilise **Doctrine**, qui est un ORM (Object Relational Mapper) pour php, pour réaliser le mapping entre les objets et leur table associée.

Par exemple, pour associer une classe à une table, on écrit les lignes de code suivantes avant de définir la classe :

```
use Doctrine\ORM\Mapping as ORM;  
/**  
 * ClassExample  
 *  
 * @ORM\Table(name="table_example")  
 * @ORM\Entity  
 */
```

Ensuite, pour pouvoir envoyer (stocker) l'objet nommé *example* dans la table associée à sa classe, on écrit :

```
$entityManager = $this->getDoctrine()->getManager();  
$example = new ClassExample();  
$entityManager->persist($example);  
$entityManager->flush();
```

Chacune des tables est associée à une classe qui est elle-même gérée par un contrôleur. C'est ce dernier qui permet de pouvoir manipuler les données stockées dans la table sous forme d'objet (ou instance) de la classe.

Dans notre projet, les trois utilités principales des tables de la base de données sont:

- le **stockage des informations saisies par l'utilisateur** via l'interface (et un formulaire) afin de pouvoir ensuite permettre un accès à ses données aux classes qui les utilisent. C'est le cas de la table *web_requests* qui stocke principalement l'identifiant Twitter de la personne qui souhaite connaître les tweets offensants qu'elle a pu recevoir sur Twitter.

- le **stockage de données utiles au fonctionnement** de certaines méthodes. En effet, notre algorithme naïf a besoin de la table *insults* pour pouvoir savoir si un tweet est insultant ou non. D'autre part, les deux algorithmes, lorsqu'ils sont utilisés en mode "démo", ont besoin de tweets parmi lesquels chercher les tweets insultants : ces tweets "démo" qui remplacent donc le rôle de l'API Twitter sont stockés dans la table *tweets_demo*. De même, dans le mode "validationNaif", ce sont les tweets annotés stockés dans la table *tweet_demo* qui sont utilisés.

- le **stockage des résultats** d'un algorithme afin qu'ils puissent être ensuite affichés sur l'interface utilisateur. C'est le cas des tables *tweets* et *statistics* qui stockent les résultats de l'algorithme naïf ou de l'intelligence artificielle en vue de les afficher à l'utilisateur via l'interface utilisateur.

3) Serveur web local et interface utilisateur

Controller	Entity	View
<ul style="list-style-type: none">- ViewController- TweetController- StatisticsController- WebRequestController	<ul style="list-style-type: none">- Tweet- Statistics- WebRequest	<ul style="list-style-type: none">- base.twig.html- dossier tab- dossier result- dossier web_request

Notre système de détection du cyber-harcèlement est utilisable sur un **serveur web local**. Nous utilisons la commande Windows suivante pour le lancer : `symfony server:start`. À l'avenir, il sera possible d'héberger notre système sur un serveur web non local pour permettre réellement à n'importe quelle victime présumée de cyber-harcèlement de pouvoir collecter les tweets offensants qu'elle a reçus. Nous avons préféré nous concentrer sur la conception même du système plutôt que sur sa mise à disposition aux utilisateurs par le biais d'un site web.

Nous nous sommes néanmoins attachés à réaliser une interface utilisateur la plus ergonomique et facile d'utilisation possible. Afin d'améliorer l'expérience utilisateur, nous avons voulu créer un site moderne sur lequel il est agréable de naviguer. Pour cela, nous avons utilisé **Twig** qui est un moteur de templates pour le langage de programmation php, utilisé par défaut par le framework **Symfony**. Nous avons également utilisé **Bootstrap** qui est une librairie facilitant la création du design de sites web. Cette librairie nous a permis de créer des codes HTML et CSS, des boutons et des outils de navigation.

Pour pouvoir générer automatiquement les templates dont nous avons besoin pour afficher les résultats des algorithmes (tweets et statistiques) sur l'interface utilisateur ainsi que récupérer l'identifiant Twitter de la victime présumée, nous avons utilisé la commande Windows suivante : `php bin\console make:crud`. Cette commande crée automatiquement le contrôleur, les vues (templates) et le formulaire associé à une entité donnée. Le contrôleur créé possède les méthodes permettant de créer un objet puis le stocker dans une table associée, le modifier, le supprimer de cette table et l'afficher sur l'interface utilisateur. Toutes ces méthodes possèdent une vue associée qui permet à l'utilisateur de pouvoir directement interagir avec la table qui stocke les objets.

D'autre part, les réglages concernant l'affichage de la barre de navigation et de la banderole du site se trouvent dans le fichier `base.html.twig` et dans le dossier `public`.

De plus, chacune des vues (contenues dans le dossier `simpleView`) des différents onglets du site est générée par le contrôleur `ViewController`. Ce contrôleur permet de faire le lien entre les différentes URL du site et les templates correspondantes. En effet, chaque contrôleur est tout d'abord associé à une route par l'annotation suivante placée avant sa déclaration :

```
/**  
 * @Route("/home", name="view_home", methods={"GET"})  
 */
```

Ensuite, le contrôleur renvoie vers la vue correspondante. Voici un exemple de code qui permet de renvoyer vers la page d'accueil du site :

```
return $this->render('simpleView/homePage.html.twig');
```

Enfin, nous avons décidé de donner le nom "**e-usticia**" à notre site web, en nous inspirant du mot latin "iusticia" qui signifie "justice". Nous avons remplacé la lettre "i" par "e" pour signifier à l'utilisateur que le site web traite des problèmes de justice en ligne.

Vous pourrez trouver des captures d'écran des différents onglets de l'interface utilisateur en annexes.

4) Mise en relation des acteurs

Controller	Entity
- MainController	- WebRequest

Avant tout, nous avons mis en relation notre projet Symfony avec la base de données MySQL en entrant l'URL de la base de données locale dans le fichier `.env` du projet.

Premièrement, dans un fonctionnement normal (autre que “démo” et “validationNaif”), l’utilisateur choisit l’algorithme qu’il souhaite utiliser et remplit un formulaire de type `WebRequestType` dans lequel il saisit l’identifiant Twitter de la personne présumée victime de cyber-harcèlement. Les informations de ce formulaire sont stockées dans la table `web_requests`. L’utilisateur clique alors sur le bouton “Lancer l’algorithme”.

C’est le fichier **MainController** qui va être lu en premier par le système et dans lequel s’opère la connexion entre tous les acteurs et l’algorithme de détection du cyberharcèlement choisi par l’utilisateur.

Avant de débuter chaque algorithme de détection du cyberharcèlement, les tables `tweets` et `statistics` sont vidées.

Ensuite, le contrôleur `MainController` acquiert l’identifiant Twitter de l’utilisateur qui est stocké dans la table `web_requests`.

Puis, il envoie 8 requêtes à l’API Twitter pour récupérer des tweets adressés à la présumée victime au cours des 8 derniers jours.

Parmi ces tweets, il détecte lesquels il considère comme témoignant d’un harcèlement en ligne selon deux méthodes différentes qui dépendent de l’algorithme de détection choisi par l’utilisateur. Il stocke ces tweets offensants dans la table `tweets`.

Il réalise quelques statistiques sur les résultats obtenus et les stocke dans la table `statistics`.

Enfin, le contrôleur `MainController` renvoie vers une vue qui permet d’afficher à l’utilisateur les tweets détectés comme offensant et les statistiques calculées sur ces tweets.

Le contrôleur **MainController** possède également des méthodes (`generateDemoNaif()`, `generateDemolA()` et `validateNaif()`) qui permettent de lancer les deux autres modes de fonctionnement : “démo” et “validationNaif”, qui sont des variantes du mode “normal” où l’API Twitter est remplacée par une base de données de tweets annotés. Ces deux modes ayant déjà été expliqués dans des parties précédentes (III.3 et IV.1), nous ne détaillerons pas à nouveau leur fonctionnement.

Difficultés rencontrées :

Il est important de spécifier qu’à ce stade du développement le site ne peut pas encore utiliser l’IA codée en python. Il y a eu des problèmes qui pour la plupart ne sont pas de notre ressort qui nous ont ralenti voir stoppés dans notre démarche.

Tout d'abord, il faut savoir que pour exécuter un code en python, le programme en PHP passe par la console de commande du système sur lequel il se trouve, le programme est donc sensible au changement d'OS. Cependant, sur Windows, il y a une incompatibilité entre les versions de *Tensorflow*, *Keras* et des éléments du système de Windows qui entraîne que la version de *Tensorflow*, nécessaire à l'exécution de l'IA, ne peut pas être utilisée. Utiliser les versions antérieures qui empêche l'incompatibilité ne permet pas d'exécuter l'IA à travers la console de commande. Etant donné que nous programmons dans un environnement Windows, cela nous a causé des problèmes.

Nous avons donc décidé d'essayer de lancer notre projet sous **Linux** et avons installé les éléments nécessaires au fonctionnement du php sur une machine virtuelle Ubuntu en utilisant VirtualBox. D'après nos tests, il est possible de lancer correctement un serveur Apache afin de lire les fichiers en php, et il est aussi possible de lancer le code en Python à travers la console. Cependant, pour l'instant, l'exécution de l'IA en Python par le code en PHP ne fonctionne pas et nous n'avons pas identifié la source du problème (nous pensons qu'il s'agit d'un problème d'autorisation de l'utilisateur web et travaillons sur ce problème). Néanmoins, dès que nous aurons compris l'erreur restante, il sera normalement possible d'utiliser l'IA codée en Python avec le code en php sur Linux.

V) Résultats et validation du système

En mettant en relation nos deux algorithmes avec les acteurs précédemment cités, nous avons pu obtenir différents résultats. Enfin, nous avons cherché à valider nos deux modèles en comparant leurs performances à celles d'autres algorithmes présentés dans un article scientifique que nous avions découvert sur le site IEEE.

1) Résultats et validation de l'algorithme naïf

Résultats :

On teste l'algorithme naïf sur plusieurs personnalités publiques clivantes pour espérer que l'API Twitter nous fournit un nombre de tweets offensants assez important.

Par exemple, l'API Twitter nous fournit 701 tweets destinés à Marine Le Pen entre le 20 et le 28 décembre 2020. Parmi ces tweets 7 sont détectés comme insultants par l'algorithme, ce qui correspond à 1% des messages que l'algorithme a pu étudier.

Date	Identifiant de la victime présumée	Nombre de tweets reçus étudiés	Nombre de tweets offensants reçus	Pourcentage de tweets offensants parmi les tweets reçus étudiés
2020-12-28	@MLP_officiel	701	7	1%

Statistiques effectués sur les résultats donnés par l'algorithme naïf

En testant l'algorithme avec l'identifiant Twitter d'Eric Zemmour, on obtient des résultats similaires :

Date	Identifiant de la victime présumée	Nombre de tweets reçus étudiés	Nombre de tweets offensants reçus	Pourcentage de tweets offensants parmi les tweets reçus étudiés
2020-12-28	@ZemmourEric	720	8	1%

Statistiques effectués sur les résultats donnés par l'algorithme naïf

On teste également l'algorithme avec le mode "test" qui permet de remplacer les tweets fournis par l'API Twitter par 100 tweets tirés au hasard dans une base de données de tweets constituée de 50% de tweets insultants. Ainsi, les résultats ne dépendent plus de la personne visée, ainsi que de la modération effectuée par Twitter sur sa plateforme. Grâce à ce mode, l'utilisateur peut avoir une vision plus juste des performances de l'algorithme qu'il est en train de tester. Nous obtenons par exemple les résultats suivants :

Date	Identifiant de la victime présumée	Nombre de tweets reçus étudiés	Nombre de tweets offensants reçus	Pourcentage de tweets offensants parmi les tweets reçus étudiés
2020-12-28	@démo	100	27	27%

Statistiques effectués sur les résultats donnés par l'algorithme naïf

Vous pourrez retrouver la totalité des tweets détectés comme insultants par l'algorithme dans ces 3 exemples en annexes.

Validation :

A l'aide de la méthode `validateNaif()` du `MainController`, on teste à présent les performances de l'algorithme naïf sur 50 000 tweets constitués de 50% de tweets offensants issus de la base de données annotée `tweets_demo`. On obtient les statistiques suivantes qui sont définies comme suit :

- ***precision*** : la proportion de messages véritablement offensants parmi les messages identifiés comme tels;
- ***recall*** : la proportion de messages offensants ayant été correctement identifiés;
- ***FP rate*** : le taux de faux positifs, donc de messages non offensants parmi les messages identifiés comme offensants (définition utilisée par la littérature, bien que différente de celle que nous avons apprise);
- ***FN rate*** : le taux de message identifiés comme non offensants parmi les messages offensants;
- ***F-score*** : la moyenne harmonique de la *precision* et du *recall*.

Nombre de tweets	Pourcentage de tweets offensants	VP	FP	VN	FN	FP rate	FN rate	Recall	Precision	F-score
50000	22%	10807	387	24613	14193	0.0345721	0.56772	0.43228	0.965428	0.597171

Statistiques obtenues pour la validation de l'algorithme naïf

On remarque que l'algorithme naïf ne fait presque aucune erreur lorsqu'il détecte un tweet comme insultant (FP rate très proche de 0). L'algorithme a donc une **précision très élevée** (proche de 1). La **sensibilité** (recall) de l'algorithme est, quant à elle, **plutôt moyenne** (proche de 0,5). Cela signifie que parmi les tweets réellement insultants, l'algorithme n'en détecte que la moitié correctement (ie insultants). En effet, l'algorithme naïf passe à côté de la moitié des tweets insultants puisqu'il n'en détecte que 22% au lieu de 50% (qui est le taux de tweets insultants dans la base de données initiale). L'algorithme ne commet donc quasiment pas d'erreur en détectant des tweets comme insultants, mais il commet une erreur assez importante lorsqu'il détecte des tweets comme non insultant.

Ainsi, on peut en conclure que les tweets détectés par l'algorithme naïf sont quasiment toujours véritablement insultants, mais qu'il peut en manquer un nombre assez important (environ la moitié).

2) Résultats et validation de l'algorithme de deep learning

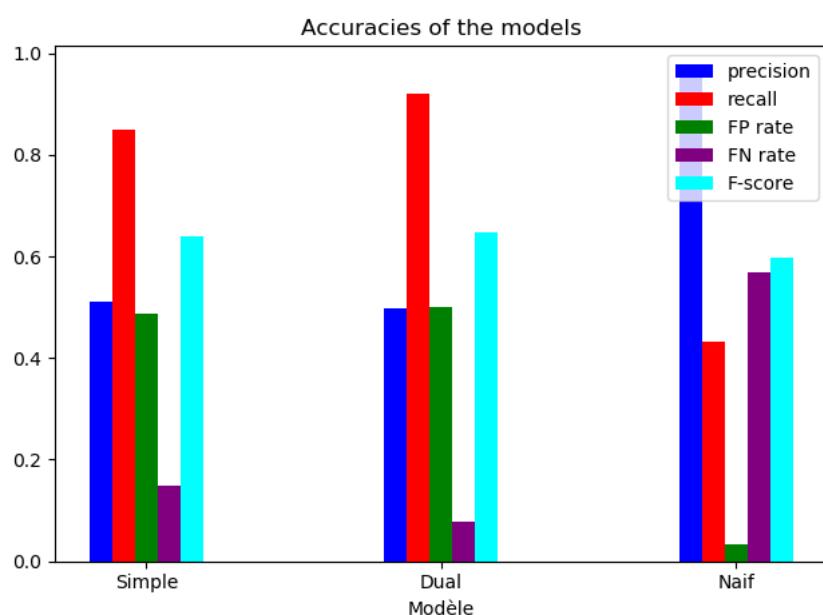
Côté algorithme de deep learning, un script est créé afin d'évaluer les modèles sur un dataset de phrases annotées, et plusieurs options sont disponibles :
-calculer les statistiques sur une partie des phrases (réglable en modifiant le code, 1% par défaut);
-calculer les statistiques sur la totalité des phrases (extrêmement long, environ 5h30);
-se servir des statistiques obtenues sur la totalité des phrases qui sont stockées en brut (option par défaut pour éviter les exécutions trop longues).

Le script est nommé `validation.py` et est disponible sur le drive du projet (il requiert comme tous les autres le téléchargement du dossier complet pour accéder aux sauvegardes des modèles ainsi qu'au dataset de test).

Les résultats des modèles sont compilés en un graphe qui résume leurs performances selon les critères évoqués plus tôt:

- ***precision*** : la proportion de messages véritablement offensants parmi les messages identifiés comme tels;
- ***recall*** : la proportion de messages offensants ayant été correctement identifiés;
- ***FP rate*** : le taux de faux positifs, donc de messages non offensants parmi les messages identifiés comme offensants (définition utilisée par la littérature, bien que différente de celle que nous avons apprise);
- ***FN rate*** : le taux de message identifiés comme non offensants parmi les messages offensants;
- ***F-score*** : la moyenne harmonique de la *precision* et du *recall*.

On obtient le graphe récapitulatif suivant :



3) Comparaison des modèles et benchmarking

On rappelle que l'on compare les **performances générales** des algorithmes naïf et de deep learning sur une base de données constituée de 50% de tweets insultants (100 000 tweets pour les algorithmes de deep learning, 50 000 pour l'algorithme naïf).

	Precision	Recall	FP rate	FN rate	F-score
Naïf	0.9654	0.4323	0.00774	0.2839	0.5971
Simple	0.5121	0.8506	0.4879	0.1494	0.6393
Dual	0.4985	0.9214	0.5015	0.0786	0.6470

En somme, les modèles obtenus grâce au deep learning sont plus sensibles, mais moins précis. Cela se traduit de la manière suivante : l'algorithme naïf n'identifie pas assez de messages offensants, mais le fait avec une bonne précision, tandis que les algorithmes de deep learning en identifient "trop" c'est à dire que pratiquement tous les messages offensants sont identifiés, mais qu'un grand nombre de messages identifiés sont en réalité non-offensants. Le taux de faux positifs de ces modèles est important, la faute à la structure des réseaux et à la "faible" quantité de données d'entraînement.

On peut tirer la conclusion suivante : un message injurieux est quasi-systématiquement offensant (d'où la grande précision du modèle naïf). Néanmoins, une bonne part des messages offensants n'est pas injurieuse, ou du moins pas détectée comme telle, d'où l'intérêt du deep learning pour améliorer le *recall* (sensibilité). Il pourrait être intéressant de combiner les deux modèles, en entraînant par exemple un réseau de neurones sur un dataset composé uniquement de messages non-injurieux, afin de s'occuper de détecter des messages qui passent à travers l'algorithme naïf, et un deuxième sur un dataset de messages uniquement injurieux, afin d'éliminer les quelques messages injurieux mais pas offensants (insultes dirigées vers un autre ou utilisées dans un but non-offensant par exemple) pour améliorer encore la précision de l'algorithme naïf.

A présent, on compare les algorithmes naïf et de deep learning selon leur efficacité pour détecter correctement différents "types" de messages insultants afin de déterminer plus précisément leurs **points forts et faibles**. Pour cela, on teste les algorithmes sur plusieurs exemples représentatifs de différents types de messages insultants. Il est à noter qu'étant donnée la nature complexe du deep learning, les réponses sont données "en principe", c'est à dire sous réserve de l'entraînement sur des données qui correspondent au cas ciblé (ce qui est naturellement le cas si on a un dataset d'entraînement assez conséquent), car les résultats peuvent différer largement selon la similitude des messages de test avec les messages d'entraînement.

	Naïf	Deep learning
Insulte couramment utilisée	oui	oui
Insulte à double sens utilisée dans son sens non insultant (ex : chien)	non	oui
Insulte couramment utilisée non dirigée vers la présumée victime	non	peu probable, car voué à être un cas marginale
Sous-entendu	non	oui
Ironie	non	oui
Humour	non	dépend largement de la taille du dataset (cas non marginal, mais rare)

Conclusion

Le projet avait pour objectif la création d'un système de détection du cyber-harcèlement sur Twitter. Ce défi a été accompli par la création de deux algorithmes complémentaires : un premier détectant les insultes simples (algorithme naïf), et un deuxième capable de détecter des messages offensants plus complexes, tels que les sous-entendus et l'ironie. En combinant les tweets offensants détectés par les algorithmes ainsi que les statistiques réalisées sur ces résultats, notre système permet d'aider une présumée victime à prouver le cyber-harcèlement dont elle est la cible. Notre site web permet également à une victime ou à ses proches de s'informer sur les possibilités d'aide qui sont mises à sa disposition par des organismes publics ou la justice.

Malgré les résultats satisfaisants livrés par les algorithmes via le site web, les performances des programmes réalisées peuvent encore être optimisées. En effet, la précision de la détection de tweets malveillants reste inférieure à celle d'autres algorithmes décrite dans la littérature (pour d'autres langues !).

Afin d'améliorer l'algorithme il serait possible d'implémenter de nouveaux critères (détection d'emoji, analyse de gifs...) comme mentionné dans la deuxième partie du rapport. Il est également envisageable de perfectionner certains aspects du projet, simplement en ayant accès à une API Twitter plus avancée (tweets en temps réel et sans limitation dans le temps) par exemple, ou via l'ajout de nouvelles fonctionnalités tel que l'obligation de se connecter avec son compte Twitter pour consulter les messages, afin d'assurer l'usage personnel des données du site.

BIBLIOGRAPHIE

- [1] Ying CHEN, Yilu ZHOU, Sencun ZHU et Heng XU. 2012. Detecting Offensive Language in Social Media to Protect Adolescent Online Safety, ASE/IEEE International Conference on Social Computing et ASE/IEEE International Conference on Privacy, Security, Risk and Trust
- [2] Michele Corazza, Stefano Menini, Elena Cabrio, Sara Tonelli, et Serena Villata. 2019. A Multilingual Evaluation for Online Hate Speech Detection. ACM Trans. Internet Technol. 1, 1, Article 1 (January 2019), 22 pages. <https://doi.org/10.1145/3377323>
- [3] Mai Ibrahim, Marwan Torki et Nagwa El-Makky. 2018. Imbalanced Toxic Comments Classification using Data Augmentation and Deep Learning. 17th IEEE International Conference on Machine Learning and Applications
- [4] Site du service public sur le cyberharcèlement :
<https://www.service-public.fr/particuliers/vosdroits/F32239>
- [5] Article 222-33-2-2 du code pénal :
https://www.legifrance.gouv.fr/codes/article_lc/LEGIARTI000037289658/2018-08-06
- [6] Trello du système :
<https://trello.com/invite/b/F8o8Lmt1/a85029bfbec6246af31bcb321d85ed42/protech-syst%C3%A8me>
- [7] Trello des algorithmes :
<https://trello.com/invite/b/r1JqjKb1/cf78e9a55cfef1e51bfe7abeaca8f1be/protech-algorithmes>
- [8] Base de données d'insultes trouvée sur Wiktionnaire :
https://fr.wiktionary.org/wiki/Cat%C3%A9gorie:Insultes_en_fran%C3%A7ais
- [9] Base de données d'insultes annotée par catégorie d'insultes, Hurtlex :
<https://github.com/valeriobasile/hurtlex>
- [10] Base de données de tweets annotée sur laquelle l'intelligence artificielle a été entraînée:
<https://github.com/Momotoculteur/Harassment-detector/tree/master/dataset>
- [11] Documentation utile pour programmer en PHP : <https://www.php.net/>
- [12] Documentation pour utiliser l'API Twitter : <https://developer.twitter.com/en/docs>
- [13] Documentation pour TwitterAPIExchange : <https://github.com/J7mbo/twitter-api-php>

ANNEXES

ANNEXE 1 : Capture d'écran de la répartition du travail via le site Trello

AG : Adrien Grimberg
G : Lisa Giordani
ET : Erwan Tinen-Touolac
O : Oscar Jozon

The screenshot shows a Trello board with the following structure:

- Board Title:** ProTech - Système
- Columns:**
 - A faire:** Contains cards like "Diagramme des cas d'utilisation" (Owner: ET), "Présentation pour la soutenance" (Owner: AG, ET, G, O), and "Création de l'interface utilisateur" (Owner: G).
 - En cours:** Contains cards like "Trouver comment mettre à disposition le projet pour le client/tuteur" (Owner: ET), "Rédaction du dossier technique" (Owner: AG, ET, G, O), and "Rédaction d'une documentation pour le code" (Owner: ET, G).
 - Vérification:** Contains cards like "Etudier l'interaction Php-Python et inclure l'IA au système" (Owner: ET) and "Compléter les informations qui seront disponibles sur le site web (onglets loi sur le cyber-harcèlement, FAQ, contacts)" (Owner: AG).
 - Terminé:** Contains cards like "Dossier pour l'évaluation du 6 octobre" (Owner: G), "Justice sur le cyberharcèlement" (Owner: AG), "Lire articles" (Owner: AG, G, O), "Mail réunion tuteur 2" (Owner: G), "Création et data cleaning de la BDD insultes" (Owner: G), "Créations des BDD pour afficher les résultats des algorithmes" (Owner: G), "Création de la BDD servant à récupérer les informations de la requête (identifiant Twitter)" (Owner: G), "Connexion du système aux BDD MySQL" (Owner: G), and "Création de la BDD de tweets annotés sur laquelle faire la démo et la validation des algorithmes" (Owner: G).
- Left Sidebar:** Shows lists under the "Idées" section, including "Utilisateur = personne harcelée ?" and "Catégoriser les insultes".
- Bottom:** A footer bar with the text "Trello - système".

ProTech - Algorithmes

- Idées** (1 card: "Tuto YouTube à reproduire")
- À faire** (1 card: "+ Ajouter une carte")
- En cours** (2 cards: "Valider l'algorithme naïf" and "Valider l'algorithme de deep learning")
- Vérification** (1 card: "+ Ajouter une carte")

Terminé

- BD insultes V1
- Interview Serena et Elena (AG, G, O)
- Recherche Langage pour l'API (Symfony, Java, C, php) (ET)
- Étudier l'API Twitter (1 comment, 3/3) (ET)
- [Pour la fin] Essayer de clean le JSON, ou renvoyer sous format différent text et nom d'utilisateur et url du tweet. (ET)
- Implémenter un algorithme naïf (G)
- Implémenter un algorithme de deep learning (O)

Trello - Algorithmes



Présentation du projet

e-ustitia est le fruit d'un projet technique lancé par quatre élèves de l'*Ecole des Mines de Saint-Etienne*, visant à combattre le **cyberharcèlement sur Twitter**.

Les algorithmes utilisés par le site détectent les tweets pouvant témoigner d'un cyberharcèlement et les mettent à disposition de la victime afin de l'aider à prouver le harcèlement dont elle est la cible.

Présentation des algorithmes

Deux algorithmes sont mis à votre disposition :

Un **algorithme naïf** qui détecte les tweets qui vous sont destinés possédant au moins une insulte largement utilisée dans le langage courant. Cette méthode est efficace pour déceler les cas flagrants de cyberharcèlement, mais peut avoir des difficultés à détecter les insultes plus élaborées.

[TESTER L'ALGORITHME NAÏF](#)

Une **intelligence artificielle** capable de détecter les insultes moins fréquemment utilisées dans le langage courant ainsi que l'ironie et l'humour. Le taux de détection de faux positifs est cependant supérieur à celui de l'algorithme naïf.

[TESTER L'INTELLIGENCE ARTIFICIELLE](#)



La loi sur le cyber-harcèlement

La loi française

Le harcèlement est défini dans le code pénal par l'article 222-33-2-2 :

« Le fait de harceler une personne par des propos ou comportements répétés ayant pour objet ou pour effet une dégradation de ses conditions de vie se traduisant par une altération de sa santé physique ou mentale est puni d'un an d'emprisonnement et de 15 000 € d'amende lorsque ces faits ont causé une incapacité totale de travail inférieure ou égale à huit jours ou n'ont entraîné aucune incapacité de travail. L'infraction est également constituée :

- Lorsque ces propos ou comportements sont imposés à une même victime par plusieurs personnes, de manière concertée ou à l'instigation de l'une d'elles, alors même que chacune de ces personnes n'a pas agi de façon répétée ;*
- Lorsque ces propos ou comportements sont imposés à une même victime, successivement, par plusieurs personnes qui, même en l'absence de concertation, savent que ces propos ou comportements caractérisent une répétition.*

Les faits mentionnés du premier à quatrième alinéas sont punis de deux ans d'emprisonnement et de 30 000 € d'amende :

- 1. Lorsqu'ils ont causé une incapacité totale de travail supérieure à huit jours ;*
- 2. Lorsqu'ils ont été commis sur un mineur de quinze ans ;*
- 3. Lorsqu'ils ont été commis sur une personne dont la particulière vulnérabilité, due à son âge, à une maladie, à une infirmité, à une déficience physique ou psychique ou à un état de grossesse, est apparente ou connue de leur auteur ;*
- 4. Lorsqu'ils ont été commis par l'utilisation d'un service de communication au public en ligne ou par le biais d'un support numérique ou électronique ;*
- 5. Lorsqu'un mineur était présent et y a assisté.*

Les faits mentionnés du premier à quatrième alinéas sont punis de trois ans d'emprisonnement et de 45 000 € d'amende lorsqu'ils sont commis dans deux des circonstances mentionnées aux 1° à 5°. »

Source : https://www.legifrance.gouv.fr/codes/article_lc/LEGIARTI000037289658/2018-08-06

Cet article a été ajouté dans le code pénal par la loi n° 2014-873 du 4 août 2014 pour l'égalité réelle entre les femmes et les hommes, puis modifiée par la loi n° 2018-703 du 3 août 2018 renforçant la lutte contre les violences sexuelles et sexistes. Toute victime de cyberharcèlement est alors en mesure de s'appuyer sur le code pénal afin de réaliser une action judiciaire, ou encore porter plainte contre le harceleur.

Le site e-ustitia

Les informations fournies par e-ustitia ne peuvent être utilisées en tant que pièces à convictions.

Le site peut cependant permettre de fournir du matériel permettant d'aider les personnes victimes de cyberharcèlement à prouver leur statut de victime auprès d'établissements scolaires par exemple.



Foire aux questions

Q : Comment fonctionne les algorithmes du site web ?

R : e-ustitia met à disposition deux algorithmes détectant les tweets pouvant témoigner d'un cyber-harcèlement à l'encontre d'un utilisateur de Twitter, qui lui ont été adressés au cours des 8 jours précédant la recherche. Pour utiliser l'un de ces algorithmes, il vous faudra suivre les étapes suivantes :

1. Vous pouvez accéder à l'un des **algorithmes** (naïf ou IA) à l'aide de la barre de navigation supérieure.
2. Ensuite, il vous sera demandé de **fournir l'identifiant Twitter** de la présumée victime de cyber-harcèlement.
3. Puis, l'algorithme choisi détectera et affichera les **tweets offensants** adressés à cette personne, ainsi que quelques **statistiques** réalisés sur ces résultats et pouvant permettre de quantifier le harcèlement qu'elle a reçu.
4. Vous pourrez accéder directement aux tweets sur Twitter en cliquant sur leur **URL** affiché sur la page de résultats.
5. Vous pourrez également **supprimer** les tweets qui ne vous paraissent pas offensants.
6. Enfin, vous pourrez **imprimer** ou **enregistrer** les tweets et les statistiques en faisant un clique droit sur la page de résultats, puis en sélectionnant "Imprimer..." ou bien "Enregistrer sous..."

Q : Comment utiliser les informations du site ?

R : Le site a pour but premier d'aider les victimes de cyber-harcèlement à récolter les informations nécessaires à **prouver le harcèlement** qu'elles subissent auprès de la justice ou d'un établissement scolaire, par exemple. Il est alors conseillé d'imprimer ou d'enregistrer les tweets et statistiques fournis par les algorithmes. Cependant, ces résultats fournis par les algorithmes du site, ne constituent pas des preuves tangibles dans une opération judiciaire, il est conseillé d'également conserver les URL des tweets (fournies par les algorithmes). Les données disponibles sur le site peuvent également permettre à une victime ou à ses proches de :

- **s'informer sur la législation** concernant le cyber-harcèlement : voir l'onglet [Loi sur le cyber-harcèlement](#)
- obtenir un **soutien psychologique** ou trouver de l'**aide** afin d'entreprendre des démarches judiciaires : voir l'onglet [Contacts](#)

Toute donnée fournie par e-ustitia ne doit pas être utilisé à des fins malveillantes visant à stigmatiser un ou plusieurs utilisateurs. L'équipe du site n'est responsable d'aucune utilisation malveillante des données fournies par les algorithmes.

Q : Je pense m'être fait harceler sur Twitter, que dois-je faire ?

R : Premièrement, vous pouvez contacter le **numéro vert** national Net Ecoute qui est une ligne d'écoute anonyme et gratuite où vous serez aidé et conseillé par des spécialistes. En parallèle, il est possible de signaler les utilisateurs malveillants sur **Twitter** pour empêcher qu'il vous nuise à nouveau. En cas de diffusion de contenu illégal ou d'insultes lié au sexe ou à l'orientation sexuelle, il est recommandé d'effectuer un signalement auprès de la **police** ou la **gendarmerie**. Vous pouvez également faire appel à un **huissier de justice** pour récolter des preuves et porter plainte (contre un individu ou contre X).

Pour plus d'informations sur les différentes démarches, rendez-vous sur l'onglet [Contacts](#) du site e-ustitia pour être redirigé vers ces différents services.

Q : Les recherches réalisées sur le site sont-elles anonymes ?

R : Partiellement. L'adresse IP des requêtes et les tweets affichés suite à une recherche ne sont pas sauvegardés. Seuls les **identifiants Twitter** saisis par les utilisateurs sur le site et la **date** à laquelle est faite la requête sont conservés afin uniquement de pouvoir mesurer l'activité sur le site.

Q : Le programme est-il open source ?

R : Les algorithmes utilisés par e-ustitia, faisant partie d'un projet scolaire, ne sont pas open source. Si vous souhaitez avoir accès aux codes utilisés, veuillez contacter l'équipe d'e-usticia (voir l'onglet [Contacts](#))

Copyright © EMSE 2020

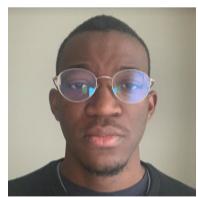


Contacts

Aide aux victimes de cyber-harcèlement

- Numéro vert national Net Ecoute (ligne d'écoute anonyme et gratuite) : 0800 200 000
- Site du gouvernement destiné aux victimes et aux témoins de harcèlement scolaire : [cliquez ici](#)
- Signaler du contenu inapproprié sur Twitter : [cliquez ici](#)
- Signaler auprès de la police : [cliquez ici](#)

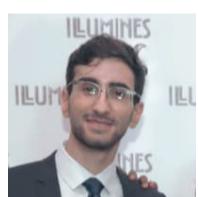
L'équipe d'e-usticia



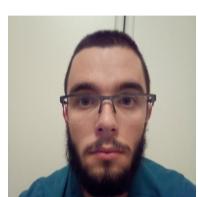
Erwan Tinen-Touolac : erwan.tinentouolac@etu.emse.fr



Lisa Giordani : lisa.giordani@etu.emse.fr



Adrien Grimberg : adrien.grimberg@etu.emse.fr



Oscar Jozon : oscar.jozon@etu.emse.fr

Remerciements

Nous remercions notre tuteur de projet M. Lefrançois pour ses conseils techniques et en matière de gestion de projet.

Nous tenions également à remercier Mme. Villata pour avoir partagé avec nous son expérience sur les algorithmes de détection du cyberharcèlement à travers une interview et un article concernant son projet CREEP.



L'algorithme naïf

Principe de l'algorithme

L'algorithme naïf détecte les tweets contenant au moins une **insulte fréquemment utilisée** dans le langage courant. Cette recherche s'effectue sur une partie des tweets reçus par la victime présumée au cours des **8 derniers jours**.

Le mode "**démo**" permet de tester les performances de l'algorithme. Dans ce mode, la recherche des tweets insultants n'a plus lieu sur Twitter mais sur une partie (100 tweets) d'une base de données de tweets contenant 50% de tweets offensants. Ces tweets ne sont donc plus destinés à un utilisateur Twitter en particulier.

Point fort : Détection des insultes communément utilisées dans le langage courant

Points faibles : Ne peut détecter l'ironie, l'humour ou les insultes plus subtiles

Utilisation de l'algorithme

Entrez l'identifiant Twitter de la présumée victime de cyber-harcèlement

[ICI](#)

Puis, [LANCER L'ALGORITHME](#)

Ou bien vous pouvez : [LANCER LA DÉMO](#)

Afin d'obtenir des informations plus complètes, il est conseillé de mettre en lien les résultats de l'algorithme naïf avec ceux donnés par l'**intelligence artificielle**.



Entrez l'identifiant Twitter de la présumée victime de cyber-harcèlement :

Recipient id



VALIDER

ANNULER

Copyright © EMSE 2020



Voici les **tweets** pouvant témoigner de cyber-harcèlement à l'encontre du compte Twitter **@MLP_officiel** que notre algorithme a détectés.

Voici également quelques **statistiques** permettant potentiellement de quantifier l'ampleur du cyber-harcèlement à destination de ce compte.

Vous pouvez **supprimer** un tweet qui ne vous paraît pas offensant, en cliquant sur le bouton supprimer à droite du tweet en question. Attention, les statistiques affichées ne seront alors plus à jour.

Enfin, vous pouvez **imprimer** ou **enregistrer** ces résultats en faisant un clic droit sur la page, puis en sélectionnant "Imprimer..." ou bien "Enregistrer sous..."

Statistiques

Date	Identifiant de la victime présumée	Nombre de tweets étudiés	Nombre de tweets offensants reçus	Pourcentage de tweets offensants parmi les tweets reçus étudiés
29/12/2020	@MLP_officiel	781	6	1%

Tweets offensants détectés par l'algorithme

Date	Identifiant de l'auteur	Nom de l'auteur	Message	URL	Actions
25/12/2020 18:39:04	@Yanice_PSG	YNBR 🎃	@MLP_officiel Je te souhaite un mauvais Noël fachiste de merde	lien	<button>AFFICHER</button> <button>SUPPRIMER</button>
24/12/2020 18:59:23	@Pablo37359563	Pablo	@MLP_officiel ftg grosse pute	lien	<button>AFFICHER</button> <button>SUPPRIMER</button>
23/12/2020 19:21:19	@ChanteurGrillon	Le Grillon Chanteur	@MLP_officiel Stupide, stupide, stupide, stupide. L'infâme raclure qui nous a déféqué ce torchon et tou... https://t.co/yIxwV89xXr	lien	<button>AFFICHER</button> <button>SUPPRIMER</button>
23/12/2020 17:51:59	@genant_supprime	PepsiGoat	@MLP_officiel marine la pute	lien	<button>AFFICHER</button> <button>SUPPRIMER</button>

Date	Identifiant de l'auteur	Nom de l'auteur	Message	URL	Actions
23/12/2020 17:40:49	@Pasc6291	Pasc629	@MLP_officiel Franchement de plus en plus débile ce gouvernement ! Moi hors de question d obéissance a quoi ce soit... https://t.co/KonjP2y8Ub	lien	AFFICHER SUPPRIMER
23/12/2020 17:35:55	@espoir32822818	veyry	@MLP_officiel donner le nom de se fils de pute	lien	AFFICHER SUPPRIMER

Copyright © EMSE 2020



Voici les **tweets** pouvant témoigner de cyber-harcèlement à l'encontre du compte Twitter **@ZemmourEric** que notre algorithme a détectés.

Voici également quelques **statistiques** permettant potentiellement de quantifier l'ampleur du cyber-harcèlement à destination de ce compte.

Vous pouvez **supprimer** un tweet qui ne vous paraît pas offensant, en cliquant sur le bouton supprimer à droite du tweet en question. Attention, les statistiques affichées ne seront alors plus à jour.

Enfin, vous pouvez **imprimer** ou **enregistrer** ces résultats en faisant un clic droit sur la page, puis en sélectionnant "Imprimer..." ou bien "Enregistrer sous..."

Statistiques

Date	Identifiant de la victime présumée	Nombre de tweets reçus étudiés	Nombre de tweets offensants reçus	Pourcentage de tweets offensants parmi les tweets reçus étudiés
29/12/2020	@ZemmourEric	666	8	1%

Tweets offensants détectés par l'algorithme

Date	Identifiant de l'auteur	Nom de l'auteur	Message	URL	Actions
28/12/2020 16:40:27	@raoullitude	Fred	@ZemmourEric ce batard à qui l'on ne dit rien alors que @StephaneGuyOff viré de canal pour son soutien à... https://t.co/BiQfDTAMo8	lien	AFFICHER SUPPRIMER
27/12/2020 19:21:40	@saidbenameur	saidbenameur	@ZemmourEric Quel batard 	lien	AFFICHER SUPPRIMER
27/12/2020 16:31:49	@EMarokinos	EI Chapo Momo Marokinos	@ZemmourEric faux de a A a Z, vous pouvez mettre zéro, je descend pas d'un singe mais de Adam et Ève , et on est tous... https://t.co/Y5qhrpU7yk	lien	AFFICHER SUPPRIMER

Date	Identifiant de l'auteur	Nom de l'auteur	Message	URL	Actions
27/12/2020 16:27:54	@EMarokinos	El Chapo Momo Marokinos	@ZemmourEric bref et l'homme descend du singe théorie de drawing,homorectus , d'abord a quatre pattes,après il se r... https://t.co/I9Yg5hGCew	lien	<button>AFFICHER</button> <button>SUPPRIMER</button>
26/12/2020 22:26:00	@LeQaLF91	Ⓜ️mohamed TNPSⓂ️	@ZemmourEric Fils de pute baise ta mère avec ton noël	lien	<button>AFFICHER</button> <button>SUPPRIMER</button>
26/12/2020 18:53:44	@dodoritto	Nuggetz 😞	@ZemmourEric Comment ça EZ j'avais pas ma bonne sensi fils de pute	lien	<button>AFFICHER</button> <button>SUPPRIMER</button>
26/12/2020 14:03:22	@slash_anti	AntiSlash	@ZemmourEric Ta gueule connard !	lien	<button>AFFICHER</button> <button>SUPPRIMER</button>
23/12/2020 18:17:33	@GenantAboie	Lam	@ZemmourEric @CNEWS bite	lien	<button>AFFICHER</button> <button>SUPPRIMER</button>

Copyright © EMSE 2020



Voici les **tweets** pouvant témoigner de cyber-harcèlement à l'encontre du compte Twitter **@démo** que notre algorithme a détectés.

Voici également quelques **statistiques** permettant potentiellement de quantifier l'ampleur du cyber-harcèlement à destination de ce compte.

Vous pouvez **supprimer** un tweet qui ne vous paraît pas offensant, en cliquant sur le bouton supprimer à droite du tweet en question. Attention, les statistiques affichées ne seront alors plus à jour.

Enfin, vous pouvez **imprimer** ou **enregistrer** ces résultats en faisant un clic droit sur la page, puis en sélectionnant "Imprimer..." ou bien "Enregistrer sous..."

Statistiques

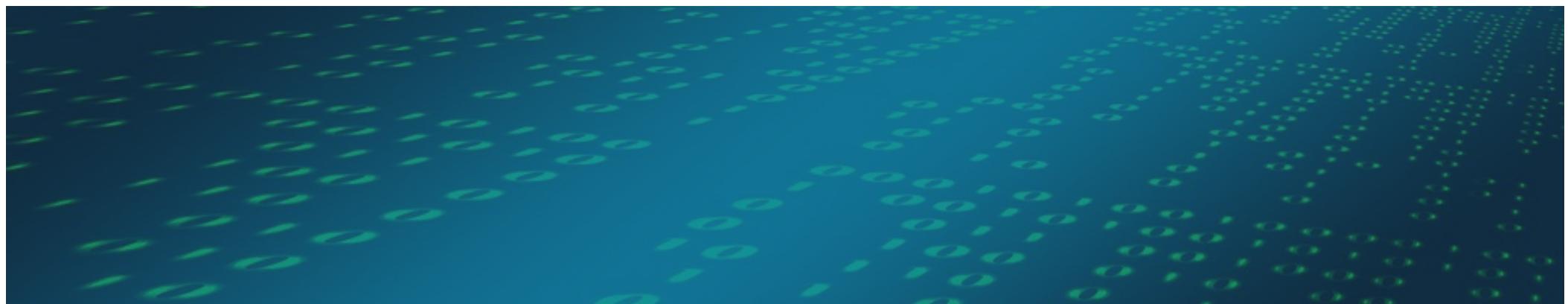
Date	Identifiant de la victime présumée	Nombre de tweets étudiés	Nombre de tweets offensants reçus	Pourcentage de tweets offensants parmi les tweets reçus étudiés
29/12/2020	@démo	100	21	21%

Tweets offensants détectés par l'algorithme

Date	Identifiant de l'auteur	Nom de l'auteur	Message	URL	Actions
29/12/2020 21:05:38	@démo	démo	Au connard qui as percuter notre voiture ce matin alors que tu étais complètement péter sous l'emprise de la drogue et que tu n'avais elle pas le permis.	lien	<button>AFFICHER</button> <button>SUPPRIMER</button>
29/12/2020 21:05:38	@démo	démo	tania cette pute me réveille à 10h du mat en m'appelant je cite « comme ça »	lien	<button>AFFICHER</button> <button>SUPPRIMER</button>
29/12/2020 21:05:39	@démo	démo	50/50 abruti de merde ? Tu merites de te faire ramasser par une caisse et de faire 1 an de reeduc sous merde	lien	<button>AFFICHER</button> <button>SUPPRIMER</button>
29/12/2020 21:05:39	@démo	démo	tu m'a tuer sale batard	lien	<button>AFFICHER</button> <button>SUPPRIMER</button>

Date	Identifiant de l'auteur	Nom de l'auteur	Message	URL	Actions
29/12/2020 21:05:40	@démo	démo	Les algorithmes n'aiment pas que je tweete Allah suce ma bite . Quand même même les algorithmes avec la modération de culture musulmane deviennent laïcophobes.	lien	<button>AFFICHER</button> <button>SUPPRIMER</button>
29/12/2020 21:05:40	@démo	démo	Pour Jonghyun je sais pas si des gens le dirais mais en tout cas pour Yoongi ils ont vraiment été bâtard	lien	<button>AFFICHER</button> <button>SUPPRIMER</button>
29/12/2020 21:05:41	@démo	démo	Ah bah y'a des gens ici qui trouvent que t'es à 100% un connard sur twitter donc bon !	lien	<button>AFFICHER</button> <button>SUPPRIMER</button>
29/12/2020 21:05:41	@démo	démo	pauvre merde...t'auras beau faire t'es un pitoyable connard ...et ça se renifle de loin...	lien	<button>AFFICHER</button> <button>SUPPRIMER</button>
29/12/2020 21:05:42	@démo	démo	Je vais y'a baiser ta mère la pute gros y'a cru j'étais ta pute tu crois Jsui qui gros Jvai te bz te mor	lien	<button>AFFICHER</button> <button>SUPPRIMER</button>
29/12/2020 21:05:43	@démo	démo	À une autre époque cette petite pute de Darmanin aurait vu sa tête plantée au bout d'une pique place de la Bastille.	lien	<button>AFFICHER</button> <button>SUPPRIMER</button>
29/12/2020 21:05:43	@démo	démo	Winter Royale de merd puteputrputsalope salope salope salop pute salope	lien	<button>AFFICHER</button> <button>SUPPRIMER</button>
29/12/2020 21:05:43	@démo	démo	Bon alors on peut imaginer que vous etes decus d'avoir perdu ? Que c'est-il passé ils etaient meilleurs que vous ? Nan nan CONNARD j'ai tout donné pour perdre c'était pas gagné vu les bites en face mais j'ai réussi à être assez mauvais pour perdre. Super heureux la !	lien	<button>AFFICHER</button> <button>SUPPRIMER</button>
29/12/2020 21:05:44	@démo	démo	@ Dybalovich parle à ta salope de pute sinon je vais la démarrer (sa va?)	lien	<button>AFFICHER</button> <button>SUPPRIMER</button>
29/12/2020 21:05:44	@démo	démo	parle bien de Stranger Things connard	lien	<button>AFFICHER</button> <button>SUPPRIMER</button>

Date	Identifiant de l'auteur	Nom de l'auteur	Message	URL	Actions
29/12/2020 21:05:45	@démo	démo	breaking news putain @QuentinWipier mort noyer la fin d'une grande aventure putain Quentin t'étais comme un frère adieu salope	lien	AFFICHER SUPPRIMER
29/12/2020 21:05:45	@démo	démo	Quooooooiiiiii Rabiot c'est pire que Neymar tu racontes quoi. Il a rien fait dans le foot et il veut plus prolonger avec vous et vous lui sucez la bite alors qu'il a rien prouvé	lien	AFFICHER SUPPRIMER
29/12/2020 21:05:46	@démo	démo	bon résumons la journée : je suis une pute qui a une sale gueule et qui mérite un sale karma. ça m'en fait des qualités	lien	AFFICHER SUPPRIMER
29/12/2020 21:05:46	@démo	démo	Nn c bon jsui qu'une pute d'acc	lien	AFFICHER SUPPRIMER
29/12/2020 21:05:47	@démo	démo	Dinguerie c vrai font les grosse putes pour de l'argent comme dans le filme « tous se qui brille » ou même la baptoup se respecte et l'arabe fait la pute avec un baptoup degeulasse	lien	AFFICHER SUPPRIMER
29/12/2020 21:05:48	@démo	démo	t une salope pk tu la bully comme ça	lien	AFFICHER SUPPRIMER
29/12/2020 21:05:48	@démo	démo	Oe Mais normalement j'aurai lâché un p'tit " connasse " jvoulais aller + loin	lien	AFFICHER SUPPRIMER



Voici quelques **statistiques** réalisées sur plusieurs appels de l'algorithme dont on cherche à évaluer la qualité. Vous trouverez également la **moyenne** de l'ensemble de ces statistiques en bas de page.

Statistiques

Type de validation	Nombre de tweets	Pourcentage de tweets offensants	VP	FP	VN	FN	FP rate	FN rate	Recall	Precision	F-score
validation_naif	10000	22%	2165	76	4960	2799	0.0339134	0.56386	0.43614	0.966087	0.600972
validation_naif	10000	22%	2149	74	4965	2812	0.0332883	0.566821	0.433179	0.966712	0.598274
validation_naif	10000	22%	2118	86	4981	2815	0.03902	0.570647	0.429353	0.96098	0.593527
validation_naif	10000	23%	2191	74	4904	2831	0.0326711	0.56372	0.43628	0.967329	0.601345
validation_naif	10000	22%	2107	66	5009	2818	0.0303728	0.572183	0.427817	0.969627	0.593688
moyenne_validation	10000	22%	2146	75	4963	2815	0.0337686	0.567426	0.432574	0.966231	0.597605



L'intelligence artificielle

Principe de l'algorithme

L'intelligence artificielle détecte les tweets pouvant témoigner d'un harcèlement à l'aide d'un algorithme de **deep learning** entraîné sur une base de données de plus de 100 000 tweets annotés (offensants ou non). Cette recherche s'effectue sur une partie des tweets reçus par la victime présumée au cours des **8 derniers jours**.

Le mode "**démo**" permet de tester les performances de l'algorithme. Dans ce mode, la recherche des tweets insultants n'a plus lieu sur Twitter mais sur une partie (100 tweets) d'une base de données de tweets contenant 50% de tweets offensants. Ces tweets ne sont donc plus destinés à un utilisateur Twitter en particulier.

Points forts : Assez bonne détection des insultes subtiles, comme les sous-entendus et l'ironie

Point faible : Taux de faux positifs plutôt élevé (tweets détectés comme offensants alors qu'il ne le sont pas)

Utilisation de l'algorithme

Entrez l'identifiant Twitter de la présumée victime de cyber-harcèlement

[ICI](#)

Puis, [LANCER L'ALGORITHME](#)

Ou bien vous pouvez : [LANCER LA DÉMO](#)



Erreur

L'intelligence artificielle n'est pas encore disponible sur le site web suite à des problèmes techniques. Nous nous excusons pour la gêne occasionnée.

Copyright © EMSE 2020