Etude de données Twitter en lien avec l'élection présidentielle française d'avril 2022

Siméon Gheorghin

▶ **To cite this version:**

Siméon Gheorghin. Etude de données Twitter en lien avec l'élection présidentielle française d'avril 2022. Informatique [cs]. 2022. hal-03897939

HAL Id: hal-03897939
https://inria.hal.science/hal-03897939

Submitted on 15 Dec 2022

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

PSL

Etude de données Twitter en lien avec l'élection présidentielle française d'avril 2022

Rapport de stage de recherche de
Licence 2

Siméon GHEORGHIN
CPES – L2
Mathématiques – Physique

Encadrant : Pierre Senellart
Laboratoire : Département d'Informatique de l'ENS
Equipe : Valda

Soutenu le 27/06/2022 à l'Ecole des Mines de Paris - PSL

Table des matières

Introduction

Les réseaux sociaux génèrent actuellement des quantités gigantesques de données (Facebook en générait déjà 500 téraoctets par jour en 2012[1]), du fait de leur utilisation par des centaines de millions de personnes sur toute la planète. Indépendamment de l'utilisation commerciale que les entreprises font de ces données celles-ci peuvent être utiles aux chercheurs : elles permettent par exemple d'étudier les opinions dans la société sur certains sujets majeurs. L'extraction et l'analyse de ces données nécessitent toutefois des techniques informatiques particulières pour traiter des volumes de données considérables et obtenir des informations à partir de données complexes.

L'équipe Valda du Département d'Informatique de l'Ecole Normale Supérieure, au sein de laquelle j'ai réalisé mon stage, est une équipe de recherche commune à l'Inria Paris, l'Ecole Normale Supérieure et le CNRS[2]. Elle se concentre sur « la gestion de données complexes, en particulier les données produites par l'activité humaine »[3]. Les principaux axes de recherche de l'équipe sont la construction d'outils adaptés à l'extraction des données, la gestion de données incomplètes ou incertaines et les systèmes de gestion des informations personnelles. Etant particulièrement intéressé par l'étude des données et les techniques de Machine Learning ce stage était pour moi l'occasion de découvrir une démarche complète d'analyse de données et des méthodes avancées pour leur extraction et stockage ainsi que des techniques pour l'analyse de données textuelles.

Dans ce rapport est présentée une étude de données issues de Twitter dont l'objectif final, que je n'ai pas eu le temps d'atteindre, aurait été d'obtenir des informations sur l'élection présidentielle française d'avril 2022. La première étape de l'analyse a été l'enregistrement des données, réalisé en utilisant l'API[4] de Twitter. La deuxième étape a consisté en la réalisation d'un classifieur permettant d'identifier les tweets en lien avec l'élection présidentielle. J'ai essayé plusieurs méthodes pour réaliser ce classifieur, afin de comparer les résultats obtenus dans chaque cas. En conclusion je propose des approches qui auraient pu être suivies pour prolonger ce travail, notamment la visualisation des données ou l'analyse d'opinion.

Les principaux objectifs de ce travail étaient la découverte et la mise en œuvre des techniques d'extraction et de stockage de données ainsi que des méthodes de traitement automatique des langues, ou « Natural Language Processing » (NLP), et de représentation de textes, en l'occurrence des tweets, comme vecteurs dans des espaces de grandes dimensions. L'utilisation de ces méthodes permet d'utiliser des algorithmes de Machine Learning sur des données textuelles.

Toutes les manipulations ont été réalisées en utilisant Python et les codes détaillés sont présentés en annexe de ce rapport, disponible sur GitHub en suivant le lien[5]. Elles ont été réalisées sur un serveur mis à disposition par l'Ecole Normale Supérieure, les données étant stockées sur le même serveur. Les modules Python utilisés pour chaque étape sont présentés à la fin du rapport.

[1] https://www.lemondeinformatique.fr/actualites/lire-facebook-supporte-500-to-de-nouvelles-donnees-par-jour-50140.html
[2] Centre National de la Recherche Scientifique
[3] https://team.inria.fr/valda/fr/
[4] Voir Glossaire
[5] https://github.com/SimCPES/Stage_Twitter.git

I/ Enregistrement des données

A) Introduction à l'API Twitter

Twitter dispose d'une API[6] à laquelle il est possible de s'inscrire simplement et qui permet d'interagir avec ce réseau social, pour utiliser un compte ou récupérer des données. Elle permet par exemple d'envoyer des messages privés, de poster des statuts ou tweets, de rechercher les tweets publiés par un utilisateur donné (recherche par utilisateur) ou ceux comportant un certain mot ou une certaine expression (recherche par mot-clé).

Dans un premier temps je me suis inscrit pour utiliser l'API : j'ai précisé l'utilisation que je ferai des données collectées. Twitter s'assure ainsi que ses données sont utilisées dans un cadre de recherche et qu'elles ne seront pas rendues publiques. Ces garanties font parties de leur politique de sécurité, des clés de sécurité permettent d'assurer une connexion sécurisée.

Le module Tweepy[7] est un module Python dédié à l'utilisation de l'API Twitter. Il propose de nombreuses fonctionnalités incluant la recherche de comptes par nom d'utilisateur, la recherche par mots-clés et utilisateur et le streaming de tweets. Les tweets sont présentés sous forme d'objets Python dont les attributs sont les caractéristiques du tweet : texte, émetteur, date de publication… etc. Ils disposent d'un attribut « _json » donnant le tweet sous forme d'un dictionnaire Python dont les clés sont les caractéristiques du tweet. Cela permet d'enregistrer les données au format de données JSON, que j'ai utilisé pour le stockage des données.

B) Extraction des données

L'annexe 1 contient le programme utilisé pour l'enregistrement des données avec les fonctions de streaming avec filtrage. On précise un certain nombre de filtres comme des mots-clés, des comptes ou une langue, puis l'API parcourt tous les tweets publiés en temps réel et retourne ceux correspondants aux critères sélectionnés. J'ai sélectionné la langue pour que tous les tweets récupérés soient en français. L'objectif était d'enregistrer des tweets ayant un lien avec l'élection présidentielle et j'ai donc choisi un certain nombre de mots-clés et de comptes qui avaient un lien avec l'élection. Pour chaque mot-clé et chaque compte j'ai vérifié manuellement, grâce à une recherche par mot-clé, que les tweets récupérés étaient en lien avec l'élection.

Mots-clés	Comptes
candidat	EmmanuelMacron
candidats	MLP_officiel (Marine Le Pen)
candidate	JLMelenchon
candidates	Anne_Hidalgo
candidature	Vpecresse
élection	yjadot
électeur	dupontaignan
électeurs	ZemmourEric
électrice	ChTaubira
électrices	PhilippePoutou
présidentielle	UPR_Asselineau
élection présidentielle	Jeanlassalle
parti politique	Fabien_Roussel
#Présidentielle2022	J_Bardella

[6] https://developer.twitter.com/en/portal/dashboard
[7] https://www.tweepy.org/

sondage présidentielle #présidentielle #Présidentielle #MélenchonPresident Pécresse Mélenchon Zemmour Macron Jadot Hidalgo Taubira Roussel Poutou Le Pen	enmarchefr RNational_off EELV partisocialiste FranceInsoumise PrimairePop PCF lesRepublicains Reconquete2022

Mots-clés et comptes utilisés pour l'enregistrement des données

Les mots-clés et comptes ont été sélectionnés de manière impartiale en essayant de représenter au maximum toutes les tendances politiques de cette élection. On ne peut cependant pas exclure que ce choix arbitraire ait biaisé les données collectées. On aurait pu évaluer la pertinence de chaque mot-clé, chose que je n'ai pas eu le temps de faire. De plus cette étape a été réalisée plusieurs semaines avant l'élection et les tendances ont pu changer entre temps – par exemple Christiane Taubira se présentait un moment comme candidate avant de finalement renoncer – rendant certains choix moins pertinents. En définitive le choix des filtres utilisés pour l'enregistrement des données a pu les biaiser et il aurait été intéressant de chercher à évaluer ce biais.

C) Stockage des données

Les tweets vérifiant les conditions des filtres sont retournés par l'API. Pour conserver les données il faut ensuite les enregistrer dans un fichier. Les quantités de données à enregistrer sont importantes : entre 500 000 et 1 000 000 de tweets par jour. Il est nécessaire d'avoir des formats spéciaux dédiés au stockage des données de manière adaptée, par exemple le format de fichier JSON.

Les tweets sont récupérés en continu et enregistrés les uns à la suite des autres dans un même fichier, en étant séparés de manière ad hoc. On peut ensuite utiliser les fonctions du module json[8] pour lire le fichier et récupérer les données sous forme d'un objet Python, par exemple une liste. La solution que j'ai retenue a consisté à créer un fichier de tweets pour chaque journée et à enregistrer les données dans chaque fichier sous forme d'une liste de tweets, c'est-à-dire :

[{tweet 1}, {tweet 2}, … , {tweet n}]

pour un fichier contenant n tweets. La forme « {tweet i} » correspond à un tweet représenté par le dictionnaire Python donné par l'attribut « _json ».
De cette manière j'ai pu, lors de la lecture d'un fichier de tweets, récupérer les données sous forme d'une liste de tweets, pratique à manipuler.

[8] https://docs.python.org/fr/3/library/json.html

```
[{"created_at": "Fri Mar 11 10:19:34 +0000 2022", "id": 1502227723711885313, "id_str": "1502227723711885313", "text": "RT @_Rowlf: Pour prot\u00e9ger vos grands-parents prenez 3 doses de vaccin exp\u00e9rimental\nPour \"sauver le climat de la plan\u00e8te\" prenez le v\u00e9lo\nPou\u2026", "source": "<a href=\"http://twitter.com/download/iphone\" rel=\"nofollow\">Twitter for iPhone</a>", "truncated": false, "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": 1681870124, "id_str": "1681870124", "name": "Tadeusz \ud83c\uddeb\ud83c\uddf7 \u24e9", "screen_name": "jetadeus", "location": null, "url": null, "description": "Souverainiste. D\u00e9mocrate. Pour l\u2019\u00e9cologie r\u00e9elle, contre l\u2019oligarchie. \u00ab La tradition n\u2019est pas le culte des cendres, mais la pr\u00e9servation du feu \u00bb #Reconqu\u00eate", "translator_type": "none", "protected": false, "verified": false, "followers_count": 5730, "friends_count": 6030, "listed_count": 1, "favourites_count": 45557, "statuses_count": 19385, "created_at": "Sun Aug 18 23:32:02 +0000 2013", "utc_offset": null, "time_zone": null, "geo_enabled": false, "lang": null, "contributors_enabled": false, "is_translator": false, "profile_background_color": "C0DEED", "profile_background_image_url": "http://abs.twimg.com/images/themes/theme1/bg.png", "profile_background_image_url_https": "https://abs.twimg.com/images/themes/theme1/bg.png", "profile_background_tile": false, "profile_link_color": "1DA1F2", "profile_sidebar_border_color": "C0DEED", "profile_sidebar_fill_color": "DDEEF6", "profile_text_color": "333333", "profile_use_background_image": true, "profile_image_url": "http://pbs.twimg
```


Début d'un fichier contenant des tweets enregistrés

Les quantités de tweets récupérés étant importantes, les fichiers étaient d'une taille conséquente : plusieurs Gigaoctets par jour. Il a donc été nécessaire de les comprimer au format GZIP pour réduire leurs tailles et pouvoir tous les stocker. Finalement j'ai créé un petit programme permettant, à la fin de chaque journée, de finaliser la forme du fichier et de le compresser. Ce programme écrit en Bash est disponible en annexe 2.

D) Conclusion

Une fois les programmes présentés en annexes 1 et 2 finalisés j'ai pu enregistrer des données en continu en laissant les deux programmes tourner sur le serveur. Des données ont été enregistrées presque en permanence de fin mars à début juin, recouvrant en particulier toute la période de l'élection. L'objectif suivant a été de réaliser un classifieur permettant de déterminer si un tweet est bien en lien avec l'élection, en utilisant des méthodes d'analyse automatique des langues et de représentation numérique de texte.

II/ « Natural Language Processing »

Pour pouvoir analyser le contenu des tweets il faut être capable d'analyser leur texte de manière automatisée. En particulier on souhaite éliminer tous les mots qui ne portent pas de sens par eux-mêmes, tels que les prépositions, et on voudrait reconnaitre les mots ayant la même racine, par exemple différentes conjugaisons d'un même verbe, et retrouver leur forme canonique. Ces étapes de traitement de texte font parties du « Natural Language Processing » (NLP) ou traitement automatique des langues.

Les opérations que j'ai utilisées sont les suivantes :

- Lemmatisation : à partir d'un terme l'algorithme donne sa forme canonique, par exemple : « mangerai » donne « manger ».
- Stopwords : de nombreux termes ne sont pas porteurs de sens dans une langue, par exemple : « de », « la », « ai » en français. Ces mots sont appelés des stopwords.

L'objectif était de réaliser une fonction pouvant, à partir d'un tweet, retourner la liste des termes porteurs de sens contenus dans ce tweet, dans leur forme canonique. Cette fonction est présentée en annexe 3 : dans un premier temps on crée une liste de stopwords à partir du corpus proposé dans le

module NLTK[9], puis une fonction permettant de retirer ces stopwords d'une liste de mots. Ensuite, pour un tweet, grâce au module Spacy, on regarde la forme canonique de chacun des termes du tweet pour déterminer si on souhaite le conserver. On applique enfin la fonction pour retirer les stopwords de la liste des termes conservés.

Pour le tweet : (issu des données collectées)

« Valérie Pécresse elle est censée avoir été fonctionnaire toute sa vie, elle a un patrimoine de presque 10millions d'euros mais des gens s'inquiète de Philippe Poutou et son f2 à 120.000 euros »,

on obtient le résultat :

[« Valerie », « Pecresse », « etre », « censer », « avoir », « etre », « fonctionnair », « tout », « vie », « avoir », « patrimoine », « presque », « 10million », « euro », « gens », « inquieter », « Philippe », « Poutou », « f2 », « euro »].

III/ Représentation numérique de textes

Pour réaliser le classifieur on souhaite appliquer des algorithmes de Machine Learning sur des données qui sont les textes des tweets. Ces algorithmes ne pouvant s'appliquer qu'à des données numériques il est nécessaire d'avoir une représentation numérique de ces textes.

A) Représentation Bag of Words

Cette première approche est la plus simpliste, elle consiste à obtenir, à partir du corpus de textes, une matrice contenant pour chaque mot de chaque texte le nombre d'occurrences du mot dans le texte. Chaque ligne de la matrice correspond à un texte du corpus tandis que chaque colonne correspond à un mot de l'ensemble des mots contenus dans tous les textes du corpus. Cette représentation ne donne pas vraiment de bons résultats avec les algorithmes de Machine Learning. La matrice obtenue peut toutefois permettre, par exemple, de trouver le terme apparaissant le plus fréquemment dans le corpus.

B) Représentation avec le score TF-IDF

Cette approche est presque semblable à celle présentée précédemment. La matrice construite ne contient pas le nombre d'occurrences de chaque mot dans chaque texte mais son score TF-IDF[1][2] ou « Term Frequency – Inverse Document Frequency ». Il existe différentes versions de ce score, j'ai utilisé la version suivante :

$$tf - idf(t) = f_{t,d} \times \log\left(\frac{Card(D)}{1 + Card(\{d_j, t \in d_j\})}\right)$$

avec : t le mot considéré dans le texte d, $f_{t,d}$ la fréquence du terme t dans le texte d, D le corpus de textes et $\{d_j, t \in d_j\}$ l'ensemble des textes du corpus contenant le mot t.

On obtient ainsi pour chaque mot et chaque texte un score qui est composé de deux parties :

- La partie TF correspond à la fréquence du mot dans le texte. Elle rend le score grand lorsque le mot apparait de nombreuses fois dans le texte. On suppose ainsi qu'un mot apparaissant souvent dans le texte porte beaucoup d'informations et est important.

[9] En Python j'ai utilisé les modules Spacy et NLTK pour réaliser ces opérations. J'ai utilisé le modèle « fr_core_news_md » de Spacy. Voir plus de détails en fin de rapport.

- La partie IDF correspond à l'inverse de la fréquence à laquelle le mot apparait dans les textes du corpus, régularisée par un logarithme. On ajoute 1 au dénominateur pour éviter une division par zéro. De cette manière le score devient plus faible lorsque le mot apparait dans de nombreux textes du corpus : la fraction tend alors vers 1, donc le logarithme vers 0. On suppose en fait qu'un mot qui est présent dans tous les textes n'apporte pas beaucoup d'informations sur un texte.

Ce score reflète donc l'importance d'un mot dans un texte, c'est-à-dire à quel point il est porteur d'informations dans ce texte. La représentation numérique des textes obtenue avec ce score peut être utilisée pour appliquer les algorithmes de Machine Learning avec de bons résultats car elle représente les informations contenues dans les textes.

C) Principal Component Analysis (PCA) ou Analyse en Composantes Principales

Les matrices obtenues avec les deux approches présentées ci-dessus sont de grandes dimensions. Les algorithmes de Machine Learning étant peu performants sur des données avec trop de caractéristiques, elles ne sont pas utilisables directement. Une approche possible pour pallier ce problème est d'utiliser l'Analyse en Composantes Principales[3][4] dont l'objectif est de réduire les dimensions de la matrice tout en conservant les informations qu'elle contient. On calcule une matrice de variance-covariance qui représente les corrélations entre les différentes lignes de la matrice initiale, ce qui correspond à l'information que l'on souhaite conserver. On calcule ensuite les valeurs propres et vecteurs propres de cette matrice. Les vecteurs propres associés aux valeurs propres les plus grandes sont ceux qui représentent le plus d'information de la matrice. Finalement on conserve une matrice constituée des n vecteurs propres représentant le plus d'information. Cette matrice, qui n'a donc plus que n colonnes, représente la même information que la matrice initiale mais a des dimensions plus petites. On prend typiquement : n = 100. On peut ainsi passer d'une matrice de taille (100 000 x 1 000 000) à une taille (100 000 x 100) et conserver l'information. Je ne me suis pas beaucoup intéressé aux détails théoriques de cette méthode mais plus à son utilisation en pratique.

D) Modèles Word2Vec

Les modèles Word2Vec ont été développés par des chercheurs de Google[5][6], ils permettent de représenter des textes de manière numérique. Ce sont des réseaux de neurones artificiels qui apprennent à réaliser des associations de mots à partir de corpus contenant énormément de textes. Ils créent un espace vectoriel de très grandes dimensions, typiquement plusieurs centaines, dans lequel chaque mot est représenté par un vecteur unique. L'espace est créé de manière à ce que des mots ayant le même contexte dans les textes soient proches. De même des mots ayant des sens proches sont représentés par des vecteurs assez semblables les uns aux autres. Les modèles entraînés par Google permettent même de réaliser des opérations sur les mots : en additionnant « roi » et « femme » on obtient « reine ». En Python le module Spacy propose ce type de modèle, entraîné sur leurs données. J'ai utilisé le même modèle de Spacy que pour le NLP.

IV/ Gestion de la mémoire

A) Manipuler les données

Les quantités de données à manipuler sont importantes, un fichier de tweets pour une journée contient entre 500 000 et 1 000 000 de tweets et représente entre 5 et 10 Gigaoctets de données. Il est donc impossible de charger toutes les données d'un coup dans la mémoire vive de l'ordinateur

pour les manipuler. Une solution consiste à réaliser du streaming JSON. Plutôt que charger l'intégralité du contenu du fichier dans la mémoire, les algorithmes réalisant cette opération ne chargent qu'un objet du fichier à la fois, un tweet dans notre cas, dans la mémoire. On peut manipuler cet objet à ce moment-là, par exemple pour traiter le texte ou récupérer des informations sur le tweet, puis il est effacé de la mémoire et l'objet suivant est chargé. L'algorithme parcourt ainsi tout le fichier, permettant de réaliser des opérations sur tous les objets qu'il contient. La manipulation de gros volumes de données est un enjeu majeur lorsque l'on s'intéresse à l'analyse de données. J'ai utilisé le module Python Naya pour faire cela.

B) Sparse Matrix ou Matrice Creuse

Avec les représentations en matrices Bag of Words ou TF-IDF, la matrice obtenue contient un score pour chaque mot de l'ensemble des mots du corpus, pour chaque texte. Or un tweet ne contient qu'une petite partie des mots de cet ensemble. Un mot qui n'apparait pas dans un tweet aura un score nul pour ce tweet donc, pour un tweet, la majorité des mots aura un score nul. On obtient donc une matrice qui, à chaque ligne, contient une grande quantité de zéros, il s'agit d'une matrice creuse. Or cette matrice peut être très grande, dès lors que l'on travaille sur beaucoup de données. Il est alors coûteux de la stocker dans la mémoire vive.

Toutefois, puisqu'elle est constituée majoritairement de zéros, une grande partie de la matrice est inutile donc on choisit une structure de données ne conservant que les informations utiles. Il peut s'agir par exemple d'un tableau de taille fixe dont chaque case représente un tweet et contient une liste simplement chainée dont chaque maillon contient le mot, ou un indice associé au mot, et le score de ce mot dans ce tweet. La quantité d'information stockée est alors bien plus faible et on gagne beaucoup de place en mémoire. En Python, le module Scipy implémente ce type d'objet et cette implémentation est reprise dans le module Scikit-Learn que j'ai utilisé.

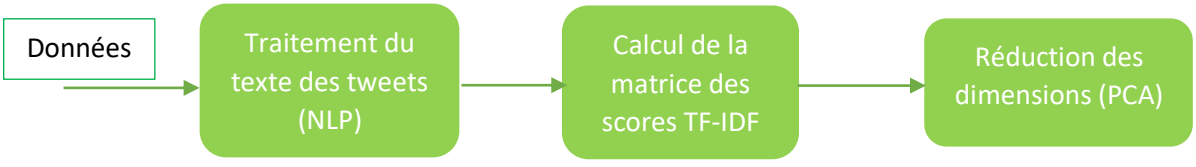
V/ Classifieur de tweets

Mon objectif principal a été la réalisation d'un classifieur capable de prédire si un tweet a un lien avec l'élection présidentielle, en utilisant les méthodes décrites précédemment.

A) Démarche, programme principal

Le problème que l'on cherche à résoudre est un problème de classification en apprentissage supervisé, c'est-à-dire que l'on cherche à prédire à quelle catégorie appartient un tweet et que l'entrainement du classifieur se fait sur des données étiquetées, pour lesquelles on sait quel doit être le résultat. Plusieurs étapes sont donc nécessaires : tout d'abord il faut sélectionner les données sur lesquelles le classifieur sera entrainé, puis on utilise les techniques présentées précédemment pour obtenir une représentation numérique complète des données sélectionnées et enfin on étiquette ces données.

Une fois que l'on dispose des données d'entrainement et des étiquettes correspondantes, la démarche de réalisation du classifieur est la suivante :



Une fois ces trois étapes réalisées on obtient une matrice de dimensions raisonnables, typiquement une centaine de colonnes, qui est une représentation numérique des données. Ayant les étiquettes

d'autre part on peut séparer les données en données d'apprentissage qui serviront à entrainer le classifieur et données de test qui serviront à évaluer ses performances. Le programme complet correspondant est disponible en annexe 4.

Dans les problèmes de Machine Learning le choix du classifieur peut se révéler crucial. Ce n'est toutefois pas un paramètre que j'ai beaucoup étudié. J'ai observé principalement les résultats des classifieurs suivants : les arbres de décisions, les forêts d'arbres de décisions ou Random Forest et les machines à vecteurs de support (SVM). Les classifieurs Random Forest donnaient systématiquement les meilleurs résultats et c'est donc ce type de classifieur que j'ai conservé.

B) Sélection des données

Le choix des données est un paramètre important dans les procédures de Machine Learning. Des données mal choisies peuvent amener à de mauvais résultats. J'ai essayé différentes méthodes, avec des volumes de données et des données différentes. Les différents volumes de données testés sont les suivant : 2 000, 10 000, 100 000 ou 200 000 tweets. Le score correspond à la proportion de données bien prédites parmi les données de test.

Volume de données en nombre de tweets	< 100 000	100 000	> 100 000
Résultat : score sur l'échantillon de test	Systématiquement moins bon que pour un volume de 100 000 tweets, avec un écart de 15% pour 2 000 tweets.	Le score dépend des données utilisées, systématiquement le meilleur compromis.	Le score n'est pas significativement meilleur que pour 100 000 tweets. Le temps d'entrainement augmente avec le volume de données.

J'ai également testé différentes approches pour sélectionner les données :

- Cas 1 : les données sélectionnées sont les 100 000 premiers tweets enregistrés le 29 mars. Le choix de cette journée, 12 jours avant le 1er tour de l'élection présidentielle, au cœur de la campagne, permettait d'avoir plus de tweets en lien avec l'élection et facilitait l'étiquetage.
- Cas 2 : les données sélectionnées sont 50 000 tweets enregistrés le 29 mars et 50 000 tweets en plus enregistrés avec du streaming sans filtrage. Ces tweets en plus n'ont a priori aucun lien avec l'élection.
- Cas 3 : les données sélectionnées sont 50 000 tweets issus de moments différents de 10 journées différentes réparties le long de la campagne électorale et les 50 000 tweets enregistrés en plus.

Procédure	Cas 1	Cas 2	Cas 3
Résultat : score sur l'échantillon de test	Inférieur à 0.7, assez mauvais. Tous les tweets se ressemblent car ils ont été enregistrés de la même manière. Il n'y a pas de tweets sans aucun lien avec l'élection.	Jusqu'à 0.3 de plus que le cas 1, le score atteint 0.9. Le prédicteur différencie vraiment les deux classes.	Meilleure approche, la plus aboutie. Le score atteint 0.95. Les données sont variées ce qui permet au prédicteur de généraliser.

En termes de volumes de données le meilleur compromis entre de bons scores sur l'échantillon de test et un temps d'entrainement du classifieur raisonnable est toujours obtenu pour un volume de 100 000 tweets. Il est nécessaire d'avoir des données les plus variées possibles, notamment des

données qui n'aient vraiment aucun lien avec l'élection, pour que le classifieur puisse clairement identifier les deux classes. J'ai enregistré des données en plus car, en raison de la manière dont elles étaient enregistrées, toutes mes données initiales se ressemblaient et je n'avais pas de données n'ayant pas du tout de lien avec l'élection. La procédure la plus aboutie et donnant les meilleurs résultats correspond au cas 3, c'est celle que j'ai utilisée par la suite.

C) Etiquetage des données d'entrainement

De même que pour les sélectionner, j'ai essayé différentes approches pour étiqueter les données. On veut donner une étiquette à chacun des tweets sélectionnés pour entrainer le classifieur, c'est-à-dire décider s'il est en lien avec l'élection. Il faut que cet étiquetage soit précis car la manière dont le classifieur fera ses prédictions dépend fortement de ces données.

Deux approches sont possibles : soit on étiquette les tweets manuellement, c'est-à-dire qu'on lit chaque tweet pour déterminer l'étiquette à donner, soit on le fait de manière automatique en choisissant des critères pour qu'un tweet soit considéré en lien avec l'élection. La première approche est longue et fastidieuse mais plus précise. J'ai étiqueté manuellement 2 000 tweets, ce qui n'est pas une quantité de données suffisante pour entrainer le classifieur. Il est plus pratique d'utiliser l'étiquetage automatique, ce que j'ai fait par la suite. J'ai d'abord réalisé une liste de comptes de candidats à l'élection, de partis politiques ou de personnalités politiques. Les tweets étaient étiquetés comme étant en lien avec l'élection s'ils étaient issus, s'ils étaient une réponse ou un retweet d'un de ces comptes. La liste est la suivante :

```
["JLMelenchon", "MLP_officiel", "RNational_off", "avecvous", "FranceInsoumise", "Anne_Hidalgo",
"vprecresse", "yjadot", "ZemmourEric", "AEC_LeProgramme", "enmarchefr", "2022avecHidalgo",
"lesRepublicains"]
```

J'ai ensuite ajouté des critères, les tweets étiquetés comme étant en lien avec l'élection étaient les mêmes que précédemment mais également ceux mentionnant un des comptes de la liste précédente ou comprenant un mot-clé de la liste donnée ci-dessous. J'ai élaboré cette liste à partir des données étiquetées manuellement : elle contient les mots qui à la fois apparaissaient souvent dans les tweets étiquetés en lien avec l'élection et peu souvent dans ceux sans lien. Ce sont donc les mots qui permettent le plus de distinguer les deux classes. Tous les autres tweets étaient étiquetés comme sans lien avec l'élection. La liste est la suivante :

```
["presidentiel", "presidentielle", "vote", "programme","#presidentielles2022",
 "#presidentielle2022", "sondage", "le pen", "meeting", "#remigration", "electeur",
 "second tour", "elire", "election", "extreme droite", "pourcent", "melenchon",
 "premier tour", "campagne", "jadot", "septennat", "#electionpresidentielle2022",
 "melenchoniste", "matignon", "legislative", "lrem", "lfi"]
```

Cette version de l'étiquetage semblait la plus aboutie avec, en général, une petite moitié des tweets étiquetés en lien avec l'élection et un étiquetage qui semblait cohérent sur les exemples que j'ai vérifiés. Avec la première version de l'étiquetage le classifieur obtenu avait d'excellents scores mais prédisait en fait seulement si un tweet était issu d'un des comptes de la liste et non s'il était en lien avec l'élection. L'annexe 5 contient le programme réalisant l'étiquetage dans sa version la plus aboutie.

D) Résultats

Différents paramètres influent sur l'entrainement du classifieur et donc sur la manière dont il fait ses prédictions. Les paramètres principaux sont la méthode utilisée pour obtenir une

représentation numérique des données et la manière dont sont sélectionnées et étiquetées les données. On présente dans le tableau ci-dessous les principaux résultats obtenus.

Données Représentation		TF-IDF	Word2Vec sans traitement de texte	Word2Vec avec traitement de texte
Cas 1	Score train	0.999	0.999	0.999
	Score test	0.947	0.901	0.920
	Score données main	0.793	0.788	0.793
Cas 2	Score train	0.995	0.997	0.993
	Score test	0.840	0.686	0.723
	Score données main	0.705	0.680	0.702
Cas 3	Score train	1.000	0.999	0.999
	Score test	0.800	0.806	0.810

Description des cas :

- Dans le cas 1 les données sont 100 000 tweets issus de 10 journées différentes et l'étiquetage a été réalisé avec la dernière méthode présentée dans le paragraphe précédent.
- Dans le cas 2 les données sont les mêmes mais on a retiré tous les tweets qui apparaissaient plusieurs fois, l'étiquetage a été réalisé de la même manière que pour le cas précédent.
- Dans le cas 3 les données sont 2 000 tweets issus d'une seule journée et qui ont été étiquetés à la main.

Le cas 1 correspond aux résultats les plus aboutis présentés dans les paragraphes précédents.

Description des colonnes :

- Dans la première colonne : la représentation numérique des données est réalisée à partir de la matrice des scores TF-IDF à laquelle on a appliqué une réduction de dimensionnalité (PCA) pour ne garder que 225 caractéristiques.
- Dans les deuxième et troisième colonnes : la représentation numérique des données est réalisée avec l'approche Word2Vec implémentée dans le module Spacy.
    - Pour la deuxième colonne le texte des tweets n'a pas été modifié avant d'être passé au modèle Word2Vec.
    - pour la troisième la même fonction de traitement des textes utilisée dans le premier cas a été appliquée sur chaque tweet au préalable.

Autres remarques :

- La mention « score données main » correspond au score obtenu par le classifieur correspondant sur les données étiquetées à la main.
- Dans tous les cas le classifieur utilisé était un classifieur Random Forest.

Principaux résultats :

- Avec seulement 2000 tweets le score de test est faible : le prédicteur n'est pas capable de généraliser car il n'a pas été entrainé sur suffisamment de données. Lorsqu'il y a plus de données les scores sont bien meilleurs.

- Il est étonnant de remarquer que lorsque l'on retire les tweets qui apparaissaient plusieurs fois dans les données le prédicteur obtenu est significativement moins bon. Je n'ai pas d'explication à ce phénomène.
- Les meilleurs scores sont obtenus avec l'approche TF-IDF mais les résultats sont presque semblables avec Word2Vec. En fait, le modèle utilisé a été entrainé sur des textes issus, principalement, de journaux. Le format des tweets étant différent, il est normal qu'il soit moins performant ici.
- Comme on pouvait s'y attendre, le prédicteur est plus performant lorsque l'on traite le texte des tweets pour ne garder que les termes porteurs de sens.

En fait un grand nombre de paramètres ont un impact que je n'ai pas évalué sur les résultats obtenus : le nombre de composantes conservées après la réduction de dimension, les différents volumes de données d'entrainement et les différents types d'étiquetage ou encore le type de classifieur utilisé. Les résultats présentés ici correspondent à la procédure la plus aboutie. Ils visaient surtout à avoir une idée de la variation des résultats en fonction des données et de l'étiquetage et à évaluer les performances selon l'approche choisie, TF-IDF ou Word2Vec. Dans un projet complet d'analyse de données, la réalisation d'un tel classifieur ne constitue qu'une étape dans le processus visant à obtenir des informations à partir des données.

Finalement le classifieur avec les meilleurs résultats a été obtenu pour un classifieur Random Forest entrainé sur des données variées, issues de journées différentes, en quantité assez importante, 100 000 tweets, dont la représentation numérique a été obtenue à partir d'une matrice des scores TF-IDF dont les dimensions ont été réduites par un algorithme de PCA. L'étiquetage de ces données d'entrainement a été réalisé automatiquement, principalement avec des mots-clés.

VI/ Conclusion

Ce stage m'a permis de découvrir des techniques spécifiques du Machine Learning et de l'analyse des données, dont certaines ayant un lien direct avec les cours suivis cette année, comme la PCA comprenant un calcul de vecteurs propres. J'ai pu réaliser une démarche complète de traitement des données, en partant de leur collecte, et comparer les résultats donnés par différentes méthodes d'analyse. J'ai appris de nombreuses techniques pratiques qui pourront me servir dans le cadre de mes études en informatique.

Je regrette de n'avoir pas eu le temps d'aller plus loin dans l'analyse des données. Certaines des analyses que j'aurais pu faire ensuite comportaient l'utilisation de *clustering*, méthode d'apprentissage non-supervisée permettant de regrouper les données ayant des similarités que l'on peut ensuite analyser. J'aurais également pu réaliser une analyse d'opinion, en particulier sur les candidats à l'élection présidentielle, pour déterminer quelle opinion est représentée sur Twitter. Enfin il aurait été intéressant de créer une interface graphique permettant de visualiser les données, notamment les liens entre comptes Twitter pour visualiser quels comptes interagissent entre eux et si ces interactions sont liées à des opinions partagées. Ce type d'analyse sur des situations politiques est mis en œuvre par des chercheurs de l'Institut des Systèmes Complexes qui ont fondé le site Le Politoscope[10]. Avec plus de temps mon objectif aurait été de réaliser ce type d'analyse et de représentations des données.

[10] https://politoscope.org/

Modules

Module	Utilisation	Documentation
Tweepy	Implémentation Python de l'API Twitter, utilisé pour l'extraction des données	https://www.tweepy.org/
json	Module Python pour manipuler des fichiers au format JSON, utilisé pour enregistrer et récupérer les données dans des fichiers json	https://docs.python.org/fr/3/library/json.html
Spacy	Module Python implémentant des techniques de NLP grâce à des modèles entrainés sur un grand nombre de texte, utilisation du modèle « fr_core_news_md » pour les opérations de lemmatisation	https://spacy.io/ Modèles en français : https://spacy.io/models/fr
NLTK	Module Python implémentant des techniques de NLP, utilisé pour obtenir une liste de stopwords en français	https://www.nltk.org/
Naya	Module Python utilisé pour réaliser du streaming JSON	
Scikit-Learn	Principal module utilisé, il contient les implémentations en Python de nombreux algorithmes de Machine Learning, utilisé pour calculer la matrice des scores TF-IDF, réaliser la réduction de dimensions (PCA) et entrainer un classifieur Random Forest	https://scikit-learn.org/stable/index.html Matrice des scores TF-IDF : voir TfidfVectorizer PCA : voir TruncatedSVD Classifieur : voir RandomForestClassifier

Glossaire

API : Application Programming Interface ou Interface de Programmation d'Application

NLP : Natural Language Processing ou Traitement Automatique des Langues

TF-IDF : Term Frequency – Inverse Document Frequency

PCA : Principal Component Analysis ou Analyse en Composantes Principales

Bibliographie

[1] H. P. Luhn. « A Statistical Approach to Mechanized Encoding and Searching of Literary Information », 1957. https://doi.org/10.1147/rd.14.0309.

[2] Sparck Jones, Karen. « A Statistical Interpretation of Term Specificity and its Application in Retrieval », 1972. https://doi.org/10.1108/eb026526.

[3] Pearson, Karl. « LIII. On lines and planes of closest fit to systems of points in space », 1901. https://doi.org/10.1080/14786440109462720.

[4] Hotelling, Harold. « Relations Between Two Sets of Variates », 1936. https://www.jstor.org/stable/2333955?origin=crossref

[5] Mikolov, Tomas. « Distributed representations of words and phrases and their compositionality », 2013. https://arxiv.org/abs/1310.4546

[6] Mikolov, Tomas. « Efficient Estimation of Word Representations in Vector Space », 2013. https://arxiv.org/abs/1301.3781

Annexes

Les annexes se trouvent sur GitHub à l'adresse : https://github.com/SimCPES/Stage_Twitter.git