Ecole Nationale Supérieure d'Informatique et d'Analyse des Systèmes - RABAT

Rapport de Projet d'Intelligence Artificielle : Fake-news detection

Réalisé par :

Abderrahman BEN SALLAH
Nabih MOCHIR

Encadré par :

Pr. Houda BENBRAHIM
Pr. Lamia BEN HIBA

Année académique 2021/2022

Remerciements :

Nous voudrons tout d'abord adresser toute notre gratitude à notre Professeur BENBRAHIM Houda, qui a eu le soin d'examiner et encadrer ce travail et ainsi nous offrir une véritable opportunité d'apprentissage.

Nous souhaitons également adresser nos vifs remerciements à notre professeur BENHIBA Lamia, ses savoir-faire et les méthodes et raisonnement qu'ils nous ont transmis lors des séances des travaux pratiques.

Enfin, nos remerciements s'adressent à notre cher coordinateur de la filière e-MBI JANATI IDRISSI Mohmamed Abdou, pour son dévouement au corps estudiantin.

Table des figures

Table des matières

Introduction générale

"Ne croyez pas tout ce que vous voyez sur Internet, ne croyez pas une chose simplement parce qu'il y a des guillemets et une signature."
- Abraham Lincoln

La diffusion de fake news, notamment via les réseaux sociaux, est devenue un problème de société majeur. De nombreuses initiatives ont été lancées pour aider les professionnels de l'information et les citoyens à se protéger. Cependant, détecter les fausses nouvelles est un défi complexe car elles sont intentionnellement écrites pour induire en erreur et tromper. Les humains ne sont pas bons pour identifier les fausses nouvelles (Fake news). La détection de fausses nouvelles par les humains serait à un taux de 54%. Ainsi, l'importance de la lutte contre les fausses nouvelles est illustrée par la pandémie actuelle. Par conséquent, les réseaux sociaux intensifient l'utilisation d'outils de détection et éduquent le public à reconnaître les fausses nouvelles. L'IA peut nous aider à l'arrêter ce phénoméne, ou au moins à le ralentir, en particulier un outil clé dans cette lutte est l'apprentissage automatique.

Ce projet étudiera comment les techniques de traitement du langage naturel et l'apprentissage automatique peuvent être combinés dans une approche de fusion pour créer un modèle qui utilisera les données des tweets précédents et prédira que l'information d'un nouveau tweet est vrai ou non.

Dans ce projet nous allons essayer d'écrire un programme en Python, dans un envorenement Conda, qui va detecter les Fake tweets (Fake news dans les tweets). Un réseau de neurones (backpropagation neural network) sera entrainé, pour faire cette detection, en utilisant un échantillon des Tweets préalablement étiqués. Un modèle d'extraction d'attributs simple sera utilisé.

Chapitre 1

Contexe général du projet

Dans ce chapitre, nous pourrons présenter le projet dans son contexte général et introduire Le sujet du projet et ses spécifications. Nous énoncerons également la problème et les objectifs pour mener à bien ce projet de l'intelligence artificielle.

1.1 Les réseaux des neurons :

Les réseaux de neurones sont des modèles mathématiques qui utilisent des algorithmes d'apprentissage inspirés du cerveau pour stocker des informations. Étant donné que les réseaux de neurones sont utilisés dans les machines, ils sont collectivement appelés «réseau de neurones artificiels».

Semblables au cerveau, les réseaux de neurones sont constitués de nombreux neurones avec de nombreuses connexions entre eux. Les réseaux de neurones ont été utilisés dans de nombreuses applications pour modéliser les relations inconnues entre divers paramètres sur la base d'un grand nombre d'exemples. Des exemples d'applications réussies des réseaux de neurones sont les classifications de chiffres manuscrits, la reconnaissance vocale et la prédiction dans les marchés boursiers.

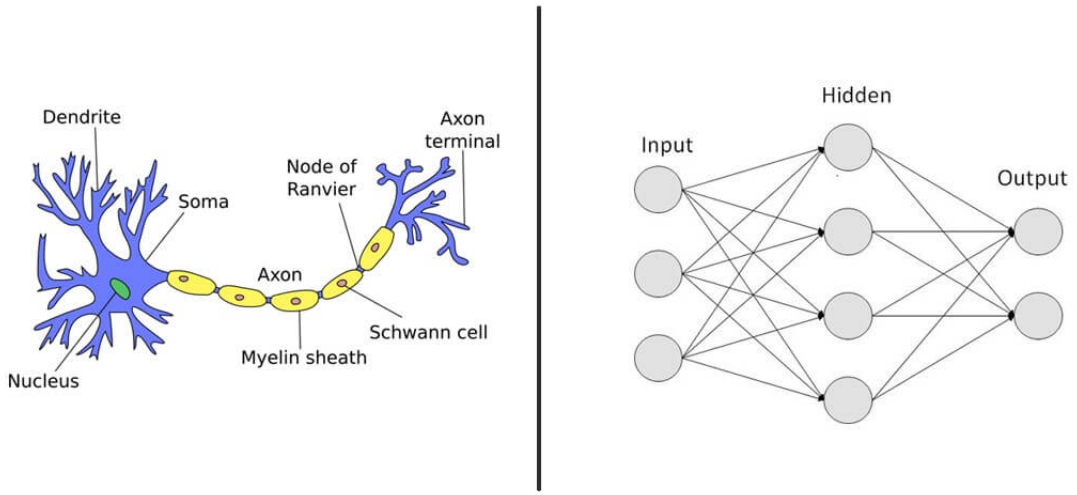

FIGURE 1.1 – Graphe représentant des neurones biologiques et un réseau de neurones artificiels [1]

1.2 Natural Language Processing - NLP :

Le traitement du langage naturel (NLP) est une branche de l'intelligence artificielle ou de l'IA qui vise à donner aux ordinateurs la capacité de comprendre le texte et les mots parlés de la même manière que les êtres humains.
La NLP combine la linguistique informatique (modélisation basée sur des règles du langage humain) avec des modèles statistiques, d'apprentissage automatique et d'apprentissage en profondeur. Ensemble, ces technologies permettent aux ordinateurs de traiter le langage humain sous forme de texte ou de données vocales et de « comprendre » sa signification complète, avec l'intention et le sentiment de l'orateur ou de l'écrivain.

1.2.1 Projet de l'Intelligence Artificielle :

1.2.1.1 Problèmatique :

- Quelle mesure peut-on exploiter l'intelligence artificielle pour luter contre les fake news ?
- A quel point notre système basé sur les réseaux de neurones va distinguer entre un fake tweet (un tweet contenant de l'information éronée) et un real tweet (un tweet contenant de vraie information) ?

1.2.1.2 Objectifs du projet :

- Exploiter les connaisssances aquises dans le module de "Intelligence Artificielle" menés par le cours de Mme BENBRAHIM et les séances des travaux pratiques de Mme BENHIBA.
- Comprendre tout les étapes nécessaires pour construire un réseau de neurons.
- Comprendre l'étape du Data Pre-Processing et Data Cleaning.
- Réaliser une interface graphique qui implément le modèle réalisé.

1.3 Conclusion :

Afin de réaliser un projet, il faut bien avant le positionner dans son contexte général et comprendre tout ces aspects. C'est aussi qu'on va par la suite faire une étude analytique et conceptuelle pour notre projet, avant de le réaliser.

Chapitre 2

Analyse et conception du projet

Nous présenterons dans ce chapitre l'analyse et la conception de notre projet, en expliquant exactement la méthodologie avec la quelle on va réaliser ce dernier.

2.1 Data pre-processing :

2.1.1 Data Cleaning :

Avant d'utliser les données qu'on a pour allimenter notre modèle d'intelligence artificielle, il est indispensable de les nettoyer d'abord. Le nettoyage des données est le processus de modification, de correction et de structuration des données dans un ensemble de données afin qu'elles soient généralement uniformes et préparées pour l'analyse. Cela comprend la suppression des données corrompues ou non pertinentes et leur formatage dans un langage que les ordinateurs peuvent comprendre pour une analyse optimale.

2.1.1.1 Suppression des mots vides "Stop Words" et des symboles :

Les mots vides "stop words" sont les mots les plus courants dans toutes les langues et n'ajoutent pas beaucoup d'informations au texte. Ils sont filtrés avant le traitement du langage naturel. Ainsi, Les ponctuations, emojies et les symboles présentes dans notre données textes n'ajoutent pas de valeur, donc on les filtre aussi.

	id	tweet	label
Avant Data Cleaning	6417	? ? ?Autopsies prove that COVID-19 is ? ?@ a blood...	fake
Après Data Cleaning	6417	autopsies prove that covid is a blood	fake

Figure 2.1 – Exemple d'avant et après le Data Cleaning

2.1.1.2 Stemming et Lemmatization :

Le Stemming et le lemmatisation génèrent toutes deux la forme racine des mots. Ces deux techniques sont utilisés pour préparer du texte, des mots et des documents en vue d'un trai-

tement ultérieur. Un exemple de de ces methodes consiste à transformer les mots "playing" "played" "plays" en un seul mot "play".

2.1.2 Tokenization :

Tokenization consiste simplement à diviser le texte en jetons. En d'autres termes, la chaîne de caractères est convertie en une liste, où chaque élément correspond à un mot.

2.1.3 Visualisation de les données :

Il est très utile de prendre l'habitude de visualiser le résultat ainsi que les données, sur ce on vous présente des graphes nuages des nots (word cloud) suivants de nos données.

FIGURE 2.2 – Nuage de mots des Tweets contenant des informations vraies

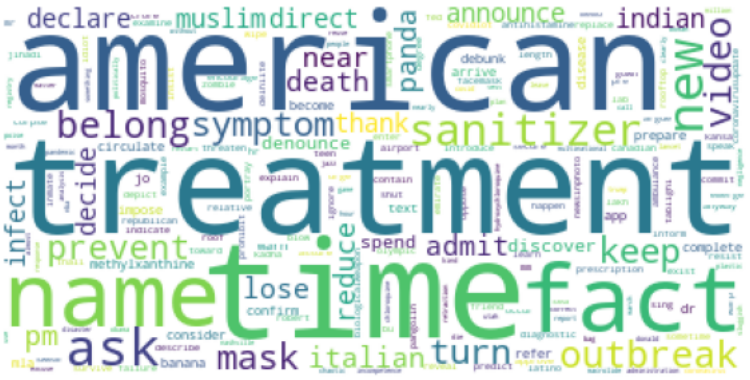

FIGURE 2.3 – Nuage de mots des Tweets contenant des informations fausses

2.1.4 Bag of words :

Dans la section précédente, nous avons discuté le nettoyage des données (Data Cleaning). Mais la donnée texte nettoyée ne suffit pas pour être transmise directement au modèle de classification. Les caractéristiques des données doivent être numériques, et non des chaînes de caractères. Il existe de nombreuses approches de pointe pour extraire les caractéristiques des données textuelles.
La méthode la plus simple et la plus connue est la représentation Bag-Of-Words. C'est un algorithme qui transforme le texte en vecteurs de longueur fixe. Ceci est possible en comptant

le nombre de fois que le mot est présent dans un tweet.
Les occurrences de mots permettent de comparer différents tweets et d'évaluer leurs similitudes pour les classifier.

2.1.5 TF-IDF :

Les modèles qui traitent d'énormes quantités de texte pour effectuer la classification nécessitent une étape supplémentaire pour traiter ces types de données. Les données textuelles doivent être transformées en quelque chose d'autre, des nombres, qui peuvent être compris par les ordinateurs. Il existe de nombreuses techniques disponibles pour créer de nouvelles fonctionnalités à l'aide de ces données.
L'un d'eux est Term Frequency-Inverse Document Frequency, également appelé TF-IDF. C'est une technique utilisée pour quantifier l'importance des mots dans un tweet et dans un ensemble de tweets en même temps.
Beaucoup de mots communs comme « le », « est », « je » qui apparaissent fréquemment dans les phrases, mais ne contribuent pas de manière significative à apporter des informations. Si l'on ne regardait que le terme fréquence, ces mots apparaîtraient plus importants que d'autres mots. Pour cette raison, TF-IDF est introduit pour avoir des fonctionnalités plus robustes dans l'ensemble de données.

2.2 la phase d'apprentissage (Training Phase) :

Il y a 2 phases dans le cycle de vie du réseau de neurones et tous les algorithmes d'apprentissage automatique, en général, qui sont la phase d'apprentissage et la phase de prédiction. Le processus de recherche des valeurs de pondération et de biais se produit lors de la phase d'apprentissage.

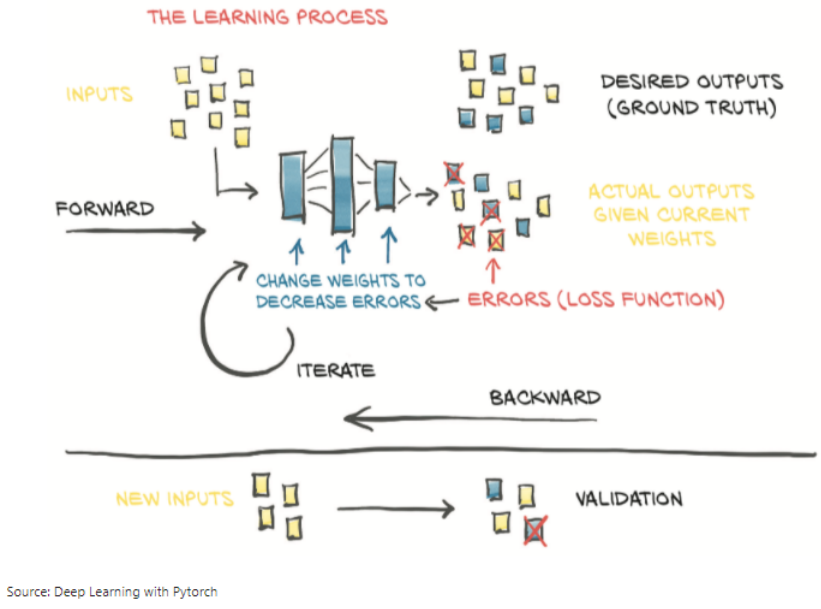

FIGURE 2.4 – Traning Phase scheme

Dans la phase d'apprentissage, la bonne classe pour chaque enregistrement est connue, et les nœuds de sortie peuvent donc se voir attribuer des valeurs relatives à chacune de nos deux classes Real et Fake. Il est ainsi possible de comparer les valeurs calculées du réseau pour les nœuds de sortie à la valeur correcte, et de calculer un terme d'erreur pour chaque nœud. Ces termes d'erreur sont ensuite utilisés pour ajuster les pondérations dans les couches masquées afin que, espérons-le, la prochaine fois, les valeurs de sortie soient plus proches dede la valeur correcte.

2.2.1 Architectures des modèles :

Après l'étape du Data preprocessing, notre nouvelle base de données maintenant contient 1888 colonnes qui represente les mots les plus frequents. Ainsi, ces dernières nous servira comme le input de la couche d'entrée de nos modèles. -couche d'entrée : composée de 1888 neurones, prend en entrée un vecteur representant un tweet.
-couches cachées : en fonction du modèle.
-couche de sortie : l'entrée depend du modèle, et elle a comme sortie 2 neurones representant les deux classes Fake ou Real.

2.2.1.1 Les optimisateurs :

Les optimiseurs sont des algorithmes ou des méthodes utilisées pour modifier les attributs du réseau de neurones tels que les poids et le taux d'apprentissage afin de réduire le taux d'erreur. On ce qui concerne notre projet, on va utiliser deux types d'optimisateurs, Adaptive Moment (Adamn) et SGD.
Adaptive Moment Estimation est un algorithme de technique d'optimisation pour la descente de gradient. La méthode est vraiment efficace lorsque vous travaillez avec un gros problème impliquant beaucoup de données ou de paramètres. Il nécessite moins de mémoire et est efficace. Intuitivement, il s'agit d'une combinaison de l'algorithme de "descente de gradient avec impulsion(momuntum)" et de l'algorithme "RMSP" [2].
La descente de gradient stochastique (souvent abrégée SGD) est une méthode itérative pour optimiser une fonction objectif avec des propriétés de lissage appropriées. Il peut être considéré comme une approximation stochastique de l'optimisation de descente de gradient, puisqu'il remplace le gradient réel (calculé à partir de l'ensemble des données) par une estimation de celui-ci (calculé à partir d'un sous-ensemble de données sélectionné au hasard). [3]

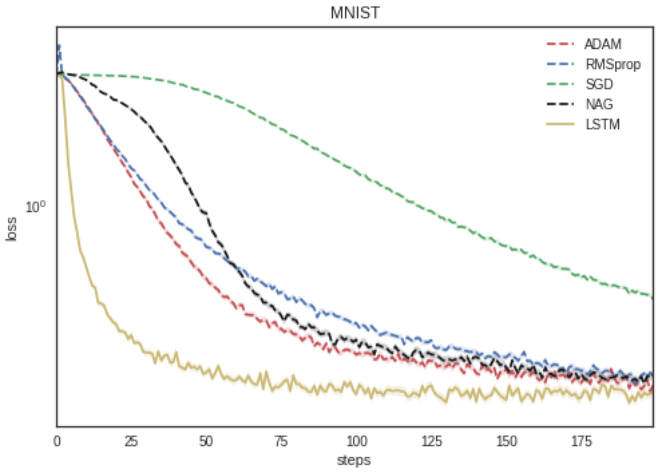

FIGURE 2.5 – Comparaison entre Adam et SGD et d'autre optimisateurs [4]

2.2.1.2 Les fonctions d'activation :

Une fonction d'activation décide si un neurone doit être activé ou non. Cela signifie qu'il décidera si l'entrée du neurone dans le réseau est importante ou non dans le processus de prédiction en utilisant des opérations mathématiques plus simples, notamment les deux fonctions qu'on va utliser Relu() et Sigmoid()

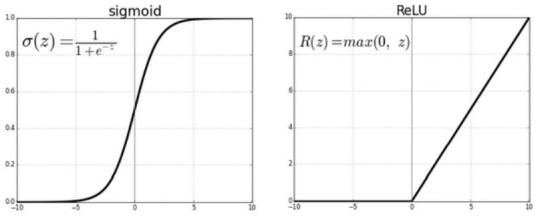

FIGURE 2.6 – Comparaison entre les fonctions d'activation Relu et Sigmoid

2.2.1.3 Fonction de perte (Loss Function) :

Les machines apprennent au moyen d'une fonction de perte. C'est une méthode pour évaluer dans quelle mesure un algorithme spécifique modélise les données données. Si les prévisions s'écartent trop des résultats réels, la fonction de perte en cracherait un très grand nombre.
La fonction Cross-Entropy a une large gamme de variantes, dont le type le plus courant est le Binary Cross-Entropy (BCE). La perte BCE est principalement utilisée pour les modèles de classification binaires ; c'est-à-dire des modèles n'ayant que 2 classes.

2.2.2 les structures des modèles :

Pour nos modèles, on va choisir des différents structures pour aboutir à la meilleure précision possible, en changeant différents paramètres comme le nombre de neurones dans chaque couche, les fonctions d'activation et les optimisateurs tout en appliquant le backpropagation.

Modèle	Nombre de couches cachées	neurones	Fonctions d'activation	Optimisateur	Learning Rate	Loss Function
1	2	128/64	relu/relu/segmoid	SGD	0.01	Cross Entropy
2	2	128/64	relu/relu/segmoid	Adam	0.01	Cross Entropy
3	2	128/64	relu/relu/segmoid	Adam	0.001	Cross Entropy
4	3	256/128/64	relu/relu/relu/segmoid	SGD	0.01	Cross Entropy
5	3	256/128/64	relu/relu/relu/segmoid	Adam	0.01	Cross Entropy
6	3	256/128/64	relu/relu/relu/segmoid	Adam	0.001	Cross Entropy
7	2	512/256	relu/relu/segmoid	SGD	0.01	Cross Entropy
8	2	512/256	relu/relu/segmoid	Adam	0.01	Cross Entropy
9	2	512/256	relu/relu/segmoid	Adam	0.001	Cross Entropy
10	2	128/64	segmoid/segmoid/segmoid	SGD	0.01	Cross Entropy
11	2	128/64	segmoid/segmoid/segmoid	Adam	0.01	Cross Entropy
12	2	128/64	segmoid/segmoid/segmoid	Adam	0.001	Cross Entropy

FIGURE 2.7 – Architecture des différents modèles

2.3 Conclusion :

Après avoir terminer la partie analyse et conception de notre projet qui englobe la méthodologie et différents structure des modèles, il est temps maintenant de vous présenter notre réalisation.

Chapitre 3

Realisation du projet :

Ce dernier chapitre contiendra les resultats obtenus de l'implémentation de la méthodologie et les différents étapes discutées dans le chapitres précédent, ainsi que l'interface utilisateur réalisée.

3.1 Technologies utilisées :

3.1.1 Python :

Python est un langage de programmation utilisé par de nombreux data scientists pour nettoyer les données, faire des visualisations et construire des modèles

FIGURE 3.1 – Logo Python

3.1.2 Pytorch :

PyTorch est une bibliothèque de tenseurs Deep Learning optimisée basée sur Python et Torch et est principalement utilisée pour les applications utilisant des GPU et des CPU. PyTorch est préféré aux autres frameworks d'apprentissage en profondeur comme TensorFlow et Keras car il utilise des graphiques de calcul dynamiques et est entièrement Pythonic.

FIGURE 3.2 – Logo Pytorch

3.2 Comparaison entre les différents modèles réalisés :

En se basant sur la précision obtenue de chaque modèle réalisé (tableau ci-dessous) on a constaté que le modèle ayant la meilleure précision et celui qu'on va construire notre application avec est le modèle numéro 2.

Modèle	Précision
1	53.19%
2	90.19%
3	78.12%
4	53.19%
5	89.72%
6	83.18%
7	53.19%
8	89.80%
9	88.79%
10	53.19%
11	88.55%
12	53.19%

FIGURE 3.3 – Comparaison entre les différents modèles au niveau de la précision

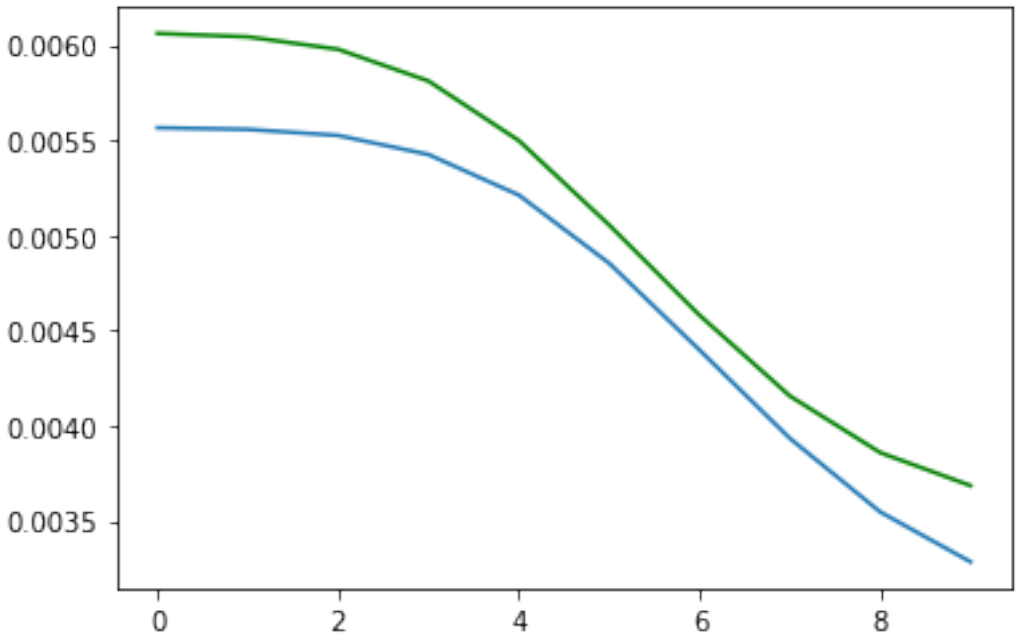

FIGURE 3.4 – taux des erreurs en fonction de nombre des epoches traités du modèle 2, de la validation (vert) et l'apprentissage (bleu)

3.3 Interface réalisée :

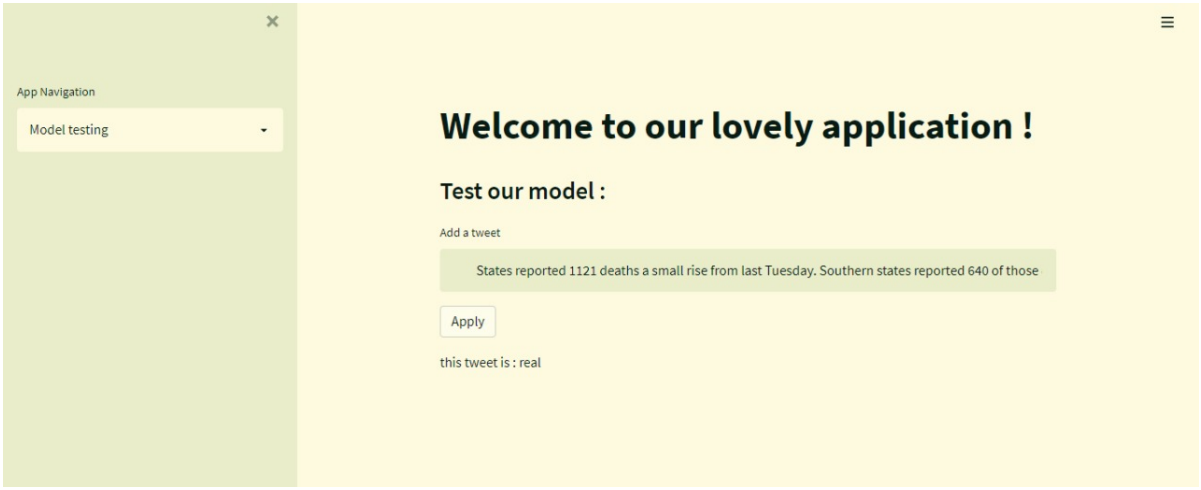

FIGURE 3.5 – interface de saisie de tweet pour la verification

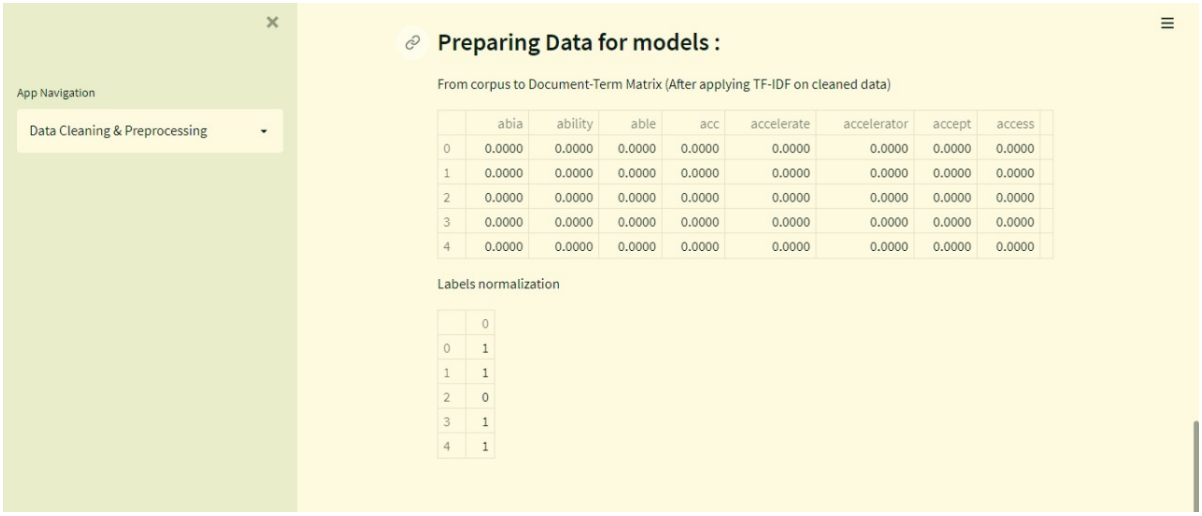

FIGURE 3.6 – interface de la prépration de donnée

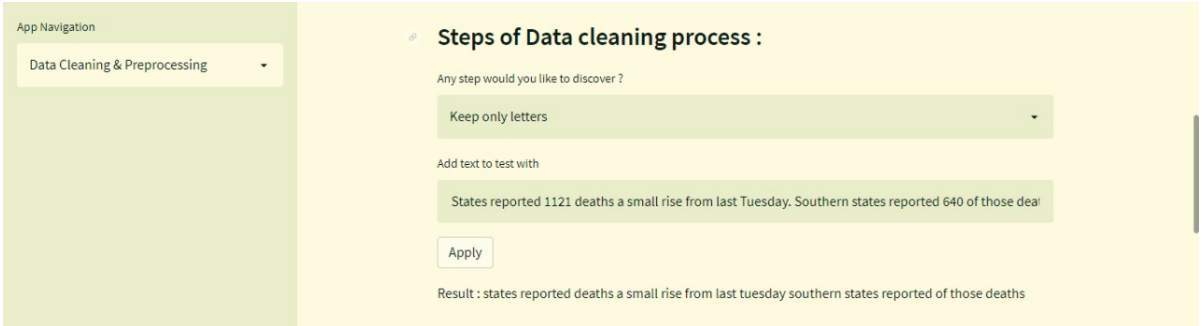

FIGURE 3.7 – interface pour l'essaye du Data Cleaning avec des paramètres customisées

Conclusion

Le traitement du langage naturel (PNL) est un sous-domaine de la linguistique, de l'informatique et de l'intelligence artificielle qui utilise des algorithmes pour interpréter et manipuler le langage humain.

Cette technologie est l'un des domaines les plus largement appliqués de l'apprentissage automatique et est essentielle pour analyser efficacement des quantités massives de données non structurées contenant beaucoup de texte. Dans notre projet titulé Fake Covid News Detection on Tweets, on a essayé de réaliser des modèles de réseau de neurones avec le backpropagation, des differentes architectures et algorithmes pour aboutir au modèle le plus fidèle possible, ayant la structure la plus convenable au problèmatique. Ainsi, une interface graphique avec un UI user-friendly a été créée pour faciliter l'utilisation de notre modèle de prédiction.

Bibliographie

https ://clevertap.com/blog/neural-networks/

https ://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/

https ://ruder.io/optimizing-gradient-descent/

https ://becominghuman.ai/paper-repro-learning-to-learn-by-gradient-descent-by-gradient-descent-6e504cc1c0de ?gi=cd235f196bfe