UNIVERSITÉ DU QUÉBEC EN OUTAOUAIS
Département de l'informatique

Projet synthèse
Rapport final

Par
Samuel Bernier
Vincent Patry

Travail de session présenté à
Omar Abdul Wahab

Dans le cadre du cours
INF4173-SO - Projet synthèse

22 avril 2022

Table des matières

Introduction du projet

Introduction

Le 21e siècle est aussi connu comme étant l'ère de l'information. Nous avons la chance de vivre dans une époque où toute personne ayant accès à un ordinateur et une connexion internet a accès à une source quasi illimitée d'information. Toute personne a la liberté de recueillir de l'information, de la partager, de la modifier, etc. Par conséquent, toute personne navigant sur le Web a la possibilité de l'utiliser afin de se renseigner sur un sujet avant d'établir son opinion.

Cependant, cette énorme quantité d'information n'est pas toujours exacte. Puisque l'information circule à grande échelle, il est plus facile que jamais de propager de la fausse information, que les intentions soient bonnes ou non. Par conséquent, nous devons faire face à un problème dont notre société n'a jamais eu à se préoccuper auparavant. Il est très facile pour une personne quelconque de lire un article scientifique, de mal interpréter les résultats et de partager une opinion déguisée en fait sur les réseaux sociaux. Cette situation à laquelle nous devons faire face cause plusieurs problèmes, notamment celui de diviser notre société en deux devant un conflit qui, quelques années auparavant, n'aurait jamais eu lieu.

Notre hypothèse est que la désinformation a un lien direct avec ce phénomène. C'est pourquoi nous avons décidé de créer un outil d'intelligence artificielle de détection de fausses nouvelles. Le but est d'analyser les articles et nouvelles à une grande échelle afin de vérifier s'il y a des corrélations entre le partage de fausses nouvelles et la réaction du public.

Problématique

Puisque nous devons faire face à une quantité massive d'informations, nous devons établir un outil intelligent capable d’analyser une grande quantité d'informations. Par conséquent, nous voulons être en mesure d’analyser automatiquement la syntaxe des nouvelles sur le Web afin de reconnaître la propagation de fausses nouvelles.

Objectifs

Pour répondre à cette problématique, notre logiciel va devoir répondre aux objectifs suivants :

Faire la comparaison d’information véridique, vérifiée d’une base de données avec les informations récoltées des textes sur le Web. Il sera donc question de trouver une base de données contenant des données vérifiées par des professionnels qui représentera une base de connaissances indispensable pour notre système, car l’algorithme de BERT nécessite un apprentissage supervisé pour le début de son apprentissage.

Se familiariser avec l’utilisation du modèle d’analyse BERT et les outils utilisés pour notre système (TensorFlow, Python notebook, etc.). Nous allons alors avoir besoin de faire des recherches sur l’algorithme afin de bien comprendre son fonctionnement et les différents aspects de ce dernier que nous allons avoir besoin de configurer dans le but de l’appliquer dans notre système pour répondre à notre objectif principal en lien avec la désinformation sur la Covid-19. Il sera alors question de comprendre le codage d’étiquettes, la tokenization, la conversion de texte en matrice, le rembourrage ainsi que la compilation du modèle avec les traitements initiaux des données et les paramètres d'entraînement.

Réussir à enregistrer les résultats dans le but de pouvoir permettre au logiciel de faire un apprentissage. Nous allons donc avoir besoin de bien comprendre le fonctionnement du prétraitement de nos données afin que l'algorithme de BERT puisse utiliser ces données prétraitées pour ensuite faire l'apprentissage du modèle et l'appliquer sur un plus grand ensemble de données qui n'auront pas été traitées dans le but qu'il puisse correctement classer les nouvelles dans la catégorie de "vraie" nouvelles ou de "fausses" nouvelles.

Être en mesure de pouvoir faire une analyse à la suite de ll'obtention de nos résultats finaux afin de pouvoir présenter une hypothèse répondant à notre problématique initiale. Nous allons donc devoir regarder si notre système a été efficace et si nous pouvons y apporter des recommandations pour l'améliorer.

Méthode scientifique utilisée

Dans le cadre de notre projet, nous allons utiliser la simulation numérique aussi connue sous le nom de simulation informatique. L'objectif de cette démarche est la suivante : faire l'identification du problème/phénomène à étudier pour ensuite faire une reproduction artificielle du phénomène grâce à un système informatique afin de démontrer les résultats obtenus, de les analyser et de finalement en tirer une conclusion.[28]

Description des outils utilisés

Python notebook (Google Colab & Jupyter)

Pour ce projet, nous allons faire l'utilisation de Google Colab. Cette plateforme nuagique interactive va nous donner un endroit où nous pourrons écrire notre code afin de pouvoir l'éditer, le commenter ainsi que l'exécuter tout en le partageant par la technologie de Google drive[25]. Il sera donc possible pour nous de travailler simultanément sur notre projet tout en étant certains que le code de ce dernier soit la version la plus à jour à chacune de ses exécutions[26]. En raison d'un manque de performance avec Google Colab pour l'exécution de notre système, nous avons opté pour notebook Jupyter, qui nous a permis d'avoir les mêmes fonctionnalités que sur notre notebook Google Colab, mais en ayant la performance d'un ordinateur que nous possédons afin de compiler localement notre système.

Utilisation du langage Python

Pour notre projet, nous avons décidé de le réaliser avec le langage Python. Nous avons effectué ce choix de langage de programmation à la suite de plusieurs lectures en ligne pour trouver quel langage était possiblement le mieux placé pour notre type de projet. Tout d'abord, nous avons trouvé que Python était un langage qui est beaucoup plus intuitif et ainsi plus concis à implémenter. Ceci dit, grâce à sa simplification, il est par conséquent plus évident de créer des algorithmes qui sont fiables sans mettre l'accent de notre projet sur l'apprentissage des nuances techniques que certains autres langages contiennent. De plus, nous avons découvert que Python était un langage riche en librairies tel que plusieurs qui sont utilisées lors de la création de systèmes en intelligence artificielle (Par exemple : Keras, TensorFlow et Scikit-learn). Pour notre projet, il semble y avoir une librairie avec TensorFlow qui nous permet de faire l'utilisation de différents modèles de BERT.[]

Librairie TensorFlow

Lors de la conception de notre programme, nous allons faire l'utilisation de la librairie TensorFlow. Cette librairie open source qui a été développée par Google nous donnera accès à plusieurs algorithmes d'apprentissage automatique de "deep learning" qui sont nécessaires dans la création de projets en IA [4]. Il est important aussi de préciser que TensorFlow supporte également l'utilisation de GPU [5]. Ceci nous permettra d'avoir une meilleure performance pour la compilation ainsi que de l'exécution de notre système. Cette librairie utilise des "Tensors" qui sont une représentation de vecteurs et matrices. Ces tensors nous permettent d'utiliser, de placer et de stocker de manière compacte [6]. Finalement, il est aussi important d'ajouter que la librairie nous permet d'avoir accès à un réseau de neurones rapidement sans la complexité du codage de celui-ci grâce à l'intégration d'API de haut niveau.En ce qui concerne le modèle utilisé pour l'entraînement de nos données, nous avons décidé d'opter pour un modèle multilinguisme, au cas où nous décidions d'analyser des nouvelles de diverses langues. Ce modèle fut entraîné sur plusieurs pages de Wikipédia contenant plusieurs langues et créé par : Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova[24].

Keras

Un autre outil, dont nous avons fait l'utilisation, est Keras. Keras est un API qui a été développé par Google afin de nous permettre de faire l'implémentation de réseau de neurones. Ce dernier a été programmé dans le langage Python. Ce qui est très intéressant avec Keras est qu'il peut facilement être utilisé en raison de son "frontend" en Python qui utilise un haut niveau d'abstraction tout en ayant l'option d'avoir plusieurs "back-ends" pour l'utilisation de calculs. Il est important de mentionner que ce dernier supporte aussi le framework de TensorFlow, qui utilise Keras comme son API de haut niveau officiel[29]. Ceci est donc très utile, car on peut faire la compilation de tensors, graph, session et autres en utilisant l'API principal de TensorFlow en même temps! Ceci nous offre donc une grande flexibilité lors du développement de notre application.

Pydot et GraphViz

Nous avons utilisé l'ensemble d'outils open source GraphViz développés par AT&T afin de pouvoir faire la création et l'assemblage de graphiques dans un langage DOT. Par la suite, nous avons fait l'utilisation de Pydot qui est une interface graphique de GraphViz qui peut "parse and dump" dans un langage "DOT" de fichier JSON. Cette interface nous a permis avec python de coder nos graphiques pour représenter certains tests ainsi que les graphes pour afficher nos statistiques en lien avec les résultats obtenus.

Algorithmes

Afin de bien expliquer le fonctionnement de BERT, nous devons tout d'abord commencer avec le fonctionnement des régressions logistiques, puis les réseaux de neurones.

Régression logistique

La régression logistique est un théorème populaire en intelligence artificielle, servant à faire de la classification de données basée sur certains modèles. Il s'avère populaire en analyse de données grâce à sa facilité d'implémentation et au besoin minime d'interaction humaine lors de la conception [11]. À des fins de synthèses, nous avons limité notre explication de cette méthode, puisque nous avons choisi un modèle différent.

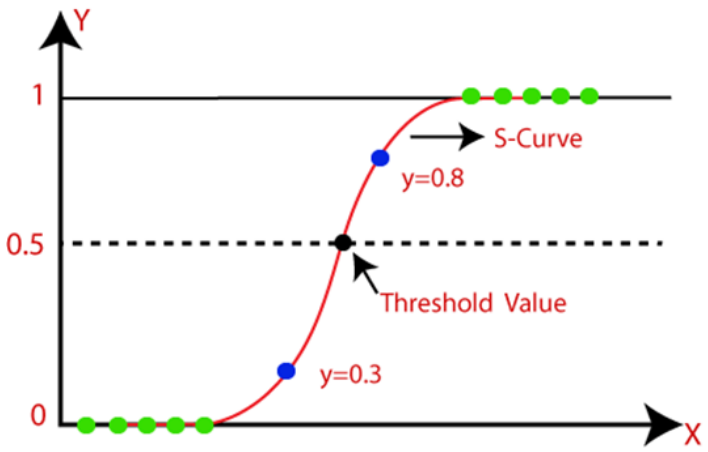
Tout d'abord, cette méthode combine chaque mot de notre base de données et leur associe une valeur, afin de les retrouver facilement. Par la suite, lors de l'analyse de notre base de données, chaque mot va être associé un poids, entre 0 et 1, déterminant si ce mot est associé avec des nouvelles véridiques ou avec des

fausses nouvelles. Une fois le modèle construit, nous allons associer les nouvelles données importées dans cette équation sigmoïde :

$$\log\left[\frac{y}{1-y}\right] = b_0 + b_1x_1 + b_2x_2 + b_3x_3 + \cdots + b_nx_n$$

Équation de régression logistique [10]

La valeur « y » représente la probabilité que notre texte soit véridique (la valeur sera donc égale à 1) ou incorrect (la valeur sera donc égale à 0), les variables « x » représentent les mots de notre texte, et les valeurs « b » représentent le poids de chaque mot. À l'aide de cette formule, nous obtiendrons un résultat entre 0 et 1, représentant si une nouvelle est véridique ou erronée. Notre fonction sigmoïde ressemblera à ceci :



Fonction sigmoïde de régression logistique [10]

Puisque cette technique s'avère très simpliste, nous avons décidé d'opter pour une solution plus complexe et précise.

Réseau de neurones

Les réseaux de neurones sont une autre solution très populaire. Ce type de modèle peut s'appliquer à une grande variété de problèmes et s'avère très efficace. Bien qu'elle soit similaire à la régression linéaire, elle est généralement plus précise,

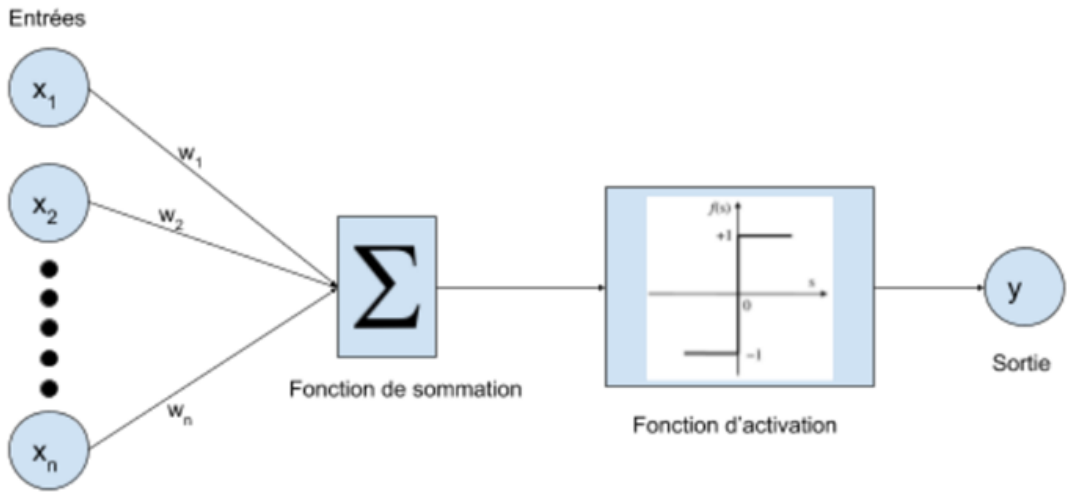
grâce au poids de chaque nœud qui s’ajuste automatiquement lors de la phase d’apprentissage.

Chaque réseau de neurones est, comme le nom l’indique, une combinaison de plusieurs neurones. Ceux-ci sont placés en plusieurs couches dépendamment du type de réseau de neurones que l’on veut utiliser. Les réseaux sont habituellement composés d’une couche d’entrée des données, d’une couche cachée et des données de sorties. Ces couches sont composées de neurones qui s’activent avec des poids. Lorsque le réseau est entraîné, le poids entre chaque neurone s’ajuste afin d’obtenir le résultat désiré. À l’intérieur de chaque neurone se trouve un perceptron. Un perceptron est tout d’abord composé d’un certain nombre d'entrées (input) allant de 1 à n. Ces entrées prennent une donnée, la transforment en valeur de -1 à 1 et l'acheminement vers le nœud à l’aide des poids. Le nœud consiste à une fonction de sommation accumulant chaque valeur des entrées multipliant celle des poids. Cette fonction de sommation est exprimée par la formule mathématique suivante:

$$\sum_{i=1}^{n} w_i * x_i + b$$

Auteur : Samuel Bernier

Dans cette formule, “w” représente le poids de chaque lien entre les entrées et le nœud, “x” représente chaque entrée et “b” représente le biais. Le résultat de cette fonction va donc se trouver entre -1 et 1. Ces résultats vont ensuite être envoyés à la fonction d’activation. Les résultats, selon les paramètres, peuvent ressembler à une fonction linéaire ou à une fonction sigmoïde. Par la suite, si le résultat se rapproche de -1, le neurone n’est pas activé. Si le résultat se rapproche de 1, il est activé. Voici un schéma représentant une perception:



Auteur : Samuel Bernier

C'est ainsi à l'aide de plusieurs neurones que l'on construit un réseau de neurones. Ensuite, il faut entraîner ce réseau de neurones. La méthode la plus efficace s'agit d'imposer des données d'entrée sachant le résultat de sortie que le réseau devrait nous donner. Si le résultat n'est pas le bon, les poids entre les données d'entrée et le nœud sont ajustés et le processus recommence. De cette façon, nous pouvons entraîner un réseau de neurones jusqu'au moment où le réseau donne le bon résultat. Le tout est fait de manière automatique. Une fois la fonction d'activation terminée, si le résultat n'est pas celui désiré, les poids sont ajustés et le calcul recommence. [9]

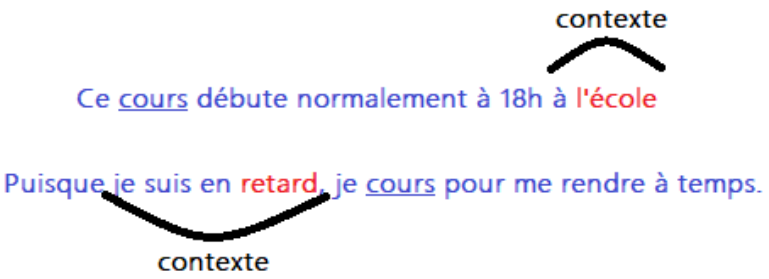
Bien que cette technique soit en temps normal très efficace, nous avons décidé de ne pas opter pour cette solution. L'algorithme BERT est spécialisé spécifiquement dans l'analyse de texte, tandis qu'un réseau de neurones est une façon générale de résoudre un problème.

Algorithme BERT

À la suite de plusieurs recherches, nous avons conclu que l'algorithme BERT qui a été conçu par une équipe chez Google serait l'algorithme parfait afin de pouvoir faire la classification de nos données. BERT (Bidirectional Encoder

Representation from Transformers) est un algorithme qui repose sur plusieurs concepts. Premièrement, BERT est basé sur une architecture de transformateur. Par la suite, il est pré entraîné avec plus de 3.3 millions de mots (Wikipedia et BookCorpus) [13]. Ceci permet à cet algorithme d'avoir une excellente compréhension du langage, ce qui le rend très performant dans l'analyse des mots et des phrases afin de comprendre le fonctionnement et le contexte du langage. Il est aussi important d'ajouter que BERT est bidirectionnel. Ceci veut dire qu'il est capable de faire l'apprentissage d'un langage de manière bidirectionnel. Ceci est extrêmement important dans le but de bien comprendre le contexte, par exemple d'un langage.

Par exemple :



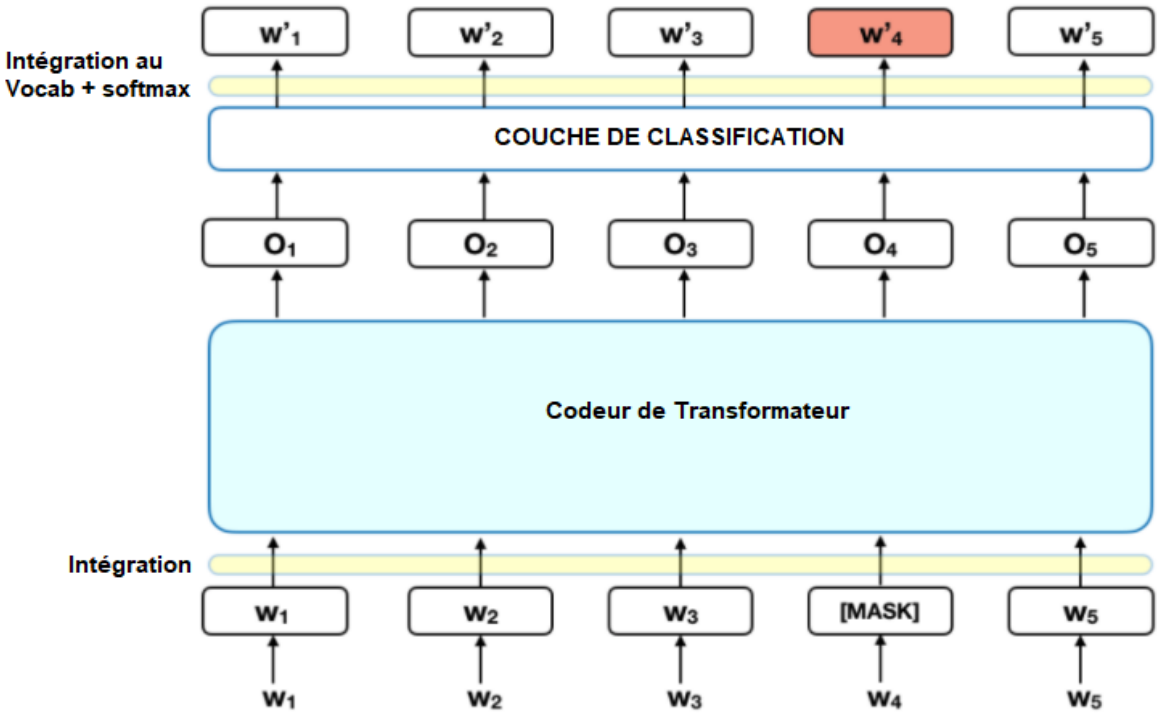
Auteur : Vincent Patry

On voit donc que le contexte de la phrase et surtout du mot "cours" peut varier selon son emplacement et surtout de l'emplacement de son contexte. Si nous prenons uniquement le contexte de gauche ou encore uniquement le contexte de droite alors nous risquons de faire une erreur dans l'une des deux phrases analysées.

Masked LM (MLM)

Débutons alors avec la première stratégie d'entraînement de données : Masked LM (MLM). Il est important de comprendre que durant l'application de cette

stratégie, près de 15% des mots de chacune des séquences se voient convertir en un masque qui est aussi nommé un token [19]. Une fois cette étape terminée, le modèle tente de faire la prédiction de la valeur originale du masque de ce mot en prenant en considération le contexte qui est offert par les autres mots qui ne sont pas masqués dans cette séquence. Pour être plus concret, on utilise les étapes suivantes : d'abord on fait l'ajout d'une couche de classification sur le dessus de la sortie du codeur. Ensuite, on multiplie le vecteur de sortie par la matrice d'intégration, permettant ainsi de faire la transformation de ceux-ci en dimension de vocabulaire. Finalement, on calcule la probabilité d'avoir chacun des mots dans le vocabulaire avec l'utilisation de "softmax"[19].

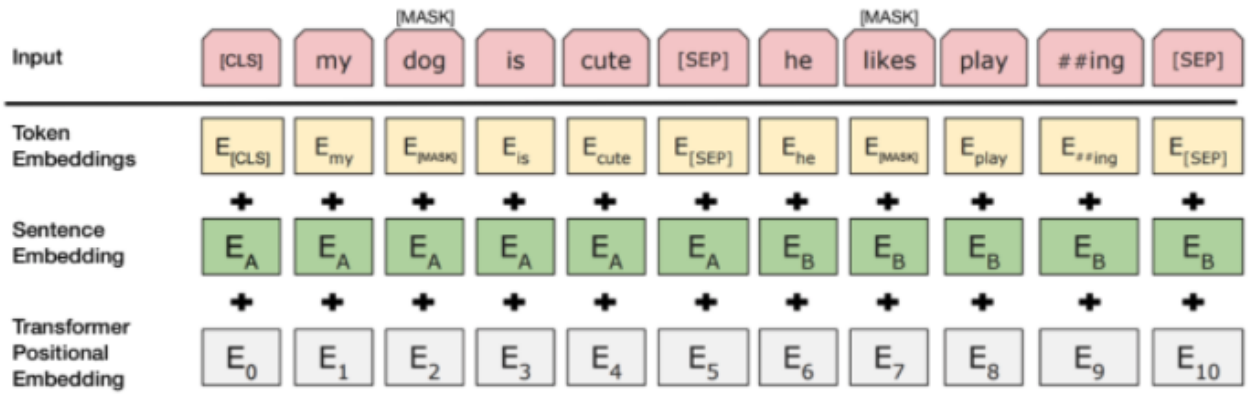


Processus Masked LM (MLM) [19]

Next Sentence Prediction (NSP)

Le Next Sentence Prédiction ou NSP est l'autre stratégie utilisée par BERT. Celle-ci permet à l'algorithme de pouvoir prendre deux phrases en entrée et d'ensuite prédire si la deuxième phrase est la phrase qui suit la première phrase dans le texte dans lequel elles ont été récupérées. Afin de réussir cette prédiction, on applique les étapes suivantes aux paires de phrases que l'on prend en entrée :

d'abord, on fait l'insertion d'un "cls token" au début de la première phrase ainsi qu'un "sep token" qui sera ajouté à la fin de chaque phrase subséquente. Ensuite, on fait l'intégration de phrases. À cette étape, on ajoute la phrase à chacun des tokens. Finalement, une "position embedding" est ajoutée à chaque TOKEN afin de déterminer la position de celui-ci dans la phrase [19]. Une fois que les entrées ont été traitées avec ces étapes pour les différencier, on passe ensuite à la partie qui s'occupe de faire la prédiction pour connaître si la deuxième phrase de la paire prise en entrée est effectivement connectée à la première phrase de cette paire. Pour ce faire, on passe la phrase à travers le modèle de transformation. Après cette étape, en utilisant la couche de classification, on fait la transformation de la sortie du "cls token" en un vecteur de 2 x 1. Finalement, on calcule la probabilité de "ItsNextSequence" avec "softmax"[19].



Représentation d'entrées BERT [20]

Utilisation de base de données

En ce qui concerne les données qui seront utilisées pour l'analyse faite par notre algorithme, nous utiliserons deux bases de données qui ont comme sujet la Covid-19. Ces deux bases de données ont été prises sur le site Web https://ieee-dataport.org. Il y a plusieurs raisons pour lesquelles nous avons fait le

choix de ce site afin de pouvoir y sélectionner des bases de données pour notre système. Tout d'abord, ce site Web est supporté par l'IEEE (L'Institut des ingénieurs électriciens et électroniciens) ce qui donne une certaine réputation de confiance pour les informations que l'on peut y trouver. De plus, nous avons été en mesure de voir les évaluations provenant d'autres utilisateurs afin de pouvoir trouver des bases de données de qualité qui sont déjà confirmées comme étant fiables et bien construites. Nous avons donc trouvé une première base de données déjà traitée, qui comprenait des informations vraies et fausses sur la Covid-19 vérifiée par : A. Saenz (Université de Wyoming), Sindhu Reddy Kalathur Gopal (Université de Wyoming) ainsi que Diksha Shukla (Université de Wyoming). Cette base de données a été supportée par le centre de justice sociale, McNairs scholars program, l'université de Wyoming et l'association CO-WY AMP. La première base de données offre 3796 fausses nouvelles manuellement vérifiées incluant la source, ainsi que 3794 nouvelles véridiques aussi manuellement vérifiées. Ce total de 7590 nouvelles manuellement vérifiées nous permet de sauver plusieurs semaines de recherche afin de faire ce traitement de données manuel nous-mêmes [2]. Ceci s'est avéré être essentiel, car l'algorithme de BERT nécessite un apprentissage supervisé avec des données prétraitées pour pouvoir faire par la suite la classification de nouvelles données. Ensuite, nous avons trouvé une autre base de données qui contenait 5,2 millions de titres de nouvelles, d'articles et de blogues sur la Covid-19[3]. Ces bases de données allaient donc être celles utilisées pour les tests effectués avec notre système, dans le but d'évaluer sa performance pour la classification.

Explication du système

Préparation des outils

Importation des librairies

Tout d'abord, afin de faire l'importation de tous les outils et composants nécessaires à la création de notre système, nous allons faire l'utilisation de plusieurs commandes en commençant par les commandes : !pip install tensorflow_hub ainsi

que de !pip install -q -U "tensorflow-text==2.8.*" afin de faire l'installation de TensorFlow. Par la suite, la commande !pip install keras tf-models-official pydot graphviz nous permettra de faire l'installation keras ainsi que pydot, graphviz. Finalement la commande : !pip install -q tf-models-official==2.7.0 pour faire l'installation des modèles offerts par TensorFlow. Une fois que tous les paquets ont été installés, nous pouvons importer toutes les composantes dont nous avons besoin. Nous avons aussi fait l'importation de : os, glob, numpy, pandas et évidemment tensorflow et tensorflow_hub.En ce qui concerne les données qui seront utilisées par notre système, nous avons pris la décision d'aller de l'avant en utilisant une source de données qui est constituée d'un ensemble de 3796 fausses nouvelles et 3794 vraies nouvelles. Les auteurs de cet ensemble de données sont Julio A. Saenz, Sindhu Reddy Kalathur Gopal, Diksha Shukla, June 12, 2021, "Covid-19 Fake News Infodemic Research base de données (CoVID19-FNIR base de données)", IEEE Dataport. [2]

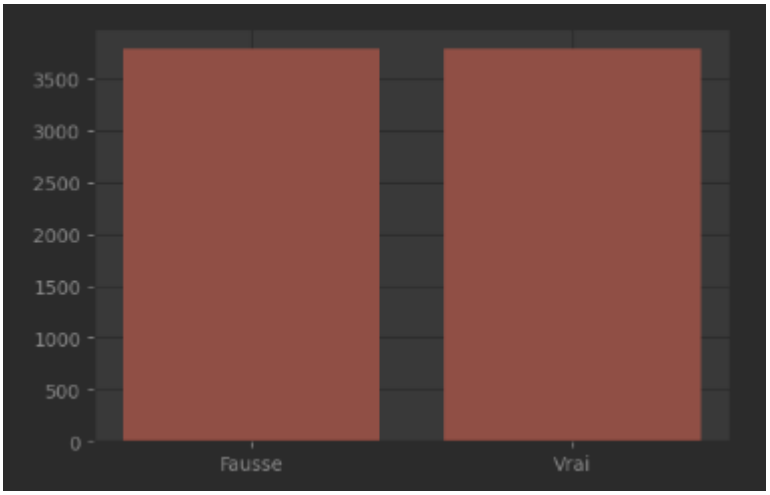
Vérification du matériel physique

Afin que notre programme s'exécute en un temps raisonnable, il est recommandé d'utiliser un GPU (Graphics processing unit) afin d'exécuter notre algorithme. Nous avons donc une section de code confirmant que notre programme s'exécute sur au moins 1 GPU. Dans le cas échéant, il serait irréaliste d'exécuter ce programme sur un CPU (Central processing unit), puisque la durée d'exécution sera excessivement longue. La raison est que CPU traditionnel possède environ 12 cœurs afin de faire les opérations. Cela veut donc dire qu'il peut exécuter 12 opérations simultanément. Cependant, un GPU possède des milliers de cœurs appelés des "CUDA Cores". Par exemple, la version gratuite de Google Colab utilise des GPU NVIDIA® Tesla® K80, possédant 4,991 CUDA Cores. Cela veut ainsi dire que nous allons être en mesure d'effectuer près de 5000 opérations simultanément, au lieu de 12. Bien évidemment, il y a plusieurs situations ou un CPU est beaucoup plus efficace qu'un GPU, mais dans notre situation ce ne serait pas réaliste d'utiliser un CPU [20]. Si votre ordinateur ne possède pas de GPU, il est recommandé d'utiliser Google Colab.

Transformation et vérification des bases de données

À la suite de l'importation de nos données, nous avons pu remarquer que celles-ci n'étaient pas uniformes. Après cette découverte, nous avons donc pris la décision de faire un prétraitement des données afin de rassembler celle-ci dans un même fichier de manière uniforme. Nous avons fait l'utilisation de ces commandes dans le but de nous assurer que les fichiers sont téléchargés si ceux-ci ne se trouvent pas déjà dans le répertoire "/content". Pour ce faire, nous avons rassemblé tous les fichiers dans un seul même fichier en utilisant la fonction ".concat" pour nous permettre de faire la concaténation de chaque fichier (fakeNew.csv ainsi que trueNews.csv) dans un dictionnaire. Ensuite, nous avons utilisé la fonction ".drop" afin d'enlever plusieurs colonnes dont les attributs ne nous intéressaient pas. Nous avons ainsi enlevé les colonnes suivantes : Date Posted', 'Link', 'Region', 'Country', 'Explanation', 'Origin', 'Origin_URL', 'Fact_checked_by', 'Poynter_Label', 'Username', 'Publisher'. À la prochaine étape de ce prétraitement de nos données, il a été question de créer une fonction "rem" afin de nous permettre d'identifier certaines choses à remplacer dans notre dictionnaire de données. Nous avons alors remplacé les éléments suivants : "http", "#", "@","\n", "pic.twitter.com" ainsi que "^\s" par un vide (" ") nous permettant ainsi de les retirer du dictionnaire. Finalement, afin de pouvoir faire la conversion de notre dictionnaire en fichier .csv, nous avons utilisé la commande suivante :df.to_csv("combined.csv", index=False, encoding='utf-8-sig'), qui le convertissait l'encodage en utf-08-sig, tout en utilisant False pour l'index afin de ne pas enregistrer l'indexage dans le fichier. Une fois le fichier créé, nous avons fait un test afin de vérifier que les données soient conformes. Nous avons tout d'abord affiché les données du fichier grâce aux commandes df.head et df.tail. Une fois que nous avions un affichage confirmant que les données étaient bel et bien compilées dans le fichier csv, nous avons utilisé la commande df.Label.unique() afin de vérifier que nous avons seulement 2 types d'arguments dans la catégorie "label" pour démontrer que les données étaient conformes. En temps normal, nous devions voir "0" et "1" en résultat ("0" est associé aux fausses nouvelles et "1" est associé aux vraies nouvelles). Finalement, nous avons utilisé une boucle for afin d'afficher la

répartition des types de données et s’assurer que les quantités de données vraies et fausses soient bien réparties.



Codage des étiquettes

Une fois que nos données sont prêtes, nous sommes prêts à créer les étiquettes. Ceux-ci vont simplement servir à séparer nos données en deux catégories. La première catégorie servira à l'entraînement de BERT. Elle sera constituée de 80% de nos données sélectionnées de façon aléatoire. Le 20% restant des données servira à tester notre algorithme face à des données respectant le même format. De cette façon, nous allons être en mesure d’obtenir la précision d'entraînement ainsi que la précision de validation à la fin de l'entraînement. BERT utilise cette méthode afin de confirmer la précision de ses prédictions face à des données jamais vues lors de l'entraînement. [21]

Ensuite, les données sont normalisées en utilisant un format acceptable par BERT. Les données d'entrées, écrites sous forme de « String », sont transformées en vecteur à l’aide de la méthode précédente, puis transformées en matrice à l’aide de la méthode « LabelEncoder ». [22]

Création des jetons (tokenization)

Continuons maintenant avec le modèle BERT. L'algorithme BERT peut être entraîné sur plusieurs types de données différents selon le besoin. Dans notre situation, nous avons décidé d'opter pour un modèle multilinguisme, au cas où nous décidons d'analyser des nouvelles de diverses langues. Ce modèle fut entraîné sur Wikipédia et les détails se trouvent ci-dessous. Tensorflow offre plusieurs modèles open source que vous êtes en mesure de retrouver ici:

Le modèle choisi a été entraîné par: Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova: "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," 2018. Ce dernier a été entraîné sur plusieurs pages de Wikipédia contenant plusieurs langues.[24]

Nous avons donc suivi les indications fournies par les auteurs quant à l'utilisation de ce modèle. Nous avons initié quelques variables. La première, "vocab_file" nous permet de lire le fichier vocab qui contient le vocabulaire du modèle que nous avons choisi. La deuxième variable, "do_lower_case" nous permet de déterminer si l'on doit transformer tous les textes en minuscule. Par défaut, ce paramètre devrait être "False". Finalement, la variable Tokenizer nous permet de pouvoir faire la création des jetons avec l'utilisation de la méthode suivante : FullTokenizer ainsi que de nos deux paramètres initiés plus tôt : "vocab_file" et "do_lower_case".

Il est important de mentionner que notre token a par la suite besoin de deux variables : classificateur ainsi que séparateur. Pour ce faire, nous avons fait l'utilisation de la fonction "convert_tokens_to_ids" avec les paramètres suivants : [CLS] et [SEP].

Une fois ces étapes terminées, notre texte sera transformé en matrice de nombre. Chaque nombre représente un mot de notre texte. Voici ce que BERT voit lorsqu'il analyse nos données :

```
<tf.Tensor: shape=(23,), dtype=int32, numpy=
array([10117, 13299, 14424, 13575, 16411, 25907, 80352, 10107, 10189,
       11155, 10301, 10798, 11084, 10233,   117, 10259, 50438, 10188,
       31206, 37715, 10251,   119,   102])>
```

Référence: python notebook du projet [21]

Une fois nos textes transformés en matrice de nombres, il sera plus facile d'analyser les tendances des fausses et vraies nouvelles. Voici à quoi ressemble notre texte original :

```
'The Chinese news website Tencent filters that there are more than 24,000 deaths from coronavirus.\t\t'
```

Référence: python notebook du projet [21]

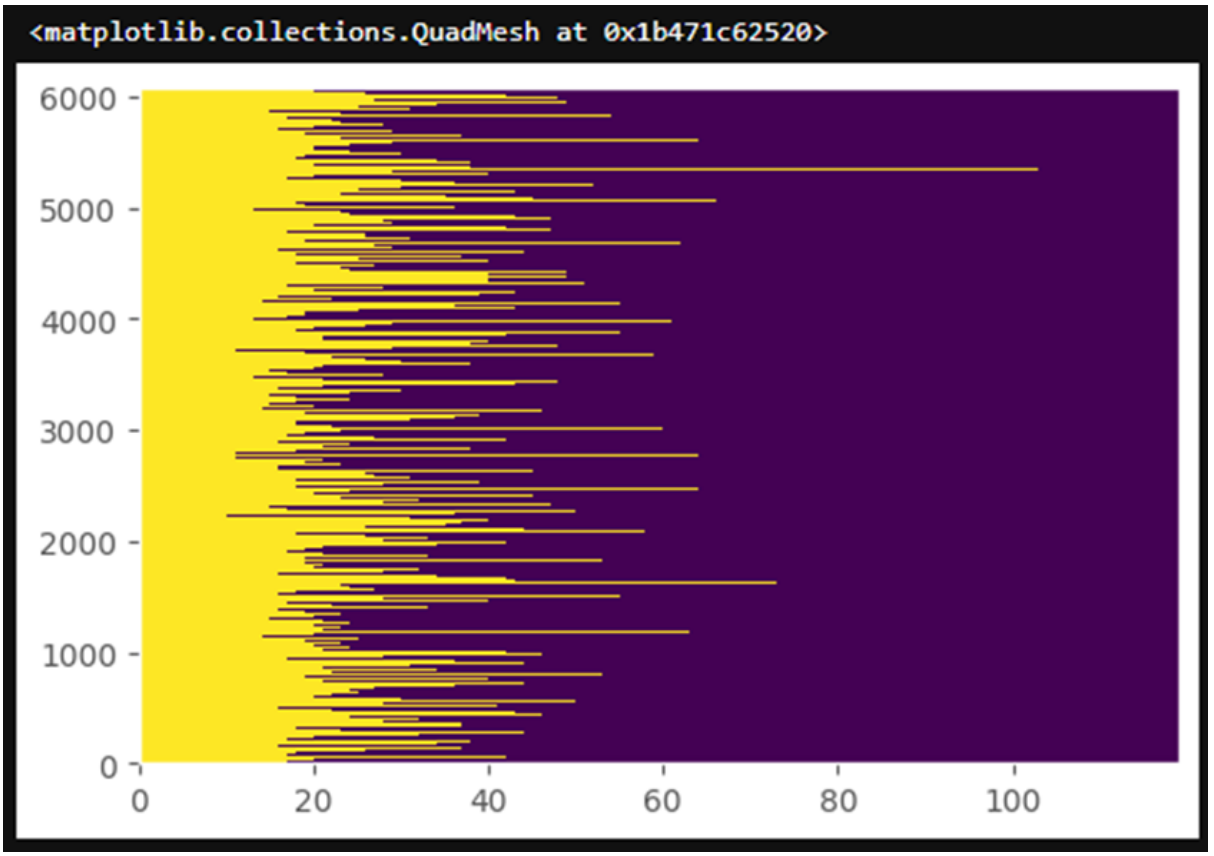
Voici à quoi ressemble chaque mot ainsi que son nombre associé :

```
The [10117]
Chinese [13299]
news [14424]
website [13575]
Ten [16411]
##cent [25907]
filter [80352]
##s [10107]
that [10189]
there [11155]
are [10301]
more [10798]
than [11084]
24 [10233]
, [117]
000 [10259]
deaths [50438]
from [10188]
corona [31206]
##vir [37715]
##us [10251]
. [119]
```

Référence: python notebook du projet [21]

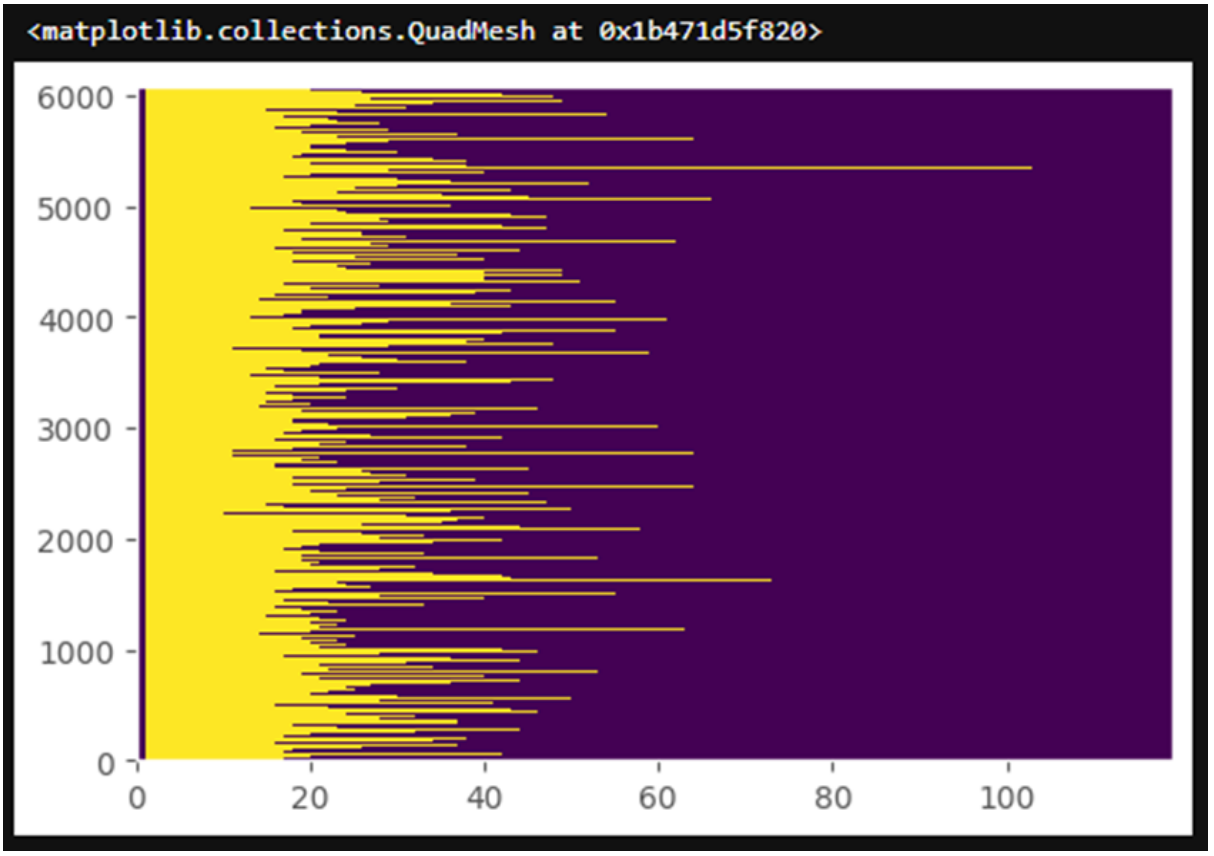
Masque et type

Le modèle que nous utilisons a besoin de deux données supplémentaires. Le masque et le type. Le masque sert à différencier le contenu d'entrée (les textes) et le rembourrage que nous avons créé. À l'aide de "tenso" et "pyplot", nous pouvons représenter visuellement ce que nous avons modifié. Voici le masque :



Référence: python notebook du projet [21]

Les lignes jaunes sont les données d'entrée et le mauve est le rembourrage. Ceci sert à normaliser la longueur de nos données. Nous devons ensuite ajouter un bloc vide [CLS] au début de chaque texte, afin d'insérer le jeton CLS (classification). Voici à quoi le type ressemble à la suite de l'ajout :



Référence: python notebook du projet [21]

Comme on peut le constater, une ligne mauve est apparue au début de chaque texte d'entrée. Il est donc plus facile de séparer chaque donnée d'entrée. Voici à quoi ressemblent maintenant nos matrices :

```
<tf.Tensor: shape=(6070, 119), dtype=int32, numpy=
array([[0, 1, 1, ..., 0, 0, 0],
       [0, 1, 1, ..., 0, 0, 0],
       [0, 1, 1, ..., 0, 0, 0],
       ...,
       [0, 1, 1, ..., 0, 0, 0],
       [0, 1, 1, ..., 0, 0, 0],
       [0, 1, 1, ..., 0, 0, 0]])>
```

Référence: python notebook du projet [21]

Le "0" représente le vide, les "1" représentent notre texte sous forme de nombre et les "…" servent à la réduction de la grandeur de notre matrice. Lors de l'analyse, des chiffres prendront la place de ces "…".

Création d'une fonction afin de simplifier les étapes

Dans le but de simplifier les étapes ultérieures, nous avons créé une fonction permettant d'utiliser notre modèle. Cette fonction reprend certains éléments définis plus haut à des fins de clarté.

Nous avons donc tout d'abord créé une longueur maximum de nos données d'entrées. Nous avons ensuite pris le texte d'entrée le plus long, puis l'avons multiplié par 1.5, au cas où un de nos textes d'entrée s'avérait plus long que nos données d'entrées initiales. Par la suite, nous avons créé les fonctions réutilisables suivantes : "def encode_names" et "def bert_encode", afin de rendre le prétraitement plus facile.

Modélisation

Traitement initial, initiation des paramètres pour le training, compilation du modèle

Nous pouvons donc maintenant commencer à modéliser. Il sera alors question de tout d'abord faire la création de notre modèle grâce à nos données d'entrées du modèle de BERT dont nous avons choisi ainsi qu'une couche de sortie selon notre nombre de classes. Nous allons alors trouver le nombre de classes dont nous avons besoin grâce à la variable "num_class" qui sera basée sur le nombre de classes utilisées. Nous allons aussi faire la création des masques ainsi que des identifiants (ID's) à partir du modèle multilinguisme choisi avec l'utilisation de couche TF keras afin de pouvoir cast des inputs de ce type en utilisant la méthode"tf.keras.layers.Input()".De plus, nous avons de plus utilisé la "sequence output"dans le but d'avoir la dimension de notre output. Nous avons aussi utilisé "pooled output" afin de faire la classification des textes . Par la suite, nous pouvons

désormais faire l'initiation des paramètres choisis pour notre entrainement. Nous allons prendre le paramètre “epochs” qui sera égal au nombre d'itérations qui seront effectués. Dans notre cas, nous allons lui attribuer la valeur de 3. Ceci nous permettra alors de faire l'entraînement de notre algorithme 3 fois. Le “batch_size” sera ajusté à 16 afin de permettre au GPU de google collab de bien performer.

Finalement, nous pouvons faire la compilation de notre modèle en utilisant la commande “model.compile()” ainsi que de voir un résumé de notre modèle en exécution grâce à la fonction “model.summary”.

Création des statistiques

Afin de créer nos statistiques, nous avons tout d'abord créé des variables globales dans le but de contenir les quantités de vraies et de fausses nouvelles pour chaque catégorie tels que le nombre de partages de sources vraies ou fausses, le nombre de mentions j'aime de sources vraies ou fausses, etc. Nous avons aussi créé des dictionnaires pour contenir et compter des occurrences par régions et par dates pour les vraies et fausses nouvelles. Une fois ces variables globales complétées, nous avons fait la création d'une méthode “def imp” qui prendrait en entrée un fichier JSON qui serait ensuite analysé ligne par ligne, pour voir si la nouvelle est vraie ou si cette dernière est fausse, l'insérant ensuite dans la variable “prediction_bool”. Il est important de préciser que notre fichier principal contenant toutes les données étant trop volumineux, nous avons séparé ce dernier en plusieurs petits fichiers auxquels on appliquait cette fonction. Dépendant du résultat de la variable “prediction_bool”, si la source se trouvait à être vraie, le code entrait ensuite dans la condition pour le calcule des variables globales des nouvelles qui sont vraies. Cependant, si la source était fausse, le code entrait ensuite dans la condition pour le calcul des variables globales des nouvelles qui sont fausses.

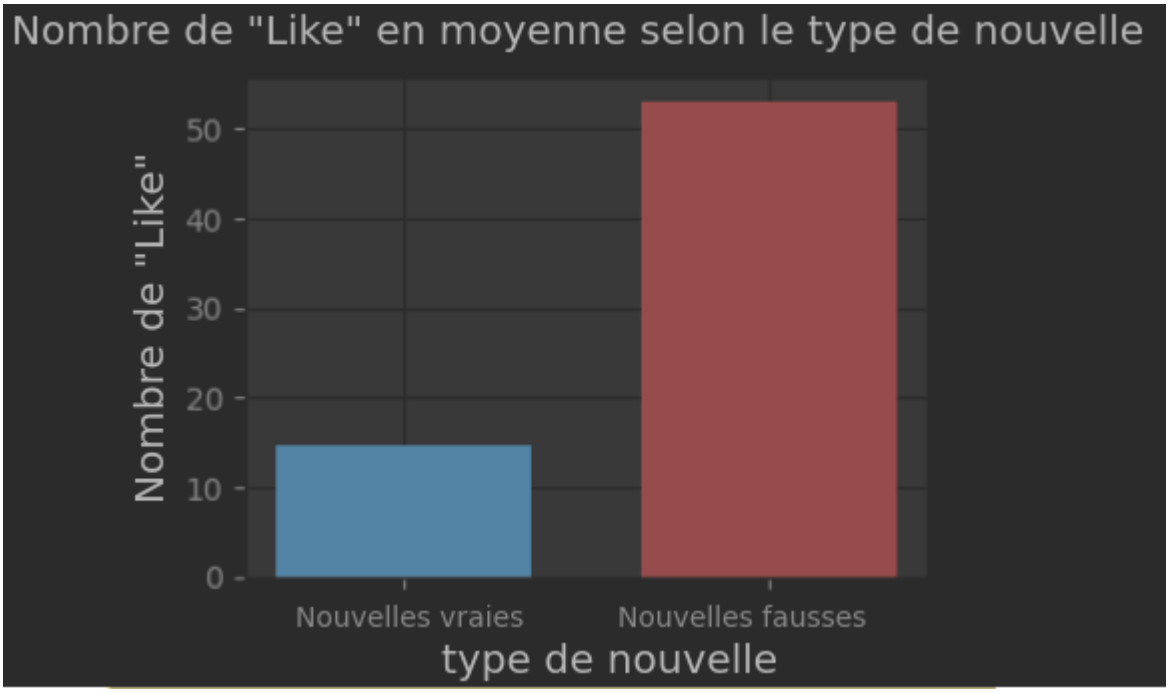
Présentation des résultats

Afin de vraiment tester notre système dans des conditions réelles, nous avons trouvé une base de données gratuite qui contient 5,2 millions d'articles/nouvelles/publications, qui ont comme thématique la Covid-19. Il est important par contre de mentionner qu'à la suite d'un prétraitement de données, ce montant a diminué. Certaines nouvelles ne comprenaient pas assez d'information pour pouvoir ensuite les utiliser lors du traitement par notre algorithme. L'idée serait de traiter cet ensemble de données dans notre système dans le but de voir si ce dernier est efficace à distinguer une vraie nouvelle d'une fausse nouvelle sur le sujet de la Covid-19 avec un grand ensemble de données variées. Il est important de préciser que puisque la base de données est très grande, soit une capacité de près de 15gb, nos ordinateurs n'étaient pas en mesure de faire le traitement ainsi que la sauvegarde de toutes ces données simultanément. Afin de répondre à cette problématique, nous avons fait la séparation de nos fichiers en 4 sources dans le but d'alléger la demande de RAM de nos ordinateurs.

Après avoir fait une compilation de notre modèle, nous avons pu constater que notre modèle choisi faisait l'utilisation de 177M de paramètres. C'est exactement pour cette raison que BERT est plus performant qu'un réseau de neurones traditionnel. Un réseau traditionnel aurait environ 10M de paramètres, ce qui est nettement inférieur à notre modèle. Par conséquent, notre modèle choisi devrait donc être plus précis. Après la compilation de notre système, nous avons pu faire une analyse de résultat très prometteuse! En effet, notre précision d'entraînement s'élevait à 99,92% et notre précision de validation à 98,29%. Il est de ce fait possible aussi de voir qu'après la deuxième itération, la précision de validation a considérablement augmenté, pour ensuite avoir une progression un peu plus faible pendant la troisième itération.

Statistique faite pour représenter la problématique de fausse nouvelle sur la Covid-19

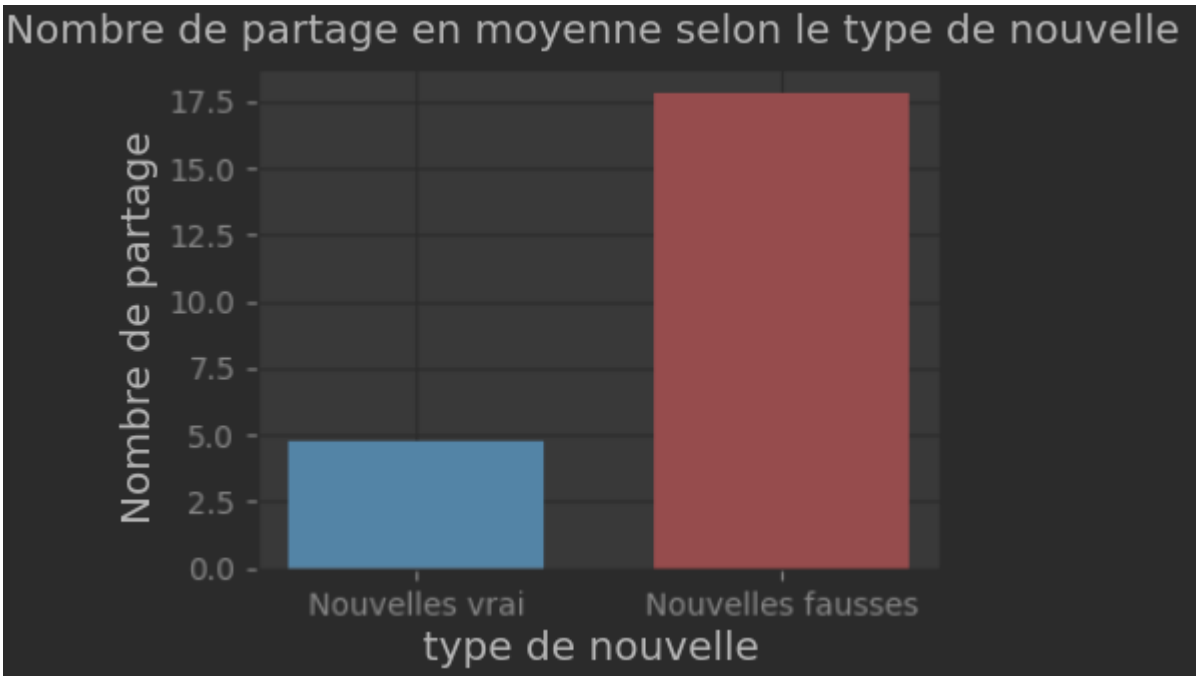
La désinformation est un sérieux problème et surtout de nos jours lorsque l'information est si facilement transmissible et accessible. Nous avons ainsi décidé de faire quelques fonctions qui nous permettraient d'avoir une meilleure compréhension sur la situation de la désinformation en lien avec la Covid-19. Pour ce faire, nous avons tout d'abord fait une fonction qui comptabilise la moyenne de mentions j'aime ainsi que de partages sur Facebook que les fausses nouvelles comparativement aux vraies nouvelles recevaient. Étonnamment, le nombre de mentions j'aime sur Facebook pour des nouvelles inexactes en moyenne était nettement plus élevé que pour de vraies nouvelles. En effet, il était presque 10 fois plus élevé!



Référence: python notebook du projet [21]

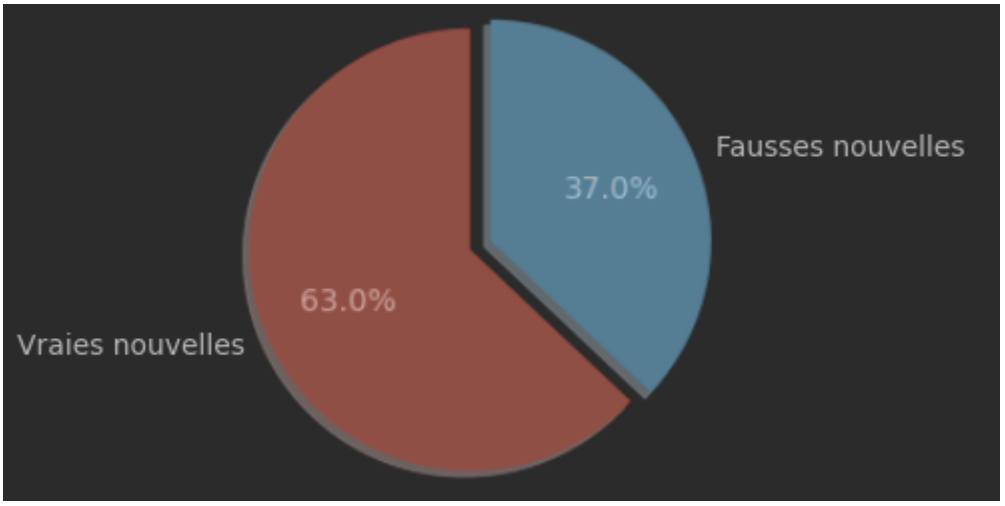
Nous avons aussi créé une fonction similaire nous permettant d'avoir le nombre moyen de partages sur Facebook qu'une fausse nouvelle et une vraie

nouvelle possédaient. Encore une fois, nos résultats démontrent que les fausses nouvelles reçoivent beaucoup plus de partages sur Facebook que les vraies nouvelles.



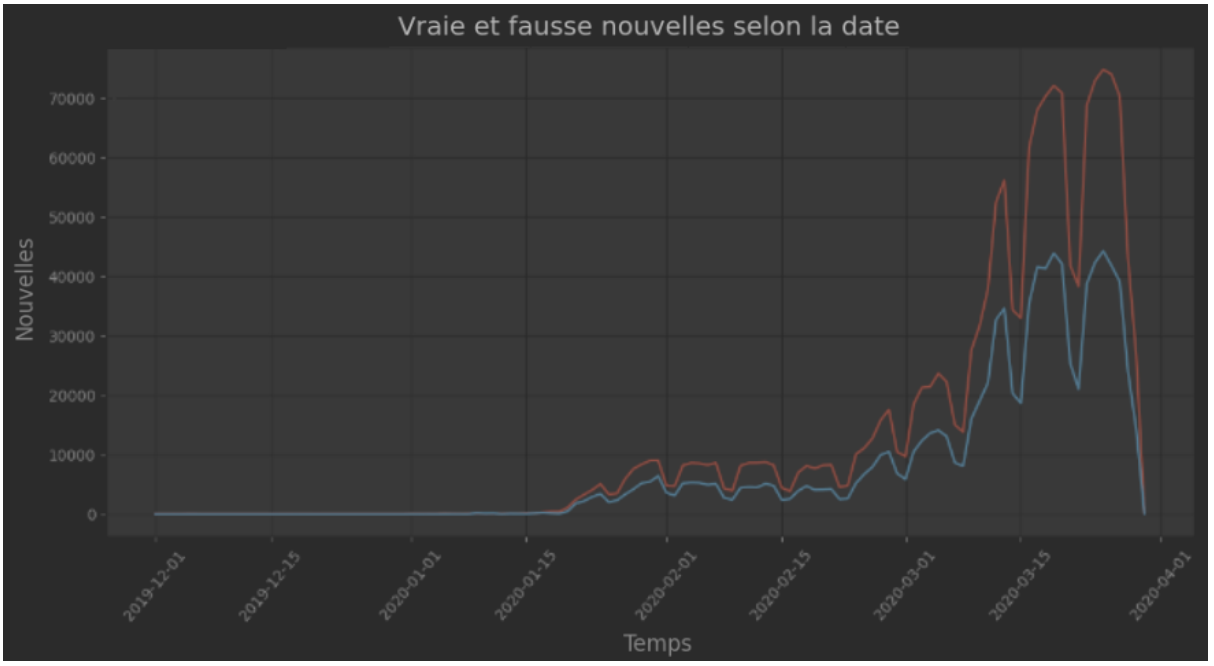
Référence: python notebook du projet [21]

Nous avons de plus créé une fonction nous affichant le nombre de vraies nouvelles ainsi que de fausses nouvelles au total :



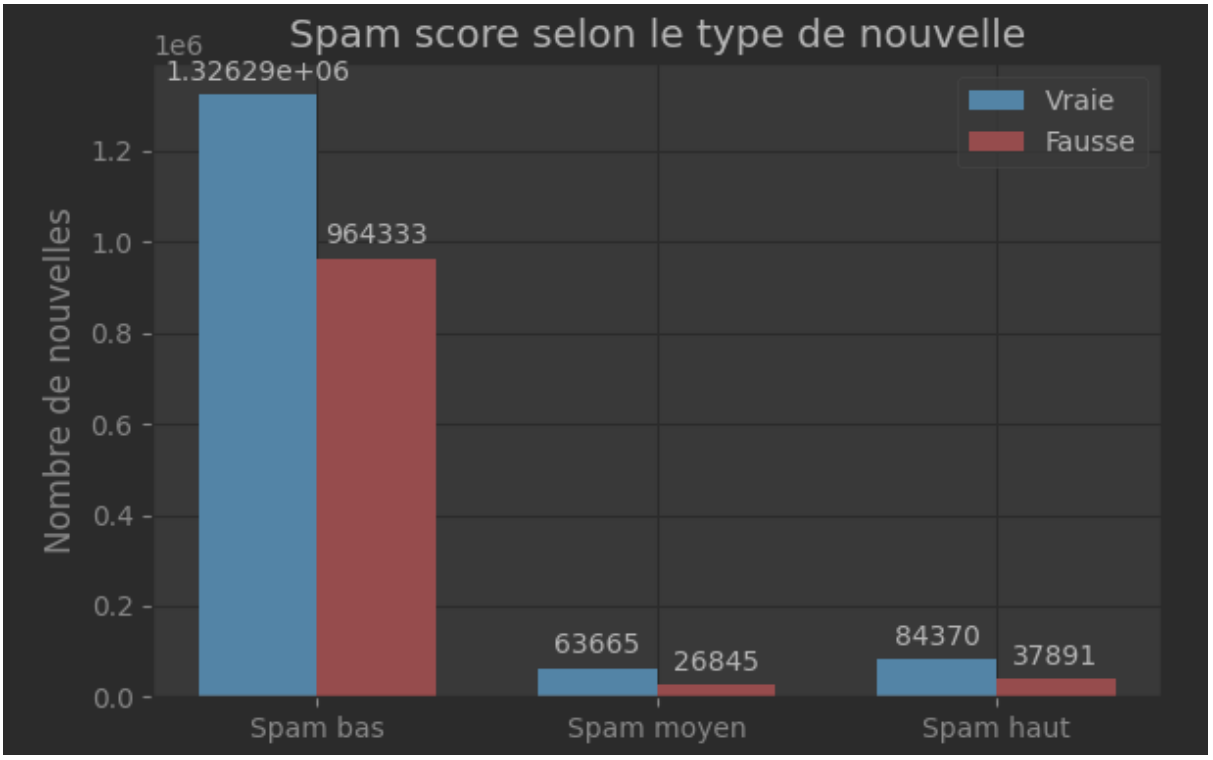
Référence: python notebook du projet [21]

Nous avons fait une fonction qui nous permettait de visualiser le nombre de vraies et fausses nouvelles par date de publication :



Référence: python notebook du projet [21]

Pour ce qui est du spam score, nous avons utilisé des conditions nous permettant de rassembler les vraies nouvelles ainsi que fausses sous trois sous-catégories pour le spam score : bas, moyen, haut.



Référence: python notebook du projet [21]

Analyse des résultats

Les résultats obtenus sont très clairs. En moyenne, les fausses nouvelles possèdent plus de mentions j'aime ainsi que de partages sur Facebook et ce même si elles sont moins nombreuses en quantité totale comparativement au total de vraies nouvelles en lien avec la Covid-19. Existe-t-il une explication pour un tel phénomène? Tout d'abord, selon une recherche effectuée par un groupe de chercheur de MIT, l'université de Regina, l'université d'Exeter Business School au Royaume-Uni et le centre de recherche et d'apprentissage en économie du Mexique Center for Research and Teaching in Economics in Mexico [30], le manque d'attention face à une nouvelle serait le facteur le plus important dans la propagation de fausses nouvelles sur les réseaux sociaux. Selon un des auteurs de la recherche, Gordon Pennycook, "Il semblerait que le contexte des réseaux sociaux distrait les gens de la précision des nouvelles qu'il partage". Cependant, cette recherche démontre que la plupart des participants trouvent une grande importance dans le partage de nouvelles véridiques, même si celles-ci sont opposées à leur façon de penser, ce qui contredirait l'idée que les personnes partagent plus souvent des informations qui sont en accord avec leur pensée. Mais alors qu'est-ce qui expliquerait un si grand écart entre le nombre de mentions j'aime et de partages en moyenne qu'une fausse nouvelle aurait comparativement à une vraie nouvelle ? Encore une fois, selon encore une fois cette même recherche, il paraît que la façon dont les réseaux sociaux tels que Facebook sont construits, favorise beaucoup le partage d'information qui fait réagir et qui amènent une réaction émotive, puisque celle-ci crée rapidement de l'interaction entre les utilisateurs. Il serait donc peut-être une bonne idée de trouver un moyen de récompenser les utilisateurs lorsqu'ils partagent ou aiment des nouvelles et du contenu qui a été vérifié et est exact. Pour ce qui est des résultats obtenus pour les dates, il est intéressant de voir que les quantités de fausses nouvelles et de vraies nouvelles augmentent pratiquement en même temps. D'après notre analyse, ce phénomène serait causé par la diffusion de nouvelles tournures d'événements en lien avec la pandémie ou encore la sortie de nouvelles informations par des gouvernements ou des organismes qui peuvent être mal comprises ou mal interprétées par des journalistes, blogueurs et autre, ce qui causerait des débats d'opinion ainsi que la rédaction d'articles qui peuvent

évidemment s'avérer inexactes. En ce qui concerne les résultats obtenus pour la statistique du spam score obtenu par de fausses nouvelles et vraies nouvelles, les résultats ont été très concluants. En effet, la plupart des sources (vraies et fausses) qui ont été utilisées se sont avérées avoir un spam score très bas. Ceci étant dit, on peut donc en déduire que les sources sont pour la grande majorité d'entre elles des sources qui ne seraient pas perçues comme "spammy" par Google. Le spam score fait la comptabilisation de différents "flag" qu'un site accumule. Chaque "flag" représente une probabilité qu'un site soit pénalisé par Google. Donc plus le spam score est élevé, plus la prédiction indiquant que le site est perçu comme "spammy" par Google est élevée. Ceci confirme alors que nos nouvelles utilisées, vraies ou fausses, sont de sites Web non reconnus pour être "spammy" par Google. Ceci donne ainsi une certaine validité aux données utilisées dans le cadre de notre projet.

(https://www.reputio.com/what-is-spam-score/)

Il est important de parler de l'enjeu que la désinformation peut avoir sur la vie des citoyens ainsi que de la gestion de la pandémie. Bien sûr, il est de la plus haute importance que les citoyens puissent avoir une relation de confiance avec les institutions publiques. Selon la Commission de l'éthique en science et en technologie (CEST), une étude qui portait sur la relation entre la vitesse de la propagation de fausses nouvelles et sur celle de la propagation d'un virus, a démontré que la diffusion de fausses nouvelles engendre une accélération dans la propagation de maladies infectieuses. Il serait donc important de lutter contre la désinformation faite sur le coronavirus afin de combattre la pandémie. La CEST ajoute que dans le but de vaincre la désinformation, il serait important de travailler sur deux aspects : premièrement, aider à la diffusion d'informations qui sont jugées de qualités et deuxièmement, éduquer les gens sur la désinformation qui est présente sur les réseaux sociaux. Bien évidemment, ce travail nécessiterait des efforts de tous les partis, c'est-à-dire des institutions et autorités publiques, des responsables de plateformes de médias sociaux et bien sûr, des citoyens. Mais que devrions-nous exiger de ceux-ci ? Tout d'abord, le gouvernement avait comme responsabilité d'assurer la mise en place d'une politique leur permettant de faire le retrait d'information qui serait d'origine de fausses nouvelles afin de veiller à la qualité de l'information. De plus, ce dernier devrait mettre en place des services éducatifs pour

éduquer la population et la sensibiliser à avoir un regard critique sur l'information qui se propage sur les réseaux sociaux. Par la suite, les responsables des plateformes de médias sociaux doivent aussi mettre en œuvre des politiques sur leur plateforme afin de retirer et condamner des comptes qui sont utilisés pour faire de la propagation de désinformation. Des politiques de la sorte ont déjà été mises en place par plusieurs grands joueurs dans ce domaine tels que Twitter qui a déjà banni le compte du président brésilien qui faisait la propagande d'un médicament qu'il ventait comme pouvant traiter la Covid-19 même si ce dernier ne faisait en aucun cas un consensus scientifique à ce sujet. Finalement, il ne peut y avoir de vraies améliorations si les citoyens ne mettent pas eux aussi la main à la pâte! Selon la CEST en 2018 "C'est, en quelque sorte, un devoir citoyen d'orienter nos interactions virtuelles de manière responsable."[31]. Ce devoir est donc réalisé par les citoyens en effectuant de bonnes recherches d'information sur le sujet, en ayant un regard critique sur les informations qu'ils voient sur les réseaux sociaux et bien sûr en ne partageant pas de fausses nouvelles.

Gestion de l'éthique

Bien entendu, lorsque l'on mentionne l'intelligence artificielle, il est impossible de ne pas se questionner sur le côté éthique de son implémentation. Il est important de préciser que dans le sujet englobant notre projet, on parle de détection de désinformation qui pourrait être dangereuse pour la santé des citoyens. Non seulement est-il important pour notre système de pouvoir détecter la désinformation, mais aussi de détecter correctement la bonne information afin de ne pas répandre par erreur de fausses nouvelles masquées comme étant véridiques par le système. Une question se pose alors. Est-ce que l'algorithme de BERT que nous avons utilisé pourrait comprendre les nuances à un point tel que son jugement éthique serait efficace ? Selon plusieurs sources, nous avons pu voir que BERT est en mesure de faire un raisonnement déontologiquement éthique en calculant un score de biais moral[32]. BERT semble donc être capable de faire une nuance plus qu'acceptable dans des questions de dilemme éthique. Ceci favoriserait alors son implémentation

dans un logiciel qui ferait la gestion de validation de sources d'information sur une pandémie mondiale. Dans le cas de notre algorithme par exemple, sa précision de 98% pour les tests nous a prouvé qu'il pouvait effectivement bien comprendre le contexte et l'essence du message afin de bien catégoriser l'information.

Conclusion

Pour conclure, le projet fut un réel succès. Nous avons tout d'abord été capables de faire l'implémentation de l'algorithme de BERT dans notre système dans le but de nous permettre de faire l'analyse d'une grande base de données sur la Covid-19 pour en tirer des résultats concluants sur notre hypothèse sur la désinformation sur la pandémie. Afin d'améliorer notre système, nous avons quelques recommandations. Il serait intéressant aussi de faire l'analyse sur une période de temps plus longue de la pandémie, donc sur une base de données qui contient des informations étalées sur 2 ans au lieu de 5 mois. Il serait aussi intéressant de trouver une façon de permettre d'aller chercher les sources directement de l'internet afin d'avoir une analyse en temps réel des informations qui sont partagées sur le sujet.

Bibliographie

[1] Gouvernement du Québec, (2020), Comment reconnaître une bonne source d'information en matière de santé, *Santé*, Obtenue à partir de: https://www.quebec.ca/sante/conseils-et-prevention/prevention-des-accidents-des-lesions-et-des-maladies/comment-reconnaitre-une-bonne-source-d-information-en-matiere-de-sante

[2] Julio A. Saenz, Sindhu Reddy Kalathur Gopal, Diksha Shukla, (12 juin 2021), Covid-19 Fake News Infodemic Research Dataset (CoVID19-FNIR Dataset), *IEEE Dataport*, Obtenue à partir de: https://dx.doi.org/10.21227/b5bt-5244

[3] Ran Geva, (7 avril 2020), FREE DATASET FROM NEWS/MESSAGE BOARDS/BLOGUES ABOUT CORONAVIRUS (4 MONTH OF DATA - 5.2M POSTS), *IEEE Dataport*, Obtenue à partir de: https://ieee-dataport.org/open-access/free-dataset-newsmessage-boardsblogs-about-coronavirus-4-month-data-52m-posts

[4] TensorFlow, (23 février 2022), Fine-tuning a BERT model, *TensorFlow*, Obtenue à partir de: https://www.tensorflow.org/text/tutorials/fine_tune_bert

[5] TensorFlow, (19 janvier 2022), Use a GPU, *TensorFlow*, Obtenue à partir de: https://www.tensorflow.org/guide/gpu

[6] SimpliLearn, (23 novembre 2021), What is Tensorflow: Deep Learning Libraries and Program Elements Explained, *AI & Machine learning*, Obtenue à partir de: https://www.simplilearn.com/tutorials/deep-learning-tutorial/what-is-tensorflow

[7] Google, (2 mars 2022), Welcome To Colaboratory, Obtenue à partir de: https://colab.research.google.com/?utm_source=scs-index

[8] SouvikMandal, (1 mai 2019), How to use Google Colab, *Machine learning*, Obtenue à partir de: https://www.geeksforgeeks.org/how-to-use-google-colab/

[9] Wikipédia, (22 février 2022), Neural network, Obtenue à partir de: https://en.wikipedia.org/wiki/Neural_network

[10] Javapoint, (2 mars 2022), Linear Regression vs Logistic Regression, *Machine learning*, Obtenue à partir de: https://www.javatpoint.com/linear-regression-vs-logistic-regression-in-machine-learning

[11] Wikipédia, (27 février 2022), Logistic regression, Obtenue à partir de: https://en.wikipedia.org/wiki/Logistic_regression#:~:text=Logistic%20regression%20is%20a%20statistical,a%20form%20of%20binary%20regression

[12] Daniel Jurafsky & James H. Martin, (29 décembre 2021), Speech and language processing, *Logistic regression*, chapitre 5, Obtenue à partir de: https://web.stanford.edu/~jurafsky/slp3/5.pdf

[13] mohdsanadzakirizvi@gmail.com, (25 septembre 2019), Demystifying BERT: A Comprehensive Guide to the Groundbreaking NLP Framework, Obtenue à partir de:

https://www.analyticsvidhya.com/blog/2019/09/demystifying-bert-groundbreaking-nlp-framework/

[14] Acervo Lima, (septembre 2021), Explication du modèle BERT – PNL, *Machine learning, Natural language processing*, Obtenue à partir de: https://fr.acervolima.com/explication-du-modele-bert-pnl/

[15] Rani Horev, (10 novembre 2018), BERT Explained: State of the art language model for NLP, *Towards Data Science*, Obtenue à partir de: https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270

[16] Jacob Devlin Ming-Wei Chang Kenton Lee Kristina Toutanova, (24 mai 2019), BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, Obtenue à partir de: https://arxiv.org/pdf/1810.04805.pdf

[17] CodeEmporium, (4 mai 2020), *BERT Neural Network - EXPLAINED!*, [Webdiffusion], Obtenue à par de: https://www.youtube.com/watch?v=xI0HHN5XKDo

[18] Codebasics, (septembre 2021), *What is BERT? | Deep Learning Tutorial 46 (Tensorflow, Keras & Python)*, [Webdiffusion], Obtenue à par de: https://www.youtube.com/watch?v=7kLi8u2dJz0&t=121s

[19] Rani Hore, (10 novembre 2018), BERT Explained: State of the art language model for NLP, *Towards Data Science*, Obtenue à partir de: https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270

[20] Datamadness, (27 octobre 2019), TensorFlow 2 - CPU vs GPU Performance Comparison, Obtenue à partir de: https://datamadness.github.io/TensorFlow2-CPU-vs-GPU

[21] Samuel Bernier et Vincent Patry,(22 avril 2022) , Analyse_de_fausses_nouvelles, Obtenue à partir de: https://github.com/BernierS/Analyse_de_fausses_nouvelles

[22] Scikit-learn (13 avril 2022), sklearn.model_selection.train_test_split, Obtenue à partir de: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html

[23] Scikit-learn (13 avril 2022), sklearn.preprocessing.LabelEncoder, Obtenue à partir de: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html

[24] TensorFlow, (13 avril 2022), bert_multi_cased_L-12_H-768_A-12, *Text embedding*, Obtenue à partir de: https://tfhub.dev/tensorflow/bert_multi_cased_L-12_H-768_A-12/2.

[25] Google, (2 mars 2022), Welcome To Colaboratory, Obtenue à partir de: https://colab.research.google.com/?utm_source=scs-index

[26] SouvikMandal, (1 mai 2019), How to use Google Colab, *Machine learning*, Obtenue à partir de: https://www.geeksforgeeks.org/how-to-use-google-colab/

[27] TensorFlowhub, (aucune date disponible), Bert Models, Obtenu à partir de: https://tfhub.dev/google/collections/bert/1

[28] Ec2modélisation, (aucune date disponible), Qu'est ce que la simulation numérique ?, Obtenu à partir de:
https://www.ec2-modelisation.fr/presentation/simulation-numerique

[29] Simplilearn, (Sep 18, 2021), What Is Keras: The Best Introductory Guide To Keras, Obtenu à partir de:
(https://www.simplilearn.com/tutorials/deep-learning-tutorial/what-is-keras)

[30] Gordon Pennycook, Ziv Epstein, Mohsen Mosleh, (17 March 2021), Shifting attention to accuracy can reduce misinformation online, Obtenu à partir de:
(https://www.nature.com/articles/s41586-021-03344-2#author-information)

[31] CEST, (10 avril 2020), Enjeux éthiques des fausses informations sur la COVID-19, Obtenu à partir de:
https://www.ethique.gouv.qc.ca/fr/actualites/ethique-hebdo/eh-2020-04-10/

[32] Patrick Schramowski, Cigdem Turan, Sophie Jentzsch, Constantin Rothkopf et Kristian Kersting, (11 Dec 2019), BERT has a Moral Compass: Improvements of ethical and moral values of machines, Obtenu à partir de:
(https://arxiv.org/pdf/1912.05238.pdf)

Annexe

Les pages suivantes offrent la totalité du python notebook en format PDF. Vous pouvez obtenir le notre code original sur Github: https://github.com/BernierS/Analyse_de_fausses_nouvelles

Introduction

Ce projet est présenté par Samuel Bernier et Vincent Patry dans le cadre du projet synthèse du baccalauréat en informatique à l'Université du Québec en Outaouais (UQO). L'objectif de ce projet est d'implémenter l'algorithme BERT à des fins de détections de fausses nouvelles en rapport à la Covid-19. Ce présent document sert de démontrastion de notre implémentation et a pour but d'offrir suffisamment de détails afin que quiconque soit en mesure d'utiliser notre code. Les détails de notre rapport final, incluant toutes les sources utilisées, se trouve à l'adresse suivante: https://github.com/BernierS/Analyse_de_fausses_nouvelles

Préparation des outils

La première étape est de confirmer que nous avons tous les outils nécessaires. Aux fins de se projet, nous recommandons d'utiliser python 3.9.7. Au moment de la rédaction de ce projet, certains outils utilisés n'étaient pas compatibles avec python 10.

In [ ]:
```
!python --version
```

```
Python 3.9.7
```

Maintenant que nous avons vérifié la version de Python, nous devons installer plusieurs outils:

1. tensorflow_hub
2. Keras
3. tf-models-official
4. pydot
5. graphviz

Afin d'obtenir plus de détails sur ces technologies, nous vous invitons à consulter le rapport: https://github.com/BernierS/Analyse_de_fausses_nouvelles

In [ ]:
```
!pip install tensorflow_hub
!pip install keras tf-models-official pydot graphviz

!pip install -q -U tensorflow-text
!pip install -q tf-models-official==2.4.0
```

```
Requirement already satisfied: tensorflow_hub in c:\users\samue\anaconda3\envs\gpu7\lib
\site-packages (0.12.0)
Requirement already satisfied: protobuf>=3.8.0 in c:\users\samue\anaconda3\envs\gpu7\lib
\site-packages (from tensorflow_hub) (3.19.4)
Requirement already satisfied: numpy>=1.12.0 in c:\users\samue\anaconda3\envs\gpu7\lib\s
ite-packages (from tensorflow_hub) (1.21.5)
Requirement already satisfied: keras in c:\users\samue\anaconda3\envs\gpu7\lib\site-pack
ages (2.8.0)
Requirement already satisfied: tf-models-official in c:\users\samue\anaconda3\envs\gpu7
\lib\site-packages (2.4.0)
Requirement already satisfied: pydot in c:\users\samue\appdata\roaming\python\python39\s
```

```
ite-packages (1.4.2)
Requirement already satisfied: graphviz in c:\users\samue\appdata\roaming\python\python3
9\site-packages (0.19.1)
Requirement already satisfied: pyyaml>=5.1 in c:\users\samue\appdata\roaming\python\pyth
on39\site-packages (from tf-models-official) (5.4.1)
Requirement already satisfied: scipy>=0.19.1 in c:\users\samue\anaconda3\envs\gpu7\lib\s
ite-packages (from tf-models-official) (1.7.3)
Requirement already satisfied: tensorflow>=2.4.0 in c:\users\samue\appdata\roaming\pytho
n\python39\site-packages (from tf-models-official) (2.8.0)
Requirement already satisfied: oauth2client in c:\users\samue\appdata\roaming\python\pyt
hon39\site-packages (from tf-models-official) (4.1.3)
Requirement already satisfied: google-cloud-bigquery>=0.31.0 in c:\users\samue\anaconda3
\envs\gpu7\lib\site-packages (from tf-models-official) (2.34.2)
Requirement already satisfied: tensorflow-hub>=0.6.0 in c:\users\samue\anaconda3\envs\gp
u7\lib\site-packages (from tf-models-official) (0.12.0)
Requirement already satisfied: opencv-python-headless in c:\users\samue\appdata\roaming
\python\python39\site-packages (from tf-models-official) (4.5.5.64)
Requirement already satisfied: py-cpuinfo>=3.3.0 in c:\users\samue\appdata\roaming\pytho
n\python39\site-packages (from tf-models-official) (8.0.0)
Requirement already satisfied: psutil>=5.4.3 in c:\users\samue\appdata\roaming\python\py
thon39\site-packages (from tf-models-official) (5.9.0)
Requirement already satisfied: tensorflow-model-optimization>=0.4.1 in c:\users\samue\ap
pdata\roaming\python\python39\site-packages (from tf-models-official) (0.7.1)
Requirement already satisfied: tf-slim>=1.1.0 in c:\users\samue\appdata\roaming\python\p
ython39\site-packages (from tf-models-official) (1.1.0)
Requirement already satisfied: pycocotools in c:\users\samue\appdata\roaming\python\pyth
on39\site-packages (from tf-models-official) (2.0.4)
Requirement already satisfied: pandas>=0.22.0 in c:\users\samue\anaconda3\envs\gpu7\lib
\site-packages (from tf-models-official) (1.4.1)
Requirement already satisfied: tensorflow-addons in c:\users\samue\appdata\roaming\pytho
n\python39\site-packages (from tf-models-official) (0.16.1)
Requirement already satisfied: Cython in c:\users\samue\appdata\roaming\python\python39
\site-packages (from tf-models-official) (0.29.28)
Requirement already satisfied: six in c:\users\samue\anaconda3\envs\gpu7\lib\site-packag
es (from tf-models-official) (1.16.0)
Requirement already satisfied: matplotlib in c:\users\samue\appdata\roaming\python\pytho
n39\site-packages (from tf-models-official) (3.5.1)
Requirement already satisfied: gin-config in c:\users\samue\anaconda3\envs\gpu7\lib\site
-packages (from tf-models-official) (0.5.0)
Requirement already satisfied: kaggle>=1.3.9 in c:\users\samue\appdata\roaming\python\py
thon39\site-packages (from tf-models-official) (1.5.12)
Requirement already satisfied: seqeval in c:\users\samue\appdata\roaming\python\python39
\site-packages (from tf-models-official) (1.2.2)
Requirement already satisfied: google-api-python-client>=1.6.7 in c:\users\samue\appdata
\roaming\python\python39\site-packages (from tf-models-official) (2.41.0)
Requirement already satisfied: numpy>=1.15.4 in c:\users\samue\anaconda3\envs\gpu7\lib\s
ite-packages (from tf-models-official) (1.21.5)
Requirement already satisfied: Pillow in c:\users\samue\anaconda3\envs\gpu7\lib\site-pac
kages (from tf-models-official) (9.0.1)
Requirement already satisfied: tensorflow-datasets in c:\users\samue\appdata\roaming\pyt
hon\python39\site-packages (from tf-models-official) (4.5.2)
Requirement already satisfied: dataclasses in c:\users\samue\anaconda3\envs\gpu7\lib\sit
e-packages (from tf-models-official) (0.6)
Requirement already satisfied: sentencepiece in c:\users\samue\appdata\roaming\python\py
thon39\site-packages (from tf-models-official) (0.1.96)
Requirement already satisfied: pyparsing>=2.1.4 in c:\users\samue\anaconda3\envs\gpu7\li
b\site-packages (from pydot) (3.0.4)
Requirement already satisfied: google-auth-httplib2>=0.1.0 in c:\users\samue\appdata\roa
ming\python\python39\site-packages (from google-api-python-client>=1.6.7->tf-models-offi
cial) (0.1.0)
```

```
Requirement already satisfied: httplib2<1dev,>=0.15.0 in c:\users\samue\anaconda3\envs\g
pu7\lib\site-packages (from google-api-python-client>=1.6.7->tf-models-official) (0.20.
4)
Requirement already satisfied: uritemplate<5,>=3.0.1 in c:\users\samue\anaconda3\envs\gp
u7\lib\site-packages (from google-api-python-client>=1.6.7->tf-models-official) (4.1.1)
Requirement already satisfied: google-auth<3.0.0dev,>=1.16.0 in c:\users\samue\anaconda3
\envs\gpu7\lib\site-packages (from google-api-python-client>=1.6.7->tf-models-official)
(2.6.0)
Requirement already satisfied: google-api-core!=2.0.*,!=2.1.*,!=2.2.*,!=2.3.0,<3.0.0dev,
>=1.31.5 in c:\users\samue\appdata\roaming\python\python39\site-packages (from google-ap
i-python-client>=1.6.7->tf-models-official) (2.7.1)
Requirement already satisfied: googleapis-common-protos<2.0dev,>=1.52.0 in c:\users\samu
e\anaconda3\envs\gpu7\lib\site-packages (from google-api-core!=2.0.*,!=2.1.*,!=2.2.*,!=
2.3.0,<3.0.0dev,>=1.31.5->google-api-python-client>=1.6.7->tf-models-official) (1.56.0)
Requirement already satisfied: protobuf>=3.12.0 in c:\users\samue\anaconda3\envs\gpu7\li
b\site-packages (from google-api-core!=2.0.*,!=2.1.*,!=2.2.*,!=2.3.0,<3.0.0dev,>=1.31.5-
>google-api-python-client>=1.6.7->tf-models-official) (3.19.4)
Requirement already satisfied: requests<3.0.0dev,>=2.18.0 in c:\users\samue\anaconda3\en
vs\gpu7\lib\site-packages (from google-api-core!=2.0.*,!=2.1.*,!=2.2.*,!=2.3.0,<3.0.0de
v,>=1.31.5->google-api-python-client>=1.6.7->tf-models-official) (2.27.1)
Requirement already satisfied: cachetools<6.0,>=2.0.0 in c:\users\samue\anaconda3\envs\g
pu7\lib\site-packages (from google-auth<3.0.0dev,>=1.16.0->google-api-python-client>=1.
6.7->tf-models-official) (4.2.2)
Requirement already satisfied: pyasn1-modules>=0.2.1 in c:\users\samue\anaconda3\envs\gp
u7\lib\site-packages (from google-auth<3.0.0dev,>=1.16.0->google-api-python-client>=1.6.
7->tf-models-official) (0.2.8)
Requirement already satisfied: rsa<5,>=3.1.4 in c:\users\samue\anaconda3\envs\gpu7\lib\s
ite-packages (from google-auth<3.0.0dev,>=1.16.0->google-api-python-client>=1.6.7->tf-mo
dels-official) (4.7.2)
Requirement already satisfied: grpcio<2.0dev,>=1.38.1 in c:\users\samue\anaconda3\envs\g
pu7\lib\site-packages (from google-cloud-bigquery>=0.31.0->tf-models-official) (1.44.0)
Requirement already satisfied: packaging>=14.3 in c:\users\samue\anaconda3\envs\gpu7\lib
\site-packages (from google-cloud-bigquery>=0.31.0->tf-models-official) (21.3)
Requirement already satisfied: google-cloud-core<3.0.0dev,>=1.4.1 in c:\users\samue\anac
onda3\envs\gpu7\lib\site-packages (from google-cloud-bigquery>=0.31.0->tf-models-officia
l) (2.2.3)
Requirement already satisfied: proto-plus>=1.15.0 in c:\users\samue\anaconda3\envs\gpu7
\lib\site-packages (from google-cloud-bigquery>=0.31.0->tf-models-official) (1.20.3)
Requirement already satisfied: python-dateutil<3.0dev,>=2.7.2 in c:\users\samue\anaconda
3\envs\gpu7\lib\site-packages (from google-cloud-bigquery>=0.31.0->tf-models-official)
(2.8.2)
Requirement already satisfied: google-resumable-media<3.0dev,>=0.6.0 in c:\users\samue\a
naconda3\envs\gpu7\lib\site-packages (from google-cloud-bigquery>=0.31.0->tf-models-offi
cial) (2.3.2)
Requirement already satisfied: grpcio-status<2.0dev,>=1.33.2 in c:\users\samue\anaconda3
\envs\gpu7\lib\site-packages (from google-api-core!=2.0.*,!=2.1.*,!=2.2.*,!=2.3.0,<3.0.0
dev,>=1.31.5->google-api-python-client>=1.6.7->tf-models-official) (1.44.0)
Requirement already satisfied: google-crc32c<2.0dev,>=1.0 in c:\users\samue\anaconda3\en
vs\gpu7\lib\site-packages (from google-resumable-media<3.0dev,>=0.6.0->google-cloud-bigq
uery>=0.31.0->tf-models-official) (1.3.0)
Requirement already satisfied: tqdm in c:\users\samue\anaconda3\envs\gpu7\lib\site-packa
ges (from kaggle>=1.3.9->tf-models-official) (4.63.0)
Requirement already satisfied: urllib3 in c:\users\samue\anaconda3\envs\gpu7\lib\site-pa
ckages (from kaggle>=1.3.9->tf-models-official) (1.26.8)
Requirement already satisfied: python-slugify in c:\users\samue\appdata\roaming\python\p
ython39\site-packages (from kaggle>=1.3.9->tf-models-official) (6.1.1)
Requirement already satisfied: certifi in c:\users\samue\anaconda3\envs\gpu7\lib\site-pa
ckages (from kaggle>=1.3.9->tf-models-official) (2021.10.8)
Requirement already satisfied: pytz>=2020.1 in c:\users\samue\anaconda3\envs\gpu7\lib\si
te-packages (from pandas>=0.22.0->tf-models-official) (2021.3)
```

```
Requirement already satisfied: pyasn1<0.5.0,>=0.4.6 in c:\users\samue\anaconda3\envs\gpu
7\lib\site-packages (from pyasn1-modules>=0.2.1->google-auth<3.0.0dev,>=1.16.0->google-a
pi-python-client>=1.6.7->tf-models-official) (0.4.8)
Requirement already satisfied: charset-normalizer~=2.0.0 in c:\users\samue\anaconda3\env
s\gpu7\lib\site-packages (from requests<3.0.0dev,>=2.18.0->google-api-core!=2.0.*,!=2.1.
*,!=2.2.*,!=2.3.0,<3.0.0dev,>=1.31.5->google-api-python-client>=1.6.7->tf-models-officia
l) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in c:\users\samue\anaconda3\envs\gpu7\lib\si
te-packages (from requests<3.0.0dev,>=2.18.0->google-api-core!=2.0.*,!=2.1.*,!=2.2.*,!=
2.3.0,<3.0.0dev,>=1.31.5->google-api-python-client>=1.6.7->tf-models-official) (3.3)
Requirement already satisfied: libclang>=9.0.1 in c:\users\samue\anaconda3\envs\gpu7\lib
\site-packages (from tensorflow>=2.4.0->tf-models-official) (13.0.0)
Requirement already satisfied: h5py>=2.9.0 in c:\users\samue\anaconda3\envs\gpu7\lib\sit
e-packages (from tensorflow>=2.4.0->tf-models-official) (3.6.0)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in c:\users\samue\an
aconda3\envs\gpu7\lib\site-packages (from tensorflow>=2.4.0->tf-models-official) (0.24.
0)
Requirement already satisfied: keras-preprocessing>=1.1.1 in c:\users\samue\anaconda3\en
vs\gpu7\lib\site-packages (from tensorflow>=2.4.0->tf-models-official) (1.1.2)
Requirement already satisfied: termcolor>=1.1.0 in c:\users\samue\anaconda3\envs\gpu7\li
b\site-packages (from tensorflow>=2.4.0->tf-models-official) (1.1.0)
Requirement already satisfied: flatbuffers>=1.12 in c:\users\samue\anaconda3\envs\gpu7\l
ib\site-packages (from tensorflow>=2.4.0->tf-models-official) (20210226132247)
Requirement already satisfied: setuptools in c:\users\samue\anaconda3\envs\gpu7\lib\site
-packages (from tensorflow>=2.4.0->tf-models-official) (58.0.4)
Requirement already satisfied: tensorboard<2.9,>=2.8 in c:\users\samue\anaconda3\envs\gp
u7\lib\site-packages (from tensorflow>=2.4.0->tf-models-official) (2.8.0)
Requirement already satisfied: google-pasta>=0.1.1 in c:\users\samue\anaconda3\envs\gpu7
\lib\site-packages (from tensorflow>=2.4.0->tf-models-official) (0.2.0)
Requirement already satisfied: tf-estimator-nightly==2.8.0.dev2021122109 in c:\users\sam
ue\anaconda3\envs\gpu7\lib\site-packages (from tensorflow>=2.4.0->tf-models-official)
(2.8.0.dev2021122109)
Requirement already satisfied: typing-extensions>=3.6.6 in c:\users\samue\anaconda3\envs
\gpu7\lib\site-packages (from tensorflow>=2.4.0->tf-models-official) (3.10.0.2)
Requirement already satisfied: opt-einsum>=2.3.2 in c:\users\samue\anaconda3\envs\gpu7\l
ib\site-packages (from tensorflow>=2.4.0->tf-models-official) (3.3.0)
Requirement already satisfied: astunparse>=1.6.0 in c:\users\samue\anaconda3\envs\gpu7\l
ib\site-packages (from tensorflow>=2.4.0->tf-models-official) (1.6.3)
Requirement already satisfied: absl-py>=0.4.0 in c:\users\samue\anaconda3\envs\gpu7\lib
\site-packages (from tensorflow>=2.4.0->tf-models-official) (0.15.0)
Requirement already satisfied: wrapt>=1.11.0 in c:\users\samue\anaconda3\envs\gpu7\lib\s
ite-packages (from tensorflow>=2.4.0->tf-models-official) (1.13.3)
Requirement already satisfied: gast>=0.2.1 in c:\users\samue\anaconda3\envs\gpu7\lib\sit
e-packages (from tensorflow>=2.4.0->tf-models-official) (0.4.0)
Requirement already satisfied: wheel<1.0,>=0.23.0 in c:\users\samue\anaconda3\envs\gpu7
\lib\site-packages (from astunparse>=1.6.0->tensorflow>=2.4.0->tf-models-official) (0.3
5.1)
Requirement already satisfied: markdown>=2.6.8 in c:\users\samue\anaconda3\envs\gpu7\lib
\site-packages (from tensorboard<2.9,>=2.8->tensorflow>=2.4.0->tf-models-official) (3.3.
4)
Requirement already satisfied: werkzeug>=0.11.15 in c:\users\samue\anaconda3\envs\gpu7\l
ib\site-packages (from tensorboard<2.9,>=2.8->tensorflow>=2.4.0->tf-models-official) (2.
0.3)
Requirement already satisfied: tensorboard-data-server<0.7.0,>=0.6.0 in c:\users\samue\a
naconda3\envs\gpu7\lib\site-packages (from tensorboard<2.9,>=2.8->tensorflow>=2.4.0->tf-
models-official) (0.6.0)
Requirement already satisfied: tensorboard-plugin-wit>=1.6.0 in c:\users\samue\anaconda3
\envs\gpu7\lib\site-packages (from tensorboard<2.9,>=2.8->tensorflow>=2.4.0->tf-models-o
fficial) (1.6.0)
Requirement already satisfied: google-auth-oauthlib<0.5,>=0.4.1 in c:\users\samue\anacon
```

```
da3\envs\gpu7\lib\site-packages (from tensorboard<2.9,>=2.8->tensorflow>=2.4.0->tf-model
s-official) (0.4.1)
Requirement already satisfied: requests-oauthlib>=0.7.0 in c:\users\samue\anaconda3\envs
\gpu7\lib\site-packages (from google-auth-oauthlib<0.5,>=0.4.1->tensorboard<2.9,>=2.8->t
ensorflow>=2.4.0->tf-models-official) (1.3.0)
Requirement already satisfied: oauthlib>=3.0.0 in c:\users\samue\anaconda3\envs\gpu7\lib
\site-packages (from requests-oauthlib>=0.7.0->google-auth-oauthlib<0.5,>=0.4.1->tensorb
oard<2.9,>=2.8->tensorflow>=2.4.0->tf-models-official) (3.2.0)
Requirement already satisfied: dm-tree~=0.1.1 in c:\users\samue\appdata\roaming\python\p
ython39\site-packages (from tensorflow-model-optimization>=0.4.1->tf-models-official)
(0.1.6)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\samue\anaconda3\envs\gpu7\l
ib\site-packages (from matplotlib->tf-models-official) (1.4.0)
Requirement already satisfied: cycler>=0.10 in c:\users\samue\anaconda3\envs\gpu7\lib\si
te-packages (from matplotlib->tf-models-official) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\samue\anaconda3\envs\gpu7\l
ib\site-packages (from matplotlib->tf-models-official) (4.31.1)
Requirement already satisfied: text-unidecode>=1.3 in c:\users\samue\anaconda3\envs\gpu7
\lib\site-packages (from python-slugify->kaggle>=1.3.9->tf-models-official) (1.3)
Requirement already satisfied: scikit-learn>=0.21.3 in c:\users\samue\appdata\roaming\py
thon\python39\site-packages (from seqeval->tf-models-official) (1.0.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\samue\anaconda3\envs\gpu
7\lib\site-packages (from scikit-learn>=0.21.3->seqeval->tf-models-official) (3.1.0)
Requirement already satisfied: joblib>=0.11 in c:\users\samue\anaconda3\envs\gpu7\lib\si
te-packages (from scikit-learn>=0.21.3->seqeval->tf-models-official) (1.1.0)
Requirement already satisfied: typeguard>=2.7 in c:\users\samue\anaconda3\envs\gpu7\lib
\site-packages (from tensorflow-addons->tf-models-official) (2.13.3)
Requirement already satisfied: dill in c:\users\samue\appdata\roaming\python\python39\si
te-packages (from tensorflow-datasets->tf-models-official) (0.3.4)
Requirement already satisfied: promise in c:\users\samue\appdata\roaming\python\python39
\site-packages (from tensorflow-datasets->tf-models-official) (2.3)
Requirement already satisfied: tensorflow-metadata in c:\users\samue\anaconda3\envs\gpu7
\lib\site-packages (from tensorflow-datasets->tf-models-official) (1.7.0)
Requirement already satisfied: colorama in c:\users\samue\anaconda3\envs\gpu7\lib\site-p
ackages (from tqdm->kaggle>=1.3.9->tf-models-official) (0.4.4)
```

Importation des librairies

Une fois que tous les paquets ont été installés, nous pouvons importer toutes les composantes dont nous avons besoin. Les librairies importées ici vont être utilisées tout au long du programme.

In [ ]:

```
import os
import glob

import numpy as np
import pandas as pd
import re

import tensorflow as tf
import tensorflow_hub as hub

from keras.utils import np_utils

from official.modeling import tf_utils
from official import nlp
from official.nlp import bert

import official.nlp.bert.bert_models
```

```
import official.nlp.bert.configs
import official.nlp.bert.run_classifier
import official.nlp.bert.tokenization as tokenization

from official.modeling import tf_utils
from official import nlp
from official.nlp import bert

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder

import matplotlib.pyplot as plt
```

Vérification du matériel physique

Afin que notre programme s'exécute en un temps raisonnable, il est recommandé d'utiliser un GPU afin d'exécuter notre algorithme. La section de code suivante vérifie que le programme s'exécute bien sur au moins 1 GPU. Dans le cas échéant, il serait irréaliste d'exécuter ce programme sur un CPU, puisque la durée d'exécution sera excessivement longue. Si votre ordinateur ne possède pas de GPU, il est recommandé d'utiliser Google Colab.

Le code est fourni directement par Tensorflow et peut se trouver ici:
https://www.tensorflow.org/guide/gpu

In [ ]:
```
gpus = tf.config.experimental.list_physical_devices('GPU')
if gpus:
  try:
    # Currently, memory growth needs to be the same across GPUs
    for gpu in gpus:
      tf.config.experimental.set_memory_growth(gpu, True)
    logical_gpus = tf.config.experimental.list_logical_devices('GPU')
    print(len(gpus), "Physical GPUs,", len(logical_gpus), "Logical GPUs")
  except RuntimeError as e:
    # Memory growth must be set before GPUs have been initialized
    print(e)

print("Version: ", tf.__version__)
print("Eager mode: ", tf.executing_eagerly())
print("Hub version: ", hub.__version__)
print("GPU is", "available" if tf.config.list_physical_devices('GPU') else "NOT AVAILAB
```

```
1 Physical GPUs, 1 Logical GPUs
Version:  2.8.0
Eager mode:  True
Hub version:  0.12.0
GPU is available
```

Transformation et vérification des Dataset

À la suite à nos recherches, nous avons décidé d'aller de l'avant avec cette source de donnée:
https://ieee-dataport.org/open-access/covid-19-fake-news-infodemic-research-dataset-covid19-fnir-dataset

Il s'agit d'un ensemble de données incluant 3796 fausses nouvelles et 3794 vraies nouvelles en lien avec le Covid. Chaque nouvelle a été séparée dans deux fichiers Excel nommés "fakeNews.csv" et "trueNews.csv". Ceux-ci contiennent plusieurs informations tels que la région d'origine, le vérificateur de la nouvelle, etc. Afin d'utiliser ces fichiers, nous avons normalisé et combiné ceux-ci en un seul fichier à l'aide du code suivant.

REMARQUE: les "paths" (chemins) des fichiers sont directement sur mon ordinateur. Par conséquent, une personne désirant utiliser ce fichier devra télécharger les fichiers de la source et modifier le chemin menant à ces fichiers.

Nous ne sommes pas les auteurs de cet ensemble de données, en voici la source:

Julio A. Saenz, Sindhu Reddy Kalathur Gopal, Diksha Shukla, June 12, 2021, "Covid-19 Fake News Infodemic Research Dataset (CoVID19-FNIR Dataset)", IEEE Dataport, doi: https://dx.doi.org/10.21227/b5bt-5244.

In [ ]:
```
# La ligne suivante doit être modifié selon l'emplacement de vos fichiers.
all_filenames = ["Data\\fakeNews.csv", "Data\\trueNews.csv"]

df = pd.concat([pd.read_csv(f) for f in all_filenames ])
df = df.drop(['Date Posted', 'Link', 'Region', 'Country', 'Explanation', 'Origin', 'Ori

# La méthode suivante sert à normaliser nos données d'entrée. Nous utilisons donc Regex
# Le but est d'obtenir deux bases de données contenant des nouvelles vraie et fausse ét
def rem(str):
    str = re.sub("(http).*?\s", "", str)
    str = re.sub("#", "", str)
    str = re.sub("@", "", str)
    str = re.sub("(pic.twitter.com).*?$", "", str)
    str = re.sub("\n", " ", str)
    str = re.sub("^\s", "", str)
    return str

# Cette ligne appelle la méthode plus haut et l'exécute sur chaque nouvelle de notre ba
df['Text'] = df['Text'].map(rem)

# Cette ligne nous sert à exporter notre document une fois toutes les données normalisé
# le contenue une fois les étapes précédentes terminer. De cette façon, nous sommes en
df.to_csv("combined.csv", index=False, encoding='utf-8-sig')
```

Nous avons maintenant terminé le téléchargement et la transformation de nos données, vérifions qu'elles sont conformes. Les deux sections suivantes servent à visualiser le résultat de nos opérations effectué à la section précédente. Une fois que nous avons confirmé que les données sont uniformes, nous sommes en mesure d'aller de l'avant.

In [ ]:
```
df.head()
```

Out[ ]:

	Text	Label
0	Tencent revealed the real number of deaths.\t\t	0
1	Taking chlorine dioxide helps fight coronaviru...	0

	Text	Label
2	This video shows workmen uncovering a bat-infe...	0
3	The Asterix comic books and The Simpsons predi...	0
4	Chinese President Xi Jinping visited a mosque ...	0

In [ ]:
```
df.tail()
```

Out[ ]:

	Text	Label
3788	Global COVID-19 prevention trial of hydroxychl...	1
3789	Bavaria's free COVID-19 test for all splits Ge...	1
3790	Britain locks down city of Leicester after COV...	1
3791	UK imposes lockdown on city of Leicester to cu...	1
3792	Grace Fusco, the matriarch of a large New Jers...	1

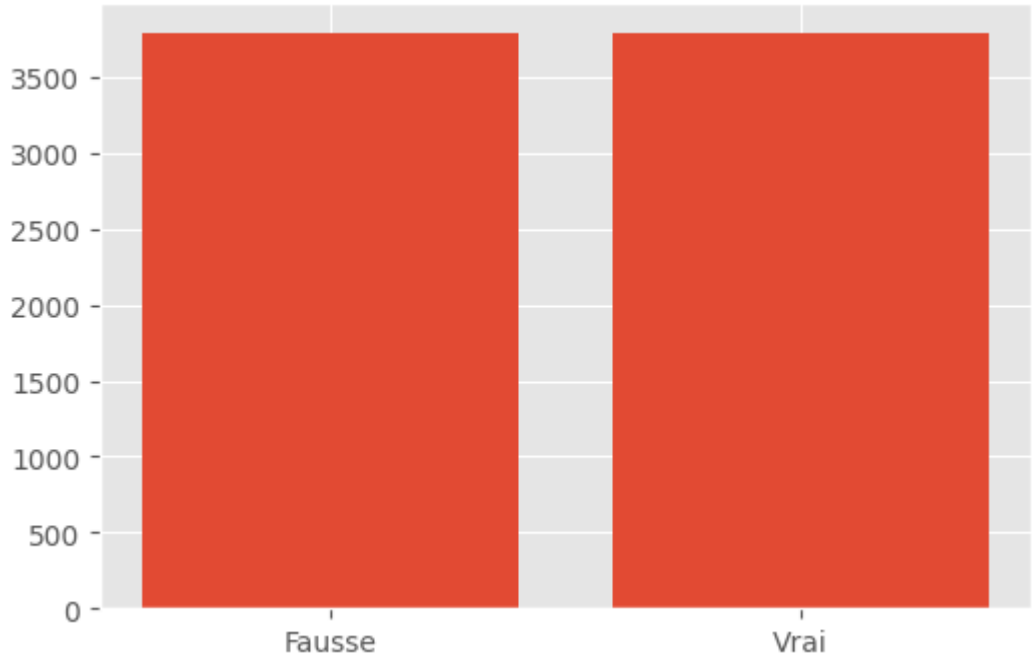
Afin de s'assurer que nos données sont conformes, vérifions que nous avons seulement 2 types d'arguments dans la catégorie "label". En temps normal, nous devons voir "0" et "1" en résultat ("0" est associé aux fausses nouvelles et "1" est associé aux vraies nouvelles).

In [ ]:
```
df.Label.unique()
```

Out[ ]:
```
array([0, 1], dtype=int64)
```

Maintenant, nous sommes en mesure de vérifier la répartition de nos données entre les nouvelles vraie et fausse. Il est important que la quantité de vraie et fausse nouvelles soit le plus près possible. Voici une représentation visuelle de notre répartition de donnée.

In [ ]:
```
classes = df.Label.unique()
counts = []

for i in classes:
  count = len(df[df.Label==i])
  counts.append(count)

plt.bar(['Fausse', 'Vrai'], counts)
plt.show()
```



Codage des étiquettes

Maintenant que nos données ont été créés et vérifiées, nous sommes en mesure de les séparer. La fonction "train_test_split()" sert à séparer efficacement nos données en quatre catégories:

1. x_train -> variable contenant 80% des textes d'entrée servant à entraîner l'algorithme BERT.
2. x_test -> variable contenant 20% des textes d'entrée servant à tester notre algorithme une fois qu'il a été entraîné.
3. y_train -> variable contenant les "labels" (0 ou 1) associer aux textes de "x_train".
4. y_test -> variable contenant les "labels" (0 ou 1) associer aux textes de "x_test".

Les détails du fonctionnement de la méthode se trouvent à cette source: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html

In [ ]:
```
# To remove?
sample_size = int(len(df))
sampleDf = df.sample(sample_size, random_state=23)

x = sampleDf.Text.values
y = sampleDf.Label.values

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.20, random_state=
```

LabelEncoder()

Cette section sert à normaliser les données d'entrée afin d'être utilisables par BERT. La documentation des méthodes utilisée peut être retrouvée ici: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html

In [ ]:
```

```

```
encoder = LabelEncoder()
encoder.fit(y)
encoded_Y_test = encoder.transform(y_test)
encoded_Y_train = encoder.transform(y_train)

# Convertis un vecteur en matrice
dummy_y_test = np_utils.to_categorical(encoded_Y_test)
dummy_y_train = np_utils.to_categorical(encoded_Y_train)
```

Une fois nos étiquettes créées, nous pouvons sauvegarder ce modèle afin de l'utiliser plus tard.

In [ ]:
```
encoder_fname = 'FakeNews_classes.npy'

# La ligne suivante sert à préciser l'endroit ou sauvegarder le modèle. Si une personne
# la ligne suivante doit être modifiée.
my_wd = ''
np.save(os.path.join(my_wd, encoder_fname) , encoder.classes_)
```

Création des jetons (Tokenization)

Continuons maintenant avec le modèle BERT. L'algorithme BERT peut être entraîné sur plusieurs types de données différents selon le besoin. Dans notre situation, nous avons décidé d'opter pour un modèle multilinguiste, au cas où nous décidons d'analyser des nouvelles de diverses langues. Ce modèle fut entraîné sur Wikipédia, les détails se trouvent ci-dessous. Tensorflow offre plusieurs modèles open source que vous êtes en mesure de retrouver ici: https://tfhub.dev/s?module-type=text-embedding,text-classification,text-generation,text-language-model,text-question-answering,text-retrieval-question-answering.

Le modèle choisi a été entraîner par: Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova: "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," 2018. Ce dernier a été entrainé sur plusieurs pages de Wikipédia contenant plusieurs langues. Plus de détails ici: https://tfhub.dev/tensorflow/bert_multi_cased_L-12_H-768_A-12/2.

REMARQUE: Nous avons utilisé la deuxième version du modèle, cependant la quatrième version est disponible. L'implémentation de celle-ci sera différente, nous avons donc décidé de continuer notre implémentation avec la version 2.

In [ ]:
```
# Importation du modèle choisi.
bert_layer = hub.KerasLayer("https://tfhub.dev/tensorflow/bert_multi_cased_L-12_H-768_A

# Déclaration des variables nécessaires selon la documentation de ce modèle. Détails ci
vocab_file = bert_layer.resolved_object.vocab_file.asset_path.numpy()
do_lower_case = bert_layer.resolved_object.do_lower_case.numpy()
tokenizer = tokenization.FullTokenizer(vocab_file, do_lower_case)
```

Nous avons initié certaines valeurs de notre modèle, voici leurs fonctionnalités:

1. "vocab_file" permet de lire le fichier "vocab", contenant le vocabulaire du modèle

2. "do_lower_case" détermine si l'on doit transformer tous les textes en minuscule. Par défaut, ce paramètre devrait être "False".
3. "tokenizer" crée le jeton selon les deux paramètres créer précédemment.

Notre jeton a ensuite besoin de deux variable, classificateur et séparateur. Nous devons donc les ajouter de cette façon:

In [ ]:
```
tokenizer.convert_tokens_to_ids(['[CLS]', '[SEP]'])
```

Out[ ]:
```
[101, 102]
```

Nous avons tous ce qu'il nous faut afin de procéder à la "tokenization". Nous ajoutons un [SEP] entre chaque jeton afin de séparer les données. Chaque mot va ensuite être remplacé par son chiffre équivalent selon le modèle de BERT choisi. Le but est de créer une matrice avec nos données d'entrée et d'analyser celles-ci.

In [ ]:
```
def encode_names(n):
   tokens = list(tokenizer.tokenize(n))
   tokens.append('[SEP]')
   return tokenizer.convert_tokens_to_ids(tokens)

text_data = tf.ragged.constant([
    encode_names(n) for n in x_train])
```

Voici donc à quoi ressemblent nos textes suite à la tokenization.

In [ ]:
```
text_data[0]
```

Out[ ]:
```
<tf.Tensor: shape=(23,), dtype=int32, numpy=
array([10117, 13299, 14424, 13575, 16411, 25907, 80352, 10107, 10189,
       11155, 10301, 10798, 11084, 10233,   117, 10259, 50438, 10188,
       31206, 37715, 10251,   119,   102])>
```

Afin de mieux comprendre comment notre texte s'est transformé en matrice de nombres, voici à quoi ressemble le texte original:

In [ ]:
```
x_train[0]
```

Out[ ]:
```
'The Chinese news website Tencent filters that there are more than 24,000 deaths from co
ronavirus.\t\t'
```

Voici ensuite la valeur de chaque mot selon la matrice créée:

In [ ]:
```
tokenizedText = tokenizer.tokenize(x_train[0])
for i in tokenizedText:
  print(i, tokenizer.convert_tokens_to_ids([i]))
```

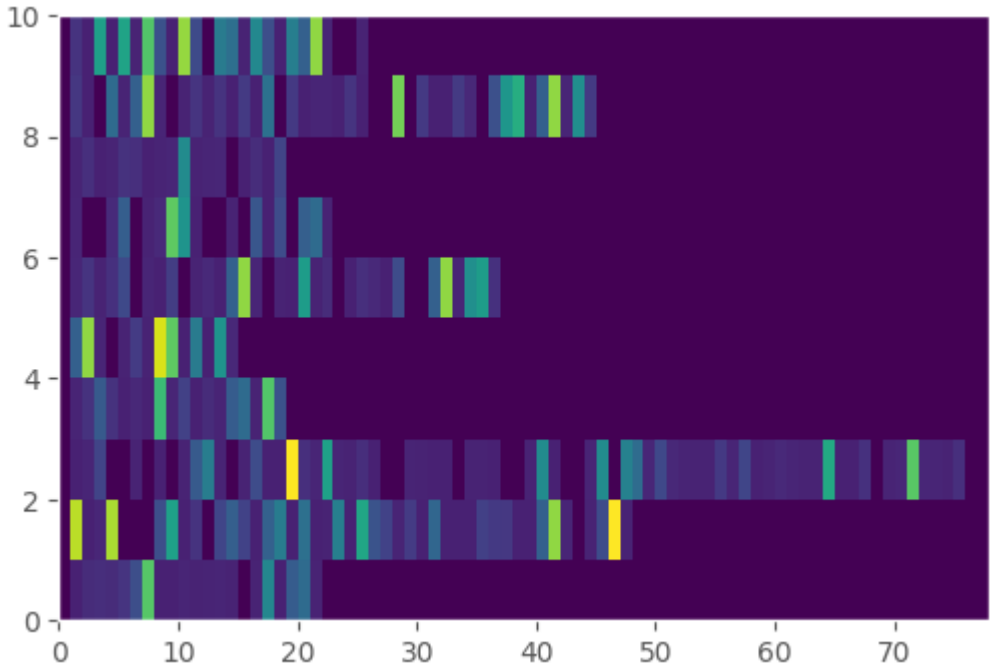
```
The [10117]
Chinese [13299]
news [14424]
website [13575]
Ten [16411]
```

```
##cent [25907]
filter [80352]
##s [10107]
that [10189]
there [11155]
are [10301]
more [10798]
than [11084]
24 [10233]
, [117]
000 [10259]
deaths [50438]
from [10188]
corona [31206]
##vir [37715]
##us [10251]
. [119]
```

Voici une représentation visuelle de nos jetons créés à l'étape précédente.

In [ ]:
```
cls = [tokenizer.convert_tokens_to_ids(['[CLS]'])]*text_data.shape[0]
input_word_ids = tf.concat([cls, text_data], axis=-1)
_ = plt.pcolormesh(input_word_ids[0:10].to_tensor())
```

```
C:\Users\samue\AppData\Local\Temp\ipykernel_3788\2371236668.py:3: MatplotlibDeprecationW
arning: Auto-removal of grids by pcolor() and pcolormesh() is deprecated since 3.5 and w
ill be removed two minor releases later; please call grid(False) first.
  _ = plt.pcolormesh(input_word_ids[0:10].to_tensor())
```



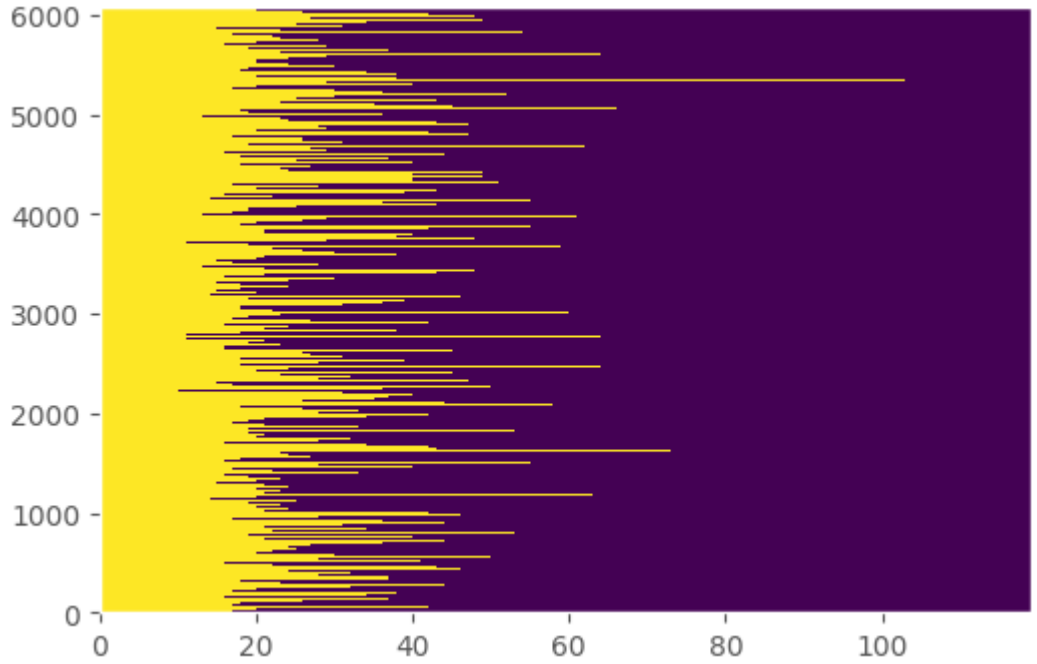
Masque & type

Le modèle que nous utilisons a besoin de deux données supplémentaires. Le masque et le type. Le masque sert à différencier le contenu d'entrée (les textes) et le rembourrage que nous avons créé. Voir le graphe suivant pour une représentation visuelle. Les lignes jaunes sont les données d'entrée, et le mauve est le rembourrage.

In [ ]:
```
input_word_ids
input_mask = tf.ones_like(input_word_ids).to_tensor()
plt.pcolormesh(input_mask)
```

```
C:\Users\samue\AppData\Local\Temp\ipykernel_3788\2782785498.py:3: MatplotlibDeprecationW
arning: Auto-removal of grids by pcolor() and pcolormesh() is deprecated since 3.5 and w
ill be removed two minor releases later; please call grid(False) first.
  plt.pcolormesh(input_mask)
```

Out[ ]:
```
<matplotlib.collections.QuadMesh at 0x1b471c62520>
```

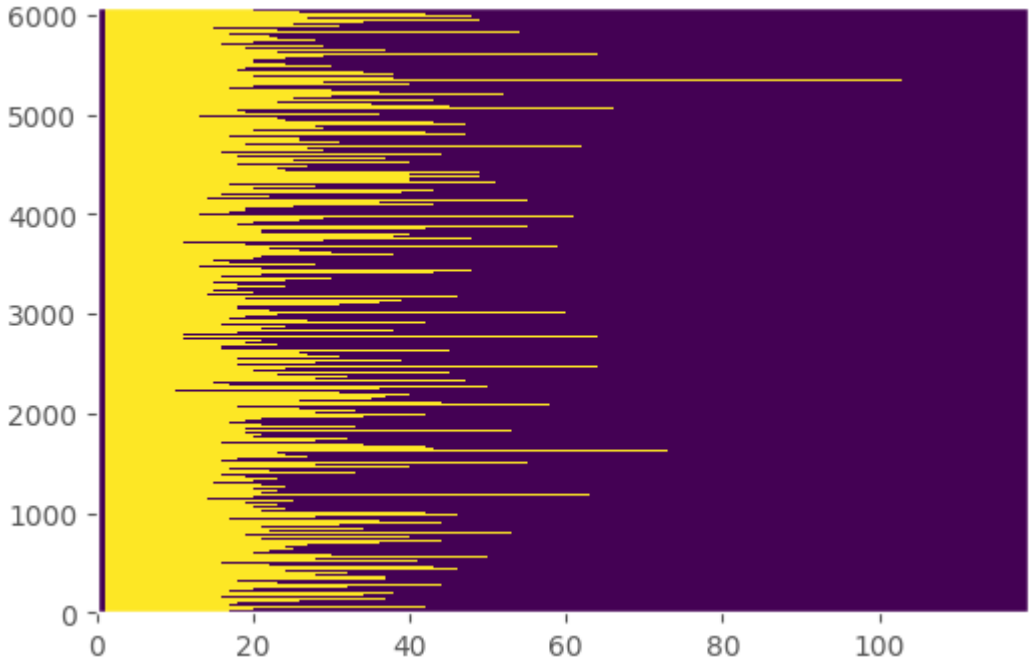


Bien que le graphe ci-dessous est très similaire, nous avons ajouté un bloc vide [CLS] au début de chaque texte (en mauve), afin d'insérer le jeton CLS (classification).

In [ ]:
```
type_cls = tf.zeros_like(cls)
type_text = tf.ones_like(text_data)
input_type_ids = tf.concat([type_cls, type_text], axis=-1).to_tensor()

plt.pcolormesh(input_type_ids)
```

```
C:\Users\samue\AppData\Local\Temp\ipykernel_3788\432401105.py:5: MatplotlibDeprecationWa
rning: Auto-removal of grids by pcolor() and pcolormesh() is deprecated since 3.5 and wi
ll be removed two minor releases later; please call grid(False) first.
  plt.pcolormesh(input_type_ids)
```

Out[ ]:
```
<matplotlib.collections.QuadMesh at 0x1b471d5f820>
```



La variable "input_type_ids" représente à quoi va ressembler nos matrices d'entrer. Plus de détails ici: https://www.tensorflow.org/hub/common_saved_model_apis/text#transformer-encoders

In [ ]:
```
input_type_ids
```

Out[ ]:
```
<tf.Tensor: shape=(6070, 119), dtype=int32, numpy=
array([[0, 1, 1, ..., 0, 0, 0],
       [0, 1, 1, ..., 0, 0, 0],
       [0, 1, 1, ..., 0, 0, 0],
       ...,
       [0, 1, 1, ..., 0, 0, 0],
       [0, 1, 1, ..., 0, 0, 0],
       [0, 1, 1, ..., 0, 0, 0]])>
```

Création d'une fonction afin de simplifier les étapes

Afin de simplifier les étapes suivantes, nous allons créer une fonction permettant d'utiliser notre modèle. Cette fonction reprend certains éléments définis plus haut à des fins de clarté.

Commençons tout d'abord à créer une longueur maximum de nos données d'entrées. Nous allons prendre le texte d'entré le plus long, puis le multiplier par 1.5, au cas ou un de nos textes d'entrer s'avère plus long que nos données d'entrées initiales.

In [ ]:
```
lens = [len(i) for i in input_word_ids]
max_seq_length = max(lens)
print('Longueure maximum initiale: ', max_seq_length)

max_seq_length = int(1.5*max_seq_length)
print('Longueure maximum utilisé: ', max_seq_length)
```

```
Longueure maximum initiale:  119
Longueure maximum utilisé:  178
```

Nous sommes ensuite en mesure de créer les fonctions réutilisables afin de rendre le prétraitement plus facile. Ceux-ci ont été décrits plus haut.

In [ ]:
```
def encode_names(n, tokenizer):
   tokens = list(tokenizer.tokenize(n))
   tokens.append('[SEP]')
   return tokenizer.convert_tokens_to_ids(tokens)

def bert_encode(string_list, tokenizer, max_seq_length):
  num_examples = len(string_list)

  string_tokens = tf.ragged.constant([
      encode_names(n, tokenizer) for n in np.array(string_list)])

  cls = [tokenizer.convert_tokens_to_ids(['[CLS]'])]*string_tokens.shape[0]
  input_word_ids = tf.concat([cls, string_tokens], axis=-1)

  input_mask = tf.ones_like(input_word_ids).to_tensor(shape=(None, max_seq_length))

  type_cls = tf.zeros_like(cls)
  type_tokens = tf.ones_like(string_tokens)
  input_type_ids = tf.concat(
      [type_cls, type_tokens], axis=-1).to_tensor(shape=(None, max_seq_length))

  inputs = {
      'input_word_ids': input_word_ids.to_tensor(shape=(None, max_seq_length)),
      'input_mask': input_mask,
      'input_type_ids': input_type_ids}

  return inputs
```

Nous sommes maintenant prêts à faire le prétraitement de nos données

In [ ]:
```
X_train = bert_encode(x_train, tokenizer, max_seq_length)
X_test = bert_encode(x_test, tokenizer, max_seq_length)
```

Modélisation

Traitement initiale

Nous sommes maintenant prêts à la modélisation. Nous créons notre modèle à l'aide de nos données d'entrées, du modèle BERT que nous avons choisi, ainsi qu'une couche de sortie selon le notre nombre de classes. Le code suivant provient de notre modèle source, les détails de l'utilisation se trouvent ici: https://tfhub.dev/tensorflow/bert_multi_cased_L-12_H-768_A-12/2

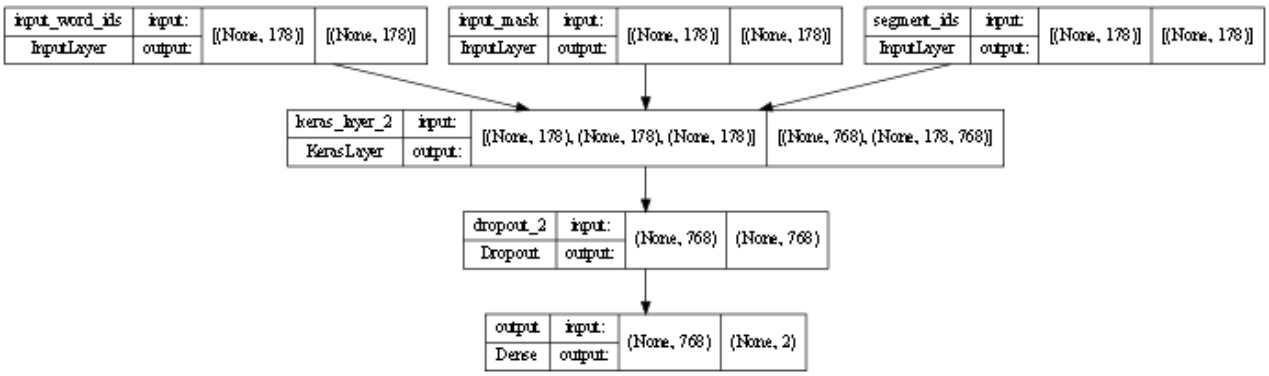
In [ ]:
```
num_class = len(encoder.classes_)  # Basée sur le nombre de classes utilisé (0-1)
max_seq_length = max_seq_length  # Longueur de texte calculé plus haut

# Créer selon le modèle multilingual choisi
```

```
input_word_ids = tf.keras.layers.Input(shape=(max_seq_length,), dtype=tf.int32, name="i
input_mask = tf.keras.layers.Input(shape=(max_seq_length,), dtype=tf.int32, name="input
segment_ids = tf.keras.layers.Input(shape=(max_seq_length,), dtype=tf.int32, name="segm

# Créer selon le modèle multilingual choisi
pooled_output, sequence_output = bert_layer([input_word_ids, input_mask, segment_ids])
output = tf.keras.layers.Dropout(rate=0.1)(pooled_output)
output = tf.keras.layers.Dense(num_class, activation='softmax', name='output')(output)

model = tf.keras.Model(
    inputs={
        'input_word_ids': input_word_ids,
        'input_mask': input_mask,
        'input_type_ids': segment_ids
        },
        outputs=output)
```

Voici à quoi ressemble notre modèle. Nous avons le input_word_ids (les données d'entrée), le input_mask (le masque), et le segment_ids (afin de segmenter nos données). Le tout est exécuté dans la couche de Keras, pour ensuite nous donner le résultat final (output).

In [ ]:
```
tf.keras.utils.plot_model(model, show_shapes=True, dpi=48)
```

Out[ ]:



Entrainement de BERT

Nous pouvons maintenant initier les paramètres de notre entrainement. Epochs est le nombre d'itérations de notre algorithme. Dans notre situation, nous avons décidé de procéder seulement à trois itérations. Le batch_size devrait être augmenté selon le VRAM de votre GPU. Puisque ce notebook a pour but d'être utilisé par plusieurs personnes avec divers GPU, nous avons décidé de le laisser à 16. Cependant, une personne ayant un bon GPU pourrait l'augmenter afin d'augmenter la rapidité de l'algorithme.

In [ ]:
```
epochs = 3  # Nombre d'itération à faire. Par conséquent, nous allons entrainer notre a
batch_size = 16 # Peut-être ajusté selon le VRAM de notre GPU. Plus le nombre est grand
eval_batch_size = batch_size  # Utile?

train_data_size = len(dummy_y_train)
steps_per_epoch = int(train_data_size / batch_size)
num_train_steps = steps_per_epoch * epochs
warmup_steps = int(epochs * train_data_size * 0.1 / batch_size)
```

```
optimizer = nlp.optimization.create_optimizer(
    2e-5, num_train_steps=num_train_steps, num_warmup_steps=warmup_steps)
```

Une fois tous les paramètres déterminés, nous sommes en mesure de compiler le modèle. Voici à quoi ressemble la compilation du modèle.

In [ ]:

```
model.compile(optimizer=optimizer,
              loss='categorical_crossentropy',
              metrics=['accuracy'])

model.summary()
```

```
Model: "model_2"
__________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to
==========================================================================================
 input_word_ids (InputLayer)    [(None, 178)]        0           []

 input_mask (InputLayer)        [(None, 178)]        0           []

 segment_ids (InputLayer)       [(None, 178)]        0           []

 keras_layer_2 (KerasLayer)     [(None, 768),        177853441   ['input_word_ids[0]
[0]',
                                 (None, 178, 768)]                 'input_mask[0][0]',
                                                                   'segment_ids[0][0]']

 dropout_2 (Dropout)            (None, 768)          0           ['keras_layer_2[0][0]']

 output (Dense)                 (None, 2)            1538        ['dropout_2[0][0]']

==========================================================================================
Total params: 177,854,979
Trainable params: 177,854,978
Non-trainable params: 1
__________________________________________________________________________________________
```

Tel que nous pouvons le constater, le modèle choisi utilise un total de 177M de paramètre. Voilà la raison d'où BERT est plus performant qu'un réseau de neurones traditionnel, un réseau traditionnel aurait environ 10M de paramètre. Par conséquent, notre modèle devrait être plus précis.

Exécution de l'algorithme

Voici maintenant l'exécution de notre algorithme. Si vous utilisez les machines virtuelles de Google Colab, l'algorithme prendra de 30 à 45 minutes à compiler. Évidemment, le temps va grandement varier selon le nombre de données d'entrées. De plus, si l'exécution se fait sur un meilleur système, il sera beaucoup plus rapide. Sur notre ordinateur personnel, cette étape prend seulement 3 à 4 minutes. Voici les spécifications de notre ordinateur utilisé pour comparaison: CPU: AMD Ryzen 9

3900X 12 Cores 24 Threads @ 4.00 GHz RAM: 32Gb DDR4 3200MHz GPU: Nvidia GeForce RTX 3090 - 10,496 CUDA Cores

REMARQUE: La pièce la plus importante ici est le GPU. Puisque nous utilisons la parallélisation, le nombre de CUDA Cores du GPU est très important. La version gratuite de Google Colab utilise des GPU NVIDIA® Tesla® K80, possédant 4,991 CUDA Cores. Notre exécution de code est donc beaucoup plus rapide, puisque nous avons un peu plus que le double de CUDA Cores.

In [ ]:
```
history = model.fit(X_train,
                    dummy_y_train,
                    epochs=epochs,
                    batch_size=batch_size,
                    validation_data=(X_test, dummy_y_test),
                    verbose=1)
```

```
Epoch 1/3
380/380 [==============================] - 77s 174ms/step - loss: 0.1868 - accuracy: 0.9
257 - val_loss: 0.1074 - val_accuracy: 0.9704
Epoch 2/3
380/380 [==============================] - 66s 173ms/step - loss: 0.0538 - accuracy: 0.9
863 - val_loss: 0.0517 - val_accuracy: 0.9855
Epoch 3/3
380/380 [==============================] - 66s 173ms/step - loss: 0.0153 - accuracy: 0.9
951 - val_loss: 0.0692 - val_accuracy: 0.9875
```

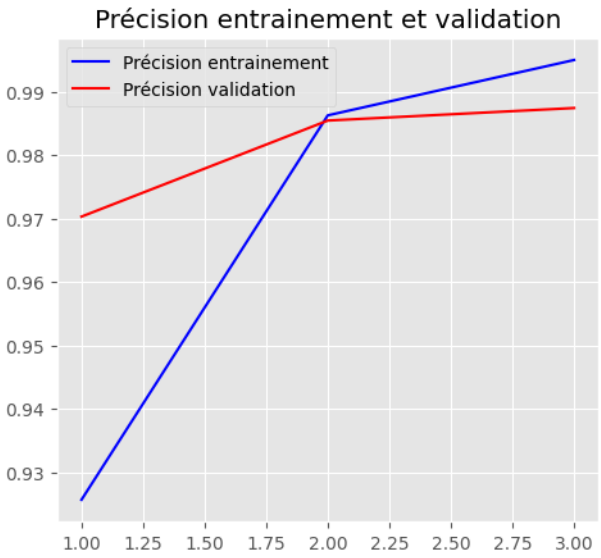
Les résultats sont très prometteurs:

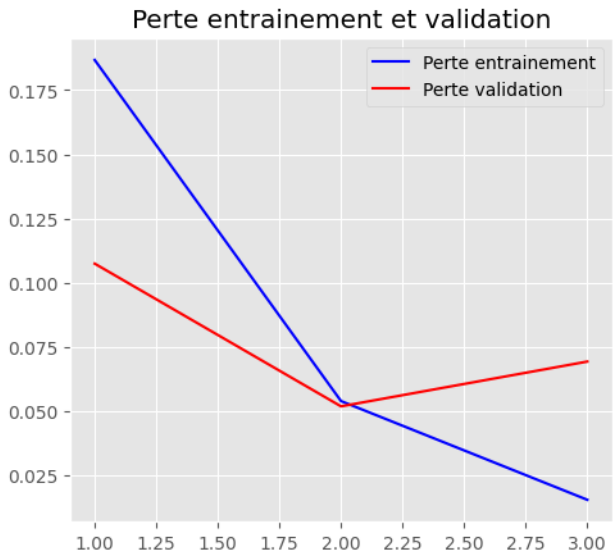
In [ ]:
```
loss, accuracy = model.evaluate(X_train, dummy_y_train, verbose=False)
print("Précision de l'entrainement: {:.4f}".format(accuracy))
loss, accuracy = model.evaluate(X_test, dummy_y_test, verbose=False)
print("Précision des tests:  {:.4f}".format(accuracy))
```

```
Précision de l'entrainement: 0.9987
Précision des tests:  0.9875
```

In [ ]:
```
# Utilisation de la librairie matplotlib afin d'afficher les résultats
plt.style.use('ggplot')

def plot_history(history):
    acc = history.history['accuracy']
    val_acc = history.history['val_accuracy']
    loss = history.history['loss']
    val_loss = history.history['val_loss']
    x = range(1, len(acc) + 1)

    plt.figure(figsize=(12, 5))
    plt.subplot(1, 2, 1)
    plt.plot(x, acc, 'b', label='Précision entrainement')
    plt.plot(x, val_acc, 'r', label='Précision validation')
    plt.title('Précision entrainement et validation')
    plt.legend()
    plt.subplot(1, 2, 2)
    plt.plot(x, loss, 'b', label='Perte entrainement')
    plt.plot(x, val_loss, 'r', label='Perte validation')
    plt.title('Perte entrainement et validation')
    plt.legend()
```

```
plot_history(history)
```





Les sections suivantes servent simplement à sauvegarder notre modèle

In [ ]:
```
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

In [ ]:
```
# Le chemin "my_wd" devra être modifié selon l'utilisateur

model_fname = 'fakenews_BERT'
my_wd = ''

model.save(os.path.join(my_wd, model_fname))
```

```
WARNING:absl:Found untraced functions such as restored_function_body, restored_function_body, restored_function_body, restored_function_body, restored_function_body while saving (showing 5 of 378). These functions will not be directly callable after loading.
INFO:tensorflow:Assets written to: fakenews_BERT\assets
INFO:tensorflow:Assets written to: fakenews_BERT\assets
```

In [ ]:
```
model_fname = 'fakenews_BERT'
new_model = tf.keras.models.load_model(os.path.join(my_wd, model_fname))

loss, accuracy = new_model.evaluate(X_test, dummy_y_test, verbose=False)
print("Précision des tests::  {:.4f}".format(accuracy))
```

```
Précision des tests::  0.9875
```

In [ ]:
```
tokenizerSaved = bert.tokenization.FullTokenizer(
    vocab_file=os.path.join(my_wd, model_fname, 'assets/vocab.txt'),
    do_lower_case=False)
```

Application de l'algorithme

In [ ]:
```
encoder_fname = 'FakeNews_classes.npy'
my_wd = ""

encoder = LabelEncoder()
encoder.classes_ = np.load(os.path.join(my_wd, encoder_fname), allow_pickle=True)
```

Création des statistiques

La section suivante démontre les résultats de notre projet. Nous avons décidé de prendre une base de données d'environ 5.2 millions d'articles et nouvelles en lien avec le Covid-19 sur une période de 4 mois (décembre 2019 à Mars 2020). Puisque notre algorithme à été conçu à l'aide de tweets et titre de nouvelle, nous avons décidé d'appliquer notre algorithme à tous les titres de cette base de données. Il est important de noter que chaque titre contenant moins de 30 caractères a été retiré de la liste, puisqu'il est difficile d'analyser un texte si court. Au final, nous avons compilé nos résultats en divers graphique selon ce que nous avons trouvé d'intéressant, à l'aide de la librairie Matplotlib: https://matplotlib.org/

Il est important de noter que l'analyse de cette massive quantité de données à duré prêt de 48h. Par conséquent, si vous désirer procéder à votre propre analyse, il sera impossible de le faire à l'aide de Google Colab gratuit. L'analyse a été effectuée sur l'ordinateur mentionné plus haut.

Source de la base de données: Ran Geva, March 31, 2020, "free dataset from news/message boards/blogs about CoronaVirus (4 month of data - 5.2M posts)", IEEE Dataport, doi: https://dx.doi.org/10.21227/kc4v-q323.

In [ ]:
```
import json
import glob
import re
import time
from tqdm import tqdm

# Liste des variables global
# Variable servant à trouver le total des sources vrai et fausse
total_true = 0
total_false = 0

# Dict des régions
region_dict_true = {}
region_dict_false = {}

# Dict des dates
date_dict_true = {}
date_dict_false = {}

# Variable servant à trouver quantité de likes qui viennent de sources vraie et fausse
total_like_fake_source = 0
total_like_true_source = 0

# Variable servant à trouver le nombre de partages sur Facebook par sources vraie et fa
total_share_fake_source = 0
total_share_true_source = 0
```

```
#Variable servant à trouver la quantité de sources crédibles et non crédibles qui ont d
#Voir https://www.reputio.com/what-is-spam-score/ en référence
total_vrai_bas_risque = 0
total_vrai_medium_risque = 0
total_vrai_haut_risque = 0
total_faux_bas_risque = 0
total_faux_medium_risque = 0
total_faux_haut_risque = 0

# Puisque la base de données utilise est de 13gb, il est impossible de la charger au co
# Par conséquent, nous avons appelons cette méthode individuellement à chaque fichier J
def imp(single_file):
    dataList = []
    with open(single_file, encoding='utf-8') as f:
        for jsonObj in f:
            data = json.loads(jsonObj)
            dataList.append(data)
    tqdm.write('\n')


    #Initialisation des variables globales
    #En python, une variable globale "int" doit être réinitialisée à l'intérieur de la
    global total_true
    global total_false
    global total_like_true_source
    global total_like_fake_source
    global total_share_fake_source
    global total_share_true_source
    global total_vrai_bas_risque
    global total_vrai_medium_risque
    global total_vrai_haut_risque
    global total_faux_bas_risque
    global total_faux_medium_risque
    global total_faux_haut_risque

    # Boucle comprenant chaque calcul désiré
    # La librairie TQDM permet de visualiser la durée de l'algorithme et de donner une
    for data in tqdm(dataList, desc='Progrès', unit=' articles', position=0, mininterva
        # Évaluation de la nouvelle
        txt = [data['title']]
        # Condition permettant de passer à la prochaine nouvelle si elle est trop court
        if len(str(txt))<30:
            continue

        # Partie la plus importante du code, cette section évalue si la nouvelle est vr
        inputs = bert_encode(string_list=list(txt),
                             tokenizer=tokenizerSaved,
                             max_seq_length=178)
        prediction = new_model.predict(inputs)
        prediction_bool = encoder.classes_[np.argmax(prediction)]


        # Conditions permettant d'obtenir les régions par type de nouvelles
        if prediction_bool:
            # Permets d'obtenir la quantité de likes et partage qui viennent de sources
            nb_like_fake_position_x = data['thread']['social']['facebook']['likes']
            total_like_fake_source = total_like_fake_source + nb_like_fake_position_x
            nb_share_fake_position_x = data['thread']['social']['facebook']['shares']
            total_share_fake_source = total_share_fake_source + nb_share_fake_position_
```

```
            # Conditions permettant d'obtenir les régions par type de nouvelles
            total_true += 1
            region = data['thread']['country']
            if region in region_dict_true:
                region_dict_true[region] +=1
            elif region== '':
                pass
            else:
                region_dict_true.update({region: 1})

            # Conditions permettant d'obtenir les dates par type de nouvelles
            date = data['published']
            date = re.sub("(T).*", "", date)
            if date in date_dict_true:
                date_dict_true[date] += 1
            else:
                date_dict_true.update({date: 1})


        else:
            # Permets d'obtenir la quantité de likes et partage qui viennent de sources
            nb_like_true_position_x = data['thread']['social']['facebook']['likes']
            total_like_true_source = total_like_true_source + nb_like_true_position_x
            nb_share_true_position_x = data['thread']['social']['facebook']['shares']
            total_share_true_source = total_share_true_source + nb_share_true_position_

            # Conditions permettant d'obtenir les régions par type de nouvelles
            total_false += 1
            region = data['thread']['country']
            if region in region_dict_false:
                region_dict_false[region] +=1
            elif region== '':
                pass
            else:
                region_dict_false.update({region: 1})

            # Conditions permettant d'obtenir les dates par type de nouvelles
            date = data['published']
            date = re.sub("(T).*", "", date)
            if date in date_dict_false:
                date_dict_false[date] += 1
            else:
                date_dict_false.update({date: 1})


        # Conditions permettant d'obtenir la quantité de sources crédibles et non crédi
        if data['thread']['spam_score'] < 0.114 and prediction_bool==True:
            total_vrai_bas_risque += 1
        elif data['thread']['spam_score'] <= 0.56 and prediction_bool==True:
            total_vrai_medium_risque += 1
        elif data['thread']['spam_score'] > 0.56 and prediction_bool==True:
            total_vrai_haut_risque += 1
        elif data['thread']['spam_score'] < 0.114 and prediction_bool==False:
            total_faux_bas_risque += 1
        elif data['thread']['spam_score'] <= 0.56 and prediction_bool==False:
            total_faux_medium_risque += 1
        elif data['thread']['spam_score'] > 0.56 and prediction_bool==False:
            total_faux_haut_risque += 1

# Variable identifiant si l'on veut exécuter l'algorithme sur la totalité des données (
```

```
# Tel que mentionné plus haut, il faudra possiblement ajuster le chemin menant à ces fi
# Nous recommandons de tester l'algorithme sur une quantité réduite de données afin de
files = glob.glob('JsonFiles/*.json', recursive=True)
filesTest = glob.glob('JsonTest/*.json', recursive=True)
```

```
C:\Users\samue\anaconda3\envs\gpu7\lib\site-packages\tqdm\auto.py:22: TqdmWarning: IProg
ress not found. Please update jupyter and ipywidgets. See https://ipywidgets.readthedoc
s.io/en/stable/user_install.html
  from .autonotebook import tqdm as notebook_tqdm
```

In [ ]:

```
start = time.time()

# La variable "files" doit être modifiée si l'on désire exécute l'analyse sur la base d
# Modifier TQDM
for single_file in tqdm(files, desc='Progrès total', unit=' Articles', position=0, leav
    imp(single_file)


time.sleep(10)
done = True

end = time.time()
print("")
print("Temps d'exécution en secondes: ", end - start)
print("Temps d'exécution en minutes: ", (end - start)/60)
print("Temps d'exécution en heures: ", ((end - start)/60)/60)
print("Temps d'exécution en jours: ", (((end - start)/60)/60)/24)
```

```
Progrès total:   0%|          | 0/31 [00:00<?, ? Articles/s]

Progrès: 100%|██████████| 104/104 [00:07<00:00, 13.38 articles/s]
Progrès total:   3%|▎         | 1/31 [00:13<04:08,  8.27s/ Articles]

Progrès: 100%|██████████| 94299/94299 [2:00:44<00:00, 13.02 articles/s]
Progrès total:   6%|▋         | 2/31 [2:00:59<34:23:14, 4268.77s/ Articles]

Progrès: 100%|██████████| 10956/10956 [14:46<00:00, 12.36 articles/s]
Progrès total:  10%|▉         | 3/31 [2:15:52<21:11:31, 2724.68s/ Articles]

Progrès: 100%|██████████| 91706/91706 [2:02:50<00:00, 12.44 articles/s]
Progrès total:  13%|█▎        | 4/31 [4:18:49<34:12:36, 4561.37s/ Articles]

Progrès: 100%|██████████| 95061/95061 [2:02:44<00:00, 12.91 articles/s]
Progrès total:  16%|█▌        | 5/31 [6:21:40<40:15:38, 5574.56s/ Articles]

Progrès: 100%|██████████| 95069/95069 [2:03:52<00:00, 12.79 articles/s]
Progrès total:  19%|█▉        | 6/31 [8:25:39<43:06:58, 6208.75s/ Articles]

Progrès: 100%|██████████| 98312/98312 [2:00:15<00:00, 13.63 articles/s]
Progrès total:  23%|██▎       | 7/31 [10:25:57<43:35:57, 6539.88s/ Articles]

Progrès: 100%|██████████| 29972/29972 [35:54<00:00, 13.91 articles/s]
Progrès total:  26%|██▌       | 8/31 [11:01:57<32:52:00, 5144.39s/ Articles]

Progrès: 100%|██████████| 98173/98173 [2:03:25<00:00, 13.26 articles/s]
Progrès total:  29%|██▉       | 9/31 [13:05:30<35:46:12, 5853.30s/ Articles]
```

```
Progrès: 100%|██████████| 96954/96954 [2:04:57<00:00, 12.93 articles/s]
Progrès total:  32%|███       | 10/31 [15:10:35<37:07:02, 6362.98s/ Articles]

Progrès: 100%|██████████| 97144/97144 [2:03:00<00:00, 13.16 articles/s]
Progrès total:  35%|███       | 11/31 [17:13:42<37:05:36, 6676.83s/ Articles]

Progrès: 100%|██████████| 96634/96634 [2:03:15<00:00, 13.07 articles/s]
Progrès total:  39%|███▊      | 12/31 [19:17:05<36:24:12, 6897.52s/ Articles]

Progrès: 100%|██████████| 96112/96112 [2:02:05<00:00, 13.12 articles/s]
Progrès total:  42%|████      | 13/31 [21:19:18<35:08:49, 7029.42s/ Articles]

Progrès: 100%|██████████| 97635/97635 [1:36:36<00:00, 16.84 articles/s]
Progrès total:  45%|████▌     | 14/31 [22:56:02<31:26:44, 6659.11s/ Articles]

Progrès: 100%|██████████| 98095/98095 [1:36:46<00:00, 16.89 articles/s]
Progrès total:  48%|████▊     | 15/31 [24:32:55<28:27:49, 6404.33s/ Articles]

Progrès: 100%|██████████| 98263/98263 [1:38:25<00:00, 16.64 articles/s]
Progrès total:  52%|█████     | 16/31 [26:11:28<26:04:03, 6256.20s/ Articles]

Progrès: 100%|██████████| 98741/98741 [1:39:29<00:00, 16.54 articles/s]
Progrès total:  55%|█████▍    | 17/31 [27:51:04<24:00:09, 6172.12s/ Articles]

Progrès: 100%|██████████| 97256/97256 [1:38:43<00:00, 16.42 articles/s]
Progrès total:  58%|█████▊    | 18/31 [29:29:55<22:01:33, 6099.48s/ Articles]

Progrès: 100%|██████████| 96924/96924 [1:38:11<00:00, 16.45 articles/s]
Progrès total:  61%|██████    | 19/31 [31:08:13<20:07:51, 6039.32s/ Articles]

Progrès: 100%|██████████| 96839/96839 [1:38:12<00:00, 16.43 articles/s]
Progrès total:  65%|██████▍   | 20/31 [32:46:33<18:19:28, 5997.18s/ Articles]

Progrès: 100%|██████████| 96457/96457 [1:37:32<00:00, 16.48 articles/s]
Progrès total:  68%|██████▊   | 21/31 [34:24:12<16:32:40, 5956.02s/ Articles]

Progrès: 100%|██████████| 95880/95880 [1:37:12<00:00, 16.44 articles/s]
Progrès total:  71%|███████   | 22/31 [36:01:32<14:48:07, 5920.86s/ Articles]

Progrès: 100%|██████████| 98957/98957 [1:39:22<00:00, 16.60 articles/s]
Progrès total:  74%|███████▍  | 23/31 [37:41:01<13:11:24, 5935.51s/ Articles]

Progrès: 100%|██████████| 97188/97188 [1:37:40<00:00, 16.58 articles/s]
Progrès total:  77%|███████▋  | 24/31 [39:18:49<11:30:05, 5915.06s/ Articles]

Progrès: 100%|██████████| 96706/96706 [1:38:11<00:00, 16.41 articles/s]
Progrès total:  81%|████████  | 25/31 [40:57:08<9:51:00, 5910.14s/ Articles]

Progrès: 100%|██████████| 95635/95635 [1:38:20<00:00, 16.21 articles/s]
Progrès total:  84%|████████▍ | 26/31 [42:35:35<8:12:27, 5909.41s/ Articles]

Progrès: 100%|██████████| 95679/95679 [1:38:13<00:00, 16.23 articles/s]
Progrès total:  87%|████████▋ | 27/31 [44:13:56<6:33:47, 5906.88s/ Articles]
```

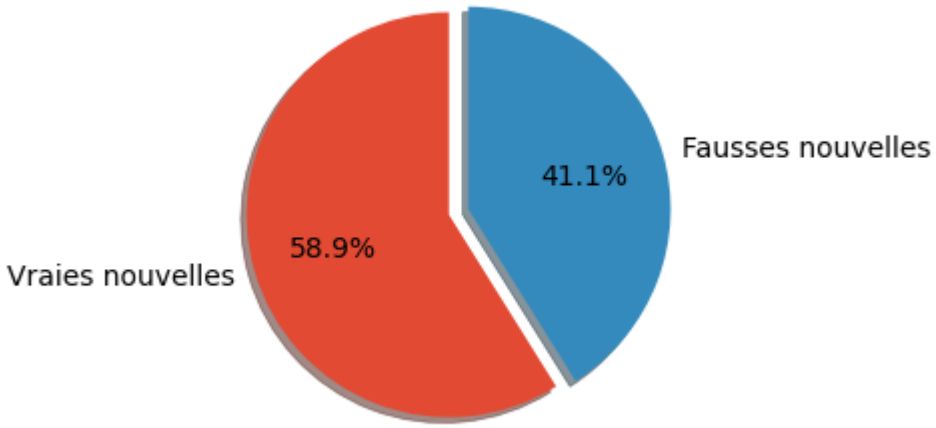
```
Progrès: 100%|██████████| 95691/95691 [1:37:59<00:00, 16.27 articles/s]
Progrès total:  90%|█████████ | 28/31 [45:52:03<4:55:02, 5900.81s/ Articles]

Progrès: 100%|██████████| 96296/96296 [2:05:24<00:00, 12.80 articles/s]
Progrès total:  94%|█████████▍ | 29/31 [47:57:34<3:33:00, 6390.06s/ Articles]

Progrès: 100%|██████████| 96302/96302 [2:05:51<00:00, 12.75 articles/s]
Progrès total:  97%|█████████▋ | 30/31 [50:03:31<1:52:20, 6740.55s/ Articles]

Progrès: 100%|██████████| 62420/62420 [1:18:38<00:00, 13.23 articles/s]

Temps d'exécution en secondes:  184941.81312155724
Temps d'exécution en minutes:  3082.363552025954
Temps d'exécution en heures:  51.372725867099234
Temps d'exécution en jours:  2.140530244462468
```

Résultat

Code affichant le nombre de vrai nouvelles détecter comparé au nombre de fausse nouvelles détecter.

Tout les graphes suivants ont été créés à l'aide de la librairie Matplotlib: https://matplotlib.org/

In [ ]:
```
plt.rcParams['figure.figsize'] = [4, 3]
plt.rcParams['figure.dpi'] = 100

labels = ['Vraies nouvelles', 'Fausses nouvelles']
sizes = [total_true, total_false]
explode = (0.1, 0)

fig1, ax1 = plt.subplots()
ax1.pie(sizes, explode=explode, labels=labels, autopct='%1.1f%%',
        shadow=True, startangle=90)
ax1.axis('equal')

plt.show()
```

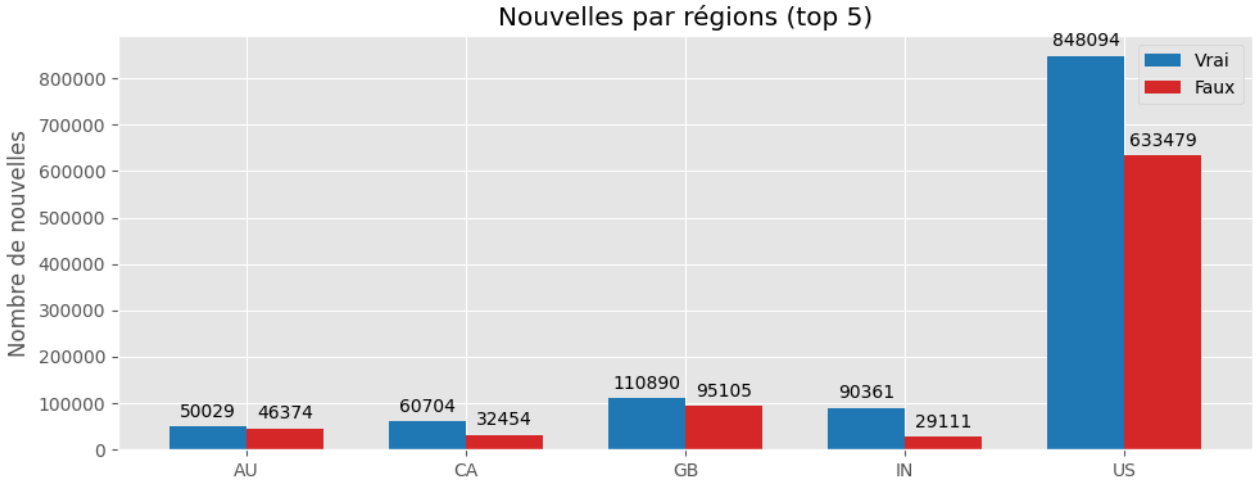


Code affichant le nombre de vraies et fausses nouvelles par régions

Il est important de noter que nous affichons seulement les régions ayant le plus d'articles, puisque notre base de données possède une centaine de régions.

In [ ]:

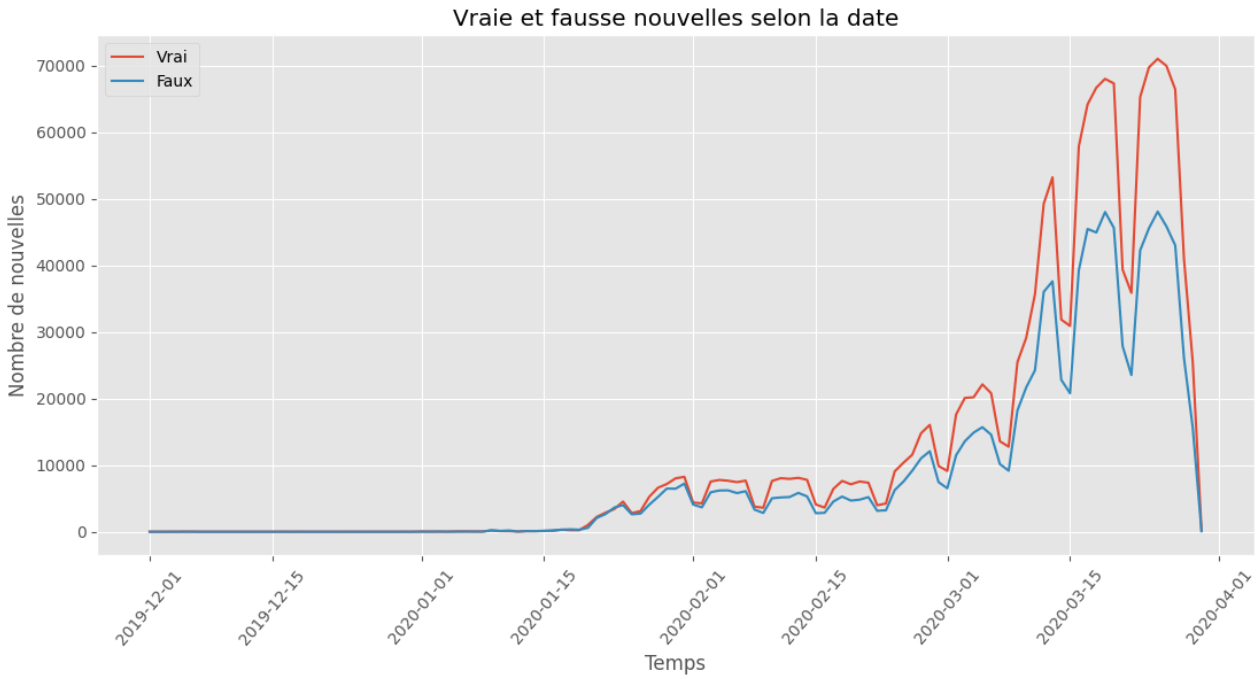
```
plt.rcParams['figure.figsize'] = [10, 4]
plt.rcParams['figure.dpi'] = 100

top_true_region = sorted(region_dict_true, key=region_dict_true.get, reverse=True)[:5]
top_false_region = sorted(region_dict_false, key=region_dict_false.get, reverse=True)[:
top_region_combined = sorted(set(top_true_region+top_false_region))

t_news_count = []
f_news_count = []

for x in range(len(top_region_combined)):
    if top_region_combined[x] in region_dict_true:
        t_news_count.append(region_dict_true[top_region_combined[x]])
    if top_region_combined[x] not in region_dict_true:
        t_news_count.append(0)
    if top_region_combined[x] in region_dict_false:
        f_news_count.append(region_dict_false[top_region_combined[x]])
    if top_region_combined[x] not in region_dict_false:
        f_news_count.append(0)


x = np.arange(len(top_region_combined))
width = 0.35

fig, ax = plt.subplots()
rects1 = ax.bar(x - width/2, t_news_count, width, label='Vrai', color='tab:blue')
rects2 = ax.bar(x + width/2, f_news_count, width, label='Faux', color='tab:red')

ax.set_ylabel('Nombre de nouvelles')
ax.set_title('Nouvelles par régions (top 5)')
ax.set_xticks(x, top_region_combined)
ax.legend()

ax.bar_label(rects1, padding=3)
ax.bar_label(rects2, padding=3)

fig.tight_layout()

plt.show()
```



Code affichant le nombre de vraie et fausse nouvelles par date

In [ ]:

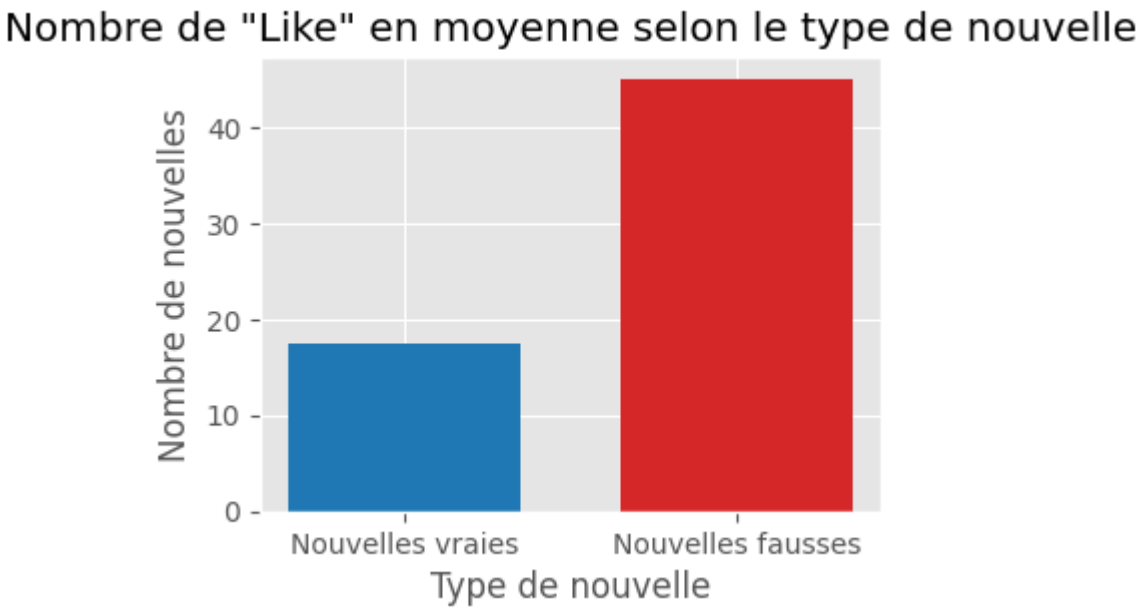
```
import collections
import datetime as dt
import matplotlib.dates as mdates
import numpy as np

plt.rcParams['figure.figsize'] = [13, 6]
plt.rcParams['figure.dpi'] = 100

# Ajout des dates manquantes par dict
for key in date_dict_true:
    if key not in date_dict_false.keys():
        date_dict_false.update({key: 0})

for key in date_dict_false:
    if key not in date_dict_true.keys():
        date_dict_true.update({key: 0})

# Mettre les dates en ordre
ddt_sorted = collections.OrderedDict(sorted(date_dict_true.items()))
date_key_true = list(ddt_sorted.keys())
date_value_true = list(ddt_sorted.values())

ddf_sorted = collections.OrderedDict(sorted(date_dict_false.items()))
date_key_false = list(ddf_sorted.keys())
date_value_false = list(ddf_sorted.values())

# Affichage des résultats
dates = mdates.date2num(date_key_true)
width = np.diff(dates).min()

plt.plot(mdates.num2date(dates), date_value_true, label = 'Vrai')
plt.plot(mdates.num2date(dates), date_value_false, label = 'Faux')

plt.xlabel('Temps')
plt.ylabel('Nombre de nouvelles')
plt.title('Vraie et fausse nouvelles selon la date')

plt.xticks(rotation=50)
ax.xaxis_date()
fig.autofmt_xdate()

plt.legend()
plt.show()
```



Code affichant le nombre en moyenne de like par type de nouvelles

In [ ]:

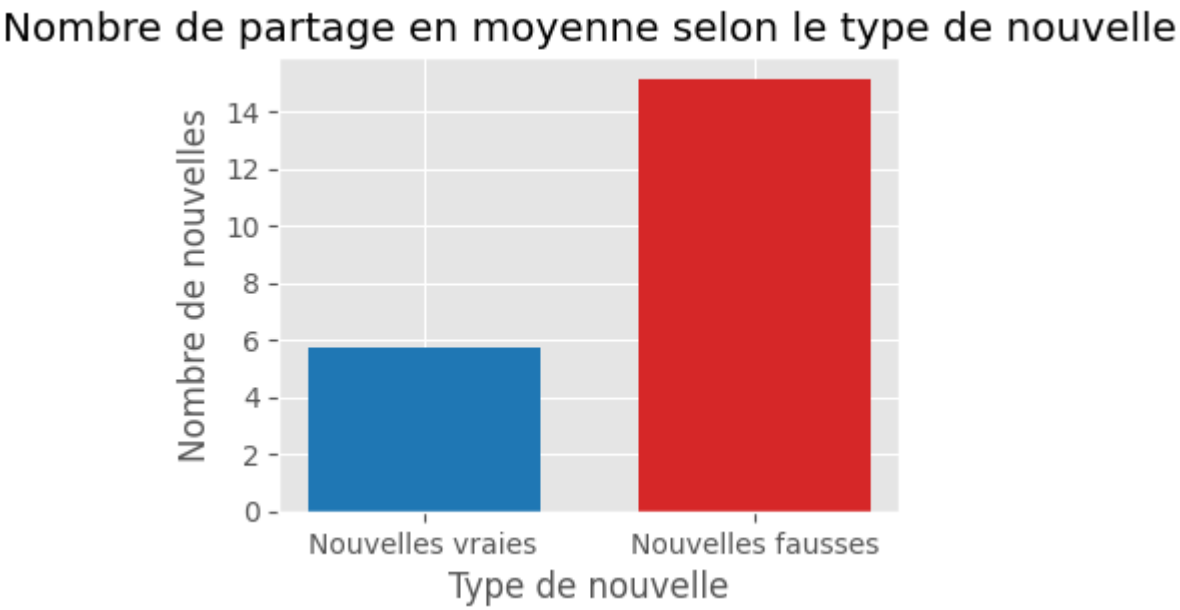
```
plt.rcParams['figure.figsize'] = [4, 3]
plt.rcParams['figure.dpi'] = 100

left = [1, 2]
height = [total_like_true_source/total_true, total_like_fake_source/total_false]

tick_label = ['Nouvelles vraies', 'Nouvelles fausses']

plt.bar(left, height, tick_label = tick_label, width = 0.7, color = ['tab:blue', 'tab:r

plt.xlabel('Type de nouvelle')
plt.ylabel('Nombre de nouvelles')
plt.title('Nombre de "Like" en moyenne selon le type de nouvelle')

plt.show()
```



Code affichant le nombre en moyenne de partage par type de nouvelles

In [ ]:

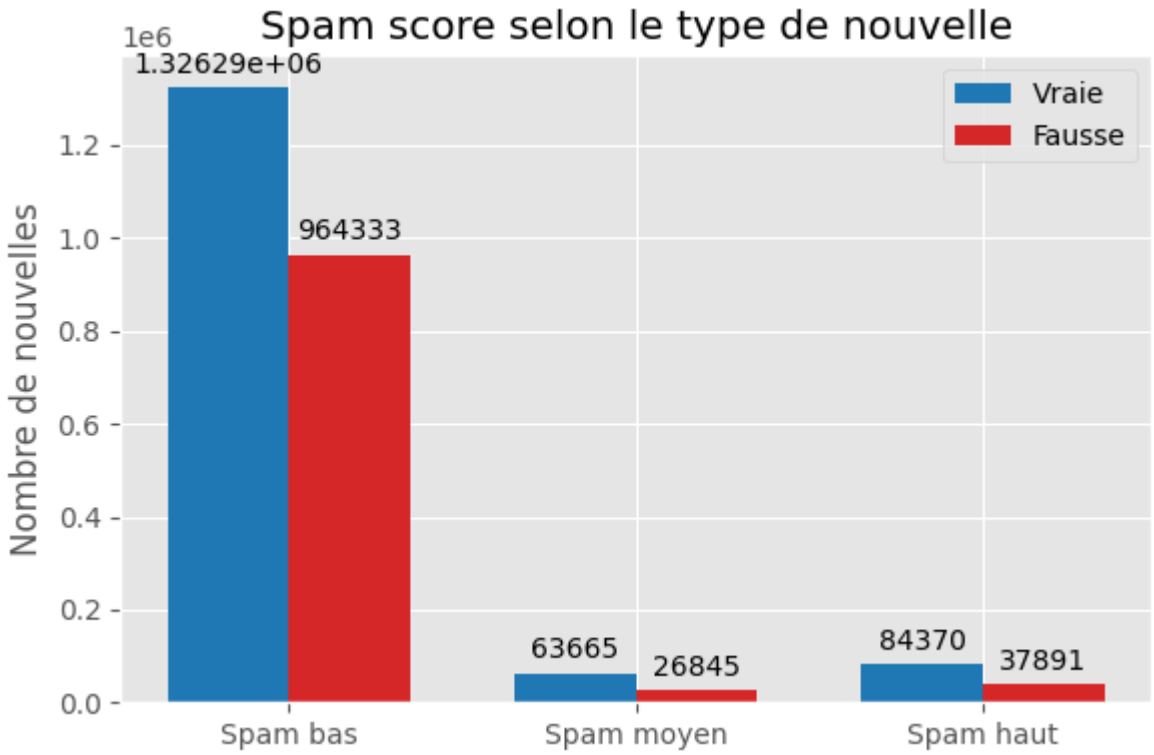
```
plt.rcParams['figure.figsize'] = [4, 3]
plt.rcParams['figure.dpi'] = 100

left = [1, 2]
height = [total_share_true_source/total_true, total_share_fake_source/total_false]

tick_label = ['Nouvelles vraies', 'Nouvelles fausses']

plt.bar(left, height, tick_label = tick_label, width = 0.7, color = ['tab:blue', 'tab:r

plt.xlabel('Type de nouvelle')
plt.ylabel('Nombre de nouvelles')
plt.title('Nombre de partage en moyenne selon le type de nouvelle')

plt.show()
```



Code affichant le spam score selon le type de source

In [ ]:

```
plt.rcParams['figure.figsize'] = [6, 4]
plt.rcParams['figure.dpi'] = 100

labels = ['Spam bas', 'Spam moyen', 'Spam haut']

spam_score_true = [total_vrai_bas_risque, total_vrai_medium_risque, total_vrai_haut_ris
spam_score_false = [total_faux_bas_risque, total_faux_medium_risque, total_faux_haut_ri

x = np.arange(len(labels))
width = 0.35

fig, ax = plt.subplots()
rects1 = ax.bar(x - width/2, spam_score_true, width, label='Vraie', color='tab:blue')
rects2 = ax.bar(x + width/2, spam_score_false, width, label='Fausse', color='tab:red')

ax.set_ylabel('Nombre de nouvelles')
ax.set_title('Spam score selon le type de nouvelle')
ax.set_xticks(x, labels)
ax.legend()

ax.bar_label(rects1, padding=3)
ax.bar_label(rects2, padding=3)

fig.tight_layout()

plt.show()
```



Conclusion

Afin d'obtenir les conclusions de nos recherches, ainsi que toutes nos sources, nous vous invitons à lire notre rapport complet: https://github.com/BernierS/Analyse_de_fausses_nouvelles