

Mémoire de projet de fin d'études

Pour l'obtention du diplôme d'Ingénieur d'État en Informatique

Option : Systèmes d'information et technologie

Détection des fausses nouvelles par apprentissage profond

Réalisé par :

Mlle. FERHATI Amina

Encadré par :

Mme. HAMDAD Leila

Ecole supérieure d'Informatique (ESI)

Mme. BERTI-EQUILLE Laure

Laboratoire d'Informatique et Systèmes (LIS)

Promotion : 2020/2021

Remerciements

Merci à Dieu, le tout puissant et miséricordieux, de nous avoir donné la force et la patience de mener à bien cet humble travail, et que nous prions de nous dresser et éclairer un chemin vers la réussite.

Je tiens à remercier mes encadrantes Mme. HAMDAD Leila et Mme. BERTI-EQUILLE Laure pour leurs précieuses orientations et leurs conseils tout au long de notre travail.

Merci également à la toute la famille de l'École Nationale Supérieure d'Informatique comptant ses enseignants, le corps administratif, pédagogique et employés. Cette école m'a offert une formation de qualité, des expériences tant ardues que bénéfiques.

Enfin, j'adresse mes plus sincères remerciements à tous mes camarades de l'ESI, mes proches et amis et à toute personne ayant contribué, de près ou de loin, à l'accomplissement de ce modeste projet.

Résumé

Depuis quelques années, et surtout depuis l'essor des médias sociaux, les fausses nouvelles sont devenues un problème de société, se répandant parfois plus et plus vite que les vraies informations. La grande propagation des fausses nouvelles a un immense impact sur la vie quotidienne des gens. Vu que ces nouvelles sont fausses, elles induisent les gens en erreur, elles les poussent à faire des jugements erronés sur des actions ou politiques importantes.

Dans ce rapport, les notions reliées aux fausses nouvelles ainsi que leurs méthodes de détection seront présentées afin d'atteindre des objectifs de recherche importants.

Vu que l'apprentissage profond a pris beaucoup d'importance ces dernières années dans le domaine des fausses nouvelles, grâce aux caractéristiques de ses réseaux de neurones, l'apprentissage profond a donné de très bons résultats de détection des fausses nouvelles. C'est pour cela, ce travail portera plus d'intérêt à la détection automatique basée sur la classification profonde des fausses nouvelles en exploitant le contenu textuel des nouvelles et leur source de provenance.

Mots-clé

Fausses nouvelles, Détection des fausses nouvelles, Apprentissage profond, Caractéristiques des fausses nouvelles.

Abstract

In recent years, and especially since the rise of social media, fake news has become a societal problem, sometimes spreading more and faster than real news. The wide spread of fake news has a huge impact on people's daily lives. Because it is fake, it misleads people into making wrong judgements about important actions or policies.

In this report, the concepts related to fake news as well as their detection methods will be presented in order to achieve important research goals. As deep learning has gained a lot of importance in recent years in the field of fake news, thanks to the characteristics of its neural networks, deep learning has given very good results in detecting fake news. For this reason, this work will focus on automatic detection based on deep classification of fake news by exploiting the textual content of the news and its origin,

Keywords

Fake news, Fake news detection, Deep learning, fake news Characteristics .

ملخص

في السنوات الأخيرة ، وخاصة منذ ظهور وسائل التواصل الاجتماعي ، أصبحت الأخبار المزيفة مشكلة اجتماعية ، وأحياناً تنتشر بشكل أسرع وأسرع من الأخبار الحقيقية. الانتشار الواسع للأخبار المزيفة له تأثير كبير على حياة الناس اليومية. بما أن هذه الأخبار خاطئة ، فإنها تضلل الناس ، وتجعلهم يصدرن أحكاماً خاطئة حول الإجراءات أو السياسات المهمة ، ويحققون أهدافاً بحثية مهمة. نظراً لتزايد أهمية التعلم العميق في السنوات الأخيرة في ساحة الأخبار المزيفة ، وبفضل خصائص شبكاتها العصبية ، أظهر التعلم العميق نتائج جيدة جداً في اكتشاف الأخبار المزيفة. لهذا السبب سيركز هذا العمل بشكل أكبر على الاكتشاف التلقائي بناءً على التصنيف العميق للأخبار المزيفة من خلال استغلال المحتوى النصي للأخبار وأصلها.

الكلمات المفتاحية:

اكتشاف الأخبار المزيفة ، التعلم تعلم العميق ، خصائص الأخبار المزيفة.

Table des matières

Introduction générale	2
1 Les fausses nouvelles et les attaques de désinformation	5
1.1 Définition des fausses nouvelles (En angl. Fake news)	6
1.2 Caractéristiques des fausses nouvelles	6
1.3 Types de fausses nouvelles	7
1.4 Impacts des fausses nouvelles	8
1.4.1 Impacts financiers	9
1.4.2 Impacts sur la santé	9
1.4.3 Impacts sur la société (intimidation et violence contre des innocents, idées racistes, etc.)	10
1.4.4 Impacts sur la politique	10
1.5 Challenges des fausses nouvelles	10
1.6 Apprentissage automatique adversaire (En angl. Adversarial machine learning)	11
1.7 Types d'attaques de l'apprentissage automatique adversaire	12
1.8 Attaques de désinformation	13
1.9 Risques de l'apprentissage automatique adversaire	13
2 Méthodes de détection des fausses nouvelles	15
2.1 Méthodes probabilistes	15
2.2 Vérification des faits par des experts (En angl. Fact-checking)	16
2.3 Production participative (En angl. Crowdsourcing)	17
2.4 Apprentissage automatique (En angl. Machine learning)	17
2.4.1 Apprentissage supervisé	18
2.4.2 Apprentissage non-supervisé	18
2.5 Apprentissage profond	19
2.5.1 Réseaux de neurones récurrents (RNN)	20
2.5.2 Réseaux de mémoire à court terme (LSTM)	20
2.5.3 Réseaux de neurones convolutifs (CNN)	21
2.5.4 VGGNet	22
3 Travaux existants	23
3.1 Travaux basés sur la classification non profonde des fausses nouvelles	23
3.1.1 Truth Inference in Crowdsourcing : Is the Problem Solved ? (GRANIK et al. 2017)	23
3.1.1.1 Modèle détaillé :	24

3.1.2	Truth Discovery against Strategic Sybil Attack in Crowdsourcing (Yue WANG et al. 2020)	24
3.1.2.1	Modèle détaillé :	25
3.2	Travaux basés sur la classification profonde des fausses nouvelles	26
3.2.1	Truth Discovery with Memory Network (LI et al. 2016)	26
3.2.1.1	Modèle détaillé	26
3.2.2	A Neural Network Approach for Truth Discovery in Social Sensing (MARSHALL et al. 2017)	27
3.2.2.1	Modèle détaillé	27
3.2.3	FAKEDETECTOR : Effective Fake News Detection with Deep Diffusive Neural Network (ZHANG et al. 2020)	28
3.2.3.1	Modèle détaillé	28
3.2.4	EANN : Event Adversarial Neural Networks for Multi-Modal Fake News Detection (Yaqing WANG et al. 2018)	29
3.2.4.1	Modèle détaillé	29
3.3	Synthèse	30
4	Conception	34
4.1	Comparaison entre les méthodes de détection des fausses nouvelles existantes	35
4.1.1	Collecte des données	35
4.1.2	Préparation des jeux de données	37
4.1.2.1	Données tabulaires	37
4.1.2.2	Données textuelles	38
4.1.3	Choisir l'environnement d'exécution	38
4.1.4	Méthodes de détection des fausses nouvelles existantes	38
4.1.4.1	Méthodes probabilistes	39
4.1.4.2	Méthodes d'apprentissage profond	40
4.1.5	Choix des métriques et les méthodes d'évaluation	47
4.2	Intégrer l'aspect source dans les méthodes de détection des fausses nouvelles	48
4.2.1	Aperçu sur les formules de calcul de la crédibilité des sources	49
4.2.2	Création de formules de calcul de crédibilité	49
4.2.3	Ajout de la composante source aux modèles d'apprentissage profond	50
4.3	Attaques de désinformation	50
4.3.1	La première version d'attaques	51
4.3.2	La deuxième version d'attaques	52
4.3.2.1	Reformulation mathématique de l'attaque	52
4.3.2.2	Méthode d'attaque	53
4.4	Construction du nouveau modèle de détection des fausses nouvelles pour les données textuelles	54
4.4.1	Architecture des modèle	54
4.4.2	Nettoyage des données	55
5	Réalisation, Tests et Résultats	57
5.1	Outils utilisés	57
5.2	Comparaison entre les méthodes de détection des fausses nouvelles	59
5.2.1	Collecte des données	59

5.2.2	Préparation des données	62
5.2.3	Méthodes de détection des fausses nouvelles existantes	66
5.2.4	Préparation de l'environnement de comparaison	68
5.2.4.1	Présentation du Cluster de LIS	68
5.2.5	Choix des métriques et des méthodes d'évaluation	72
5.3	Évaluation des méthodes de détection des fausses nouvelles existantes . . .	73
5.3.1	Interprétation des résultats de comparaison entre les méthodes d'apprentissage profond sur les données textuelles	74
5.4	Intégration de l'aspect source dans les modèles d'apprentissage profond . .	76
5.4.1	Interprétation des résultats de la première formule de calcul de crédibilité	76
5.5	Implémentation des attaques de désinformation	79
5.5.1	La première version de l'attaque	79
5.5.2	La 2-ème version de l'attaque	84
5.6	Implémentation du modèle d'apprentissage profond proposé pour la détection des fausses nouvelles	88
5.6.1	Les entrées du modèle	88
5.6.1.1	Nettoyage des entrées	88
5.6.1.2	Résultats obtenus	89
Conclusion générale		91
Annexe		96
5.7	Outils utilisé	96
5.7.1	Magic plot	96
5.7.2	Zotero	97
5.7.3	Drive	98
5.7.4	Organisation des réunions	98
5.7.5	MS project	98
5.8	Présentation de l'organisme d'accueil	99
5.9	Équipe de travail : DIAMS : Data Integration, Analysis, and Management as Services	100
5.9.1	Mots clés	100
5.9.2	Applications	100
5.9.3	Présentation	100

Table des figures

1.1	Types de fausses nouvelles.	8
1.2	Types détaillés de fausses nouvelles.	8
1.3	Chute de "Dow Jones" après le tweet (CNBC News, 2013).	9
1.4	Les principaux défis des fausses nouvelles.	11
1.5	Récapitulatif des attaques	12
2.1	Aperçu sur l'apprentissage supervisé (SAINT-CIRGUE 2019)	18
2.2	Aperçu sur l'apprentissage non supervisé (DJAMEL 2018)	19
2.3	Aperçu de l'architecture RNN (wiki)	20
2.4	Différence entre LSTM et RNN (PANJABI 2020)	21
2.5	Architecture du CNN	21
2.6	Architecture VGG16 (HASSAN 2018)	22
2.7	Architecture VGG19(JAWOREK-KORJAKOWSKA et al. 2019)	22
3.1	Framework proposé (GRANIK et al. 2017)	24
3.2	Aperçu de TDSSA (Yue WANG et al. 2020)	25
3.3	Architecture du modèle (LI et al. 2016)	27
3.4	Architecture du modèle (MARSHALL et al. 2017)	28
3.5	Architecture du modèle (ZHANG et al. 2020)	29
3.6	Architecture du modèle EANN (Yaqing WANG et al. 2018)	30
4.1	Processus global pour la comparaison des méthodes	35
4.2	Schéma explicatif des entités : objet, attribut et déclaration	37
4.3	Architecture du modèle LSTM 1 (Andrii Shchur, 2020)	41
4.4	Architecture du modèle LSTM 2 (Andrii Shchur, 2020)	42
4.5	Architecture du modèle LSTM 3 (Andrii Shchur, 2020)	43
4.6	Architecture du modèle Bi-LSTM	44
4.7	Architecture du modèle RNN	45
4.8	Architecture du modèle (MARSHALL et al. 2017)	46
4.9	Matrice de confusion	47
4.10	Ajout de la composante source aux entrées du modèle d'apprentissage profond	50
4.11	Première version de l'attaque	51
4.12	Entrées du modèle "Fake news detector"	54
4.13	Architecture du modèle proposé Fake news detector	55
5.1	Outils utilisés	58
5.2	Exemple du dataset "Kaggle"	59
5.3	Exemple du dataset "Snopes"	59
5.4	Exemple du dataset "Gossip"	60

5.5	Exemple du dataset "Politifact"	60
5.6	Exemple du dataset "PubHealth"	60
5.7	Exemple du dataset "Population"	61
5.8	Exemple du dataset "Weather"	61
5.9	Exemple dans du dataset "Flights"	62
5.10	Notebook utilisé pour le nettoyage du jeu de données "Kaggle" - Suppres- sion des valeurs nulles	63
5.11	Notebook utilisé pour le nettoyage du jeu de données "Kaggle" - Suppres- sion des doublons	64
5.12	Les 3 notebooks utilisés pour la préparation des données	64
5.13	Notebook de suppression des valeurs nulles du jeu de données "Population"	64
5.14	Notebook de suppression des doublons du jeu de données "Population"	64
5.15	Notebook de suppression des objets non vérité de terrain du jeu de données "Population"	65
5.16	Notebook utilisés pour l'encodage des objets du jeu de données "Popula- tion"	65
5.17	Notebook de filtrage du jeu de données "Population"	65
5.18	Notebooks utilisés pour le nettoyage des jeux des données	66
5.19	Notebooks méthodes d'apprentissage profond	67
5.20	Notebooks méthodes probabilistes	67
5.21	L'interface de Gitlab LIS Marseille	68
5.22	Documentation du Cluster	69
5.23	Les étapes d'installation des modules Python sur le Cluster	69
5.24	Les différents noeuds/machines virtuelles présents sur le Cluster	70
5.25	Conversion des notebooks en fichiers ".py"	70
5.26	Terminal de connexion au cluster	71
5.27	Exécution du modèle Bi-LSTM sur le cluster	71
5.28	Notebook de technique de validation K-folds	72
5.29	Résultat de K-folds avec K=10	72
5.30	La déconnexion du cluster	73
5.31	Accuracy/precision/recall/Fscore/ temps d'exécution (s) des méthodes d'ap- prentissage profond sur les données textuelles	73
5.32	Accuracy/precision/recall/Fscore/temps d'exécution (s) des méthodes des- tinées pour les données tabulaires	74
5.33	Exemples des sources qui existent dans le jeu de données "Kaggle"	77
5.34	Les résultats récupérés	77
5.35	Les résultats sur le jeu de données Kaggle et Covid-19	78
5.36	Les résultats sur le jeu de données Politifact et de PubHealth	78
5.37	Netbooks avec lesquels cette attaque a été implémentée	79
5.38	Comparaison entre le la taille de jeu de données et le nombre des "not"	80
5.39	Effet du nombre de "not" sur l'exactitude du modèle - Kaggle	80
5.40	Effet du nombre de "not" sur l'exactitude du modèle - PubHealth	81
5.41	Étude sur l'effet attaque sur l'exactitude - PubHealth	81
5.42	Étude sur l'effet attaque sur l'exactitude - Kaggle	82
5.43	Étude sur l'effet attaque sur l'exactitude - Gossip	82
5.44	Étude sur l'effet attaque sur l'exactitude - Covid-19	83

5.45	Étude sur l'effet attaque sur l'exactitude - Politifact	83
5.46	Profilage pour l'allégation ciblée	84
5.47	Exemple de profilage	84
5.48	Résultats de profilage	85
5.49	L'implémentation du copieur de sources malicieuses	86
5.50	Création de la source malicieuse	86
5.51	Stratégie de choix de la bonne valeur	86
5.52	L'ajout de la fausse allégation aux jeux de données	87
5.53	L'ajout de la fausse allégation aux jeux de données aux jeux de données avec un label mis à "0"	87
5.54	Notebook qui affiche le résultat après l'attaque	87
5.55	Entrées du modèle	88
5.56	Taille des entrées	88
5.57	Nettoyage des entrées	88
5.58	Résultats du modèle proposé "Fake news detector"	89
5.59	Exemple d'utilisation de l'outil Magic plot	96
5.60	Exemple d'utilisation de l'outil Zotero	97
5.61	Exemple d'utilisation de l'outil Drive	98
5.62	Comptes rendus des réunions	98
5.63	Exemple d'utilisation de l'outil MS project	98
5.64	Exemple d'utilisation de l'outil MS project	99
5.65	Diagramme de Gantt	99
5.66	Les pôles de recherche	100

Liste des tableaux

3.1	Synthèse détaillée des travaux lus.	32
3.2	Synthèse détaillée des travaux lus (suite).	33
4.1	Tableau de notations	52
5.1	Caractéristiques des jeux de données après le nettoyage	66
5.2	Caractéristiques des jeux de données après le nettoyage dédié aux méthodes d'apprentissage profond	66
5.3	Caractéristique de la machine choisie	70

INTRODUCTION GÉNÉRALE

Contexte

Depuis l'apparition du covid-19, les réseaux sociaux ont été envahis par les fausses informations, comme : la vidéo fallacieuse d'une personne mangeant une chauve-souris, des bilans exagérés, de prétendus remèdes miracle, etc. Autant de "fausses nouvelles" qui alimentent les craintes des populations. Ce qui a mis le terme de "fausses nouvelles" sous les projecteurs à nouveau depuis les élections présidentielles aux États-unis en 2016.

Les fausses nouvelles ont été définies par le New York Times comme "une histoire inventée avec l'intention de tromper, souvent pour un gain secondaire". Elles sont sans doute l'un des défis les plus sérieux auxquels est confronté le secteur de l'information aujourd'hui. La quantité des fausses nouvelles sur Internet est immense ce qui a poussé la communauté des informaticiens (la fouille des données, l'intelligence artificielle, etc.) à offrir une pléthore de techniques de détection automatique des fausses nouvelles. D'ailleurs, elles ont fait l'objet de plusieurs projets, tutoriels et compétitions, à titre d'exemple : "FNC" (Fake News Challenge)¹ en décembre 2016, tutoriel de "Fact-Checking, Fake News, Propaganda, and Media Bias : Truth Seeking in the Post-Truth Era"² présenté par Preslav Nakov et Giovanni Da San Martino en Novembre 16, 2020, etc.

Problématique

Les fausses nouvelles ont un impact énorme sur notre vie quotidienne vu que les décisions qu'on prend sont faites à bases des informations collectées et si ces informations sont fausses, ça peut provoquer beaucoup de dégâts, par exemple : durant l'épidémie du covid-19, plusieurs personnes ont consommé le gel désinfectant pour éviter cette épidémie,

¹<http://www.fakenewschallenge.org/>

²<https://propaganda.qcri.org/emnlp20-tutorial/>

suite aux déclarations du président Donald Trump, etc.

Il existe beaucoup de méthodes pour la détection des fausses nouvelles qui exploitent une ou plusieurs de leurs caractéristiques à savoir : le contenu textuel, la crédibilité de la source, les caractéristiques visuelles, etc. Dans ce rapport, on va se focaliser sur les méthodes d'apprentissage profond et les méthodes probabilistes, vu leur importance dans le domaine de la détection automatique de fausses nouvelles. Afin de mettre en évidence leur points forts et points faibles, nous allons les confronter les unes aux autres à travers une étude comparative.

Nous allons aussi voir l'importance de la caractéristique source en testant la performance des méthodes d'apprentissage profond sans et avec la caractéristique source.

Nous irons entamer, par la suite, l'un des grands risques qui entourent le domaine de la découverte de vérité (En angl. Truth Discovery) qui est les attaques de désinformation. Nous allons, plus concrètement, tenter de mener différents types d'attaques afin de tester la robustesse d'un modèle de détection des fausses nouvelles. A la lumière de l'étude et des expérimentations effectuées, pourrait-on proposer une solution informatique assez pertinente et robuste au problème de la détection des fausses nouvelles ?

Objectifs

Dans ce PFE qui est un projet de recherche, on a pour objectif de préparer les jeux de données et développer un environnement unique pour exécuter et comparer plusieurs méthodes existantes pour la détection de fausses nouvelles dans les réseaux sociaux et sur le Web afin d'avoir les résultats de l'étude comparative. Deuxièmement, développer un environnement de test permettant la génération de différents types d'attaque de désinformation pour étudier la robustesse des méthodes de détection des fausses nouvelles. Enfin, proposer une architecture d'apprentissage profond pour la détection de fausses nouvelles en se basant sur le contenu textuel et la provenance de la nouvelle.

Plan

Le présent rapport est une synthèse du travail de recherche fait en guise de réaliser les objectifs ci-dessus. Il contient les chapitres suivants :

- Le premier chapitre : généralités sur les fausses nouvelles et les attaques de désinformation, contenant toutes les notions et concepts de base qui permettent de cerner la suite du rapport.
- Le deuxième chapitre : méthodes de détection des fausses nouvelles, présentant brièvement toutes les méthodes existantes, mais en portant un intérêt aux méthodes profondes en détaillant les concepts de l'apprentissage profond, étant un champ d'étude de l'intelligence artificielle.
- Le troisième chapitre est dédié à faire la liaison entre les l'étude théorique des méthodes de détection des fausses nouvelles faite dans le chapitre précédent et les fausses nouvelles. Une synthèse sur les travaux existants sera présentée à la fin.

- Le quatrième chapitre est dédié à la conception faite afin d'atteindre les objectifs du PFE.
- Le cinquième chapitre est dédié pour expliquer les différentes étapes d'implémentation de la conception, les résultats des tests ainsi que l'interprétation de ces derniers.

CHAPITRE

1

LES FAUSSES NOUVELLES ET LES ATTAQUES DE DÉSINFORMATION

Introduction

Le terme “fausses nouvelles” a gagné en popularité au cours de ces dernières années surtout après les élections présidentielles aux états unis en 2016. Elles ont été utilisées à maintes reprises dans les campagnes politiques, des deux côtés de l’allée politique, tant pour soutenir que pour attaquer. Elles ont été largement partagées sur les réseaux sociaux ce qu’il les a rendues populaires et influentes. C’est pour cela qu’un certain nombre de commentateurs ont suggéré que Donald Trump n’aurait pas été élu président sans l’influence des fausses nouvelles vu qu’elles ont eu tendance à le favoriser par rapport à Hillary Clinton. Donc, qu’est-ce que c’est que les fausses nouvelles? Et quels sont leurs impacts?

Le chapitre présenté ci-dessous, a pour but aussi de étudier la vulnérabilité des approches d’apprentissage automatique qui ont pour but de détecter les fausses nouvelles, c’est pour cela que le concept d’ Apprentissage automatique adversaire a été introduit, ce dernier est reparti en plusieurs types selon le mode d’attaque (Temps, But ou information), néanmoins ce concept pourra présenter des risques pouvant s’avérer fatal au domaine de la découverte de la vérité (En angl. Truth Discovery).

1.1 Définition des fausses nouvelles (En angl. Fake news)

La tâche de définir les fausses nouvelles est un défi en soi-même, car il existe beaucoup de définitions, mais aucune n'est standard. Pour bien comprendre ce que sont les fausses nouvelles, on doit tout d'abord bien comprendre le terme "nouvelles". Les nouvelles sont souvent considérées comme un produit du journalisme, un domaine censé fournir "des informations indépendantes, fiables, précises et complètes" (YASMIN et al. 2020).

Si les nouvelles sont supposées être fiables et fondées sur la vérité, la précision et l'objectivité, comment est-il possible d'avoir une "fausse" nouvelle ? le mot "faux", selon le dictionnaire d'Oxford, est un adjectif qui signifie "non authentique ; imitation ou contre-façon", pourtant cela est possible, le terme "fausses nouvelles" n'est pas nouveau du tout, il a existé depuis toujours pour servir aux intérêts de l'homme : divertir et aider à vendre plus d'exemplaires des journaux, etc.

L'une des définitions les plus répandues et les plus acceptées que l'on peut trouver dans la littérature est la suivante : "les fausses nouvelles sont des articles de presse qui sont intentionnellement et de manière vérifiable faux, et qui pourraient induire les lecteurs en erreur"(ALLCOTT et al. 2017).

Selon (ZHANG et al. 2020), "Le terme général de fausse nouvelle est utilisé pour désigner les faux articles, créateurs et sujets ", cette définition permet de voir les caractéristiques de ces nouvelles qui sont rien d'autre que l'union de trois entités : créateur, sujet et article.

1.2 Caractéristiques des fausses nouvelles

En fait, il y a beaucoup de caractéristiques essentielles qu'on peut extraire à partir de la majorité des définitions des fausses nouvelles :

- Authenticité (WARDLE 2017) : fait référence à la mesure dans laquelle les fausses nouvelles s'appuient sur des faits.
- Intention (WARDLE 2017) : fait référence au degré dans lequel le créateur des fausses nouvelles a l'intention d'induire en erreur ou de tromper le public ex. Des fois, l'auteur ne vise pas à tromper le public, mais plutôt à le divertir.
- Nouveauté : les événements nouvellement apparus génèrent souvent de nouvelles connaissances qui n'ont pas été sauvegardées déjà ou qui sont difficiles à déduire (Yaqing WANG et al. 2018).
- Domaine : les caractéristiques qui ont bien représenté des fausses nouvelles dans un domaine, peuvent ne pas être aussi utiles dans les autres domaines.
- Style d'écriture trompeur : peut-être que l'information ne présente qu'une partie de l'histoire (information biaisée). Par exemple, les trompeurs produisent plus de mots

au total et ces mots montrent généralement une complexité cognitive inférieure et des émotions extrêmes.

- **Crédibilité de la source** : il faut s'assurer que la source a la formation universitaire ou l'expérience nécessaires pour écrire avec autorité sur le sujet.
- **Date de publication** : l'information change dans certains domaines et doit donc être actualisée. Beaucoup d'informations non-mises à jour sont utilisées dans le but de tromper le public.
- **Contexte social** : on peut enquêter et utiliser les informations liées à la diffusion de fausses nouvelles, par exemple, la manière dont les utilisateurs les diffusent et qui sont les utilisateurs qui ont partagé ou bien réagi à ces fausses nouvelles.
- **Caractéristiques visuelles** (Yaqing WANG et al. 2018) : généralement, on trouve toujours des images/vidéos attachées au texte, parfois une fausse nouvelle ne peut pas être détectée par son texte, mais par les images/les vidéos attachées à elle.

1.3 Types de fausses nouvelles

On peut définir les types des fausses nouvelles de plusieurs manières (tout dépend des caractéristiques utilisées.). Dans notre cas, comme le montre la figure 1.1, on se basera sur deux caractéristiques principales : l'authenticité et l'intention (WARDLE 2017) :

- **Mésinformation** : c'est des erreurs involontaires mais qui ne sont pas destinées à causer un préjudice, telles que des photos, des dates, des statistiques, des traductions inexactes ou lorsque la satire est prise au sérieux.
- **Désinformation** : c'est un contenu audio/visuel fabriqué ou délibérément manipulé dans le but exprès de causer un préjudice. Les producteurs de désinformations ont généralement des motivations politiques, financières, psychologiques ou sociales. Comme les théories de conspiration créées intentionnellement et les rumeurs.
- **Malinformation** : c'est une information authentique qui est partagée pour causer un préjudice. Cela comprend les informations privées ou révélatrices qui sont diffusées pour l'intérêt personnel : nuire à une personne ou sa réputation ou celui de l'entreprise plutôt que pour l'intérêt public. Comme le changement délibéré du contexte, de la date ou de l'heure du contenu authentique.

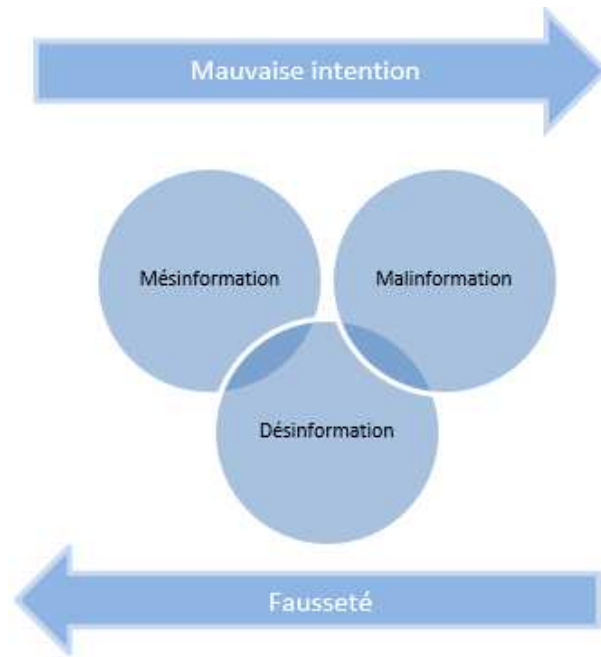


FIG. 1.1 : Types de fausses nouvelles.

À partir de ces 3 catégories majeures, on pourra classer tous les autres types de fausses nouvelles qui existent, comme le montre la figure 1.2 :

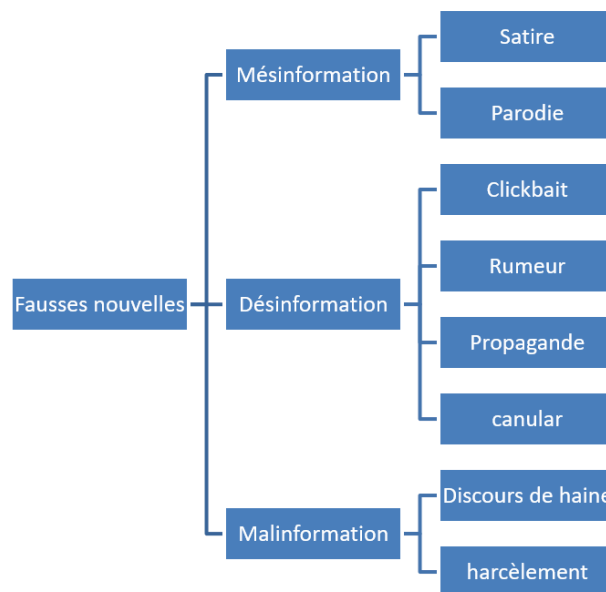


FIG. 1.2 : Types détaillés de fausses nouvelles.

1.4 Impacts des fausses nouvelles

Les fausses nouvelles ont des impacts importants, car l'information façonne notre manière de penser : les gens prennent des décisions en se basant sur les informations

disponibles. Ainsi, si ces informations sont inventées, fausses, exagérées ou déformées, d'énormes dégâts peuvent être causés.

1.4.1 Impacts financiers

En avril 2013, l'agence de presse mondiale "Associate Press" a envoyé un tweet prétendant qu'il y avait eu deux explosions à la Maison Blanche et que le président de l'époque, Barack Obama, avait été blessé. "Associate Press" a 2 millions de followers sur Twitter : c'est une source très estimée, donc cette nouvelle a été rapidement crue. Le tweet est sorti à 13 h 07 et à 13 h 08, le "Dow Jones" (indice de bourses) a chuté de 150 points, avant de se stabiliser à 13 h 10.

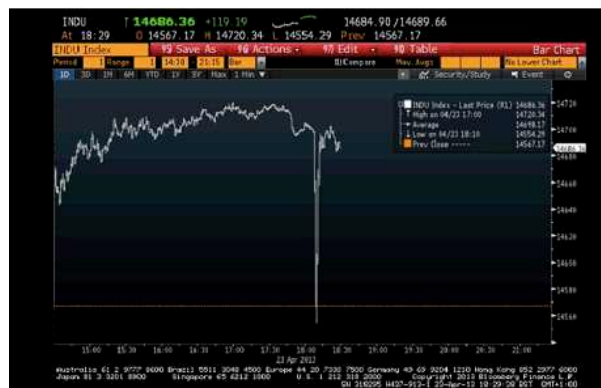


FIG. 1.3 : Chute de "Dow Jones" après le tweet (CNBC News, 2013).

1.4.2 Impacts sur la santé

L'organisation mondiale de la santé (OMS) a annoncé le 8 avril 2020 ce qui suit sur le covid-19 :

" Pour contribuer à atténuer les souffrances et à sauver des vies, l'OMS travaille jour et nuit dans cinq domaines clés :

1. Aider à renforcer la capacité des pays à se préparer et à réagir.
2. Fournir des informations précises et lutter contre l'infodémie, en collaboration avec de nombreux partenaires,etc. "

On voit clairement que le deuxième objectif de l'OMS est de combattre les fausses nouvelles concernant le covid-19 vu qu'elles ont eu des effets considérables tant sur la conformité du public aux recommandations de prévention des infections que sur les réponses nationales mises en œuvre par les pays.

1.4.3 Impacts sur la société (intimidation et violence contre des innocents, idées racistes, etc.)

En 2018, les Indiens ont reçu une vidéo qui montre le corps mutilé d'un enfant sur "WhatsApp". On ne sait pas exactement d'où la vidéo provient, ni si elle a été trafiquée. Une voix implore les gens de la transmettre à d'autres et de rester vigilants. Les messages ont poussé les parents à garder leurs enfants à l'intérieur. Les enseignants ont signalé une baisse énorme de la fréquentation scolaire.

1.4.4 Impacts sur la politique

Plus de la moitié des Européens pourraient avoir été témoins d'une forme de désinformation promue par des acteurs russes sur les réseaux sociaux à l'approche des élections parlementaires en 2019, selon une analyse faite par 'POLITICO'. L'objectif de ces efforts, selon l'analyse, est d'amplifier les questions qui divisent les pays européens afin de saper les institutions démocratiques et de créer des tensions internes d'une manière qui, en fin de compte, favorise l'état russe.

1.5 Challenges des fausses nouvelles

Les fausses nouvelles présentent beaucoup de défis qui sont résumés ci-dessous et dans la figure 1.4 :

- La définition du terme "fausses nouvelles" est un challenge en soi-même, car il y a beaucoup de définitions et chacune traite une caractéristique ou plusieurs des fausses nouvelles.
- Les caractéristiques variantes des fausses nouvelles : déjà citées au niveau de la section 1.2.
- L'audience : ce sont les personnes qui sont exposées aux fausses nouvelles et qui sont susceptibles de les croire et ils peuvent même participer à la diffusion de ces dernières. L'audience présente un challenge important, car :
 - Les gens ont tendance à croire ce qu'il leur plaît/ce qui est proche de leurs valeurs ou bien ce que leurs amis partagent.
 - Dans l'ère de l'infobésité, la majorité des gens n'ont plus un esprit critique donc ils ont tendance à croire tout ce qu'ils lisent.
 - Il y'a aussi les utilisateurs influencés par "l'opinion de la foule", donc, ils ont tendance à croire les publications qui ont beaucoup de "j'aimes/retweets/commentaires confirmant la publication".
 - La rareté des informations sur les personnes diffusant les fausses nouvelles sur les plateformes numériques constitue un défi important dans le profilage des fausses nouvelles.

- Le manque d'informations peut avoir un impact négatif sur la performance des méthodes automatiques de détection des fausses nouvelles et même les ressources disponibles ne pourraient pas aider à porter un jugement novateur sur les propriétés pertinentes des fausses nouvelles et à construire des modèles pouvant fonctionner correctement dans un scénario où les données arrivent en streaming.

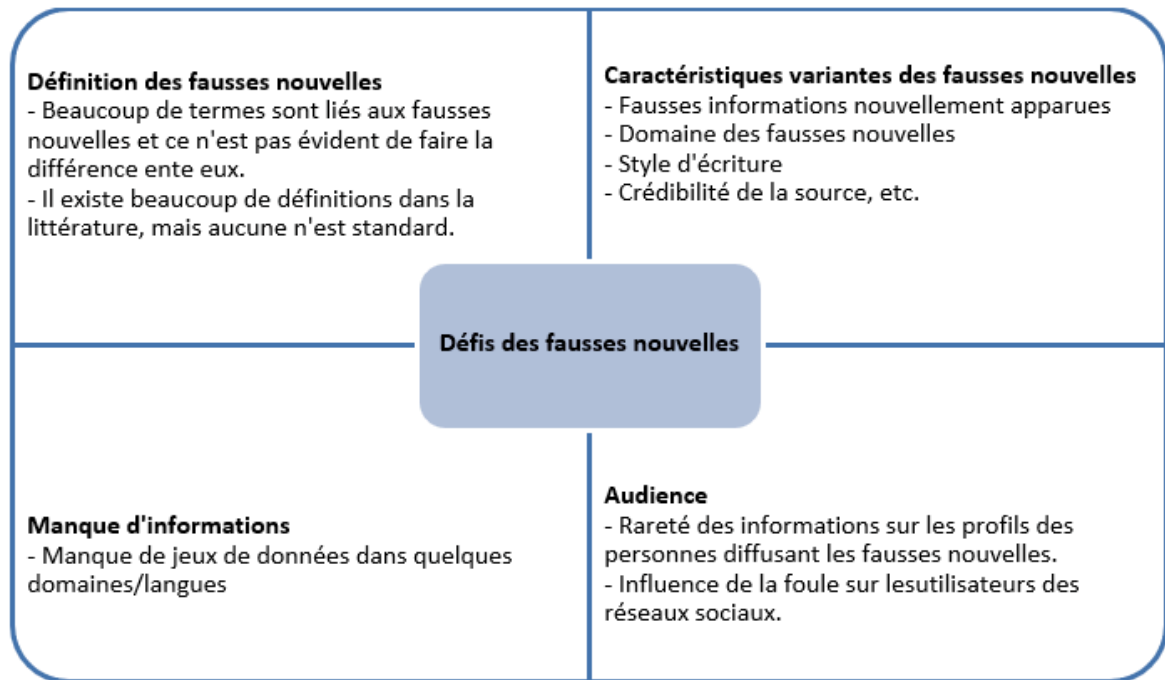


FIG. 1.4 : Les principaux défis des fausses nouvelles.

Vu qu'on a pour but de mener des attaques pour savoir les vulnérabilités des modèles de détection des fausses nouvelles et comment implémenter des modèles de défense contre ces attaques, on va présenter plus en détail dans la section suivante ces attaques et leurs types.

1.6 Apprentissage automatique adversaire (En angl. Adversarial machine learning)

Le domaine de l'apprentissage automatique adversaire est apparu pour étudier les vulnérabilités des approches d'apprentissage automatique et pour développer des techniques permettant de rendre les modèles d'apprentissage robustes aux manipulations (VOROBAYCHIK 2018).

C'est une technique d'apprentissage automatique qui tente de tromper les modèles en fournissant des données trompeuses. La raison la plus courante est de provoquer un dysfonctionnement dans un modèle d'apprentissage automatique.

1.7 Types d'attaques de l'apprentissage automatique adversaire

Il existe plusieurs types d'attaques. On peut classer ces dernières selon 3 dimensions (VOROBAYCHIK 2018) : le temps, le but et l'information.

- Le temps : la première chose à laquelle on pense c'est quand est ce que l'attaque prend place. Dans ce cas, il y a deux possibilités :
 - Temps d'entraînement (En angl. Poisoning attacks) : on mène cette attaque quand le modèle apprend sur les données d'entraînement.
 - Temps de décision (En angl. Evasion attacks) : Cette attaque effectuée lors de la phase "test" quand le modèle prédit les sorties sur les données de test.
- Le but : généralement le but de l'attaquant est de réduire l'exactitude du modèle (maximiser le taux de prédictions fausses), mais il peut aller plus loin, par exemple, si l'attaquant veut réduire la crédibilité d'une source, il va falloir attaquer toutes les nouvelles qu'elle rapportent et changer leur label.
- L'information : On distingue deux cas, le premier cas, "the white box" : on assume que l'attaquant connaît tout (le modèle, les données, les paramètres, etc.), le deuxième cas, "the black box" : l'attaquant a une connaissance limitée.

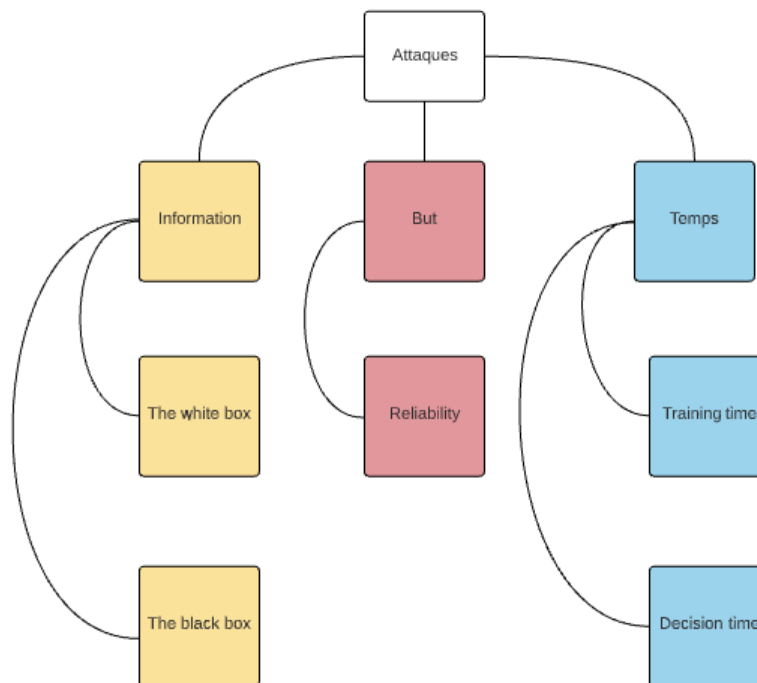


FIG. 1.5 : Récapitulatif des attaques

1.8 Attaques de désinformation

Les attaques de désinformation sont la diffusion intentionnelle de fausses nouvelles, dans le but de tromper, de confondre ou de manipuler un public.

Les attaques de désinformation peuvent être exécutées par des acteurs étatiques ou non étatiques pour influencer les populations nationales ou étrangères. Ces attaques sont généralement employées pour modifier les attitudes et les croyances, promouvoir un programme particulier ou susciter certaines actions de la part d'un public cible.

Il y a différentes manières pour mener ces attaques de désinformation et l'une d'entre elles est en utilisant l'apprentissage automatique adversaire qui est une technique très avancée et qui peut endommager beaucoup de modèles de détection des fausses nouvelles, surtout que la majorité de ces modèles utilisent l'apprentissage automatique.

1.9 Risques de l'apprentissage automatique adversaire

Les chercheurs en cybersécurité craignent que les attaques adverses deviennent un problème grave à l'avenir, à mesure que l'apprentissage automatique est intégré dans un éventail plus large de systèmes, notamment les voitures à conduite autonome et d'autres technologies où des vies humaines pourraient être en danger.

Dans ce travail (Yue WANG et al. 2020) qui sera présenté en détail dans le chapitre "travaux existants", nous pouvons voir ce que les attaques peuvent produire comme dégâts sur les plateformes de la découverte de vérité. Malgré que ces attaques ne sont pas basées sur l'apprentissage automatique adversaire, on peut toujours l'inclure dans le cadre des attaques de désinformation.

Conclusion

Ces dernières années ont vu l'essor des médias sociaux, qui ont permis aux gens de partager facilement des nouvelles avec un grand nombre d'utilisateurs en ligne, sans contrôle de qualité. Le bon côté des choses, c'est que tout le monde peut devenir un créateur de contenu et que la diffusion des informations est beaucoup plus rapide. Le mauvais côté, c'est que cela a également permis aux acteurs malveillants de diffuser plus rapidement les informations erronées, qui peuvent atteindre un public très large.

En général, la lutte contre la désinformation n'est pas facile ; comme dans le cas du spam, il s'agit d'un problème compliqué, où les acteurs malveillants changent et améliorent constamment leurs stratégies surtout avec l'arrivée de l'apprentissage automatique adversaire.

Vu que les fausses nouvelles n'ont pas de forme spécifique pouvant servir à la mise en place d'un modèle de détection automatique. De là, apparaissent plusieurs défis relativement à l'absence des règles régissant le contenu textuel d'une fausse nouvelle.

Dans la suite, le second maillon nécessaire à la bonne compréhension des travaux existants relatifs à la détection des fausses nouvelles, sera élaboré notamment les notions

de l'apprentissage profond.

CHAPITRE

2

MÉTHODES DE DÉTECTION DES FAUSSES NOUVELLES

Introduction

En raison du développement rapide des fausses nouvelles et de la complexité de sa résolution, beaucoup de méthodes ont été proposées pour les détecter. Les principales approches seront présentées dans ce chapitre de manière abstraite et puis on va faire le lien entre ces méthodes et les fausses nouvelles dans le troisième chapitre "travaux existants", comme : les méthodes probabilistes, la vérification des faits, la production participative, l'apprentissage automatique et l'apprentissage profond tout en portant plus d'intérêt à l'approche d'apprentissage profond vu son importance pour la détection des fausses nouvelles.

2.1 Méthodes probabilistes

Ce sont des méthodes qui utilisent des modèles probabilistes afin de détecter les labels des données tabulaires. Selon (BERTI 2018), on peut les organiser en trois catégories majeures

- Les méthodes basées sur le vote pondéré calculent les étiquettes de valeurs (c'est-à-dire vrai ou faux) et de confiance en utilisant une variante du vote majoritaire (MV : En angl. Majority voting).
- Les méthodes bayésiennes et probabilistes graphiques, ce sont des méthodes basées sur des modèles bayésiens et graphiques qui ont été proposées pour régler les pro-

blèmes de MV principalement liés aux propriétés latentes (inconnues) des sources et des nouvelles.

- Les méthodes basées sur l'optimisation reposent sur la définition de la fonction d'optimisation qui peut capturer les relations entre les qualités des sources et la vérité des affirmations, avec une méthode itérative afin de calculer ces deux ensembles de paramètres conjointement.

2.2 Vérification des faits par des experts (En angl. Fact-checking)

Les vérificateurs de faits ou bien les experts (En angl. Fact-checkers) sont un petit groupe de professionnels de diverses disciplines qui sont capables de vérifier la véracité de certaines nouvelles et de décider si ces informations sont fausses ou authentiques (ZHOU et al. 2018). Cependant, certains vérificateurs de faits professionnels ne sont pas indépendants et travaillent pour une organisation.

Voici quelques exemples de sites de vérification des faits :

- Snope¹ : traite les faits politiques et sociaux.
- Hoaxslayer² : se concentre sur divers domaines, dont la santé, la religion et l'économie.
- PolitiFact³ : se concentre principalement sur les questions politiques aux États-Unis.

Parmi les points positifs de cette approche :

- Les experts sont peu nombreux, donc faciles à gérer.
- Les experts ont un taux de précision très élevé.

Cette approche a aussi des points négatifs :

- L'approche est très lente, car le processus est manuel et les experts doivent vérifier un grand nombre d'informations.
- La plupart des personnes chargées de vérifier les faits ont souvent été vivement critiquées pour être biaisées à cause de leur caractère politique.

¹<https://www.snopes.com/>

²<https://www.hoax-slayer.net/>

³<https://www.politifact.com/>

2.3 Production participative (En angl. Crowdsourcing)

Le crowdsourcing consiste littéralement à externaliser une activité vers la foule c'est-à-dire vers un grand nombre d'acteurs anonymes. Cette approche est apparue comme un nouveau paradigme de résolution de problèmes, qui facilite la résolution de problèmes difficiles pour les ordinateurs, par exemple la résolution d'entités et l'analyse des sentiments (ZHENG et al. 2017).

Voici quelques exemples de sites de vérification des faits :

- MTurk ⁴ : Amazon Mechanical Turk est un marché de crowdsourcing qui permet aux particuliers et aux entreprises d'externaliser plus facilement leurs processus et leurs travaux à une main-d'œuvre distribuée qui peut effectuer ces tâches virtuellement. Cela peut aller de la simple validation de données et de la recherche à des tâches plus subjectives comme la participation à des enquêtes, la modération de contenu, etc.
- Zhongbao ⁵ : est une plate-forme de crowdsourcing bilingue développée indépendamment par le groupe de base de données du Département d'informatique de l'Université Tsinghua. Elle est destinée à aider les entreprises et les utilisateurs à résoudre des problèmes auxquels les machines ont du mal à répondre, tels que les codes de vérification d'image, les tâches d'étiquetage , etc.

Parmi les points positifs de cette approche :

- L'approche est non-coûteuse

Cette technique a des points négatifs aussi, comme :

- Ces deux méthodes se basent sur le vote, ce qui rend facile la domination du résultat de l'agrégation par des sources malveillantes (Yue WANG et al. 2020).
- Impossible de faire confiance à tous les travailleurs parce qu'ils ne sont pas spécialisés dans tous les domaines.

2.4 Apprentissage automatique (En angl. Machine learning)

Le Machine Learning ou apprentissage automatique est une sous-catégorie de l'intelligence artificielle. Elle consiste à laisser des algorithmes découvrir des " patterns ", à savoir des motifs récurrents, dans les ensembles de données.

Les algorithmes de Machine Learning apprennent de manière autonome à réaliser des prédictions à partir des caractéristiques des données et ils améliorent leurs performances

⁴<https://www.mturk.com/>

⁵<http://www.chinacrowds.com/>

au fil du temps.

Parmi les types les plus connus d'algorithmes dans l'apprentissage automatique, on trouve :

2.4.1 Apprentissage supervisé

L'apprentissage supervisé (En angl. supervised learning) consiste en des variables d'entrée x et une variable de sortie y . Il utilise un algorithme pour apprendre la fonction de mapping qui permet d'associer à chaque entrée une sortie : $y = f(x)$. Il peut prédire les variables de sortie y pour ces données. On introduit aussi la fonction perte qui mesure la différence entre la vraie réponse y' et la réponse $f(x)$ fournie par l'algorithme.

$$L(y', f(x)) = \begin{cases} 0 & \text{if } y = y' \\ 1 & \text{if } y \neq y' \end{cases} \quad (2.1)$$

On distingue deux catégories d'algorithmes d'apprentissage supervisé :

- Classification : ça correspond à déterminer un modèle de données étiquetées. Par exemple : les arbres de décision, ce sont un outil d'aide à la décision représentant un ensemble de choix sous la forme graphique d'un arbre. Les différentes décisions possibles sont situées aux extrémités des branches, et sont atteintes en fonction de décisions prises à chaque étape.
- Régression : consiste à prédire l'étiquette d'une nouvelle donnée, connaissant le modèle préalablement appris. Par exemple : L'analyse discriminante est un exemple typique. Il s'agit d'expliquer et de prédire l'appartenance d'un individu à une classe prédéfinie à partir de ses caractéristiques, mesurées à l'aide de variables prédictives.

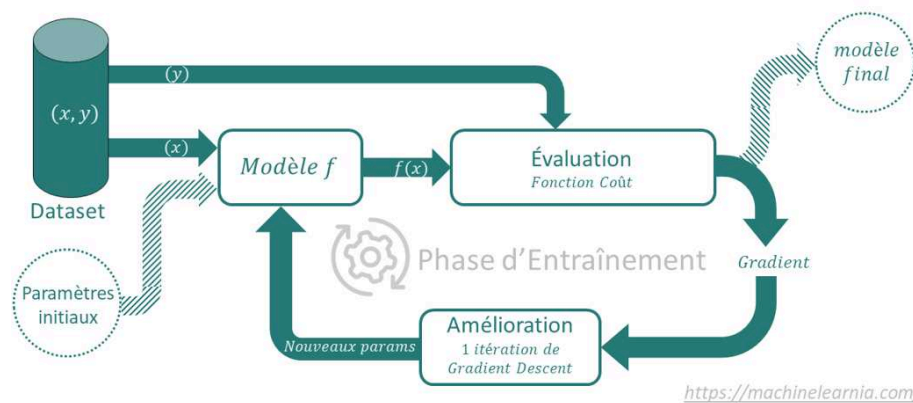


FIG. 2.1 : Aperçu sur l'apprentissage supervisé (SAINT-CIRGUE 2019)

2.4.2 Apprentissage non-supervisé

L'apprentissage non supervisé (Unsupervised Learning) consiste à ne disposer que d'entrées X et pas de variables de sortie correspondantes. L'objectif de l'apprentissage non supervisé est de modéliser la structure ou la distribution sous-jacente à ces données non étiquetées afin d'en apprendre davantage sur ces dernières.

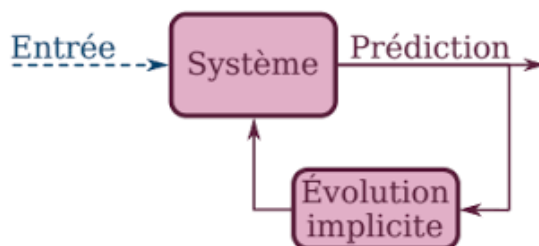


FIG. 2.2 : Aperçu sur l'apprentissage non supervisé (DJAMEL 2018)

Voici une liste de certains algorithmes d'apprentissage automatique non supervisés les plus connus :

- K-means clustering : c'est une méthode de partitionnement de données et un problème d'optimisation combinatoire. Étant donnés des points et un entier k , le problème est de diviser les points en k groupes, souvent appelés clusters, de façon à minimiser une certaine fonction.
- Association : elle consiste à découvrir des relations intéressantes entre des variables dans de grandes bases de données. Par exemple, les personnes qui achètent une nouvelle maison ont aussi tendance à acheter de nouveaux meubles. Il découvre la probabilité de co-occurrence d'éléments dans une collection.

Parmi les points positifs de l'approche d'apprentissage automatique sont :

- En général, ces modèles ont une bonne performance.
- C'est une approche automatique, donc, pas besoin d'intervention humaine.

Les points négatifs de cette approche sont :

- Les problèmes des modèles de Machine Learning (Tunning des paramètres, overfitting, etc.).

2.5 Apprentissage profond

Le terme d'apprentissage profond (DL) a été introduit pour la première fois dans la communauté de l'apprentissage machine par Dechter (1986). Le DL est une technique émergente qui est utilisée dans diverses applications, notamment la vision par ordinateur, la reconnaissance de la parole, le traitement du langage naturel, la détection d'anomalies, etc. Aujourd'hui, il est de plus en plus utilisé pour le traitement des données et la création de modèles d'aide à la décision.

Le nombre d'architectures et d'algorithmes utilisés dans l'apprentissage profond est vaste et varié. Cette section explore des exemples d'architectures d'apprentissage profond les plus connues au cours des 20 dernières années.

2.5.1 Réseaux de neurones récurrents (RNN)

Un réseau de neurones récurrents est constitué de neurones interconnectés et pour lequel il existe au moins un cycle dans la structure. Les neurones sont reliés par des arcs qui possèdent un poids. La sortie d'un neurone est une combinaison de ses entrées et des sorties calculées précédemment. Ainsi, les RNN sont utilisés pour les applications où les informations historiques/séquentielles sont importantes. Ces réseaux nous aident à prévoir les séries chronologiques dans les applications commerciales et à prévoir les mots dans les applications de type chatbot.

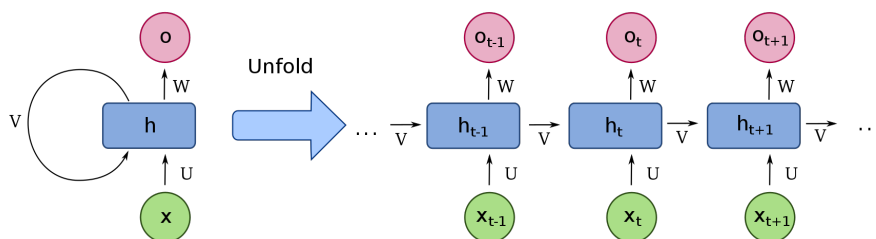


FIG. 2.3 : Aperçu de l'architecture RNN (wiki)

2.5.2 Réseaux de mémoire à court terme (LSTM)

L'un des types les plus courants de modèle RNN est le réseau de mémoire à court terme (LSTM). Contrairement aux réseaux neuronaux feedforward ⁶ standard, le LSTM possède des connexions de rétroaction. Il peut traiter non seulement des points de données uniques (comme des images), mais aussi des séquences entières de données (comme la parole ou la vidéo). Par exemple, LSTM est applicable à des tâches telles que la reconnaissance d'écriture non segmentée et connectée et la reconnaissance vocale.

Les RNNs n'ont pas de cellules d'état (En ang. cell state). Ils ont seulement des cellules cachées et ces cellules servent de mémoire pour les RNN. Les LSTM, quant à eux, possèdent à la fois des cellules d'état et cachées. Les cellules d'état ont la capacité de supprimer ou d'ajouter des informations au modèle, régulée par des "portes" (En angl. gates). Et grâce à cette "cellule", le LSTM devrait être capable de gérer la dépendance à long terme.

⁶Le réseau de neurones à propagation avant est le premier type de réseau neuronal artificiel conçu. C'est aussi le plus simple. Dans ce réseau, l'information ne se déplace que dans une seule direction, vers l'avant, à partir des nœuds d'entrée, en passant par les couches cachées (le cas échéant) et vers les nœuds de sortie. Il n'y a pas de cycles ou de boucles dans le réseau.

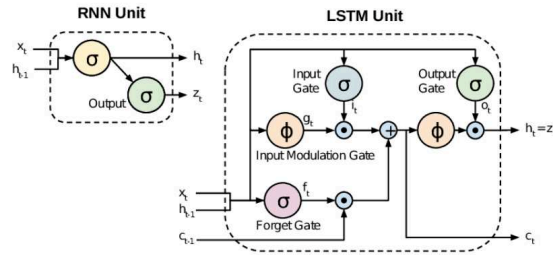


FIG. 2.4 : Différence entre LSTM et RNN (PANJABI 2020)

2.5.3 Réseaux de neurones convolutifs (CNN)

Un réseau neuronal convolutif est un réseau de neurones de type feed-forward qui est généralement utilisé pour analyser des images visuelles en détectant et classant les objets dans une image, tout cela est possible en traitant des données avec une topologie en forme de grille. Il est également connu sous le nom de ConvNet. Les quatre couches importantes de CNN sont :

- Convolution layer : Il s'agit de la première étape du processus d'extraction des caractéristiques d'une image. Une couche de convolution a plusieurs filtres qui effectuent l'opération de convolution. Chaque image est considérée comme une matrice de valeurs de pixels. Un filtre peut être appliqué pour déterminer les bords d'angle, les bords verticaux et bords horizontaux, etc.
- ReLU layer : Une fois les cartes d'entités extraites, l'étape suivante consiste à les déplacer vers une couche ReLU. ReLU effectue une opération élément par élément et définit tous les pixels négatifs sur 0. Il introduit une non-linéarité sur le réseau et la sortie générée est une carte d'entités rectifiée. Voici le graphique d'une fonction ReLU :
- Pooling layer : Le pooling est une opération de sous-échantillonnage qui réduit la dimensionnalité des features map. La rectified feature map passe maintenant par une couche de regroupement pour générer a pooled feature map.
- Fully connected layer : l'étape suivante du processus s'appelle flattening. Flattening est utilisé pour convertir tous les tableaux bidimensionnels résultants à partir pooled feature maps en un seul vecteur linéaire continu et long.

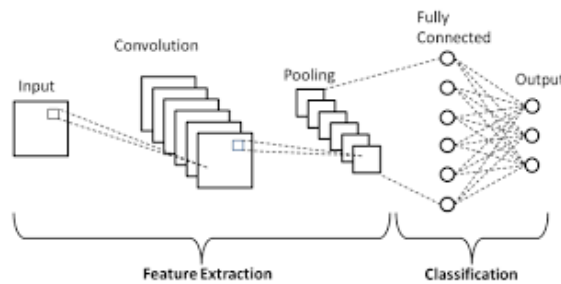


FIG. 2.5 : Architecture du CNN

2.5.4 VGGNet

VGG est un réseau de neurones convolutionnels proposés par K. Simonyan et A. Zisserman de l'université d'Oxford en 2014.

L'entrée du VGG est une image RVB. La couche de prétraitement prend l'image RVB avec des valeurs de pixels comprises entre 0 et 255 et soustrait les valeurs moyennes de l'image qui sont calculées sur l'ensemble des images utilisées pour l'entraînement. Les images d'entraînement sont passées à travers une pile de couches de convolution (En angl. Convolution layer). Il y a au total 13 couches de convolution et 3 couches entièrement connectées (En angl. Fully connected layer) dans l'architecture VGG16. Une autre variante de VGGNet (VGG19) comporte 19 couches de poids, soit 16 couches convolutionnelles, 3 couches entièrement connectées et les mêmes 5 couches de mise en commun (En angl. pooling layers). Dans les deux variantes de VGGNet, la dernière couche entièrement connectée utilise la couche softmax pour la classification.

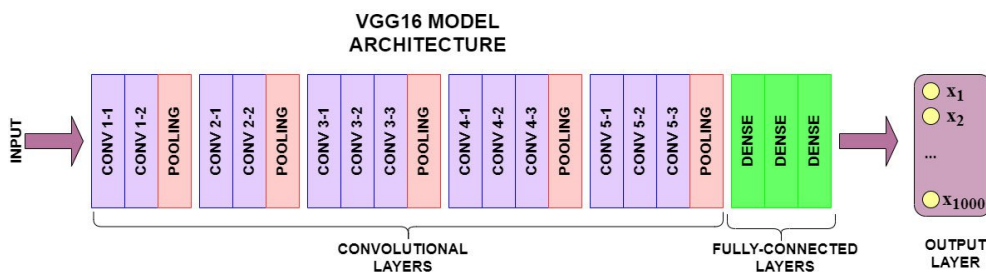


FIG. 2.6 : Architecture VGG16 (HASSAN 2018)

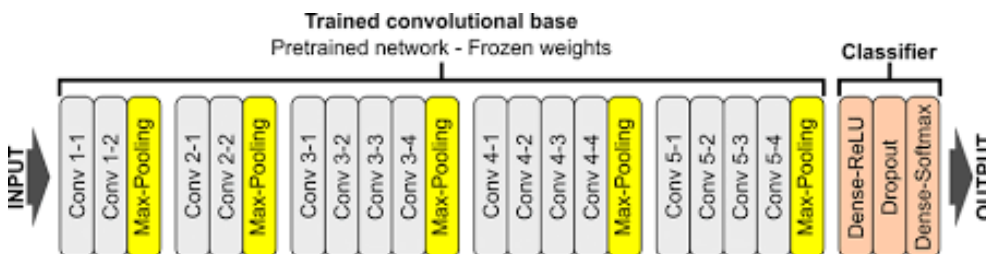


FIG. 2.7 : Architecture VGG19(JAWOREK-KORJAKOWSKA et al. 2019)

Conclusion

Dans ce chapitre, les notions de base du "Crowdsourcing", "Fact checking" et l'apprentissage automatique ont été présentées de manière théorique et abstraite. De plus, les notions de base liées à l'apprentissage profond ont été présentées, notamment leur définitions et leurs types.

À ce stade, les principes et concepts élucidés permettent une meilleure compréhension des travaux existants à propos de la détection automatique des fausses nouvelles. Les travaux qui abordent l'apprentissage profond pour la détection des fausses nouvelles seront plus détaillés dans le chapitre suivant.

CHAPITRE

3

TRAVAUX EXISTANTS

Introduction

La détection des fausses nouvelles a été répertoriée parmi les problèmes de classification. Les travaux effectués dans ce cadre ont débuté par les méthodes de classification non-profonde. Ensuite, de récents travaux ont inclus les méthodes de classification profonde.

Dans un premier lieu, les travaux de classification-non profonde seront présentés en expliquant les modèles, concepts et résultats, un intérêt particulier sera porté sur les travaux de détection automatique des fausses nouvelles basés sur la classification profonde. Ce chapitre sera clôturé avec une analyse et synthèse des travaux existants.

3.1 Travaux basés sur la classification non profonde des fausses nouvelles

3.1.1 Truth Inference in Crowdsourcing : Is the Problem Solved ? (Granik et al. 2017)

Cet article traite le problème fondamental de la communauté des bases de données et la communauté de l'exploration des données qui est l'inférence de la vérité. Beaucoup d'algorithmes ont été proposés. Cependant, ces algorithmes ne sont pas comparés de manière approfondie dans le même cadre. Pour remédier à ce problème, cet article fournit un framework qui fournit une étude comparative détaillée de 17 algorithmes existants afin d'identifier leur limites et le meilleur algorithmes.

Les entrées du modèle sont l'ensemble des réponses des travailleurs¹ pour toutes les tâches et les sorties sont la vérité estimée par l'algorithme pour chaque tâche et la qualité du travailleur estimée par l'algorithme.

3.1.1.1 Modèle détaillé :

L'approche générale adoptée par la plupart des algorithmes existants de découverte de la vérité est présentée dans ce framework.

Étape 1 : inférer la vérité : il infère la vérité de chaque tâche à partir des réponses et des qualités des travailleurs.

Étape 2 : Estimation de la qualité du travailleur : sur la base des réponses des travailleurs et de la vérité de chaque tâche, il estime la qualité de chaque travailleur.

Convergence : les deux itérations sont exécutées jusqu'à convergence. En général, pour identifier la convergence, les travaux existants vérifient si le changement de deux ensembles de paramètres (c'est-à-dire les qualités des travailleurs et la vérité des tâches) est inférieur à un seuil défini.

Enfin, la vérité déduite et les qualités des travailleurs sont retournées.

Algorithm 1: Solution Framework

Input: workers' answers V
Output: inferred truth v_i^* ($1 \leq i \leq n$), worker quality q^w ($w \in \mathcal{W}$)

```

1 Initialize all workers' qualities ( $q^w$  for  $w \in \mathcal{W}$ );
2 while true do
3   // Step 1: Inferring the Truth
4   for  $1 \leq i \leq n$  do
5     | Inferring the truth  $v_i^*$  based on  $V$  and  $\{q^w \mid w \in \mathcal{W}\}$ ;
6   // Step 2: Estimating Worker Quality
7   for  $w \in \mathcal{W}$  do
8     | Estimating the quality  $q^w$  based on  $V$  and  $\{v_i^* \mid 1 \leq i \leq n\}$ ;
9   // Check for Convergence
10  if Converged then
11    | break;
12 return  $v_i^*$  for  $1 \leq i \leq n$  and  $q^w$  for  $w \in \mathcal{W}$ ;

```

FIG. 3.1 : Framework proposé (GRANIK et al. 2017)

3.1.2 Truth Discovery against Strategic Sybil Attack in Crowdsourcing (Yue Wang et al. 2020)

Dans cet article, ils ont parlé d'une attaque spécifique de crowdsourcing qui est 'l'attaque Sybil' où l'attaquant gagne des récompenses faciles en coordonnant plusieurs travailleurs pour partager une étiquette aléatoire sur chaque tâche afin de dominer le résultat de l'agrégation. Cet article discute aussi les raisons qui ont motivé ce genre d'attaques et étudie les limites des anciennes solutions qui ont été utilisées pour combattre ces attaquants, c'est pourquoi ils ont développé un framework appelé 'TDSSA' qui garantit une inférence plus précise de la vérité dans divers scénarios d'attaque Sybil, par rapport aux

¹Les personnes individuelles qui participent au Crowdsourcing sont appelées "crowd workers" ou "clickworkers" souvent inscrits sur des plateformes de crowdsourcing.

méthodes de base.

Les entrées du modèle sont l'ensemble des réponses des travailleurs pour toutes les tâches et les sorties sont l'ensemble des étiquettes agrégées des tâches.

3.1.2.1 Modèle détaillé :

Le framework 'TDSSA' contient trois modules principaux qui seront présentés ci-dessous, comme le montre la figure 5.28.

Extended Truth Discovery (ETD) : met à jour de manière itérative les étiquettes agrégées des tâches normales² et le poids des travailleurs en associant à chaque travailleur un score de Sybil qui capture la probabilité d'être un attaquant et un score de fiabilité qui capture la confiance d'être un bon travailleur indépendant.

L'affectation probabiliste des tâches (PTA) : attribue une tâche normale ou une tâche en or³ de manière probabiliste à chaque travailleur demandeur en fonction de leur score de Sybil et de leur score de fiabilité. L'objectif est d'assigner des tâches en or de manière stratégique tout en les camouflant pour que les attaquants ne les reconnaissent pas.

Batched Processing (BP) : exécute les composants ci-dessus en mode batch avec deux objectifs :

- Mettre à jour les paramètres du travailleur (score de Sybil et le score de fiabilité) au fur et à mesure que 'les tâches en or' sont appliquées
- Promouvoir les tâches achevées en de nouvelles tâches en or à la fin de chaque lot.

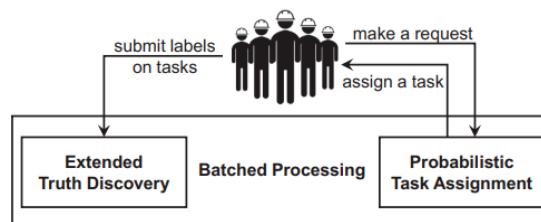


FIG. 3.2 : Aperçu de TDSSA (Yue WANG et al. 2020)

²Ce sont des tâches auxquelles on ne connaît pas l'étiquette et elles sont attribuées aux travailleurs afin de trouver leur vraie étiquette.

³Les tâches en or dont le véritable label est connu sont souvent attribuées aux travailleurs dans une phase de démarrage, à titre de test de qualification. Un attaquant stratégique de Sybil peut d'abord étiqueter honnêtement les tâches en or pour améliorer son score de fiabilité, mais attaquer plus tard les tâches normales.

3.2 Travaux basés sur la classification profonde des fausses nouvelles

3.2.1 Truth Discovery with Memory Network (Li et al. 2016)

Dans cet article, deux modèles de réseaux de mémoire (LSTM) ont été proposés pour apprendre la crédibilité des sources et prédire la vérité. L'un d'eux est un réseau de neurones de type feedforward (FFMN) avec un mécanisme de réseau à mémoire et l'autre est un réseau de neurones de type feed-back⁴ (FBMN) dont la rétroaction de ses neurones est transmise aux autres neurones en tant qu'entrée après un pas de temps.

L'entrée du modèle est l'observation qui est un vecteur composé de : (Objet, propriété, Valeur) et la sortie du modèle est la Crédibilité de l'observation.

3.2.1.1 Modèle détaillé

- FFMN : L'architecture du FFMN est présentée dans la partie gauche de la figure 3.3. L'entrée de chaque itération dans la composante I est l'ensemble des vecteurs d'observations selon une même entrée. M stocke K vecteurs de mémoire, qui représentent la fiabilité des sources. Dans le FFMN, les vecteurs de mémoire servent de matrice de poids et sont combinés avec les informations d'entrée par multiplication matricielle. Ainsi, comme pour les autres paramètres du FFMN, le vecteur mémoire est actualisé à chaque fois par rétropropagation des erreurs. La réponse du FFMN R est la fiabilité de chaque observation.
- FBMN : Le FBMN est illustré dans la partie droite de la figure 3.3. Ils ont ajouté un composant mémoire M au LSTM pour stocker la fiabilité des sources. L'entrée du modèle I est une série de valeurs provenant de différentes sources selon une même entrée. Le pas de temps est l'ordre des valeurs d'entrée. La réponse du modèle R est le résultat est la crédibilité de l'observation de chaque entrée. Les mémoires dans M sont mises à jour sur la base de la réponse par la rétro propagation de la dérivée.

Le test des deux modèles proposés est fait sur des données catégorielles et continues : stock et flight. FBMN donne le meilleur résultat pour les données catégorielles et continues. Cette excellente performance vérifie l'efficacité du mécanisme de mémoire. Malgré son architecture simple, le FFMN a donné aussi de bons résultats sur les deux types de données mais vu la simplicité de son architecture par rapport à celle de FBMN, il est considéré comme le meilleur modèle.

⁴L'architecture des réseaux neuronaux à rétroaction est également appelée interactive ou récurrente, bien que ce dernier terme soit souvent utilisé pour désigner les connexions de rétroaction dans les organisations à une seule couche. Les boucles de rétroaction sont autorisées dans de tels réseaux. Elles sont utilisées dans les mémoires adressables par le contenu.

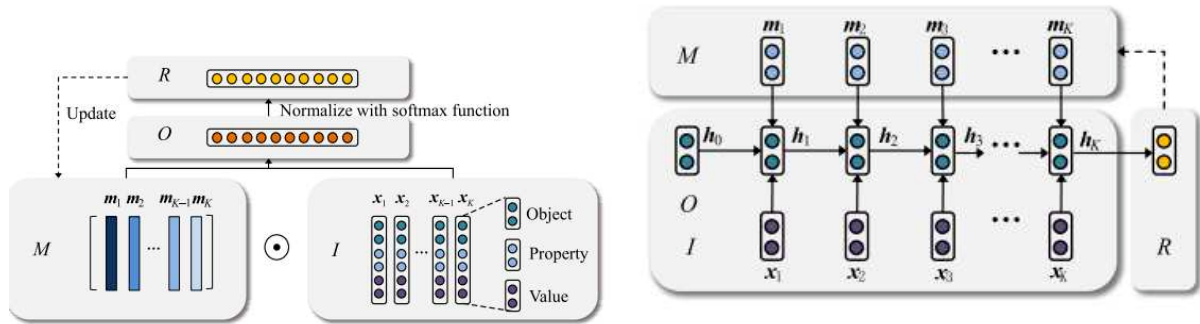


FIG. 3.3 : Architecture du modèle (Li et al. 2016)

3.2.2 A Neural Network Approach for Truth Discovery in Social Sensing (Marshall et al. 2017)

L'article parle des limitations importantes des solutions de base pour la découverte de la vérité qui supposent que la relation entre la fiabilité de la source et la véracité de la déclaration peut être représentée par des fonctions simplifiées (par exemple, linéaire, quadratique et binomiale). Dans cet article, un modèle de réseau de neurones à 4 couches de type feed-forward résout le problème de la découverte de la vérité sans aucune hypothèse sur la connaissance préalable de la relation de dépendance entre la source et les déclarations faites par la source.

Les entrées du modèle sont les déclarations reportées par les sources et la sortie est le label de la déclaration : Vrai ou faux.

3.2.2.1 Modèle détaillé

Un réseau neuronal a été développé avec la méthode Dropout pour résoudre le problème de découverte de la vérité. L'idée principale du schéma de réseau neuronal proposé est illustrée à la figure 4.13. L'entrée de ce schéma est un vecteur S représentant les sources qui ont signalé une réclamation. La sortie est un vecteur O qui indique si l'allégation est vraie ou fausse. La fiabilité d'une source peut alors être calculée comme le rapport entre le nombre de déclarations correctes et le nombre total de déclarations faites par la source.

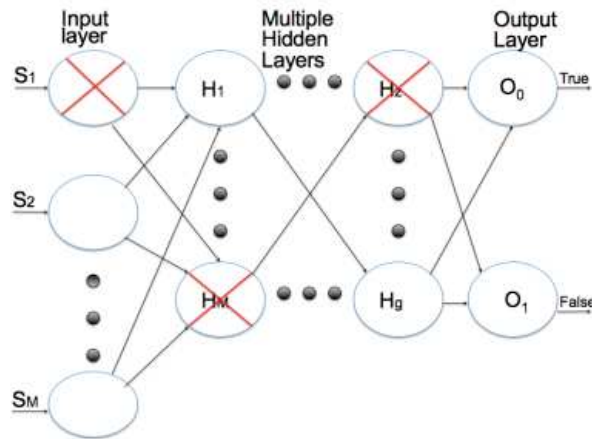


FIG. 3.4 : Architecture du modèle (MARSHALL et al. 2017)

La performance du modèle a été évaluée à l'aide des données extraites de Twitter de deux événements réels (l'attentat de Bruxelles et les émeutes de Baltimore). La performance du modèle est excellente : une accuracy qui est au voisinage de 80% et la F1-mesure est au voisinage de 85%.

3.2.3 FAKEDETECTOR : Effective Fake News Detection with Deep Diffusive Neural Network (Zhang et al. 2020)

Cet article aborde les défis introduits par les caractéristiques inconnues des fausses nouvelles et les divers liens entre les articles, les créateurs et les thèmes des nouvelles. Cet article présente un nouveau modèle d'inférence automatique de la crédibilité des fausses nouvelles, à savoir FAKE DETECTOR. Sur la base d'un ensemble de caractéristiques explicites et latentes extraites des informations textuelles et la corrélation entre les nouvelles, ses créateurs et leurs thèmes qui sont représentés en utilisant un modèle de réseau de neurones profond (ce réseau est détaillé dans la section ci-dessous).

Les entrées proviennent de différentes sources simultanément, une entrée est un vecteur désignant les caractéristique extraites des articles, le(s) thème(s) des articles et les créateurs de ces articles et les sorties sont des vecteurs qui représentent les labels de crédibilité déduits des articles, sujets et créateurs.

3.2.3.1 Modèle détaillé

Le framework FAKEDETECTOR couvre deux composantes principales, comme le montre la figure 3.5 :

- Representation Feature Learning (HFLU) : ce module permet d'extraire les caractéristiques explicites et latentes à partir du contenu textuel des articles.
- Credibility label inference (GDU) : c'est le module qui exploite les caractéristiques textuelles des articles et la relation entre les trois entités : article, sujet et auteur afin de donner un vecteur de crédibilité pour chacune de ces entités.

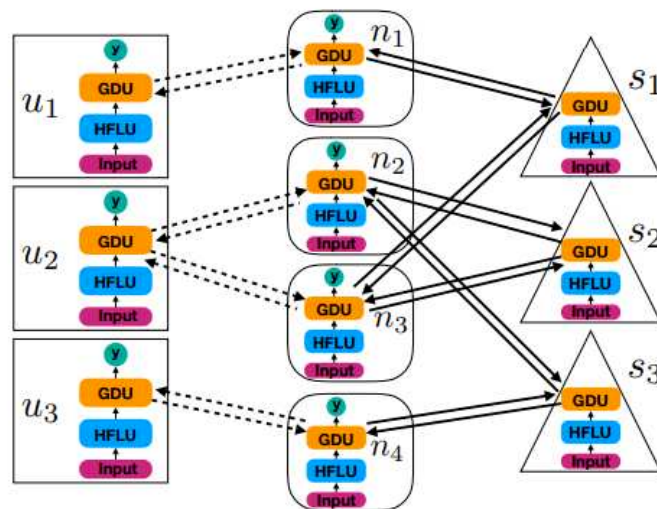


FIG. 3.5 : Architecture du modèle (ZHANG et al. 2020)

Ils ont fait leurs expériences sur le jeu de données "Politifact". La méthode FAKE-DETECTOR peut atteindre la meilleure performance parmi toutes les autres méthodes (CNN, RNN, etc.) pour inférer les étiquettes des articles de presse, des créateurs et des sujets.

3.2.4 EANN : Event Adversarial Neural Networks for Multi-Modal Fake News Detection (Yaqing Wang et al. 2018)

L'un des défis uniques de la détection des fausses nouvelles sur les médias sociaux est de savoir comment identifier ces dernières sur des événements nouvellement apparus. L'EANN est conçu pour extraire les caractéristiques partagées entre tous les événements afin d'améliorer efficacement les performances de détection des fausses nouvelles sur les événements jamais vus.

Les entrées du modèle sont les articles (texte avec ou sans une image attachée) et les sorties sont des vecteurs qui représentent les labels de crédibilité déduits des articles, sujets et créateurs.

3.2.4.1 Modèle détaillé

Le modèle EANN proposé intègre trois composants principaux, comme le montre la figure 3.6 :

- L'extracteur de caractéristiques multimodales : comprend des extracteurs de caractéristiques textuelles (Texxt-CNN) et visuelles (VGG-19) pour traiter différents types d'entrées. Après l'apprentissage des représentations des caractéristiques latentes textuelles et visuelles, elles sont concaténées ensemble pour former la représentation finale des caractéristiques multimodales.
- Le détecteur de fausses nouvelles : prend les caractéristiques représentées comme

entrée pour prédire si les articles sont faux ou réels.

- Le discriminateur d'événements : identifie le label d'événement de chaque article en fonction de cette représentation latente.

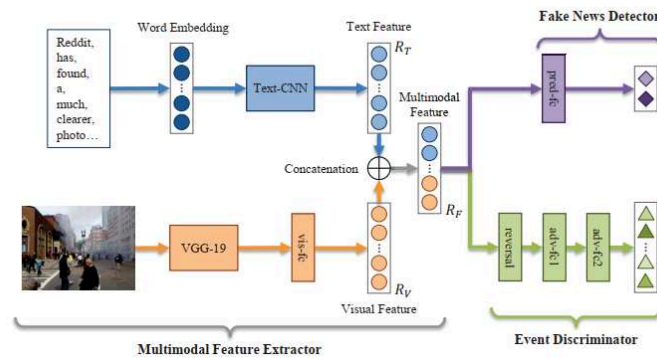


FIG. 3.6 : Architecture du modèle EANN (Yaqing WANG et al. 2018)

Ils ont fait leurs expériences sur les jeu de données réels "Twitter" et "Weibo". Les performances globales de l'EANN proposé sont bien meilleures que celles des approches de base (CNN, RNN, etc.) en termes d'exactitude, de précision et de score F1.

3.3 Synthèse

Dans cette partie, des remarques et des conclusions seront présentées sur les travaux lus sur la détection automatique basée sur la classification profonde :

- les travaux faits pour la détection des fausses nouvelles peuvent être classés de différentes manières, par exemple :
 - Selon la source : il y a des travaux qui utilisent la crédibilité ou bien la réputation de la source pour détecter la vérité et chacun de ces travaux utilise une formule précise pour calculer ce score.
 - Selon le type des entrées : généralement, les travaux faits dans cette catégorie utilisent uniquement le contenu textuel des entrées, d'autres utilisant les images/vidéos qui viennent avec le texte et d'autres utilisant le contexte social aussi. Beaucoup de travaux aussi dans la littérature utilisent les données tabulaire qui viennent sous cette forme : objet, propriété, source.
- Récemment beaucoup de modèles d'apprentissage profond ont été proposés pour la détection des fausses nouvelles car :
 - Les techniques d'apprentissage profond sont plus robustes et plus efficaces que les autres approches et ont montré leur force dans diverses applications, notamment dans la détection des fausses nouvelles : spam, rumeur, fausse information, désinformation, etc. (Wu et al. 2019).

- Les techniques d'apprentissage profond sont très flexibles, surtout avec l'avènement de ces bibliothèques : Tensorflow, Keras, Caffe, PyTorch et Theano.
 - Ces modèles permettent de prendre en considération les caractéristiques variantes des fausses nouvelles, comme la crédibilité de la source (en utilisant le mécanisme de la mémoire des LSTMs), les images attachées (en utilisant les CNNs).
 - Les modèles RNNs sont plus adéquats à la classification textuelle car ils capturent le comportement séquentiel des données, ce qui en fait une approche plus "naturelle" pour traiter des données textuelles, puisque le texte est naturellement séquentiel.
 - les modèles d'apprentissage profond permettent d'intégrer la composante "source" dans leur architecture.
- La comparaison des performances des modèles de classification n'est pas possible, car elle n'est utile que si ces modèles là sont appliqués aux mêmes jeux de données. On essaiera quand-même de citer les points faibles et forts de chaque approche dans le tableau 3.2.
 - L'utilisation de la fiabilité de la source devrait être un élément important lors de la vérification des faits. Pourtant, il s'agit d'un problème peu étudié et beaucoup de modèles d'apprentissage profond n'utilise pas la source pour détecter la véracité des nouvelles.
 - Malgré qu'il y a beaucoup de modèles qui sont proposés pour résoudre ce problème, la plupart ne traite pas le challenge des données nouvellement apparues et qui peuvent être de différents domaines jamais traités auparavant.
 - Les méthodes qui n'utilisent pas la classification automatique comme le "crowd-sourcing" et le "fact-checking" sont moins performantes que celles qui utilisent l'apprentissage profond, car ces méthodes utilisent des travailleurs et souvent ce sont des attaquants ou bien des travailleurs de mauvaise qualité ; c'est dur de trouver des travailleurs spécialisés dans le même domaine que les nouvelles qu'on cherche à vérifier.

Article	Points faibles	Points forts
(ZHANG et al. 2020)	<ul style="list-style-type: none"> • Il aurait fallu séparer les résultats de la comparaison en fonction des méthodes qui n'utilisent que les caractéristiques latentes ou bien explicites et celles qui utilisent les deux. • Certaines méthodes de base (avec une architecture plus simple que celle du modèle proposé) ont également eu d'excellents résultats. • La performance du modèle n'étaient pas très bonnes dans l'inférence de crédibilité des données multi-classes (articles/créateurs). 	<ul style="list-style-type: none"> • Outre l'extraction de caractéristiques explicites, les caractéristiques latentes du texte des articles/créateurs/sujets ont été également utilisées.
(Yaqing WANG et al. 2018)	<ul style="list-style-type: none"> • La comparaison n'était pas juste (les paramètres utilisés dans les méthodes de base n'étaient pas adéquats pour les données utilisées dans l'article, ils ont pris les mêmes valeurs que celles trouvées dans les articles qui proposent ces méthodes de base). 	<ul style="list-style-type: none"> • l'extracteur de caractéristiques multimodales comprend des extracteurs de caractéristiques textuelles et visuelles pour traiter différents types d'entrées et permet de détecter si le message est faux ou non (car parfois l'image peut être réelle mais la description qui l'accompagne est trompeuse). • Introduction du concept de discriminateur d'événements qui permet d'apprendre de traiter les données nouvellement apparues.

TAB. 3.1 : Synthèse détaillée des travaux lus.

Article	Points forts	Points faibles
(GRANIK et al. 2017)	<ul style="list-style-type: none"> • Ce framework peut être appliqué à toutes les méthodes existantes avec différentes techniques. 	<ul style="list-style-type: none"> • Offrir plus d'argent aux travailleurs pour calculer leur confiance n'est pas une bonne idée car ils auront plus de tâches et répondront de manière paresseuse.
(Yue WANG et al. 2020)	<ul style="list-style-type: none"> • Ils ont abordé deux problèmes importants dans cette approche : le camouflage des tâches en or de l'attaquant et la problème d'insuffisance des tâches en or pour tous les travailleurs. • TDSSA incorpore gracieusement le comportement de travailleurs "Sybil" (ou attaquants) et la fiabilité des labels qu'ils attribuent dans le processus de découverte de la vérité. 	<ul style="list-style-type: none"> • Si de nombreux travailleurs sont des travailleurs Sybil ou des travailleurs indépendants de faible qualité, les tâches en or peuvent être épuisées avec peu d'itérations.

TAB. 3.2 : Synthèse détaillée des travaux lus (suite).

CHAPITRE

4

CONCEPTION

Introduction

Dans ce sujet de PFE, on s'intéresse à l'application de l'apprentissage profond pour la détection de fausses informations notamment dans le contexte actuel de la crise sanitaire à la fois sur les forums, le web et les médias sociaux où de nombreuses informations intentionnellement fausses sont propagées et relayées à plus ou moins grande ampleur. Le présent chapitre détaille la conception utilisée pour atteindre les objectifs du PFE, qui sont les suivants :

- Comparer les méthodes de détection des fausses nouvelles existantes. En premier lieu, il a fallu préparer un pipeline et développer le même environnement commun pour exécuter et comparer plusieurs méthodes existantes pour la détection de fausses informations dans les réseaux sociaux et sur les Web.
- Pour rendre la comparaison plus intéressantes, nous avons transformé quelques modèles d'apprentissage profond qui sont destinés pour les données textuelles uniquement, afin qu'ils soient opérationnels sur les données tabulaires.
- Rajouter la composante "source" aux modèles de détection des fausses nouvelles afin de savoir l'impact de cette dernière sur ces modèles.
- Afin de mener différentes attaques contre ces modèles créés dans l'étape précédente, c'est à dire des modèles qui utilisent aussi la source des nouvelles pour prédire leur nature, nous allons essayer différents types d'attaques qui ciblent une déclaration particulière, des sources et un groupe aléatoire de déclarations.
- Construire un nouveau modèle d'apprentissage profond pour la détection des fausses nouvelles en se basant sur le texte des nouvelles et la source de ces dernières.

4.1 Comparaison entre les méthodes de détection des fausses nouvelles existantes

Afin de comparer les méthodes de détection des fausses nouvelles existantes dans la littérature (surtout celles que nous avons vues lors de l'étude bibliographique). Nous avons créé le pipeline illustré dans la figure (4.1) pour pouvoir obtenir des résultats justes pour la comparaison.

Les étapes du pipeline sont les suivantes :

1. Collecter les données sur lesquelles nous allons tester les méthodes.
2. Nettoyer les données collectées pour qu'elles soient prêtes à être exploitées par les modèles.
3. Préparer un environnement d'exécution pour ces méthodes : ça doit être sur la même machine, sur la même version de Python, etc.
4. Récupérer le code des méthodes de détection des fausses nouvelles s'il existe sinon nous devons les implémenter.
5. Vu que le but est de comparer ces méthodes, il va falloir choisir des métriques d'évaluation, par exemple : l'exactitude, la précision, le temps, etc.
6. Afin d'obtenir ces mesures d'évaluation, on doit opter pour une méthode d'évaluation, par exemple : "Cross validation" pour les méthodes d'apprentissage profond, etc.
7. Après avoir choisi tous les paramètres cités ci-dessus, on pourra passer à l'évaluation de ces méthodes et interpréter les résultats.

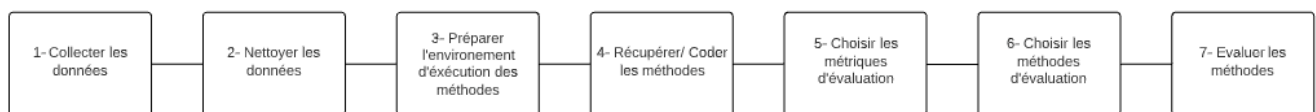


FIG. 4.1 : Processus global pour la comparaison des méthodes

4.1.1 Collecte des données

Le but de la collecte des données est de récupérer tous les jeux de données vus dans les travaux que nous avons lus lors de l'étude bibliographique et les nettoyer afin de tester les modèles de détection des fausses nouvelles sur ces derniers.

Quelques jeux de données ont été récupérés à partir des liens github donnés au niveau des articles que nous avons vu lors de l'étude bibliographique et d'autres jeux de données ont

été récupérés à partir des sites de vérification des faits comme : Snopes¹, Politifact², etc. La majorité des données abordaient les sujets suivants : politique, les nouvelles des stars, conspiration, médecine, etc.

Ces données sont structurées et regroupées en deux catégories : textuelles et tabulaires.

- **Données textuelles** : c'est généralement les données récupérés à partir des articles publiés sur les réseaux sociaux et le web. On dispose de 6 jeux de données textuels :
 - Kaggle ³ : il s'agit d'un jeu de données d'entraînement ouvert complet qui contient des articles d'actualité sur la politique et d'autres sujets différents.
 - Snopes ⁴ : c'est un site de vérification des faits connu qui valide les rumeurs sur Internet, les canulars, les légendes urbaines et autres histoires d'origine inconnue ou douteuse. Il recueille généralement ces rumeurs et ces affirmations sur les médias sociaux, les sites d'informations, etc.
 - Gossip : Vu que la demande de vérifications fiables des faits a augmenté. C'est aujourd'hui le plus ancien et le plus grand site de vérification des faits en ligne, largement considéré par les chercheurs.
 - Politifact ⁵ : c'est un site internet de vérification des faits, qui vérifie la véracité des promesses et engagements pris par les politiciens américains et leurs dires.
 - PubHealth (SHAH et al. 2020) : Il s'agit d'un ensemble de données exhaustif pour la vérification automatique qui explique des allégations de la santé publique.
 - Covid-19 : c'est le premier ensemble de données multi-langues tiré à partir de 7623 articles d'actualité parlant du COVID-19, collectés du 04/01/2020 au 01/07/2020. Ils ont collecté les articles vérifiés auprès de 92 sites Web de vérification des faits comme : Poynter et Snopes. L'ensemble de données généré au final est en 40 langues et provient de 105 pays.
- **Données tabulaires** : ce sont des données qui contiennent des champs bien définis et renseignés. On dispose de 3 jeux de données tabulaires :
 - Population : ce jeu de données décrit les villes et leur populations dans une année bien précise par des sources différentes.
 - Weather : ce jeu de données décrit soit les températures, le degré d'humidité et le degré de pression atmosphérique des villes dans la même année par des sources différentes.
 - Flights : ce jeu de données rapporte les horaires de départ et d'atterrissage des vols et les portes de départ et d'arrivée des vols par des sources différents.

Pour bien comprendre le reste des notions qui viennent dans ce chapitre, il faut bien comprendre la notion d'objet, attribut et déclaration (affirmation/allégation), voici le

¹<https://www.snopes.com/>

²<https://www.politifact.com/>

³<https://www.kaggle.com/c/fake-news/overview>

⁴<https://www.snopes.com/>

⁵<https://www.politifact.com/>

schéma 4.2 qui illustre ces concepts :

Un objet peut avoir plusieurs déclarations pour chacun de ses attributs, par exemple : chaque source attribue une valeur à un attribut de l'objet.

Dans le cas des données textuelles, le même titre (objet), par exemple : "Boire le gel désinfectant peut-il nous protéger du covid-19?", peut avoir plusieurs textes (attributs) qui parlent du même titre, par exemple : des sources qui confirment l'allégation et d'autres qui la nient.

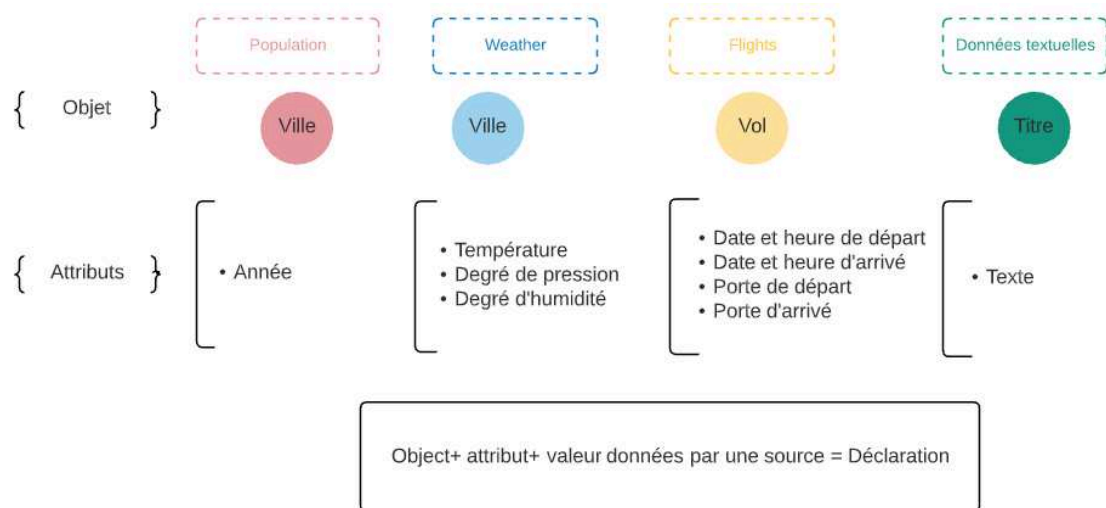


FIG. 4.2 : Schéma explicatif des entités : objet, attribut et déclaration

4.1.2 Préparation des jeux de données

Le nettoyage des données est une étape importante et qui prend énormément de temps, pour que les données soient prêtes à être exploitées, dans notre cas, la préparation des jeux de données dépend de :

- leur catégorie : tabulaire ou textuelles.
- leur caractéristiques, par exemple un jeu de données qui contient des dates aura d'autres étapes supplémentaires pour son nettoyage (mettre toutes les dates au même format)
- la méthode de détection, il y a des méthodes qui nécessitent au moins deux allégations pour le même objet, donc, on devra supprimer les allégations uniques.

4.1.2.1 Données tabulaires

Voici les étapes majeures pour le nettoyage des données tabulaires :

- Enlever les valeurs nulles, ces dernières se présentent, des fois, sous une autre forme, par exemple : "??", "no value", etc.
- Enlever les doublons.

- Pour le jeux de données “Flights”, le format des dates est varié entre des formats reconnus par les bibliothèques de Python et d’autres formats écrits manuellement, donc, il faut convertir toutes les dates au même format, en utilisant les bibliothèques de Python (datetime) pour les formats qu’elle a pu reconnaître et en insérer manuellement les formats non-reconnus par les bibliothèques de Python.

4.1.2.2 Données textuelles

Voici les étapes majeures pour le nettoyage des données textuelles :

- Enlever les valeurs nulles quand le nombre de lignes du jeu de données est grand ou bien nous remplaçons les valeurs nulles par une autre colonne, par exemple : si la colonne titre n’existe pas pour une certaine instance, on peut la remplacer par son texte.
- Enlever les doublons.
- Pour le jeux de données “Covid-19”, nous avons gardé uniquement les lignes écrites en anglais.
- Pour certains jeux de données, les labels des lignes étaient représentés par un chiffre entre 0 et 5 pour signifier le degré d’exactitude. Autre jeux de données, les lignes sont étiquetées par l’un des labels suivants : “mostly true”, “conspiracy”, “fake”, etc. Donc, nous avons transformé tous ces labels en un seul format : “fake” et “true” (fake = [“0”, “1”, “2”, “fake”, “conspiracy”]).
- Enlever la ponctuation.
- Transformer le texte en minuscule.
- Remplacer les mots par leur racine.
- Enlever les mots vides : (En angl. stop word) c’est un mot qui est tellement commun qu’il est inutile de l’indexer ou de l’utiliser dans une recherche ou comme entrée dans un modèle.

4.1.3 Choisir l’environnement d’exécution

Pour comparer les méthodes de détection des fausses nouvelles, il va falloir préparer le même environnement, c’est à dire que toutes les méthodes doivent être écrites sur Python 3 et exécutées sur Jupyter Notebook. Cette partie sera détaillée encore plus dans le chapitre “Réalisation”.

4.1.4 Méthodes de détection des fausses nouvelles existantes

Nous avons récupéré deux types de méthodes :

- **Probabilistes** : ces méthodes sont faites pour détecter les fausses nouvelles au niveau des données tabulaires, mais ce n'est toujours pas possible de tester ces méthodes sur les données textuelles.
- **Apprentissages profond** : ces méthodes sont destinées soit pour les données textuelles ou bien tabulaires, dans la littérature, c'est quasiment impossible de trouver le même modèle d'apprentissage profond qui traite les deux types de données. Dans notre cas, nous avons pu transformer quelques modèles destinés pour les données textuelles en modèles qui peuvent traiter aussi les données tabulaires afin de rendre la comparaison plus intéressante.

4.1.4.1 Méthodes probabilistes

Voici les méthodes statistiques que nous avons pu récupérer pour la détection des fausses nouvelles :

Préparation des données : il n'y a aucune autre préparation spéciale pour toutes ces méthodes à part celles qui ont été citées dans 4.1.2.1

Les entrées du modèles : ce sont les déclarations rapportées par les sources.

- **Confidence-Aware Truth Discovery CATD** (Qi Li et al. 2014) : cette méthode a été proposée pour résoudre les conflits et trouver les vérités parmi les différentes sources en adoptant des estimateurs efficaces basés sur l'intervalle de confiance de la fiabilité des sources. Elle peut aussi calculer efficacement la fiabilité réelle des sources avec différents niveaux de participation.
- **Latent Credible Analysis LCA** (PASTERNAK et al. 2013) : Il s'agit de modèles probabilistes latents qui utilisent des variables cachées (latentes) pour représenter la fiabilité des sources et les valeurs de vérité. La véracité de chaque affirmation est une variable latente et la crédibilité d'une source est calculée par un ensemble de paramètres du modèle.
- **Majority Voting MV** (Yin and Han 2007) : La MV considère la valeur revendiquée par la majorité des sources comme la vraie valeur et choisit aléatoirement une valeur en cas d'égalité.
- **Conflict Resolution on Heterogeneous Data CRH** (Qi Li et al., 2014) : elle calcule les vérités et les poids des sources à partir de données hétérogènes à sources multiples. Elle formule le problème de résolution des conflits comme un problème d'optimisation qui modélise les vérités comme la combinaison pondérée des observations de sources multiples et incorpore une variété de fonctions de perte pour les types de données hétérogènes. Elle utilise une procédure itérative de calcul des poids et des vérités pour résoudre ce problème d'optimisation. L'idée de base est que les sources fiables fournissent des observations fiables, donc les vérités devraient être proches des observations des sources fiables, et donc elle devrait minimiser la déviation pondérée des vérités à l'entrée multi-source où le poids reflète le degré de fiabilité des sources.

- ZenCrowd (Gianluca Demartini et al., 2012) : c'est un système basé sur un cadre probabiliste qui englobe à la fois les techniques automatiques et les retours ponctuels d'intelligence humaine capturés sur une plateforme de crowdsourcing.
- Moyenne : cette méthode est utilisée sur les données numériques, elle considère la vraie valeur comme la moyenne entre toutes les valeurs proposées par les sources.
- Mediane : cette méthode est utilisée sur les données numériques, elle considère la vraie valeur comme la médiane entre toutes les valeurs proposées par les sources.
- TruthFinder (YIN et al. 2008) : elle prend en entrée un ensemble de sources, de faits (affirmations) et d'objets. Sa mission est d'estimer la fiabilité des sources et la véracité des faits. La confiance d'un fait "f" est définie comme la probabilité que f soit correct. La fiabilité d'une source "s" est définie comme la confiance attendue des faits fournis par "s".

Préparation des données : il y a une étape supplémentaire à ajouter : garder uniquement les sources qui ont au moins deux déclarations pour que la méthode puissent calculer la crédibilité des sources.

4.1.4.2 Méthodes d'apprentissage profond

Nous avons pu récupérer/implémenter beaucoup de modèles d'apprentissage profond qui prennent comme entrée soit les données tabulaires ou bien les données textuelles :

- **Données textuelles** : Voici les modèles destinés aux données textuelles :

- Long Short-Term Memory (LSTM) model 1 (Andrii Shchur, 2020) :
Préparation des entrées : Pour avoir l'entrée finale du modèle, après avoir nettoyé les données textuelles de manière générale, maintenant, il faudra nettoyer chaque instance du jeu de données en suivant ces étapes :
 - Enlever la ponctuation.
 - Rendre les lettres minuscules.
 - Diviser les phrases en mots.
 - Prendre uniquement la racine des mots (En angl. Stemming)
 - Enlever les stop words.

Les entrées du modèles : C'est une phrase de 20 mots, on obtient ces mots là après l'encodage des mots et la création de séquences avec une longueur maximale de phrase (20 mots/phrase). Cette étape est importante pour créer la bonne forme de l'ensemble de données afin de l'utiliser dans les réseaux neuronaux.

L'encodage des mots se fait en utilisant "one hot representation", cette fonction reçoit en entrée une chaîne de texte et renvoie une liste d'entiers codés correspondant chacun à un mot dans la chaîne d'entrée donnée.

Les couches du modèles :

La première couche est l'embedding (l'embedding des mots est une représentation

apprise pour un texte où les mots qui ont la même signification ont une représentation similaire. Chaque mot est mis en correspondance avec un vecteur et les valeurs vectorielles sont apprises d'une manière qui ressemble à celle d'un réseau neuronal). Avec cette couche on est sûr que chaque mot sera représenté par un entier. La deuxième couche est le LSTM, les principaux concepts de cette couche sont : elle utilise des informations séquentielles et elle a une mémoire qui capture ce qui a été calculé jusqu'à présent. La sortie du réseau est la couche dense avec une fonction d'activation sigmoïde pour la classification binaire.

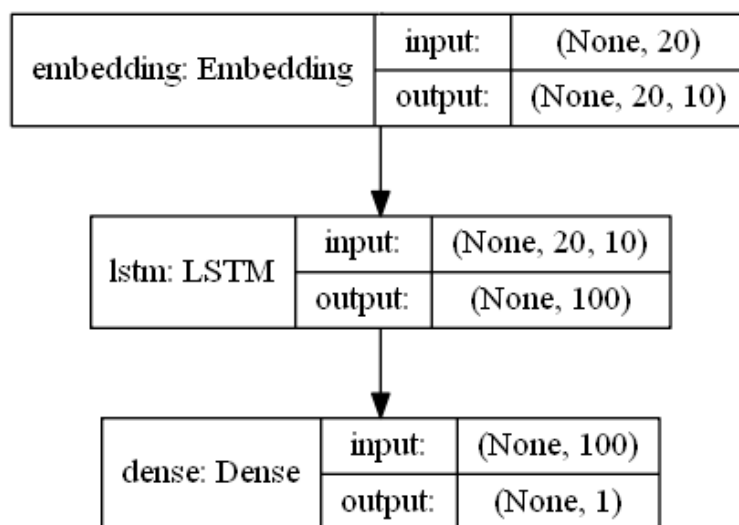


FIG. 4.3 : Architecture du modèle LSTM 1 (Andrii Shchur, 2020)

- Long Short-Term Memory (LSTM) model 2 (Andrii Shchur, 2020) :
Préparation des entrées : C'est le même nettoyage des entrées que celui de LSTM 1.

Les entrées du modèles : ce modèle part du même principe que le premier, cette fois-ci, la taille d'une phrase est de 100 mots.

Les couches du modèles :

Les couches sont les mêmes que celles du premier modèle comme on peut le voir dans la figure 4.4 à part pour la couche GlobalMaxPool1D, cette couche réduit l'échantillonnage de la représentation d'entrée en prenant la valeur maximale parmi toutes les valeurs utilisées pour la représentation. Elle a été ajoutée pour sous-échantillonner la représentation des entrées.

Les couches denses ont été ajoutées pour générer plus de caractéristiques pour les entrées.

La couche Dropout a été utilisée pour éviter le surajustement (En angl. overfitting) du modèle.

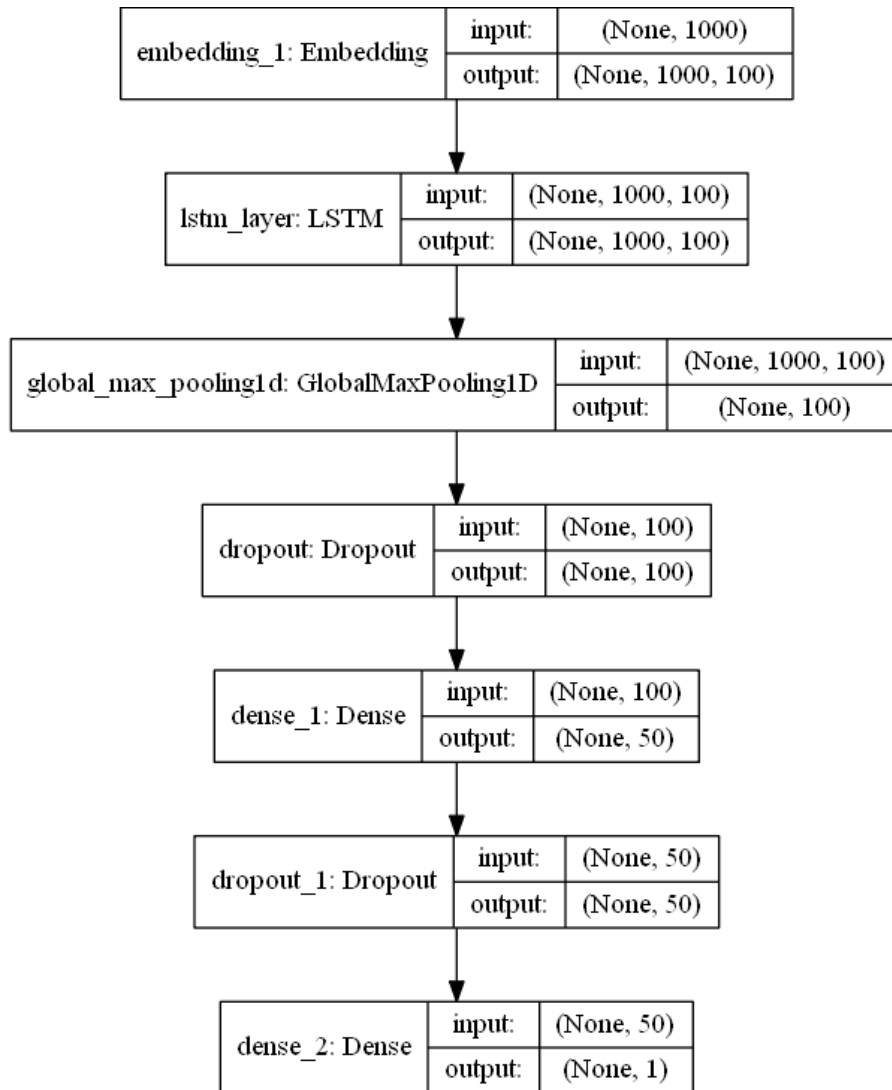


FIG. 4.4 : Architecture du modèle LSTM 2 (Andrii Shchur, 2020)

- Long Short-Term Memory (LSTM) model 3 (Andrii Shchur, 2020) :
Préparation des entrées : C'est le même nettoyage des entrées que celui de LSTM 1.
Les entrées du modèles : c'est un modèle à entrées multiples, où les deux architectures précédentes sont fusionnées en une seule. On peut faire entrer une phrase de 20 mots et une autre de 100 mots.
Les couches du modèles : En utilisant cette architecture, on peut obtenir le meilleur des deux architectures précédentes pour deux fonctionnalités distinctes et les fusionner en un seul modèle. Pour cela, une API fonctionnelle (elle peut gérer les modèles à topologie non linéaire, les modèles à couches partagées et les modèles à entrées ou sorties multiples) a été utilisée pour jumeler ces deux modèles.

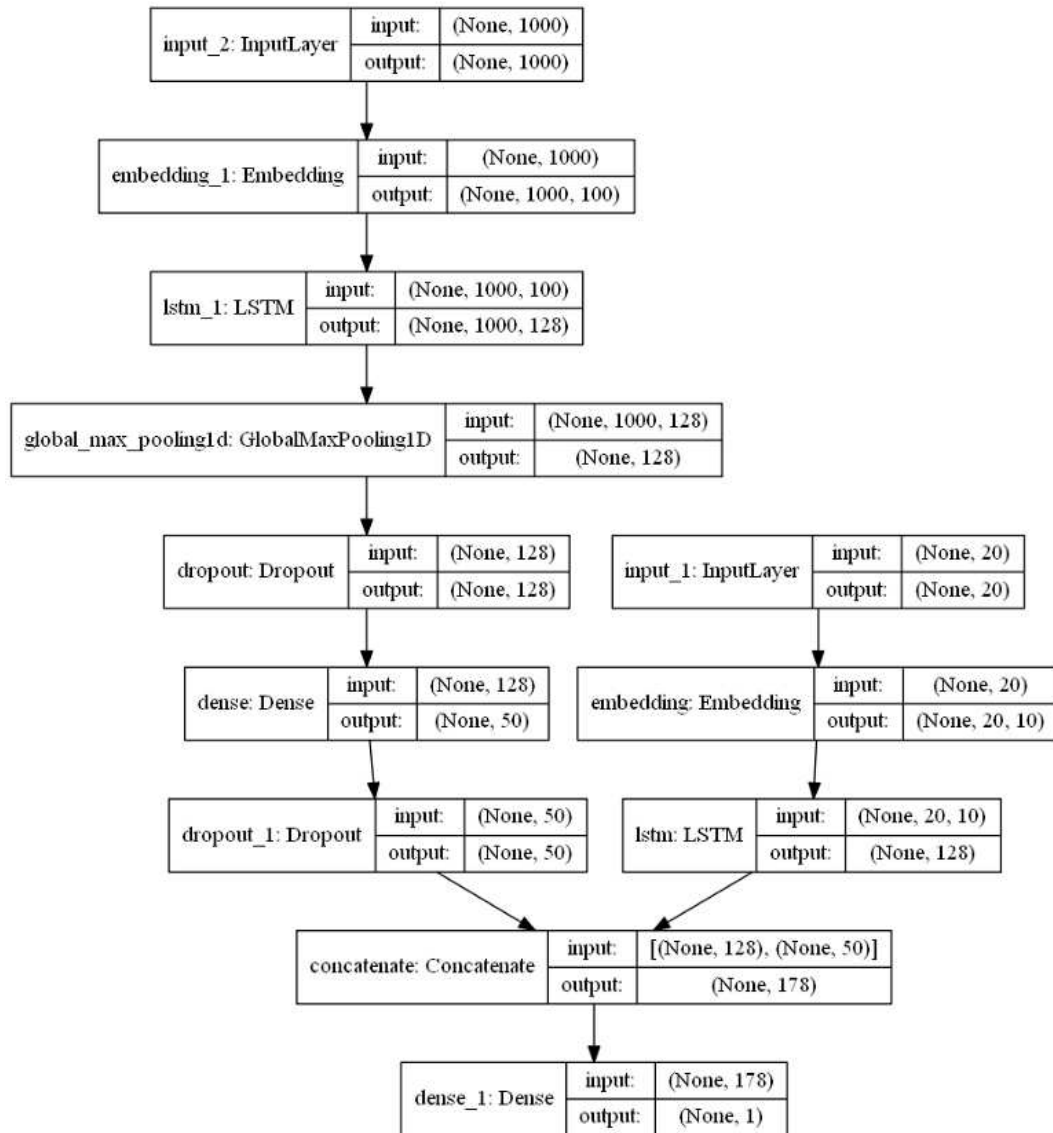


FIG. 4.5 : Architecture du modèle LSTM 3 (Andrii Shchur, 2020)

- Bidirectionnel LSTM (Bi-LSTM) :

Les LSTM bidirectionnels sont une extension des LSTM traditionnels qui peut améliorer les performances du modèle sur les problèmes de classification de séquences. Dans les problèmes où les entrées sont des séquences, les LSTM bidirectionnels forment deux LSTM au lieu d'un sur la séquence d'entrée. Le premier sur la séquence d'entrée telle quelle et le second sur une copie inversée de la séquence d'entrée. Cela peut fournir un contexte supplémentaire au réseau et permettre un apprentissage plus rapide et encore plus complet du problème.

Préparation des entrées : C'est le même nettoyage des entrées que celui de LSTM 1.

Les entrées du modèle : c'est une phrases composée de 20 mots.

Les couches du modèle :

Nous avons construit ce modèle qui a pour entrée une séquence de mots avec une cible binaire Vrai/Faux. La première couche est l'embedding . La deuxième couche

est le BiLSTM. La sortie du réseau est la couche dense avec une fonction d'activation sigmoïde pour la classification binaire.

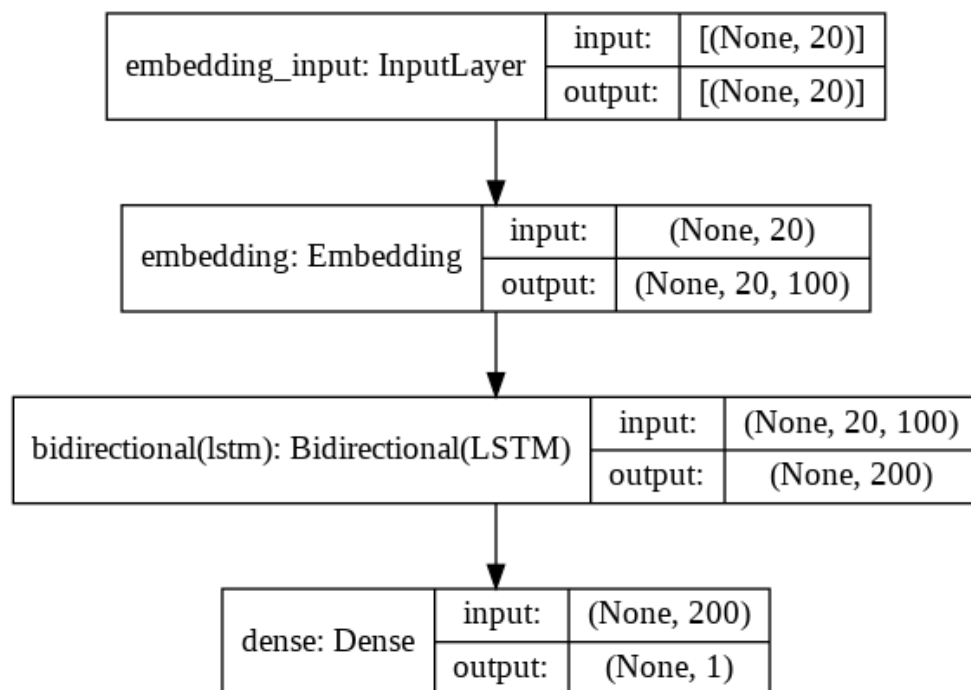


FIG. 4.6 : Architecture du modèle Bi-LSTM

- Recurrent neural network (RNN) model :

Nous avons construit ce modèle RNN pour la simple raison que les RNNs sont très utilisés pour les données textuelles qui sont une séquence de mots.

Préparation des entrées : C'est le même nettoyage des entrées que celui de LSTM 1.

Les entrées du modèles : Autre caractéristique que nous avons rajouté à ce modèle est la longueur des phrases qui varie d'un jeu de données à l'autre, généralement, la longueur d'une entrée est fixée à 20 mots/phrased pour les textes courts (titre) et 100 mots/phrased pour les textes longs (texte), mais dans ce cas, la longueur d'une entrée est la moyenne des longueurs des phrases dans le jeux de donnée entier.

Les couches du modèles :

Nous avons construit ce modèle qui a pour entrée une séquence de mots avec une cible binaire Vrai/Faux. La première couche est l'embedding . La deuxième couche est le LSTM. La troisième couche Flatten qui consiste à convertir les données en un tableau à une dimension pour les introduire dans la couche suivante. La sortie du réseau est la couche dense avec une fonction d'activation sigmoïde pour la classification binaire.

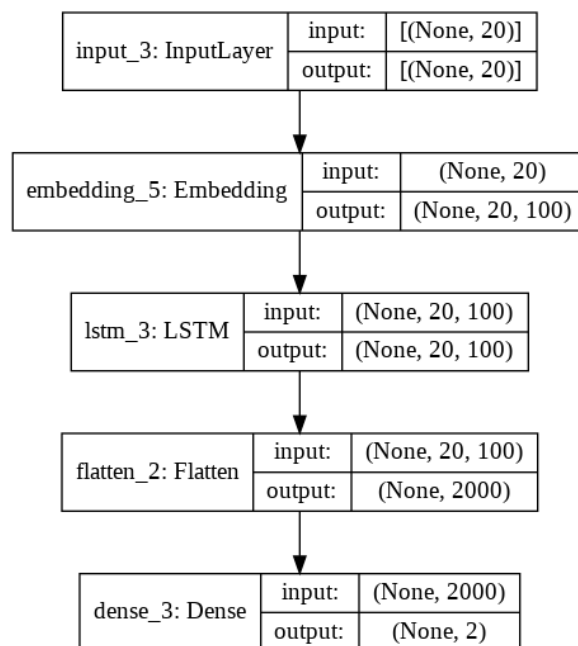


FIG. 4.7 : Architecture du modèle RNN

- **Données tabulaires** : Passons aux modèles destinés aux données tabulaires

- Réseaux de neurones (MARSHALL et al. 2017) : L'article parle des limitations importantes des solutions de base pour la découverte de la vérité qui supposent que la relation entre la fiabilité de la source et la véracité de la déclaration peut être représentée par des fonctions simplifiées (par exemple, linéaire, quadratique et binomiale). Dans cet article, un modèle de réseau de neurones à 4 couches de type feed-forward résout le problème de la découverte de la vérité sans aucune hypothèse sur la connaissance préalable de la relation de dépendance entre la source et les déclarations faites par la source.

Préparation des entrées : Il n'y a aucune autre préparation spéciale pour ce modèle à part celles qui ont été citées dans 4.1.2.1

Les entrées du modèles : sont les déclarations reportées par les sources.

Les couches du modèles :

Un réseau neuronal a été développé avec la méthode Dropout pour résoudre le problème de découverte de la vérité. L'idée principale du schéma de réseau neuronal proposé est illustrée à la figure 4.13.

L'entrée de ce schéma est un vecteur "S" représentant les sources qui ont signalé une réclamation. La sortie est un vecteur "O" qui indique si l'allégation est vraie ou fausse. La fiabilité d'une source peut alors être calculée comme le rapport entre le nombre de déclarations correctes et le nombre total de déclarations faites par la source.

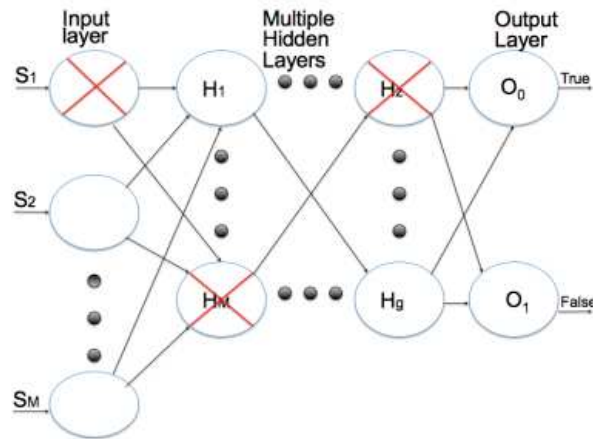


FIG. 4.8 : Architecture du modèle (MARSHALL et al. 2017)

- LSTM model 1 (Andrii Shchur, 2020) et Bi-LSTM model : ces modèles étaient destinés au départ uniquement pour les données textuelles. En gardant la même architecture et en modifiant le nettoyage des données tabulaires, ces modèles sont devenus exploitables pour les données tabulaires.

Préparation des entrées : Il a fallu bien préparer les jeux de données tabulaires pour qu'ils soient exploités par ces modèles d'apprentissage profond destinés uniquement aux données textuelles. Cette section sera bien expliquée ci-dessous 4.1.4.2

Les entrées du modèles : sont les déclarations reportées par les sources.

Les couches du modèles :

Les architectures de ces modèles étaient bien expliquées ci-dessus

- Nettoyage des données tabulaires pour les méthodes d'apprentissage profond :

- Annoter les données : Les données tabulaires ne disposent pas de labels, Elles sont réparties en deux jeux de données, celui des affirmations (En angl. claims) et celui de vérité de terrain (En angl. ground truth). Il a fallu chercher les affirmations dans les vérités de terrain, si elles se trouvent parmi les vérités, donc, on leur accorde un label "vrai" sinon "faux".
- Normaliser les données : Pour les données tabulaires numériques comme "population" et "weather" : l'objet et son attribut sont représentés par deux valeurs numériques (l'objet après l'encodage et l'attribut après la normalisation car il est déjà numérique), ces deux valeurs numériques représentent : la ville et l'année, la ville et la date pour "population" et "weather" respectivement, mais il faudra normaliser les valeurs des attributs, car le modèle ne peut pas prendre en compte les grandes valeurs des populations des villes et les valeurs négatives des températures.
- Encoder les données : Pour les données tabulaires de type caractères comme "Flights", on doit encoder la valeur de l'objet mais on devra tout d'abord enlever les caractères spéciaux ("-", "vide", etc.) avant de les encoder, par exemple : la valeur de l'objet "date de départ du vol numéro 101" est "12-02-20 13:00:00".

4.1.5 Choix des métriques et les méthodes d'évaluation

Métriques d'évaluation des méthodes d'apprentissage profond : Pour bien comprendre ces équations, voici une image qui résumé la codification des termes utilisés dans les équations ci-dessous.

		Réponse de l'expert	
		p	n
Réponse du classifieur	Y	Vrai Positif	Faux Positif
	N	Faux Négatif	Vrai Négatif

FIG. 4.9 : Matrice de confusion

- Temps : Temps d'exécution des méthodes mesuré en secondes.
- Exactitude : signifie le nombre de points de données prédits correctement. C'est l'une des formes les plus simples de mesure d'évaluation. Le score d'exactitude est le nombre de points prédits correctement/nombre total de points de données prédits.

$$Exactitude = \frac{TP + TN}{TP + TN + FP + FN} \quad (4.1)$$

- Precision : c'est le nombre d'entités attribuées correctement à une classe sur le nombre d'entités attribuées à la classe

$$Precision = \frac{TP}{TP + FP} \quad (4.2)$$

- Recall : c'est le nombre d'entités attribuées correctement à une classe sur le nombre d'entités appartenant à cette classe

$$Rappel = \frac{TP}{TP + FN} \quad (4.3)$$

- F1-score : Une mesure qui combine la précision et le rappel est leur moyenne harmonique.

$$F1 - score = \frac{2 * Precision * Recall}{Precision + Recall} \quad (4.4)$$

Méthodes d'évaluation des méthodes d'apprentissage profond : Afin d'avoir toutes ces métriques sur l'ensemble des données, on a utilisé une technique de validation appelée K-fold avec K=10 (Généralement dans la littérature, on prend K

est égal 10), c'est une méthode d'estimation de fiabilité d'un modèle fondé sur une technique d'échantillonnage. Dans notre cas ($K=10$), le jeu de données est divisé en 10 échantillons, 9 échantillons sont utilisés pour l'entraînement et le 10-ème pour le test. On répète ça 10 fois. Après, on prend la moyenne des métriques calculées à chaque itération.

Métriques d'évaluation des méthodes probabilistes : Afin d'évaluer la performance des méthodes statistiques, on a de différentes mesures d'évaluation selon le type des méthodes :

- Exactitude : déjà définie précédemment
- Temps : Temps d'exécution des méthodes mesuré en secondes.
- Mean Absolute Error (MAE) : c'est l'un des paramètres les plus couramment utilisés pour mesurer la précision des variables continues. MAE mesure la magnitude moyenne des erreurs dans un ensemble de prédictions, sans tenir compte de leur direction. Il s'agit de la moyenne, sur l'échantillon testé D , des différences absolues entre la prédiction y_i et l'observation réelle x_i , toutes les différences individuelles ayant le même poids.

$$\sum_{i=1}^D |x_i - y_i| \quad (4.5)$$

- Root mean squared error (RMSE) : La RMSE est une règle de notation quadratique qui mesure également l'ampleur moyenne de l'erreur. Il s'agit de la racine carrée de la moyenne des différences au carré entre la prédiction et l'observation réelle.

$$\sum_{i=1}^D (x_i - y_i)^2 \quad (4.6)$$

Dans notre cas, on va utiliser MAE et RMSE pour mesurer la performance des méthodes moyenne et médiane.

4.2 Intégrer l'aspect source dans les méthodes de détection des fausses nouvelles

Dans cette section, on va se focaliser sur les modèles d'apprentissage profond, comme ils sont déjà décrits, ces modèles s'appuient uniquement sur l'aspect textuel des nouvelles pour détecter leur label car, le style d'écriture d'une nouvelle qui emploie trop les sentiments et peu de références scientifiques révèle énormément d'informations sur la nouvelle. Malgré ça, on peut pas négliger le rôle d'une source/degré de sa fiabilité pour détecter le label d'une nouvelle, car, généralement les sources faibles rapportent des nouvelles fiables et vis-versa. C'est pour cela que nous avons décidé de rajouter l'aspect "source" pour ces modèles.

4.2.1 Aperçu sur les formules de calcul de la crédibilité des sources

Il y a eu plusieurs travaux qui parlent de l'importance de la source et de savoir la provenance de l'information pour détecter sa nature, car une source fiable a tendance à donner des nouvelles vraies et vis-versa.

La formule la plus connue pour calculer la crédibilité d'une source est la suivante :

$$Cred_i = \frac{\text{Nombre de déclarations vraies rapportées par } source_i}{\text{Nombre de déclarations total rapportées par } source_i} \quad (4.7)$$

Cette formule a été utilisée dans plusieurs travaux de la littérature comme celui de (MARSHALL et al. 2017). C'est une formule simple qui décrit le degré de crédibilité de n'importe quelle source mais le problème avec cette formule : elle ne prend pas en considération le phénomène de "Long-Tail Data", c'est à dire qu'on ne peut pas calculer la crédibilité de toutes les sources en utilisant cette formule car il y a des cas spéciaux comme : une source qui rapporte une seule déclaration et celle-ci est vraie, du coup, sa crédibilité sera de 100% sans savoir réellement grand chose sur cette source.

4.2.2 Création de formules de calcul de crédibilité

Afin de régler le problème "Long-Tail Data", nous avons proposé deux formules de calcul de crédibilité :

- Éliminer les sources qui rapportent aucune déclaration vraie et favoriser celles qui ont plus de déclarations avec cette formule :

$$Cred_i = rate_participation_i * (rate_true_claims_i) \quad (4.8)$$

Avec :

$$rate_participation_i = \frac{\text{Nombre de déclarations total rapportées par } source_i}{\text{Nombre maximal de déclarations rapportées par une source dans le dataset}} \quad (4.9)$$

Et :

$$rate_true_claims_i = \frac{\text{Nombre de déclarations vraies rapportées par } source_i}{\text{Nombre de déclarations total rapportées par } source_i} \quad (4.10)$$

- Garder toutes les sources telles qu'elles sont et favoriser les sources qui rapportent plus de déclarations avec la formule suivante :

$$Cred_i = \frac{\text{Nombre de déclarations vraies données par } source_i}{\text{Nombre de déclarations vraies données par toutes les sources}} - \frac{\text{Nombre de déclarations fausses données par } source_i}{\text{Nombre de déclarations fausses données par toutes les sources}} \quad (4.11)$$

Remarques sur les formules de calcul de crédibilité des sources : Au niveau de la 1-ère formule, on risque de perdre beaucoup de lignes en éliminant les sources, c'est pour cela, on laissera les sources qui donnent aucune déclaration vraie et on mettra leur crédibilité à 0.

Dans la 2-ème formule, on doit normaliser les valeurs obtenues avant de les utiliser car on risque d'avoir des valeurs négatives dans le cas où une source ne rapporte pas des déclarations vraies.

4.2.3 Ajout de la composante source aux modèles d'apprentissage profond

Dans cette partie, nous allons ajouter une autre entrée aux modèles d'apprentissage profond pour la détection des fausses nouvelles. A la place d'avoir une seule entrée pour les modèles d'apprentissage profond : le texte pour les données textuelles et les objets pour les données tabulaires. On va ajouter les sources et leur degré de fiabilité comme entrées, dans le but d'améliorer la performance de ces modèles.

Comme le montre la figure ci-dessous 4.10, on va concaténer le vecteur contenant les fiabilités des sources avec la matrice qui contient la représentation du contenu textuel/objets du jeu de données. Tout cela, va constituer la nouvelle entrée du modèle d'apprentissage profond.

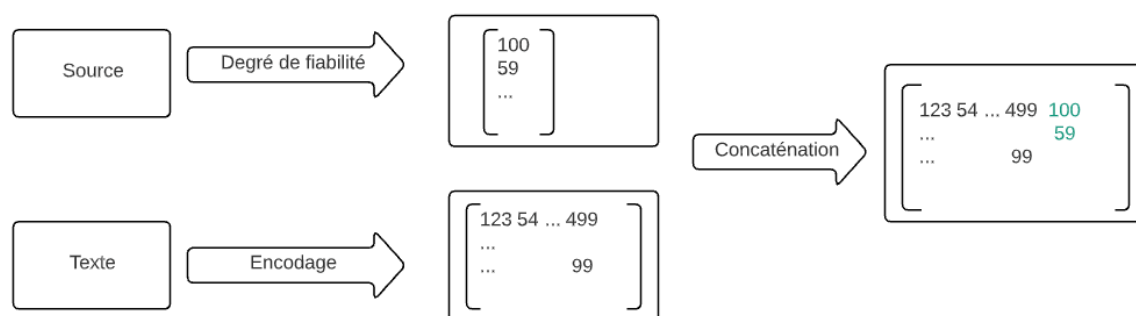


FIG. 4.10 : Ajout de la composante source aux entrées du modèle d'apprentissage profond

4.3 Attaques de désinformation

Comme on l'a déjà présenté dans l'étude bibliographique, les attaques d'apprentissage automatique adversaire sont devenues très dangereuses surtout dans le domaine de la découverte de la vérité. Pour qu'on puisse tester la robustesse des modèles de découverte de la vérité et développer des algorithmes de défense pour ces derniers. On va mener créer des attaques contre les modèles d'apprentissage profond utilisés pour la détection des fausses nouvelles.

Il y a beaucoup de travaux qui traitent ce problème d'attaques sur les plateformes de crowd-sourcing comme (Yue WANG et al. 2020), (VOROBAYCHIK 2018), etc.

Dans le cadre du PFE, nous avons essayé de mener différentes versions d'attaques sur

les modèles qu'on a créés. C'est pour cela que nous avons rajouté l'aspect sources à ces modèles, car elles vont jouer un rôle majeur dans les attaques. Les sources peuvent devenir des outils pour attaquer les modèles d'apprentissage automatique surtout si ces dernières ont un bon degré de crédibilité et il y a aussi des attaquants qui visent à ruiner la réputation/degré de crédibilité des sources.

4.3.1 La première version d'attaques

Dans cette section, nous avons créé une attaque de type "white box" : on suppose que l'attaquant connaît tout sur le modèle et de type "decision time" : l'attaquant va mener son attaque sur le jeu de données "test" et le but de cet attaquant est de baisser l'exactitude du modèle d'apprentissage profond.

Cette attaque est faite uniquement sur les données textuelles car nous avons décidé d'attaquer le texte des nouvelles. Afin de baisser l'exactitude du modèle, on va essayer de modifier le jeu de données test (l'injecter avec des données malicieuses). Pour cela, on prendra des lignes de ce dernier et on les changera en utilisant la forme négative.

Les modèles d'apprentissage profond qui sont ciblés par ces attaques sont les modèles qu'on a déjà présentés et auxquels on a ajouté l'aspect source.

Forme négative : Afin de transformer les lignes du jeu de données test en forme négative, il faut savoir d'abord qu'on la définit.

L'une des formes négatives les plus répandues en anglais est en utilisant le mot "not". Selon le dictionnaire de Cambridge, "Not" est l'un des mots les plus courants que nous utilisons pour indiquer la négation. Il est souvent abrégé en "n't" et joint à un verbe auxiliaire.

Pour cela, nous avons opté pour ces étapes afin de transformer quelques lignes du jeu de données "test" à la forme négative :

- On enlève le mot "not" à partir des mots vides (En angl. stop words).
- On prend chaque ligne et on la décompose en mots.
- En utilisant l'étiquetage morpho-syntaxique (En angl. Part-of-speech tagging), on attribut à chaque mot sa classe grammaticale.
- Si le mot appartient à l'une de ces classes grammaticales suivante : adjectif, verbe, adverbe, verbe auxiliaire (En angl. modal), on ajoute juste avant ce mot, le "not".

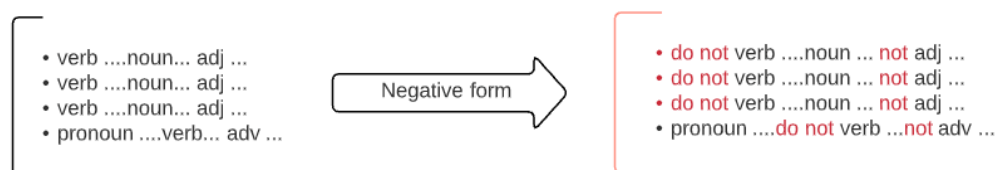


FIG. 4.11 : Première version de l'attaque

Évaluation de l'attaque : Pour savoir si l'attaque a réussi à baisser l'exactitude, on va comparer les résultats de l'exactitude après l'attaque avec ceux avant l'attaque. L'exactitude sera calculée lors de la prédiction sur le jeu de données "test".

4.3.2 La deuxième version d'attaques

Dans cette section, on va généraliser les attaques : on présentera des attaques menées sur les données tabulaires : on suppose que l'attaquant connaît tout sur le modèle (une attaque de type "white box"). L'attaquant peut avoir divers objectifs ; dans notre cas, nous essayons de maximiser l'erreur de prédiction par des attaques de retournement d'étiquette (En angl. label flipping).

Notations : Ce tableau 4.1 résumé l'ensemble des notations utilisées pour définir les attaques.

D_0	le jeu de données initial
D_1	le jeu de données injecté
T	l'ensemble des affirmations qu'on veut flipper
A	l'ensemble des affirmations injectées à rajouter au jeu de données
M	Modèle d'apprentissage profond
n	le nombre total des affirmations dans le jeu de données
m	le nombre total de sources dans le jeu de données
S	l'ensemble des sources, $S = \{S_i\}, 1 \leq i \leq m$
C	l'ensemble de déclarations $C = \{C_j\}, 1 \leq j \leq n$
S^j	l'ensemble des sources qui ont rapporté la déclaration C_j
C^i	l'ensemble des S_i
l_j^i	le label de la déclaration C_j donné par la source S_i
c_j^i	la j-ème déclaration reportée par la i-ème source
v_j^i	la valeur donnée par la source S_i à la déclaration C_j
Y	l'ensemble des labels $Y = \{y_j^i\}$
l_j^*	le vrai label pour la déclaration C_j
S_i	la i-ème source
C_j	la j-ème déclaration

TAB. 4.1 : Tableau de notations

4.3.2.1 Reformulation mathématique de l'attaque

En général, on commence avec un ensemble de données d'entraînement D_0 , sur lequel un vrai modèle peut être appris et se transforme en un autre D_1 en ajoutant de nouveaux points de données (qui représentent l'attaque). Ensuite, l'algorithme est appris sur D_1 , ce qui donne un nouveau modèle empoisonné qui est très différent du vrai modèle.

Nous définissons l'utilité de l'attaquant (la fonction objective que l'attaquant essaie de maximiser) par $U(D_0, T)$.

T est l'ensemble des affirmations dont les labels sont ciblés par l'attaquant veut changer leur labels ; ces affirmations peuvent appartenir à une seule source (afin de ruiner sa réputation) ou bien des affirmations aléatoires appartenant à l'ensemble de données d'apprentissage ou encore à l'ensemble des données de test.

$Cost(D_0, D_1)$ sera utilisé pour représenter la fonction de coût.

Le problème d'optimisation de l'attaquant prend la forme suivante :

$$\max U(D_0, T) = \max \sum_{k \in T} l(l_k^i, M(c_k^i, D_1)) \quad (4.12)$$

s. t.

$$\min Cost(D_0, D_1) = \sum_{k \in T} Cost(c_k) \quad (4.13)$$

$$T \subset D_0 \quad (4.14)$$

$$D_0 \cup A = D_1 \quad (4.15)$$

$$l(l_k^i, M(c_k^i, D_1)) = \begin{cases} 0 & \text{if } l_k^i = M(c_k^i, D_1) \\ 1 & \text{if } l_k^i \neq M(c_k^i, D_1) \end{cases} \quad (4.16)$$

4.3.2.2 Méthode d'attaque

Nous n'avons pas encore trouver de solution mathématique à ce problème d'optimisation, mais on a créé un algorithme qui peut nous mener vers la solution, il va falloir suivre les étapes suivantes : Cette attaque a été testée sur le modèle d'apprentissage profond MARSHALL et al. 2017 avec le jeu de données "Population".

Voici les différentes étapes de l'algorithme :

- Faire un profilage pour l'allégation cible : on doit savoir le nombre de valeurs attribuées à cette allégation et par quelles sources elle ont été rapportées et quel est le degré de crédibilité de ces sources, le nombre d'occurrence de chaque valeur et leurs labels.
- Créer la source malicieuse qui a au moins le même degré de crédibilité que la source qui a rapporté l'allégation cible, la création se fait de cette manière, pour que le coût soit minimale, on va chercher une source dans le même jeu de données qui a au moins le même degré de crédibilité que la source qui a rapporté l'allégation cible et cloner ses allégations.
- Choisir la bonne fausses valeur (En angl. fake value), c'est la valeur la plus répétée parmi l'ensemble de valeurs qui ont été données à cet objet cible. Au cas où, on a deux valeurs ou plus qui ont la même occurrence maximale, nous allons choisir la valeur la plus proche de la valeur cible.
- Affecter cette fausses valeur à la source malicieuse créée.
- Ajouter cette fausses allégation au jeu de données D_0 avec un label mis à "0" pour dire que cette fausses allégations est vraie.

- Répéter ce processus "la création de fausses allégations" en évitant de cloner de la même source (Chaque source malicieuse créée aura des allégations d'une source différente parmi celles qui existent dans D_0) jusqu'à ce que le label prédit soit modifié.

Évaluation de l'attaque : Pour savoir si l'attaquant a réussi son attaque, il suffit de voir si le label des affirmations ciblées a été changé après l'attaque.

4.4 Construction du nouveau modèle de détection des fausses nouvelles pour les données textuelles

Après l'étude bibliographique que nous avons faite et après avoir testé et implémenté différents modèles d'apprentissage profond, nous avons remarqué que les modèles qui se basent sur la séquences des entrées ont donné les meilleurs résultats et aussi les modèles qui prennent en considération les sources des nouvelles ont aidé à augmenter le taux de détection des fausses nouvelles. C'est pour cela, nous allons concevoir un modèle de détection des fausses nouvelles appelé "Fake news detector" qui prend en charge tous les critères qui semblent prometteurs suite aux résultats des tests que nous avons effectués.

4.4.1 Architecture des modèle

Comme les couches LSTM sont les plus adéquates pour traiter les données textuelles et que la source joue un rôle important.

Nous avons décidé de travailler avec l'aspect textuel de la source (c'est à dire les noms des auteurs). Avec ce modèle basé sur LSTM, on pourra faire la liaison entre le nom de l'auteur et le label de véracité des nouvelles. En fait, on va entraîner le modèle LSTM à apprendre la crédibilité des sources à partir du texte de ces sources, ce qui constitue son point fort pour les modèles LSTM (les séquences textuelles).

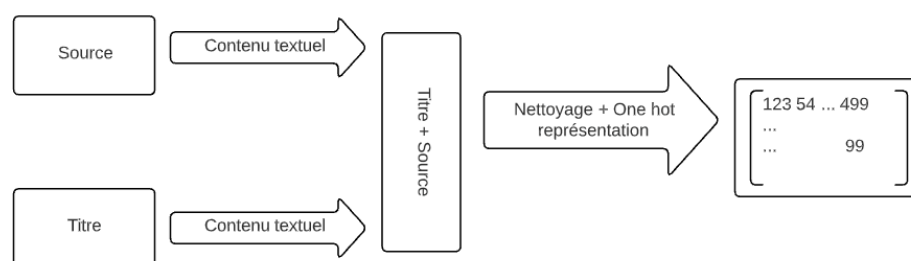


FIG. 4.12 : Entrées du modèle "Fake news detector"

Le modèle de "Fake news detector" est différent des autres modèles vus, car :
 Nous nous sommes focalisés sur la colonne titre des nouvelles, car généralement le titre est plus représentatif que la colonne texte des nouvelles (les titres ont tendance à utiliser

les mots clés, donc ils peuvent aider facilement à découvrir la nature de la nouvelle). Pour tirer plus d'avantage de cette colonne on va utiliser 25 mots par phrase à la place de 20 mots par phrase.

Nous allons introduire l'aspect source dans ce modèle, à la place d'avoir une seule entrée qui est le titre des nouvelles, nous allons concaténer le contenu textuel de la colonne titre et de la colonne source.

Afin d'éviter l'overfitting du modèle, nous allons utiliser 3 couches "dropout". Sans oublier la couche "Dense" avec la fonction d'activation "sigmoid" pour la classification binaire.

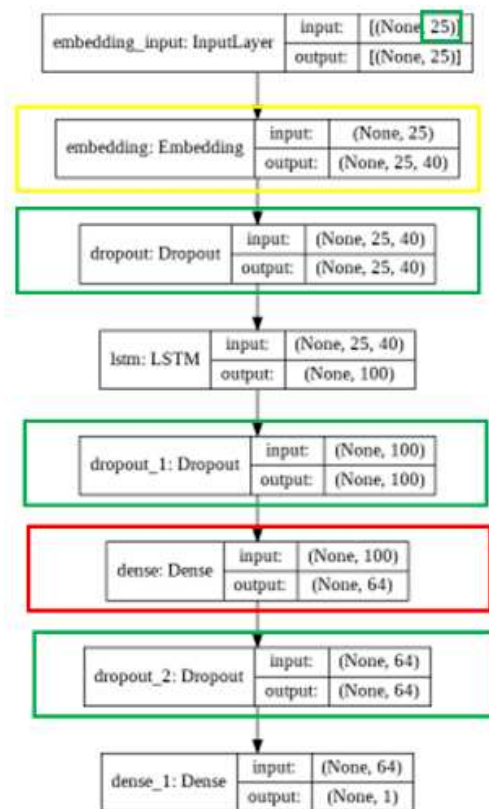


FIG. 4.13 : Architecture du modèle proposé Fake news detector

4.4.2 Nettoyage des données

Comme les entrées sont textuelles, dans ce cas là, voici les étapes standards de nettoyage :

- Enlever les valeurs nulles.
- Enlever les doublons.
- Enlever la ponctuation.
- Transformer le texte en minuscule.
- Remplacer les mots par leur racine.

- Enlever les mots vides : (En angl. stop word).
- Encodage des entrées en utilisant "one hot representation"

Conclusion

Ce chapitre contient les différentes phases pour atteindre chaque objectif du PFE. De plus, les modèles de classification non-profonds et profonds permettent une comparaison instructive pour de prochains travaux. Nous allons détailler dans le chapitre suivant, la réalisation de la conception des objectifs définis dans ce chapitre.

CHAPITRE

5

RÉALISATION, TESTS ET RÉSULTATS

Introduction

Ce chapitre est organisé en cinq sections : dans un premier temps on va définir les outils utilisés pour la réalisation du PFE, puis dans un second lieux, nous allons faire une comparaison entre les méthodes de détection des fausses nouvelles existantes en se basant sur les métriques d'évaluations définies dans le chapitre précédent.

Troisièmement, nous allons présenter l'intégration de l'aspect source dans les modèles d'apprentissage profond afin de voir l'importance de cet aspect dans la découverte de la vérité. Ensuite, nous allons présenter l'implémentions des versions d'attaques.

Pour terminer ce chapitre, nous allons montrer les résultats du nouveau modèle de détection des fausses nouvelles "Fake news detector".

5.1 Outils utilisés

Nous avons fait recours à plusieurs technologies durant la réalisation de ce PFE. Des technologies pour l'implantation et d'autres por la gestion et le bon suivi du PFE. Nous les présentons dans ce qui suit :

- **Jupyter Notebook python** : est un environnement de programmation python interactif basé sur le Web permettant de créer des documents Jupyter Notebook.
- **Anaconda** : Anaconda est un utilitaire pour Python offrant de nombreuse fonctionnalité. Il offre par exemple la possibilité d'installer des librairies et de les utiliser dans des programmes.

- **Drive (5.7.3)** : est un service de stockage et de partage de fichiers dans le cloud.
- **Gitlab** : GitLab est un logiciel libre basé sur git proposant les fonctionnalités de wiki, un système de suivi des bugs, l'intégration continue et la livraison continue.
- **Magic plot(5.7.1)** : est une application de traçage technique, d'ajustement de courbe et d'analyse de données
- **MS office project (5.7.5)** : est un logiciel de gestion de projets édité par Microsoft.
- **Zotero (5.7.2)** : est un logiciel de gestion de références gratuit, libre et open source. Il permet de gérer des données bibliographiques et des documents de recherche.



FIG. 5.1 : Outils utilisés

5.2 Comparaison entre les méthodes de détection des fausses nouvelles

5.2.1 Collecte des données

Dans cette section, nous allons présenter la structure des jeux de données avant le nettoyage des données.

- **Données textuelles :**

- Kaggle ¹ : il contient initialement 20800 lignes et ses attributs sont :
 - * id : identifiant unique pour un article d'actualité
 - * titre : le titre d'un article d'actualité
 - * texte : le texte de l'article ; peut être incomplet
 - * label : une étiquette qui marque l'article comme potentiellement non fiable (1 : non fiable et 0 : fiable).

	id	title	author	text	label
0	0	House Dem Aide: We Didn't Even See Comey's Let...	Darrell Lucus	House Dem Aide: We Didn't Even See Comey's Let...	1
1	1	FLYNN: Hillary Clinton, Big Woman on Campus - ...	Daniel J. Flynn	Ever get the feeling your life circles the rou...	0

FIG. 5.2 : Exemple du dataset “Kaggle”

- Snopes ² : Il contient initialement 30112 lignes. La structure du jeu de données est la suivante :
 - * credLabel : crédibilité de l'allégation (chiffre entre 0 et 5)
 - * claimId : identifiant unique de l'allégation
 - * claimText : texte de l'allégation
 - * l'allégation-evidence : extrait pertinent de l'article d'évidence
 - * evidenceSource : la source de l'article d'évidence

	label	title	text	author
0	0	politics_christmas_bestbuy best buy chain eschewing use word christmas 20...	www.godlikeproductions.com	
1	0	politics_christmas_bestbuy best buy chain eschewing use word christmas 20...	www.sjpba.net	
2	0	politics_christmas_bestbuy best buy chain eschewing use word christmas 20...	www.englisher.net	

FIG. 5.3 : Exemple du dataset “Snopes”

- Gossip : Les attributs de ce jeu de données sont les mêmes que celui Snopes.
 - * credLabel : crédibilité de l'allégation (chiffre entre 0 et 5)
 - * claimId : identifiant unique de l'allégation
 - * claimText : texte de l'allégation

¹<https://www.kaggle.com/c/fake-news/overview>

²<https://www.snopes.com/>

- * l'allégation-evidence : extrait pertinent de l'article d'évidence
- * evidenceSource : la source de l'article d'évidence

	author	title	label
0	https://www.brides.com/story/teen-mom-jenelle-...	Teen Mom Star Jenelle Evans' Wedding Dress Is ...	0
1	https://www.dailymail.co.uk/tvshowbiz/article-...	Kylie Jenner refusing to discuss Tyga on Life ...	0
2	https://en.wikipedia.org/wiki/Quinn_Perkins	Quinn Perkins	0

FIG. 5.4 : Exemple du dataset "Gossip"

- Politifact ³ : Les attributs de ce jeu de données sont les mêmes que celui Snopes.
 - * credLabel : crédibilité de l'allégation (chiffre entre 0 et 5)
 - * claimId : identifiant unique de l'allégation
 - * claimText : texte de l'allégation
 - * l'allégation-evidence : extrait pertinent de l'article d'évidence
 - * evidenceSource : la source de l'article d'évidence

	author	label	title
0	Paul LePage	1	About 47 percent of able-bodied people in the ...
1	Battleground Texas	0	Says Dan Patrick has "called immigration into ...
2	Battleground Texas	0	In 2008, "only 54 percent of Latinos in Texas ...

FIG. 5.5 : Exemple du dataset "Politifact"

- PubHealth (SHAHI et al. 2020) : Il contient initialement 12288 lignes. Nous avons obtenu les colonnes suivantes :
 - * claimId : identifiant unique pour un article de presse
 - * label : labels (TRUE, FALSE, mixture, unproven et snopes)
 - * claim : texte de la déclaration.
 - * datePublished : date de publication de la déclaration.
 - * source : la source qui a rapporté la déclaration.
 - * subjects : le sujet de la déclaration.

	title	text	author	label
1	Bat from Shawnee County tests positive for rab...	Topeka television station KSNT reports that th...	https://www.ksnt.com/news/bat-tests-positive-f...	0
2	Germany has banned pork from school canteens b...	On 7 March 2016, British tabloid Express repor...	http://bnp.org.uk/news/regional/bnp-victory-br...	1

FIG. 5.6 : Exemple du dataset "PubHealth"

- Covid-19 : Il contient initialement 12805 lignes. ce jeu de données a la même structure que celle de PubHealth :
 - * claimId : identifiant unique pour un article de presse

³<https://www.politifact.com/>

- * label : labels (TRUE, FALSE, mixture, unproven et snopes)
 - * claim : texte de la déclaration.
 - * datePublished : date de publication de la déclaration.
 - * source : la source qui a rapporté la déclaration.
 - * subjects : le sujet de la déclaration.
- **Données tabulaires** : Nous allons présenter la structure de ce jeu de données avant les étapes de nettoyage :
 - Population : ce jeu de données décrit les villes et leur populations dans une année bien précise par des sources différentes. La structure de ce jeu de données est la suivante :
 - * Object : le nom de l'objet (le nom d'une ville)
 - * Property : le nom de l'attribut de l'objet (l'année)
 - * SourceID : le nom de la source
 - * Value : c'est la valeur qui a été donnée pour la propriété d'un objet par une source. (la valeur de la population d'une ville dans une année bien précise)

	SourceID	object	value	label
0	0 (68.162.248.83)	abudhabi_Population2006	1000230	1
1	1513217: Mohammedfairouz	abudhabi_Population2006	1850230	1

FIG. 5.7 : Exemple du dataset "Population"

- Weather : ce jeux de données décrit soit les températures, le degré d'humidité, le degré de pression des villes dans la même année par des sources différentes. La structure de ce jeu de données est la suivante :
 - * Object : le nom de l'objet (le nom d'une ville)
 - * Property : le nom de l'attribut de l'objet (température)
 - * SourceID : le nom de la source
 - * Value : c'est la valeur qui a été donnée pour la propriété d'un objet par une source. (la valeur de la température d'une ville)

	SourceID	object	value	label
298	msn	charlottefrijan2912_Visibility	10.0	0
305	msn	sandiegofrijan2912_Temperature	54.0	0

FIG. 5.8 : Exemple du dataset "Weather"

- Flights : ce jeux de données reporte les horaires de décollage et d'arrivée des vols et les portes de départ et d'arrivée des vols par des sources différents. La structure de ce jeu de données est la suivante :

- * Object : le nom de l'objet (le nom du vol)
- * Property : le nom de l'attribut de l'objet (l'heure d'arrivée)
- * SourceID : le nom de la source
- * Value : c'est la valeur qui a été donnée pour la propriété d'un objet par une source. (l'heure d'arrivée du vol)

	object	SourceID	value	label
0	AA-3468-CVG-MIA2011-12-20_ActualArrivalTime	aa	2011-12-20 09:26:00	1
1	AA-3468-CVG-MIA2011-12-20_ActualArrivalTime	flightexplorer	2011-12-20 09:24:00	1

FIG. 5.9 : Exemple dans du dataset “Flights”

5.2.2 Préparation des données

Afin que le nettoyage de données soit bien compris et explicite pour chaque jeu de donnée tout en prenant en considération les caractéristique de ce jeu de données et la méthode de détection des fausses nouvelles utilisée pour ce jeu de données. On a opté pour la création d'un notebook pour chaque jeu de données.

Pour que le processus de préparation des jeux de données soit explicite, on prendra un exemple à partir des jeux de données textuels et un autre à partir des jeux de données tabulaires.

- **Données textuelles** : exemple du jeu de données Kaggle
 - Enlever les valeurs nulles (en remplaçant la colonne texte par la colonne titre et vis-versa)

Deal with the null values

```
# Find Na
train.isnull().sum()
```

C:\Users\CBS Compter\anaconda3\envs\tensorflow7\lib\site-packages\ipykernel\ipkernel.py:287: DeprecationWarning: `should_run_async` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_cell` argument and any exception that happen during the transform in `preprocessing_exc_tuple` in IPython 7.17 and above.
and should_run_async(code)

```
id      0
title   558
author  1957
text    39
label   0
dtype: int64
```

- We are going to replace the null values in the column text by the values of the column title

```
# replace na in text
for n in range(0, len(train)):
    if pd.isnull(train['text'][n]):
        train['text'][n] = str(train['title'][n])
train.isnull().sum()
```

C:\Users\CBS Compter\anaconda3\envs\tensorflow7\lib\site-packages\ipykernel\ipkernel.py:287: DeprecationWarning: `should_run_async` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_cell` argument and any exception that happen during the transform in `preprocessing_exc_tuple` in IPython 7.17 and above.
and should_run_async(code)

C:\Users\CBS Compter\anaconda3\envs\tensorflow7\lib\site-packages\ipykernel_launcher.py:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
after removing the cwd from sys.path.

```
id      0
title   558
author  1957
text    0
label   0
dtype: int64
```

- We are going to replace the null values in the column title by the values of the column text

```
# replace na in title
for n in range(0, len(train)):
    if pd.isnull(train['title'][n]):
        train['title'][n] = str(train['text'][n])
train.isnull().sum()
```

C:\Users\CBS Compter\anaconda3\envs\tensorflow7\lib\site-packages\ipykernel\ipkernel.py:287: DeprecationWarning: `should_run_async` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_cell` argument and any exception that happen during the transform in `preprocessing_exc_tuple` in IPython 7.17 and above.
and should_run_async(code)

C:\Users\CBS Compter\anaconda3\envs\tensorflow7\lib\site-packages\ipykernel_launcher.py:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
after removing the cwd from sys.path.

```
id      0
title   0
author  1957
text    0
label   0
dtype: int64
```

```
# Find Na
train.isnull().sum()
```

C:\Users\CBS Compter\anaconda3\envs\tensorflow7\lib\site-packages\ipykernel\ipkernel.py:287: DeprecationWarning: `should_run_async` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_cell` argument and any exception that happen during the transform in `preprocessing_exc_tuple` in IPython 7.17 and above.
and should_run_async(code)

```
id      0
title   0
author  1957
text    0
label   0
dtype: int64
```

FIG. 5.10 : Notebook utilisé pour le nettoyage du jeu de données "Kaggle" - Suppression des valeurs nulles

- Enlever les doublons (dans ce cas, le jeu de données n'a pas de doublons)

Dealing with duplicate values

```
train.drop_duplicates().shape, train.shape
```

C:\Users\CBS Computer\anaconda3\envs\tensorflow7\lib\site-packages\ipykernel\ipkernel.py:287: DeprecationWarning: `should_run_async` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_cell` argument and any exception that happen during the transform in `preprocessing_exc_tuple` in IPython 7.17 and above.
and should_run_async(code)

```
((20800, 5), (20800, 5))
```

FIG. 5.11 : Notebook utilisé pour le nettoyage du jeu de données "Kaggle" - Suppression des doublons

- **Données tabulaires** : exemple du jeu de données Population

Dans ce cas, nous avons fait 3 notebooks pour préparer les données, un pour le nettoyage des données pour les méthodes d'apprentissage profond, un autre pour les méthodes probabilistes, et le dernier pour la méthode TruthFinder car elle peut pas manipuler les sources qui donnent une seule allégation.

[Preprocessing_Population_DL_methods.ipynb](#)

[Preprocessing_Population_Non_DL_methods.ipynb](#)

[Preprocessing_Population_TruthFinder.ipynb](#)

FIG. 5.12 : Les 3 notebooks utilisés pour la préparation des données

- Enlever les valeurs nulles (Dans ce cas, il n'y a pas d'objets nuls)

Null values ¶

```
claims.isnull().sum()
```

ObjectID	0
PropertyID	0
PropertyValue	0
SourceID	0
TimeStamp	49943
dtype:	int64

FIG. 5.13 : Notebook de suppression des valeurs nulles du jeu de données "Population"

- Enlever les doublons

Drop duplicates

```
claims.drop_duplicates().shape, claims.shape
```

```
((49943, 5), (49943, 5))
```

FIG. 5.14 : Notebook de suppression des doublons du jeu de données "Population"

- Enlever les objets qui n'ont pas de vérité de terrain

Not all the claims has their corresponding truths. We need to filter them out

```
truths[truths.object=='st.joseph_missouri_Population2000'].shape[0], claims[claims.object == 'st.joseph_missouri_Population2000']
(1, 0)

object_with_claims_and_truth = list(set(claims.object).intersection(set(truths.object)))

truths = truths.set_index('object').loc[object_with_claims_and_truth].reset_index()

claims = claims.set_index('object').loc[object_with_claims_and_truth].reset_index()

truths.shape, claims.shape
((301, 2), (1034, 3))
```

FIG. 5.15 : Notebook de suppression des objets non vérité de terrain du jeu de données “Population”

- Encoder les objets pour les méthodes probabilistes en utilisant LabelEncoder, nous avons proposé cet encodage car c'est une méthode simple de la classe sklearn.preprocessing qui code les entités avec une valeur comprise entre 0 et le nombre de classes -1.

Data Encoding

```
object_le = LabelEncoder()
claims['object_id'] = object_le.fit_transform(claims.object)
```

claims

	object	SourceID	value	object_id
0	stockholm_wisconsin_Population2000	5512121: CapitalBot	75	252
1	stockholm_wisconsin_Population2000	1960810: Nyttend	97	252
2	johnday_oregon_Population2000	249030: Ajbenj	1821	118
3	johnday_oregon_Population2000	5512121: CapitalBot	1821	118
4	johnday_oregon_Population2000	0 (70.103.54.101)	1987650	118
...
1014	utica_newyork_Population2000	266817: Cornellrockey	60651	266
1015	utica_newyork_Population2000	5512121: CapitalBot	60651	266

FIG. 5.16 : Notebook utilisés pour l'encodage des objets du jeu de données “Population”

- Garder les sources qui ont au moins deux allégations pour la méthode TruthFinder

Keep sources that have more then one claim

```
mask = claims.groupby(['SourceID']).nunique().value < 2
source_with_single_value = list(claims.groupby(['object']).nunique().value[mask].index)
```

FIG. 5.17 : Notebook de filtrage du jeu de données “Population”

Après le nettoyage des données, voici un tableau récapitulatif de ces données, (il faut savoir que le terme objet pour les données textuelles désigne le titre et pour les données tabulaires comme population et weather, il désigne la ville et pour flights il désigne le vol) :

Jeu de données	Nb d'objets	Nb de vraies déclarations	Nb de sources	Nb de déclarations
Kaggle competition	20321	10387	4201	20800
Snopes	4341	7507	12236	29242
Gossip	20743	16817	20691	22140
Politifact	10457	4541	2889	10464
Covid-19	11315	80	7298	12235
PubHealth	11874	6305	6113	11874
Weather	20905	215243	16	401395
Population	291	49352	624	49943
Flights	13050	161234	34	175691

TAB. 5.1 : Caractéristiques des jeux de données après le nettoyage

Voici un tableau récapitulatif des jeux de données tabulaires après ce nettoyage :

Jeu de données	Nb d'objets	Nb de vraies déclarations	Nb de sources	Nb de déclarations
Weather	20905	20905	16	278540
Population	291	291	624	10019
Flights	13050	13050	34	175691

TAB. 5.2 : Caractéristiques des jeux de données après le nettoyage dédié aux méthodes d'apprentissage profond

5.2.3 Méthodes de détection des fausses nouvelles existantes

Les méthodes sont réparties en deux catégories comme déjà expliqué :

1. Les méthodes d'apprentissage profond.
2. Les méthodes probabilistes.

Pour avoir plus de détails sur méthodes, veuillez consulter ce lien github : ⁴

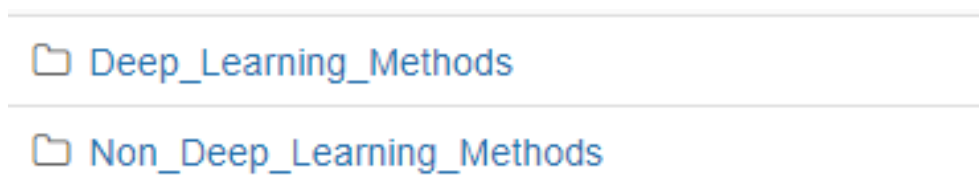


FIG. 5.18 : Notebooks utilisés pour le nettoyage des jeux des données

⁴https://github.com/LaureBerti/Deep_Learning_based_Fact-Checking

1. **Méthodes d'apprentissage profond** : Ces méthodes ont été organisées selon les jeux de données concernés par ces méthodes (5.19a).

Par exemple, prenons le cas, du répertoire de jeu de données Kaggle, voici toutes les méthodes d'apprentissage profond concernées par ce jeu de données (5.19b). Le même modèle peut être appliqué pour le titre et le texte de l'allégation.



(a) Le jeux de données

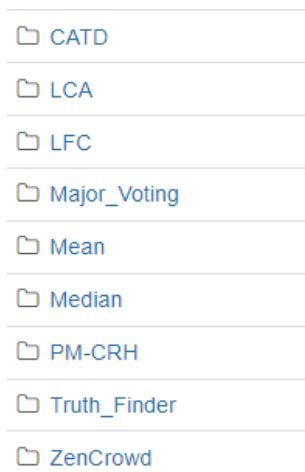


(b) les méthodes d'apprentissage profond concernées par Kaggle

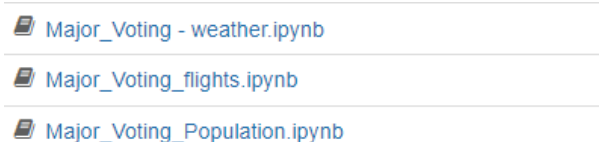
FIG. 5.19 : Notebooks méthodes d'apprentissage profond

2. **Méthodes probabilistes** : Voici les méthodes probabilistes que nous avons implémenté (5.20a).

Rentrons dans l'un des répertoires, par exemple "Majority Voting", on va trouver tous les jeux de données tabulaires concernés par cette méthode (5.20b).



(a) Les méthodes probabilistes implémentées



(b) Les jeux de données tabulaires concerné par Majority Voting

FIG. 5.20 : Notebooks méthodes probabilistes

5.2.4 Préparation de l'environnement de comparaison

Afin de préparer l'environnement parfait pour l'exécution des méthodes de détection des fausses nouvelles existantes et comparer ces dernières, on a opté pour l'utilisation du langage de programmation Python 3. Pour que la visualisation du code soit agréable, on a opté pour l'utilisation de Jupyter Notebook installé via Anaconda. Nous avons choisi aussi la version 3 de Python, ce qui a posé problème pour quelques méthodes probabilistes qui étaient codées en Python 2, donc il a fallu les ré-implémenter en Python 3. Autre problème qui s'est posé lors de l'exécution des méthodes était la performance de ma machine qui avait les caractéristiques suivantes : (CPU : i5, RAM : 8 Go). Il y a des méthodes qui n'arrivaient pas à s'exécuter sur ma machine avec le grand nombre d'itérations et le grand nombre de lignes des jeux de données. C'est pour cela qu'il fallait une machine plus puissante afin d'exécuter toutes les méthodes et permettre une comparaison juste entre ces dernières. Du coup, nous avons opté à l'utilisation du cluster du Laboratoire d'informatique et systèmes (LIS) afin d'atteindre les performances demandées.

5.2.4.1 Présentation du Cluster de LIS

L'accès à ce cluster s'est fait à travers l'interface de Gitlab après la création du compte.

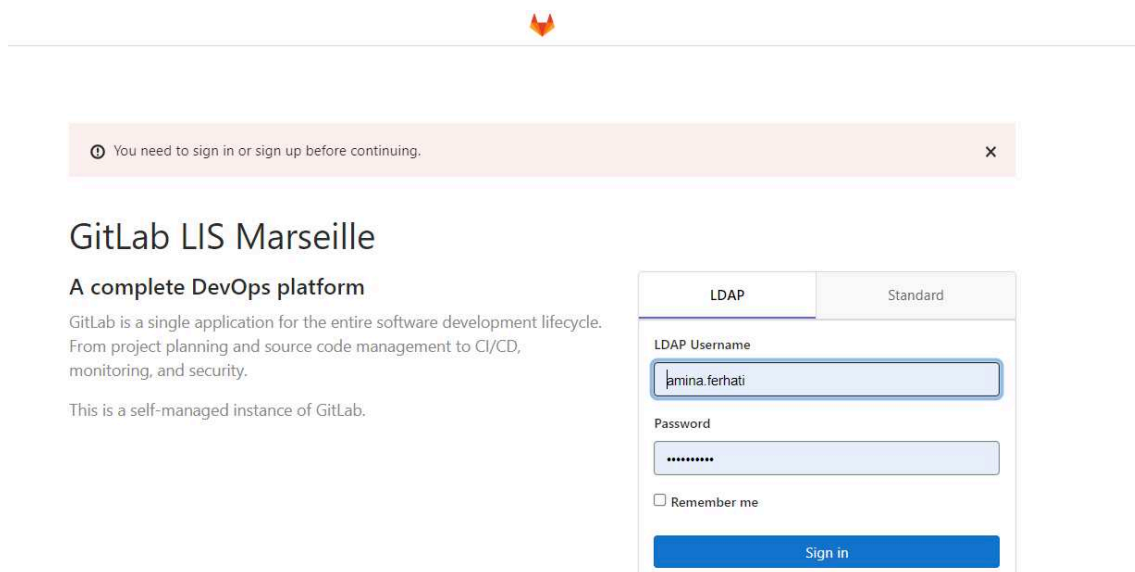


FIG. 5.21 : L'interface de Gitlab LIS Marseille

Voici l'espace wiki pour avoir accès à toute la documentation du Cluster.

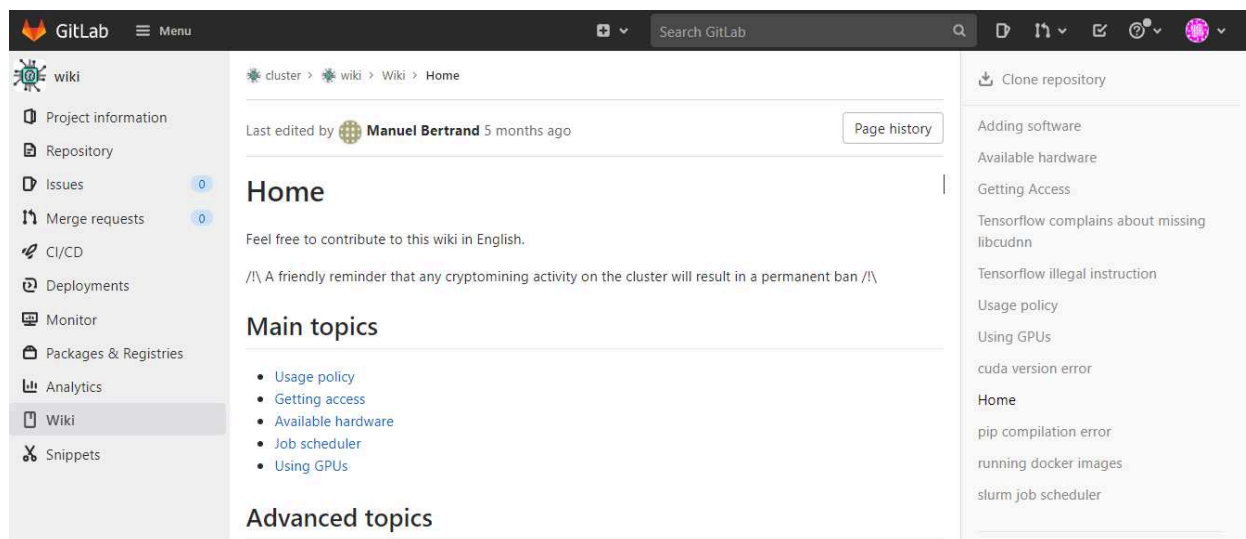


FIG. 5.22 : Documentation du Cluster

Cette section montre les différentes étapes pour installer l'environnement d'exécution python. Dans notre cas, c'était un fichier "requirements.txt" qui contenait tous les packages et les bibliothèques utilisés pour l'exécution des méthodes.

Le fichier "requirements.txt" a été récupéré de ma machine où j'ai installé mon environnement d'exécution au départ sur Anaconda.

Adding software

For stability and reproducibility, system admins will not install new software on the cluster. It is the user's responsibility to install new software in her home directory.

Installing python modules

First, you need to update your python tools

```
python -mpip install -U --user pip wheel virtualenv
```

As of 10/2020, you need to unload gnu9 modules:


```
module unload gnu9
```

Then, you can create virtual environments and install requirements inside them

```
virtualenv .venv
```

FIG. 5.23 : Les étapes d'installation des modules Python sur le Cluster

On peut voir dans cette section, les différents noeuds/machines virtuelles présentes sur le cluster et leur caractéristiques.

Last edited by  **Valentin Emiya** 9 months ago

Page history

Available hardware

The list of nodes is available on sms in `/usr/local/share/nodes.identities`

```

# compute hostnames
  nom      / CPUs / Core / Thread / Mem / GPU / X x Nom - Mem - GPUname - CUDA cap - Slu
*bolt2     / 2 / 8 / 32 / 72G / 0 /
*bolt3     / 2 / 8 / 16 / 64G / 0 /
*decoda2   / 2 / 8 / 16 / 80G / 0 /
*videosense1 / 2 / 8 / 16 / 56G / 0 /
*videosense4 / 2 / 8 / 16 / 32G / 0 /
*sensei1   / 2 / 6 / 12 / 128G / 2 / 2 x Tesla K40m - 12G - GK110B - 3.5 - kepl
*asfalda1  / 2 / 6noHT / 12 / 64G / 3 / (2) x Tesla K80 - 12G - GK210 - 3.7 - kepl
*lifnode1  / 2 / 8 / 16 / 96G / 2 / 1 x Tesla K40m - 12G - GK110B - 3.5 - kepl
*adnvideo1 / 2 / 8 / 16 / 128G / 4 / (4) x Tesla K80 - 12G - GK210 - 3.7 - kepl
*see4c1    / 2 / 8 / 16 / 128G / 4 / (4) x Tesla K80 - 12G - GK210 - 3.7 - kepl
*diflives1 / 2 / 10 / 20 / 192G / 8 / 8 x RTX 2080Ti - 11G - TU102 - 7.5 - turin
*lisnode2  / 2 / 10 / 20 / 192G / 6 / 6 x GTX 1080Ti - 11G - GP102 - 6.1 - pasca
*lisnode3  / 2 / 8 / 16 / 192G / 4 / 2 x GTX 1080Ti - 11G - GP102 - 6.1 - pasca

```

FIG. 5.24 : Les différents noeuds/machines virtuelles présents sur le Cluster

Nous avons choisi la machine la plus puissante pour l'exécution (noeud) des méthodes de détection des fausses nouvelle existantes

Nom du noeud	CPUs	Core	Thread	Mem	GPU
diflives1	2	10	20	192G	8

TAB. 5.3 : Caractéristique de la machine choisie

Vu que le cluster ne dispose pas d'interface graphique, il a fallu transformer les notebooks des méthodes en fichier (.py) et les exécuter sur le cluster.

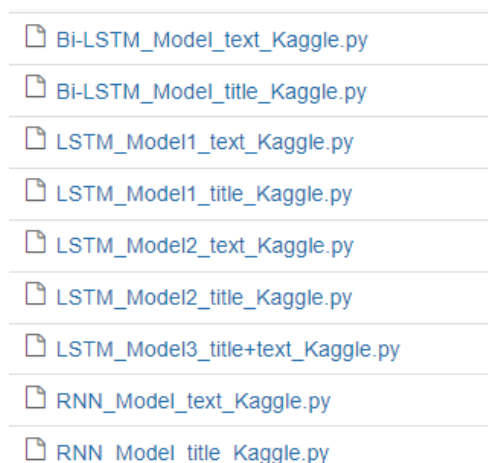


FIG. 5.25 : Conversion des notebooks en fichiers “.py”

On pouvait bien travailler sur *Google Collab* qui possède une interface graphique mais la version gratuite est limitée à 13 Go. Donc, le problème de mémoire n'est toujours pas résolu.

```
C:\Users\CBS Compter>ssh sms
amina.ferhati@139.124.22.4's password:
amina.ferhati@sms-ext's password:
Welcome to the LIS Cluster - Featuring absolutely no backup of your files

LIS      Wiki & Infos: COMING SOON
         CPU Jobs : COMING SOON
         GPU Jobs : COMING SOON

Please note that any bad behavior on the cluster like crypto-mining,
tampering with your rights/resources or anything that might endanger
other users jobs will result in perma-ban.

NO INTENSIVE COMPUTING PROCESS SHOULD BE LAUNCH ON THIS SERVER

/!\ This server does not have any backup of your files /!\

Last login: Thu Sep  9 14:11:11 2021 from 139.124.22.4
[amina.ferhati@sms ~]$ client_loop: send disconnect: Connection reset
client_loop: send disconnect: Unknown error

C:\Users\CBS Compter>
```

FIG. 5.26 : Terminal de connexion au cluster

Il a fallu recopier tous les dossiers au clusters et activer l'environnement d'exécution nommé "tensorflow7" qui contient toutes les bibliothèques et tous les packages d'exécution et exécuter le fichier (Dans 5.27, on voit clairement que le fichier ".py" contenant le modèle Bi-LSTM est exécuté sur le jeux de données Covid-19).

```
/!\ This server does not have any backup of your files /!\

Last login: Thu Sep  9 15:18:03 2021 from 139.124.22.4
[amina.ferhati@sms ~]$ cd FakeNews/
[amina.ferhati@sms FakeNews]$ . tensorflow7/bin/activate
(tensorflow7) [amina.ferhati@sms FakeNews]$ cd FN/Baseline_Methods/Deep_Learning_Methods/Notebook/Covid_19/
(tensorflow7) [amina.ferhati@sms Covid_19]$ srun --pty --nodelist=dflives1 --time=36:00:00 Bi-LSTM_Model_text_Covid-19.py
```

FIG. 5.27 : Exécution du modèle Bi-LSTM sur le cluster

5.2.5 Choix des métriques et des méthodes d'évaluation

Afin de comparer les méthodes de détection des fausses nouvelles existantes, nous avons choisi les métriques sur lesquelles on doit se baser et pour avoir ces métriques, il a fallu mettre en place des techniques de validation.

Dans notre cas, on a choisi la technique de validation K-folds avec $K = 10$.

Training the model

```
: start=time.time()
# fix random seed for reproducibility
seed = 7
np.random.seed(seed)
# define 10-fold cross validation test harness
kfold = StratifiedKFold(n_splits=10, shuffle=True, random_state=seed)
cvscores1 = []
cvscores2 = []
cvscores3 = []
cvscores4 = []
num_iter=0

for train, test in kfold.split(X_final, y_final):
    num_iter=num_iter+1
    model=get_model(vo_size,embedding_vector_feature,sent_length)
    # Fit the model
    history=model.fit(X_final[train], y_final[train], epochs=10, batch_size=256,verbose=0)
    # evaluate the model
    scores = model.evaluate(X_final[test], y_final[test], verbose=0)
    print("-----subset number", num_iter,"-----")
    print("%s: %.2f%%" % (model.metrics_names[1], scores[1]*100))
    cvscores1.append(scores[1] * 100)
    print("%s: %.2f%%" % (model.metrics_names[2], scores[2]*100))
    cvscores2.append(scores[2] * 100)
    print("%s: %.2f%%" % (model.metrics_names[3], scores[3]*100))
    cvscores3.append(scores[3] * 100)
    print("%s: %.2f%%" % (model.metrics_names[4], scores[4]*100))
    cvscores4.append(scores[4] * 100)
print("-----" )
print("accuracy", "%.2f%% (+/- %.2f%%)" % (np.mean(cvscores1), np.std(cvscores1)))
print("precision", "%.2f%% (+/- %.2f%%)" % (np.mean(cvscores2), np.std(cvscores2)))
print("recall", "%.2f%% (+/- %.2f%%)" % (np.mean(cvscores3), np.std(cvscores3)))
print("f1-measure", "%.2f%% (+/- %.2f%%)" % (np.mean(cvscores4), np.std(cvscores4)))
end=time.time()
```

FIG. 5.28 : Notebook de technique de validation K-folds

```
-----subset number 8 -----
accuracy: 90.98%
precision_mesure: 87.29%
recall_mesure: 95.68%
f1_mesure: 91.08%
-----subset number 9 -----
accuracy: 91.37%
precision_mesure: 87.42%
recall_mesure: 96.93%
f1_mesure: 91.63%
-----subset number 10 -----
accuracy: 90.60%
precision_mesure: 88.32%
recall_mesure: 93.65%
f1_mesure: 90.63%
-----
accuracy 91.07% (+/- 0.84%)
precision 88.13% (+/- 1.57%)
recall 94.75% (+/- 1.97%)
f1-measure 91.04% (+/- 0.86%)
```

FIG. 5.29 : Résultat de K-folds avec $K=10$

5.3 Évaluation des méthodes de détection des fausses nouvelles existantes

Voici le tableau ci-dessous qui résume les différents résultats obtenus par les méthodes d'apprentissage profond sur les données textuelles :

Au niveau de ce tableau, il y a des résultats qui manquent pour la simple raison que ces modèles prennent énormément de temps pour s'exécuter ce qui fait que le connexion au cluster se coupe (à partir de 12h).

```
(tensorflow7) [amina.ferhati@sms population]$ client_loop: send disconnect: Connection reset
```

FIG. 5.30 : La déconnexion du cluster

		Mesures				
		Accuracy	Precision	Recall	F1-score	Time
Kaggle dataset	LSTM1 model title	0.909	0.882	0.942	0.909	986.799
	LSTM1 model text	0.889	0.882	0.902	0.887	39162.179
	LSTM2 model title	0.914	0.886	0.949	0.914	630.878
	LSTM2 model text	0.903	0.896	0.913	0.901	38750.100
	LSTM3 model title and text	0.991	0.993	0.988	0.990	5639.225
	Bi-LSTM model title	0.912	0.883	0.949	0.912	1830.157
	Bi-LSTM model text	0.885	0.874	0.900	0.884	72126.685
	RNN Model title	0.912	0.497	0.100	0.659	3465.553
	RNN Model text	/	/	/	/	/
Politifact dataset	LSTM1 model title	0.582	0.519	0.448	0.469	441.383
	LSTM2 model title	0.586	0.523	0.478	0.484	326.21
	Bi-LSTM model title	0.581	0.513	0.453	0.469	916.253
	RNN Model title	0.581	0.435	0.990	0.596	2279.753
Gossip dataset	LSTM1 model title	0.808	0.253	0.103	0.145	960.855
	LSTM2 model title	0.807	0.252	0.108	0.150	608.064
	Bi-LSTM model title	0.804	0.253	0.09	0.135	1710.605
	RNN Model title	0.809	0.249	0.257	0.251	3196.736
Snopes dataset	LSTM1 model title	0.892	0.8914	0.894	0.892	1293.644
	LSTM1 model text	0.973	0.980	0.983	0.981	73240.231
	LSTM2 model title	0.908	0.877	0.949	0.908	571.714
	LSTM2 model text	0.902	0.895	0.912	0.900	35959.898
	LSTM3 model title and text	0.859	0.861	0.953	0.890	7685.999
	Bi-LSTM model title	0.900	0.918	0.949	0.931	2247.605
	Bi-LSTM model text	/	/	/	/	/
	RNN Model title	0.841	0.742	0.100	0.847	1460.599
	RNN Model text	0.986	0.742	0.1000	0.847	4691.817
Covid-19 dataset	LSTM1 model title	0.994	0.994	0.999	0.996	257.674
	LSTM1 model text	0.984	0.984	0.999	0.992	13524.001
	LSTM2 model title	0.983	0.1000	0.991	0.984	184.389
	LSTM2 model text	0.983	0.984	0.100	0.991	7152.829
	LSTM3 model title and text	0.995	0.996	0.998	0.997	1673.289
	Bi-LSTM model title	0.990	0.993	0.996	0.995	432.181
	Bi-LSTM model text	/	/	/	/	/
	RNN Model title	0.892	0.984	0.1000	0.991	1305.558
	RNN Model text	0.986	0.742	0.1000	0.847	4691.817
PubHealth dataset	LSTM1 model title	0.677	0.657	0.668	0.651	550.767
	LSTM1 model text	0.712	0.680	0.747	0.6950	29172.008
	LSTM2 model title	0.687	0.677	0.638	0.647	355.769
	LSTM2 model text	00.775	0.727	0.824	0.765	22545.880
	LSTM3 model title and text	0.84	0.801	0.891	0.839	3498.003
	Bi-LSTM model title	0.686	0.669	0.662	0.654	901.173
	Bi-LSTM model text	0.744	0.706	0.782	0.730	59647.350
	RNN Model title	0.677	0.470	0.997	0.633	2216.930
	RNN Model text	/	/	/	/	/

FIG. 5.31 : Accuracy/precision/recall/Fscore/ temps d'exécution (s) des méthodes d'apprentissage profond sur les données textuelles

Voici le tableau ci-dessous qui résume les différents résultats obtenus par les méthodes d'apprentissage profond et probabilistes sur les données textuelles et tabulaires :

		Mesures				
		Accuracy	Precision	Recall	F1-score	Time
Population dataset	NN model (MARSHALL et al. 2017)	0.825	0.872	0.746	0.788	539.49
	LSTM1 model	0.988	0.988	0.100	0.993	480.120
	CATD	0.711	/	/	/	11.15
	MV	0.752	/	/	/	0.374
	CRH	0.642	/	/	/	2.185
	Zen Crowd	0.807	/	/	/	6.217
	Mean	MAE 584576989.550 RMSE 5958317217.637	/	/	/	0.332
	Median	MAE 11024.931271477662 RMSE 122719.7012624712	/	/	/	0.164
Weather dataset	NN model (MARSHALL et al. 2017)	0.935	0.779	0.879	0.811	13122.201
	LSTM1 model	0.909	0.882	0.942	0.909	986.799
	CATD	0.523	/	/	/	34.15
	MV	0.464	/	/	/	28.46
	CRH	0.674	/	/	/	502.365
	Zen Crowd	0.557	/	/	/	2302.953
	Mean	MAE 12.824 RMSE 29.731	/	/	/	1.14
	Median	MAE 1.532 RMSE 3.416	/	/	/	2.326
Flights dataset	NN model (MARSHALL et al. 2017)	0.935	0.779	0.879	0.811	13122.201
	LSTM1 model	0.917	0.917	0.100	0.955	1620.524
	CATD	0.711	/	/	/	11.158
	MV	0.132	/	/	/	17.873
	CRH	0.642	/	/	/	2.185
	Zen Crowd	0.765	/	/	/	205.76

FIG. 5.32 : Accuracy/precision/recall/Fscore/temps d'exécution (s) des méthodes destinées pour les données tabulaires

5.3.1 Interprétation des résultats de comparaison entre les méthodes d'apprentissage profond sur les données textuelles

Pour le premier tableau (5.31) :

- Le même modèle LSTM1/LSTM2/RNN/Bi-LSTM avec différentes colonnes (titre et texte) : On remarque que tous ces modèles donnent de meilleurs résultats avec la colonne titre, malgré qu'avec la colonne texte on prend 100 mots par phrase à la place de 20 mots par phrase (quand il s'agit de la colonne titre). On peut expliquer cela, car les 20 mots de la colonne titre sont plus représentatifs et donnent plus d'informations sur la nature de la nouvelle.
- Le même modèle LSTM1/LSTM2/RNN/Bi-LSTM avec deux différentes colonnes (titre et texte) : En voyant le temps c'est tout à fait logique que le modèle traitant la

colonne texte prend plus de temps qu'en traitant la colonne titre (20 mots/ phrase) car il y a plus de données avec la colonne texte (100 mots/ phrase).

- LSTM3 qui exploite la colonne titre et texte est mieux que LSTM1 et LSTM2 : Ce modèle tire avantage des deux architectures LSTM1 et LSTM2 et exploite deux colonnes différentes complémentaires du même jeu de données.
- RNN est le meilleur modèle pour les jeux de données qui ont une seule colonne (titre) : Dans les jeux de données qui disposent uniquement de la colonne titre et qui n'ont pas de colonne texte comme Politifact et Gossip, on remarque que le modèle RNN a donné les meilleurs résultats car c'est un modèle qui est adéquat pour les entrées séquentielles (textuelles). Le modèle évalue chaque mot d'une phrase, l'un après l'autre, mais en gardant en mémoire les mots qu'il a déjà traités. Cela permet au modèle de comprendre le contexte de chaque mot et sa place dans la phrase car les mots précédemment traités vont être analysés de manière récurrentes. Dans le cas de Gossip, RNN model a utilisé 68 mots et il semble qu'avec ce nombre de mots il arrive à extraire plus d'informations à partir des nouvelles mieux que 20 mots (nombre de mots utilisés par LSTM1 model) qui semble peu ou bien 100 mots (nombre de mots utilisés par LSTM2 model) qui semble beaucoup et distrait le modèle.
- le jeu de données qui a donné les résultats les moins performants c'est Gossip et Politifact. Gossip, pour la simple raison que ses classes sont déséquilibrées malgré que le nombre de lignes dans le jeu de données est grand. Le pourcentage des vraies allégations dans Gossip est de 75.95% .
Par contre Politifact a un nombre de lignes non suffisant (10464 lignes)en le comparant avec le reste des jeux de données.

Pour le 2 ème tableau (5.32) :

- Le jeu de données tabulaire qui a donné les résultats les plus faibles est Population car il possède le plus petit nombre de lignes.
- Les méthodes d'apprentissage profond ont donné les meilleurs résultats sur les données tabulaires, car elles utilisent l'aspect séquentiel et temporel des jeux de données population et weather et flights.
- Zen-Crowd est la méthode probabiliste qui a donné les meilleurs résultats sur tous les jeux de données, comme : population car ce sont des réponses données par des travailleurs dans une plateforme de crowd-sourcing (ce n'est pas difficile d'estimer la population d'une ville).
- Le temps d'exécution des méthodes probabilistes est moins que le temps d'exécution des méthodes de l'apprentissage profond car les méthodes statistiques sont des modèles simples qui ne contiennent pas beaucoup de couches complexes comme ceux de l'apprentissage profond.
- La médiane est la méthode numérique la plus performante par rapport à la moyenne car elle estime la vérité avec des valeurs qui existent déjà dans le jeu de données,

mais ce n'est pas pareil pour la méthode moyenne qui considère que la moyenne des valeurs d'un objet est la vraie valeur de cet objet.

- Le modèle d'apprentissage profond qui est destiné uniquement pour les données textuelles et que nous avons transformé pour qu'ils soient exploitables par les données tabulaires ont donné les meilleurs résultats par rapport au modèle (MARSHALL et al. 2017) qui était destiné uniquement pour les données tabulaires.
- le modèle e (MARSHALL et al. 2017) est plus coûteux en terme de temps d'exécution car il est plus complexe : en plus des couches, il doit construire la matrice source-valeur (mettre 1 la cellule (S_0, V_0) si la source S_0 a donné la valeur V_0 pour objet, 0 sinon), la construction de cette matrice prend énormément de temps.
- On ne peut tester les méthodes Moyenne et Médiane sur les données tabulaire Flights car elles sont pas numériques.
- Les résultats du jeu de données Flights sont les meilleurs car ce jeu de données a le plus grand nombre de lignes ce qui facilite l'entraînement des méthodes.

Pour conclure cette comparaison, nous avons constaté que les méthodes d'apprentissage profond sont plus performantes que les méthodes probabilistes car elles ont tendance à correspondre mieux l'aspect séquentiel des données. Malgré la performance de ces méthodes d'apprentissage profond, elles sont coûteuses en terme de temps.

Les modèles d'apprentissage profond qui sont destinés uniquement pour les données textuelles et que nous avons transformés pour qu'ils soient exploitables par les données tabulaires ont donné de meilleurs résultats sur les données tabulaires que le modèle d'apprentissage profond (MARSHALL et al. 2017) qui était destiné pour les données tabulaires.

5.4 Intégration de l'aspect source dans les modèles d'apprentissage profond

Nous avons constaté lors de l'étude bibliographique l'importance de la source dans les modèles de détection des fausses nouvelles. C'est pour cela qu'on a implémenté deux formules de calcul de crédibilité des sources qu'on a déjà décrites dans le chapitre "Conception".

5.4.1 Interprétation des résultats de la première formule de calcul de crédibilité

Voici quelques exemples des sources qui existent dans le jeu de données "Kaggle" :


```
train['author'].value_counts()

Pam Key                243
admin                  193
Jerome Hudson          166
Charlie Spiering       141
John Hayward           140
...
Manohla Dargis, Wesley Morris and A.O. Scott    1
Sam Wenkert                                     1
Jomo Merritt                                    1
Alanna Ketler                                   1
Lizette Alvarez, Jess Bidgood, Mitch Smith and Sabrina Tavernise  1
Name: author, Length: 4201, dtype: int64
```

FIG. 5.33 : Exemples des sources qui existent dans le jeu de données "Kaggle"

Afin d'évaluer ces formules de crédibilité, on a décidé de calculer la métrique d'exactitude en fonction de la taille du jeu de données "test". Nous avons fait varier la taille entre 5% et 30% de la taille globale du jeu de données.

Avec la contrainte du temps, toutes les expériences sont faites sur le modèle LSTM 1 car il a une architecture simple et donne de bons résultats et il n'est pas coûteux.

Afin de bien comparer les résultats obtenus, nous avons décidé de récupérer l'exactitude, le nombre de sources total, le nombre de sources nulles, le nombre de sources fiables (les sources qui ont plus de 50% de degré de fiabilité) dans chaque itération.

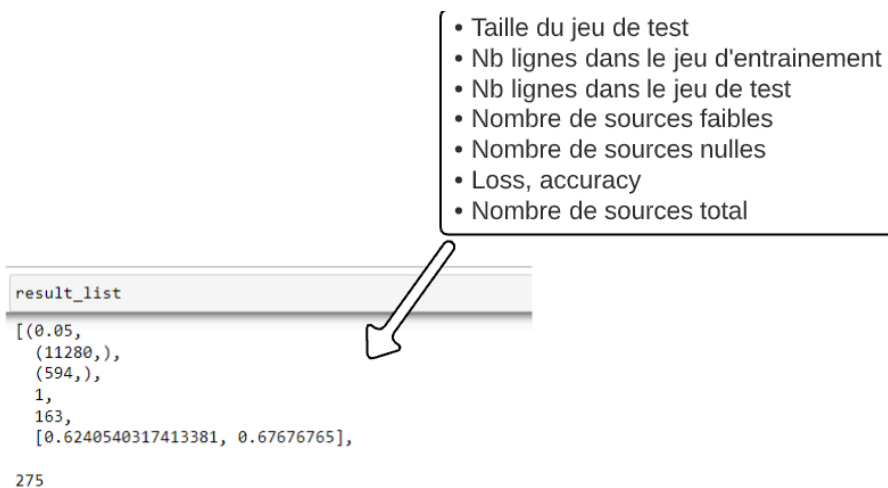


FIG. 5.34 : Les résultats récupérés

Voici les résultats sur le jeu de données Kaggle et de PubHealth :

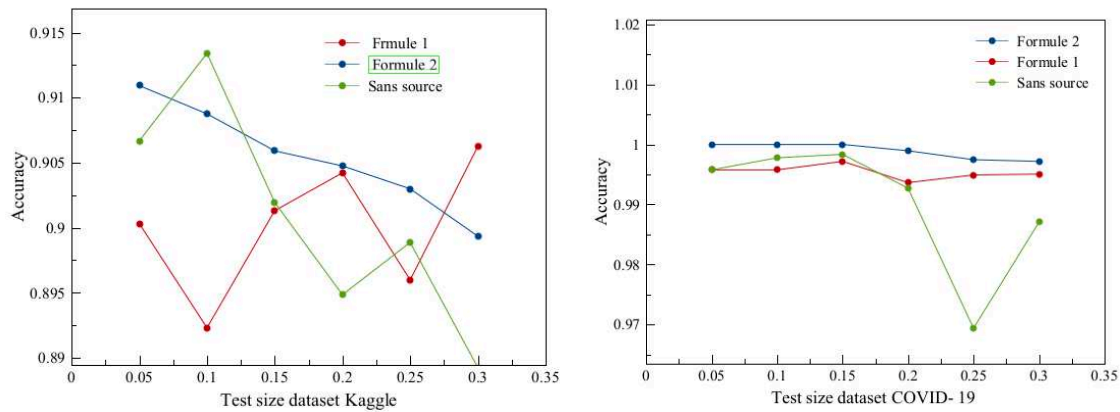


FIG. 5.35 : Les résultats sur le jeu de données Kaggle et Covid-19

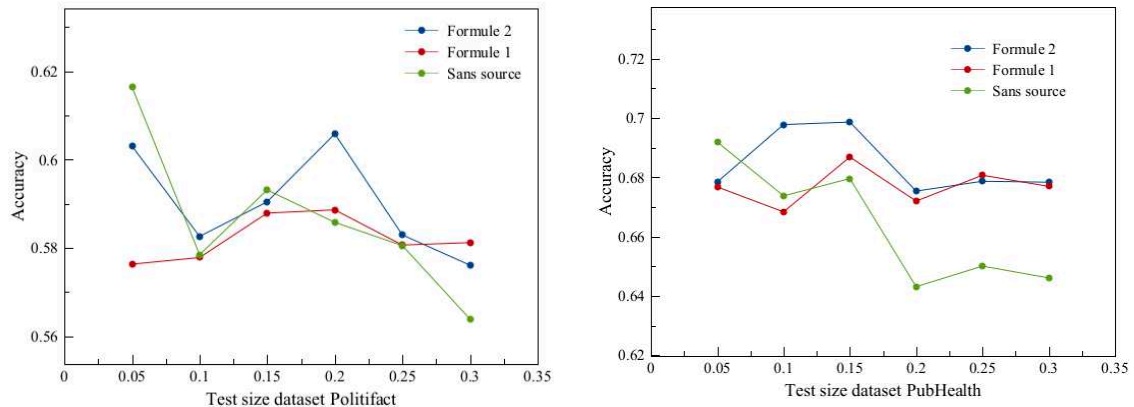


FIG. 5.36 : Les résultats sur le jeu de données Politifact et de PubHealth

A partir des résultats obtenus, on constate que :

Les modèles sans la composante source sont moins performants qu'avec la source, il y a uniquement quelques points où le modèle sans source est plus performant qu'avec la source et on peut expliquer cela en disant que les lignes choisies aléatoirement dans le jeu de données test contiennent du texte plus pertinent pour détecter sa nature.

On remarque qu'à l'augmentation de la taille du jeu de données "test", l'exactitude du modèles avec les différentes formules de calcul de crédibilité augmente, car, plus de source sont incluses dans le jeu de données, plus grand sera l'impact de la composante source sur le modèle. On ne s'attend pas à ce que l'exactitude augmente toujours avec l'augmentation de la taille du jeu de données test, car à partir d'une certaine taille, le modèle ne peut pas s'entraîner bien, vu que la majorité des lignes sont mises dans le jeu de données test.

La 2-ème formule (celle avec la couleur bleue) a donné les meilleurs résultats pour l'exactitude sur l'ensemble des jeux des données car elle valorise le taux de participation de la source et le nombre de vraies allégations et fausses allégation qu'elle a rapportées. Avec le jeu de données Kaggle, la première formule (celle avec la couleur rouge) donne des résultats bien pire que les résultats du modèle sans source car elle se débarrasse de beaucoup de sources en leur affectant des valeurs nulles.

Le seul point dans le jeu de données Kaggle où la première formule a donnée de meilleurs résultats que la deuxième, c'est quand la taille du jeu de données test est à 0.3 et cela à cause du nombre de sources utilisées car plus de sources pousse le modèle à s'entraîner mieux. Avec la première formule, on a 2015 source qui est plus grand que 1987 source obtenues par la deuxième formule. La même explication s'applique sur le jeu de données PubHealth quand la taille du jeu de données test est à 25%.

Il faut savoir aussi que l'une des raisons qui ont aidé la deuxième formule à être la meilleure c'est la normalisation de ses valeurs, car de cette façon, il est plus facile de créer plus de sources faibles (qui dépasse 50% de taux de crédibilité).

5.5 Implémentation des attaques de désinformation

Dans cette section, nous avons décidé de mener des attaques contre les modèles d'apprentissage profond auxquels on a ajouté la composante source. Vu que la 2^{ème} formule de calcul de crédibilité a donné les meilleurs résultats, nous avons calculé la crédibilité de ces sources en utilisant la 2^{ème} formule.

5.5.1 La première version de l'attaque

Nous avons décidé d'attaquer uniquement les données textuelles, en utilisant la négation (la forme négative).

Avec la contrainte du temps, toutes les expériences sont faites sur le modèle LSTM 1 car il a une architecture simple et donne de bons résultats et il n'est pas coûteux en terme de temps d'exécution.

Voici les deux notebooks avec lesquels cette attaque a été implémentée :


 [LSTM_Model1_Kaggle_source_formule 3_general_attack-version2.ipynb](#) [LSTM_Model1_Kaggle_source_formule 3_general_attack.ipynb](#)

FIG. 5.37 : Netbooks avec lesquels cette attaque a été implémentée

Comme on peut le voir sur la figure (5.38), à chaque fois qu'on augmente la taille du jeu de données test, le nombre de "not" inséré augmente. C'est tout à fait logique car plus d'allégations vont être injectées par des "not".

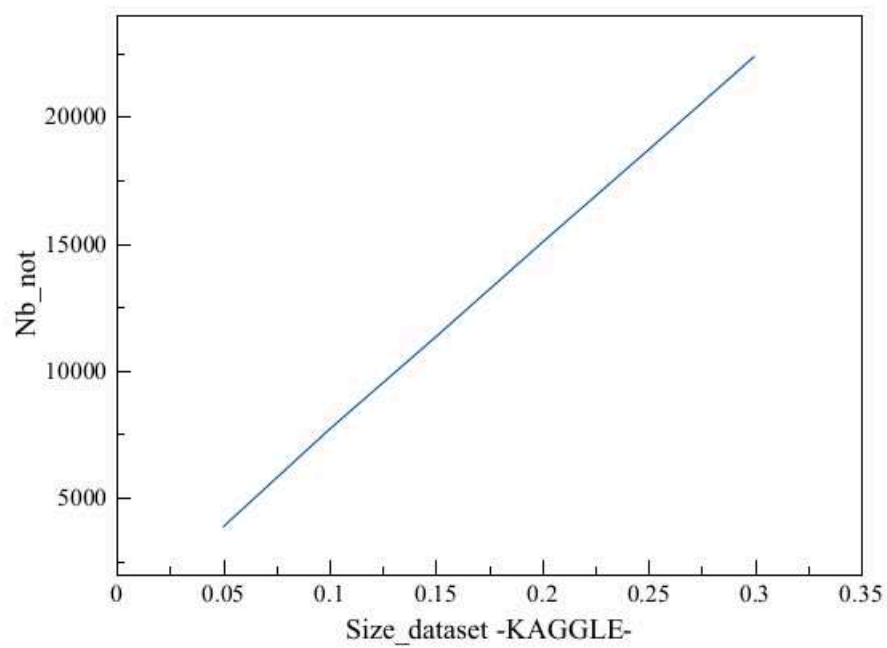


FIG. 5.38 : Comparaison entre le la taille de jeu de données et le nombre des "not"

Une série d'expériences ont été faites pour voir l'effet du nombre de "not" sur l'exactitude du modèle en variant la taille du jeu de données test entre 5% et 35%.

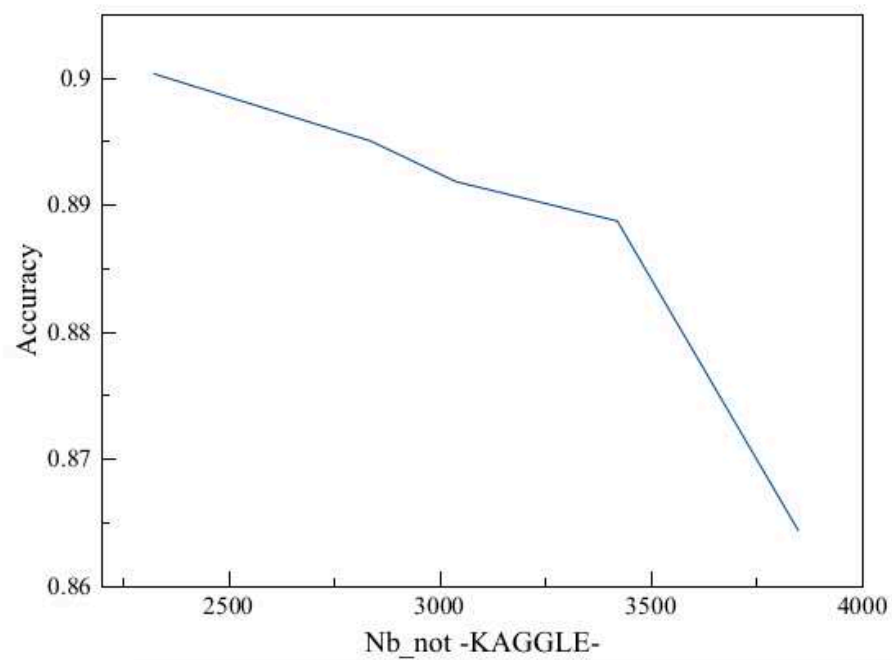


FIG. 5.39 : Effet du nombre de "not" sur l'exactitude du modèle - Kaggle

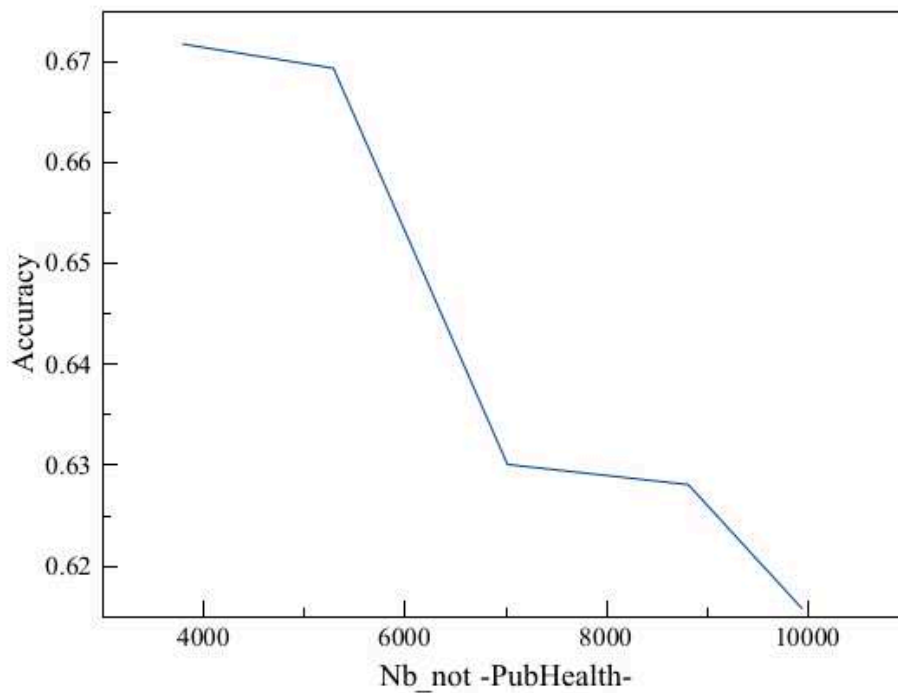


FIG. 5.40 : Effet du nombre de "not" sur l'exactitude du modèle - PubHealth

Comme les graphes le montrent, notre attaque a réussi à baisser l'exactitude du modèle LSTM 1 sur les deux jeux de données. Il est important de mentionner que ces "not" insérés ont perturbé le modèle d'un point de vue sémantique, ce qu'il l'a rendu moins performant.

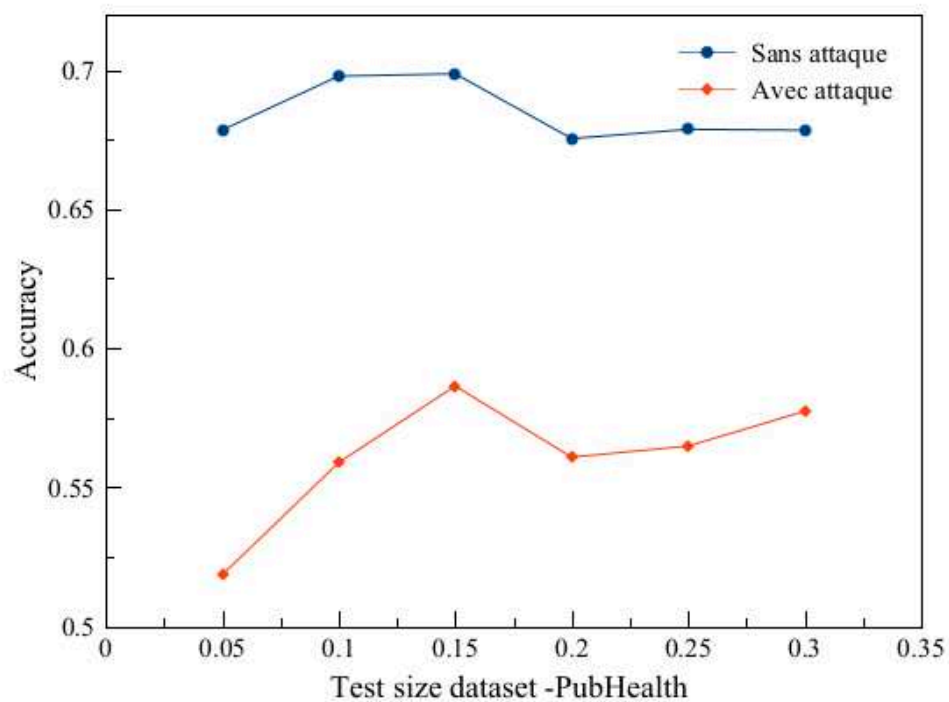


FIG. 5.41 : Étude sur l'effet attaque sur l'exactitude - PubHealth

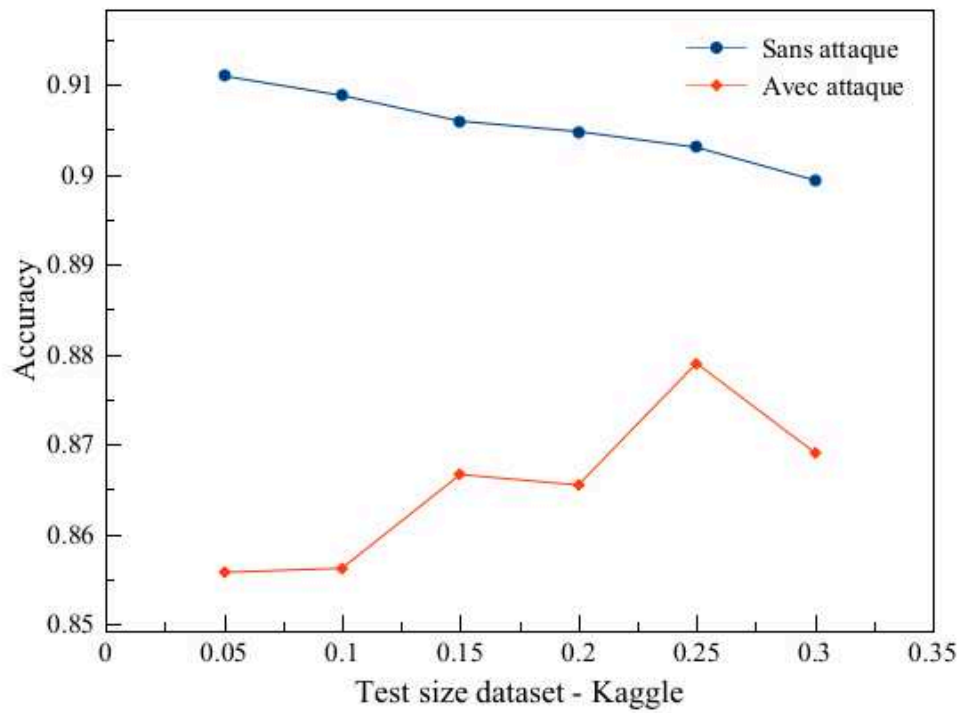


FIG. 5.42 : Étude sur l'effet attaque sur l'exactitude - Kaggle

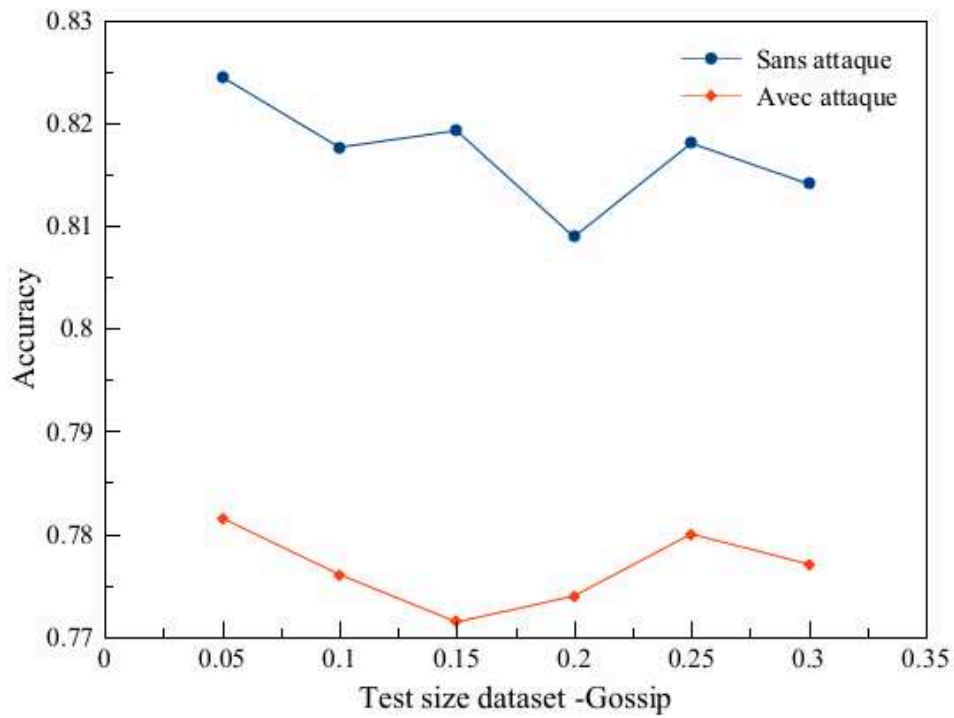


FIG. 5.43 : Étude sur l'effet attaque sur l'exactitude - Gossip

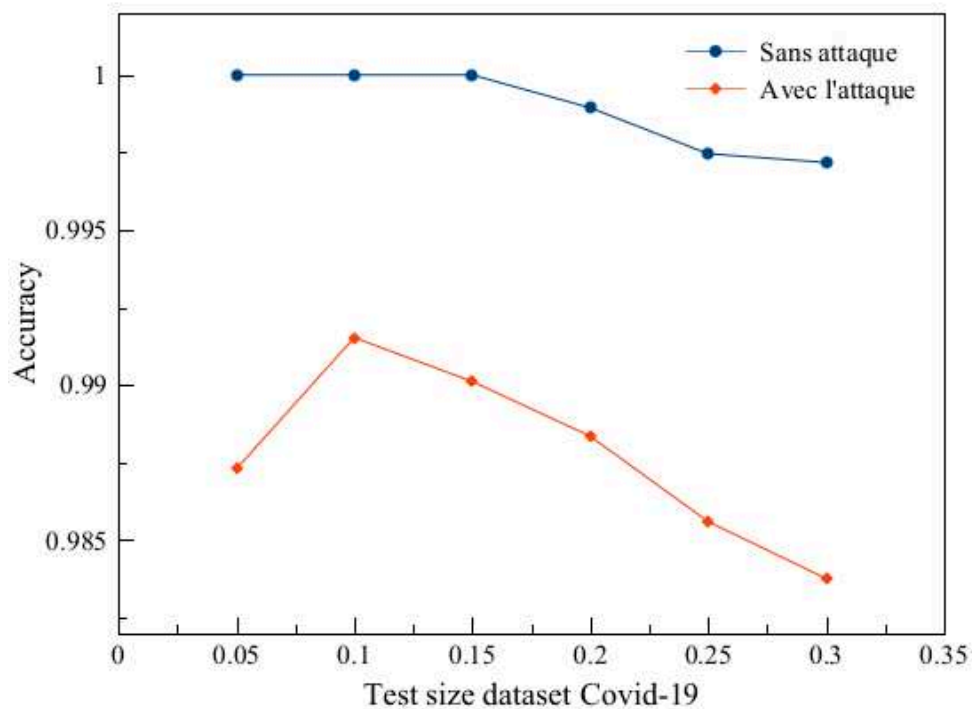


FIG. 5.44 : Étude sur l'effet attaque sur l'exactitude - Covid-19

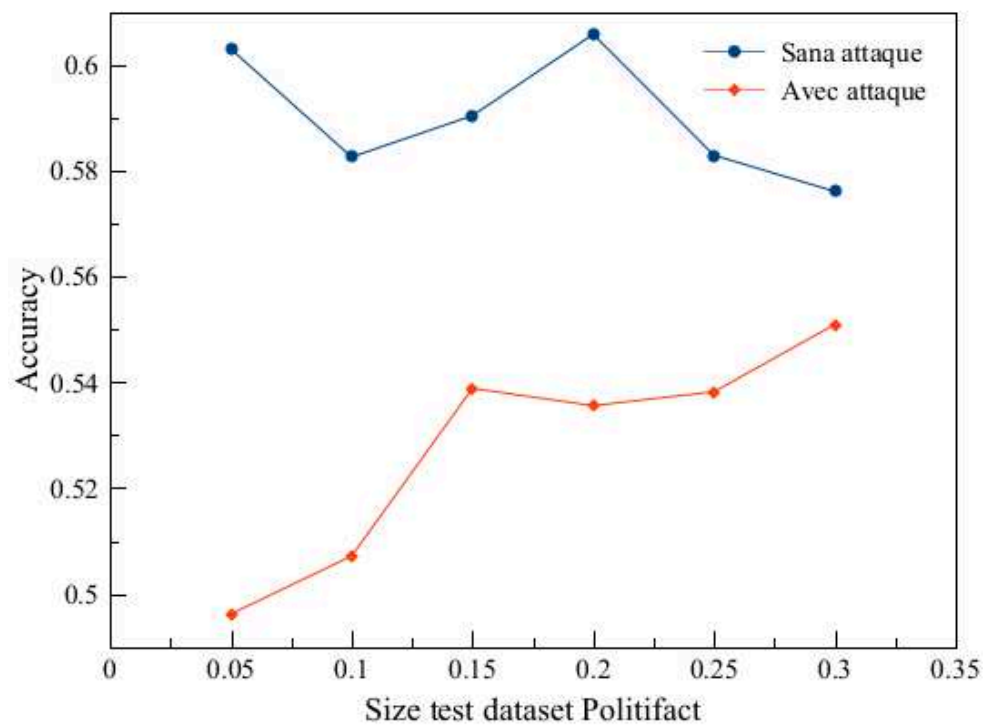


FIG. 5.45 : Étude sur l'effet attaque sur l'exactitude - Politifact

5.5.2 La 2-ème version de l'attaque

Cette attaque a été testée sur le modèle d'apprentissage profond MARSHALL et al. 2017 avec le jeu de données "Population".

Voici l'implantation des différentes étapes de l'algorithme :

- Faire un profilage pour l'allégation ciblée

```
def object_profiling(object_target):
    list1={}
    list2={}
    src=[]
    #for object_value in objects.keys():
    object_value='buffalo_newyork_Population2000'
    values_0=(claims.loc[claims['object'] == object_value, 'value'].value_counts(dropna=True).keys().tolist())
    counts_0=(claims.loc[claims['object'] == object_value, 'value'].value_counts(dropna=True).tolist())
    value_dict = dict(zip(values_0, counts_0))
    for val in value_dict.keys():
        #occ={val:value_dict[val]}
        #occ=dict(occ)
        #list2.update(occ)
        lines=claims.loc[(claims['object'] == object_value) & (claims['value'] == val)]
        src=[]
        for index, line in lines.iterrows():
            source=line['SourceID']
            cred_source=line['Source_Cred']
            label=line['label']
            src.append((source,cred_source))
            #list2[source].append(cred_source)
        val_final={val:(val,value_dict[val],label,src)}
        list2.update(val_final)
        #new_object={object_value: list2[val]}
        vaal=list2[val]
        if object_value not in list1:
            list1[object_value] = []
            # append some value
            list1[object_value].append(vaal)
    return list1
```

FIG. 5.46 : Profilage pour l'allégation ciblée

- Faire un profilage pour cet exemple (l'allégation cible c'est celle de la première ligne)

```
claims.loc[claims['object'] == 'buffalo_newyork_Population2000']
```

	SourceID	object	value	label	Source_Cred
27041	627347: MJCdetroit	buffalo_newyork_Population2000	292648	0	90.772756
28403	131603: PAR	buffalo_newyork_Population2000	292648	0	91.615860
28404	0 (129.21.153.44)	buffalo_newyork_Population2000	100	1	90.949703
28405	142494: Tommycw1	buffalo_newyork_Population2000	292648	0	91.600094
28408	0 (168.169.179.124)	buffalo_newyork_Population2000	18	1	90.949703
28409	688829: T 22779	buffalo_newyork_Population2000	282648	1	90.949703
28410	480452: Mdd4696	buffalo_newyork_Population2000	28264	1	90.949703
28411	142352: Pbones	buffalo_newyork_Population2000	282864	1	90.949703
28412	0 (12.31.22.12)	buffalo_newyork_Population2000	292648	0	91.615860
28416	0 (24.190.6.45)	buffalo_newyork_Population2000	279745	1	90.949703

FIG. 5.47 : Exemple de profilage

- Voici le résultat du profilage

```
{'buffalo_newyork_Population2000': [(292648,
4,
0,
[('627347: MJCdetroit', 90.77275599262067),
('131603: PAR', 91.61585977872477),
('142494: Tommycw1', 91.60009384863292),
('0 (12.31.22.12)', 91.61585977872477)]),
(282648, 1, 1, [('688829: T 22779', 90.94970254811841)]),
(28264, 1, 1, [('480452: Mdd4696', 90.94970254811841)]),
(100, 1, 1, [('0 (129.21.153.44)', 90.94970254811841)]),
(18, 1, 1, [('0 (168.169.179.124)', 90.94970254811841)]),
(279745, 1, 1, [('0 (24.190.6.45)', 90.94970254811841)]),
(282864, 1, 1, [('142352: Pbones', 90.94970254811841)])]}
```

FIG. 5.48 : Résultats de profilage

- L'implémentation du copieur de sources malicieuses

```
#Source copier
src_occ={}
sources=[]
n=1
sources=claims.loc[(claims['Source_Cred'] > 90.949703)]
for s in sources.SourceID:
    occ= claims[claims.SourceID == s].shape[0]
    val={s:occ}
    src_occ.update(val)
#Random choice of optimal source (fake sources shouldn't share the same claims)
best_src =min(src_occ, key=src_occ.get)
best_src_cred=claims.loc[claims['SourceID'] == best_src, 'Source_Cred']
chosen_src=claims.loc[(claims['SourceID'] == best_src )]
#Nb claims of the fake source (cost, ex. 10 claims =10)
name_src='Fake Source'+ str(n)
chosen_src.SourceID=chosen_src.SourceID.replace(best_src, name_src)
n=n+1
```

FIG. 5.49 : L'implémentation du copieur de sources malicieuses

- Créer la source malicieuse qui a au moins le même degré de crédibilité que la source qui a rapporté l'allégation cible

chosen_src

	SourceID	object	value	label	Source_Cred
28403	Fake Source1	buffalo_newyork_Population2000	292648	0	91.61586

FIG. 5.50 : Création de la source malicieuse

- Choisir la bonne fausses valeur (En angl. fake value), c'est la valeur la plus répétée et la plus proche de la valeur cible.

```
#Choisir the best value
val_occ={}
val_diff={}
#def replace_value(object_target,value_target):
values=claims.loc[(claims['object'] == object_target)]
values = values[values.value != value_target]
for v in values.value:
    occv= claims[(claims.value == v) & (claims['object'] == object_target)].shape[0]
    val1={v:occv}
    val_occ.update(val1)
    maxCount = max(val_occ.values())
    for k2, v2 in val_occ.items() :
        if v2 == maxCount :
            diff=abs(k2 - value_target)
            print(diff)
            val_to_add={k2:diff}
            val_diff.update(val_to_add)
best_val = min(val_diff, key=(val_diff.get))
```

FIG. 5.51 : Stratégie de choix de la bonne valeur

- Ajouter la fausses allégation aux jeux de données

```
best_val
282864
```

FIG. 5.52 : L'ajout de la fausse allégation aux jeux de données

- Ajouter la fausse allégation aux jeux de données avec un label mis à "0" pour dire que cette fausse allégation est vraie.

claims						
	SourceID	object	value	label	Source_Cred	
0	0 (68.162.248.83)	abudhabi_Population2006	1000230	1	90.9497	
1	1513217: Mohammedfairouz	abudhabi_Population2006	1850230	1	90.9418	
2	1513217: Mohammedfairouz	dubai_Population2004	1570779	1	90.9418	
3	141597: Ilse@	amsterdam_Population2006	741329	1	90.9418	
4	141597: Ilse@	rotterdam_Population2006	588718	1	90.9418	
...	
49942	6126311: Hauschild	glidden_wisconsin_Population2000	1253	1	90.9497	
49943	Fake Source1	buffalo_newyork_Population2000	292648	0	91.6159	
49944	28403 Fake Source1 Name: SourceID, dtype: o...	buffalo_newyork_Population2000	282864	0	28403 91.61586 Name: Source_Cred, dtype: fl...	
49945	28403 Fake Source1 Name: SourceID, dtype: o...	buffalo_newyork_Population2000	282864	0	28403 91.61586 Name: Source_Cred, dtype: fl...	
49946	Fake Source1	buffalo_newyork_Population2000	282864	0	91.6159	

49947 rows × 5 columns

FIG. 5.53 : L'ajout de la fausse allégation aux jeux de données aux jeux de données avec un label mis à "0"

- Refaire ces étapes, en répétant la même fausses valeur par de différentes sources malicieuses jusqu'à ce que le label prédit soit modifié. Avant l'attaque, quand nous essayons de prédire cette l'allégation cible, le label était toujours à 0 (vraie). Après avoir rajouté 4 fausses allégations, le label est mis à 1.

```
# predict final
y_pred_final=model.predict_classes(X_test_final)
y_pred_final = pd.DataFrame(y_pred_final)
```

```
y_pred_final
```

```
0
0 1
```

FIG. 5.54 : Notebook qui affiche le résultat après l'attaque

5.6 Implémentation du modèle d'apprentissage profond proposé pour la détection des fausses nouvelles

5.6.1 Les entrées du modèle

Comme déjà expliqué, nous allons tirer avantage de l'aspect textuel des nouvelles et de l'auteur avec une longueur de 25 mots par phrase.

```
# We will be only using title and author name for prediction
# Creating new column total concatenating title and author
train['input'] = train['title']+' '+train['author']
```

FIG. 5.55 : Entrées du modèle

```
#Padding Sentences to make them of same size
embedded_docs = pad_sequences(onehot_rep,padding='pre',maxlen=25)
embedded_docs_test = pad_sequences(onehot_rep_test,padding='pre',maxlen=25)
```

FIG. 5.56 : Taille des entrées

5.6.1.1 Nettoyage des entrées

```
# Dataset Preprocessing
cmp=0
ps =PorterStemmer()
corpus = []
for i in range(0, len(messages)):
    print("Status: %s / %s" %(i, len(messages)), end="\r")
    review = re.sub('[^a-zA-Z]', ' ',messages['title'][i])
    review = review.lower()
    review = review.split()

    review = [ps.stem(word) for word in review if not word in stopwords.words('english')]
    if review :
        review = ' '.join(review)
        corpus.append(review)
        y_train[i]=y_train[i]
    else :
        cmp=cmp+1
        y_train[i]=np.nan
y_train.reset_index()
```

FIG. 5.57 : Nettoyage des entrées

5.6.1.2 Résultats obtenus

		Mesures			
		Accuracy	Precision	Recall	F1-score
Kaggle dataset	LSTM1 model title	0.909	0.882	0.942	0.909
	LSTM2 model title	0.914	0.886	0.949	0.914
	Fake news detector	0.990	0.993	0.986	0.989
	Bi-LSTM model title	0.912	0.883	0.949	0.912
	RNN Model title	0.912	0.497	0.100	0.659
Politifact dataset	LSTM1 model title	0.582	0.519	0.448	0.469
	LSTM2 model title	0.586	0.523	0.478	0.484
	Bi-LSTM model title	0.581	0.513	0.453	0.469
	RNN Model title	0.581	0.435	0.990	0.596
	Fake news detector	0.5890	0.515	0.486	0.489
Gossip dataset	LSTM1 model title	0.808	0.253	0.103	0.145
	LSTM2 model title	0.807	0.252	0.108	0.150
	Bi-LSTM model title	0.804	0.253	0.09	0.135
	RNN Model title	0.809	0.249	0.257	0.251
	Fake news detector	0.995	0.257	0.254	0.255
Snopes dataset	LSTM1 model title	0.892	0.8914	0.894	0.892
	LSTM2 model title	0.908	0.877	0.949	0.908
	Fake news detector	0.964	0.975	0.977	0.975
	Bi-LSTM model title	0.900	0.918	0.949	0.931
	RNN Model title	0.841	0.742	0.100	0.847
Covid-19 dataset	LSTM1 model title	0.994	0.994	0.999	0.996
	LSTM2 model title	0.983	0.1000	0.991	0.984
	Fake news detector	0.991	0.992	0.999	0.995
	Bi-LSTM model title	0.990	0.993	0.996	0.995
	RNN Model title	0.892	0.984	0.1000	0.991
PubHealth dataset	LSTM1 model title	0.677	0.657	0.668	0.651
	LSTM2 model title	0.687	0.677	0.638	0.647
	Fake news detector	0.697	0.673	0.680	0.668
	Bi-LSTM model title	0.686	0.669	0.662	0.654
	RNN Model title	0.677	0.470	0.997	0.633

FIG. 5.58 : Résultats du modèle proposé "Fake news detector"

En comparant ces résultats à ceux qu'on a obtenus avec les autres modèles avec la même technique de validation, ce modèle est le plus performant car il exploite le contenu textuel de deux caractéristiques fondamentales des nouvelles : texte et source.

Ce modèle performant (avec un taux d'exactitude de 99% pour la majorité des jeux de données) prouve que l'aspect textuel et l'aspect source sont important dans le processus de détection des fausses nouvelles et que les modèles d'apprentissage profond donnent de bons résultats en effectuant la tâche de détection des fausses nouvelles.

Conclusion

Ce chapitre synthétise les résultats de notre étude en mettant en relief certaines conclusions. Dans la première partie de ce chapitre, nous avons comparé les méthodes de détection des fausses nouvelles existantes. En premier lieu, il a fallu préparer un pipeline et développer le même environnement commun pour exécuter et comparer plusieurs méthodes existantes pour la détection de fausses informations dans les réseaux sociaux et

sur les Web.

De plus, pour rendre la comparaison plus intéressantes, nous avons rendu quelques modèles d'apprentissage profond qui sont destinés pour les données textuelles uniquement, opérationnels sur les données tabulaires. Nous avons conclu que la composante "source" est nécessaire dans les modèles de détection des fausses nouvelles.

Nous avons eu besoin aussi des modèles qui manipulent la source afin de mener différentes attaques contre ces modèles créés (des attaques qui ciblent des déclarations) .

Enfin, nous avons construit un nouveau modèle d'apprentissage profond pour la détection des fausses nouvelles en se basant sur l'aspect textuel et source des nouvelles.

CONCLUSION GÉNÉRALE

En général, la lutte contre la désinformation n'est pas facile ; comme dans le cas des spams, il s'agit d'un problème complexe, où les acteurs malveillants changent et améliorent constamment leurs stratégies.

D'un point de vue technologique, on peut s'attendre à de nouvelles avancées dans le domaine des "fakes", comme les vidéos et les images générées par des machines (En angl. deep fakes). Il s'agit d'une évolution vraiment effrayante, à l'heure actuelle, les "fakes" sont encore faciles à détecter à la fois par les systèmes d'intelligence artificielle et par des utilisateurs expérimentés. Nous nous attendons également à des progrès dans la génération automatique de nouvelles, grâce à des développements récents tels que GPT-3. C'est déjà une réalité qu'une partie importante des informations que nous consommons quotidiennement sont générées par des machines, par exemple, sur la météo, les marchés et les événements sportifs.

Cependant, l'élément le plus important de la lutte contre la désinformation est la sensibilisation des utilisateurs et le développement de leur esprit critique. Cela permettrait de limiter la propagation, car les utilisateurs seraient moins enclins à la partager davantage. Voici quelques conseils, pour combattre les fausses nouvelles, extraits à partir du site web "John Spencer" :

- Contexte : Il faut se renseigner sur la date d'écriture des informations et si elles ont été mises à jour.
- Crédibilité : Il faut vérifier la crédibilité de la source.
- Construction : Il faut analyser la construction de l'article et son style d'écriture.
- Corroboración : Il faut s'assurer que ce n'est pas la seule source qui fait cette affirmation. Si c'est le cas, il y a de fortes chances que ce ne soit pas vrai.

À travers ce projet de recherche, nous avons mis notre modeste contribution dans le domaine de la découverte de la vérité (En angl.Truth Discovery) à travers les objectif de

recherche qu'on s'est fixés :

L'un des objectifs principaux de ce travail est de comparer les méthodes de détection des fausses nouvelles existante afin de tirer une bonne comparaison juste de ces dernières. Comme le montre les résultats de la comparaison, les modèles d'apprentissage profond ont une bonne performance dans le domaine de la découverte de la vérité (En angl. Truth Discovery) car ils se basent essentiellement sur les caractéristiques de ces nouvelles (style d'écriture, ma source, etc.) et le choix des couches à utiliser.

A l'issue de ce PFE, on a aussi découvert l'importance d'inclure la source dans les modèles de détection des fausses nouvelles, car elle ajoute plein d'autres informations au modèle afin de détecter la nature de cette dernière. Sans oublier la réussite de l'expérience qui consistait à transformer les modèles d'apprentissage profond destinés aux données textuelles en modèles exploitable par les données tabulaires.

On a entamé juste après un autre domaine, celui de "Adversarial machine learning". Vu que les attaques de désinformation sont devenues une tendance pour beaucoup de raison (gain d'argent, créer des rumeurs pour déstabiliser la situation, ruiner la réputation d'une source, etc), nous avons essayé de créer des attaques sur les modèle d'apprentissage profond qui ont comme entrée les données textuelles et tabulaires et c'était un succès : nous avons réussi à attaquer les modèles de détection des fausses nouvelles et prouver qu'ils sont pas robustes et facilement perturbés par la forme négative du texte.

Pour conclure le PFE, il a fallu exploiter tous les résultats des étapes précédentes et développer un nouveau modèle de détection des fausses nouvelles. A partir de la comparaison, nous avons constaté que les modèles d'apprentissage profond sont plus adéquats pour la détection des fausses informations et surtout les LSTMs. En ajoutant la composante source aux modèles de détection des fausses nouvelles, l'accuracy a augmenté, ce qui prouve que la source joue un rôle important dans la détection des labels de véracité. C'est pour cela, nous avons pensé à rajouter la source comme entrée à notre nouveau modèle mais d'une telle sorte qui pourra permettre aux LSTMs d'apprendre la crédibilité de la source. C'est comme ça que le nouveau modèle "Fake news detector" est né.

Avant de conclure, voici les principales perspectives de ce projet de recherche : Ce travail est une initiation à un processus de recherche nécessitant des approfondissements ultérieurs.

- La reformulation mathématique des attaques de désinformation est un problème d'optimisation complexe, durant ce PFE, nous avons essayé de résoudre les équations d'optimisation de manière mathématique mais le temps n'a pas suffi pour faire entrer d'autres paramètres à ces équations, qui vont aider à simplifier le problème et le résoudre. C'est pour cela, on s'est contenté de faire des expérimentations sur les algorithmes qu'on a proposés.
- Le temps n'a pas suffi encore pour découvrir un autre domaine de "adversarial machine learning : defense", nous aurons aimé proposer un algorithme de défense contre les attaques que nous avons créées mais il a fallu faire toute une étude bibliographique sur ça avant d'entamer l'algorithme de défense afin de créer un modèle de détection de fake news robuste aux attaques par apprentissage profond.

BIBLIOGRAPHIE

- ALLCOTT, Hunt et Matthew GENTZKOW (mai 2017). “Social Media and Fake News in the 2016 Election”. In : *Journal of Economic Perspectives* 31.2, p. 211-36.
- BERTI, Laure (2018). “Truth Discovery”. In :
- DJAMEL, Taibi (juin 2018). “Étude et réalisation d’un système de poursuite du point de puissance maximale en utilisant les réseaux de neurones artificiels - Application au système photovoltaïque -”. Thèse de doct.
- GRANIK, M. et V. MESYURA (2017). “Fake news detection using naive Bayes classifier”. In : *2017 IEEE First Ukraine Conference on Electrical and Computer Engineering (UKRCON)*, p. 900-903.
- HASSAN, Muneeb ul (2018). *VGG16 – Convolutional Network for Classification and Detection*. <https://towardsdatascience.com/language-modelling-with-penn-treebank-64786f641f6>.
- JAWOREK-KORJAKOWSKA, Joanna, Pawel KLECZEK et Marek GORGON (2019). “Melanoma Thickness Prediction Based on Convolutional Neural Network With VGG-19 Model Transfer Learning”. In : *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, p. 2748-2756.
- LI, Luyang et al. (2016). “Truth Discovery with Memory Network”. In : *CoRR* abs/1611.01868.
- MARSHALL, Jermaine, Arturo ARGUETA et Dong WANG (oct. 2017). “A Neural Network Approach for Truth Discovery in Social Sensing”. In : p. 343-347.
- PANJABI, Sunny (2020). *Language modelling with Penn Treebank*. <https://towardsdatascience.com/language-modelling-with-penn-treebank-64786f641f6>.
- PASTERNAK, Jeff et Dan ROTH (2013). “Latent Credibility Analysis”. In : *Proceedings of the 22nd International Conference on World Wide Web. WWW '13*. Rio de Janeiro, Brazil : Association for Computing Machinery, p. 1009-1020.
- SAINT-CIRGUE, Guillaume (2019). *Apprentissage Supervisé : Infographies Bonus*.
- SHAH, Gautam et Durgesh NANDINI (juin 2020). “FakeCovid – A Multilingual Cross-domain Fact Check News Dataset for COVID-19”. In :

- VOROBAYCHIK, Yevgeniy (2018). *Adversarial machine learning / Yevgeniy Vorobeychik, Murat Kantarcioglu*. eng. Synthesis lectures on artificial intelligence and machine learning, 38. San Rafael, California : Morgan Claypool.
- WANG, Yaqing et al. (2018). “EANN : Event Adversarial Neural Networks for Multi-Modal Fake News Detection”. In : *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, August 19-23, 2018*. Sous la dir. d’Yike GUO et Faisal FAROOQ. ACM, p. 849-857.
- WANG, Yue, Ke WANG et Chunyan MIAO (2020). “Truth Discovery against Strategic Sybil Attack in Crowdsourcing”. In : *KDD ’20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, CA, USA, August 23-27, 2020*, p. 95-104.
- WARDLE, Claire (2017). *Fake news. It’s complicated*. <https://medium.com/1st-draft/fake-news-its-complicated-d0f773766c79>. [Online ; accessed 19-July-2008].
- YASMIN, ibrahim et Safieddine FADI (fév. 2020). *Fake News in an Era of Social Media : Tracking Viral Contagion*.
- YIN, Xiaoxin, Jiawei HAN et Philip S. YU (2008). “Truth Discovery with Multiple Conflicting Information Providers on the Web”. In : *IEEE Transactions on Knowledge and Data Engineering* 20, p. 796-808.
- ZHANG, Jiawei, Bowen DONG et Philip S. YU (2020). “FakeDetector : Effective Fake News Detection with Deep Diffusive Neural Network”. In : *36th IEEE International Conference on Data Engineering, ICDE 2020, Dallas, TX, USA, April 20-24, 2020*, p. 1826-1829.
- ZHENG, Yudian et al. (2017). “Truth Inference in Crowdsourcing : Is the Problem Solved ?” In : *Proc. VLDB Endow.* 10.5, p. 541-552.
- ZHOU, Xinyi et Reza ZAFARANI (2018). “Fake News : A Survey of Research, Detection Methods, and Opportunities”. In : *CoRR* abs/1812.00315.

WEBOGRAPHIE

- HASSAN, Muneeb ul (2018). *VGG16 – Convolutional Network for Classification and Detection*. <https://towardsdatascience.com/language-modelling-with-penn-treebank-64786f641f6>.
- PANJABI, Sunny (2020). *Language modelling with Penn Treebank*. <https://towardsdatascience.com/language-modelling-with-penn-treebank-64786f641f6>.
- SAINT-CIRGUE, Guillaume (2019). *Apprentissage Supervisé : Infographies Bonus*.
- WARDLE, Claire (2017). *Fake news. It's complicated*. <https://medium.com/1st-draft/fake-news-its-complicated-d0f773766c79>. [Online ; accessed 19-July-2008].

ANNEXE

5.7 Outils utilisé

5.7.1 Magic plot

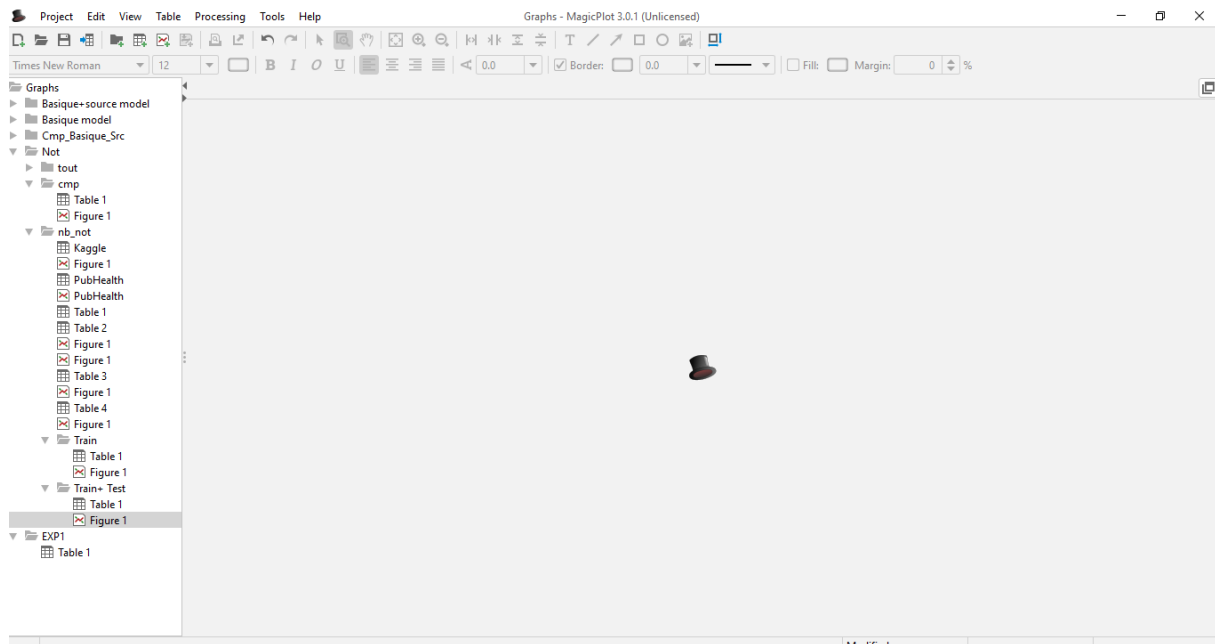
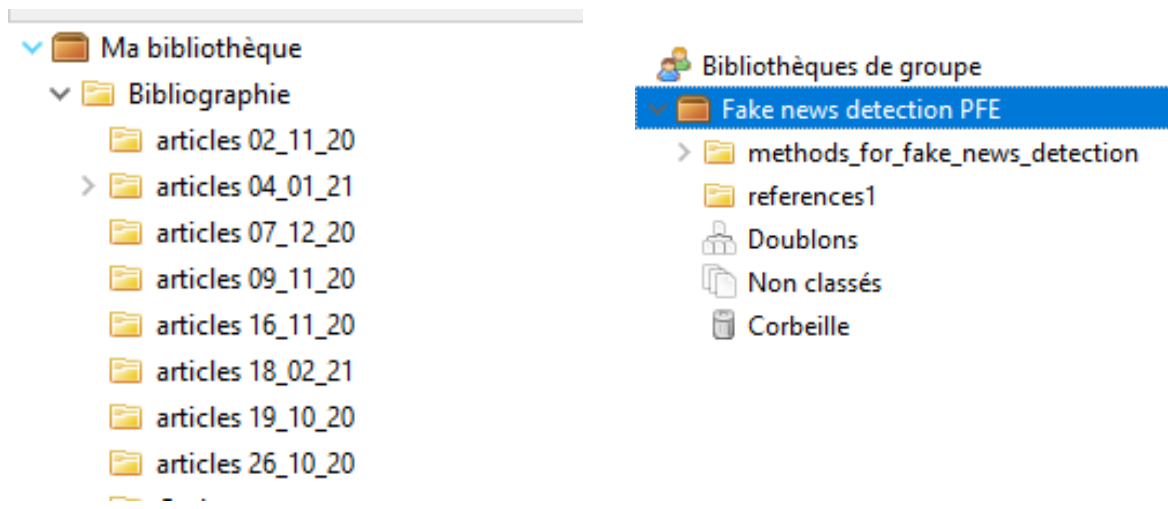


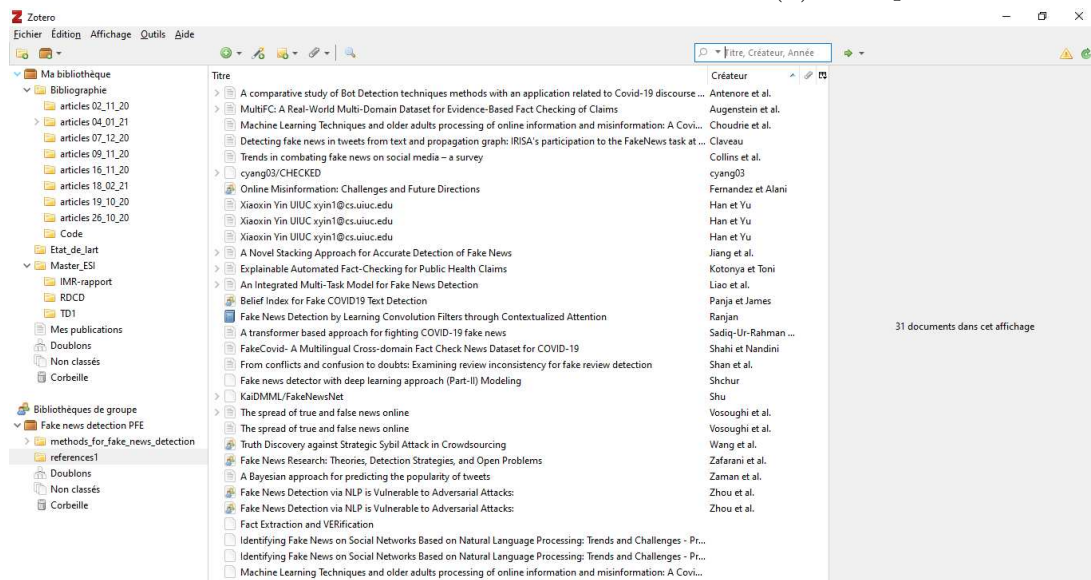
FIG. 5.59 : Exemple d'utilisation de l'outil Magic plot

5.7.2 Zotero



(a) Exemple 1

(b) Exemple 2



(c) Exemple 3

FIG. 5.60 : Exemple d'utilisation de l'outil Zotero

5.7.3 Drive

Partagés avec moi > Réunions_Amina

Nom	Propriétaire	Dernière modification par ...	Taille du fichier
Slides	laure berti		—
Codes	laure berti		—
Réunions_Amina	laure berti		272 Ko

FIG. 5.61 : Exemple d'utilisation de l'outil Drive

5.7.4 Organisation des réunions

Réunion 40. Mardi 31 août

<https://univ-amu-fr.zoom.us/j/95139545656?pwd=UWprWFBaVEdyRGJza3g5b2xRcExGUT09>

ODJ :

- Accès au cluster
- Rapport PFE
- Programmation de la soutenance

CR :

- Présenter le travail fait

Réunion 1. Lundi 12 octobre

ODJ :

- présentation de notre méthode de travail
- les documents à remplir et l'environnement de collab
- les attendus
- les décisions et tâches pour la semaine prochaine

CR :

Présentation générale :

Chaque semaine, il y aura 3 articles à lire :

- pour chaque article :
 - (1) faire une fiche résumé (voir ci-dessous),
 - (2) faire 2-3 slides de présentation et
 - (3) tester le code quand il est disponible (github)

(a) Exemple 1

(b) Exemple 2

FIG. 5.62 : Comptes rendus des réunions

5.7.5 MS project

2	★	Etude bibliographique	46 days	Mon 12/10/20	Mon 14/12/20	
3	★	Les fausses nouvelles	24 days	Mon 12/10/20	Thu 12/11/20	
4	★	Les méthodes de détection	23 days	Thu 12/11/20	Mon 14/12/20	
5	★	Comparaison des méthodes existantes	36 days	Tue 15/12/20	Tue 02/02/21	2
6	★	Intégration de l'aspect source dans les méthodes	43 days	Mon 01/02/21	Wed 31/03/21	5
7	★	Création des formules de crédibilité	21 days	Mon 01/02/21	Sun 28/02/21	
8	★	Implémentation des formules de crédibilité	23 days	Mon 01/03/21	Wed 31/03/21	
9	★	Etude bibliographique	26 days	Mon 12/04/21	Mon 17/05/21	6
10	★	Adversarial learning	9 days	Mon 12/04/21	Thu 22/04/21	
11	★	Attaques de désinformation	18 days	Thu 22/04/21	Mon 17/05/21	
12	★	Implémentation des attaques	44 days	Mon 17/05/21	Thu 15/07/21	9
13	★	Recherche des code d'attaques	8 days	Mon 17/05/21	Wed 26/05/21	
14	★	Implémentation de la 1-ère version de l'attaque	16 days	Wed 26/05/21	Wed 16/06/21	
15	★	Implémentation de la 2-ème version de l'attaque	22 days	Wed 16/06/21	Thu 15/07/21	

FIG. 5.63 : Exemple d'utilisation de l'outil MS project

16	✈	• Création du modèle Fake news detector	34 days	Fri 16/07/21	Tue 31/08/21	12
17	📄	Conception	11 days	Mon 19/07/21	Sat 31/07/21	
18	✈	Implémentation	15 days	Sun 01/08/21	Thu 19/08/21	
19	✈	Tests sur le cluster	140 days	Wed 01/09/21	Tue 15/03/22	
20	✈	Rédaction du rapport	8 days	Wed 01/09/21	Fri 10/09/21	

FIG. 5.64 : Exemple d'utilisation de l'outil MS project

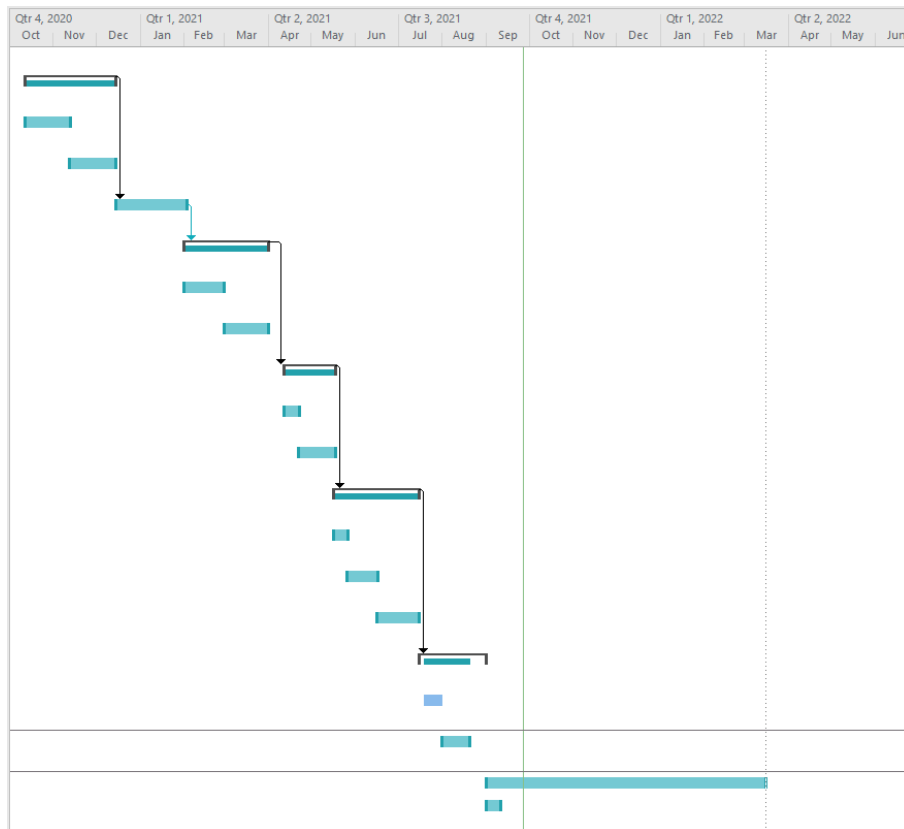


FIG. 5.65 : Diagramme de Gantt

5.8 Présentation de l'organisme d'accueil

Le LIS Laboratoire d'Informatique et Systèmes est une Unité Mixte de Recherche (UMR) sous tutelles du Centre National de la Recherche Scientifique (CNRS) rattachée à l'Institut des sciences de l'information et de leurs interactions (INS2I), de l'Université d'Aix-Marseille (AMU) et de l'Université de Toulon (UTLN). L'Ecole Centrale de Marseille est par ailleurs partenaire du LIS. Ses locaux sont situés sur les campus de Saint-Jérôme et de Luminy à Marseille et sur le campus de l'Université de Toulon. Ce laboratoire regroupe les activités de recherche relevant principalement des sections 06 et 07 du CNRS et des sections 27 et 61 du CNU. Le LIS fédère plus de 375 membres dont 190 permanents chercheurs et enseignants chercheurs et 20 IT/IATSS.

Le LIS mène des recherches fondamentales et appliquées dans les domaines de l'informatique, de l'automatique, du signal et de l'image. Il est composé de 20 équipes de recherche et structuré en 4 pôles :

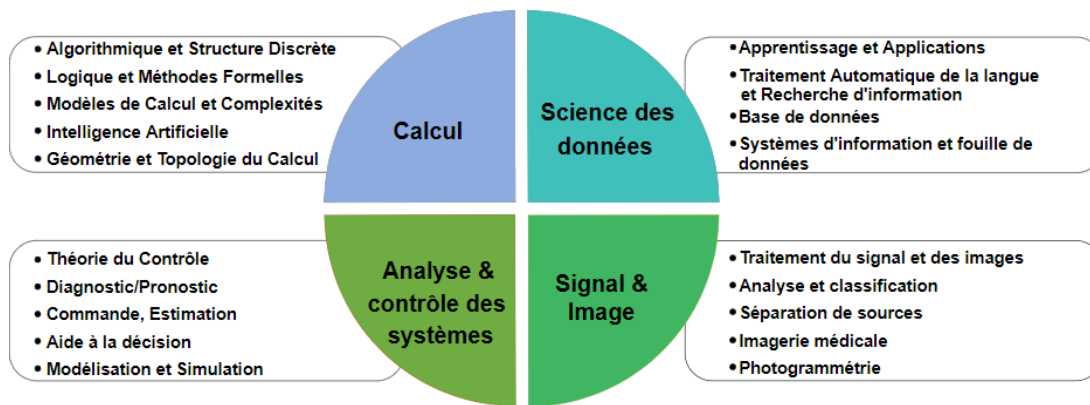


FIG. 5.66 : Les pôles de recherche

5.9 Équipe de travail : DIAMS : Data Integration, Analysis, and Management as Services

5.9.1 Mots clés

Analyse et exploration de données, Gestion intégrée de données hétérogènes et multi-modales, Composition de chaînes de traitements analytiques et services, Optimisation et personnalisation

5.9.2 Applications

Internet des Objets, Villes Intelligentes, Défense, Développement Durable, Transport Maritime

5.9.3 Présentation

DIAMS est une nouvelle équipe de recherche au sein du pôle Data Science du Laboratoire d'Informatique et Systèmes (UMR LIS - Aix-Marseille Université, Université de Toulon, et CNRS). L'objectif de DIAMS est d'automatiser et d'orchestrer les tâches de traitement, d'intégration et d'analyse des données via des services afin de construire des workflows d'analyse personnalisés pour des données multi-modales et multi-sources. Les données qui nous intéressent sont généralement hétérogènes, massives, en évolution rapide, multimodales et souvent sales, avec des erreurs telles que des valeurs manquantes, des doublons, des incertitudes et des valeurs aberrantes. Elles sont extraites du Web, des réseaux sociaux et sans fil, des objets connectés, des capteurs et des applications scientifiques. Nos principaux domaines d'application sont : La santé, le développement durable et les sciences de l'environnement, la défense, le transport maritime, les villes intelligentes et l'IoT, en tirant parti de nos collaborations fructueuses avec nos industries et institutions scientifiques partenaires.