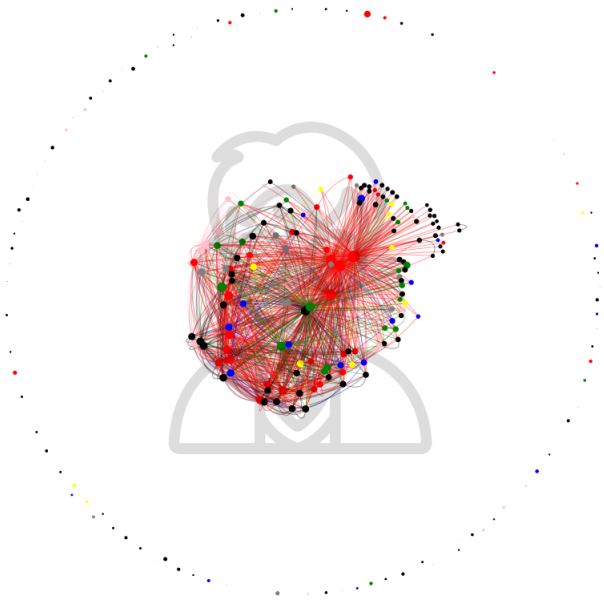
CAPES NSI 2021

Deuxieme épreuve d'admission

Epreuve sur dossier

Célian RINGWALD

Activité pédagogique : le réseau d'Harry Potter



Le réseau social d'Harry Potter

Durée : 120 min
Niveau : fin Première NSI - début Terminale NSI
Matériel Numérique : Depot github
Environement : Python - Jupyter - matplotlib - csv - networkx - scikit-learn
Notions :
- Réseaux sociaux et de graphes [2° SNT]
- Manipuler des tableaux - Algorithme des plus proches voisins [1° NSI]
- Manipuler des graphes - Analyser un graphe [Tle NSI]

Pré-requis

dépot github

La totalité du matériel numérique nécéssaire à cette activité est disponible dans ce dépot git : `https://github.com/datalogism/Capes_NSI`.
Vous pouvez le récupérer directement sur votre machine en ouvrant votre console et en entrant la commande suivante :

```
git clone https://github.com/datalogism/Capes_NSI.git chemin_de_mon_dossier
```

Vous y trouverez :
- les données à exploiter
- les notebooks à prendre en main
- des documents complémentaires

Environement

Le présent tutoriel demandera d'utiliser un notebook Jupyter, si ce n'est déjà fait installez jupyter : `https://jupyter.org/`.
Il demandera aussi d'avoir installé les librairies suivantes : *matplotlib*, *csv*, *networkx*, *scikit-learn*.
On rappelle que pour installer une nouvelle librairie python il faut lancer la commande suivante dans votre console :

```
pip install csv matplotlib scikit-learn networkx
```

Introduction

Cette activité a été conçue comme un pont entre les concepts aperçus en 2^{nde} et ce que vous allez voir en T^{ale}, nous mobiliserons ici les compétences qui vous avez acquises en 1^{ere} : notamment en programmation (variables, boucles, comparaison, fonctions...), ainsi que votre habilité en traitement de données en table. Vous avez pu découvrir les années passées que nous pouvions représenter un réseau social à travers un graphe. Comme vous avez pu le comprendre ces structures permettent de conceptualiser des interactions entre des objets physiques ou conceptuels : que ce soit les réseaux d'infrastructure d'Internet ou un réseau de personnes. Son utilisation est très courante dans dans de nombreux autres domaines : sciences naturelles, médicales, physique, chimie... Elle est aussi aujourd'hui de plus en plus présente de le domaine de la recherche en histoire, en littérature, car ces disciplines travaillent généralement sur des textes issus de documents d'archive ou d'artefacts culturels, dont sont extrait des réseaux d'auteurs, de lieux, d'événements qui contractualisent des questions d'études. Cette utilisation de l'informatique en sciences humaines est aujourd'hui regroupée sous la désignation d'"Humanités Numériques". Nous allons aujourd'hui travailler sur un graphe issu d'un travail de "fouille de textes" sur la saga d'Harry Potter.

Pour commencer rappelons que dans un graphe les sommets notés V, et les arêtes représentant les relations entre ces noeuds sont notés E. Ici les sorciers sont des sommets et sont reliés en fonction de s'ils apparaissent ou non ensemble dans des passages de la saga. Ces données sont issues de la continuité d'un travail durant lequel j'ai extrait le réseau des personnages pour en analyser la dynamique émotive du premier tome. Nous avons cette fois-ci analysé la totalité des volumes en repérant les références des personnages par paragraphe. Ainsi le graphe obtenu est un **graphe non orienté**, les noeuds de celui-ci sont pondérés par le nombre de références dans le texte complet d'un personnage, les liens entre chaque noeud sont eux pondérés en fonction du nombre de paragraphes où les personnages sont référés ensembles.

Nous avons pu voir que Javascript était un langage du web, et qu' il permet l'échanges d'informations entre client et serveur. Mais il permet aussi par extension de réaliser des interfaces dynamiques. Représenter des données à l'aide de cette technologie permet de réaliser des applications favorisant une interaction avec l'affichage : ce qui est très pratique surtout quand on travaille avec beaucoup d'informations. Représenter l'information est parfois un véritable défi, réaliser des interfaces de ce genre s'appelle faire de la visualisation de données, autrement dit *data visualisation* ou *dataviz*.

— Vous pouvez ici consulter une version dynamique, interactive et complète du graphe réalisée à l'aide une librairie graphique Javascript appelée Vis.js : ICI.
— Vous pouvez consulter ici quelques statistiques concernant le découpage de la saga : ICI cette dataviz a été réalisée à l'aide de la libraire Chart.js.

Nous allons donc dans un premier temps analyser les noeuds de notre réseau, sous la forme d'un tableau (partie I). Nous implémenterons ensuite notre premier graphe grâce à la librairie *Networkx*, nous l'afficherons ensuite en faisant apparaitre les 'Maisons' de chaque noeud (partie II). Nous réaliserons ensuite quelques mesures sur notre graphe afin de l'analyser selon le spectre de la Théorie du petit monde (partie III). Nous filtrerons ensuite ce graphe et essaierons ensuite de mettre en oeuvre un algorithme d'apprentissage automatique sur la demande du ministère de la magie. Nous utiliserons pour cela l'algorithme des **k-plus proches voisins** sur une version tabulaire de ce graphe.

Partie I : Analyse des personnages

Ouvrez le notebook `HarryPotter_Part1.ipynb` :

1. Ouvrez le fichier `perso_list_with_desc_clean.csv`, qui contient un descriptif de tout les personnages et chargez le à l'aide de la librairie *csv*. Nous avons vu plusieurs moyens d'ouvrir un fichier tableur avec Python. Utilisez celle qui permet de lire celui-ci et de l'enregister dans une variable comme un tableau de dictionnaires ordonnés que vous appellerez `persDetails`.
   Voilà à quoi ressemble `perso_list_with_desc_clean.csv` :

Id	Nom	Maison	type	nb_occ
42	Harry Potter	Gryffondor	sorcier	15429
79	Ron Weasley	Gryffondor	sorcier	5656
44	Hermione Granger	Gryffondor	sorcier	4897
3	Albus Dumbledore	Gryffondor	sorcier	2896
...	...	...	...	...

2. Combien de personnages compte ce tableau ?
3. Chaque personnage y est décrit à travers différentes clefs, ce qui devraient correspondre à l'en-tête de notre fichier csv. Vérifier que c'est bien le cas dans `persDetails`.
4. Quelle est le type des données que décrit chaque clef ?
5. Nous avons vu en algorithmique comment étaient réalisés les tris, python propose des solutions plus complexes, mais plus rapide pour effectuer cela sur des tableaux de dictionnaires. Triez donc `persDetails` en fonction de la clef décrivant le nombre d'apparitions de chaque personnage dans la saga par ordre décroissant. Faites attention le type des données influence la manière dont est réalisé le tri, celui-ci doit se faire du point de vue numérique et pas *lexicographique*.
6. Quels sont les 20 personnages principaux de l'histoire ?
7. Construire une fonction nommée `describe` qui prend en entrée un tableau de dictionnaires et une clef de celui-ci. Elle devra réaliser une opération d'agrégation comptant le nombre de sorciers pour chaque caractéristique de cette clef. Concrètement, elle devra pouvoir permettre de compter le nombre de sorciers pour chaque *'Maison'*.
8. Utilisez la fonction `describe` sur les clefs *'Maison'* et *'type'*. Que pouvez-vous en dire ?
9. Créez un dictionnaire clefs-valeur permettant d'accèder aux données d'un personnage à travers son id : nommez-le `id_pers`.

Partie II : Le réseau des sorciers

Maintenant que nous connaissons un peu mieux nos personnages nous allons voir comment ceux-ci sont liés au sein de l'histoire. Afin de prendre en main facilement le concept de réseau nous allons sans trop nous arrêter sur son implémentation et utiliser directement une librairie *networkx* pour construire le graphe et l'analyser. Nous verrons l'année prochaine comment implémenter un graphe en faisant de la programmation orientée objet, et comment le parcourir à l'aide de fonctions récursives.

1. Lancez les commandes permettant d'importer *networkx* et chargeant le fichier contenant les liens entre les noeuds.

Voilà à quoi ressemble `edges_clean.csv`, c'est une liste de liens entre les personnages, qui sont identifiés à travers leurs **Id**. Chaque lien a une intensité définie par le nombre d'apparitions de chaque personnage ensemble dans un paragraphe :

from	to	value
42	44	0,17
42	44	1
42	45	0,23
...	...	...

2. Créez une fonction `getColor` prenant en entrée une ligne du tableau et qui renvoi la chaine de caractère 'grey' par défault. Faites une recherche sur la couleur de chaque 'Maison' et écrivez une chaine conditionnelle renvoyant une chaine de caractère liée à la couleur (écrite en anglais) dont dépend chaque maison. Enfin Si un sorcier n'est pas étudiant à poudlard (*'type'="sorcier"*) renvoyez 'black'.
3. Lancez la commande permettant de construire l'objet `G`, de la classe `nx.Graph()` à partir de la liste de liens `edges` et des personnages du tableau `persDetails`.
4. Lancez maintenant la commande permettant d'afficher ce graphe. A première vue que pouvez vous dire de celui-ci ?

Partie III : Quelques caractéristiques du Réseau des sorciers

Vous avez pu voir en seconde quelques caractéristiques permettant de décrire succitement un réseau social. Vous devez donc connaitre le concept de **centralité** permettant de retranscrire l'importance du noeud dans le réseau. Il existe plusieurs moyens de mesurer la centralité d'un noeud dans le réseau. La *centralité de degré* est calculé comme étant le nombre de liens capté par un noeud donné rapporté au nombre de liens maximum captés par un noeud dans le graphe. Nous allons pouvoir grace à *networkx* pouvoir obtenir directement cette mesure.

1. Calculez cet indicateur pour chacun des noeuds du graphe en vous aidant de la documentation de la librairie. Sachant que la *centralité de degré* est appelé *degree centrality* en anglais.
2. En vue de cet indicateur affichez les six personnages les plus centraux de l'histoire.
3. En vue des mesures proposées par *networkx* la quelle permet d'obtenir le diamètre du graphe ? Calculez en fonction le diamètre de ce graphe. Que pouvez-vous dire de ce résultat ?

La théorie des six degré de séparations dit qu'il existe une chaine d'environ six relations me menant à n'importe qui sur la terre. On imagine bien que dans le cadre d'une aventure telle que celle d'Harry Potter, le réseau social doit être très resserré, d'autant que les personnages sont pour la plupart issus d'un microcosme scolaire : Poudlard, où tout les élèves sont régulièrement rassemblés dans la salle commune de l'école. Tous ceux-ci doivent s'être au moins vus une fois.

4. Effectuez une recherche concernant le degré de séparation sur Facebook et Twitter. Que vallent-ils ? Utilisez la fonction `degree_of_separation` sur G le réseau des sorciers que nous étudions et affichez le résultat obtenu. Comparez cela aux degrés de séparation moyen de réseaux tels que Twitter et de Facebook. Que pouvez-vous en dire ?
5. Nous comprenons que notre graphe est un petit peu biaisé, car il retranscrit là seulement le fait que deux personnes soient présentes dans un même paragraphe. Ils pourraient être dans ces passages en train de se battre, ou bien de discuter... Comment aurions-nous pu obtenir une version plus réaliste du réseau des sorciers ?

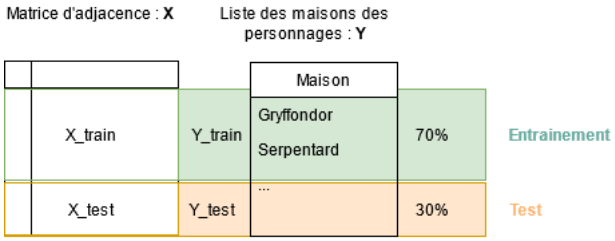
Partie IV : Algorithme d'apprentissage du Choixpeau

Le ministère de la magie souhaite imposer au Choixpeau un algorithme d'apprentissage automatique basé sur les relations des élèves avec les personnages historiques de l'école, afin de prévenir les risques d'une éventuelle reprise du controle des mangemorts. Le ministère souhaite alors dissoudre la maison des Serpentards et ne pu admettre des élèves pouvant se reveller dangereux à l'école.

Pour cela il propose d'utiliser le réseau que nous venons de décrire pour deviner à quelle maison pourrait être rattaché un personnage ne faisant pas partie de Poudlard et ceci sur la base de ces relations sociales avec les personnages de la saga. Cette propriété qui fait que nous ressemblons à notre cercle social est appelée l'"homophilie".

Ce problème peut être modélisé sous la forme du problème de classification, que nous allons résoudre grace à l'algorithme du plus proche voisin, en notant C un ensemble de classes (ici les maisons), un ensemble E de personnages où chaque élèment de E possède une classe $c(e)$) de C. Afin de pouvoir utiliser cet algorithme nous allons représenter le graphe sous la forme d'une matrice d'adjacence, où les lignes et colonnes représentent les noeuds du graphe ordonnées dans le même ordre. Ce tableau de tableau de taille $N \times N$, où N nombre de personnages de E. Aux intersections de chaque couple de noeuds i et j sont enregistrés des entiers. Si un couple (i, j) de noeuds n'est pas lié alors $matrix[i, j] = 0$. S'il existe un lien entre ces deux noeuds, $matrix[i, j]$ est égal à l'intensité de cette liaison comprise entre zéro et un. Cette mesure est calculée sur la base de nombres occurrences communes de chaque couple dans la saga. Cette matrice est symétrique donc $matrix[i, j] = matrix[j, i]$, car notre graphe n'est pas dirigé.

1. Dans un premier temps nous allons nous concentrer sur les personnages ayant une Maison définie et apparaissant au moins 5 fois dans la saga. Créez une liste nommée `pers_ok` qui regroupera les id des personnages que l'on souhaite conserver. Affichez le nombre de sorciers obtenus dans cette liste.
2. Créez une variable `matrix` tableau de tableau que vous construirez **par compréhension** de la taille adaptée, et dont toutes les cases sont égales à 0.
3. Parcourrir la liste de liens `edges` et ajouter à `matrix` le valeur de l'insité du lien entre chaque couple de personnages (i, j).
4. Chargez les fonctions de *scikit-learn* : `KNeighborsClassifier`, qui implémente l'algorithme du plus proche voisin, `classification_report`, qui permet résumé des résultats de la classification réalisée.
5. Créez une variable `X`, qui contient `matrix`, puis créez une liste `Y` qui contient les maisons des personnages de la liste `pers_ok`. Aidez vous du dictionnaire `id_pers` pour construire cette liste.
6. Nous allons donc utiliser la matrice X et la liste Y pour concevoir notre système de prévision de maison. Nous allons diviser en deux ces deux jeux de données afin d'utiliser 70% de celles-ci pour apprendre des relations des personnages et de leurs maison. Le reste du jeux de données sera utilisé comme *"jeux de données test"* afin de réaliser une "validation croisée" nous permettant de mesurer les performances de nos prévisions. Lancez la commande permettant de découper le jeux de données d'une telle manière.



7. Vous avez pu comprendre en cours que l'algorithme des plus proches voisins se basait sur un calcul de distances entre les caractéristiques fournies par une entrée comparée à une base de données définie à l'avance. Ici ce calcul se base sur la différence en terme de relations que possède un nouvel individu donné avec les sorciers ayant servis à l'apprentissage. Les k-plus proches personnages à notre entré sont donc alors invoqués et la classe majoritaire de cette sous-population est affectée à la nouvelle entrée. Lancez la commande lançant l'apprentissage de cet algorithme.
8. Créez la liste de prévisions `y_pred` obtenues sur `X_test` à l'aide de la fonction `predict` du modèle.
9. Lancez la commande permettant de résumer les performances du modèle. Quand est-il de la précision pour la maison Serpentard ? Quelle précision à le modèle global ?
10. Créez une liste de personnages à classer : `pers_to_classify`, contenant les ids de *Dudley Dursley*, le cousin d'Harry Potter, *Dobby* l'elfe de maison, *Viktor Krum* le joeur de Quidditch, *Quirrell* le nouveau professeur de Poudlard du premier tome, *Pomfrey* l'infirmière de l'école et *Antonin Dolohov*, mangemort qui a assassiné *Remus Lupin*.
11. Que pouvez-vous dire des résultats de cet algorithme ? Pourquoi les performances de ce modèle sont-elles d'après vous aussi faible ? Que pouvez-vous dire tout simplement de l'utilisation d'un tel système dans le fonctionement de Poudlard ? Quels problèmes pouraient entrainer un tel système ?

Références

[1] Thibaut Balabonski, Sylvain Conchon, Jean-Christophe Filliâtre, Kim Nguyen (2020). Numérique et sciences informatiques - Terminale *Ellipses*, p189-218.

[2] Thibaut Balabonski, Sylvain Conchon, Jean-Christophe Filliâtre, Kim Nguyen (2020). Numérique et sciences informatiques - 1ere *Ellipses*, p175-184.

[3] John R. Ladd, Jessica Otis, Christopher N. Warren, and Scott Weingart (2020). Exploring and Analyzing Network Data with Python, *Programming Historian*, p175-184.

[4] Programme de sciences numériques et technologie de seconde générale Officiel de l'Education Nationnale, spécial n° 1 du 22 janvier 2019

[5] Programme de numérique et sciences informatiques de première générale Officiel de l'Education Nationnale, spécial n°1 du 22 janvier 2019

[6] Programme de numérique et sciences informatiques de terminale générale Officiel de l'Education Nationnale, spécial n° 8 du 25 juillet 2019