HUGON Alexandre
GRANGER Alexis
LAICH Mouna
4IABD1
2021

Sarcas'Tweet

Est-ce que tu te moques de moi ?

Documentation

SOMMAIRE

SOMMAIRE **2**

Introduction **4**

Contenu du document 4

Contexte 4

Objectifs 4

Description fonctionnelle **5**

Fonctionnement 5

Mode opératoire 5

Architecture 5

Données mises en œuvre 6

Formats *6*

Récupération *6*

Saisie et feedback utilisateur *6*

Documentation Technique **7**

Model et machine learning 7

Algorithmes mis en œuvre *7*

Cleaning 7

Embedding *7*

Librairies utilisé *8*

Paramétrage *8*

API Sarcas'Tweet 9

Installation *9*

Lancement *9*

Mise à jour du modèle 10

Routes *10*

Extension Firefox 10

Installation et utilisation *10*

Organisation **11**

Gestion de projet 11

Planning de réalisation 11

Répartition du travail 11

Bilan **12**

Problèmes rencontrés 12

Bilan fonctionnel: 12

Améliorations possibles: 12

Adresse du dépôt GitHub : 13

ANNEXE A : GUIDE UTILISATEUR **13**

Installation 13

Navigateur 13
Firefox 13
Chrome 14
Certificat 16
Evaluer le sarcasme d’un tweet 17
Envoyer un retour sur une prédiction 18

I. Introduction

a. Contenu du document

Dans le cadre de notre projet annuel au sein de l'ESGi pour notre première année de master en intelligence artificielle et big data, nous avons mis au point une solution web ayant pour but l'identification du sarcasme et cela en utilisant les différentes compétences apprises durant l'année.

b. Contexte

Les réseaux sociaux sont aujourd'hui incontournable et au centre des échanges non professionnels, twitter fait partie des plus utilisés et son format de message de type "Tweet" laisse peu de place pour étayer les propos de son auteur.
Les Tweet sont donc des messages court dans lesquels ils est parfois difficile d'identifier les réels intentions de l'auteur, un style d'écriture en particulier, le sarcasme, est souvent difficile à identifier, c'est dans ce but que nous avons créer un add-on web utilisant un modèle de reconnaissance textuel entraîné afin d'aider nos utilisateur à identifier le sarcasme.

c. Objectifs

Le but principal de notre add-on est d'indiquer la probabilité qu'un message soit sarcastique, notre utilisateur peut ainsi avoir un second regard sur le message et possiblement d'identifier un sarcasme qui lui aurait échappé.
Si il estime que notre application n'a pas jugé correctement, il peut lui indiquer si le message est sarcastique ou non afin de nous aider à améliorer notre application.

II. Description fonctionnelle

a. Fonctionnement

La solution se présente sous la forme d'un add-on web, donc une extension web téléchargeable, qui ajoute sous les tweets visible sur la page consulté par notre utilisateur un indicateur de sarcasme, cet indicateur apparaît immédiatement sous le tweet au même niveau que les autres options déjà disponible pour un tweet normal.

b. Mode opératoire

- Récupération d'un grand nombre de tweet grâce à des requêtes par mot-clés.

-Labellisation de tous les tweets sous la forme sarcastique/non sarcastique.

-Création du dataset et stockage sur Elasticsearch afin de composer un dataset d'apprentissage et un dataset de test.

-Création du modèle et entraînement sur les datasets concernés.

-Création de l'add-on web affichant sous chaque tweet les résultats du modèle afin d'aider l'utilisateur, lui permettre de faire un retour afin d'enrichir nos datasets.

c. Architecture

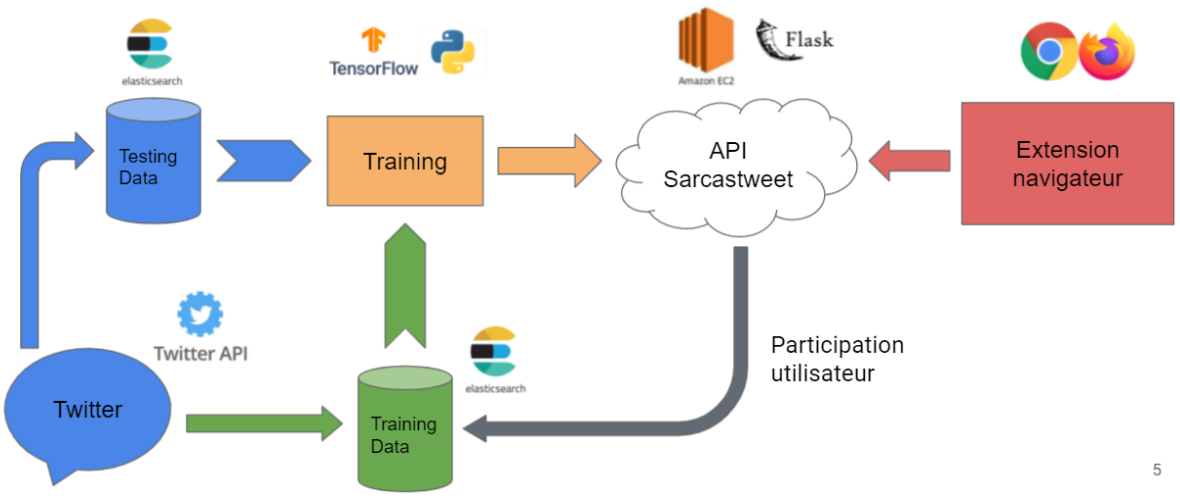


Image 1 : Schéma d'architecture du projet

d. Données mises en œuvre

i. Formats

Les données sont des tweets labellisés et stockés au format json, donc sous la forme d'une longue chaîne de caractères suivie d'un label pour identifier si le tweet est sarcastique ou non.

ii. Récupération

Afin de récupérer facilement une grande quantité de tweet nous avons utilisé l'API twitter qui permet d'obtenir le nombre souhaiter de tweet et permet aussi de leur choisir des mot-clé communs, nous avons ainsi pu sélectionner divers sujets susceptible de contenir des tweets sarcastique, cela permet aussi d'éviter les tweets de type spam qui sont souvent impersonnel, ils ne contiennent donc pas de sarcasme et sont généralement de hors-sujet concernant notre API puisqu'un utilisateur n'as pas besoin de savoir si le tweet de la météo de france-info est sarcastique ou non par exemple.
Une fois que nous avons récupéré une grande quantité de tweet nous les labellisons à la main, en effet notre application travaillant sur une langue encore peut utilisée par les IA, il n'y a que l'humain pour labellisé comme sarcastique ou non des tweet français aujourd'hui.

iii. Saisie et feedback utilisateur

Notre seconde source de données qui n'est possible qu'une fois l'API mise en place est le feedback utilisateur, nos utilisateurs peuvent d'un simple clic indiqué si un tweet est sarcastique ou non, le retour est envoyer en base de donnée afin de venir enrichir celle-ci et permettre à notre modèle de mieux s'entraîner avec des données plus variées.

III. Documentation Technique

a. Model et machine learning

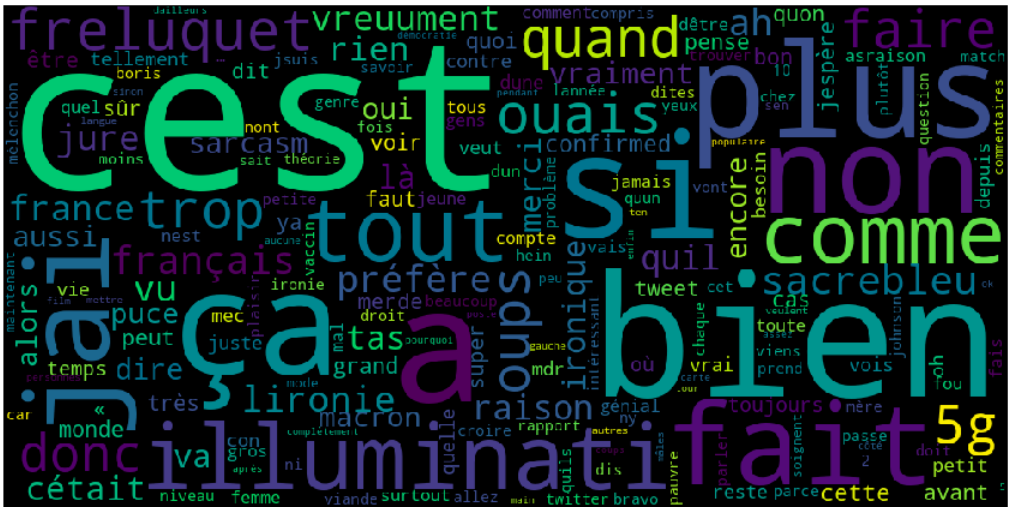
i. Algorithmes mis en œuvre

L'algorithme d'apprentissage se fera en trois parties, tout d'abord le cleaning qui est la partie la plus importante du modèle, permettant à celui-ci de catégoriser les mots les plus utilisés dans le sarcasme.
Ensuite, nous mettons en place la partie d'assemblage grâce au modèle Glove Word.
Enfin, nous finissons par apprendre le modèle à l'aide des neurones LSTM et Bidirectional

ii. Cleaning

L'étape la plus importante est le nettoyage des données. Pour une meilleure utilisation du modèle, nous devons retirer tous les pronoms qui se répètent (le,la,les, etc), l'étape suivante est de modifier toutes les abréviations afin de se retrouver avec un seul mot. Et enfin nous finissons par retirer toutes les ponctuations.

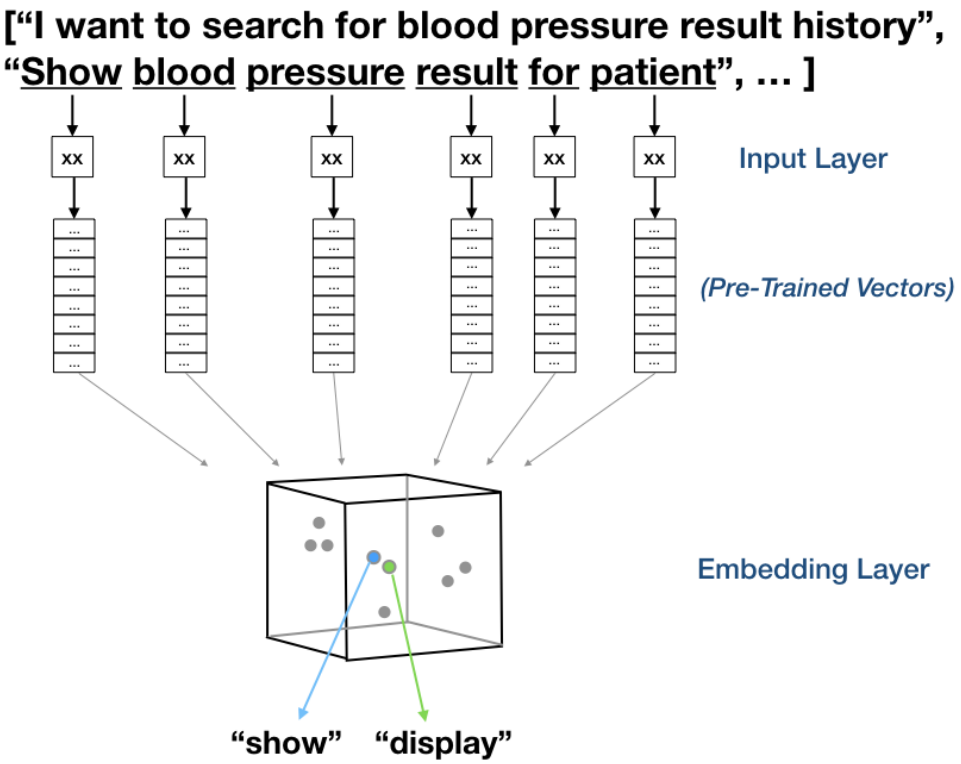
Après avoir complètement nettoyé les données, il ne nous reste plus qu'à faire une tokenisation des mots les plus employés dans le sarcasme.



iii. Embedding

GloVe est un algorithme d'apprentissage non supervisé pour obtenir des représentations vectorielles pour des mots.
L'embedding permettra de créer un dictionnaire regroupant tous les mots en leurs donnant une forme vectorielle, si le mot existe, il lui retournera sa forme vectorielle et si ce mot lui est inconnu, il lui donnera comme nouveau vecteur 0

Ci dessous voici un schema representatif de cette méthode d'apprentissage



iv. Librairies utilisé

Les libraires utilisées pour un modèle NLP sont nombreux, nos choix se sont tournés vers Bidirectional et LSTM.

Le Bidirectional signifie que les entrées seront lues dans les deux sens et conservera alors dans deux états mémoire les informations passées et celle prédite. Parfait pour notre contexte qui nécessite avant tout d'avoir un contexte passé, mais aussi de mieux maîtriser les contextes futurs.

Le LSTM (Long Short Term Memory) est un type de neurone permettant la mémorisation des informations passées.
Il permet via des mécanismes internes la mémorisation et/ou l'oubli de certaines informations selon des critères associés.

v. Paramétrage

On initialise une couche Dense de 128 unité et on régule les overfit grâce à au dropout. Afin que le modèle continue son apprentissage en fonction des nouvelles données qu'on lui donnera.

b. API Sarcas'Tweet

i. Installation

Dans le schéma d'architecture présenté précédemment, on a pu introduire les technologies utilisées par notre serveur d'API. L'application utilise le framework python Flask et Gunicorn pour créer un serveur web qui recevra les requêtes de l'extension via différentes routes.

L'API Sarcas'Tweet a besoin de Python au minimum pour pouvoir se lancer. Il est recommandé d'utiliser Linux même si une installation sous Windows est envisageable.

Flask est un framework web léger pour Python, il permet de créer rapidement une API avec différentes routes. Gunicorn est un WSGI (Web Server Gateway Interface) Python compatible avec Flask, il permet de créer une interface entre le serveur et l'application. Gunicorn va gérer les différentes requêtes entrantes et les faire parvenir ensuite à l'application Flask, ce qui améliora la rapidité de l'API. Le nombre de tweets à estimer en direct dépend du nombre d'extension installées et peut vite surcharger un simple Framework.

Donc pour utiliser l'API il faut ses prérequis :

- Python > 3
- Gunicorn (Optionnel)

Les dépendances du projet sont installables via :
pip install -r SarcasTweet/API_Server/requirements.txt

Enfin notre installation permet de faire tourner en standalone l'API sur un EC2 d'AWS et de fournir un serveur qui accepte les requêtes dite Cross-Origin. Les requêtes depuis l'extension sont effectuées depuis une page twitter. Les navigateurs et les serveurs bloquent les requêtes qui concernent une page différente de celle ouverte. Ces requêtes dites "CORS" ou Cross-Origin Resource Sharing sont à indiquer dans l'extension et dans le serveur.

De plus, une requête depuis une page sécurisée par le protocole HTTPS bloque également les requêtes vers un serveur en HTTP distant. C'est pour cela qu'il faut générer des clés TLS auto signées qu'il faut ensuite accepter dans son navigateur par la suite.

openssl req -x509 -newkey rsa:4096 -nodes -out cert.pem -keyout key.pem -days 365

ii. Lancement

Pour lancer le serveur, plusieurs commandes sont possible :

- Avec Gunicorn :
  cd SarcasTweet/API_Server/
  gunicorn wsgi:app --certfile cert.pem --keyfile key.pem -b 0.0.0.0:5000
- Avec Python :
  cd SarcasTweet/API_Server/app
  python3 main.py

iii. Mise à jour du modèle

Dans le dossier **SarcasTweet/API_Server/app** se trouve plusieurs dossiers et scripts python :

- main.py => Tout le code concernant l'API. Récupère à son lancement le modèle et le tokenizer sauvegardés.
- train_model.py => Script permettant d'entrainer un modèle à partir de la base de données elasticsearch distante.
- model/ => Dossier contenant le modèle et le tokenizer sauvegardés qui a été généré par train_model.py

Afin de mettre à jour le modèle on peut mettre en place un service cron qui lancera quotidiennement le script train_model.py puis relancera le serveur main.py via les méthodes vues précédemment (Lancement).

iv. Routes

L'API Sarcas'Tweet possède 2 routes :

- Une route permettant d'évaluer le taux de sarcasme d'un tweet
  [POST] **/evaluate_sarcasm** => body (json) : { "text" : "message_to_evaluate" }
  => return (json) : {"label": "sarcasm_rate"}

- Une route permettant d'envoyer un feedback sur l'évaluation d'un tweet
  [POST] **/send_feedback** => body (json) : {"text": "message_to_improve", "label": 0/1}
  => return elasticsearch_message

c. Extension Firefox

i. Installation et utilisation

Voir ANNEXE A : GUIDE UTILISATEUR

IV. Organisation

a. Gestion de projet

Pour la gestion de ce projet nous avons décidé de mettre à profit les différentes compétences au sein de notre groupe, pour cela il nous fallait un plan détaillé de notre application ainsi qu'une architecture fonctionnelle.
Nous avons donc commencé par décider quels types d'applications notre projet utiliseras, la base de données ainsi que le format des données, puis nous avons discuté de comment intégrer la partie IA à notre projet et comment lui fournir un nombre de données suffisant et efficace.
Suite à cette mise au point nous avions notre projet découpé en 3 grande parties, la partie applicative, la partie récupération et stockage des données, et la partie entrainement et production de modèles à l'aide du machine learning.

b. Planning de réalisation

Le planning de notre projet était assez simple, la partie entraînement de modèles nécessite des données pour pouvoir réellement commencer à produire des résultats, nous avons donc commencé par réunir des données et les labelliser.
La réalisation de l'application nécessitant beaucoup de temps, nous avons décidé de lancer sa production dès le début du projet afin de rapidement avoir une meilleure vision de nos user-cases.

c. Répartition du travail

La répartition du travail c'est faite afin de mettre en valeur les compétences de chacun, Alexis qui à de l'expérience en programmation WEB et en gestion des données à commencé par travailler sur la base de données puis sur l'application web hébergé sur un EC2, enfin il c'est occupé de l'add-on et des retours utilisateurs qui est le front-end de notre application.
Nous avions très peu d'infos sur la difficulté que poserais l'utilisation du machine learning dans notre projet, c'est pourquoi dès le début de celui-ci Moona à commencé des recherches et avant même que les bases de données soient finit à commencé à chercher quel type de modèle utilisé, à évaluer les temps d'entraînement nécessaire et effectué des test.
Une fois la base de données créée, elle s'est occupée de produire et a entraîné un modèle pour notre application.
Alexandre était chargé de trouvé un moyen d'obtenir un dataset labellisé de tweet afin de fournir la base de donnée, l'objectif de base était la quantité plutôt que la qualité mais après quelques recherches sont objectifs fut inversé et il s'occupa de la labellisation de tweet manuellement afin d'assurer une quantité de tweet labellisé suffisante au fonctionnement des autres modules du projet.

V. Bilan

a. Problèmes rencontrés

Le premier problème rencontré concernait le dataset, en effet en regardant le travail fait par nos homologue américain sur l'analyse du sarcasm sur internet il nous semblait possible d'obtenir pas simple requête un dataset de bonne qualité, en effet sur internet il est commun d'ajouter un /s à la fin d'un message sarcastique en anglais.
Pour nos homologues il suffisait donc de rechercher uniquement les messages contenant cet indicateur pour obtenir un dataset constitué à 98% de sarcasme.
Malheureusement pour nous ce n'est pas du tout le cas en français, nous avons commencé par rechercher un mot-clé qui donnerait le même résultat mais cela n'a donné aucun résultat.
Après avoir testé plusieurs combinaisons de mots-clés nous n'étions pas du tout satisfait du résultat.
Nous avons donc décidé de labellisé nous même un dataset de taille suffisante, ce qui nous a fait perdre beaucoup de temps.

Nous avons également rencontré des problèmes sur les échanges entre l'API et l'extension. En effet les navigateurs et les serveurs bloquent par sécurité les requêtes HTTP vers HTTPS ou bien les requêtes qui sont effectuées à une autre adresse que la page visitée (CORS). Contourner et attribuer les bons droits à notre application a pris du temps mais a permis de mieux comprendre le fonctionnement des navigateurs et des interactions serveurs/extensions.

b. Bilan fonctionnel:

Aujourd'hui l'extension SarcasTweet fonctionne sur Firefox comme sur Chrome et permet d'estimer les tweets parcourus en temps réel.
Derrière une API permet de générer un modèle et de prédire un taux de sarcasme sur chaque tweet des utilisateurs.
L'ensemble des données sont stockées sur un elasticsearch disponible sur AWS.

Quant au modèle employé pour estimer le taux de sarcasme, il est suffisant pour obtenir de bons résultats malgré le peu de données dont nous disposions à la base. Le modèle gagne en précision au fil du temps.

c. Améliorations possibles:

Pour améliorer le projet Sarcas'Tweet, on attend beaucoup de la participation des utilisateurs pour pouvoir fournir leurs feedbacks et grandement améliorer le modèle.

Au niveau des architecture également des choses sont à changer, nous avons utilisé une petite instance EC2 pour rendre disponible notre projet depuis internet. Les capacités ne permettent pas d'exécuter un entraînement complet périodiquement.

L'extension ne peut actuellement pas être mise sur les stores parce que l'API ne fournit pas encore un certificat TLS avec une sécurité suffisante. Actuellement le certificat a été généré

et auto-signé depuis nos machines et une génération via **Let's Encrypt** par exemple serait davantage sécurisée.

d. Adresse du dépôt GitHub :

https://github.com/agranger13/SarcasTweet

ANNEXE A : GUIDE UTILISATEUR

I. Installation

L'application s'installe à partir des sources et n'est pas persistante d'une session à une autre. Pour l'instant l'application n'est pas disponible sur les stores. Sarcas'Tweet a été développé et testé à partir de Firefox mais il est aussi possible de l'installer sur Chrome.

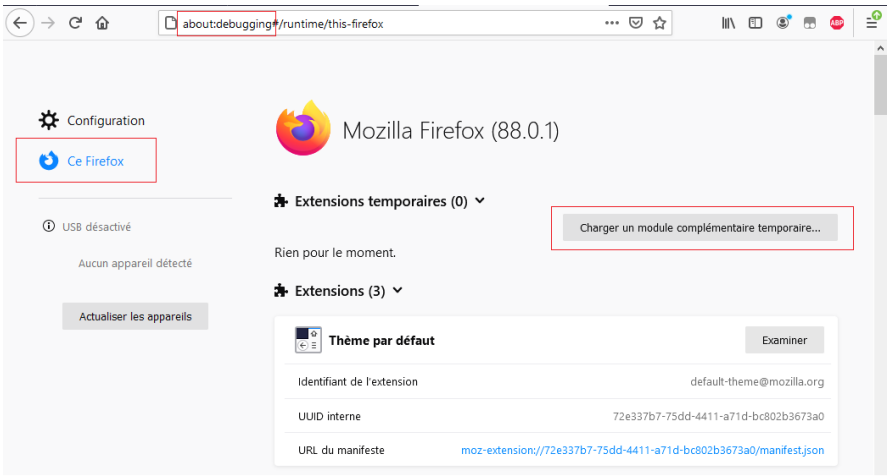
Il faut donc commencer par télécharger les sources depuis le GitHub :
https://github.com/agranger13/SarcasTweet

Puis il faut ne garder que le dossier "**Browser_Addon**" peut importe le type

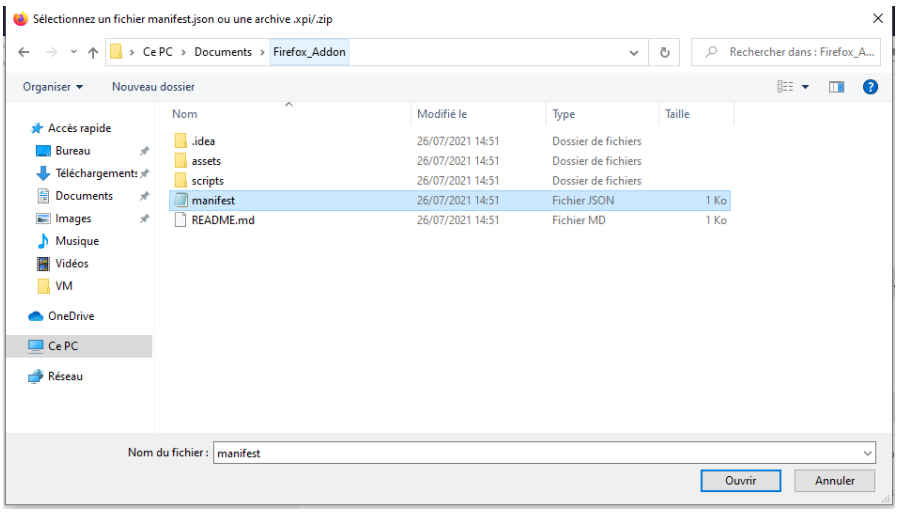
a. Navigateur

i. Firefox

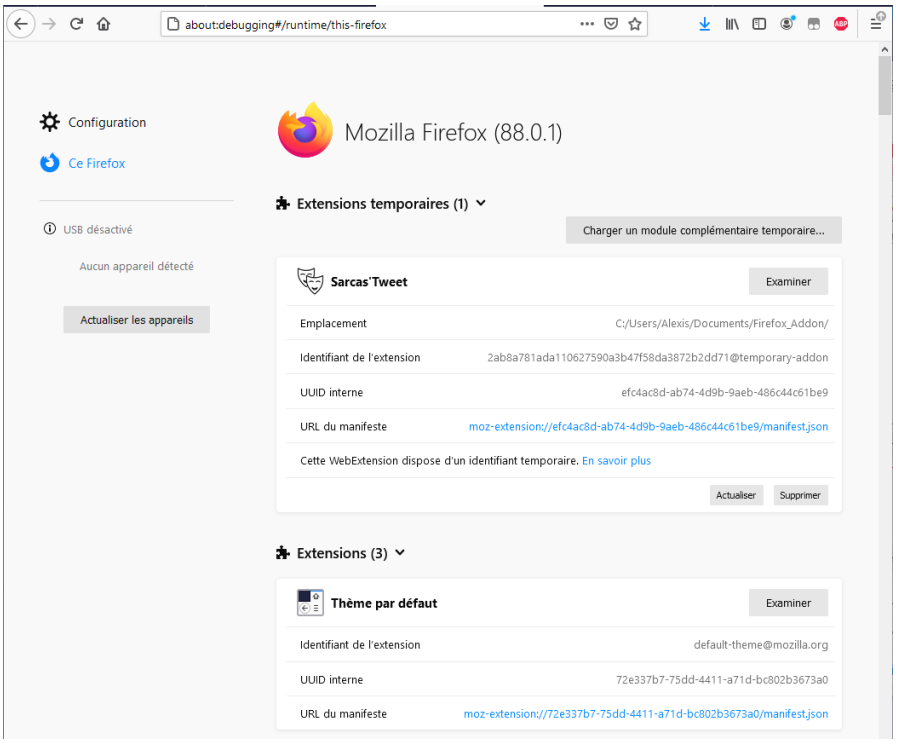
Il faut se rendre à l'url : about:debugging
Puis sélectionnez **Ce firefox** dans le menu sur la gauche et enfin cliquez sur "**Charger un module complémentaire temporaire…**"



Puis il faut sélectionner dans le dossier **Browser_Addon** le fichier **manifest.json** et valider.

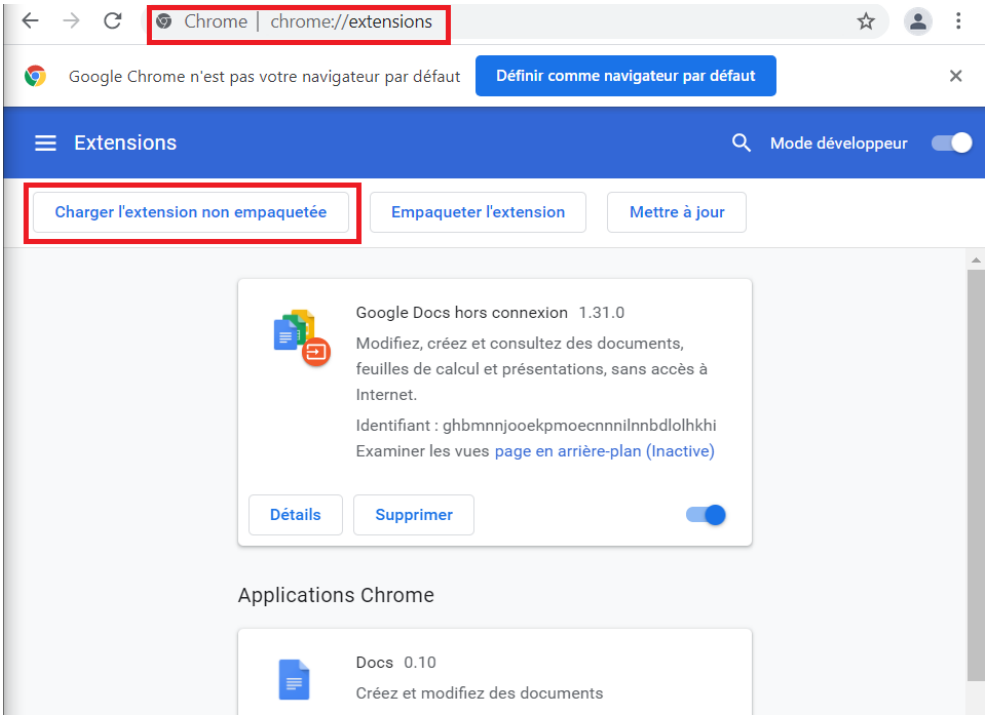


Notre extension Sarcas'Tweet devrait dorénavant apparaître dans la fenêtre.

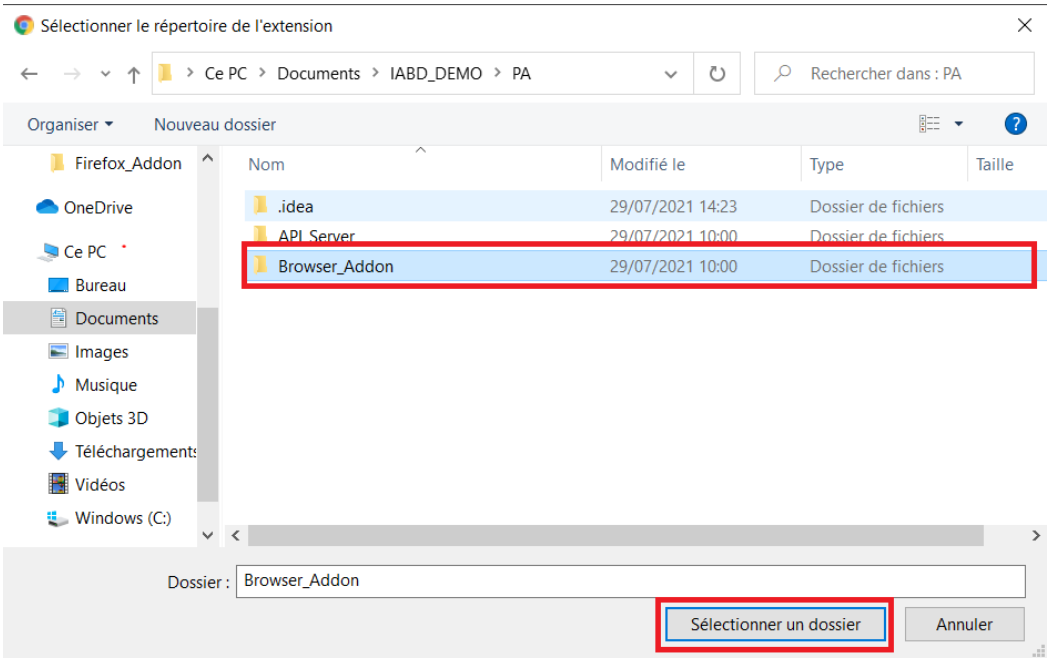


ii. Chrome

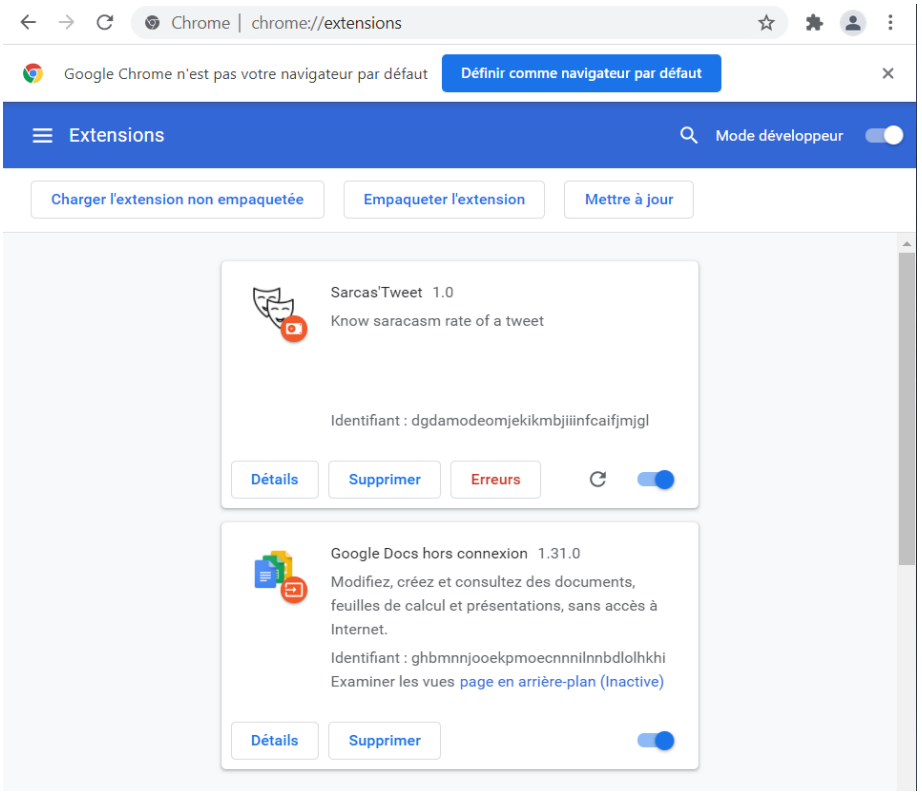
Il faut se rendre à l'url : chrome://extensions/
Puis cliquez sur "**Charger l'extension non empaquetée…**" et activer le mode **développeur**



Charger le dossier “**Browser_Addon”** et puis valider.



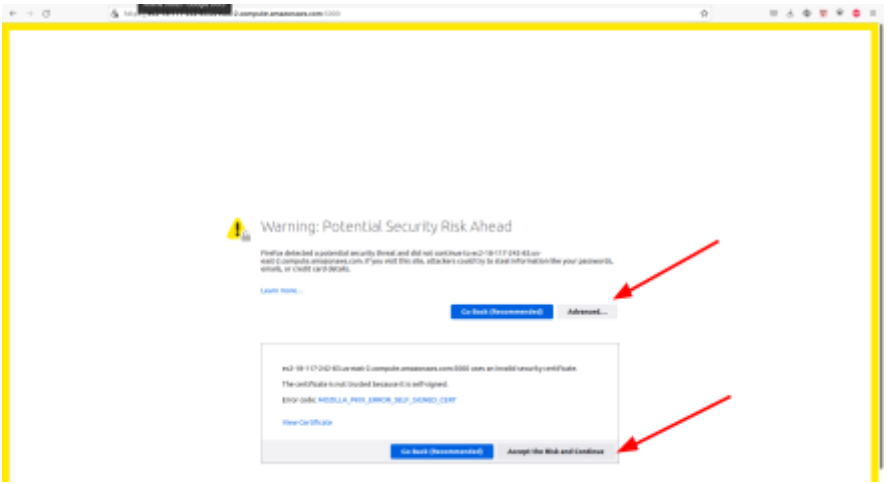
Notre extension Sarcas’Tweet devrait dorénavant apparaître dans la fenêtre.



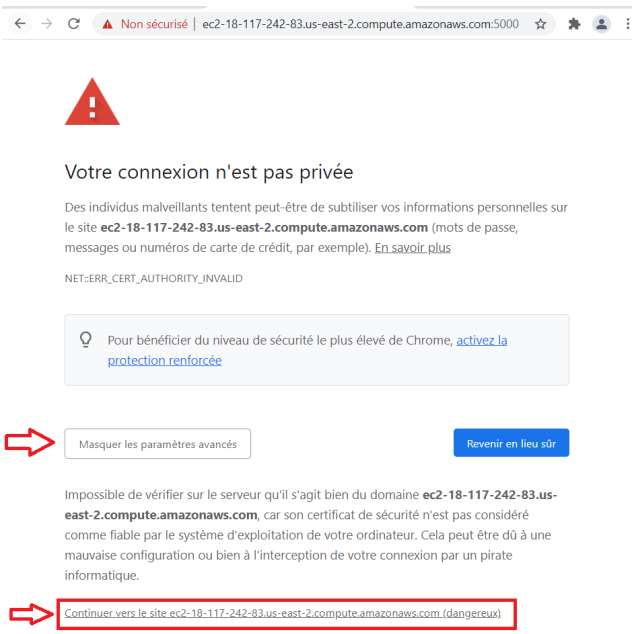
b. Certificat

Le serveur utilise un certificat TLS auto-signé pour pouvoir utiliser le protocole HTTPS. Il faut donc l'accepter avec son navigateur pour que l'extension puisse fonctionner correctement. Pour cela il faut se rendre à l'url https://ec2-18-117-242-83.us-east-2.compute.amazonaws.com:5000/

Et enfin accepter le certificat en ajouter une exception.
Firefox :



Chrome :



Sarcas'Tweet est enfin prêt !

II. Evaluer le sarcasme d'un tweet

Pour évaluer des tweets, il suffit de naviguer sur twitter comme à votre habitude. Un nouvel icône doit apparaître sous chaque tweet à côté des autres icônes de like et de partages. Ce nouvel icône vient avec un pourcentage qui détermine à quel point un message est sarcastique. Les pourcentages sont accompagnés d'une couleur pour indiquer le sarcasme du tweet :

- Vert : Ce tweet n'est pas sarcastique ! Vous pouvez avoir confiance dans ce message

- Bleu : L'algorithme n'a pas pu trancher, nous vous encourageons à nous envoyer votre feedback pour que Sarcas'Tweet s'améliore !

- Rouge : Ce tweet est sarcastique, certains mots et/ou tournures de phrase ont retenu l'attention de l'application. Méfiez-vous du message.

III. Envoyer un retour sur une prédiction

Pour envoyer un retour sur taux sarcasme qui vous semble bas ou haut, vous le pouvez **en cliquant sur l'estimation** puis en sélectionnant **"+"** ou **"-".**

Une popup doit alors apparaître pour signaler que tout s'est bien passé.

Vos feedbacks seront pris en compte dès la prochaine mise à jour du modèle donc pas de panique si l'estimation ne change pas tout de suite