Rapport projet :

**Présentation du projet :**

Afin de mettre en œuvre ce que nous avons appris dans ce cours, nous avons décidé de travailler sur la détection de *Cyber Troll*.

« *Le terme troll désigne, dans le jargon de l'internet, un personnage malfaisant dont le but premier est de perturber le fonctionnement des forums de discussion* », ainsi, il existe différents types de troll :

* Troll débutant : Le troll débutant (appelé aussi « troll qui s’ignore ») est une manière de troller par ignorance de la *nétiquette* (i.e. les règles de conduite et de politesse) et du fonctionnement technique, sans véritable intention de nuire.
* Troll bête : Persuadé d’avoir une opinion valable sur tout, d’être de bonne foi, le troll bête prend l’apparence d’un message véritable.
* Troll méchant : Son but est, consciemment, de tuer les forums. Il cumule tous les types détaillés plus haut.

Ainsi, la détection de troll est un problème récurrent dans la modération des forums.

Afin de réussir à détecter les trolls, nous avons décider d’utiliser un Neural Network que nous avons entrainé sur un ensemble de données annotées.

**Description des tâches :**

Dans un premier temps, nous avons téléchargé et prétraité l’ensemble des données sur lesquels nous avons choisis de travailler.

Ensuite, nous avons essayé de mettre en place différents « *Classifier* » afin de voir lequel était le plus compétent pour répondre au problème posé.

Enfin, nous pourrions résumer notre travail en 4 parties :

1. Prétraitement
2. Conversion des données en nombres
3. Entrainement du classifier et prédiction
4. Evaluation des modèles

Bien évidemment, nous mettrons aussi en place différents graphiques afin de visualiser les données et les résultats obtenus par nos classifiers.

**Conception et implémentation :**

Dans cette partie nous détaillerons pas à pas les différentes étapes que nous avons suivies lors de notre expérimentation.

Pour commencer, nous avons importé le jeu de données. Et afin de manipuler la data, nous avons transformé le fichier csv en « DataFrame ». Ce qui nous a permis par la suite d’utiliser les méthodes de **pandas** afin d’observer et de comprendre les différentes features et d’exploiter le matériel sur lequel nous travaillerons par la suite.

Notre première partie de travail a consisté à supprimer les colonnes/features inutiles :

Nous avons donc commencé à chercher s’il existait des colonnes dont le contenu serait nul. Nous en avons trouvé deux : annotation/notes et extras.

Ensuite nous avons cherché à supprimer les colonnes qui ne seraient composées que d’une unique valeur. Quatre d’entre elles se sont révélées correspondre à cette description : metadata/sec\_taken, metadata/last\_updated\_by, metadata/status, metadata/evaluation. Elles seraient inutiles dans l’algorithme de classification.

Pour finir nous nous sommes rendu compte que deux autres colonnes n’étaient pas pertinentes : metadata/lastupdates\_at et metadata/first\_done\_at car elles étaient composées uniquement de timestamps.

Lorsque ce tri a été fait, il ne nous restait que 2 colonnes pertinentes pour la classification : content (composée du texte) et annotation/label (composée de 0 [Not Cyber-Troll] ou 1 [Cyber-Troll])

L’étape suivante de notre projet a été le prétraitement du texte.

Le texte peut contenir des nombres, des caractères spéciaux ou encore des espaces indésirables. C’est pourquoi nous les avons retirés du texte à l’aide de WordNetLemmatizer.

Après avoir appliqué notre méthode au contenu, la visualisation nous a permis de confirmer que nous pouvions passer à l’étape suivante.

L’analyse de données nous a permis de convertir le texte afin qu’il soit compréhensible par l’algorithme de classification.

Différentes approches existent. Nous avons choisi d’utiliser le « Bag of Words Model » (Sac de mots en français) qui consiste à transformer le texte en représentation numérique.

Pour être plus précis et mieux comprendre son utilisation voici un exemple tiré de notre projet : (en théorie)

1. Phrase extraite du jeu de donnée :

« i just deleted all the people that i have on my facebook off myspace Damn if everyone switched i could just delete the account »

1. Liste construite à partir des mots de la phrase :

« i », « just », « deleted », « all », « the », « people », « that », « i », « have », « on », « my », « facebook », « off », « myspace », « Damn », « if », « everyone », « switched », « i », « could », « just », « delete », « the », « account »

1. La liste est représentée en « sacs de mots » : chaque clef est un mot distinct tandis que sa valeur est le nombre d’occurrences du mot dans la phrase (ou groupe de phrases)

{« i » : 3, « just » : 2, « deleted » : 1, « all » : 1, « the » : 2, « people » :1, « that » :1, « have » :1, « on » : 1, « my » :1, « facebook » : 1, « off » :1, « myspace » :1, « Damn » : 1, « if » :1, « everyone » :1, « switched » :1, « could » : 1, « delete » :1, « account » :1}

Dans notre cas, le modèle va tout d’abord définir le vocabulaire (c’est l’ensemble des mots distincts provenant de la phrase extraite et arrangé comme un vecteur).

Ensuite le vecteur de vocabulaire est utilisé pour convertir la phrase en un vecteur de fréquence. (Comme expliqué ci-dessus)

Output : array([3, 2, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]

Cependant l’utilisation de ce modèle ne suffit pas.

Il ne prend pas en compte que le mot peut aussi avoir une fréquence importante dans d’autres cas d’utilisations (d’autres documents).

TFIDF permet de résoudre ce problème en multipliant la fréquence d’un mot par la fréquence inverse du document. C’est pourquoi nous l’avons implémentée.

Lorsque ces étapes ont été réalisées nous avons pu nous attaquer à l’implémentation des différents modèles de classification.

Nous avons décidé de tester 5 modèles différents dans le but de trouver le plus optimal d’entre tous.

En effet afin de répondre à la question quel modèle devons-nous utiliser ?

Nous nous sommes rendu compte qu’il était impossible de prédire lequel d’entre eux serait le plus efficace sans les expérimenter au préalable.

C’est pourquoi nous avons mis en place les plus connus d’entre eux et ceux qui nous paraissaient les plus appropriés.

Dans l’ordre nous avons testé : RandomForestClassifier, MLPClassifier, LogisticRegression, KNeighborsClassifier et finalement LinearSupportVectorMachine Classifier.

Sources :

Dataset : <https://dataturks.com/projects/abhishek.narayanan/Dataset%20for%20Detection%20of%20Cyber-Trolls>

Définition du cyber troll : <https://www.commentcamarche.net/faq/3610-qu-est-ce-qu-un-troll-informatique>

Inspirations :

<https://stackabuse.com/text-classification-with-python-and-scikit-learn/>