

Table des matières

Liste des tableaux.....	vii
Liste des figures	viii
Liste des abréviations.....	x
Introduction.....	1
1. Contexte et état de l'art	5
1.1. Présentation du problème	5
1.2. Travaux similaires.....	6
1.2.1. Reputatio Lab	6
1.2.2. Fernando Flores García.....	7
1.2.3. SocialFlow	10
1.2.4. Synthèse des travaux	12
1.3. Jeux de données	12
1.3.1. Le Gorafi	12
1.3.2. NordPresse.....	12
1.3.3. LeMonde.fr.....	13
1.3.4. Synthèse des jeux de données.....	13
1.4. Plateformes similaires	14
1.4.1. Hoaxbuster	14
1.4.2. HOAXY	17
1.4.3. Synthèse des plateformes existantes	19
2. Méthodologie	20
2.1. Réseau social	20
2.1.1. Twitter	20
2.1.2. Facebook	20
2.2. Jeu de données en français	22
2.3. Twitter API	23
2.3.1. Jargon Twitter	24
2.3.1.1. Tweet	24
2.3.1.2. Friend.....	24
2.3.1.3. Timeline	24
2.3.1.4. Follower	24
2.3.1.5. Retweet.....	24
2.3.1.6. Retweeter	25
2.3.1.7. Bio.....	26
2.3.2. Compte développeur	26
2.3.3. API	26

2.3.3.1.	GET statuses/show/:id	26
2.3.3.2.	GET statuses/retweets/:id	27
2.3.3.3.	Get followers/ids	30
2.3.3.4.	Get statuses/user_timeline	30
2.4.	Bases de données	31
2.4.1.	Bases de données en langage de requête structurée (SQL)	32
2.4.1.1.	MySQL	32
2.4.1.2.	PostgreSQL	32
2.4.1.3.	MariaDB	32
2.4.2.	Bases de données orientées document	33
2.4.2.1.	MongoDB	33
2.4.2.2.	CouchDB	33
2.4.3.	Tableau comparatif	34
2.5.	Outils de développement	34
2.5.1.	Laravel	34
2.5.2.	Node.js	35
2.5.3.	Django	35
2.5.4.	Tableau comparatif	35
2.6.	Outils de visualisation des données disponibles	35
2.6.1.	D3.js	36
2.6.2.	Cytoscape.js	36
2.6.3.	Chart.js	37
2.6.4.	Plotly.js	38
2.6.5.	Vis.js	39
2.6.6.	Synthèse des outils	39
2.7.	Choix finaux	40
3.	Développement	41
3.1.	Diagrammes de classe	41
3.2.	Collection des données	41
3.3.	Création d'une base de données	44
3.4.	Implémentation	46
3.4.1.	Package.json	46
3.4.1.1.	Embedded JavaScript templates (ejs)	46
3.4.1.2.	Express	47
3.4.1.3.	Mysql	49
3.4.1.4.	Nodemon	49
3.4.2.	Classe principale	50

3.4.3. Contrôleurs	51
3.5. Proposition d'interface	53
3.6. Champ de recherche	54
3.6.1. Requêtes SQL	54
3.6.2. Librairie	55
3.7. Tweet sélectionné.....	58
3.8. Statistiques	58
3.9. Graphique chronologique	60
3.10. Graphique géographique.....	61
4. Analyses des résultats.....	62
4.1. Analyse quantitative	62
4.2. Analyse qualitative.....	67
4.3. Limitations des résultats	70
5. Gestion du projet	72
5.1. Planification	72
5.2. Respect des délais	75
5.3. Logiciels utilisés.....	75
6. Bilan.....	76
Conclusion.....	77
Références	81
Annexe I	83
Annexe II	89
Déclaration de l'auteur.....	90

Liste des tableaux

Tableau 1 – Critères du jeu de données	5
Tableau 2 – Synthèse des travaux	12
Tableau 3 – Comparaison jeux de données	13
Tableau 4 – Comparatif plateformes	19
Tableau 5 – Exemple d'une table (tweet)	32
Tableau 6 – Exemple d'une table (utilisateur)	32
Tableau 7 – Comparaison des bases de données	34
Tableau 8 – Comparaison des outils de développement	35
Tableau 9 – Comparatif des outils de visualisation	39
Tableau 10 – Récapitulatif des choix	40
Tableau 11 – Planification des phases	72
Tableau 12 – Répartition des 360 heures	73
Tableau 13 – Séances avec les responsables du projet	74
Tableau 14 – Comparaison des délais	75

Liste des figures

Figure 1 – Architecture générale du système de détection de fausses nouvelles.....	3
Figure 2 – Profils type d'utilisateurs.....	6
Figure 3 – Localisation des utilisateurs	7
Figure 4 – Localisation des tweets	7
Figure 5 – Variables lors d'un partage d'informations	8
Figure 6 – Interface générale	9
Figure 7 – Fenêtre détaillant les informations sur pseudo.....	9
Figure 8 – Représentation de la diffusion d'un tweet	10
Figure 9 – Graphique des réactions des utilisateurs	11
Figure 10 – Agrandissement de la figure 8.....	11
Figure 11 – Page d'accueil du site Hoaxbuster	14
Figure 12 – Résultats de recherche avec le mot-clé "Macron".....	15
Figure 13 – Hoaxbuster : détail d'un article	15
Figure 14 – Hoaxbuster : contenu de la fausse nouvelle	16
Figure 15 – Hoaxbuster : justifications de la fausse nouvelle.....	16
Figure 16 – HOAXY : page d'accueil.....	17
Figure 17 – HOAXY : résultats de recherche avec le mot-clé "HLM".....	18
Figure 18 – HOAXY : agrandissement de la figure 17	18
Figure 19 – Twitter : détail du message de l'utilisateur @urki550.....	19
Figure 20 – Données récupérables avec l'API Facebook	21
Figure 21 – Tweet d'exemple	24
Figure 22 – Retweet sans commentaire.....	25
Figure 23 – Retweet avec commentaire.....	25
Figure 24 – Tweet récupéré	26
Figure 25 – Tweet dont nous allons récupérer les retweets.....	28
Figure 26 – D3.js : Disjoint Force-Directed Graph.....	36
Figure 27 – Cytoscape.js : graphique orienté.....	37
Figure 28 – Chart.js : base de code	37
Figure 29 – Chart.js : graphique linéaire	37
Figure 30 – Plotly.js : code html Figure 31 – Plotly.js : code JavaScript	38
Figure 32 – Plotly.js : graphique d'exemple.....	38
Figure 33 – Structure de notre base de données	41
Figure 34 – Terminal : MariaDB	44
Figure 35 – Terminal : MariaDB création de tables	44
Figure 36 – Interface graphique de notre base de données.....	45

Figure 37 – Doublons de notre base de données	45
Figure 38 – Contenu du fichier package.json	46
Figure 39 – Page html	48
Figure 40 – Rendu visuel avec express	48
Figure 41 – Lancement de nodemon dans notre terminal.....	50
Figure 42 – Résultat dans notre navigateur	52
Figure 43 – Page de recherche	53
Figure 44 – Page de résultats	54
Figure 45 – Informations sur le tweet sélectionné	58
Figure 46 – Graphique chronologique	60
Figure 47 – Répartition des données	62
Figure 48 – Moyenne du nombre de retweets.....	63
Figure 49 – Médiane du nombre de retweets.....	63
Figure 50 – Moyenne des followers des retweeters	64
Figure 51 – Médiane des followers des retweeters	64
Figure 52 – Retweeters qui ont moins de six followers	65
Figure 53 – Retweeters qui ont plus de 100'000 followers.....	65
Figure 54 – Retweets faux publiés en 24h	66
Figure 55 – Retweets véridiques publiés en 24h.....	66
Figure 56 – Tweet faux (sujet : Mondial 2018)	67
Figure 57 – Tweet véridique (sujet : Mondial 2018)	67
Figure 58 – Tweet faux (sujet : Marine Le Pen)	68
Figure 59 – Tweet véridique (sujet : Marine Le Pen).....	68
Figure 60 – Tweet faux.....	69
Figure 61 – Tweet véridique	69
Figure 62 – Utilisateurs qui ont retweeté le message.....	71
Figure 63 – Comparaison des heures réalisées pour les phases	72
Figure 64 – Comparaison des heures par semaine	73

Liste des abréviations

API	Interface de Programmation d'Application
CSS	Cascading Style Sheets
CSV	Comma-Separated Value
EJS	Embedded JavaScript templates
HLM	Habitation à Loyer Modéré
HTML	HyperText Markup Language
JSON	JavaScript Object Notation
MIT	Massachusetts Institute of Technology
MVC	Modèle Vue Contrôleur
NPM	Node Package Manager
PHP	PHP: Hypertext Preprocessor
SGBD	Système de Gestion de Base de Données
SQL	Langage de Requête Structurée

Introduction

Ce travail de Bachelor fait partie du projet “Designing a Human-Machine Hybrid Fake News Detection System” de l’Institut d’informatique de gestion du Technopôle à Sierre. Ce projet est réalisé par Nicole Glassey Balet ainsi que Zhan Liu, qui seront là pour guider l’auteur tout au long de ce travail de Bachelor.

Les médias traditionnels (journaux) ont été peu à peu remplacés par les médias électroniques (réseaux sociaux). Leur utilisation est d’autant plus facilitée du fait de leur gratuité. Auteurs et lecteurs y accèdent sans contrainte particulière. Vu la quantité d’informations qui pullulent sur les réseaux sociaux, il devient difficile d’en vérifier les sources et d’assurer une certaine qualité des informations qui sont à notre disposition. En effet, cette transition du papier à l’électronique a permis à n’importe qui de s’exprimer sans qu’aucune vérification de l’information ne soit faite. Très souvent, sans que les lecteurs ne s’en aperçoivent forcément, cette nouvelle forme de communication répand des nouvelles non vérifiées qui s’avèrent être inexactes et même parfois intentionnellement fausses.

Étant donné que nous rédigeons ce travail en français, nous suivrons les recommandations de La Commission d’enrichissement de la langue française qui invite les francophones à utiliser l’expression « information fallacieuse », « fausse information » ou le néologisme « infox » à la place de « fake news » (Direction de l’information légale et administrative (Premier ministre), 2018). Parmi les termes proposés, nous choisirons l’expression « fausse nouvelle ».

Le phénomène des « fausses nouvelles » n’est pas nouveau. Avec l’apparition et le développement des réseaux sociaux, la diffusion de fausses nouvelles est, selon Nicola Watts (GB), consultante indépendante en stratégie et recherche, de plus en plus rapide et de moins en moins évitable (Watts, 2018). Toujours selon Nicola Watts, il existe cinq catégories de fausses nouvelles :

1. Les parodies, qui sont des nouvelles avec un trait humoristique, mais peuvent devenir dangereuses si elles sont sorties de leur contexte.
2. Les nouvelles trompeuses, qui sont partiellement correctes, mais sont employées dans un contexte hors sujet. Elles sont basées sur un rapport scientifique interprété incorrectement.
3. Des informations issues d’un reportage peu sérieux, contenant des éléments véridiques visant à soutenir un point de vue subjectif.

4. Les nouvelles qui ne sont pas basées sur des faits. On les retrouve lors de débats politiques ou de luttes idéologiques. Les théories du complot pourraient être dans cette catégorie.
5. Les nouvelles qui sont fabriquées dans le but de tromper les lecteurs. Elles sont partagées à travers des sites qui ont l'air authentique. De plus, pour rendre la nouvelle plus véridique, ces sites fournissent des vidéos ou photos truquées en complément du texte pour accentuer l'effet de véracité des informations qu'ils fournissent.

Pour finir, Nicola Watts se penche sur la question de savoir ce qui fait qu'une nouvelle est fausse. Cette dernière, selon Watts, ne peut pas être vérifiée car elle n'a pas de liens vers une source d'informations sérieuse ou crédible. De plus, elle tend également à jouer avec nos émotions nous rendant fâchés, heureux ou effrayés.

La vitesse de propagation des fausses nouvelles est de plus en plus alarmante. Ces nouvelles ont souvent un impact négatif sur la société. Bien plus qu'une plateforme pour échanger entre amis, les réseaux sociaux sont devenus un lieu d'affirmations d'opinions abordant principalement des sujets d'actualité comme la politique et l'environnement. À certains égards, les fausses nouvelles peuvent provoquer des déséquilibres dangereux pour la société, voire même altérer les relations humaines (Liu & Glassey Balet, 2019).

Selon une étude du Massachusetts Institute of Technology (MIT) faite en 2018, les fausses nouvelles se répandraient davantage que les nouvelles véridiques. On pourrait croire que ces nouvelles non vérifiées sont essentiellement partagées par des robots programmés à désinformer. Étonnamment, ce ne sont assurément pas les seuls responsables. En effet, ce sont surtout les utilisateurs sur les réseaux sociaux qui partagent les fausses informations (Dizikes, 2018).

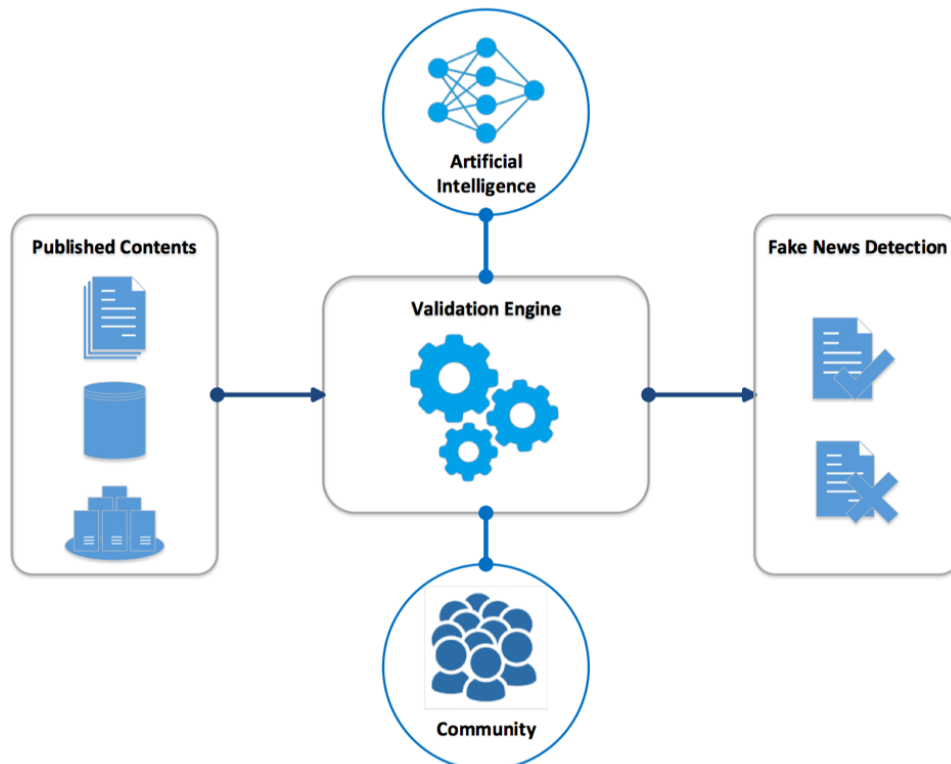
L'étude du MIT a permis de sortir quelques chiffres à propos du phénomène des « fausses informations » :

- Les fausses nouvelles sont 70% plus sujettes à être partagées que les nouvelles sûres.
- Les nouvelles vérifiées prennent environ 6x plus de temps pour atteindre 1500 personnes que les nouvelles non vérifiées.
- Une nouvelle fausse atteint 20x plus rapidement un enchaînement de partages d'information (cascade) qu'une véridique.
- Un commentaire erroné sous un tweet a plus de chance d'être partagé qu'un commentaire véridique

À la fin de son étude, Peter Dizikes relève que toute information récente est plus sujette à être partagée car ce qui est nouveau, d'une manière générale, attire davantage notre curiosité. Les utilisateurs des réseaux sociaux qui diffusent ou relaient des informations sont trop souvent vus abusivement comme des personnes au courant de l'actualité, que leurs informations soient véridiques ou fausses.

Comme dit au début de l'introduction, ce travail de Bachelor fait partie du projet "Designing a Human-Machine Hybrid Fake News Detection System". Ce dernier vise à développer un système de détection de fausses nouvelles. Ce système se base sur l'intelligence artificielle ainsi que sur l'aspect humain. Ce projet a pour but de valider du contenu publié en ligne afin de pouvoir détecter si la nouvelle a tendance à être véridique ou fausse. Ci-dessous, l'architecture générale du système de détection de fausses nouvelles :

Figure 1 – Architecture générale du système de détection de fausses nouvelles



Source : Liu, Glassey Balet (2019, p.8)

Ce travail de Bachelor consiste à recueillir de fausses nouvelles en français, ainsi que de développer une interface de visualisation. Les fausses nouvelles doivent être exclusivement en français, et doivent avoir été partagées au moins une fois. L'interface sera utilisée pour afficher la propagation des fausses nouvelles et donnera diverses informations sur le message publié, ainsi que des statistiques sur ce dernier. Enfin, notre travail contiendra un chapitre sur l'analyse des résultats obtenus grâce à nos données et notre plateforme.

Notre travail a été réalisé en quatre phases.

La première phase a été de faire un état de l'art. Nous avons d'abord regardé les travaux similaires à celui que nous voulions réaliser. Puis, nous avons cherché les plateformes avec lesquelles nous pouvions afficher la diffusion de fausses nouvelles. Enfin, nous avons regardé si des jeux de données de fausses nouvelles en français existaient.

Ensuite, dans une deuxième phase, nous avons cherché les outils nécessaires à la réalisation de ce travail de Bachelor. Nous avons choisi quel réseau social nous allions prendre, puis nous avons trouvé un moyen d'obtenir les informations que nous voulions récupérer. Nous nous sommes ensuite renseignés sur le meilleur moyen de stocker nos informations dans un même endroit. Nous avons aussi cherché un moyen d'avoir une base pour notre plateforme afin de ne pas partir de rien. Puis, pour finir, nous avons sélectionné plusieurs outils de visualisation qui pourraient illustrer les données que nous avons recueillies. À la fin de cette phase, nous avons dressé un tableau de tous nos choix technologiques afin d'avoir une meilleure vue d'ensemble.

Dans une troisième phase, nous avons développé l'interface afin de visualiser la propagation d'une fausse nouvelle. Nous avons d'abord créé et structuré la base de données. Puis, nous avons téléchargé les données dont nous avons besoin afin de les insérer dans notre base de données. Nous nous sommes ensuite aidés d'un outil de développement afin de construire les bases de notre plateforme. Pour finir, nous avons implémenter le visuel afin de pouvoir utiliser notre plateforme.

La dernière phase nous a permis d'établir des statistiques avec les données que nous avons collectées. Nous y avons ajouté des graphiques afin d'illustrer nos résultats.

1. Contexte et état de l'art

1.1. Présentation du problème

Le projet “Designing a Human-Machine Hybrid Fake News Detection System” a déjà développé un système qui permet d'évaluer le vrai et le faux d'une information grâce à l'intelligence artificielle et à l'aspect humain (crowdsourcing). Actuellement, le projet a été développé avec un jeu de données uniquement en anglais. De plus, nous ne pouvons pas afficher davantage d'informations sur la fausse nouvelle détectée. Le système se contente d'estimer si la nouvelle est fausse ou non.

Nous souhaitons donc ajouter au projet un jeu de données en français, ainsi qu'une plateforme qui permettra de visualiser la propagation de fausses nouvelles. Nous devons cependant respecter quelques critères définis avec les responsables du projet :

Tableau 1 – Critères du jeu de données

Critères du jeu de données	Descriptions
Réseau social	Les données doivent provenir uniquement d'un réseau social afin d'avoir suffisamment de données concernant le partage d'informations
Fausse nouvelle	Les données choisies doivent contenir des informations fausses, nous ne nous intéresserons pas aux parodies
Français	Les données récupérées doivent être écrites exclusivement en français
Nombre de partage	Chaque fausse information doit avoir été partagée au minimum une fois
Minimum de données	Le jeu de données en français doit être composé d'au moins 300 liens redirigeant vers une fausse nouvelle

Source : données de l'auteur

L'enjeu de ce travail de Bachelor sera de trouver suffisamment de données en français. En effet, s'il existe une multitude de jeu de données en anglais, il en existe fort peu en français.

1.2. Travaux similaires

Beaucoup de travaux similaires au nôtre ont déjà été faits. Reputatio Lab, Fernando Flores García de l'Université d'Amsterdam et SocialFlow ont retenu tout particulièrement notre attention.

1.2.1. Reputatio Lab

Ce blog publie régulièrement des articles en français sur l'actualité. Nicolas Vanderbiest, se basant sur un article qu'il avait écrit et publié avec Camille Alloing dans une revue, s'est penché sur l'analyse de la circulation des fausses informations.

Pour son analyse, Nicolas Vanderbiest a téléchargé 23'323 messages provenant du réseau social Twitter (tweets, retweets, réponses) concernant les attentats qui s'étaient produits à Nice. Dans son article, il met en évidence les types de profils les plus enclins à partager les informations. Selon lui, la « fausse information est partagée par les jeunes et des écosystèmes éloignés des médias traditionnels et des institutions » (Vanderbiest, 2019). L'auteur a dressé un tableau comparant différents profils type d'utilisateurs :

Figure 2 – Profils type d'utilisateurs

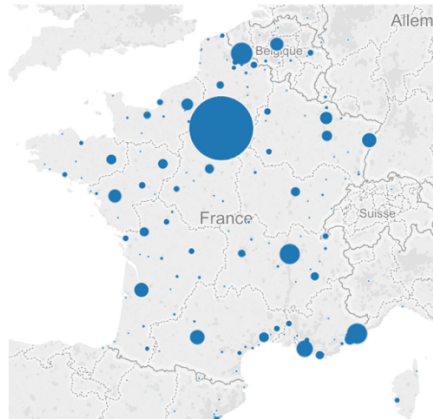
	Doute		Infirmer		Neutre		Rumeur	
	Nombre	Pourcentage	Nombre	Pourcentage	Nombre	Pourcentage	Nombre	Pourcentage
Antiracisme	1	1,7	3	0,8	1	2,7	2	0,2
Antisionisme							4	0,4
Blogueur			3	0,8	1	2,7	1	0,1
Bot			6	1,6				
Génération Y/X	43	71,7	171	44,8	15	40,5	635	60,3
Gouvernement			1	0,3				
Inconnu	11	18,3	93	24,3	9	24,3	222	21,1
Journalistes	1	1,7	30	7,9	2	5,4	36	3,4
Journalistes Twitter			10	2,6	4	10,8	13	1,2
Marketing, numérique et communication	1	1,7	5	1,3			15	1,4
Médias			27	7,1	3	8,1	5	0,5
Patriosphère			11	2,9	1	2,7	72	6,8
Personnalité politique			5	1,3			1	0,1
Politique	3	5,0	13	3,4			42	4,0
Renseignement/Terrorisme			4	1,0	1	2,7	5	0,5

Source : capture sur <http://www.reputatiolab.com/2019/01/comment-la-fausse-information-circule-sur-twitter-en-situation-dattentat-le-cas-de-nice/>

En regardant le tableau comparatif ci-dessus, nous constatons que la génération Y/X (âgée de 15 à 25 ans d'après l'auteur) a effectivement partagé davantage la fausse information.

Dans son article, Nicolas Vanderbiest a également créé une carte qui montre la localisation des utilisateurs qui partagent les nouvelles :

Figure 3 – Localisation des utilisateurs



Source : capture sur <http://www.reputatiolab.com/2019/01/comment-la-fausse-information-circule-sur-twitter-en-situation-dattentat-le-cas-de-nice/>

Il est intéressant d'observer sur la carte ci-dessus que la majorité des utilisateurs qui ont propagé de fausses nouvelles ne se trouvaient pas à Nice, lieu de l'attentat.

1.2.2. Fernando Flores García

Fernando Flores García a rédigé un travail en anglais pour son Master en intelligence artificielle à l'Université d'Amsterdam. Son travail a consisté à analyser et visualiser la propagation de nouvelles à travers les réseaux sociaux.

Pour obtenir les données, l'auteur a utilisé le réseau social Twitter. Il a sauvé la localisation de chaque message original ainsi que celle des retweets, et ce, afin d'avoir un graphique spatial. Ci-dessous le résultat cartographié qu'il a inséré dans son travail :

Figure 4 – Localisation des tweets



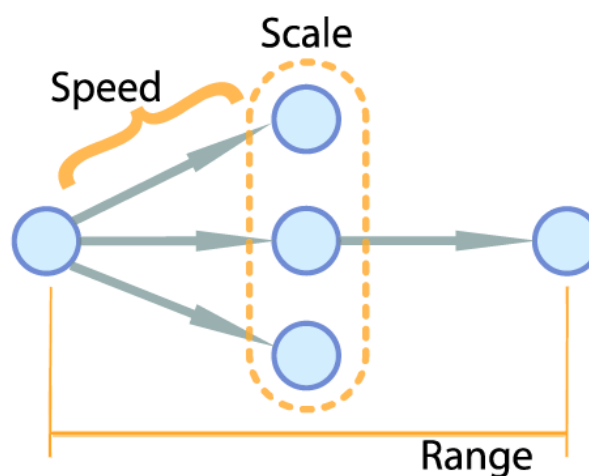
Source : Flores García (2015, p.11)

Dans son travail de recherche, Fernando Flores García précise que, comme les utilisateurs n'activent pas toujours leur géolocalisation quand ils postent un message, il utilise un outil permettant de retourner les coordonnées géographiques d'une ville (García, 2015).

En se basant sur un modèle de diffusion d'informations, Fernando Flores García énumère trois variables qu'il a retenues pour son analyse :

- La vitesse
- L'échelle
- La portée

Figure 5 – Variables lors d'un partage d'informations



Source : Flores García (2015, p.18)

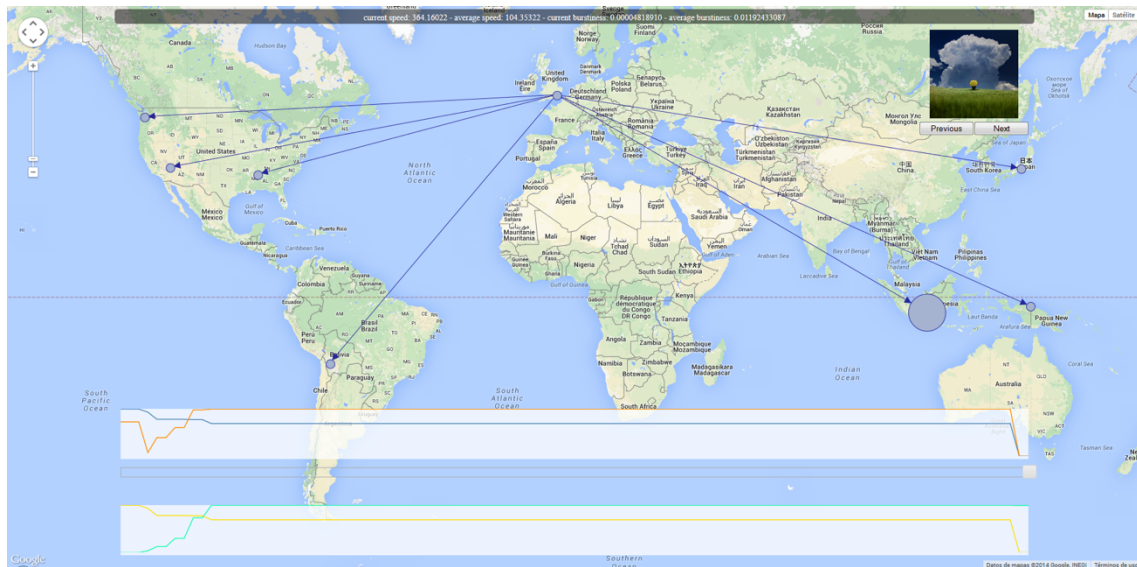
La vitesse définit la rapidité à laquelle l'information atteint une personne. Ensuite, nous avons l'échelle qui est le nombre de cibles atteintes par le message original. La dernière dimension, la portée, est la profondeur que va atteindre l'information.

Un point intéressant mentionné dans le travail de Flores García est la rapidité. En effet, il dit que la rapidité à laquelle la nouvelle se répand détermine le succès de celle-ci. Si la nouvelle attire l'attention par son message, ou que c'est une grande entreprise qui publie cette nouvelle, celle-ci sera considérée comme une information sûre et sera diffusée plus rapidement que les autres.

Selon Flores García, une information peut avoir différentes qualités dont des liens vers une source qui lui donne une certaine crédibilité, une police de texte différente qui la distingue des autres ou encore le fait de mentionner une personne qui lui donnera du poids. À noter aussi que la rapidité de propagation dépendra aussi de ce que l'utilisateur a l'habitude de publier, ses thèmes de prédilection comme la politique ou l'écologie.

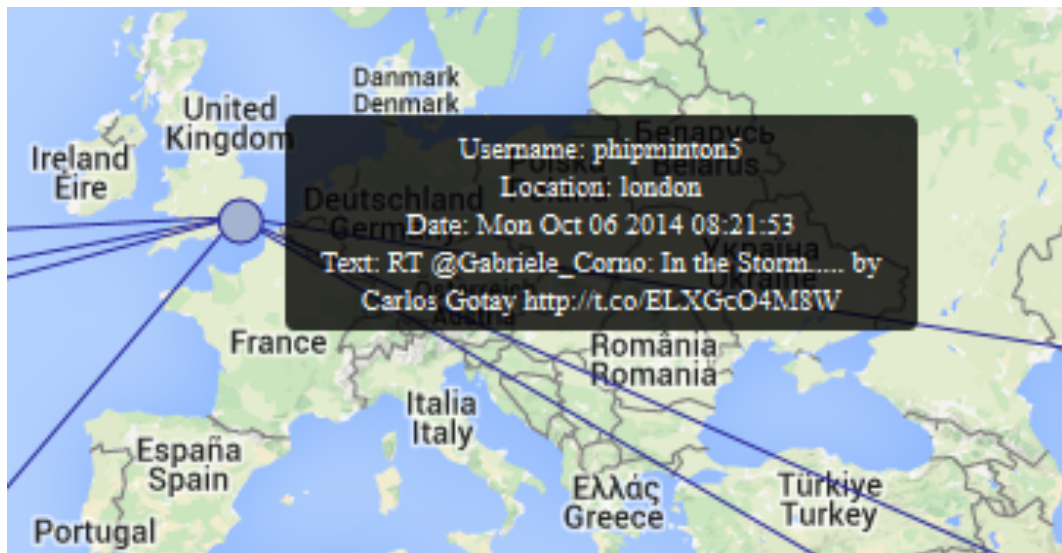
Pour illustrer le fruit de ses recherches, Flores García a créé comme interface la carte ci-dessous sur laquelle nous retrouvons non seulement les trois variables (vitesse, échelle, portée), mais également la localisation des messages publiés :

Figure 6 – Interface générale



Source : Flores García (2015, p.33)

Figure 7 – Fenêtre détaillant les informations sur pseudo



Source : Flores García (2015, p.34)

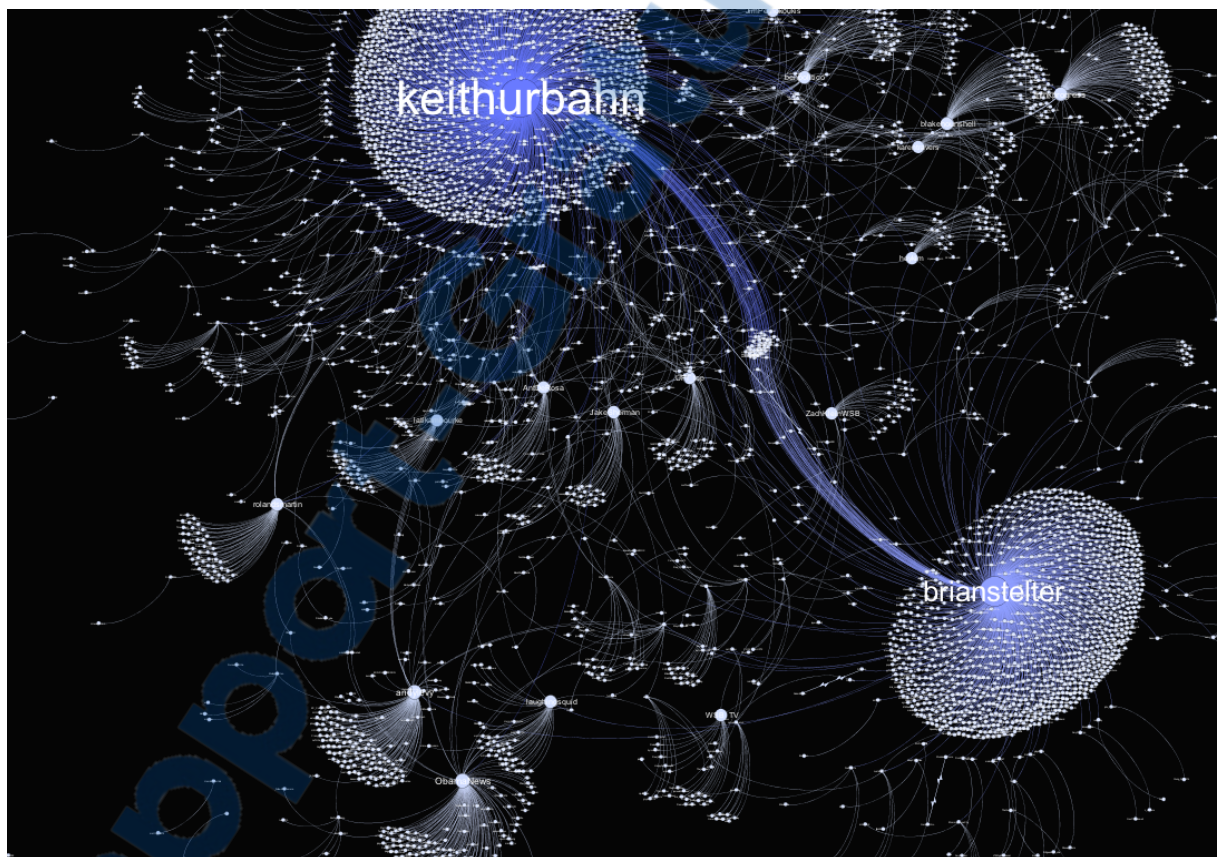
Lorsque nous survolons un nœud, nous pouvons obtenir des informations supplémentaires comme le pseudo de l'utilisateur, sa localisation, la date de son message (original ou partagé) ainsi qu'une partie du texte.

1.2.3. SocialFlow

L'article de SocialFlow, écrit en anglais, parle non pas d'un jeu de données, mais d'un seul tweet qui a été responsable d'une avalanche de tweets. Keith Urbahn, en se servant de l'adresse mail d'urgence du président Obama, a annoncé sur Twitter, une heure avant le discours officiel de la Maison Blanche à Washington, la mort de Bin Laden (Gilad & Devin, 2011).

Le groupe SocialFlow a donc analysé l'impact du message de Keith Urbahn. Le groupe a analysé 14,8 millions de tweets afin de représenter cet événement graphiquement :

Figure 8 – Représentation de la diffusion d'un tweet

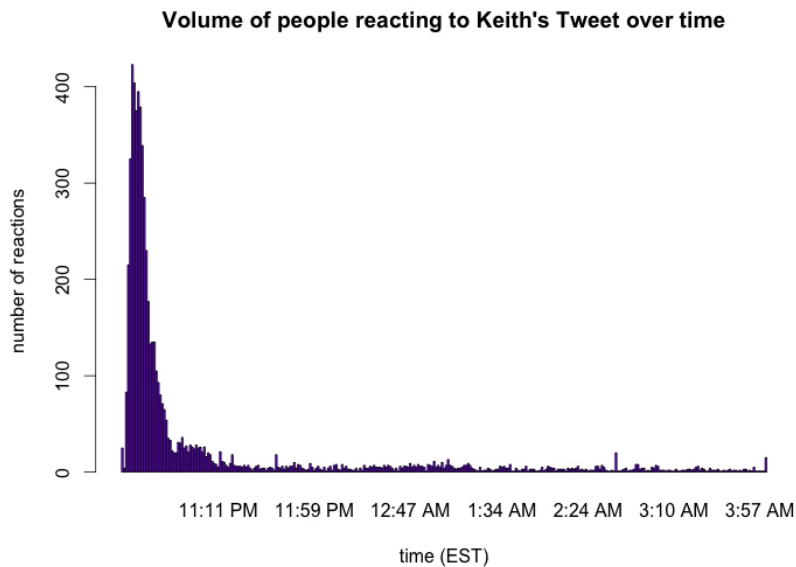


Source : Gilad & Devin (2011)

En seulement une minute, 80 personnes ont partagé le message de Keith Urbahn. Comme vous pouvez le voir, Brian Stelter (sous le nom de « brianstelter »), journaliste au NYTimes, a de son côté accéléré la propagation de la fausse nouvelle. Il a fait partie des premières personnes à relayer cette information. Deux minutes après, plus de 300 personnes avaient partagé la nouvelle.

Pour mieux visualiser ces chiffres, ci-dessous un graphique représentant le volume de personnes qui ont réagi au tweet de Keith Urbahn :

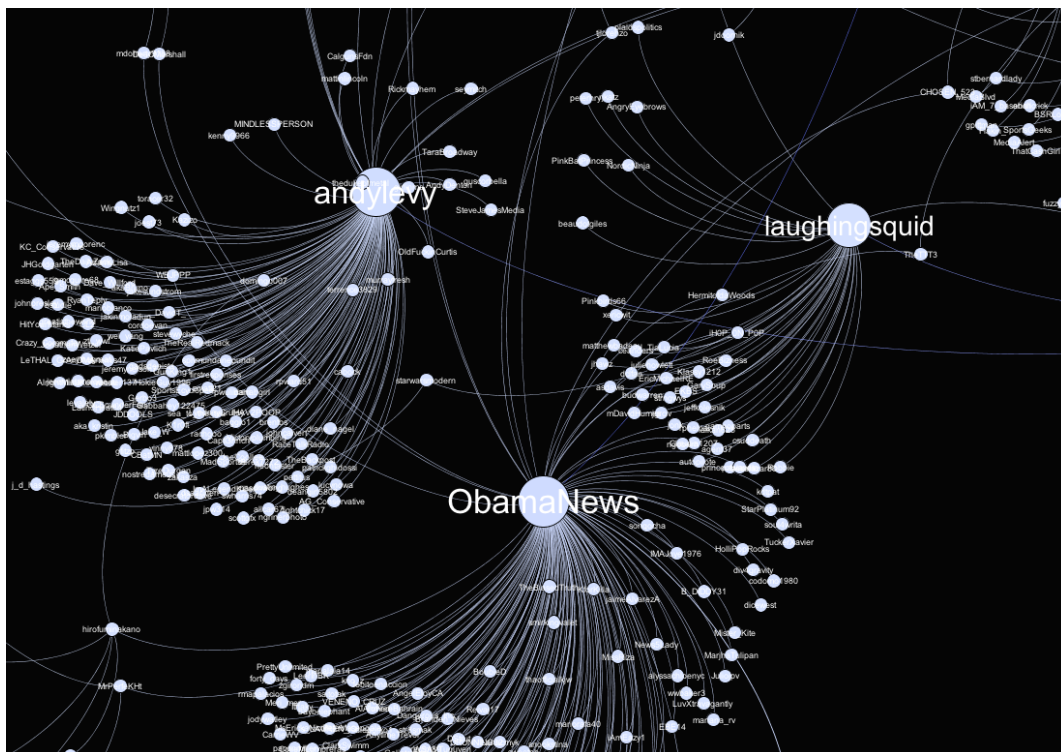
Figure 9 – Graphique des réactions des utilisateurs



Source : Gilad & Devin (2011)

Le graphique de Gilad & Devin montre que d'autres utilisateurs Twitter ont aussi partagé l'information :

Figure 10 – Agrandissement de la figure 8



Source : Gilad & Devin (2011)

Gilad & Devin pensent que la persuasion, l'autorité ainsi que la confiance jouent un rôle essentiel d'influence. Nous pouvons voir que ces trois utilisateurs mentionnés sur la Figure 10 ont vraisemblablement poussé leurs abonnés à partager également la nouvelle.

1.2.4. Synthèse des travaux

Après avoir cherché les travaux similaires, nous n'en avons trouvé aucun correspondant à ce que nous voulions faire. Cependant, certains travaux pourront nous aider à mieux analyser les résultats que nous obtiendrons.

Tableau 2 – Synthèse des travaux

	Reputatio Lab	Fernando Flores García	SocialFlow
Analyse d'un jeu de données en français	✓	✗	✗
Sources variées / sujet variés	✗	✓	✗
Graphique de rapidité de diffusion	✓	✓	✓
Analyse des utilisateurs	✓	✗	✗

Source : données de l'auteur

1.3. Jeux de données

Comme dit précédemment, il n'y a malheureusement que très peu de jeux de données en français regroupant des fausses nouvelles. Cependant, nous en avons tout de même trouvé quelques-uns que nous pourrions exploiter.

1.3.1. Le Gorafi

Le Gorafi est un journal satirique français qui se plaît à inventer et diffuser des articles parodiques. Ces articles sont écrits uniquement par leur équipe. Ce journal est suivi par plus d'un million d'utilisateurs sur les réseaux sociaux Facebook et Twitter et a publié à ce jour plus de 20'000 messages sur Twitter.

1.3.2. NordPresse

NordPresse, journal belge, rejoint l'idée du Gorafi. Il est également présent sur les réseaux sociaux, mais est nettement moins suivi et actif. Sur Facebook, Nordpresse comptabilise plus de 100'000 abonnés. En revanche, sur Twitter, le journal a publié plus de 3000 tweets mais n'est suivi que par une dizaine de milliers d'utilisateurs.

1.3.3. LeMonde.fr

LeMonde est un journal français qui a publié une enquête réalisée en 2017 permettant de recenser des articles mensongers diffusés sur les réseaux sociaux (Sénécat, Les premiers diffuseurs de fausses informations sont souvent des pages Facebook douteuses, 2017).

LeMonde a une équipe qu'il nomme « Les Décodeurs ». Cette équipe a, pendant toute l'année (2017), recenser presque 5000 liens redirigeant vers une fausse nouvelle. À la fin de l'année, il y avait 2865 liens redirigeant sur le réseau social Facebook, et environ 2000 liens vers divers réseaux sociaux comme Twitter ou Youtube (LeMonde, 2017).

Les Décodeurs ont procédé à la récupération de ces données fausses en les compilant, en expliquant en quoi l'information était fausse et finalement en publiant une rectification, un démenti dans le but de rétablir les faits.

L'équipe des Décodeurs a mis à disposition gratuitement sa base de données, et il est donc possible de la télécharger à condition d'en mentionner la source.

1.3.4. Synthèse des jeux de données

Comme nous pouvons voir sur le Tableau 3 ci-dessous, le jeu de données du journal LeMonde.fr est celui qui correspond le mieux à nos critères. Même si tous les liens que LeMonde a collectés ne redirigent pas vers un réseau social, il a l'avantage d'avoir des fausses données écrites par différentes personnes.

Tableau 3 – Comparaison jeux de données

	Le Gorafi	NordPresse	LeMonde.fr
Données en français	✓	✓	✓
Données disponibles sur un réseau social	✓	✓	(✓)
Sources variées (plusieurs auteurs)	✗	✗	✓
> 300 sources	✓	✓	✓

Source : données de l'auteur

1.4. Plateformes similaires

Les plateformes ne sont pas nombreuses pour les fausses nouvelles écrites en français, surtout si l'on compare avec les plateformes qui traitent des données en anglais. Cependant, nous en avons trouvé quelques-unes.

1.4.1. Hoaxbuster

Hoaxbuster est un site qui existe depuis 2000. Ce site référence ce qui circule sur internet, et démontre pourquoi l'information en question est fausse.

Sur la page d'accueil de Hoaxbuster, nous voyons qu'au milieu il y a les dernières fausses nouvelles. Grâce à son champ de recherche, le site nous laisse la possibilité de chercher parmi les articles ou forums le sujet qui nous intéresse.

Figure 11 – Page d'accueil du site Hoaxbuster

Source : capture sur <http://www.hoaxbuster.com>

Afin de tester la fonctionnalité de recherche du site, nous allons chercher des articles ayant comme mot-clé « Macron ». Ci-dessous les résultats :

Figure 12 – Résultats de recherche avec le mot-clé "Macron"

The screenshot shows the Hoaxbuster website interface. On the left, under 'RÉSULTATS DE LA RECHERCHE', several article snippets are listed, including 'Exit la langue française !', 'Fake sms au FN', 'Instituteurs, tous séducteurs ?', 'Le decret de la mort qui tue', 'Ridiculous Anonymous France', and 'Sondages à gogos'. On the right, a search bar contains the text 'Macron'. Below the search bar, there are tabs for 'articles' and 'forums', and a red pin icon. Further down, a section titled 'LES HOAX LES PLUS...' lists 'Consultés' and 'Commentés' items, such as 'Actualisation Windows Live', 'fillette chinoise écrasée', 'Démolition contrôlée du WTC', 'Facebook, Google coupent la désinfo ?', 'Le policier blessé', and 'Données personnelles & Facebook'.

Source : capture sur <http://www.hoaxbuster.com>

Après avoir entré le mot-clé, Hoaxbuster nous propose une série d'articles en relation avec « Macron ». Nous pouvons voir que le site met en évidence le mot-clé cherché. Nous avons sélectionné l'article « Fake sms au FN » comme exemple :

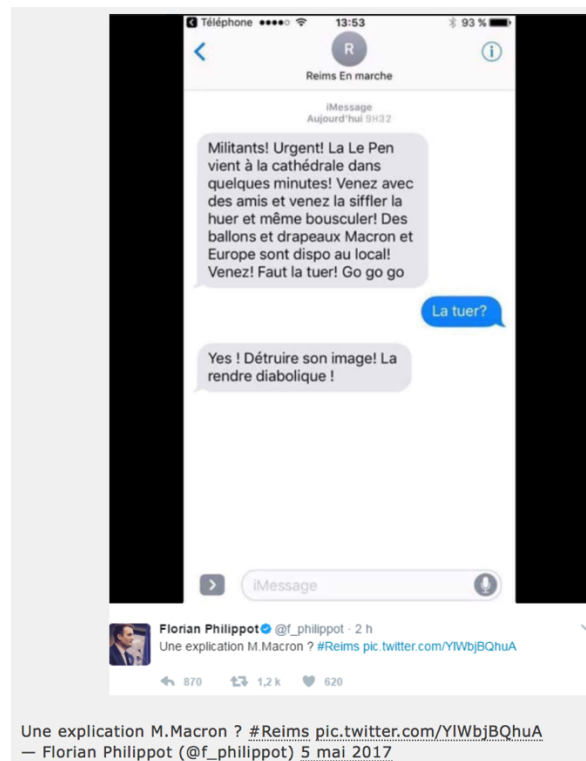
Figure 13 – Hoaxbuster : détail d'un article

The screenshot shows the detailed view of an article on Hoaxbuster. The article title is 'La palme du ridicule'. It is categorized as 'Type : Propagande' and 'En circulation depuis : Mai 2017'. The status is marked as 'Faux' with a red dot. Below the title, there are tabs for 'ARTICLE', 'VOS RÉACTIONS', and 'INFO/INTOX'. The main content area shows a snippet of the article titled 'Fake sms au FN', which discusses a campaign of fake news and a festival of fake news, mentioning Florian Philippot.

Source : capture sur <http://www.hoaxbuster.com>

Cet article relaie un tweet (qui n'est actuellement plus disponible) que le camp Macron aurait envoyé. Nous avons donc dû faire une capture d'écran du site. En voici le message :

Figure 14 – Hoaxbuster : contenu de la fausse nouvelle



Source : capture sur <http://www.hoaxbuster.com>

En dessous de la fausse nouvelle, l'équipe de Hoaxbuster prouve que le message est un montage. Ci-dessous, une capture d'écran qui montre un extrait de la rectification apportée par Hoaxbuster :

Figure 15 – Hoaxbuster : justifications de la fausse nouvelle



Source : capture sur <http://www.hoaxbuster.com>

Les nouvelles sont classées selon différents types tels que rumeur, information, désinformation, mise en garde, etc. Le site donne aussi le mois où elles ont été lancées et leur statut qui se décline par faux, « du vrai, du faux », vrai.

Cela étant, les informations sur l'utilisateur qui a diffusé la nouvelle fausse ne sont que très peu détaillées. Hoaxbuster ne donne aucune information à propos de la vitesse à laquelle la fausse nouvelle s'est propagée, pas plus qu'il ne permet de visualiser la propagation de l'information.

1.4.2. HOAXY

HOAXY est un site plus récent que Hoaxbuster. Sur la page FAQ du site, l'équipe derrière HOAXY précise qu'elle puise aussi bien ses informations auprès de diffuseurs de nouvelles peu fiables qu'auprès de sites comme factcheck.org qui, comme son nom l'indique, vérifie les nouvelles. La plateforme HOAXY ne classe pas les nouvelles par type comme Hoaxbuster. Elle les diffuse en laissant l'utilisateur juger de la qualité de l'information (FAQ Index, s.d.).

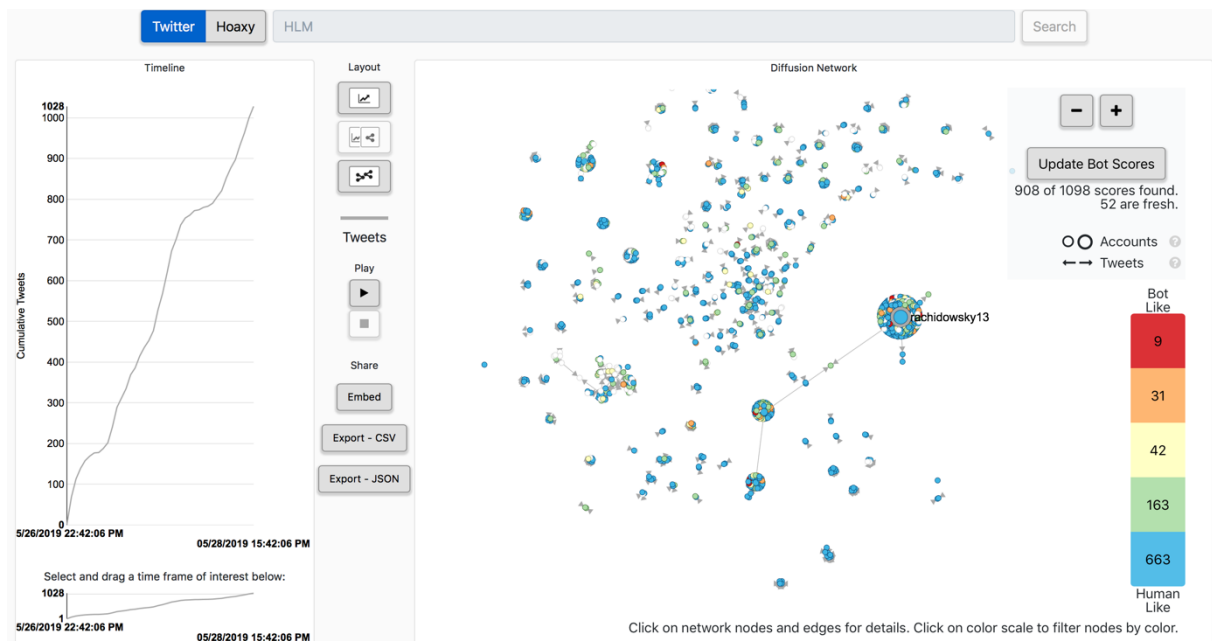
Ce site permet de chercher un thème à travers Twitter ou dans sa propre base de données. Nous avons également la possibilité de choisir la langue des résultats.

Figure 16 – HOAXY : page d'accueil

Source : capture sur <https://hoaxy.iuni.iu.edu>

Afin de démontrer les fonctionnalités de Hoaxy, nous avons choisi un sujet qui a fait polémique en France sur les habitations à loyer modéré (HLM). En insérant l'acronyme HLM, voici les résultats fournis par Hoaxy :

Figure 17 – HOAXY : résultats de recherche avec le mot-clé "HLM"

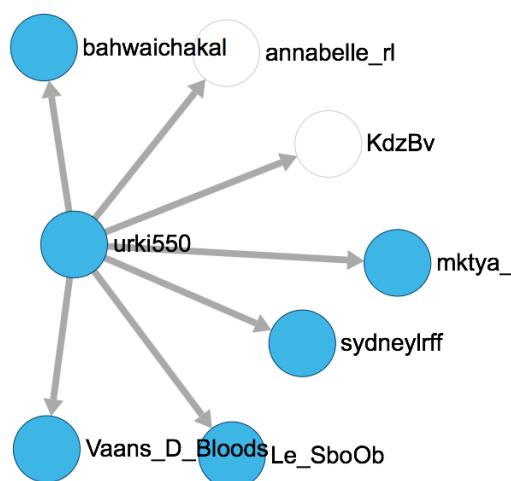


Source : capture sur <https://hoaxy.iuni.iu.edu>

Nous pouvons voir que le site affiche la Timeline avec l'évolution du nombre de tweets ainsi qu'un graphique connecté représentant la diffusion du sujet recherché.

En agrandissant la Figure 17 au hasard, nous sommes tombés sur cette partie du graphique connecté. Nous avons cherché à connaître le contenu du message de l'utilisateur « urki550 ».

Figure 18 – HOAXY : agrandissement de la figure 17



Source : capture sur <https://hoaxy.iuni.iu.edu>

Nous pouvons voir que 7 personnes ont partagé son message. Si nous nous rendons sur Twitter, que nous cherchons son profil ainsi que le message qui parle du sujet, nous pouvons voir que son message n'était pas une fausse nouvelle :

Figure 19 – Twitter : détail du message de l'utilisateur @urki550


 Source : capture sur <https://twitter.com/home?lang=fr>

1.4.3. Synthèse des plateformes existantes

Voici tous les critères que nous suivrons pour notre plateforme. Comme nous pouvons le voir, il n'existe à ce jour aucune plateforme respectant tous les éléments que nous souhaitons implémenter.

Tableau 4 – Comparatif plateformes

	Hoaxbuster	HOAXY
Données en français	✓	✓
Graphique diffusion d'informations	✗	✓
Fausse informations uniquement	✓	✗
Champ de recherche	✓	✓
Données sur l'utilisateur	✗	(✓)
Informations sur la localisation	✗	✗

Source : données de l'auteur

2. Méthodologie

Avant de nous lancer dans le développement, nous devons faire quelques choix technologiques. Il est important de pouvoir comparer toutes nos possibilités avant de développer.

2.1. Réseau social

Le but de notre travail est de visualiser la propagation des fausses nouvelles diffusées sur les réseaux sociaux. Nous devons donc choisir un réseau social qui nous permettra de télécharger ses données que nous pourrions afficher sur notre plateforme de visualisation. Nous avons sélectionné deux réseaux sociaux les plus connus afin d'examiner leurs interfaces de programmation d'application (API). Une API permet de rendre accessible publiquement des données que nous souhaitons partager.

2.1.1. Twitter

Twitter a une multitude d'API permettant de récupérer toutes sortes d'informations. L'avantage est que Twitter ne limite pas ses API à une application que nous aurions créée, comme c'est le cas pour Facebook.

Sur son site internet, Twitter met en avant le fait de pouvoir analyser l'audience, la conjoncture du marché, les tendances, les sujets à la mode et les dernières nouvelles/informations. Twitter propose des API qui permettent par exemple de filtrer les tweets à l'aide de mots-clés proposés par Twitter, de chercher des tweets qui correspondent à ce que nous désirons ou encore de trouver les tendances qui sont propres à une région (Evaluate Twitter data to inform business decisions., s.d.).

2.1.2. Facebook

Les données auxquelles nous pouvons accéder sont plus nombreuses que sur Twitter. Le réseau social a même mis en place un simulateur qui renvoie les requêtes que nous lui envoyons. Voici un exemple de tous les champs que nous avons pu récupérer de notre utilisateur :

Figure 20 – Données récupérables avec l'API Facebook

Node: me

- ☒ id
- ☒ name
- ☒ about
- ☒ address
- ☒ age_range
- ☒ birthday
- ☒ education
- ☒ email
- ☒ favorite_athletes
- ☒ favorite_teams
- ☒ first_name
- ☒ gender
- ☒ hometown
- ☒ inspirational_people
- ☒ interested_in
- ☒ languages
- ☒ last_name
- ☒ political
- ☒ religion
- ☒ friends

8 Debug Messages (Show)

```
{
  "id": "2453829534639782",
  "name": "Nicolas Piguet",
  "email": "piguetsnico@gmail.com",
  "first_name": "Nicolas",
  "last_name": "Piguet"
}
```

Source : capture sur

https://developers.facebook.com/tools/explorer/?method=GET&path=me%3Ffields%3Did%2Cname%2Cabout%2Caddress%2Cage_range%2Cbirthday%2Ceducation%2Cemail%2Cfavorite_athletes%2Cfavorite_teams%2Cfirst_name%2Cgender%2Chometown%2Cinspirational_people%2Cinterested_in%2Clanguages%2Clast_name%2Cpolitical%2Creligion%2Cfriends&version=v3.3

Beaucoup de champs sont grisés car nous n'avons pas donné l'autorisation à l'application de pouvoir récupérer ces informations.

Grâce à ces API qui permettent de faire des statistiques, les créateurs de contenu peuvent mieux orienter leur offre pour attirer davantage d'utilisateurs. Malheureusement, les API de Facebook permettent uniquement de récupérer les données des utilisateurs qui ont recours à un service de Facebook (comme un jeu ou un système de login). Nous ne pouvons donc pas utiliser leurs API pour récupérer du contenu publié sur Facebook.

Après avoir étudié les possibilités de récupérer du contenu de ces réseaux sociaux, nous nous sommes vus contraints d'utiliser le réseau social Twitter. En effet, en le choisissant, nous pourrions récupérer une partie de ses données afin de les afficher sur notre plateforme.

2.2. Jeu de données en français

Nous avons pris la décision de nous baser sur le jeu de données mis à disposition par le journal LeMonde.fr. En effet, ses données correspondent le mieux à nos critères mentionnés précédemment. Ces données sont constamment mises à jour par l'équipe du journal. Elles sont consultables en ligne et nous avons pu les télécharger dans le format JavaScript Object Notation (JSON).

Le fichier JSON est divisé en deux parties :

1. La première partie liste tous les sujets qui ont fait polémique et qui ont été prouvés comme faux. Au moment où nous avons récupéré le fichier (18 mai 2019), l'équipe des Décodeurs du journal LeMonde en avait listé 294.

Pour chaque cas, le thème est numéroté, une phrase résume l'affirmation diffusée, ensuite LeMonde donne une brève rectification des faits qui résume l'article que LeMonde a publié sous lien.

Ci-après, un exemple :

« "16":["77 000 logements HLM attribués aux migrants ?", "FAUX", "Il s'agit d'une déformation de ce qui n'était qu'une proposition consistant à attribuer «une partie» de ces 77 310 logements sociaux durablement vides en France.", "https://www.lemonde.fr/les-decodeurs/article/2015/09/16/non-77-000-logements-sociaux-ne-vont-pas-etre-attribues-aux-refugies_4758934_4355770.html"] » (LeMonde.fr).

2. La deuxième partie du fichier JSON recense toutes les publications trouvées (souvent sur les réseaux sociaux) à propos de chaque sujet. Ci-dessous, les liens correspondant au sujet 16 :

« "https://twitter.com/f_philippot/status/643511238908055552": "16", "http://lagauchematuer.fr/2015/09/15/letat-reserve-77000-hlm-pour-les-migrants-la-france-compte-150000-sdf-et-3-millions-de-mal-loges": "16", "http://lesobservateurs.ch/2016/11/30/paris-annonce-une-aide-additionnelle-de-250-millions-deuros-par-an-pour-la-tunisie": "16", "http://www.prnewswire.com/news-releases/hillarys-health-concerns-serious-say-most-doctors-polled-by-the-association-of-american-physicians-and-surgeons-aaps-300325065.html" » (LeMonde.fr).

Comme nous pouvons le voir, le fichier JSON ne contient pas que des sources provenant du réseau social Twitter. En effet, sur un total d'un peu moins de 5000 liens, il n'y a que 540 sources provenant de Twitter.

Sur ces 540 sources, nous nous sommes aperçus qu'elles n'étaient pas toutes en français, que certaines n'avaient même pas été partagées et que d'autres avaient été supprimées par leur auteur. Après avoir supprimé les sources ne correspondant pas à nos critères (voir Tableau 1), il ne nous en est resté que 162.

Nous avons donc dû compléter nos sources avec d'autres liens. Nous nous sommes basés sur un article écrit par Adrien Sénecat dans le journal LeMonde qui liste trois sujets qui ont massivement circulé sur les réseaux sociaux (Sénecat, Trois fausses informations récentes qui ont influencé l'opinion, 2017). Voici les trois sujets que l'auteur a traités :

1. L'Etat français a mis de côté 77'000 logements pour les migrants
2. La candidate Hillary Clinton à la présidentielle américaine était très malade lors de sa campagne en 2017
3. Les pics de pollution en France sont causés par les usines de charbon allemandes

Nous avons donc pu chercher manuellement ces trois sujets sur Twitter. Nous les avons ensuite rassemblés avec ceux collectés par leMonde. Au total, nous avons une base de 217 liens vers une fausse nouvelle provenant de Twitter.

Après avoir travaillé uniquement avec un jeu de données de fausses nouvelles, nous avons trouvé judicieux, avec les responsables du projet, de procéder à une comparaison entre notre jeu de fausses nouvelles et un jeu de nouvelles véridiques.

D'un commun accord, le journal LeMonde nous est apparu comme une source fiable. Par conséquent, nous avons décidé de télécharger un jeu de nouvelles véridiques publiées sur le compte Twitter du journal LeMonde.fr

2.3. Twitter API

Maintenant que nous avons nos sources de fausses nouvelles, nous allons les télécharger. Twitter met à disposition différentes API qui permettent d'obtenir des informations venant de leur plateforme. Leurs API sont disponibles gratuitement à condition d'avoir un compte développeur.

2.3.1. Jargon Twitter

Avant de nous attarder sur la création d'un compte développeur, nous allons définir quelques termes propres à Twitter. En effet, ce réseau social utilise un jargon qu'il est nécessaire de définir afin de savoir de quoi nous parlons. Nous nous sommes aidés du glossaire écrit en anglais publié sur le site de Twitter (Glossary, s.d.). Le compte personnel Twitter de l'auteur a été utilisé afin d'illustrer les définitions du glossaire.

2.3.1.1. Tweet

Un tweet est un message qu'un utilisateur poste depuis son compte. Il peut contenir du texte, une photo ou une vidéo. Voici un exemple de tweet avec uniquement du texte :

Figure 21 – Tweet d'exemple



Source : capture sur <https://twitter.com/PiguetNicolas/status/1138445053960032256>

2.3.1.2. Friend

Sur Twitter, un « friend » est un utilisateur que vous suivez, mais lui ne vous suit pas forcément. En français, Twitter traduit friend par abonnement.

2.3.1.3. Timeline

Une timeline regroupe plusieurs tweets publiés ou partagés. Twitter distingue deux types de timeline. Il y a la « Home timeline » (le fil d'actualité de votre compte) qui est la page d'accueil regroupant tous les tweets partagés par vos abonnements. Lorsque vous vous rendez sur le profil d'un utilisateur, tous les tweets qu'il a partagés et publiés sont aussi appelés timeline.

2.3.1.4. Follower

Un follower (ou abonné en français) est un utilisateur qui s'est abonné à vous afin de recevoir les tweets que vous publiez dans sa « Home timeline ».

2.3.1.5. Retweet

Un retweet est un tweet que vous avez partagé et qui s'est ajouté à votre « Home timeline » afin que vos followers le voient également.

Il existe deux types de retweet :

1. Retweet sans commentaire

Ce retweet consiste à partager le tweet original tel quel. Voici un exemple d'un tweet que nous avons retweeté sans y ajouter de commentaire :

Figure 22 – Retweet sans commentaire



Source : capture sur <https://twitter.com/BMWFrance/status/1149350179201605632>

2. Retweet avec commentaire

Via notre compte twitter, nous avons également retweeter notre tweet, et y avons ajouté un commentaire :

Figure 23 – Retweet avec commentaire



Source : capture sur <https://twitter.com/PiguetNicolas/status/1138448014283030529>

2.3.1.6. Retweeter

Un retweeter est une personne qui partage le tweet qu'un utilisateur a écrit.

2.3.1.7. Bio

Une bio est une description personnelle de votre profil d'utilisateur.

2.3.2. Compte développeur

Comme dit précédemment, il a fallu que nous créions un compte développeur afin d'avoir un accès aux API de Twitter. Un compte utilisateur normal est requis afin de migrer vers un compte développeur. Nous avons donc utilisé le compte que nous avons créé.

Une fois le compte développeur obtenu, nous avons dû justifier auprès de Twitter les raisons pour lesquelles nous devons utiliser leurs API pour finalement obtenir leur approbation.

2.3.3. API

Les quatre API que nous utiliserons pour récolter les informations sont les suivantes :

2.3.3.1. GET statuses/show/:id

Cette API donne accès à toutes les informations détaillées d'un tweet. Elle nous permettra de récupérer tous les liens que nous avons collectés et il suffira de mettre l'identifiant d'un tweet pour accéder au contenu de ce dernier.

Voici un exemple d'une requête pour récupérer le tweet de la Figure 24 :

GET <https://api.twitter.com/1.1/statuses/show.json?id=834293078592741377>

Figure 24 – Tweet récupéré



Source : capture sur <https://twitter.com/OlivierCOLLAS/status/834293078592741377>

Voici une partie de ce que l'API nous renvoie :

```
{
  "created_at": "Wed Feb 22 06:45:45 +0000 2017",
  "id": 834293078592741377,
  "id_str": "834293078592741377",
  "text": "Et ce pompier caillassé par les racailles, il aurait droit à une visite d'Hollande et un maillot de Ribery ? https://t.co/KWNq1qBSQg",
  ...
  "user": {
    "id": 585482800,
    "id_str": "585482800",
    "name": "Olivier COLLAS",
    "screen_name": "OlivierCOLLAS",
    "location": "Versailles",
    "description": "Président de diverses organisations\nCo-fondateur de l'UNI & du MIL mais surtout fervent Gaulliste et fier de l'être\nPrésident co-fondateur d'Union Républicaine.",
    ...
    "protected": false,
    "followers_count": 5841,
    "friends_count": 3138,
    "listed_count": 109,
    "created_at": "Sun May 20 08:22:27 +0000 2012",
    "favourites_count": 1603,
    "utc_offset": null,
    "time_zone": null,
    "geo_enabled": true,
    "verified": false,
    "statuses_count": 87576,
    "lang": null,
    ...
  },
  "geo": null,
  "coordinates": null,
  "place": null,
  "contributors": null,
  "is_quote_status": false,
  "retweet_count": 717,
  "favorite_count": 416,
  "favorited": false,
  "retweeted": false,
  "possibly_sensitive": false,
  "possibly_sensitive_appealable": false,
  "lang": "fr"
}
```

Source : Twitter API

2.3.3.2. GET statuses/retweets/:id

Cette API permet de récupérer les retweets d'un tweet. Il suffit uniquement de passer comme paramètre d'entrée l'identifiant du tweet original, et l'API se chargera de renvoyer les retweets de ce dernier. Malheureusement, pour cette API, Twitter ne permet pas de télécharger plus de 100 retweets.

Voici un exemple de requête pour le tweet ci-dessous :

GET <https://api.twitter.com/1.1/statuses/retweets.json?id=841324754678272000>

Figure 25 – Tweet dont nous allons récupérer les retweets



Source : capture sur <https://twitter.com/RTenfrancais/status/841324754678272000>

Voici une partie de ce que l'API nous renvoie :

```
[
  {
    "created_at": "Mon Mar 13 19:08:34 +0000 2017",
    "id": 841365384729112577,
    "id_str": "841365384729112577",
    "text": "RT @RTenfrancais: Ce qui fut l'angle d'attaque préféré de #DonaldTrump pendant
    plusieurs années... + D'INFOS 📺 https://t.co/sa6mGeyzah",
    "truncated": false,
    "entities": {
      "hashtags": [
        {
          "text": "DonaldTrump",
          "indices": [
            58,
            70
          ]
        }
      ],
      "symbols": [],
      "user_mentions": [
        {
          "screen_name": "RTenfrancais",
          "name": "RT France",
          "id": 3028247758,
          "id_str": "3028247758",
          "indices": [
            3,
            16
          ]
        }
      ]
    },
    ...
  },
  ...
  "user": {
    "id": 4228895379,
    "id_str": "4228895379",
    "name": "maral",
    "screen_name": "maralpoutine",
    ...
  }
}
```

```

...
},
"geo": null,
"coordinates": null,
"place": null,
"contributors": null,
"retweeted_status": {
  "created_at": "Mon Mar 13 16:27:07 +0000 2017",
  "id": 841324754678272000,
  "id_str": "841324754678272000",
  "text": "Ce qui fut l'angle d'attaque préféré de #DonaldTrump pendant plusieurs années... +
D'INFOS https://t.co/sa6mGeyzah",
  ...
  "in_reply_to_status_id": null,
  "in_reply_to_status_id_str": null,
  "in_reply_to_user_id": null,
  "in_reply_to_user_id_str": null,
  "in_reply_to_screen_name": null,
  "user": {
    "id": 3028247758,
    "id_str": "3028247758",
    "name": "RT France",
    "screen_name": "RTenfrancais",
    "location": "",
    "description": "Les actualités de RT France",
    ...
  },
  "geo": null,
  "coordinates": null,
  "place": null,
  "contributors": null,
  "is_quote_status": false,
  "retweet_count": 6,
  "favorite_count": 1,
  "favorited": false,
  "retweeted": false,
  "possibly_sensitive": false,
  "lang": "fr"
},
"is_quote_status": false,
"retweet_count": 6,
"favorite_count": 0,
"favorited": false,
"retweeted": false,
"possibly_sensitive": false,
"lang": "fr"
},
{
  "created_at": "Mon Mar 13 17:52:50 +0000 2017",
  "id": 841346326789029888,
  "id_str": "841346326789029888",
  "text": "RT @RTenfrancais: Ce qui fut l'angle d'attaque préféré de #DonaldTrump pendant
plusieurs années... + D'INFOS https://t.co/sa6mGeyzah",
  ...
}

```

Source : Twitter API

2.3.3.3. Get followers/ids

Cette API permet d'avoir une liste des identifiants des abonnés d'un utilisateur. Chaque requête permet de télécharger jusqu'à 5000 identifiants. Twitter nous impose de ne lancer que 15 requêtes toutes les 15 minutes. Afin de pouvoir établir des statistiques, nous avons besoin de cette API pour récupérer les abonnés des auteurs des tweets. Grâce à cette API, nous saurons si l'utilisateur qui a partagé l'information était abonné à l'auteur du message.

Voici un exemple d'une requête :

GET https://api.twitter.com/1.1/followers/ids.json?screen_name=PiguetNicolas

Voici une partie de la liste que l'API nous renvoie :

```
{
  "ids": [
    3002841635,
    1149261904487096320,
    1116605617,
    1149258635865939968,
    2549638393,
    ...
  ]
}
```

Source : Twitter API

2.3.3.4. Get statuses/user_timeline

Cette API permet de récupérer une partie des tweets les plus récents d'un utilisateur.

Voici un exemple d'une requête :

GET https://api.twitter.com/1.1/statuses/user_timeline.json?screen_name=lemondefr

Ci-dessous, une partie de la réponse après avoir appelé l'API :

```
{
  "created_at": "Sun Jul 07 07:11:07 +0000 2019",
  "id": 1147764967908753408,
  "id_str": "1147764967908753408",
  "text": "Il fait trop chaud ? Cinq villes où piquer une tête en Europe https://t.co/VY3KxXi19yv",
  ...
  "user": {
    "id": 24744541,
    "id_str": "24744541",
    "name": "Le Monde",
    "screen_name": "lemondefr",
    "location": "Paris",
    "description": "L'actualité de référence par la rédaction du Monde | Piloté par @marieslavicek @bricelaemle @charlotteherzog | Snapchat, FB, Instagram : lemondefr",
    "url": "https://t.co/Ver70UGkbir",
  }
}
```

```

    "entities": {
      "url": {
        "urls": [
          {
            "url": "https://t.co/Vt.coVer70UGkbir",
            "expanded_url": "https://www.lemonde.fr",
            "display_url": "lemonde.fr",
            "indices": [
              0,
              23
            ]
          }
        ]
      },
      "description": {
        "urls": []
      }
    },
    "protected": false,
    "followers_count": 8268671,
    "friends_count": 624,
    "listed_count": 35274,
    "created_at": "Mon Mar 16 18:44:51 +0000 2009",
    "favourites_count": 1619,
    ...
  },
  "geo": null,
  "coordinates": null,
  "place": null,
  "contributors": null,
  "is_quote_status": false,
  "retweet_count": 4,
  "favorite_count": 9,
  "favorited": false,
  "retweeted": false,
  "possibly_sensitive": false,
  "lang": "fr"
},
{
  "created_at": "Sun Jul 07 06:48:30 +0000 2019",
  "id": 1147759275101556736,
  "id_str": "1147759275101556736",
  "text": "Accord sur le nucléaire : Macron fait part de sa « forte préoccupation » à l'Iran",
  "truncated": false,
  "entities": {
    "url": {
      "urls": [
        {
          "url": "https://t.co/Vt.coVJ9nLVzveCR",
          "expanded_url": "https://www.lemonde.fr",
          "display_url": "lemonde.fr",
          "indices": [
            0,
            23
          ]
        }
      ]
    },
    "description": {
      "urls": []
    }
  },
  "protected": false,
  "followers_count": 8268671,
  "friends_count": 624,
  "listed_count": 35274,
  "created_at": "Mon Mar 16 18:44:51 +0000 2009",
  "favourites_count": 1619,
  ...
}

```

Source : Twitter API

2.4. Bases de données

Nous avons créé une base de données afin de stocker les données récupérées sur Twitter. Celle-ci sera accessible depuis notre plateforme. Il existe une multitude de bases de données. Parmi les plus répandues, nous en avons sélectionné cinq que nous avons classées en deux catégories.

2.4.1. Bases de données en langage de requête structurée (SQL)

Ce système de gestion de base de données (SGBD) utilise SQL afin de manipuler les données. Ce système a été utilisé pendant des années et l'est encore. Son désavantage est qu'il faut définir une structure de table, avec le nom des colonnes, avant même d'insérer des données (Shiff & Rowe, 2018). Ci-dessous, un exemple d'implémentation de ce système avec deux tables :

Tableau 5 – Exemple d'une table (tweet)

Tweet		
Id (clé primaire)	Description	Utilisateur (clé étrangère)
1	Il fait beau aujourd'hui	1
2	Je n'arrive pas à dormir	1
3	Il fait chaud, j'espère qu'il pleuvra bientôt	1

Source : données de l'auteur

Tableau 6 – Exemple d'une table (utilisateur)

Utilisateur	
Id (clé primaire)	Prénom
1	Nico

Source : données de l'auteur

Voici trois bases de données qui font partie de ce SGBD :

2.4.1.1. MySQL

MySQL est la base de données la plus répandue. Elle est fréquemment mise à jour et est disponible gratuitement. Cependant, la version proposée gratuitement est focalisée sur la rapidité et la stabilité plutôt que le nombre de fonctionnalités (Arsenault, 2017).

2.4.1.2. PostgreSQL

Cette base de données est aussi populaire. Elle peut stocker beaucoup de données et est également compatible avec le format JSON. Malgré tous ces avantages, la documentation n'est pas aussi claire que celle de MySQL.

2.4.1.3. MariaDB

Cette base de données est fondée sur MySQL. MariaDB est aussi gratuite. Par ailleurs, si nous désirons les mêmes fonctionnalités payantes que nous trouvons sur MySQL, MariaDB nous revient moins cher. L'équipe derrière MariaDB vante sur son site que « MariaDB ne possède pas de modules ayant les sources fermées tel que ceux pouvant être trouvés dans MySQL Enterprise Edition. De plus, toutes les fonctionnalités non open source de MySQL 5.5 Enterprise Edition sont disponibles dans la version open source de MariaDB » (MariaDB, s.d.).

Rapport-gratuit.com
 LE NUMERO 1 MONDIAL DU MÉMOIRES

2.4.2. Bases de données orientées document

Ce SGBD est plus récent que les bases de données structurées. Il a l'avantage de ne pas dépendre d'une structure prédéfinie. Cependant, il peut y avoir de la répétition de données car ce SGBD ne prend pas en charge les identifiants ainsi que les clés étrangères. Sur l'exemple suivant, on peut voir en jaune que les données sont répétées :

```
[
  {
    "id": 1,
    "description": "il fait beau aujourd'hui",
    "utilisateur": {
      "id": 1,
      "prénom": "Nico"
    }
  },
  {
    "id": 2,
    "description": "Je n'arrive pas à dormir",
    "utilisateur": {
      "id": 1,
      "prénom": "Nico"
    }
  },
  {
    "id": 3,
    "description": "il fait chaud, j'espère qu'il pleuvra bientôt",
    "utilisateur": {
      "id": 1,
      "prénom": "Nico"
    }
  }
]
```

Source : code de l'auteur

Voici deux bases de données qui sont orientées document :

2.4.2.1. MongoDB

Même si MongoDB peut gérer des données relationnelles, ce n'est pas son premier but. En effet, cette base de données est la référence pour les données non structurées. Elle supporte les fichiers JSON et ne nécessite pas de structure comme les bases de données relationnelles.

2.4.2.2. CouchDB

Cette base de données est peu connue comparée à MongoDB. En revanche, si nous faisons du développement mobile, cette base de données est idéale.

2.4.3. Tableau comparatif

Tableau 7 – Comparaison des bases de données

Note 1/6	MySQL	PostgreSQL	MariaDB	MongoDB	CouchDB
Gratuit	4	6	6	6	6
Compatible macOS	6	6	6	6	6
Répétition de code	6	6	6	1	1
Fonctionnalités	4	5	6	5	4
Web	6	6	6	6	4
Documentation	6	4	6	5	3
Total sur 36	32	33	36	29	24

Source : données de l'auteur

2.5. Outils de développement

Avant de commencer à développer notre plateforme, nous devons d'abord examiner les outils de développement disponibles afin de trouver celui dont nous aurons besoin. De moins en moins de sites sont créés uniquement en HTML, CSS & JavaScript. En effet, de puissants framework facilitent le développement web. De plus, étant donné que nous avons besoin d'accéder à des données externes, nous devons trouver une solution nous le permettant. Nous avons sélectionné ci-dessous trois outils qui sont populaires, gratuits, et avec lesquels nous pourrions développer notre plateforme :

2.5.1. Laravel

Laravel est un framework PHP. Il est réputé pour rendre le code agréable à lire, s'exécute rapidement et a une communauté de plus en plus nombreuse. Laravel est également orienté vers une architecture Modèle Vue Contrôleur (MVC). Cette architecture a pour avantage de séparer et structurer le code afin de lui donner une meilleure visibilité. En effet, le Modèle correspond à la structure d'une table dans la base de données. La Vue est l'interface qui permettra de visualiser les données et pour finir le Contrôleur sera là pour gérer les données demandées afin de les transmettre à la Vue.

Cette architecture est très utile lorsque nous avons beaucoup de tables dans notre base de données et que nous créons, modifions et supprimons fréquemment des données de nos tables.

2.5.2. Node.js

Cette plateforme est un environnement JavaScript qui fonctionne en dehors du navigateur. Il lance des scripts afin d'afficher des pages web dynamiques. Il est réputé pour avoir de bonnes performances car il peut également réaliser des tâches asynchrones.

Node.js est l'outil de développement qui est choisi par la majorité des développeurs. En plus d'être développé en JavaScript, Node.js propose beaucoup de bibliothèques qui facilitent la tâche aux développeurs. Comme il intègre relativement bien les API, il permet de créer des projets complexes (Laravel vs Node vs Django, s.d.).

2.5.3. Django

Le premier point fort de Django est la rapidité. Ce framework écrit en Python simplifie le développement web. Django met en avant également la gestion de la sécurité grâce à une communauté qui nous évitera de faire des erreurs récurrentes (Django makes it easier to build better Web apps more quickly and with less code., s.d.). Django a l'avantage d'inclure une administration (utilisateurs, groupes, ...). D'ailleurs, parmi les outils que nous avons choisi de comparer, Django est le seul à proposer cette fonction.

2.5.4. Tableau comparatif

Tableau 8 – Comparaison des outils de développement

Note 1/6	Laravel	Node.js	Django
Facilité	4	5	6
Modules	3	6	3
MVC	6	4	5
Documentation	5	6	5
Communauté	4	5	4
Total sur 30	22	26	23

Source : données de l'auteur

2.6. Outils de visualisation des données disponibles

Plusieurs outils (bibliothèques, API) sont disponibles afin de créer des graphiques qui afficheront diverses données à propos de la nouvelle. Nous en avons trouvé quelques-uns pour notre plateforme.

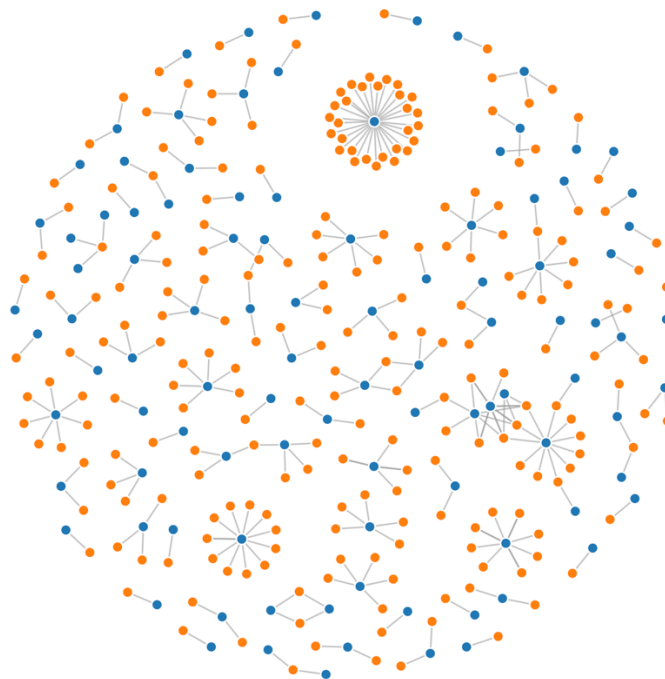
2.6.1. D3.js

D3.js est une librairie Javascript qui permet de manipuler des documents qui reprennent des données. D3.js offre la possibilité de créer des graphiques ou tableaux gratuitement. Par exemple, nous pouvons générer un tableau ou un graphique dans le langage HTML en se basant uniquement sur un tableau de nombres (Data-Driven Documents, s.d.).

D3.js est décrite comme flexible, facile à utiliser. Elle est capable de gérer de gros jeux de données, de réutiliser du code et peut finalement associer des données à un élément d'une page html. Les avantages de cette librairie sont une bonne visualisation des données, la possibilité de ne télécharger qu'une partie de la librairie et l'utilisation d'objets html (D3.js - Introduction, s.d.).

Le développeur de cette librairie, Mike Bostock, a mis en ligne quelques exemples d'utilisation de sa librairie. Le graphique de la Figure 26 est le « Disjoint Force-Directed Graph » (Bostock, 2018) :

Figure 26 – D3.js : Disjoint Force-Directed Graph



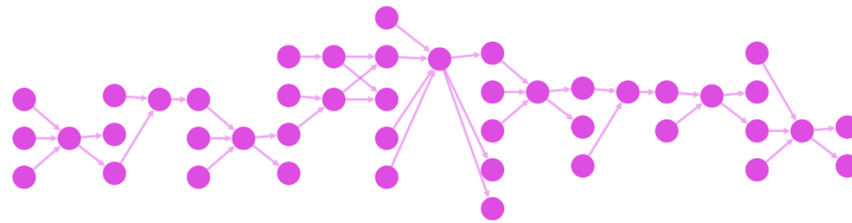
Source : capture sur <https://d3js.org>

2.6.2. Cytoscape.js

Cytoscape.js est aussi une librairie Javascript pour dessiner des graphiques. Elle a l'avantage d'être très optimisée, ainsi que compatible avec des modules comme Node.js ainsi que des gestionnaires de contenus comme le célèbre Node Package Manager (npm). Nous pouvons aussi traiter des fichiers JSON et y ajouter des extensions (Cytoscape.js, 2019).

Ci-dessous un exemple de graphique qu'il est possible de construire avec Cytoscape.js :

Figure 27 – Cytoscape.js : graphique orienté



Source : capture sur <http://js.cytoscape.org>

2.6.3. Chart.js

Cette librairie est une des plus connues car son utilisation est très simple. Il est facile d'obtenir un résultat en quelques lignes de code. Elle propose huit graphiques différents. La structure du code pour déclarer un graphique est concise :

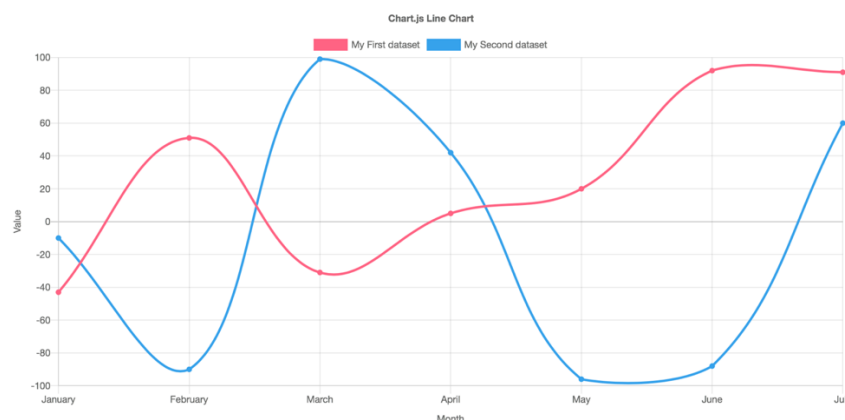
Figure 28 – Chart.js : base de code

```
var myLineChart = new Chart(ctx, {
  type: 'line',
  data: data,
  options: options
});
```

Source : capture sur <https://www.chartjs.org>

Une fois que nous y ajoutons des données, ainsi que des options d'affichage, nous pouvons obtenir un graphique tel que celui que Chart.js montre en exemple sur son site :

Figure 29 – Chart.js : graphique linéaire



Source : capture sur <https://www.chartjs.org>

Malheureusement, cette librairie ne propose pas de graphiques connectés.

2.6.4. Plotly.js

Cette librairie est sûrement la plus simple de toutes. Elle se base sur une autre librairie plus complexe D3.js (Line Charts in plotly.js, 2019). En effet, en très peu de lignes de code, elle nous permet de créer un graphique. De plus, beaucoup d'exemples sont disponibles dans la documentation de la librairie. Voici un exemple de code que nous pouvons trouver afin de créer un graphique simple :

Figure 30 – Plotly.js : code html

```
<head>
  <!-- Plotly.js -->
  <script
    src="https://cdn.plot.ly/plotly-
    latest.min.js"></script>
</head>

<body>

  <div id="myDiv"></div>

</body>
```

Source : capture sur <https://plot.ly/javascript/>

Figure 31 – Plotly.js : code JavaScript

```
var trace1 = {
  x: [1, 2, 3, 4],
  y: [10, 15, 13, 17],
  type: 'scatter',
};

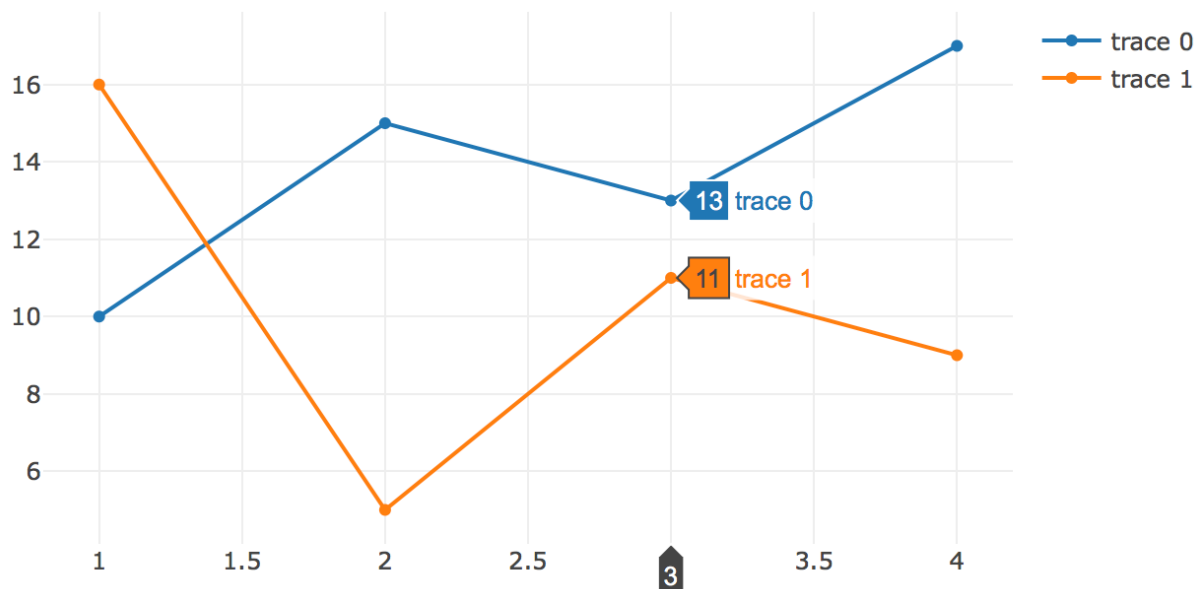
var trace2 = {
  x: [1, 2, 3, 4],
  y: [16, 5, 11, 9],
  type: 'scatter'
};

var data = [trace1, trace2];

Plotly.newPlot('myDiv', data, {},
{showSendToCloud: true});
```

Source : capture sur <https://plot.ly/javascript/>

Figure 32 – Plotly.js : graphique d'exemple



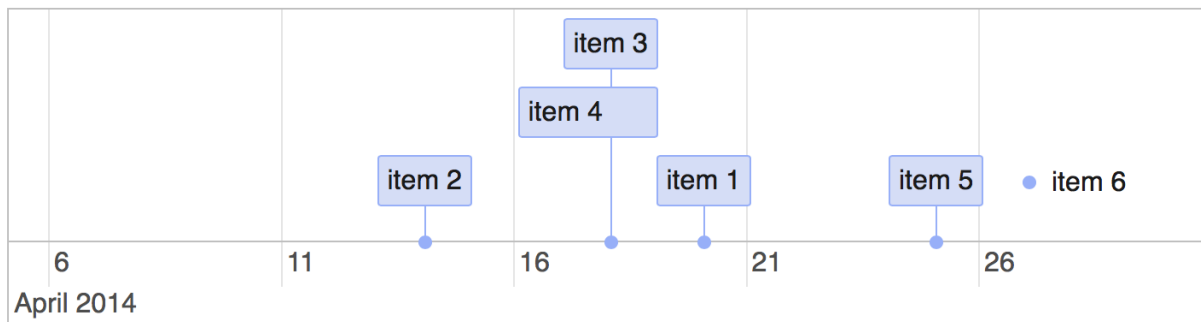
Source : capture sur <https://plot.ly/javascript/>

Avec les quelques lignes de la Figure 30 et de la Figure 31, nous pouvons facilement avoir un résultat. Seul désavantage, cette librairie ne propose pas de graphiques connectés.

2.6.5. Vis.js

Vis.js est une librairie destinée à visualiser dynamiquement des données. Elle a été conçue pour être simple à utiliser, pour gérer une quantité considérable de données et offre également la possibilité d'interagir avec le graphique. Il est possible de créer des graphiques de type « Timeline » et également des graphiques connectés (vis.js, s.d.).

Voici un des exemples basiques que vis.js donne sur son site :



Source : capture sur <https://visjs.org/examples/timeline/basicUsage.html>

2.6.6. Synthèse des outils

Tableau 9 – Comparatif des outils de visualisation

	d3.js	cytoscape.js	chart.js	plotly.js	vis.js
Gratuit	✓	✓	(✓)	✓	✓
Support Javascript	✓	✓	✓	✓	✓
Bonne documentation	✓	(✓)	✓	✓	✓
Graphique Timeline	✓	✓	✓	✓	✓
Graphique connecté	✓	✓	✗	✗	✓
Facilité d'emploi	✗	(✓)	✓	✓	✓
Popularité élevée	✓	✗	✓	(✓)	✗

Source : données de l'auteur

2.7. Choix finaux

Après avoir comparé toutes les technologies qui pourraient nous être utiles, voici un tableau récapitulatif des technologies choisies pour le développement de notre plateforme.

Tableau 10 – Récapitulatif des choix

	Choix	Justifications
Réseau social	Twitter	Nous avons choisi ce réseau social car il est le seul à proposer des API où nous pouvons télécharger des informations librement. Facebook n'autorise pas à télécharger librement du contenu de sa plateforme.
Base de données	MariaDB	Cette base de données bénéficie de tous les avantages de MySQL. MariaDB offre davantage de fonctionnalités gratuites ou moins chères que sur MySQL. Il nous semble donc judicieux de choisir cette base de données afin d'héberger localement toutes nos données.
Outils de développement	Node.js	Nous allons utiliser Node.js pour développer notre plateforme. En plus de tous les points positifs mentionnés précédemment, il a l'avantage d'être écrit en Javascript. En choisissant Node.js, nous nous limitons à 3 langages (html, css, javascript). Les deux autres framework en utilisent 4. Comme notre travail sera intégré au projet global, notre code sera plus simple à utiliser pour les développeurs qui le reprendront.
Librairies de visualisation	Charts.js Vis.js	Ces deux librairies ont pour avantage d'être très simples à manipuler. Lorsque nous nous pencherons sur le développement des graphiques, nous les choisirons en premier.

Source : données de l'auteur

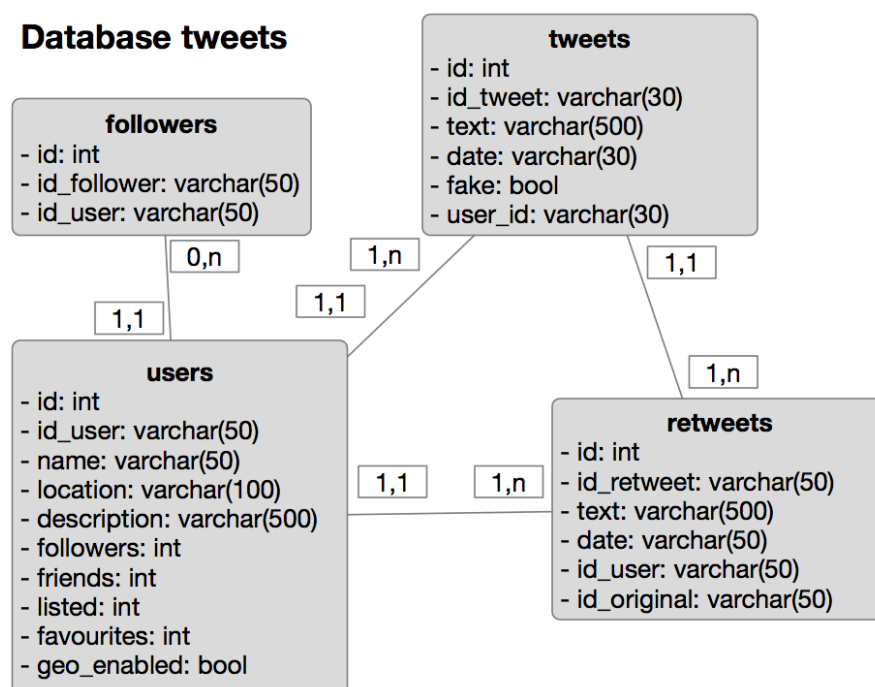
3. Développement

Jusqu'à ce jour, aucun travail n'a encore été effectué par l'Institut d'informatique de gestion. Nous devons donc réfléchir à structurer la base de données et télécharger les données nécessaires. Par la suite, nous pourrons développer la plateforme afin d'afficher des données pertinentes.

3.1. Diagrammes de classe

Avant de développer notre plateforme, nous avons dû réfléchir aux données que nous voulions afficher ainsi qu'à la structure de la base de données. Nous avons donc fait un diagramme de classes afin de mieux visualiser notre solution :

Figure 33 – Structure de notre base de données



Source : capture d'écran de l'auteur

3.2. Collection des données

Ayant choisi une base de données relationnelle, nous avons dû convertir les données renvoyées de l'API en fichiers avec un format « Comma-separated values » (csv). Ce format a l'avantage de pouvoir être lu par n'importe quel programme et est facilement exportable dans d'autres formats.

Un script python avait déjà été écrit dans le projet global afin de récupérer tous les tweets d'un utilisateur Twitter. Cependant, nous avons dû le modifier car il ne correspondait pas à ce

que nous voulions télécharger de Twitter. Au total, nous avons créé quatre scripts avec lesquels nous avons pu obtenir toutes les données.

Voici les parties principales de nos scripts. Ces derniers contiennent tous les clés d'accès pour être autorisés à utiliser les API Twitter :

```
api_key='EkVX0cK8tWXpqnyG9cHTUnRu'
api_secret='A0FksG5QXs0uDD2b5rBKu2MRyWE4gEn91KPOSh4jZUOWlSnvmS'
access_token='1117784259413065729-IaaiXsUJ8ENSvtmqORpXept7w85zEa'
access_token_secret='TiG3td8vqUOf9XCdEivheVwnq0NGLkhPiubMCBF8dLHjY'
```

Source : code de l'auteur

À chaque requête vers l'API de Twitter, nos scripts écrivent un fichier JSON afin de garder les données brutes. Tous ces fichiers seront sauvés dans le dossier *data*. Chaque requête ajoute les données à la variable *data_list*. Celle-ci contiendra la totalité des données qui seront converties en format csv. Il est nécessaire de préciser les chemins où seront écrits les fichiers JSON et csv :

```
# paths where the data will be written
json_path = 'data/'
tweets_file_path = json_path+'tweets.csv'
users_file_path = json_path+'users.csv'
```

Source : code de l'auteur

La méthode *get_data* permet ensuite d'appeler l'API désirée dans une boucle afin d'assigner toutes les données dans la variable *data_list*. La méthode *get_data* appelle également une autre méthode afin de sauvegarder les données de chaque appel vers l'API :

```
def get_data(self, ids, json_path):
    data_list = []
    counter = 0
    auth = requests_oauthlib.OAuth1(self.api_key, self.api_secret,
    self.access_token, self.access_token_secret)
    for x in ids:
        try:
            url =
            'https://api.twitter.com/1.1/statuses/show.json?id={}&include_entities=0'.format(x
            )
            print(url)
            data = self.export_with_url(auth, url, json_path,
            "original_tweet_{}.json".format(counter))
            data_list.append(data)
            counter += 1

        except Exception as e:
            print(str(e))
    return data_list
```

Source : code de l'auteur

Afin de créer un fichier csv contenant tous les tweets, nous appelons une deuxième méthode qui se chargera de lire les données de la variable *data_list*, et de les convertir dans le format csv afin que notre base de données puisse lire nos données :

```
def create_tweet_table(self, data_list, file_path):
    tweets = open(file_path, 'w', encoding='utf-8')
    header = '"id","text","date","number of retweets","user_id"'
    tweets.write(header + '\n')
    for data in data_list:
        user = data['user']
        id_str = data['id_str']
        text = data['text'].replace('\n', ' ').replace('\"', '')
        created = data['created_at']
        retweet = data['retweet_count']
        user_id = user['id_str']
        row_string =
        '"{}","{}","{}","{}","{}"\n'.format(id_str, text, created, retweet, user_id)
        tweets.write(row_string)
    tweets.close()
    print("!! FINISHED !! Generated tweets.csv")
```

Source : code de l'auteur

Afin de remplir la table « users » de notre base de données, la dernière méthode servira à collecter les données de l'utilisateur dans un deuxième fichier csv afin d'avoir l'auteur du tweet. Voici la méthode qui s'en charge :

```
def create_user_table(self, data_list, file_path):
    users = open(file_path, 'w', encoding='utf-8')
    header =
    '"id","name","location","description","followers_count","friends_count","listed_co
    unt","favourites_count","geo_enabled"'
    users.write(header + '\n')
    for data in data_list:
        user = data['user']
        id_str = user['id_str']
        name = user['name']
        location = user['location']
        description = user['description'].replace('\n', ' ')
        followers_count = user['followers_count']
        friends_count = user['friends_count']
        listed_count = user['listed_count']
        favourites_count = user['favourites_count']
        geo_enabled = user['geo_enabled']
        row_string =
        '"{}","{}","{}","{}","{}","{}","{}","{}"\n'.format(id_str, name, location, descriptio
        n, followers_count, friends_count, listed_count, favourites_count, geo_enabled)
        users.write(row_string)
    users.close()
    print("!! FINISHED !! Generated users.csv")
```

Source : code de l'auteur

Pour lancer les scripts, nous devons passer par le Terminal (Mac) afin de pouvoir taper la commande *python nomDuFichier.py*

3.3. Création d'une base de données

Maintenant que nous avons nos données dans le format csv, nous pouvons les importer dans une base de données. Cette base de données sera accessible depuis notre plateforme. Après avoir téléchargé MariaDB, nous pouvons y accéder en ligne de commande :

Figure 34 – Terminal : MariaDB

```
Last login: Sun Jul 7 18:25:27 on ttys002
(base) macbook-pro-retina-de-nicolas:Scripts Python Nico$ sudo mysql
[Password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 343
Server version: 10.3.14-MariaDB Homebrew

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> █
```

Source : capture d'écran de l'auteur

Après avoir lancé la commande `create database tweets`, nous pouvons y accéder en mettant `use tweets`. Nous pouvons ensuite créer les tables que nous voulons.

Figure 35 – Terminal : MariaDB création de tables

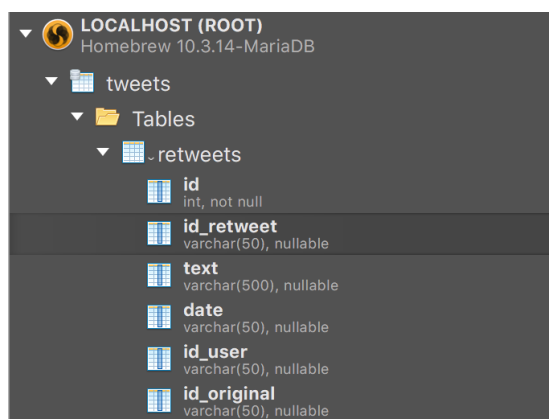
```
MariaDB [(none)]> use tweets
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
MariaDB [tweets]> CREATE TABLE retweets_test ( id INT NOT NULL AUTO_INCREMENT, i
d_retweet INT NOT NULL, text VARCHAR(500), date VARCHAR(50), id_user VARCHAR(50)
, id_original VARCHAR(50), PRIMARY KEY(id)); █
```

Source : capture d'écran de l'auteur

Dans le terminal, MariaDB comprend d'office un affichage d'une table. Cependant, comme nous aurons beaucoup de données, nous préférons avoir une interface graphique de notre base de données. Nous avons donc utilisé le programme SQLPro for MySQL et nous nous sommes connectés à notre base de données « tweets ». Nous avons pu ensuite importer nos fichiers csv dans les tables correspondantes. La Figure 36 montre notre base de données affichée dans le programme SQLPro for MySQL :

Figure 36 – Interface graphique de notre base de données



Source : capture d'écran de l'auteur

Comme il se peut qu'un utilisateur partage plusieurs tweets que nous avons recensés, la table « users » peut contenir des doublons. Par conséquent, avant d'importer notre fichier csv et afin de supprimer les doublons, nous avons exécuté la commande dans notre terminal :

```
$ sort -u users.csv -o users.csv
```

Quand bien même nous pensions les avoir supprimés, nous nous sommes aperçus qu'il y avait toujours des doublons dans la table « users ». Nous avons remarqué qu'un même utilisateur pouvait avoir des différences, par exemple pour le nombre de followers. En voici un exemple :

Figure 37 – Doublons de notre base de données

id	id_user	name	loca	description	followers	friend	listed	favourites
8612	971862548	eden* #NotreDameDeParis		Mal à ma France.en rés...	4352	3738	146	43861
2753	971862548	eden* #NotreDameDeParis		Mal à ma France.en rés...	4356	3744	147	43694

Source : capture d'écran de l'auteur

Nous avons donc effectué une requête SQL afin de supprimer les doublons en les éliminant de la colonne id_user.

```
DELETE
FROM users
WHERE id NOT IN
(
  SELECT MAX(id)
  FROM users
  GROUP BY id_user
)
```

Source : code de l'auteur

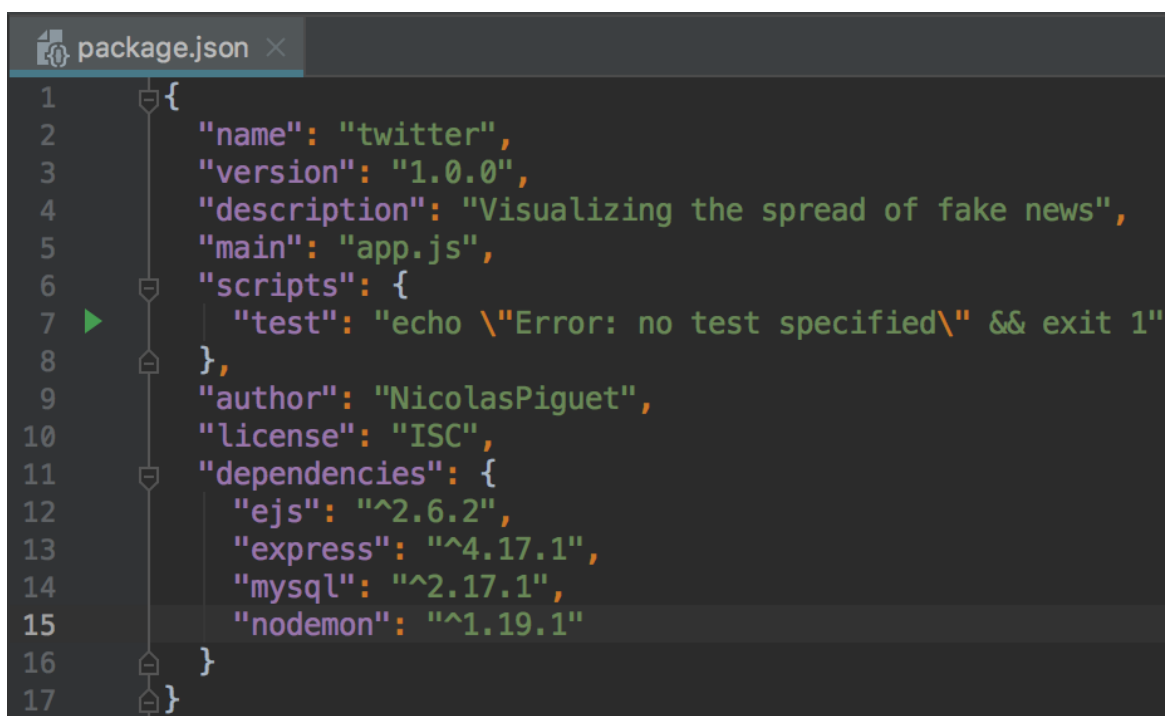
3.4. Implémentation

Les données étant accessibles grâce à notre base de données, nous pouvons créer la base de notre plateforme. Voici les principaux fichiers pour Node.js que nous avons :

3.4.1. Package.json

Ce fichier permet de lister les détails de notre application Node.js. Il permet également de pouvoir transmettre notre code à une autre personne sans que cette dernière doive installer les composants nécessaires (appelés dépendances) manuellement afin de lancer notre application. Voici notre fichier *package.json* :

Figure 38 – Contenu du fichier *package.json*



```
1 {  
2   "name": "twitter",  
3   "version": "1.0.0",  
4   "description": "Visualizing the spread of fake news",  
5   "main": "app.js",  
6   "scripts": {  
7     "test": "echo \"Error: no test specified\" && exit 1"  
8   },  
9   "author": "NicolasPiguet",  
10  "license": "ISC",  
11  "dependencies": {  
12    "ejs": "^2.6.2",  
13    "express": "^4.17.1",  
14    "mysql": "^2.17.1",  
15    "nodemon": "^1.19.1"  
16  }  
17 }
```

Source : capture d'écran de l'auteur

Nous pouvons voir que nous avons 4 dépendances (*dependencies*). Ces dépendances seront appelées dans les classes à l'aide de l'expression *require('...')* :

3.4.1.1. Embedded JavaScript templates (ejs)

Les vues ejs remplacent les fichiers *.html* classiques et ont l'avantage de pouvoir recevoir et afficher des données via les balises `<%= variable %>`. Ejs peut aussi exécuter du code JavaScript directement dans les vues à l'aide des balises `<% %>`, sans devoir mettre les balises `<script>`. Nous avons opté pour cette solution car notre plateforme a besoin de recevoir des valeurs. Par ailleurs, cette syntaxe rend le code plus lisible et donc plus simple.

Voici un exemple de code pour afficher une liste d'éléments sans utiliser les vues ejs :

```
<div>
  <p id="demo"></p>

  <script>
    var fruits = ["tweet 1", "tweet 4", "tweet 2"];
    fruits.forEach(myFunction);

    function myFunction(item, index) {
      document.getElementById("demo").innerHTML += index+1 + ": " + item +
" <br>";
    }
  </script>
</div>
```

Source : code de l'auteur

Et ci-dessous, pour comparer, le code nécessaire pour afficher le même résultat, mais en utilisant le module des vues ejs :

```
<div>
  <% var items = ["tweet 1", "tweet 4", "tweet 2"];
    for (var x in items){ %>
      <p> <%= x+': ' + items[x] %></p>
    <% } %>
</div>
```

Source : code de l'auteur

3.4.1.2. Express

Le framework express est un serveur très complet. Il existe un module plus basique nommé http, mais il est trop limité pour notre plateforme. Le module http peut transmettre des données, mais celles-ci ne peuvent pas être formatées.

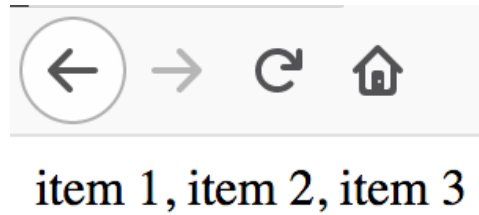
En revanche, express permet de simplifier les chemins d'accès des différentes vues et de leur transmettre des données. Ci-dessous un exemple de différentes requêtes possibles avec trois pages que nous pouvons afficher :

```
var http = require('http');
var fs = require('fs');
var server = http.createServer(function (req, res) {
  if(req.url === '/page1'){
    res.writeHead(200, {'Content-Type': 'text/html'});
    res.end("item 1, item 2, item 3");
  } else if(req.url === 'page2'){
    res.writeHead(200, {'Content-Type': 'text/html'});
    fs.createReadStream(__dirname+'/page2.html').pipe(res);
  } else if(req.url === 'page3'){
    res.writeHead(200, {'Content-Type': 'text/html'});
    fs.createReadStream(__dirname+'/page3.html').pipe(res);
  }
});
server.listen(3001);
```

Source : code de l'auteur

Voici la page affichée lorsque nous nous rendons à l'adresse : <http://localhost:3001/page1>

Figure 39 – Page html



Source : capture d'écran de l'auteur

Nous pouvons voir sur la Figure 39 qu'il n'y a pas de mise en forme des données, celles-ci sont envoyées brutes. Voici la même solution en utilisant express :

```
const express = require('express');
const app = express();
app.set('view engine', 'ejs');

app.get('/page1', function (req, res) {
  res.render('page1', {items: ["item 1", "item 2", "item 3"]});
});

app.get('/page2', function (req, res) {
  res.render('page2');
});

app.get('/page3', function (req, res) {
  res.render('page3');
});

app.listen(3001);
```

Source : code de l'auteur

Nous voyons que la structure du code est plus lisible. De plus, à l'aide des vues ejs, voici le résultat lorsque nous nous rendons à l'adresse suivante : <http://localhost:3001/page1>

Figure 40 – Rendu visuel avec express

Voici les données formatées en liste

0: item 1

1: item 2

2: item 3

Source : capture d'écran de l'auteur

3.4.1.3. Mysql

Cette dépendance est obligatoire pour accéder à notre base de données MariaDB. Nous avons donc créé un fichier *db.js* qui accèdera à celle-ci. Après avoir importé le module *mysql*, nous avons défini la variable *connection* en indiquant les coordonnées de la base de données que nous avons créée sur MariaDB :

```
const mysql = require('mysql');

var connection = mysql.createConnection({
  host: 'localhost',
  user: 'root',
  password: '',
  database: 'tweets'
});
```

Source : code de l'auteur

Puis, nous avons pu lancer la connexion. Nous avons trouvé judicieux d'afficher des messages ('*Error*' ou '*Connected to database*') dans la console afin d'avoir la confirmation que la connexion s'est faite avec succès :

```
connection.connect(function (error) {
  if(!error){
    console.log('Error');
  } else {
    console.log('Connected to database');
  }
});
```

Source : code de l'auteur

Node.js permet non seulement d'importer des modules (comme *mysql*, *ejs*, ...), mais aussi d'en créer et de les exporter. Cela rend le code plus modulaire et plus organisé. Nous avons donc exporté notre variable *connection* afin de pouvoir l'importer dans d'autres classes. Pour ce faire, il a fallu ajouter une ligne à la fin de notre fichier *db.js* :

```
module.exports = connection;
```

3.4.1.4. Nodemon

Nodemon, tout comme le mot-clé *node* présent de base dans Node.js permet de lancer notre fichier *app.js*. Cependant, l'avantage de nodemon est de nous éviter de redémarrer le serveur à chaque modification dans le code.

Nous pouvons le lancer à l'aide de la commande *nodemon app*. Ci-dessous, nous avons fait une capture d'écran de notre terminal lorsque nous avons changé un caractère dans notre code :

Figure 41 – Lancement de nodemon dans notre terminal

```
[nodemon] starting `node app.js`  
You are listening on port : 3000  
Connected to database  
[nodemon] restarting due to changes...  
[nodemon] starting `node app.js`  
You are listening on port : 3000  
Connected to database  
█
```

Source : capture d'écran de l'auteur

3.4.2. Classe principale

Le fichier *app.js* sera lancé par nodemon (ou node si nous n'utilisons pas nodemon) et qui servira de serveur. Cette classe peut être vue comme le « main » d'un code Java. En effet, notre fichier *app.js* se chargera d'appeler les composants requis afin de faire fonctionner la plateforme. Voici les points importants que nous avons configurés :

- Premièrement, nous avons importé plusieurs modules dont nous aurons besoin. Toutes les méthodes incluses avec le module express pourront être appelées avec la variable *app*. La variable *port* contient le numéro de port sur lequel notre application sera accessible. Puis, les deux dernières variables sont les contrôleurs nécessaires à notre application. Nous les expliquerons plus en détail après.

```
const express = require('express');  
const app = express();  
const port = 3000;  
const searchController = require('./controllers/searchController');  
const resultController = require('./controllers/resultController');
```

Source : code de l'auteur

- Comme décrit précédemment, notre application n'utilise pas les vues classiques html, mais les vues ejs. La ligne ci-dessous permet de l'indiquer à notre application. Sans précision supplémentaire, le dossier dans lequel l'application s'attend à avoir les vues ejs doit être nommé *views*.

```
app.set('view engine', 'ejs');
```

- Nous sommes obligés d'utiliser le middleware d'express afin d'inclure les fichiers complémentaires à notre application. Nous avons donc créé un dossier *assets* dans lequel nous y avons mis nos fichiers de style (css) et nos images.

```
app.use('/assets', express.static('assets'));
```

- Après avoir configuré notre serveur express, nous pouvons lancer les contrôleurs afin qu'ils se chargent de répondre aux requêtes de l'utilisateur. Nous pouvons les lancer en écrivant ces lignes.

```
searchController(app);
resultController(app);
```

Source : code de l'auteur

- Pour finir, notre plateforme sera accessible sur un port que nous devons définir. Nous pourrons ensuite lancer notre serveur grâce à nodemon.

```
app.listen(port);
```

3.4.3. Contrôleurs

Les contrôleurs permettent de gérer les requêtes de l'utilisateur et d'exécuter du code. Notre solution a été conçue avec deux contrôleurs car nous avons deux vues (page html). Nous aurions pu mettre les contrôleurs dans le fichier *app.js*, mais par souci de lisibilité et par convention, nous avons décidé d'assigner un contrôleur par vue. Voici un exemple basique de code d'un contrôleur et d'une vue ejs :

```
app.get('/vue', function
(req, res) {
  res.render('vue');
});
```

Source : code de l'auteur

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>page1</title>
</head>
<body>
  <p>Hello world!</p>
</body>
</html>
```

Source : code de l'auteur

Après avoir démarré *app.js*, la vue sera générée et affichée à l'adresse : <http://localhost:3000/vue>

Figure 42 – Résultat dans notre navigateur



Source : capture d'écran de l'auteur

Nos deux contrôleurs ont plusieurs éléments qui se retrouvent dans leur structure de code.

- Si un contrôleur doit avoir accès à notre base de données afin de pouvoir y effectuer des requêtes, nous devons importer le module que nous avons créé. Dans nos deux contrôleurs, nous avons donc dû importer le module *db.js* qui se charge d'accéder à notre base de données.

```
const db = require('../db');
```

- Nos contrôleurs doivent être exportables, car nous les appelons dans notre fichier *app.js*.

```
module.exports = function (app) {
```

- Nous allons prendre comme exemple un de nos contrôleurs nommé *searchController*. Celui-ci se charge d'afficher la page *index.ejs* ainsi que toutes les données nécessaires. Le code sera exécuté lorsque l'utilisateur se rendra à l'adresse suivante : <http://localhost:3000/>. Nous l'avons donc configuré de la sorte.

```
app.get('/', function (req, res) {
```

- Nous avons fait une requête SQL afin de récupérer certaines informations de la base de données. Dans *searchController*, nous avons récupéré tous les tweets de la table « tweets » et nous avons stocké le résultat de la requête dans la variable *data*.

```
db.query("SELECT * FROM tweets", function (error, rows, fields) {
  if(!error){
    console.log('Error in the query');
  } else {
    console.log('Successful query');
    var data = rows;
  }
});
```

Source : code de l'auteur

- Une fois récupérées, nous devons transmettre les données à la vue et afficher cette dernière. Pour transmettre des données à la vue, le serveur express propose une solution très simple. Il suffit de mettre le nom qui permettra d'identifier les données ainsi que les données. Dans notre cas, la vue que nous voulons afficher se nomme *index.ejs* et les données sont dans la variable *data*.

```
res.render('index', {all: data});
```

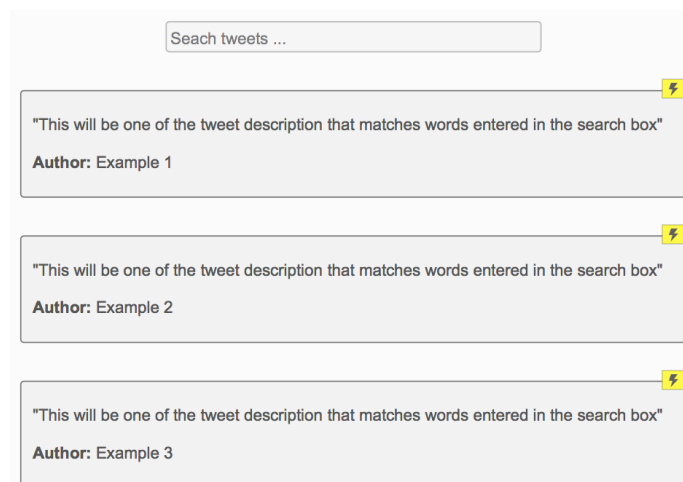
Depuis la vue *index.ejs*, nous pourrons donc accéder aux données à l'aide de ce code : `<%= data %>`

3.5. Proposition d'interface

Maintenant que nous avons une base, nous pouvons penser à l'affichage des données. Nous avons donc conçu une simulation d'interface sur le logiciel Axure.

La première page ci-dessous affichera la description des tweets qui correspondent aux mots-clés entrés dans la barre de recherche. L'auteur du tweet sera placé également en dessous de la description du tweet.

Figure 43 – Page de recherche

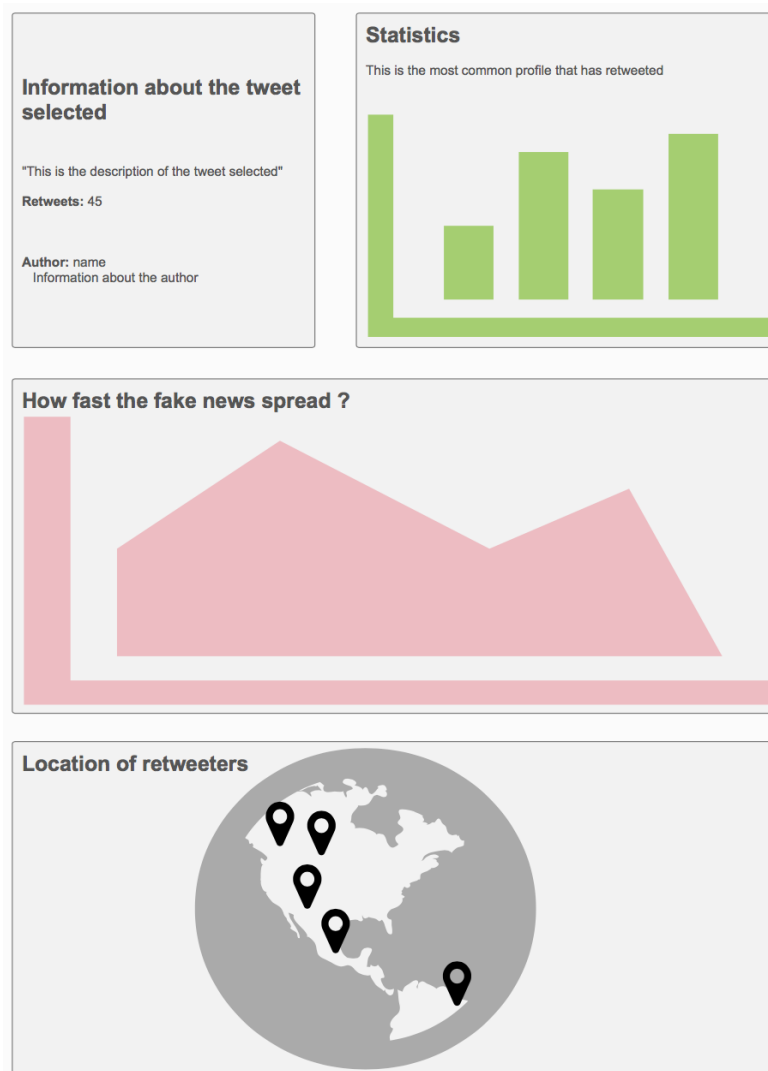


Source : capture d'écran de l'auteur (sur Axure)

La deuxième page affichera toutes les informations relatives au tweet sélectionné. Nous détaillerons les données que nous avons récupérées sur l'auteur du tweet. Puis, nous rassemblerons plusieurs données afin d'établir des statistiques.

Nous essaierons d'établir un profil type des utilisateurs ayant partagé la fausse information. Un graphique sera généré par rapport à la quantité de retweets dans une durée déterminée, puis un graphique montrera également une carte afin de situer où les utilisateurs résident.

Figure 44 – Page de résultats



Source : capture d'écran de l'auteur (sur Axure)

3.6. Champ de recherche

Étant donné le nombre très élevé de données, nous avons décidé de mettre en place un champ de recherche afin de trouver des tweets. Deux possibilités se sont offertes à nous :

3.6.1. Requêtes SQL

Une des deux options envisagées a été de chercher les tweets à l'aide de requêtes SQL. Nous avons écrit une requête et adapté le paramètre de la requête par rapport à ce que l'utilisateur entre comme mot-clé dans le champ de recherche. Ci-dessous la requête quand nous entrons le mot-clé *Macron*:

```
SELECT text FROM tweets WHERE text LIKE '%Macron%';
```

La requête retourne de nombreux résultats, et tous contiennent le mot-clé que nous avons entré. Lorsque nous entrons un deuxième mot-clé, la requête se comporte correctement. Voici ci-dessous la même requête que la précédente mais avec un mot-clé supplémentaire :

```
SELECT text FROM tweets WHERE text LIKE '%30 000%' AND text LIKE '%Macron%';
```

Afin de simplifier l'implémentation de la recherche, il faudrait rajouter une condition dans la requête à chaque fois que l'utilisateur met un espace entre les mots-clés. La requête serait par conséquent écrite différemment, plus longue, mais plus facile à mettre en place. Les résultats resteraient identiques :

```
SELECT text FROM tweets WHERE text LIKE '%30%' AND text LIKE '%000%' AND text LIKE '%Macron%';
```

Voici les valeurs qui nous sont renvoyées :

- **Macron** annonce vouloir supprimer **30 000** emplois d'infirmière dans les hôpitaux Français. A Retweeter <https://t.co/OfkpV2N1H3>
- **Macron** annonce vouloir supprimer **30 000** emplois d'infirmière dans les hôpitaux Français. A Retweeter <https://t.co/mQR7xAwPXS>
- **Macron** annonce vouloir supprimer **30 000** emplois d'infirmière dans les hôpitaux Français <https://t.co/SYRegMo1VM>

Malheureusement, si l'utilisateur rentre comme mot-clé le chiffre 30000 (sans espace), la requête n'affiche aucun résultat.

Nous devons donc trouver un moyen qui permette de renvoyer des résultats, quand bien même l'utilisateur ne saisit pas exactement le mot-clé écrit comme dans la base de données.

3.6.2. Librairie

Nous nous sommes penchés vers une librairie permettant de faire de la recherche de données. Nous avons choisi Fuse.js car elle est très rapide et permet, même en cas d'erreurs de saisie (espace, accents, ...), d'obtenir des résultats. De plus, elle permet de chercher sur une ou plusieurs colonnes précises, comme les requêtes SQL ci-dessus.

La librairie Fuse.js n'est pas parfaite, car elle nous oblige à récupérer toutes les données de la table « tweets » afin d'effectuer la recherche. Nous avons donc comparé le temps de réponses pour chacune des requêtes ci-dessous :

```
SELECT text FROM tweets WHERE text LIKE '%Macron%';
```

44 résultats ~0.25s

```
SELECT text FROM tweets;
```

532 résultats ~0.40s

Nous pouvons voir que la recherche avec un paramètre prend moins de temps. Nous avons été plus loin et avons décidé de comparer le temps de recherche entre SQL et Fuse.js. Nous avons entré la lettre 's' comme critère de recherche afin qu'il y ait beaucoup de résultats.

Ce faisant, nous avons observé qu'il y avait une nette différence entre les requêtes SQL et l'utilisation de la librairie Fuse.js. En effet, la requête SQL a pris environ **0.42s** pour retourner tous les résultats de notre base de données. Concernant Fuse.js, la librairie a pris 9ms, ce qui correspond à **0.009s**. Même si la récupération de la totalité des tweets prend davantage de temps, à chaque recherche nous gagnons un temps considérable.

Nous avons donc pris la décision d'utiliser la librairie Fuse.js. Nous avons fait en sorte qu'à chaque lettre du clavier pressée, nous fassions une recherche avec les mots-clés entrés par l'utilisateur. Nous appelons cette méthode l'autocomplétion. Elle consiste à proposer des résultats avant même d'écrire le mot en entier. Cela rend la recherche plus facile à utiliser. Voici le code JavaScript que nous avons développé pour y parvenir.

```
$(document).ready(function () {
  // Every time user write on the search field
  $('#search').keyup(function () {

    // Stock the value that is currently in search field
    var val = $('#search').val();

    if(val != '') {

      // Get everything from the tweets table in the database
      var data = <%- JSON.stringify(all)%>;

      // If the user has checked a checkbox, change the variable data
      if ($('#fake').is(":checked") && !$('#true').is(":checked"))
      {
        data = <%- JSON.stringify(fakes)%>;
      } else if ($('#true').is(":checked") && !$('#fake').is(":checked"))
      {
        data = <%- JSON.stringify(reals)%>;
      }

      // Tell fuse to search only in the text column
      var options = {keys: ["text"]}
      var fuse = new Fuse(data, options)

      // Make the search and stock the result
      var searchResult = fuse.search(val);
```

Source : code de l'auteur

À chaque touche du clavier pressée par l'utilisateur, les résultats varient. Tous ces résultats sont assignés à la variable `searchResult`. La prochaine étape dans le code est d'afficher tous

les tweets que Fuse.js a trouvés. Nous avons créé un `<div>` pour chaque résultat. Voici le code que nous avons écrit afin d'afficher tous les tweets trouvés :

```

// For every result, create a div to display the result, and make it
clickable
if(searchResult.length > 0)
{
  $('#results').empty();
  for(var i=0; i<searchResult.length; i++){
    let link = '/tweet/'+searchResult[i].id_tweet;
    b = document.createElement("DIV");

    b.setAttribute("class", "shadow-sm p-3 mb-5 rounded");
    b.setAttribute("id", "res");

    // Change background color depending if news is fake or real
    if(searchResult[i].fake == 1){
      b.setAttribute("style", "background-color: rgba(219, 0, 0,
0.2)");
    } else {
      b.setAttribute("style", "background-color: rgba(31, 198,
249, 0.2)");
    }

    b.addEventListener("click", function () {
      location.href=link;
    });

    var title = document.createElement("h5");
    title.innerHTML=searchResult[i].text;

    var date = document.createElement("p");
    var niceDate = new
Date(searchResult[i].date).toUTCString().slice(0,25);
    date.innerHTML=("date: ").bold()+niceDate;

    var retweets = document.createElement("p");
    retweets.innerHTML=("retweets:
").bold()+searchResult[i].nb_retweet;

    b.append(title);
    b.append(date);
    b.append(retweets);

    $('#results').append(b);
  }
}
else
{
  $('#results').empty();
}
});
});

```

Source : code de l'auteur

3.7. Tweet sélectionné

Sur une deuxième page ejs, nous avons affiché les informations du tweet sélectionné par l'utilisateur. Voici les informations que nous mettons :

Figure 45 – Informations sur le tweet sélectionné

Tweet's information

"Angers : à l'école publique, ils ont appris à mon fils de 5 ans « Bismillah au nom d'Allah » !... <https://t.co/NRTdImv2W6>"

Retweets 43

Date "Fri Apr 13 06:15:00 +0000 2018"

Author "Riposte Laïque"

Description "Nous entendons alerter l'opinion sur ce péril que représente l'islam, et créer les conditions d'une riposte laïque massive"

Location ""

Followers 11772

Friends 181

Retweeters that have >1000 followers 43.90 %

Source : capture d'écran de l'auteur

Nous obtenons tout d'abord des informations générales sur le tweet. Puis, nous précisons quel utilisateur a posté ce tweet et pour finir nous donnons le nombre d'utilisateurs ayant plus de 1000 abonnés et qui ont partagé le tweet.

3.8. Statistiques

Nous avons aussi trouvé intéressant de dresser quelques statistiques. Nous avons créé, entre autres, un graphique avec la librairie Chart.js. Ce dernier indique le nombre de retweets par rapport à un nombre de jours. L'utilisateur doit rentrer au minimum un jour afin de pouvoir afficher la proportion de retweets qu'il y a eu durant cette période. Chaque fois que le nombre de jours est changé, le nombre de retweets est recalculé.

Voici le code que nous avons écrit afin de calculer le nombre de retweets entre deux dates :

```
// source : https://stackoverflow.com/questions/563406/add-days-to-javascript-date
Date.prototype.addDays = function(days) {
  var date = new Date(this.valueOf());
  date.setDate(date.getDate() + days);
  return date;
}
let counter = 0;
// Get all dates of the retweets tweet selected
let alldates = <%- JSON.stringify(dates)%>;
// Get the date of the original tweet, and set it as a start date
let dateStart = new Date(<%- JSON.stringify(original[0].date)%>);
// Every time the user change the number of days on the stat input
$(document).ready(function () {
  $('#stat').change(function () {
    var nb = $('#stat').val();
    let dateEnd = dateStart.addDays(parseInt(nb));
    counter = 0;
    // Count how many retweets there is before dateEnd
    for (var i=0; i<alldates.length; i++){
      var start = new Date(alldates[i].start);
      if (start <= dateEnd) {
        counter+=1;
      }
    }
  })
})
```

Source : code de l'auteur

Après avoir calculé le nombre de retweets pendant une durée déterminée, nous avons pu créer un graphique. Nous avons choisi un graphique de type « *doughnut* » car c'est celui qui représente le mieux les données que nous comparons.

Nous avons ensuite précisé les labels afin d'identifier les données. Et pour finir, nous avons inséré les données calculées par le code que nous avons décrit ci-dessus. Voici la suite du code ci-dessus qui permet de créer le graphique désiré :

```
// Create the chart to visualize it
var ctx = document.getElementById('myChart');
var myChart = new Chart(ctx, {
  type: 'doughnut',
  data: {
    labels: ['Retweets within the first '+nb+' day(s)', 'Other'],
    datasets: [{
      data: [counter, alldates.length-counter],
      backgroundColor: [
        'rgba(249, 191, 31, 0.5)'
      ]
    }]
  },
});
})
```

Source : code de l'auteur

3.9. Graphique chronologique

Nous avons aussi implémenté un graphique afin de visualiser la densité des retweets dans le temps. Vis.js est une librairie qui permet de créer des graphiques, dont des graphiques chronologiques. Bien que cette librairie soit peu connue, elle est relativement bien documentée et prend en charge le format de nos données, sans que nous ayons à les convertir.

Voici le code que nous avons ajouté à notre projet afin d'insérer ce graphique :

```
var container = document.getElementById('visualization');

// Get all retweets date
let data = <%- JSON.stringify(dates)%>;

// Add the date of the tweet with a label "original"
data.push({id: 0, content: 'original', start: <%-
JSON.stringify(original[0].date)%>});
var dates = new vis.DataSet(data);

// Configuration for the Timeline : return no local time but utc (avoid adding 2
hours)
var options = {moment: function(date) {
    return vis.moment(date).utc();
}};

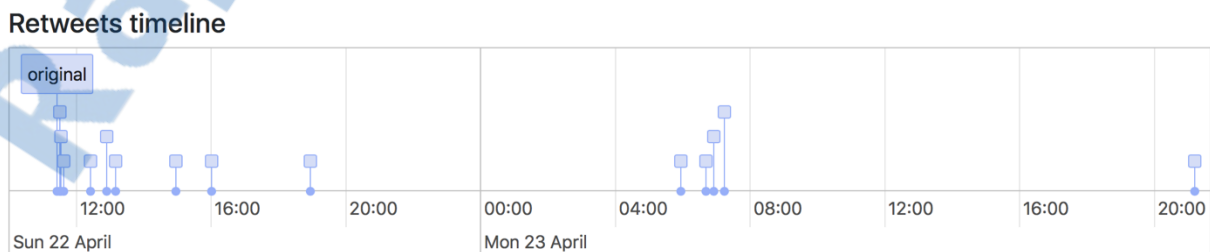
// Create a Timeline
var timeline = new vis.Timeline(container, dates, options);
```

Source : code de l'auteur

De base, nous pensions uniquement ajouter les dates des retweets, mais nous avons trouvé judicieux d'inclure la date du tweet initial avec le label « original ». En effet, c'est plus pratique d'avoir également cette information sur le graphique que de devoir consulter les informations du tweet.

Voici un exemple d'un tweet qui a été partagé 14 fois :

Figure 46 – Graphique chronologique



Source : capture d'écran de l'auteur

Nous pouvons voir que ce graphique est plus précis, qu'il nous donne plus d'informations que celui que nous avons décrit plus haut. En effet, le graphique chronologique permet de voir dans le temps la réaction des utilisateurs.

3.10. Graphique géographique

Lorsque nous avons conçu la proposition d'interface, nous avons prévu d'y inclure un graphique géographique. Twitter nous permet de récupérer le lieu des profils des utilisateurs. Cependant, Twitter ne met aucune contrainte et n'oblige pas ses utilisateurs à préciser leur situation géographique. Nous nous sommes aperçus qu'il manquait la localisation de beaucoup d'utilisateurs. Et quant aux utilisateurs ayant précisés leur localisation, certains ont mis un texte qui ne permet pas de les situer sur la carte.

Actuellement, dans notre base de données, nous avons plus de 12'000 utilisateurs ayant partagé au moins un tweet. Parmi tous ces utilisateurs, plus de 4500 n'ont pas ajouté leur localisation. Plus de 800 utilisateurs ont uniquement mis « France » ou « Europe » comme localisation, ce qui n'est pas assez précis.

Certains utilisateurs ont même mis un texte (« Dans une autre dimension », « dans ma litière », « échouée... ») qui n'a rien à voir avec des coordonnées géographiques.

Étant donné qu'il manquera en tout cas la moitié des données, nous avons pris la décision de ne pas créer un graphique géographique.

4. Analyses des résultats

Les données que nous avons téléchargées nous ont permis d'obtenir quelques résultats. Nous nous sommes aidés de notre base de données ainsi que de notre plateforme pour fournir deux types d'analyse.

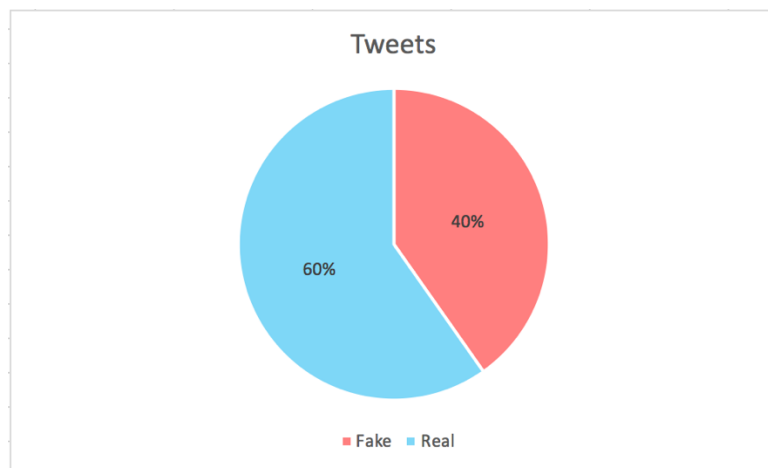
4.1. Analyse quantitative

Pour cette analyse, nous avons interrogé notre base de données. Nous nous sommes basés sur toutes les données que nous avons téléchargées.

Notre base de données contient actuellement **532 tweets** écrits par **155 auteurs** et **18'165 retweets** partagés par **12'253 retweeters**. Nous avons inséré dans la base de données deux jeux de données : un jeu de fausses nouvelles ainsi qu'un jeu de nouvelles véridiques.

Parmi toutes les données que nous avons dans la table « tweets », nous pouvons voir que nous avons 40% de fausses nouvelles. Nous avons moins de fausses nouvelles car il est plus difficile d'en trouver comme nous l'écrivions dans la présentation du problème.

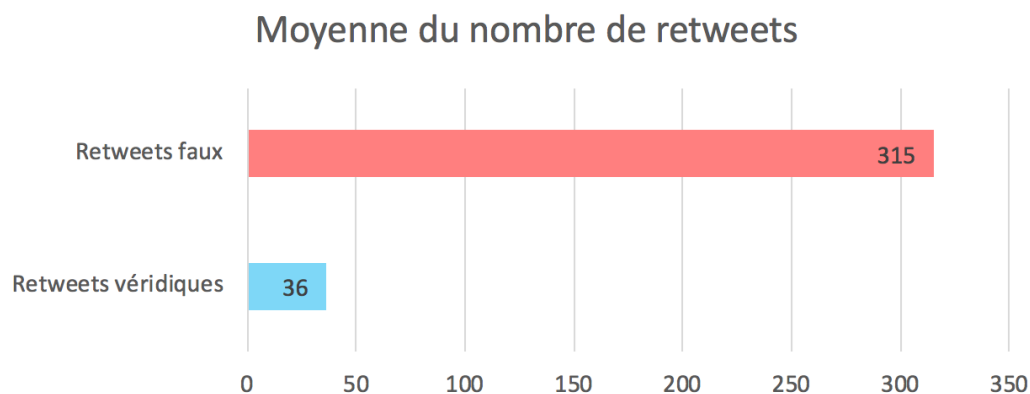
Figure 47 – Répartition des données



Source : données de l'auteur

Sur la Figure 48 de la page suivante, nous comparons l'échelle (le nombre de personnes qui réagissent) d'une fausse nouvelle à une nouvelle véridique. Si nous calculons la moyenne du nombre de retweets, nous observons qu'il y a, en moyenne, davantage de retweets lorsqu'une nouvelle est fausse.

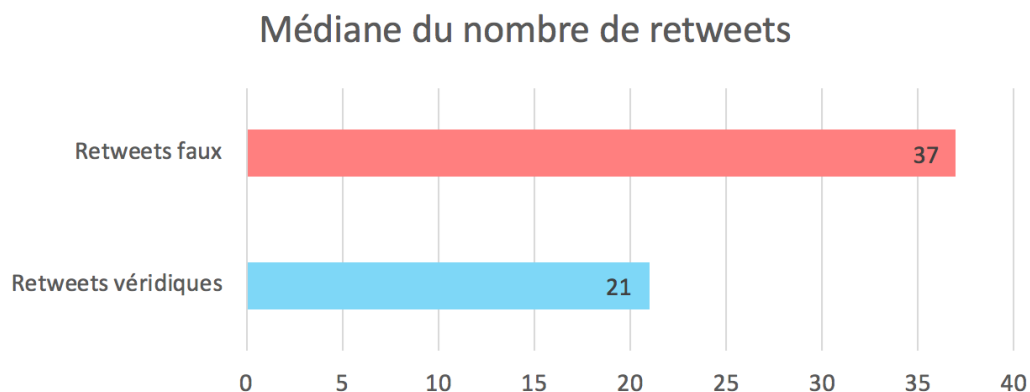
Figure 48 – Moyenne du nombre de retweets



Source : données de l'auteur

De plus, pour compléter notre résultat ci-dessus, nous avons également calculé la médiane du nombre de retweets de nos deux jeux de nouvelles. La médiane est le milieu du jeu de données, 50% des données sont supérieures à la médiane et 50% inférieures. Dans notre cas, la médiane illustre mieux nos données que la moyenne, car nous avons des valeurs très disparates. En effet, certaines fausses informations ont beaucoup plus de retweets (>1000) que d'autres. La médiane permet donc « d'exclure » ces extrêmes.

Figure 49 – Médiane du nombre de retweets



Source : données de l'auteur

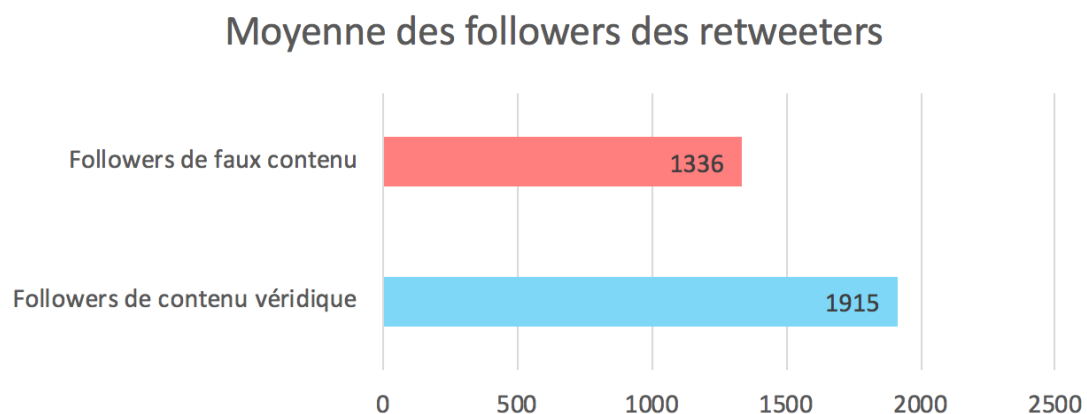
Nous voyons qu'il y a une grande différence entre la médiane et la moyenne du jeu de données de fausses informations. Nous constatons que pour nos données, le nombre de retweets d'une fausse nouvelle n'est pas stable, il dépend sûrement de facteurs externes comme la réputation de son auteur, les preuves (liens vers une source, photos), etc. En revanche, la moyenne de retweets d'une nouvelle véridique est constante.

Nous verrons dans l'analyse qualitative que la plupart des fausses nouvelles présente le sujet d'une manière choquante ou subjective. Nous pensons que c'est une des raisons qui justifie le nombre de partages plus élevé.

Nous nous pencherons ci-après sur l'impact (l'échelle) que peut avoir une fausse nouvelle. Nous ne pouvons pas établir de statistiques sur la moyenne des followers des auteurs de nos données car nous n'avons qu'une seule source (LeMonde) pour notre jeu de données véridiques.

Nous avons donc calculé la moyenne des followers des utilisateurs qui ont partagé les nouvelles (retweeters). Ces résultats nous donneront le nombre potentiel d'utilisateurs qui ont lu le tweet.

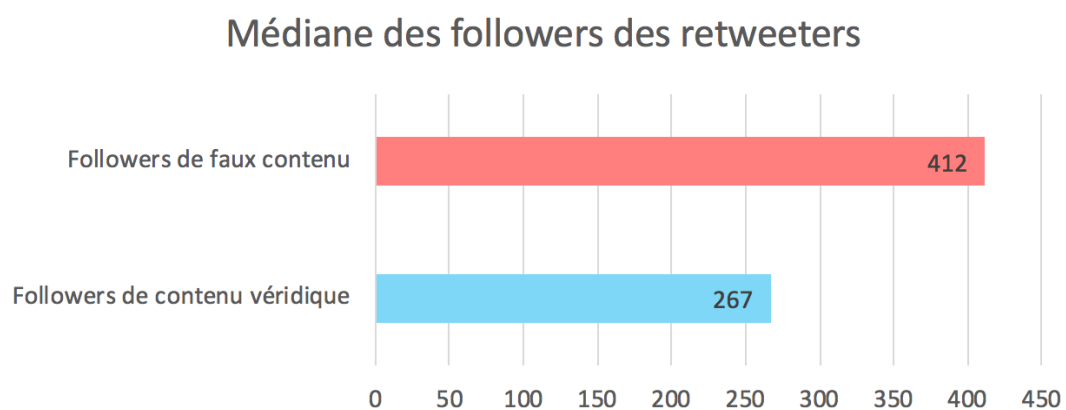
Figure 50 – Moyenne des followers des retweeters



Source : données de l'auteur

Nous pouvons voir que les utilisateurs qui ont partagé des informations véridiques ont davantage d'abonnés. Afin de vérifier la pertinence du calcul de moyenne, nous allons établir la médiane avec les mêmes données que la Figure 50 ci-dessus.

Figure 51 – Médiane des followers des retweeters

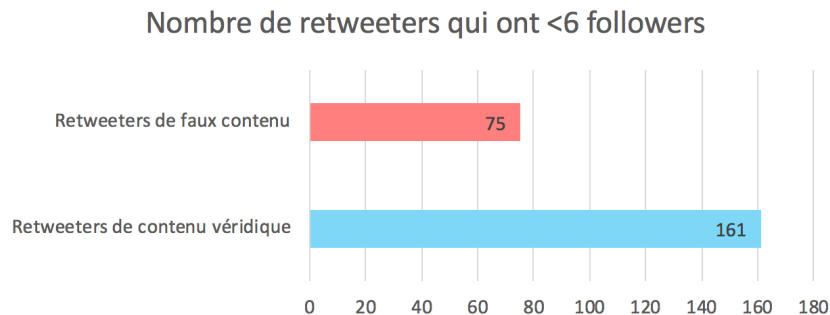


Source : données de l'auteur

En calculant la médiane des followers des retweeters, nous nous sommes aperçus qu'il y avait à nouveau des « extrêmes ». Nous avons donc cherché à savoir pourquoi nous obtenions de telles différences.

Nous avons cherché à savoir combien de retweeters avaient strictement moins de six followers. Il s'est avéré que les retweeters de nouvelles véridiques en avaient environ deux fois plus.

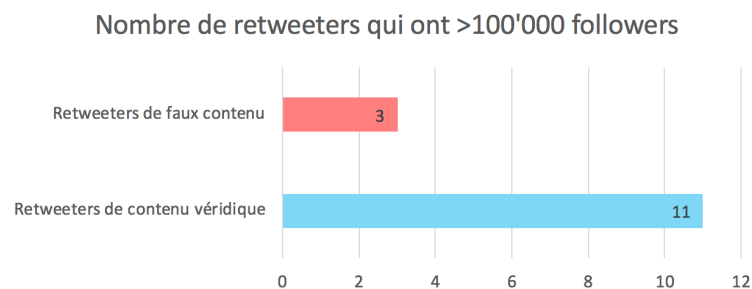
Figure 52 – Retweeters qui ont moins de six followers



Source : données de l'auteur

Pour aller plus loin, nous avons également calculé, pour chacun des jeux de données, combien de retweeters avaient plus de 100'000 followers. Il se trouve, qu'à nouveau, les retweeters de nouvelles véridiques en avaient presque quatre fois plus.

Figure 53 – Retweeters qui ont plus de 100'000 followers



Source : données de l'auteur

Nous nous sommes intéressés aux onze utilisateurs qui ont diffusé une nouvelle véridique. Il s'est avéré que beaucoup étaient des personnes connues, comme Anne Hidalgo (Maire de Paris), Greenpeace France, Eduardo Rothe (homme politique vénézuélien), Ezio Mauro (Directeur du journal italien *La Repubblica*) ou encore Oxfam France (ONG internationale).

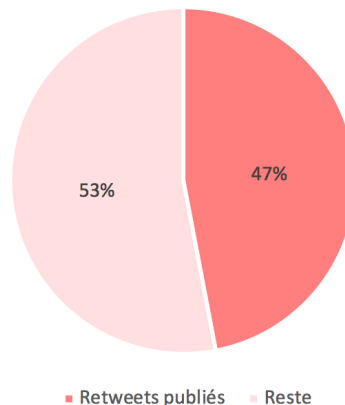
Nous avons recalculé la moyenne des followers des retweeters en ne prenant que ceux qui avaient moins de 100'000 followers afin d'ignorer les retweeters connus. Les retweeters d'informations véridiques ont en moyenne 1290 followers et pour les retweeters de fausses informations, ceux-ci ont en moyenne 1177 followers.

Nous pouvons constater que l'échelle de nouvelles fausses ou véridiques est plus ou moins le même. Cependant, lorsque l'information est fausse, l'utilisateur a tendance à plus la partager.

La dernière comparaison que nous avons faite a été sur la rapidité de propagation des nouvelles. Nous avons donc fait une moyenne du nombre de retweets pendant les premières 24 heures. Nous avons commencé avec le jeu de données de fausses nouvelles et il en est ressorti que moins de la moitié des retweets avaient été publiés pendant les premières 24 heures.

Figure 54 – Retweets faux publiés en 24h

Retweets faux publiés les premières 24h

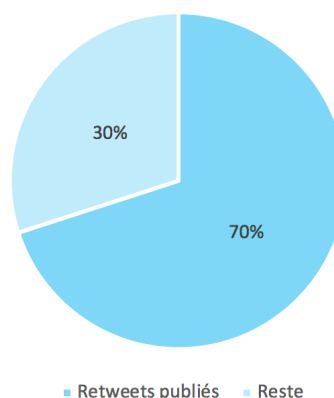


Source : données de l'auteur

Après avoir changé la requête SQL, nous avons pu comparer le résultat obtenu ci-dessus, de la même manière, avec le jeu de données de nouvelles véridiques. Nous avons constaté que 70% des retweets sont émis durant les premières 24 heures.

Figure 55 – Retweets véridiques publiés en 24h

Retweets véridiques publiés les premières 24h



Source : données de l'auteur

En voyant ces résultats, nous concluons que les tweets relatant des informations véridiques sont diffusés plus rapidement que les autres. Nous pensons que cela est dû à la confiance que portent les retweeters à l'auteur. Même si nous avons vu que les utilisateurs ont tendance à moins partager de nouvelles véridiques, celle-ci sont répandues plus vite.

4.2. Analyse qualitative

Cette deuxième analyse a permis de sélectionner et de comparer un choix de données. Nous avons donc sélectionné deux informations avec les mêmes critères afin de les comparer en nous aidant de notre plateforme.

Nous avons décidé, dans un premier temps, de comparer le nombre de retweets d'un tweet faux et d'un tweet véridique en prenant le même sujet (Mondial 2018 de football). Le tweet de couleur rouge est faux et le tweet bleu est véridique. Ci-dessous les tweets que nous avons choisis :

Figure 56 – Tweet faux (sujet : Mondial 2018)

Mondial 2018 : Les russes ont accueilli les pays musulmans avec des hotesses voilées par respect a leur religion ...
<https://t.co/NKjw8Blr77>
 date: Wed, 13 Jun 2018 12:03:27
 retweets: 9381

Source : capture d'écran de l'auteur

Figure 57 – Tweet véridique (sujet : Mondial 2018)

Pourquoi le football est-il un sport si imprévisible ? Cette vidéo a été publiée l'an dernier pour le mondial 2018....
<https://t.co/mviWWIMS5v>
 date: Fri, 07 Jun 2019 12:28:24
 retweets: 6

Source : capture d'écran de l'auteur

Il est fort probable qu'en raison de son sujet sensible (la religion) et de son ton affirmatif, cette fausse nouvelle a suscité de nombreuses réactions. Par contre, le tweet véridique ne contenant aucune polémique est bien moins partagé.

Le fait que nous observons une propagation plus rapide de fausses nouvelles est révélateur de notre société, friande de sensationnalisme. Les plateformes sont des lieux où la provocation et l'exagération prennent le pas sur la réflexion et la remise en question. Tout doit aller vite pourvu que l'utilisateur attire et garde l'attention du public. D'une manière générale, le retweet d'une fausse nouvelle se prête moins à une réflexion de fond. Elle répond aux besoins d'immédiateté, d'être au courant, de faire partie d'une communauté au fait de l'actualité. L'important n'étant pas que la nouvelle soit véridique ou fausse, mais de participer à cette foire aux informations, pléthoriques.

Pour continuer notre analyse, nous avons pris un deuxième sujet en exemple pour voir si la différence de nombre de retweets était aussi considérable que pour le premier sujet (Mondial 2018). Nous avons choisi deux tweets concernant Marine Le Pen, présidente du Front national, parti d'extrême droite en France. Le premier concerne un dessin animé où le personnage principal porte un foulard et le deuxième, plus politique, annonce à la mise en examen de Marine Le Pen.

Figure 58 – Tweet faux (sujet : Marine Le Pen)

Oui Marine, elle est Russe. Comme n'importe quelle paysanne Russe, Masha porte un foulard. Et toi tu es inculte. No...
<https://t.co/cyaTh956OZ>
date: Sun, 26 Feb 2017 14:42:00
retweets: 1720

Source : capture d'écran de l'auteur

Figure 59 – Tweet véridique (sujet : Marine Le Pen)

Mise en examen prochaine de Marine Le Pen pour avoir dévoilé un document judiciaire : la leader d'extrême droite s'...
<https://t.co/4d78sxKXny>
date: Thu, 06 Jun 2019 07:03:55
retweets: 48

Source : capture d'écran de l'auteur

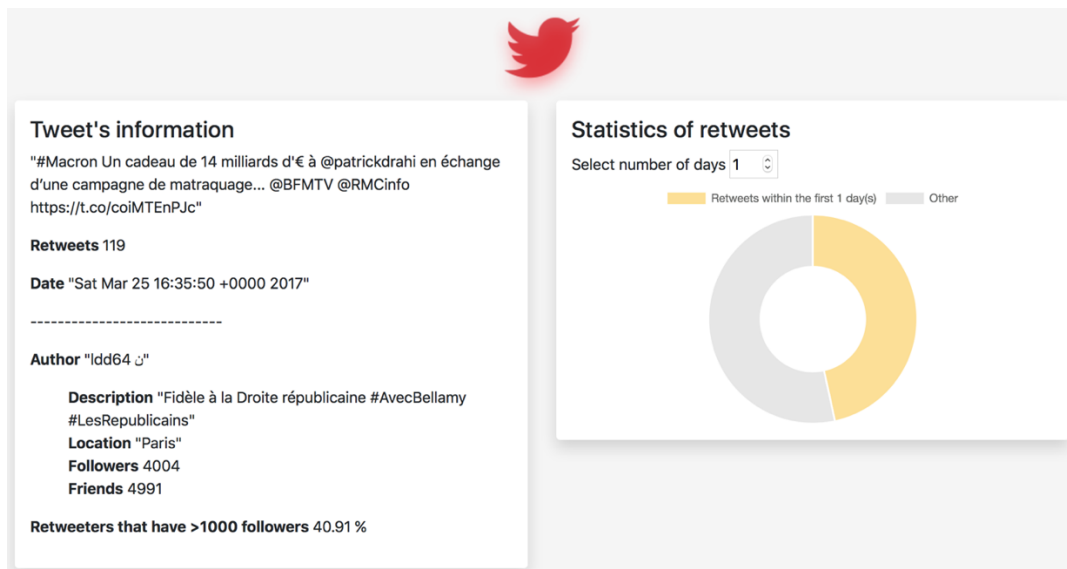
Le tweet en rouge, répertorié comme faux par le journal LeMonde, a été beaucoup plus partagé que le tweet véridique, en bleu. Nous avons cherché à savoir pourquoi il y avait, à nouveau, une telle différence. Il s'est avéré que l'auteur du tweet rouge, si l'on se réfère à son profil, est un utilisateur qui n'approuve pas du tout le FN (parti de Marine Le Pen). Nous pouvons assumer que la plupart de ses followers (plus de 10'000) sont du même avis politique.

Le tweet bleu a donc été publié par notre seule source de nouvelles véridiques, le journal LeMonde.fr. Quand bien même ce journal est réputé pour être de gauche, et qu'il dénonce une erreur de Marine Le Pen, son tweet n'a eu que peu de retweets.

Nous pensons que le caractère moqueur du message de l'utilisateur anti FN a joué un rôle primordial quant au nombre élevé de retweets.

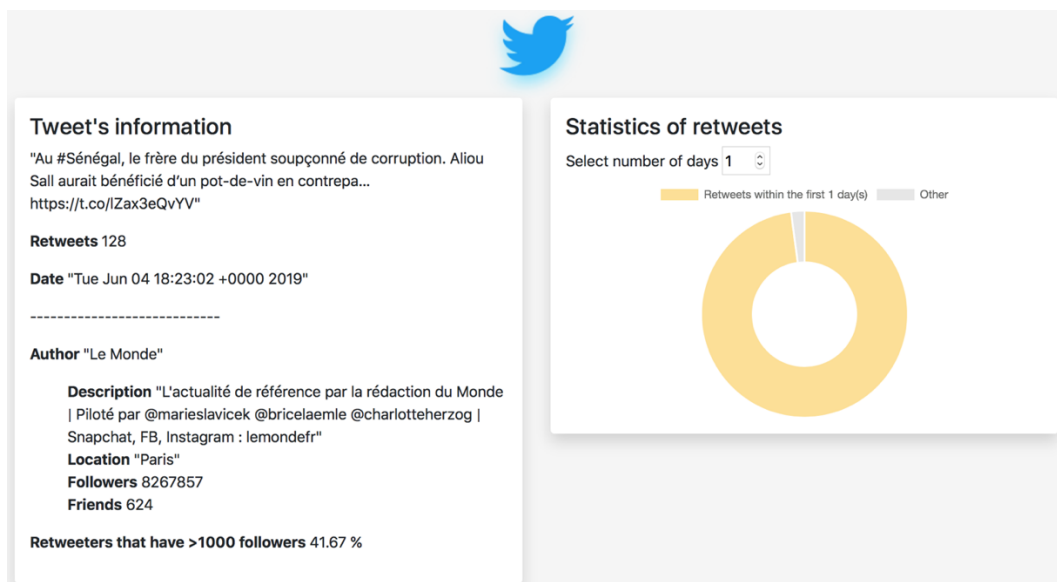
Afin de compléter cette analyse qualitative, nous allons également comparer deux tweets, un véridique et un faux, pour voir leur rapidité de propagation. Nous avons choisi ces deux tweets, qui n'ont certes pas le même sujet, mais qui ont presque le même nombre de retweets.

Figure 60 – Tweet faux



Source : capture d'écran de l'auteur

Figure 61 – Tweet véridique



Source : capture d'écran de l'auteur

Nous constatons que nous avons effectivement les mêmes résultats que pour l'analyse quantitative. Le tweet véridique de la Figure 61 a été beaucoup plus rapidement partagé que celui de la Figure 60. Nous pouvons conclure que la confiance des abonnés du journal LeMonde est la cause de la rapidité de propagation des tweets véridiques.

4.3. Limitations des résultats

Les résultats, obtenus par les analyses que nous avons faites précédemment, sont limités par deux éléments.

Premièrement, les résultats que nous avons obtenus sont valables uniquement pour les données que nous avons pu télécharger. En effet, comme nous l'avons mentionné dans la présentation du problème, il n'y a que peu de jeux de données de fausses nouvelles en français. Nos analyses sont donc basées sur relativement peu de données.

Deuxièmement, les API que nous avons utilisées sont pour la plupart limitées par Twitter. Nous n'avons pas pu intégrer certaines statistiques que nous désirions car les données étaient incomplètes.

Nous voulions calculer le pourcentage de retweets partagés par les followers d'un utilisateur afin d'établir une statistique. Pour ce faire, nous aurions alors comparé les followers d'un utilisateur avec les retweeters d'un tweet. Comme décrit dans le chapitre [2.3. Twitter API](#), l'API permettant de récupérer les followers est limitée à 5000 identifiants. Cette limite peut paraître élevée, mais puisque plus de deux tiers des utilisateurs de notre base de données ont plus de 5000 abonnés, nous avons, par conséquent, jugé cette statistique non pertinente pour nos résultats.

Une autre API, qui est aussi limitée, est celle pour récupérer les retweets. En effet, Twitter met une limite de 100 retweets par tweet. Sur 532 tweets véridiques et faux, 17% ont plus de 100 retweets. Nous avons donc cherché à savoir combien de retweets avaient été ignorés à cause de la limite de l'API. En réalité, dans notre base de données, tous nos tweets ont moins de 100 retweets, ce qui signifie que l'API n'est pas responsable du nombre de retweets incomplets. En revanche, nous pensons que nous n'avons pas pu récupérer tous les retweets car beaucoup d'utilisateurs ont protégé leurs tweets, ne rendant accessible leur contenu publié qu'à leurs abonnés. Il se peut aussi que certains retweeters se soient vus suspendre leur compte à cause du non-respect des conditions générales de Twitter. Une troisième possibilité, qui est la moins probable, est que l'utilisateur ait supprimé son retweet après que nous avons téléchargé le tweet original.

Pour finir, nous avons cherché à connaître le total de retweets actuels que nous avons réussi à télécharger. Au total, sur 79'345 retweets de tous les tweets collectés, nous n'avons que 18'165 retweets enregistrés dans notre base de données. Plus en détail, nous avons :

9151 retweets restant pour les tweets véridiques du Monde.fr sur 11'734 retweets initiaux

9014 retweets restant pour le jeu de fausses données sur 67'611 retweets initiaux

Pour démontrer ce problème, nous avons pris un tweet retweeté six fois. Nous pouvons constater qu'il y a trois personnes sur six qui manquent dans la liste de la Figure 62.

Figure 62 – Utilisateurs qui ont retweeté le message



Source : capture sur <https://twitter.com/RTenfrancais/status/841324754678272000/retweets>

5. Gestion du projet

5.1. Planification

Nous avons planifié ce travail durant les deux premières semaines. Nous avons joint dans ce document le cahier des charges en Annexe I. Nous l'avons rédigé avant de commencer ce travail afin d'estimer le temps de chaque tâche pour se fixer des délais. Le tableau ci-dessous résume les différentes phases ainsi que le nombre d'heures estimées pour chacune d'entre elles :

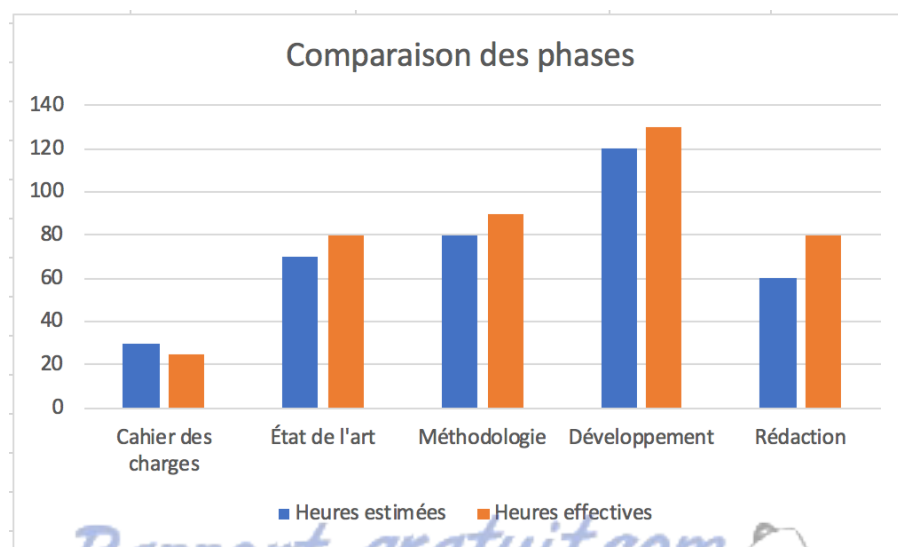
Tableau 11 – Planification des phases

	Phases	Heures
0	Cahier des charges	30
1	État de l'art	70
2	Méthodologie	80
3	Développement	120
4	Rédaction	60
Total		360

Source : cahier des charges

En calculant approximativement les heures que nous avons passées pour chaque phase, nous avons créé un graphique comparant ces dernières avec les heures que nous avons estimées dans le cahier des charges :

Figure 63 – Comparaison des heures réalisées pour les phases



Source : chiffre de l'auteur

Notre travail de Bachelor a été prévu pour être réalisé sur un total d'environ 360 heures réparties sur 14 semaines. Le Tableau 12 montre la répartition des heures que nous devons consacrer à notre travail en moyenne :

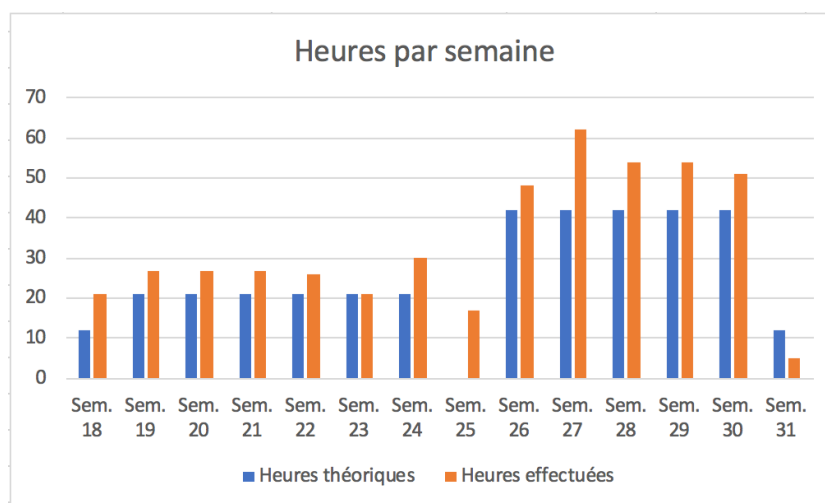
Tableau 12 – Répartition des 360 heures

Semaines	Heures théoriques
1. Sem. 18	12
2. Sem. 19	21
3. Sem. 20	21
4. Sem. 21	21
5. Sem. 22	21
6. Sem. 23	21
7. Sem. 24	21
8. Sem. 25 (examens de module)	0
9. Sem. 26	42
10. Sem. 27	42
11. Sem. 28	42
12. Sem. 29	42
13. Sem. 30	42
14. Sem. 31	12

Source : données de l'auteur

Chaque jour, nous avons noté le travail réalisé ainsi que le temps que nous y avons consacré. Ainsi, nous avons pu calculer le nombre d'heures que nous avons travaillé par semaine. Ci-après un graphique comparant les heures saisies dans le Tableau 12 et les heures effectives :

Figure 64 – Comparaison des heures par semaine



Source : données de l'auteur

Tout au long de ce travail, nous avons rencontré les responsables du projet “Designing a Human-Machine Hybrid Fake News Detection System” toutes les deux semaines. Nous avons pu ainsi valider, petit à petit, le travail réalisé entre ces séances.

À chaque rencontre, nous avons rédigé un procès-verbal afin de garder une trace des décisions prises lors de nos rencontres. Voici un tableau qui résume toutes les séances que nous avons eues :

Tableau 13 – Séances avec les responsables du projet

Dates	Résumé de la séance
1 mai 2019	- Séance chaque deux semaines + PV - Noter le nombre d'heures consacrées au travail de Bachelor ainsi qu'une brève description de la tâche réalisée - Cahier des charges
15 mai 2019	- Finalisation du cahier des charges - État de l'art (données, plateformes)
28 mai 2019	- Données validées par les responsables du projet - API Twitter - Rédaction du chapitre méthodologie
25 juin 2019	- Terminer la base de données - Chercher un jeu de données de nouvelles véridiques - Établir les statistiques avec des données récupérées
9 juillet 2019	- Statistiques supplémentaires - Évaluer la pertinence d'ajouter les données du journal le Figaro
24 juillet 2019	- Ajout du chapitre analyses résultats - Déploiement de la plateforme

Source : données de l'auteur

5.2. Respect des délais

Lors de la création du cahier des charges, nous avons fixé des délais afin de s'assurer de l'avancement de notre travail. Ci-dessous l'échéancier que nous avons réalisé ainsi que les dates où nous avons effectivement rendus les éléments requis :

Tableau 14 – Comparaison des délais

Échéancier	Date prévue	Date effective
Remise du cahier des charges	15 mai 2019	16 mai 2019
État de l'art	28 mai 2019	10 juin 2019
Méthodologie	12 juin 2019	28 juin 2019
Développement	17 juillet 2019	14 juillet 2019
Rédaction du travail de Bachelor	24 juillet 2019	27 juillet 2019

Sources : données de l'auteur (et cahier des charges)

Certaines phases ont pris davantage de temps, car nous avons dû les adapter tout au long de la réalisation de ce travail de Bachelor. Nous pouvons voir que, malgré le retard accumulé sur les phases « État de l'art » et « Méthodologie », nous avons terminé le développement légèrement plus tôt que prévu. Nous avons donc eu plus de temps pour la relecture de ce document.

5.3. Logiciels utilisés

Nous avons développé notre plateforme sur le logiciel WebStorm de JetBrains. Nous avons choisi cet outil car nous avons eu l'habitude, pendant notre formation, d'utiliser les outils de JetBrains. WebStorm est idéal car notre plateforme repose sur les langages HTML, CSS, JavaScript (avec NodeJS).

Afin d'examiner les API de Twitter, nous nous sommes aidés du logiciel Postman qui permet de gérer des requêtes d'API. Grâce à son usage convivial, Postman nous a fait gagner un temps considérable et nous a permis d'essayer différentes requêtes avant de trouver exactement celles que nous voulions.

Les graphiques comparant les heures ont été réalisés avec Excel, puis nous les avons importés dans notre travail.

6. Bilan

Ce travail de Bachelor, réalisé en trois mois, a été une expérience professionnelle excellente pour l'auteur. Ce travail a contribué avec succès au projet global "Designing a Human-Machine Hybrid Fake News Detection System". Les jeux de données sélectionnés sont pertinents et cela constitue un réel atout pour les responsables du projet global. De plus, une plateforme opérationnelle a été créée et peut être utilisée avec davantage de données.

Presque toutes les technologies utilisées pour le développement étaient nouvelles pour l'auteur qui n'avait pas vu, durant sa formation, le cœur de la plateforme, Node.js. La librairie Fuse.js, qui a permis de rendre possible l'autocomplétion, était inconnue de l'auteur avant ce travail. Beaucoup d'heures ont donc été nécessaires afin de maîtriser toutes ces technologies.

Il a été parfois compliqué de respecter les délais car ce travail a été réalisé en parallèle des cours, des projets/contrôles continus et des examens de module les deux premiers mois. De plus, nous aurions dû prendre davantage de temps pour comprendre les technologies avant de les utiliser, cela nous aurait permis de développer la plateforme plus rapidement et de manière plus exhaustive. Malgré ces deux points négatifs, l'auteur a apprécié mettre en pratique ses connaissances et ses compétences au profit de ce travail de Bachelor.

Conclusion

La **première partie** de notre travail recense les travaux similaires. Nous avons tout d'abord cité le travail de Nicolas Vanderbiest du blog Reputatio Lab. Vanderbiest a analysé plus de 20'000 messages du réseau social Twitter concernant les attentats à Nice et s'est intéressé particulièrement aux utilisateurs qui ont partagé les informations. Il a créé un tableau listant une multitude de profils type ainsi que le nombre d'utilisateurs ayant partagé l'information. Il s'est aperçu que les utilisateurs diffusant la majorité des nouvelles sont dans la tranche d'âge 18-25 ans. Il a également démontré à l'aide d'un graphique géographique que les utilisateurs diffusant l'information n'étaient pas à Nice lors des attentats. Nous nous sommes ensuite penchés sur un travail de Master réalisé par Fernando Flores García. Ce dernier a élaboré une plateforme qui affiche une carte ainsi qu'un graphique connecté pour montrer jusqu'où vont certaines informations géographiquement. Le dernier travail, publié par le groupe SocialFlow, a représenté, grâce à un graphique, l'impact d'un message qui avait été publié par Keith Urbahn. Ce dernier avait posté sur Twitter, peu de temps avant l'annonce officielle, la mort de Bin Laden. Les auteurs, Gilad & Devin du groupe SocialFlow, ont aussi créé un graphique chronologique qui a montré le nombre de partages par tranches d'heure.

Nous nous sommes ensuite penchés sur les jeux de données que nous pouvions télécharger gratuitement. Nous nous sommes aperçus qu'il en existait très peu en français. Nous avons quand même trouvé trois jeux de données qui correspondaient à nos critères. Les deux premiers, Le Gorafi et NordPresse, étaient assez similaires. Ce sont des journaux électroniques qui publient des satires/parodies à but humoristique. Nous nous sommes vite rendu compte qu'en choisissant un de ces journaux, nous aurions manqué de sources variées, nos statistiques auraient été moins pertinentes. Nous avons donc continué nos recherches et avons trouvé la base de données de fausses nouvelles que recense le journal LeMonde.fr.

Il a fallu aussi examiner les plateformes similaires à celle que nous envisagions de créer. La première, Hoaxbuster, permet de chercher, parmi divers articles et forums, une fausse information. Le site donne ensuite diverses informations, ainsi qu'une justification de sa position sur la nouvelle. Hoaxbuster ne donne cependant aucune statistique de propagation de l'information. Il n'y a également pas de graphiques montrant la diffusion de la nouvelle sélectionnée. HOAXY, la deuxième plateforme que nous avons trouvée, est plus récente. Elle permet de chercher sur Twitter un sujet et d'afficher un graphique connecté montrant la diffusion sur le sujet entré. Malheureusement, HOAXY ne trie pas la catégorie d'informations retournées par Twitter. Autrement dit, HOAXY n'affiche pas que des fausses nouvelles.

En **deuxième partie**, nous avons réfléchi aux choix auxquels nous étions confrontés, à savoir quel réseau social, quelles API, quelle base de données, quel outil de développement et finalement quelles librairies de visualisation étaient les plus appropriés à notre plateforme.

Dans un premier temps, nous avons cherché le réseau social qui correspondrait le mieux à nos besoins. Nous nous sommes rapidement rendu compte que Twitter était le seul réseau social sur lequel nous pouvions télécharger du contenu.

Après avoir choisi le jeu de données, nous l'avons examiné de plus près afin d'en extraire les liens redirigeant vers de fausses nouvelles publiées sur Twitter. Après avoir nettoyé les données, il nous restait moins de 200 liens Twitter. En effet, beaucoup de tweets avaient été supprimés par leur auteur. Nous avons donc cherché manuellement sur Twitter des tweets relatant de fausses nouvelles afin de compléter notre jeu de données de fausses informations. Après avoir passé le cap des 200 liens redirigeant vers une fausse nouvelle, d'un commun accord avec les responsables du projet, nous avons trouvé judicieux d'intégrer un jeu de données véridiques afin de pouvoir créer davantage de statistiques. Nous nous sommes tournés vers le journal LeMonde afin de télécharger une partie des tweets qu'il avait publiés.

Après avoir défini quelques termes propres à Twitter, nous avons eu besoin ensuite de voir quelles informations nous pouvions obtenir du réseau social Twitter. Nous avons consulté différentes API et en avons retenu quatre. Après avoir obtenu un compte développeur, ces API ont permis de remplir la base de données avec diverses informations pour pouvoir, par la suite, les afficher sur notre plateforme.

Nous nous sommes ensuite intéressés aux différentes bases de données afin de stocker les données que nous avons téléchargées. Nous avons classé dans deux SGBD cinq bases de données. Au final, la base de données MariaDB remplissait le plus de critères que nous avons définis.

Le choix de l'outil de développement a été un des plus importants de ce travail. Nous avons choisi Node.js car il était l'outil de développement le plus approprié pour notre plateforme. Il nous a permis, entre autres, de ne pas ajouter un langage de programmation de plus pour le développement. Grâce à la complétude de Node.js, nous n'avons pas eu besoin d'installer un serveur Apache afin d'accéder à notre plateforme localement.

Avant de commencer à développer notre plateforme, nous avons cherché quelques librairies permettant de créer des graphiques. Nous avons pu dresser un tableau comparatif qui nous a permis, une fois que nous avons notre plateforme, de facilement choisir les librairies qui seraient les plus adéquates.

Pour la **troisième partie**, nous avons élaboré la plateforme qui permet de visualiser la propagation de nouvelles. Comme aucun travail n'avait été effectué préalablement, nous avons dû faire, dans un premier temps, un diagramme de classes afin de modéliser les données.

Après avoir préparé la structure des données, nous avons repris et ajusté plusieurs scripts python avec lesquels nous avons appelé les API de Twitter. Grâce au diagramme de classes, nous savions quelles informations récupérer. Nous avons converti les données pour qu'elles puissent être importées dans notre base de données. Avant de le faire, nous avons dû supprimer les doublons que nous avions afin qu'ils ne faussent pas les futures statistiques.

Nous avons ensuite créé une nouvelle base de données sur MariaDB que nous avons nommée « tweets ». Après avoir ajouté les tables nécessaires, nous avons décidé d'employer une application permettant d'avoir une interface graphique de notre base de données. Cela a facilité grandement la visualisation de nos données.

Une fois Node.js installé, nous avons pu créer les fondations de notre plateforme. Nous avons fait attention de séparer notre code en plusieurs fichiers JavaScript afin de faciliter sa compréhension. Grâce à Node.js, nous avons gagné un temps considérable car nous n'avons pas eu besoin de mettre en place un serveur Apache, et nous avons importé des modules Node.js qui ont facilité le développement de notre plateforme.

Après avoir validé avec les responsables du projet l'interface modélisée avec le logiciel Axure, nous avons créé deux vues. Pour la première page, nous avons utilisé la librairie Fuse.js qui nous a facilité l'implémentation de la recherche de tweets. De plus, elle s'est avérée beaucoup plus rapide que la recherche avec des requêtes SQL. Pour la deuxième page, nous avons eu besoin des librairies Chart.js et Vis.js. Ces librairies nous ont permis de rapidement créer des graphiques dynamiques. Un graphique géographique a été envisagé, mais trop de données manquaient. Nous avons donc pris la décision de ne pas le créer, car il aurait manqué de pertinence.

*

La **limite** de trois mois nous a imposé de mettre des priorités quant au développement de ce travail de Bachelor. En effet, nous n'avons pas pu exploiter tout le potentiel de Node.js. Il vaudrait la peine d'améliorer le code des contrôleurs afin de gagner en rapidité d'exécution.

Les retweets que nous avons récupérés ne proviennent pas systématiquement des abonnés directs de l'auteur du message. Actuellement, nous n'avons que le pourcentage de retweeters qui sont abonnés à l'auteur. Il serait donc intéressant de créer un graphique connecté pour montrer la profondeur de partage qu'ont certains tweets.

Quant aux **perspectives de recherches ultérieures**, nous pourrions envisager de faire du data mining sur la description des auteurs. En effet, diverses API permettent de définir une émotion à travers un texte. Nous pourrions alors comparer les utilisateurs partageant de fausses nouvelles avec ceux partageant des nouvelles véridiques. Il serait donc intéressant d'implémenter cette fonctionnalité à notre plateforme, cela lui apporterait une valeur ajoutée.

Nous pourrions aussi permettre à l'utilisateur d'importer ses propres données, dans un format csv par exemple. Il pourrait ainsi choisir d'utiliser les données que nous avons collectées ou les données qu'il a sélectionnées. Une fois déployée, cette fonctionnalité permettrait à notre plateforme de rencontrer un intérêt auprès d'un public plus large qu'à l'heure actuelle.

Pour finir, avec les données que nous avons téléchargées, il est difficile de calculer la portée que peut avoir une nouvelle. Nous avons, actuellement, uniquement les retweeters d'un tweet. Nous ne savons cependant pas si ces retweeters font partie des followers de l'auteur du tweet. D'ailleurs, c'est pour cette raison que nous n'avons pas pu établir de graphique connecté. Pour pallier ce problème, il faudrait télécharger tous les followers des retweeters afin d'établir une « chaîne » de retweets jusqu'au tweet de base pour pouvoir en analyser la portée.

Références

- Arsenault, C. (2017, Avril 20). *The Pros and Cons of 8 Popular Databases*. Consulté le Juillet 2, 2019, sur keycdn: <https://www.keycdn.com/blog/popular-databases>
- Bostock, M. (2018, Septembre 13). *Disjoint Force-Directed Graph*. Consulté le Juin 2, 2019, sur Observable: <https://observablehq.com/@d3/disjoint-force-directed-graph?collection=@d3/d3-force>
- Cytoscape.js*. (2019, Juin 2). Récupéré sur Cytoscape: <http://js.cytoscape.org/#introduction>
- D3.js - Introduction*. (s.d.). Consulté le Juin 2, 2019, sur tutorialspoint: https://www.tutorialspoint.com/d3js/d3js_introduction.htm
- Data-Driven Documents*. (s.d.). Consulté le Juin 2, 2019, sur d3js: <https://d3js.org>
- Direction de l'information légale et administrative (Premier ministre). (2018, Octobre 10). *Comment dit-on « fake news » en français ?* Consulté le Mai 23, 2019, sur Service-Public: <https://www.service-public.fr/particuliers/actualites/A12950>
- Dizikes, P. (2018, Mars 8). *Study: On Twitter, false news travels faster than true stories*. Consulté le Mai 22, 2019, sur MIT News: <http://news.mit.edu/2018/study-twitter-false-news-travels-faster-true-stories-0308>
- Django makes it easier to build better Web apps more quickly and with less code*. (s.d.). Consulté le Juillet 2, 2019, sur Django: <https://www.djangoproject.com>
- Evaluate Twitter data to inform business decisions*. (s.d.). Consulté le Juillet 12, 2019, sur Twitter: <https://developer.twitter.com/en/use-cases/analyze>
- FAQ Index*. (s.d.). Consulté le Juillet 18, 2019, sur Hoaxy: <https://hoaxy.iuni.iu.edu/faq.php#faq-q2>
- García, F. F. (2015, Janvier). *Analyzing and visualizing news spread based on images in Social Media networks*. Consulté le Mai 30, 2019, sur <https://esc.fnwi.uva.nl/thesis/centraal/files/f862085995.pdf>
- Gilad, & Devin. (2011, Mai 6). *Breaking Bin Laden: Visualizing The Power Of A Single Tweet*. Consulté le Mai 29, 2019, sur SocialFlow: <http://www.socialflow.com/breaking-bin-laden-visualizing-the-power-of-a-single/>
- Glossary*. (s.d.). Consulté le Juillet 17, 2019, sur Twitter: <https://help.twitter.com/en/glossary>
- Laravel vs Node vs Django*. (s.d.). Consulté le Juillet 2, 2019, sur Stackshare: <https://stackshare.io/stackups/laravel-vs-nodejs-vs-django>
- LeMonde. (2017, Décembre 19). *Le blog du Décodex*. Consulté le Mai 26, 2019, sur Le Monde: https://www.lemonde.fr/le-blog-du-decodex/article/2017/12/19/fausses-informations-les-donnees-du-decodex-en-2017_5231605_5095029.html
- LeMonde.fr. (s.d.). Consulté le Mai 18, 2019, sur LeMonde: https://s1.lemde.fr/mmpub/data/decodex/hoax/hoax_debunks.json
- Line Charts in plotly.js*. (2019, Juin 3). Récupéré sur plot: <https://plot.ly/javascript/line-charts/#connect-gaps-between-data>

- Liu, Z., & Glassey Balet, N. (2019). *Designing a Human-Machine Hybrid Fake News Detection System*. Demande de subvention, HES-SO Valais-Wallis, Informatique de gestion, Sierre.
- MariaDB. (s.d.). *Les différences de fonctionnalités entre MariaDB et MySQL*. Consulté le Juillet 2, 2019, sur MariaDB: <https://mariadb.com/kb/fr/mariadb-vs-mysql-features/>
- Sénécat, A. (2017, Décembre 20). *Les premiers diffuseurs de fausses informations sont souvent des pages Facebook douteuses*. Consulté le Mai 26, 2019, sur LeMonde: https://www.lemonde.fr/les-decodeurs/article/2017/12/20/les-grosses-pages-facebook-premieres-sur-l-intox_5232248_4355770.html
- Sénécat, A. (2017, Mars 9). *Trois fausses informations récentes qui ont influencé l'opinion*. Consulté le Juillet 1, 2019, sur LeMonde: https://www.lemonde.fr/les-decodeurs/article/2017/03/09/comment-3-faussees-informations-recentes-ont-faconne-l-opinion_5091843_4355770.html
- Shiff, L., & Rowe, W. (2018, Mars 5). *NoSQL vs SQL: Examining The Differences and Deciding Which To Choose*. Consulté le Juillet 2, 2019, sur bmc blogs: <https://www.bmc.com/blogs/sql-vs-nosql/>
- Vanderbiest, N. (2019, Janvier 30). *Comment la fausse information circule sur Twitter en situation d'attentat ? Le cas de Nice*. Consulté le Mai 29, 2019, sur Reputatio Lab: <http://www.reputatiolab.com/2019/01/comment-la-fausse-information-circule-sur-twitter-en-situation-dattentat-le-cas-de-nice/>
- vis.js. (s.d.). Consulté le Juillet 7, 2019, sur visjs: <https://visjs.org/#>
- Watts, N. (2018, Juillet 5). *5 Types of 'Fake News' and Why They Matter*. Consulté le Mai 23, 2019, sur ogilvy: <https://www.ogilvy.com/feed/5-types-of-fake-news-and-why-they-matter/>

Annexe I

Cahier des charges

Travail de Bachelor 2019

Visualizing the spread of fake news on social media.



Étudiant : Nicolas Piguet

Professeur : Nicole Glassey Balet

Nicolas Piguet
 Cahier des charges

Hes-so VALAIS WALLIS
 Haute Ecole de Gestion & Tourisme
 Hochschule für Wirtschaft & Tourismus

Table des matières

Introduction	1
Contexte	1
Problématique	1
Travail à effectuer	2
Planification	2
Phase 0 – Cahier des charges	2
Phase 1 – État de l'art	2
Phase 2 – Méthodologie	3
Phase 3 – Développement	3
Phase 4 – Rédaction	3
Échéancier	3
Rendus	4
Rapport	4
Plateforme	4
Conclusion	4

Nicolas Piguet
Cahier des charges



Introduction

Ce travail de Bachelor fait partie du projet "Designing a Human-Machine Hybrid Fake News Detection System" de l'Institut d'informatique de gestion du Technopôle à Sierre.

Contexte

Le sujet des fake news est de plus en plus alarmant. Ces nouvelles, qui sont fausses, ont un impact négatif sur la société. Elles sont principalement diffusées sur les réseaux sociaux. Ces derniers sont devenus bien plus qu'une plateforme pour échanger entre amis. En effet, ils abordent des sujets comme la politique, l'environnement et divers événements.

Les médias traditionnels ont été peu à peu remplacés par les médias électroniques. Ces derniers sont très accessibles, tout autant aux auteurs qu'au public car souvent gratuits. Vu la quantité d'informations qui pullulent sur les réseaux sociaux, il devient difficile d'en vérifier les sources et d'assurer une certaine qualité des informations qui sont à notre disposition. En effet, cette transition du papier à l'électronique a permis à n'importe qui de s'exprimer sans qu'aucune vérification ne soit faite. Cette nouvelle forme de communication peut donc chaque jour répandre de nouvelles fausses, sans que les lecteurs ne s'en aperçoivent.

Problématique

Ce travail vise à analyser la propagation de nouvelles fausses sur les réseaux sociaux. Cette analyse sera basée sur des articles en français, qui sont moins nombreux que ceux écrits en anglais. La principale difficulté sera donc de rassembler assez d'articles afin d'avoir suffisamment de données.

La plateforme sur laquelle nous afficherons les résultats devra être relativement simple pour visualiser rapidement les différences parmi les articles. Nous disposerons les résultats sous forme de graphiques, ainsi que des informations sur le type de profil utilisateur qui répand l'information.

Nicolas Piguet
Cahier des charges



Travail à effectuer

Voici en résumé les différents objectifs de ce travail :

- Collecter suffisamment d'articles en français via l'API de Twitter et les sauver dans une base de données afin de pouvoir par la suite les intégrer dans le projet de recherche.
- Développer une plateforme web permettant d'afficher ces résultats. Cette interface devra afficher un graphique ainsi que des informations sur les utilisateurs Twitter.

Planification

Étant donné que ce travail sera effectué par un seul étudiant, l'utilisation de la méthode Scrum ne semble pas justifiable. En effet, un daily meeting individuel n'a que peu de sens. Cependant, nous garderons l'équivalent du sprint review. Chaque deux semaines, nous nous entretiendrons avec les responsables du projet afin de nous assurer du bon avancement du projet.

Environ 360 heures devront être consacrées à ce travail. Ce projet doit être effectué du 30 avril 2019 au 31 juillet 2019. Avant les examens de module, ce travail sera réalisé en parallèle avec les cours. D'après le planning scolaire, 21 heures par semaine sont consacrées à la réalisation du travail de Bachelor jusqu'aux examens finaux, ce qui fait un total de 147 heures. Après la semaine sans cours consacrée aux examens, il restera environ 220 heures à répartir sur 5 semaines.

Phase 0 – Cahier des charges

Cette étape consiste à planifier les phases de ce travail. À la fin de la phase 0, le cahier des charges sera rendu, soit le 15 mai 2019.

Phase 1 – État de l'art

Cette phase consistera à chercher un jeu de données existant de fausses nouvelles en français. Si malheureusement aucun jeu n'existe, nous devrons en créer un manuellement à l'aide de l'API Twitter.

Nicolas Piguet
Cahier des charges



 Haute Ecole de Gestion & Tourisme
 Hochschule für Wirtschaft & Tourismus

Phase 2 – Méthodologie

Nous nous occuperons ici de choisir la meilleure solution pour afficher les résultats. Nous passerons en revue tous les outils que nous pourrions utiliser, et tâcherons de faire une sélection parmi eux.

Phase 3 – Développement

Après avoir choisi notre approche, nous implémenterons cet outil de visualisation. Notre plateforme accèdera directement à notre base de données de tweets en français. Elle devra correspondre à ce qui a été convenu avec les responsables de projet.

Phase 4 – Rédaction

Nous devons, après avoir fini les trois premières phases, terminer la rédaction du travail de Bachelor. Il nous restera les chapitres : Résumé, Avant-Propos, Planification, Bilan, Synthèse.

Échéancier

Début du travail de Bachelor	30.04.2019	
Phase 0 : Remise du cahier des charges	15.05.2019	(30h)
Phase 1 : État de l'art	28.05.2019	(70h)
Phase 2 : Méthodologie	12.06.2019	(80h)
Phase 3 : Développement	17.07.2019	(120h)
Phase 4 : Rédaction	24.07.2019	(60h)
Rendu du travail de Bachelor	31.07.2019	

Nicolas Piguet
Cahier des charges


Haute Ecole de Gestion & Tourisme
Hochschule für Wirtschaft & Tourismus

Rendus

Rapport

Ce document sera composé d'une introduction sur le sujet, ainsi qu'une partie de recherche et de statistiques à l'aide du jeu de données. Il comparera aussi les résultats parmi différentes nouvelles grâce au jeu de données que nous aurons dans la base de données.

Plateforme

La plateforme devra afficher les résultats sous forme de graphiques. Elle permettra aussi d'afficher le profil utilisateur type. Tout ceci sera possible grâce au lien entre la plateforme et notre base de données.

Conclusion

Ce cahier des charges planifie approximativement le développement de ce travail de Bachelor. En début de projet, il est parfois difficile de se rendre compte de la longueur des phases. Cependant, nous pensons que les phases ainsi que l'échéancier décrivent suffisamment bien le déroulement de ce projet.

Annexe II

Nicolas Piguet

Déploiement

Ce document décrit la manière dont nous avons rendu disponible en ligne notre application. La plateforme a été déployée sur le serveur de la HES-SO.

Requis sur le serveur

Node.js

Base de données mySQL

Exporter la base de données

Afin d'exporter notre base de données « tweets », nous avons écrit la ligne de commande suivante dans le Terminal : `sudo mysqldump -p tweets > tweets.sql`

Créer une base de données

Après avoir exporté notre base de données, nous avons créé une base de données sur phpMyAdmin. Nous avons pu, ensuite, importer notre fichiers `tweets.sql`.

Cloner le projet

Nous avons dupliqué notre projet (sans le dossier « node_modules ») afin de le mettre dans un dossier sur le serveur. Depuis le Terminal, nous nous sommes rendus dans le dossier de notre projet et avons lancé la commande : `npm install`

Cette commande a permis d'initialiser Node.js et, grâce au fichier `package.json`, de télécharger les dépendances nécessaires au projet.

Nous avons dû faire attention au numéro de port que nous avons défini.

Lancer le projet

Afin de lancer notre projet, nous pouvons écrire dans le Terminal la commande suivante : `nodemon app`

Déclaration de l'auteur

Je déclare, par ce document, que j'ai effectué le travail de Bachelor ci-annexé seul, sans autre aide que celles dûment signalées dans les références, et que je n'ai utilisé que les sources expressément mentionnées. Je ne donnerai aucune copie de ce rapport à un tiers sans l'autorisation conjointe du RF et du professeur chargé du suivi du travail de Bachelor, y compris au partenaire de recherche appliquée avec lequel j'ai collaboré, à l'exception des personnes qui m'ont fourni les principales informations nécessaires à la rédaction de ce travail et que je cite ci-après : Nicole Glassey Balet, Zhan Liu.

