Royaume du Maroc
Ministère de l'Enseignement Supérieur
et de la Recherche Scientifique

La Faculté Pluridisciplinaire de Nador

الكلية متعددة التخصصات الناظور
ⵜⴰⵖⵡⴰⵙⵜ ⵜⴰⵣⵔⵣⵎⵜ ⵏ ⵍⵏⴰⴹⵓⵔ
Faculté Pluridisciplinaire de Nador

:
Master Sciences des Données et Systèmes Intelligents (SDSI)

RAPPORT MINI PROJET

présenté à

La Faculté Pluridisciplinaire de Nador

par

Ahmed Hamouni

Encadré par

EL Ouahabi Safâa

"l'Analyse de Réseaux sociaux avec NetworkX en Python"

TABLE DES MATIÈRES

INTRODUCTION GÉNÉRALE

Dans sa forme la plus basique,un réseau peut être envisagé comme un ensemble complexe de connexions entre différents éléments,représentés par des nœuds et des liens.Cette conception s'étend bien au-delà du domaine informatique pour englober un large éventail de phénomènes interconnectés dans le monde réel.L'essence même d'un réseau réside dans sa capacité à capturer et à représenter des relations, des interactions et des flux entre des entités diverses,qu'elles soient des individus,des objets physiques,des organisations ou même des concepts abstraits.

L'analyse des réseaux sociaux (SNA) émerge comme une discipline puissante qui utilise la théorie des réseaux et des graphes pour explorer la structure sous-jacente des interactions sociales.Cette approche trouve des applications dans divers domaines,allant des sciences sociales et de la psychologie aux affaires,à la biologie et à la technologie de l'information.

Dans le contexte des médias sociaux,par exemple,les réseaux sociaux sont devenus des terrains fertiles pour étudier les schémas de connexion entre individus,la propagation des informations, l'influence sociale et même les dynamiques de comportement collectif.Les plateformes telles que Facebook,Twitter,LinkedIn et Instagram fournissent des données riches et vastes qui peuvent être analysées pour découvrir des tendances,desommunautés et des clusters d'intérêt commun.

Au sein des organisations,l'analyse des réseaux sociaux peut révéler des structures informelles de communication et de collaboration qui peuvent influencer la prise de décision, la diffusion des connaissances et la productivité. Comprendre qui communique avec qui,qui occupe des positions centrales dans le flux d'informations,et qui agit en tant que ponts entre différents départements ou équipes peut aider les gestionnaires à optimiser les flux de travail,à favoriser l'innovation et à renforcer la cohésion organisationnelle.

Même dans le règne animal,lechercheurs utilisent des techniques d'analyse des réseaux pour étudier les schémas sociaux,les hiérarchies de dominance,les comportements de coopération et les stratégies de survie.Des études sur les troupeaux d'éléphants aux colonies de fourmis, en passant par les bancs de poissons et les groupes de primates,fournissent des informations précieuses sur la manière dont les animaux interagissent, ommuniquent et collaborent pour répondre à leurs besoins biologiques et sociaux.

En somme,l'analyse des réseaux sociaux offre un cadre conceptuel et méthodologique puissant pour étudier une variété de phénomènes complexes dans divers contextes.En cartographiant les relations entre les entités,en identifiant les acteurs clés et en analysant les dynamiques de réseau,les chercheurs et les praticiens peuvent acquérir des perspectives nouvelles et approfondies sur la structure et le fonctionnement des systèmes sociaux,offrant ainsi des insights précieux pour la prise de décision,la résolution de problèmes et la conception de stratégies efficaces.

Chapitre

1 NetworkX

Sommaire

1.1 Introduction

1.1.1 Qu'est-ce que NetworkX ?

NetworkX est une bibliothèque open source pour la manipulation, l'analyse et la visualisation des réseaux et des graphes en Python. Elle fournit un large éventail de fonctionnalités et d'outils pour travailler avec des données basées sur des graphes, qu'il s'agisse de réseaux sociaux, de réseaux biologiques, de réseaux de transport ou d'autres types de relations complexe

1.1.2 Les principales caractéristiques de la bibliothèque NetworkX

Voici une description détaillée des principales caractéristiques de la bibliothèque NetworkX :

1.Structures de données de graphe flexibles : NetworkX permet de créer, manipuler et analyser des graphes dirigés et non dirigés. Les graphes peuvent être composés de nœuds (vertices) et d'arêtes (edges) qui représentent les relations entre les nœuds. NetworkX propose plusieurs types de graphes, notamment des graphes simples, multigraphes (graphes pouvant avoir plusieurs arêtes entre les mêmes nœuds) et graphes dirigés.

2.Prise en charge d'une large gamme d'algorithmes : NetworkX offre une collection étendue d'algorithmes pour l'analyse des graphes. Cela inclut des algorithmes de recherche de chemins, de centralité (comme la centralité de proximité et

3.Importation, exportation et intégration de données : NetworkX permet d'importer, d'exporter et d'intégrer des données graphiques à partir de diverses sources.

4.Visualisation graphique : La bibliothèque comprend des outils pour la visualisation graphique des données. NetworkX permet de représenter les graphes sous forme de dessins à l'aide de différentes dispositions, notamment des dispositions circulaires, en coquille, en force-directed, etc. Il est également possible de personnaliser l'apparence des nœuds, des arêtes et des étiquettes pour créer des visualisations claires et informatives.

5.Intégration avec d'autres bibliothèques : NetworkX s'intègre bien avec d'autres bibliothèques scientifiques et de traitement des données en Python, telles que NumPy et Pandas. Cela permet d'effectuer des analyses plus avancées en combinant les fonctionnalités de NetworkX avec d'autres outils statistiques et de visualisation.

6.Communauté active et support continu : NetworkX est une bibliothèque populaire avec une communauté active de développeurs et d'utilisateurs. Cela se traduit par un support continu, des mises à jour régulières et une documentation complète, ce qui facilite son utilisation et son apprentissage.

7.Conclusion : NetworkX est une bibliothèque incontournable pour travailler avec des réseaux et des graphes en Python. Que vous ayez besoin de modéliser des réseaux sociaux, des réseaux biologiques, des réseaux de transport ou d'autres types de relations complexes, NetworkX fournit les outils nécessaires pour analyser, visualiser et comprendre vos données graphiques.

1.1.3 Installation de NetworkX

Installation avec pip

Installez la version stable la plus récente de networkx avec pip :

```
$ pip install networkx
```

Pour mettre à niveau vers une version plus récente, utilisez le –upgradedrapeau :

```
$ pip install –upgrade networkx
```

Si vous ne disposez pas des autorisations nécessaires pour installer le logiciel à l'échelle du système, vous pouvez installer NetworkX dans votre répertoire utilisateur à l'aide du –userdrapeau :

```
$ pip install –user networkx
```

Vous pouvez également télécharger manuellement le référentiel networkx depuis GitHub ou PyPI. Pour installer l'une de ces versions, décompressez-la et exécutez la commande suivante à partir du répertoire source de niveau supérieu :

```
$ pip install
```

Installation avec Anaconda

NetworkX est actuellement disponible via la plateforme de distribution Anaconda. Miniconda n'est pas fourni avec NetworkX par défaut.

Vous pouvez installer la dernière version de NetworkX avec :

```
$ conda install networkx
```

Si vous souhaitez mettre à jour votre version actuelle de NetworkX, exécutez :

```
$ conda update networkx
```

1.2 Concepts de base associées à la bibliothèque networkX

1.2.1 les graphes

Un graphe, dans son essence, est une représentation abstraite et puissante des relations entre des entités diverses, offrant un moyen efficace de modéliser des systèmes complexes et de comprendre les interconnexions qui les sous-tendent. Cette structure de données trouve des applications dans une multitude de domaines, allant des mathématiques et de l'informatique à la biologie, la sociologie, l'économie et bien d'autres encore.

Au cœur d'un graphe se trouvent deux éléments fondamentaux : les nœuds (ou sommets) et les arêtes (ou liens). Les nœuds représentent les entités individuelles, tandis que les arêtes définissent les relations entre ces entités. Ces relations peuvent être de nature variée : elles peuvent indiquer des connexions physiques, des interactions sociales, des dépendances fonctionnelles,

des similitudes structurelles, des flux d'information, ou toute autre forme de lien entre les entités.

La structure d'un graphe peut être dirigée ou non dirigée. Dans un graphe dirigé, chaque arête a une orientation spécifique, indiquant une relation asymétrique entre deux nœuds. Par exemple, dans un réseau routier, les arêtes dirigées pourraient représenter des routes à sens unique. En revanche, dans un graphe non dirigé, les arêtes ne sont pas orientées, ce qui signifie que la relation entre deux nœuds est symétrique. Par exemple, dans un réseau social, une amitié entre deux personnes serait souvent représentée par une arête non dirigée, car elle est généralement réciproque.

Un graphe peut également être pondéré ou non pondéré. Dans un graphe pondéré, chaque arête est associée à une valeur numérique, appelée poids, qui représente une mesure quantitative de la force ou de l'intensité de la relation entre les nœuds connectés. Par exemple, dans un réseau de transport, le poids d'une arête pourrait représenter la distance entre deux villes. Dans un graphe non pondéré, toutes les arêtes ont la même valeur par défaut, ce qui signifie que la force des relations n'est pas prise en compte.

Les graphes peuvent également être utilisés pour représenter des structures complexes telles que les réseaux sociaux, les réseaux de transport, les réseaux biologiques, les réseaux informatiques, les réseaux de neurones, et bien d'autres encore. En modélisant ces systèmes sous forme de graphes, les chercheurs et les praticiens peuvent analyser leur topologie, leur connectivité, leurs propriétés émergentes et leur dynamique, ce qui leur permet de découvrir des modèles cachés, de prédire des comportements futurs et de concevoir des interventions stratégiques.

En résumé, les graphes sont des outils conceptuels polyvalents et flexibles qui permettent de représenter et d'analyser les relations entre des entités complexes dans une variété de contextes. Leur utilisation offre un cadre formel et unifié pour étudier les systèmes interconnectés, facilitant ainsi la compréhension et la manipulation de la complexité du monde qui nous entoure.

NetworkX permet la création facile de graphes, qu'ils soient dirigés ou non dirigés, pondérés ou non pondérés, à l'aide d'une syntaxe simple et intuitive. Cette bibliothèque offre également une variété d'algorithmes pour l'analyse de graphes, permettant aux utilisateurs d'explorer des

caractéristiques telles que la centralité des nœuds, la détection de communautés, la recherche de chemins les plus courts, et bien d'autres métriques et mesures importantes.

1.2.2 Création de graphes

Les nœuds représentent les entités dans le graphe.

nx.Graph() : Crée un graphe non dirigé.

nx.DiGraph() : Crée un graphe dirigé.

G.add_node() : Ajoute un nœud au graphe.

G.add_edge() : Ajoute une arête au graphe.

Les arêtes définissent les relations entre les nœuds.

1.2.3 Visualisation des graphes

cette liste n'est pas exhaustive, mais elle donne un aperçu des principales méthodes et fonctions disponibles dans la bibliothèque NetworkX. Pour plus de détails sur les fonctionnalités et les options, je vous recommande de consulter la documentation officielle de NetworkX : https://networkx.org/documentation/stable/

nx.draw() : Dessine le graphe à l'aide de Matplotlib.

nx.draw_networkx() : Dessine le graphe avec des options de personnalisation.

nx.write_dot() : Exporte le graphe au format DOT pour la visualisation avec Graphviz.

1.2.4 Type des Graphes

GRAPHES NON DIRIGÉS (UNDIRECTED GRAPHS)

```
import networkx as nx
G = nx.Graph()
type(G)
```

```
networkx.classes.graph.Graph
```

GRAPHES MULTI-ARÊTES (MULTIGRAPHS)

```
import networkx as nx
G = nx.DiGraph()
type(G)
```

```
networkx.classes.graph.DiGraph
```

GRAPHES MULTI-ARÊTES (MULTIGRAPHS)

```
import networkx as nx
G = nx.MultiGraph()
type(G)
```

```
networkx.classes.digraph.MultiDiGraph
```

GRAPHES MULTI-ARCS DIRIGÉS (MULTIDIGRAPHS)

```
import networkx as nx
G = nx.MultiDiGraph()
type(G)
```

```
networkx.classes.digraph.MultiDiGraph
```

1.2.5 Liste des méthodes associées à la bibliothèque networkX

Création de graphes

nx.Graph() : Crée un graphe non dirigé.

nx.DiGraph() : Crée un graphe dirigé.

nx.MultiGraph() : Crée un graphe non dirigé avec des arêtes multiples.

nx.MultiDiGraph() : Crée un graphe dirigé avec des arcs multiples.

Ajout d'arêtes et de nœud

G.add_node(node) : Ajoute un nœud au graphe.

G.add_nodes_from(nodes) : Ajoute plusieurs nœuds.

G.add_edge(node1, node2) : Ajoute une arête entre deux nœuds.

G.add_edges_from(edges) : Ajoute plusieurs arêtes.

Suppression d'arêtes et de nœud

G.remove_node(node) : Supprime un nœud du graphe.

G.remove_nodes_from(nodes) : Supprime plusieurs nœuds du graphe.

G.remove_edge(node1, node2) : Supprime une arête entre deux nœuds.

G.remove_edges_from(edges) : Supprime plusieurs

Manipulation des nœuds et des arêtes

G.nodes() : Renvoie les nœuds du graphe.

G.edges() : Renvoie les arêtes du graphe.

G.degree : Renvoie le degré d'un nœud.

1.3 Visualisation rationnelle(nxviz)

1.3.1 Installation

nxviz est un package de visualisation graphique pour NetworkX. Avec nxviz, nous pouvons créer de superbes visualisations graphiques grâce à une API déclarative. Il s'agit d'un logiciel gratuit distribué sous licence MIT.

```
$ conda install -c conda-forge nxviz
```

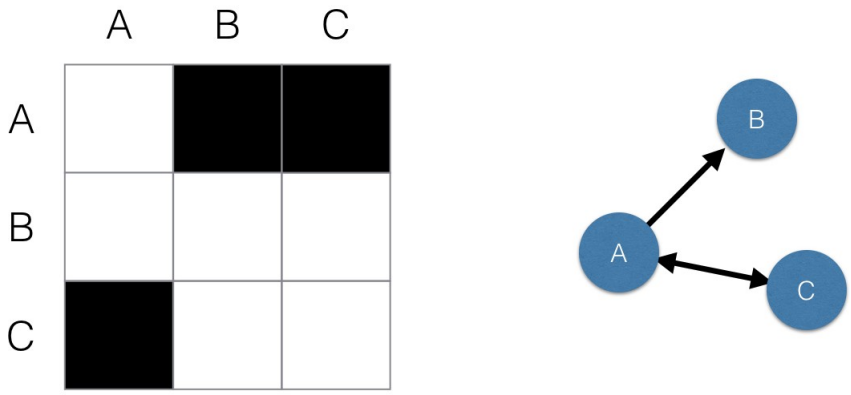
Alternativement, il est également disponible sur [PyPI].

```
$ pip install nxviz
```

1.3.2 Matrix Plots

Une représentation visuelle matricielle des connexions entre les nœuds d'un graphe. -> Visualisation claire des relations et des motifs au sein du réseau.
-> Utile pour identifier des structures spécifiques.

```
from nxviz import MatrixPlot
import matplotlib.pyplot as plt
# Créer une instance de MatrixPlot
matrixplot = MatrixPlot(G)
# Afficher le graphe
matrixplot.draw()
plt.show()
```

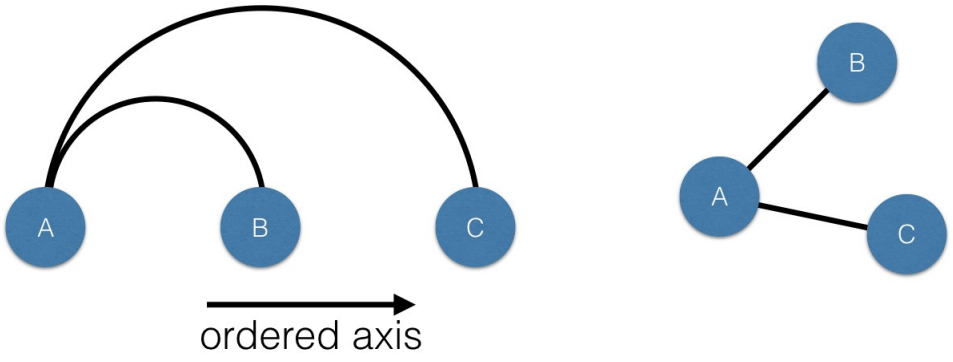


1.3.3 Arc Plots

Un diagramme où les connexions entre les nœuds sont représentées sous forme d'arcs dans un cercle.

-> Met en évidence les liens entre les nœuds de manière circulaire.

```
from nxviz import ArcPlot
import matplotlib.pyplot as plt
# Créer une instance d'ArcPlot
arcplot = ArcPlot(G)
# Afficher le graphe
arcplot.draw()
plt.show()
```



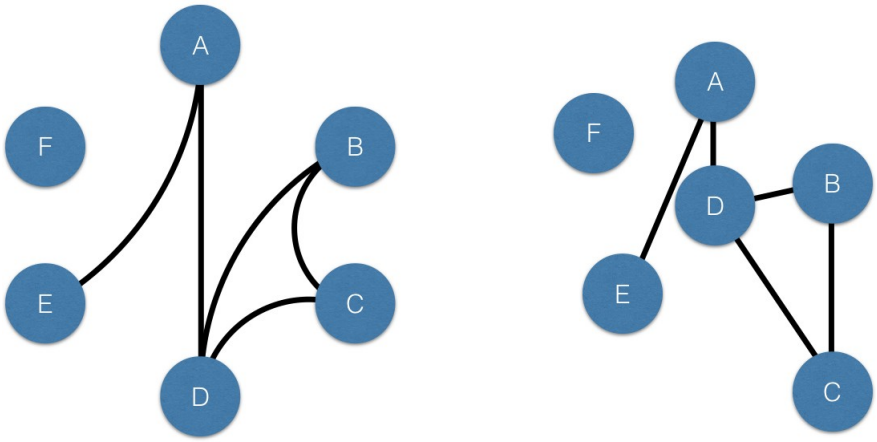
1.3.4 Circos Plots

Une variante d'Arc Plot plaçant les nœuds autour d'un cercle pour une vue circulaire.

-> Offre une perspective circulaire du réseau pour une visualisation compacte.

-> Utile pour identifier des schémas de connexion globaux

```
from nxviz import CircosPlot
import matplotlib.pyplot as plt
# Créer une instance de Circos Plot
circos = CircosPlot(G)
# Afficher le graphe
circos.draw()
plt.show()
```



1.4 Les nœuds importants

En théorie des graphes et en théorie des réseaux, les indicateurs de centralité sont des mesures censées capturer la notion d'importance dans un graphe, en identifiant les sommets les plus significatifs. Les applications de ces indicateurs incluent l'identification de la ou des personnes les plus influentes dans un réseau social, les nœuds clés dans une infrastructure comme internet ou un réseau urbain, et encore des foyers d'infection, qu'ils soient de nature nosocomiales ou superinfecteurs, pour certaines maladies. Des exemples plus ponctuels incluent

les relations de coopération dans les réseaux professionnels, comme les travaux scientifiques menés en commun ou, dans l'industrie cinématographiques, les acteurs ayant joué dans les mêmes film.

Un nœud important est généralement déterminé par des mesures de centralité, qui évaluent l'importance d'un nœud dans un réseau.

.Centralité de degré (Degree centrality).

.Centralité intermédiaire (Betweenness centrality).

1.4.1 Nombre de Voisins

La méthode **G.neighbors(node)** renvoie un itérable sur les voisins du nœud spécifié "node" dans le graphe "G".

list(G.neighbors(node)) renvoie les voisins du nœud "node" dans le graphe "G".

1.4.2 Recherche de Chemins

Fonction has_path

nx.has_path(G, source, target) : vérifie l'existence d'un chemin entre deux nœuds.

Fonction shortest_path

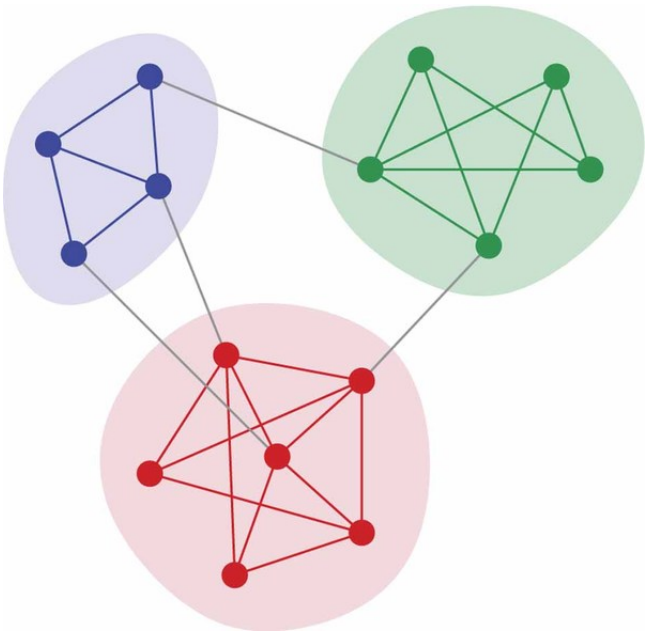
En employant **nx.shortest_path(G, source, target)**, on identifie le trajet le plus court entre deux nœuds.

Fonction all_shortest$_{p}aths$

La fonction **nnx.all_shortest_paths(G, source, target) r** renvoie tous les chemins les plus courts entre deux nœuds.

1.4.3 Sous-graphes

Un sous-graphe est un graphe qui est formé à partir d'un ensemble de nœuds et/ou d'arêtes d'un graphe plus large.



```
subgraph = G.subgraph([1, 2, 3])
```

En créant un sous-graphe, il partage par défaut les données avec le graphe principal. Pour obtenir une copie indépendante, il suffit d'utiliser la fonction **copy**.

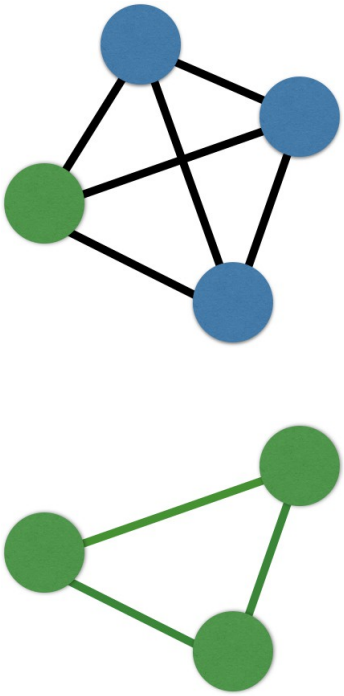
```
subgraph_copy = subgraph.copy()
```

Les sous-graphes offrent une flexibilité dans l'exploration de structures plus petites au sein de graphes plus vastes.

1.4.4 Cliques

Les cliques sont des sous-graphes complets, où chaque nœud est connecté à tous les autres.

La plus simple et complexe Clique : un triangle



1.4.5 Cliques maximales

Une clique maximale est une groupe de nœuds où l'ajout d'un nœud supplémentaire ne maintient plus la propriété de clique.

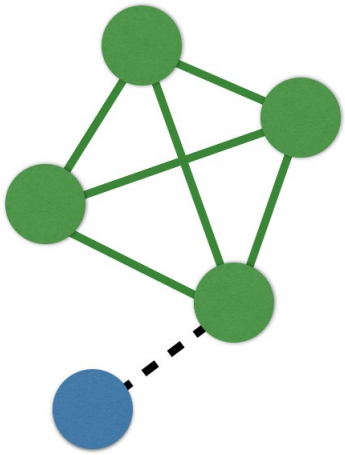
find_cliques Trouve toutes les cliques maximales

Applications :

• identification de communautés

• Trouver des cliques

• Trouver des unions de cliques



1.4.6 Centralité de degré

La centralité de degré représente la forme la plus simple et la plus intuitive de la notion de centralité. Elle est basée sur l'idée que l'importance d'un individu au sein d'un groupe dépend du nombre total de personnes qu'il connaît ou avec lesquelles il interagit directement. Selon cette mesure, déterminer l'importance d'un nœud dans un graphe revient donc à calculer le nombre de ses sommets voisins, ou de manière équivalente, à calculer le nombre de liens qui lui sont incidents. En théorie des graphes, ce nombre est appelé degré du nœud, d'où l'appellation de centralité de degré.

Soit $G = (V, E)$ un graphe d'ordre N représenté par sa matrice d'adjacence A. Dans le cas où le graphe G est non-orienté, la centralité de degré d'un nœud $v_i \in V$ est définie par :

$$C^{\deg}(v_i) = \frac{1}{N-1} \sum_{j=1}^{N} a_{ij}$$

En notation matricielle, le vecteur de la centralité de degré est donné par : $\mathbf{c}^{\deg} = \frac{\mathbf{A} \times \mathbf{1}}{N-1}$, où 1 est un vecteur colonne de dimension N contenant des uns.

Dans le cas où le graphe G est orienté, chaque nœud $v_i \in V$ possède alors deux mesures de centralité de degré : une par rapport aux liens sortants et une par rapport aux liens entrants. Elles sont définies respectivement par :

$$C_{out}^{\deg}(v_i) = \frac{1}{N-1} \sum_{j=1}^{N} a_{ij}; C_{in}^{\deg}(v_i) = \frac{1}{N-1} \sum_{j=1}^{N} a_{ji}$$

En termes de matrices, les vecteurs de centralité de degré sortant et de degré entrant sont donnés respectivement par $\mathbf{c}_{out}^{\deg} = \frac{\mathbf{A}\times\mathbf{1}}{N-1}$ et $\mathbf{c}_{in}^{\deg} = \frac{\mathbf{A}^T\times\mathbf{1}}{N-1}$.

1.4.7 Centralité d'intermédiarité

La centralité d'intermédiarité est une autre mesure de centralité globale proposée par Freeman. L'intuition de cette mesure est que, dans un graphe, un nœud est d'autant plus important qu'il est nécessaire de le traverser pour aller d'un nœud quelconque à un autre. Plus précisément, un sommet ayant une forte centralité d'intermédiarité est un sommet par lequel passe un grand nombre de chemins géodésiques (i.e. chemins les plus courts) dans le graphe. Dans un réseau social, un acteur ayant une forte centralité d'intermédiarité est un sommet tel qu'un grand nombre d'interactions entre des sommets non adjacents dépend de lui . Dans un réseau de communication, la centralité d'intermédiarité d'un nœud peut être considérée comme la probabilité qu'une information transmise entre deux nœuds passe par ce nœud intermédiaire.
Soit $G = (V, E)$ un graphe (orienté ou non) d'ordre N. La centralité d'intermédiarité d'un nœud $v_i \in V$ est définie par :

$$C^{\text{int}}(v_i) = \sum_{j=1}^{N}\sum_{k=1}^{N} \frac{g_{jk}(v_i)}{g_{jk}}$$

où $g_{jk}(v_i)$ est le nombre total de chemins géodésiques entre les nœuds v_j et v_k qui passent par le nœud v_i, et g_{jk} est le nombre total de chemins géodésiques entre les nœuds v_j et v_k.
La centralité d'intermédiarité est basée sur l'hypothèse que les nœuds ne communiquent ou interagissent entre eux qu'à travers les chemins les plus courts. Certains chercheurs ont alors proposé de modifier cette hypothèse afin de prendre en compte le fait que les nœuds peuvent interagir en utilisant des chemins autres que les chemins géodésiques. Par exemple, Freeman a proposé la centralité du flux d'intermédiarité qui n'utilise pas que les chemins géodésiques mais plutôt tous les chemins indépendants entre deux nœuds i.e. les chemins dont les ensembles d'arcs sont disjoints.

Chapitre 2

Ctude de cas

LIEN DATASET ET CODE DE CETTE ÉTUDE DE CAS : "cliquez ici"

Chapitre 3

Conclusion

L'utilisation de Python avec NetworkX facilite grandement l'analyse et la manipulation des réseaux de différentes tailles et structures. NetworkX est une bibliothèque Python open-source qui fournit des outils pour la création, la manipulation et la visualisation de graphes et de réseaux complexes. Cette combinaison offre une grande flexibilité pour répondre aux besoins spécifiques des utilisateurs, que ce soit pour des analyses simples ou des projets de recherche complexes.

L'un des principaux avantages de l'utilisation de Python avec NetworkX est sa facilité d'utilisation. Python est un langage de programmation réputé pour sa lisibilité et sa syntaxe simple, ce qui le rend accessible même aux débutants en programmation. De plus, NetworkX est conçu pour être convivial et offre une interface intuitive pour créer, manipuler et analyser des graphes. Cela permet aux utilisateurs de se concentrer sur leurs objectifs d'analyse plutôt que sur des détails techniques.

En outre, Python offre une vaste gamme de bibliothèques et de modules complémentaires qui peuvent être facilement intégrés avec NetworkX pour étendre ses fonctionnalités. Par exemple, des bibliothèques telles que NumPy, Pandas et Matplotlib peuvent être utilisées en conjonction avec NetworkX pour effectuer des calculs numériques, manipuler des données tabulaires et visualiser des graphes, respectivement. Cette capacité d'intégration facilite l'adaptation de NetworkX aux besoins spécifiques de chaque projet.

Pour les analyses simples, NetworkX offre une multitude de fonctions prédéfinies pour effectuer des opérations de base sur les graphes, telles que le calcul de la centralité, la recherche de chemins les plus courts et la détection de communautés. Ces fonctions peuvent être facilement utilisées grâce à une documentation complète et à des exemples d'utilisation.

Pour les projets de recherche complexes, NetworkX permet une personnalisation avancée grâce à sa flexibilité et à sa modularité. Les utilisateurs peuvent créer leurs propres algorithmes, modifier des fonctionnalités existantes et intégrer des techniques avancées d'analyse de réseaux pour répondre à des questions de recherche spécifiques. De plus, NetworkX est régulièrement mis à jour et bénéficie d'une communauté active, ce qui garantit un support continu et l'ajout de nouvelles fonctionnalités.