RAPPORT DE PROJET DE FIN DE PREMIÈRE ANNÉE

FILIÈRE: E-MANAGEMENT AND BUSINESS INTELLIGENCE

Analyse SNA du réseau collaboratif d'un projet Github

RÉALISÉ PAR :

Kassaoui Wissal

Sahlaoui Botaina

JURY :

Pr. Benhiba Lamia

Pr. Yasser Alami

Pr. Mohammed Abdou

Janati Idrissi *(Encadrant)*

Année académique : 2020/2021

Remerciements

Nous voudrons tout d'abord adresser toute notre gratitude à notre encadrant Mohammed Abdou Janati Idrissi, pour sa confiance, sa disponibilité et surtout ses conseils pertinents. Nous exprimons également notre gratitude aux membres du jury, qui nous ont honorés en acceptant de juger ce travail. Nous tenons à remercier l'ensemble du corps enseignant de l'école Nationale Supérieure d'Informatique et d'Analyse des Systèmes. Enfin, Nous remercions toute personne qui a contribué de près ou de loin à l'élaboration de ce travail.

Résumé

Les interactions dans une communauté open source de développement définissent implicitement un réseau social entre développeurs. Dans ce travail, nous abordons le problème de l'analyse et la caractérisation du réseau inhérent pour un projet GitHub actif: Electron. L'extraction de la structure graphique sous-adjacente peut nous fournir un aperçu intéressant sur la structure de la communauté de développement. Dans ce projet, nous utilisons des métriques de réseaux sociaux SNA et des techniques d'analyse pour caractériser ce réseau de collaboration et identifier les principaux innovateurs dans le projet Electron.

Nous avons développé cette analyse à l'aide des librairies python : *Pandas, networkx, numpy, matplotlib...* pour extraire et analyser les données. Enfin, nous avons utilisé différents plugins et fonctionnalités de *Cytoscape* pour calculer les métriques SNA, visualiser les graphes et identifier les communautés clés avec l'algorithme de clustering K-means.

Mots clés : Analyse des réseaux sociaux, réseau de collaboration, Github, Cytoscape, Python.

Résumé

The interactions in an open source development community implicitly define a social network between developers. In this work we tackle the problem of providing an analysis and characterization of the inherent network for an active GitHub repository: *Electron*. Extracting the underlying graph structure can provide us with interesting insight into the structure of the development community. In this project we use social network metrics and analysis techniques to characterize this collaboration community and identify key innovators in a project.

We developped this analysis using python librairies : *Pandas, networkx, numpy, matplotlib ...* to extract and analyse data. Finally, we used different *Cytoscape* features and plugins to calculate SNA metrics, visualiaze the graph structure and indentify the key communities with the K-means clustering algorithm.

Key words : Social Network Analysis, Collaboration Network, Github, Cytoscape, Python.

Table des matières

Table des figures

1 | Introduction :

L'analyse des réseaux sociaux, *Social Network Analysis (SNA)*, a été développé dès les années 1940 afin de résoudre les problématiques des interactions sociale dans divers réseaux.

Cet analyse a été appliquée dans plusieurs études de cas, notamment pour choisir les employés les plus populaires de l'entreprise, pour cartographier et mesurer les relations et les flux entre les personnes d'une communauté, ou pour choisir le contributeur le plus influent d'un projet.

D'autre part, les outils de codage social, notamment GitHub, ont transformé la manière dont les logiciels sont développés en collaboration et de manière ouverte sur le World Wide Web.

Ainsi, notre projet de fin de première année s'intéresse à l'analyse d'un répositoire github actif en se référant aux méthodologies et métrique SNA.

Notre rapport sera donc structuré comme suit : le premier chapitre c'est une présentation de notre projet en terme de problématique, présentation du projet Github et objectifs. Le deuxsiéme chapitre présente le contexte du projet concernant SNA, Github et les métriques calculés. Le troisième chapitre décrit la démarche suivie lors de l'analyse et la conception commençant par l'extraction des données en arrivant à classifier les graphes sous forme de paquets.

Le dernier chapitre présente les outils utilisés et évalue les résultats obtenue : (dataset, mesures SNA....).Et enfin une conclusion générale qui résume le travail réalisé.

2 | Présentation du projet

2.1 Problématique

Github revendique pas moins de 27 millions d'utilisateurs, ce qui représente une bonne partie des développeurs informatiques dans le monde. La plate-forme est aussi bien utilisé par des développeurs indépendants que par de grandes entreprises comme Google, Apple, Facebook ou encore Amazon, qui s'en servent pour bâtir leurs propres produits[8]. Pour un développeur, exister sur GitHub est devenu incontournable : les recruteurs s'appuient sur les données tirées de l'analyse des réseaux collaboratifs de cette plateforme pour sélectionner les bons candidats.

FIGURE 2.1 – Exemple de réseau sociaux

Notre projet intitulé "Analyse SNA du réseau collaboratif d'un projet Github" repose sur une analyse des interactions au sein du projet framework Electron qui décrit des applications de bureau multiplateformes en utilisant JavaScript, HTML

et CSS. Le répositoire git du projet date depuis 2013 a environ 1035 collaborateurs et utilisé par plus de 100k utilisateurs et 25,296 contributions (commits)

2.2 Présentation du sujet

"Analyse SNA du réseau collaboratif d'un projet Github" est un projet qui s'intèsse principalement à construire une discussion analytique du progrès des contributions dans un projet open source de Github, le projet Electron.

Les relations entre les collaborateurs peuvent être représentées sous la forme d'un graphe, ou d'un réseau de collaboration, où les collaborateurs sont des nœuds.Ce projet aide à déterminer les auteurs (committers) ayant le plus d'influence à l'aide des métriques de SNA globales du réseau ainsi que les mesures de centralité de chaque nœuds, et par la suite **rélever les collaborateurs ayant le plus d'influence et contribution au sein du réseau.**

Identifier les groupes clés et évaluer leur pertinence peut être une tâche difficile qui nécessite un un soutien approprié. Une visualisation adéquate des réseaux de collaboration peut améliorer la compréhension humaine en faisant correspondre les propriétés des données à des dimensions graphiques. Pour cela, nous avons recouru dans ce projet à plusieurs bibliothèques python *(Pandas, Numpy, Networkx...)* et à l'outil de visualisation Cytoscape.

2.3 Objectifs

Pour atteindre notre but à savoir rélever les collaborateurs les plus actifs, nous avons fixer les objectives suivants :

- Extraire les données d'un projet collaboratif Github
- Filter et organiser ses données dans des fichiers
- Visualiser le réseau collaboratif à l'aide de Cytoscape
- Analyser les métriques du réseaux collaboratifs
- Interprétation des résultats obtenus
- Faire une classification du réseau obtenu à l'aide de Cytoscape.

2.4 Phases du projet

Afin d'atteindre ces objectifs, nous avons commencé par une évaluation de nos connaissances dans le domaine de la théorie des graphes et une augmentation des compétences dans le domaine de l'analyse des réseaux sociaux SNA, ainsi que l'outil de programmation collaborative, Github. En conséquence, nous avons établi la conception de l'approche analytique de notre étude de cas : le projet Electron. Enfin, nous avons présenté sous forme de graphiques, de diagrammes et de diverses métriques les résultats obtenus.

Nous allons maintenant passer au chapitre suivant qui va décrire le Background du projet, c'est à dire, la définition de la SNA et ses métrique.Nous allons présenté également la plateforme Github et le logiciel de contrôle de version Git.

3 | Backgroud du projet

3.1 Analyse des réseaux sociaux (SNA)

3.1.1 Définitions des réseaux sociaux

Réseau social désigne un arrangement des liens entre des individus ou des organisations, formant un groupement qui a un sens : famille, collègue, amis, communauté. D'autre part, l'expression « réseaux sociaux » renvoie aux entreprises de réseautage social sur Internet et à leurs utilisateurs à travers le monde. Les applications désignées comme « service de réseautage social en ligne » servent à constituer un réseau social virtuel en reliant, non pas des personnes, mais des identités virtuelles. [4]

3.1.2 Analyse de réseaux sociaux

L'analyse de réseaux sociaux (social network analysis – SNA) caractérise les structures en réseau en termes de nœuds (acteurs individuels, personnes ou choses au sein du réseau) et les arêtes ou liens (relations ou interactions) qui les relient. Cette vision du monde social comme ensemble de liens a été développée par l'anthropologue John Barnes dès 1954 : « Il me semble approprié de parler de réseau pour désigner cette sphère sociale. L'image que j'ai en tête est celle d'un ensemble de points qui sont reliés par des lignes. Les points de cette image sont des individus, ou parfois des groupes, et les lignes indiquent quelles sont les personnes qui interagissent les unes avec les autres. » [7]

Linton Freeman propose quatre critères permettant de définir plus précisément la SNA, en expliquant que l'analyse des réseaux sociaux est basé sur une l'approche structurel qui étudie des interactions entre les acteurs sociaux. Il repose sur des données empiriques systématiques, il s'appuie fortement sur les visualisations graphiques et il repose sur des modèles mathématiques et informatiques.

Cette analyse structurale porte spécifiquement sur la description et l'analyse des différents modes de relation possibles : interdépendance des membres, réciprocité ou non des relations, place centrale de certains, absence de relations créant des « trous » relationnels au sein du réseau, fréquence des relations (liens forts versus liens faibles).

La force de l'analyse structurale réside dans sa capacité à représenter de façon simplifiée la complexité et la diversité des relations entre acteurs. Le système d'interdépendance est modélisé en prenant en compte l'imbrication progressive des acteurs au sein d'une « forme » structurale qui évolue, se contracte ou se dilate en fonction de l'activité de ses membres. La plasticité des réseaux est accentuée par leur absence définie de frontières.[3]

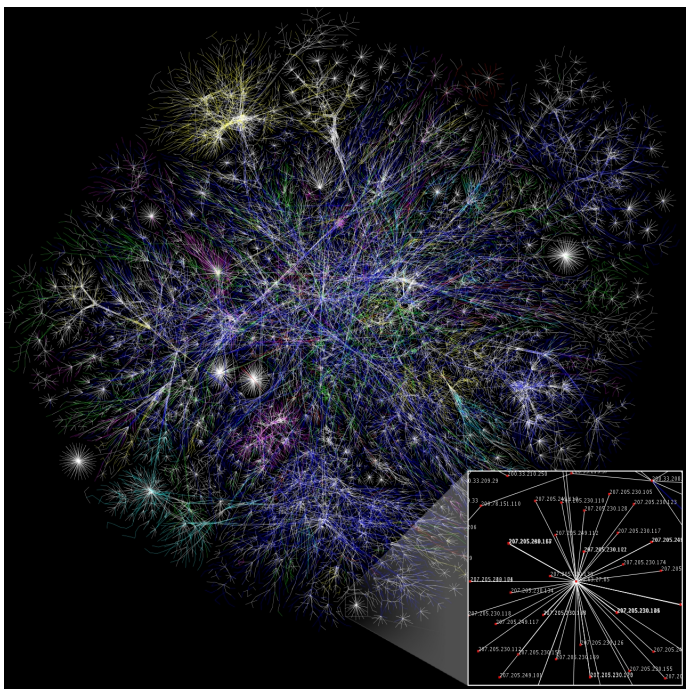

FIGURE 3.1 – Visualisation des données pour le réseau Internet. Chaque ligne relie deux adresses IP, la longueur représentant le délai entre elles.

3.1.3 Représentation des réseaux sociaux

Un réseau social est constitué d'un ensemble fini de sommets et des relations, ou liens, définies sur ceux-ci. La structure de ces réseaux est généralement représentée par des graphes. Par conséquent, *les réseaux sont souvent considérés comme équivalents aux graphes.*

Un graphe est composé de noeuds et d'arêtes. Ainsi, **chaque noeud** peut représenter une grande variété d'entités individuelles (par exemple, des personnes, des organisations, pays, papiers, produits, plantes et animaux) en fonction du domaine d'application. De même, **une arête** est une ligne qui relie deux noeuds et, par analogie, elle peut représenter de nombreux types de relations entre des entités individuelles (par exemple, la communication, la coopération, l'amitié, la parenté, les connaissances et le commerce).

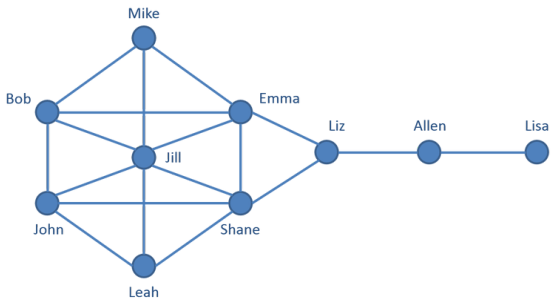

FIGURE 3.2 – Exemple de graphe de réseau social simple (acteurs=nœuds ; relations=lignes)

3.2 Métriques d'analyse des réseaux sociaux

Dans cette section, nous présentons quelques mesures de graphes et métriques populaires utilisées dans l'analyse des réseaux sociaux [6]. Ces mesures sont utiles dans le sens où elles nous donnent un aperçu sur le rôle des nœuds dans notre réseau. L'étude du rôle des individus et la façon dont ils interagissent dans le contexte du réseau vise à comprendre le comportement des systèmes sociaux qui ont généré ces réseaux, ce qui est normalement l'objectif final de la SNA.

Les mesures que nous allons introduire dans les sous-sections suivantes peuvent être divisées en fonction du niveau d'analyse que l'on souhaite effectuer : au niveau de petites unités, telles que les nœuds, ou au niveau du réseau entier.[2]

3.2.1 Notations de base des graphes

Notre graphe G, dans cette représentation, sera constitué d'un ensemble non vide et fini V des sommets d'un ensemble E des arcs correspondent à des couples ordonnés de sommets. On note $G = (V; E)$.

— L'ordre d'un graphe est la somme totale des sommets n. On note $|V| = n$

— La taille d'un graphe est la somme total de ses arêtes m. On note $|E| = m$

— Le nombre maximum d'arêtes dans un graphe est $m_{max} = \frac{n(n-1}{2}$ pour les graphes non orientés et $m_{max} = n(n-1)$, pour les graphes orientés.

3.2.2 Mesures Locales :

La centralité ou le prestige est une mesure générale de la position d'un acteur dans la structure globale du réseau social et peut être calculée à l'aide de plusieurs métriques. Les plus utilisées sont le degré, l'intermédiarité, la proximité et la centralité du vecteur propre. Ces mesures ont pout but de capturer la notion d'importance dans un graphe, en identifiant les sommets les plus significatifs [1]

Degré

Le degré, d'un sommetv_i, généralement désigné par $deg(vi)$, est une mesure de l'adjacence immédiate et de l'implication du sommet dans le réseau et est calculé comme le nombre d'arêtes incidentes d'un sommet donné.

$$deg(vi) = |\{e \in E; e = (vi, vj) \in E\}| \tag{3.1}$$

L'intermédiarité ou Centralité d'intermédiarité

L'intermédiarité $I(v)$ est une mesure de centralité d'un sommet v d'un graphe. Elle désigne le nombre de plus courts chemins passant par le sommet v.

$$I(v) = \frac{1}{\binom{n-1}{2}} \sum_{\{v1,v2\} \subseteq V \setminus \{v\}} \frac{|\ plus\ courts\ chemins\ entre\ v1\ et\ v2\ passant\ par\ v\ |}{|\ plus\ courts\ chemins\ entre\ v1\ et\ v2\ |} \tag{3.2}$$

La proximité ou Centralité de proximité

La proximité $C(v)$ d'un sommet v est la distance moyenne de v aux autres sommets.

$$C(v) = \sum_{u} \frac{1}{d(u,v)} \tag{3.3}$$

La centralité du vecteur propre.

La centralité de vecteur propre est une mesure de l'influence d'un sommet dans un réseau. Elle est basée sur l'attribution d'un score relatif à chaque sommet et la mesure du degré de connexion de ce dernier avec d'autres sommets bien connectés.

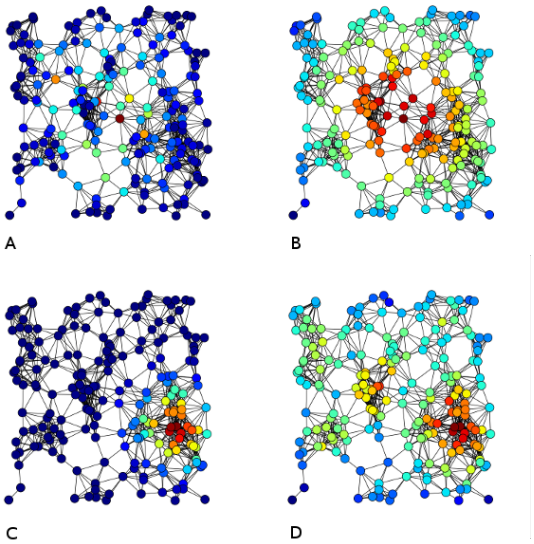

FIGURE 3.3 – Exemples de A) Centralité d'intermédiarité, B) Centralité de proximité, C) Centralité des vecteurs propres, D) Centralité des degrés.

Coefficient de clustering locale

Les réseaux sociaux sont naturellement transitifs, ce qui signifie que "les amis de mes amis sont susceptibles d'être mes amis". Cette propriété de transitivité est quantifiée par un coefficient de clustering qui peut être global, c'est-à-dire calculé pour l'ensemble du réseau, ou local, c'est-à-dire calculé pour chaque nœud. Le coefficient de clustering local d'un sommet v d'un graphe est le ratio de paires de sommets, qui sont des voisins de v donné et qui sont connectés à v par des arêtes. Il indique le niveau de cohésion entre les voisins d'un nœud donné.

$$C_v = \frac{|\text{ paires de voisins de } v \text{ connectés }|}{\binom{deg(v)}{2}} \tag{3.4}$$

3.2.3 Mesures globales :

Rayon et Diamètre

Le diamètre D est plus grande des distances (longueur d'un plus court chemin) entre deux sommets. Tandis que, le rayon R est plus grande distance du centre (le centre est lesommet le moins éloigné de tout autre sommet) à un sommet donné

$$D = max_{v1 \in V, v2 \in V}\, d(v1, v2) \tag{3.5}$$

$$R = min_{u \in V}\, (max_{v \in V}\, d(v, u)) \tag{3.6}$$

Longueur moyenne des plus courts chemins

La longueur moyenne des plus courts chemins L est une mesure d'efficacité du réseau : plus cette longueur est courte, plus l'efficacité à faire transiter le flux au sein du réseau est grande. Il s'agit de la moyenne de la distance entre chaque couple de sommets.

$$L = \sum_{v1 \in V, v2 \in V} \frac{d(v1, v2)}{n(n-1)} \tag{3.7}$$

Densité

La densité d est une mesure importante au niveau du réseau, qui permet d'expliquer le niveau général de connectivité dans un réseau. C'est la proportion du nombres des arêtes du graphe par rapport au nombre totales des liens possible.

$$d = \frac{2m}{n(n-1)} \tag{3.8}$$

Degré moyen

Le degré moyen K est la moyenne des degrés des sommets. Il peut être utilisé pour mesurer la connectivité globale d'un réseau.

$$K = \frac{2m}{n} \tag{3.9}$$

Coefficient de clustering global

Le coefficient de clustering global C est la moyenne des coefficients de clustering locals des sommets. Plus précisément, ce coefficient est la probabilité que deux nœuds soient connectés sachant qu'ils ont un voisin en commun.

$$C = \frac{3 \times \text{nbr. de sous graphes triangulaire}}{\text{nbr. de sous graphe connexes à 3 sommets}} \tag{3.10}$$

3.3 Rappel Github

3.3.1 Git

Git est système de gestion de version, ou VCS, qui permet de suivre l'historique des modifications lorsque des personnes ou des équipes collaborent sur des projets. Au fur et à mesure que le projet évolue, les équipes peuvent apporter des changements faire des tests avec la posibilité de récupérer les versions précédentes à tout moment.

Git est couramment utilisé pour le développement de logiciels open source et commercial, avec des avantages significatifs pour les particuliers, les équipes et les entreprises.

FIGURE 3.4 – Git

3.3.2 L'open source Github

Les projets open source (Open Source Software (OSS)), ou les projets avec un code source accessible au public, ont changé radicalement notre vision sur le processus du développement logiciel. Les projets OSS ne sont pas seulement viable mais aussi dynamique et contribue à l'épanouissement des technologies. GitHub représente donc l'un des moyens les plus populaires pour héberger des projets open source et partager du contenu. C'est une plateforme collaboratif construit au-dessus du système de contrôle de version git.

En janvier 2014, il hébergeait plus de 10,6 millions de dépôts (repositories). Il comprend une variété de fonctionnalités qui encouragent le travail d'équipe et la discussion continue tout au long de la durée du projet. GitHub utilise un modèle "fork and pull" dans lequel les développeurs créent leurs propres copies d'un dépôt et soumettent des requêtes lorsqu'ils veulent que le responsable du projet transfère leurs modifications dans la branche principale, fournissant ainsi un environnement dans lequel les gens peuvent facilement réviser leurs codes.[5]

GitHub contient également des fonctions sociales intégrées : les utilisateurs peuvent"s'abonner" à des projets et "suivre" d'autres utilisateurs, ce qui donne lieu à un flux constant de mises à jour sur les personnes et les projets qui les intéressent. Le système prend en charge les profils d'utilisateurs qui fournissent un résumé de l'activité récente d'une personne sur le site, comme ses "commits", ses "forks", ou les problèmes qu'elle a signalés.

Flux Github

Le flux GitHub est un workflow léger basé sur une branche qui prend en charge les équipes et les projets où des déploiements sont effectués régulièrement.

Le flux GitHub comporte six étapes, chacune avec des avantages distincts lors de sa mise en œuvre :

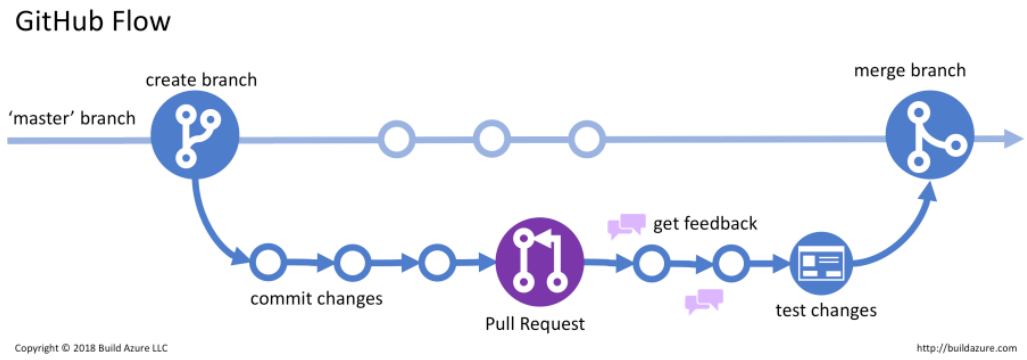

FIGURE 3.5 – GitHub-Flow

Créer une branche

Le branchement existe pour vous aider à gérer ce flux de travail.Les branches de sujet créées à partir de la branche de déploiement canonique "master" permettent aux équipes de contribuer à de nombreux efforts parallèles.

Lorsque on crée une branche dans le projet,on crée un environnement dans lequel on peut essayer de nouvelles idées. Les modifications apportées sur une branche n'affectent pas la "main" branche.

Ajouter une commit

Une fois la branche créée, on peut commencer à apporter des modifications. Chaque fois que'on ajoute, modifie ou supprime un fichier, on effectue un commit et l'ajoute à la branche. Ce processus d'ajout de commits permet de suivre la progression lorsque on travaille sur une branche de fonctionnalité.

Les commits créent également un historique transparent de votre travail que les autres peuvent suivre pour comprendre ce que vous avez fait et pourquoi. De plus, chaque commit est considéré comme une unité de changement distincte. Cela vous permet d'annuler les modifications si un bogue est détecté ou si vous décidez de vous diriger dans une direction différente.

Ouvrir une pull request

Les Pull Requests initient une discussion sur les commits.Tout le monde peut voir exactement quelles modifications seraient fusionnées.On peut ouvrir une Pull Request à tout moment du processus de développement.

Discuter et réviser le code

Une fois qu'une Pull Request a été ouverte, la personne ou l'équipe qui examine vos modifications peut avoir des questions ou des commentaires. Les Pull Requests sont conçues pour encourager et capturer ces participations.

Déployer

Avec GitHub, on peut déployer à partir d'une branche pour les tests finaux en production avant de fusionner avec main.

Une fois que la pull request a été examiné et que la branche a réussi les tests, on peut déployer les modifications pour les vérifier en production. Si la branche pose des problèmes, on peut la restaurer en déployant la branche principale existante en production.

Merge

Maintenant que les modifications ont été vérifiées en production, il est temps de fusionner le code dans la branche principale.

Une fois fusionnées, les Pull Requests conservent un enregistrement des modifications historiques apportées au code.

4 | Analyse et conception

Dans ce chapitre, on va détailler la démarche suivie dans notre projet PFA. On va expliquer l'étude du cas du projet "Electron" sur GitHub, en commençant par l'extraction des données, leur visualisation et enfin l'analyse de ces données extraites.

4.1 Etude de cas : Le projet github Electron

Il s'agit d'examiner le réseau de collaboration de l'un des plus grand projet sur Github : le framework Electron.

Le Framework Electron permet d'écrire des applications de bureau multiplateformes en utilisant JavaScript, HTML et CSS. Il est basé sur Node.js et Chromium et est utilisé par l'éditeur Atom et de nombreuses autres applications.

FIGURE 4.1 – Logo Electron

Le répositoire git du projet date depuis 2013 a environ 1035 collaborateurs et utilisé par plus de 100k utilisateurs Nous remarquons que dans ce projet, il y

a des collaborateurs plus actifs et indisponsables pour l'avancement du projet. Ci-dessous est le top 30 des contributeurs avec le plus grand nombre de commits.

4.2 Méthodologie suivie

L'analyse SNA de ce projet sera selon la démarche suivante :

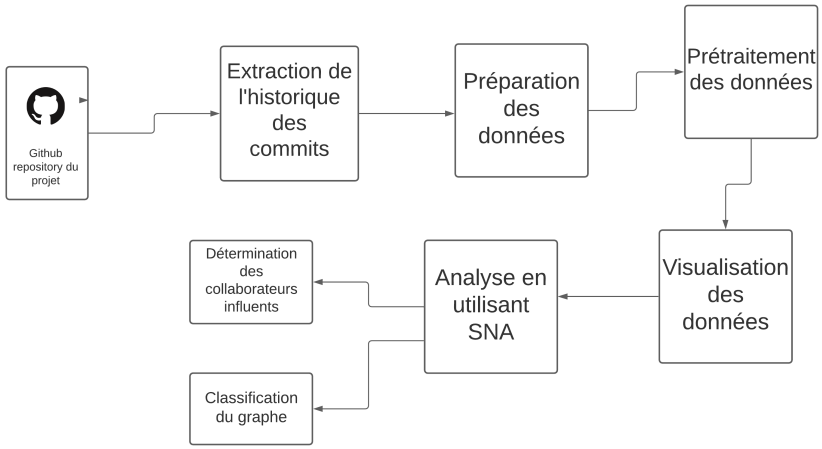

FIGURE 4.2 – Organigramme de la méthodologie .

4.2.1 Extraction des données

Les systèmes de gestion de version, notamment Git, ont un objectif principal : enregistrer l'évolution de votre code-base au fil du temps.

FIGURE 4.3 – Logo Git .

Cela vous permet de voir comment vos projets ont progressé. Vous pouvez savoir qui a contribué à votre projet et identifier les modifications qui ont été apportées

à votre code et à quel moment. Pour ce faire, la commande "git log" est l'outil le plus basique et le plus puissant.

4.2.2 Préparation des données :

Le processus de préparation des données extraites du projet electron comprend :

- La gestion des valeurs manquantes
- La suppression des colonnes non nécessaire
- La suppression des données redondants
- Le regroupement des données selon l'année
- L'affectation de l'ensemble des fichiers modifiés à leurs éditeurs
- L'enregistrement des données sous format .csv

4.2.3 Pré-traitement des données :

L'ensemble de données "electron" permet d'envisager des applications intéressantes de l'analyse des réseaux sociaux. Afin d'extraire proprement cette structure graphique et d'analyser diverses propriétés du réseau social résultant, nous avons construit une **matrice adjacence** telle que :

- **Les noeuds** sont l'ensemble des collaborateurs durant la durée choisie
- **Les arêtes** représentent la collaboration sur le même projet GitHub
- On définit la relation de collaboration comme suit : "deux personnes collaborent si et seulement si il ont fait un commit au même fichier"
- **Le poids** des arêtes correspond au nombre de collaborations entre les deux noeuds

La matrice obtenue est par la suite stockée dans un fichier '.mat'.

4.2.4 Visualisation du graphe de collaboration

La visualisation du graphe nous aidera à comprendre le réseau de collaborations, à identifier les acteurs clés, les futurs collaborateurs potentiels,et enfin contex-

tualiser les relations entre les noeuds. C'est donc la représentation visuelle des nœuds et des arêtes d'un graphe.

Les outils de visualisation de graphiques fournissent des interfaces conviviales pour interagir et explorer les données des graphiques.

Dans notre projet, nous avons utilisé l'outil de visualisation Cytoscape

4.2.5 Analyse des données

Afin de caractériser ce réseau de collaboration et identifier les principaux innovateurs dans le projet Electron, nous avons analyser le graphe du réseau de collaboration selon les métrique globales du réseau ainsi que les mesures de centralité des noeuds.

Métrique globales du réseau

Pour les attributs du réseau, nous avons choisi ce qui suit :

— Nombre de noeuds

— Nombre d'arêtes

— Degré moyen

— Diamètre

— le plus court chemin moyen

— Nombre des composantes connexes

Mesures de centralité des noeuds

Pour les mesures locales des noeuds, nous avons exploité les 4 attributs ci-dessous :

— La centralité de degré

— Centralité d'intermédiarité (Betweenness)

— Centralité de proximité (Closness)

— La centralité du vecteur propre.

4.2.6 Classification des graphes : Clustering

Le partitionnement de données (ou data clustering en anglais) est une méthode en analyse des données. Elle vise à diviser un ensemble de données en différents « paquets » homogènes, en ce sens que les données de chaque sous-ensemble partagent des caractéristiques communes.

5 | Réalisation

5.1 Vue de l'ensemble des outils :

5.1.1 Cytoscape

Présentation de Cytoscape :

Cytoscape est une plateforme logicielle open source qui permet l'intégration, la visualisation et l'analyse des données dans le contexte des réseaux. Cytoscape nous permet de visualiser un grand nombre interactions de manière explicite afin de pouvoir explorer la structure de ses interactions et analyser le rôle des noeuds individuelles ainsi que la structure générale du réseau.

FIGURE 5.1 – Logo de Cytoscape

Exploitationn de l'outil cytoscape

aMatReader est une application Cytoscape 3 qui offre la possibilité de lire une matrice d'adjacence et généré le graphe correspondant . Les noms des noeuds sont dans la première ligne et comme premier champ dans chaque ligne suivante. Chacune des valeurs des champs est interprétée comme un poids d'arête.

NetworkAnalyzer effectue une analyse des réseaux et calcule les paramètres de topologie du réseau, notamment le diamètre d'un réseau, le nombre moyen de voisins et le nombre de paires de nœuds connectés. Il calcule également les distributions de paramètres de réseau plus complexes tels que les degrés des nœuds, les coefficients de clustering moyens et les longueurs des plus courts chemins. Il affiche les résultats sous forme de diagrammes, qui peuvent être enregistrés sous forme d'images ou de fichiers texte.

MClique est une application qui trouve toutes les cliques maximales de taille >= 3 et les affiche dans un panneau de résultats triés par taille L'utilisateur a la possibilité d'exporter toutes les cliques (ou) les cliques sélectionnées en tant que réseaux Cytoscape séparés.

5.1.2 Python

Python est un langage de programmation le plus utilisé dans le domaine du Machine Learning, du Big Data et de la Data Science. Il s'agit d'un élément moteur de l'explosion du Big Data.

FIGURE 5.2 – Logo de Python

5.2 Extraction des données "Electron" avec git log

La commande git log affiche un enregistrement des commits dans un dépôt Git. Par défaut, la commande git log affiche le hash du commit, le message de commit, et d'autres métadonnées de commit. On peut filtrer la sortie de git log en utilisant diverses options.

La commande suivante permet d'extraire le hash du commit, le nom de l'auteur, la date du commit et le nom des fichiers modifiés puis stockés ces résultats dans un fichier git.log

```
git log  --since='last year' --date=short --all --pretty="%x40%h%
    x2C%an%x2C%ad%x2C" --name-only | tr "\n" " " | tr "@" "\n" >git
    .log
```

Listing 5.1 – Commande git

La commande ci-dessous permet d'avoir le nombre de fichier changés, ainsi que le nombre des lignes ajoutées et supprimées dans chaque commit. Nous avons stockés l'output dans le fichier stat.csv

```
git log  --date=local --all --pretty="%x40%h%x2C%an%x2C%ad%x2C%x22%
    s%x22%x2C" --shortstat | tr "\n" " " | tr "@" "\n" >> stat.csv
   sed -i 's/ files changed//g' stat.csv
   sed -i 's/ file changed//g' stat.csv
   sed -i 's/ insertions(+)//g' stat.csv
   sed -i 's/ insertion(+)//g' stat.csv
   sed -i 's/ deletions(-)//g' stat.csv
   sed -i 's/ deletion(-)//g' stat.csv
```

Listing 5.2 – Commande git

5.3 Préparation et prétraitement des données Electron :

Afin d'exploiter et d'analyser efficacement les données collectées, le prétraitement des données doit être mis en œuvre comme une série d'étapes à l'aide des bibibliotèques *Pandas, Numpy, et Collections* de Python.Ce traitement vise à convertir les données extraites dans deux formats facile à analyser :

— Une matrice DataFrame de Pandas où les lignes correspondent à l'ensemble des commits, les colonnes à des attributs décrivant chaque commit (auteur, date, fichier changé...) qui sera stockée dans le fichier historique.csv

— La matrice adjacente de taille $n \times n$
avec n = Nombres d'auteurs = Nombres de noeuds.(Voir la section 3.2.3)
qui sera stockée dans le fichier electron.mat

5.3.1 Création du fichier historique.csv

Premièrement, on a fait une joiture des données des fichier git.log et stat.csv sur le hash du commit. On obtient le dataframe de structure ci-dessous content 142551 lignes et 8 colonnes représentant le hash ou code du commit, le nom de l'auteur, le commentaire qu'il a fait :

```
[31]: data.head()
```

[31]:

	sha	author	comment	changed file	lines added	lines deleted	timestamp	filename
0	fa2db00e5	electron-roller[bot]	chore: bump chromium to 93.0.4532.2 (main) (#2...	37	169	212	2021-06-04	DEPS patches/chromium/.patches patches/chrom...
1	063980e5f	Jeremy Rose	Merge branch 'main' into refactor-logging		NaN	NaN	2021-06-04	
2	0736367fa	Jeremy Rose	lint	1	5	NaN	2021-06-04	shell/common/logging.h
3	7184b05db	Jeremy Rose	default to stderr	1	11	5	2021-06-04	shell/common/logging.cc
4	8fc2a61fa	Jeremy Rose	fix sandboxed children not being able to write...	16	216	107	2021-06-04	filenames.gni shell/app/electron_crash_repor...

FIGURE 5.3 – Extrait de la Dataframe initiale de l'historique des commits.

Puisqu'on s'intéresse au modification sur les fichiers, on a supprimé les lignes dont la valeur de 'filename' est vide, ensuite on a remplacé les valeurs numériques manquantes par 0. La dataframe obtenu est stocké dans un fichier historique.csv et sera exploité dans la partie d'analyse préliminaire des données Elle est comporte 136801 lignes et 8 colonnes représentant le hash ou code du commit, le nom de l'auteur, le commentaire qu'il a fait.

	sha	author	comment	changed file	lines added	lines deleted	timestamp	filename
3	7184b05db	Jeremy Rose	default to stderr	1.0	11	5	2021-06-04	shell/common/logging.cc
4	8fc2a61fa	Jeremy Rose	fix sandboxed children not being able to write to log dir	16.0	216	107	2021-06-04	filenames.gni shell/app/electron_crash_reporter_client.cc shell/app/electron_main_delegate.cc shell/app/electron_main_delegate.h shell/browser/api/electron_api_app.cc shell/browser/api/electron_api_crash_reporter.cc shell/browser/browser.cc shell/browser/browser_process_impl.cc shell/browser/electron_browser_client.cc shell/browser/electron_browser_client.h shell/browser/electron_browser_context.cc shell/browser/net/system_network_context_manager.cc shell/browser/ui/devtools_manager_delegate.cc shell/common/electron_paths.h shell/common/logging.cc shell/common/logging.h
5	8047798b3	Shelley Vohr	chore: combine BoringSSL patches	3.0	118	130	2021-06-04	patches/node/.patches patches/node/fix_handle_boringssl_and_openssl_incompatibilities.patch patches/node/fix_key_gen_apis_are_not_available_in_boringssl.patch
6	51faa70a5	Shelley Vohr	fix: accidental silent crypto failures	1.0	17	12	2021-06-04	patches/node/fix_handle_boringssl_and_openssl_incompatibilities.patch
7	314e627df	Samuel Attard	Revert Revert "fix: change ASAR archive cache to per-process to fix leak (#29535)""	7.0	58	36	2021-06-04	shell/browser/electron_browser_main_parts.cc shell/common/asar/archive.cc shell/common/asar/archive.h shell/common/asar/asar_util.cc shell/common/asar/asar_util.h shell/renderer/electron_renderer_client.cc shell/renderer/web_worker_observer.cc
8	b5c79632e	Electron Bot	Bump v13.1.1	3.0	6	6	2021-06-04	ELECTRON_VERSION package.json shell/browser/resources/win/electron.rc
9	0dcc66079	Samuel Attard	Revert fix: change ASAR archive cache to per-process to fix leak (#29535)""	7.0	36	58	2021-06-04	shell/browser/electron_browser_main_parts.cc shell/common/asar/archive.cc shell/common/asar/archive.h shell/common/asar/asar_util.cc shell/common/asar/asar_util.h shell/renderer/electron_renderer_client.cc shell/renderer/web_worker_observer.cc
10	0753e4911	Samuel Attard	fix: disable CET as v8 deoptimization is incompatible with it (#29546)	1.0	2	0	2021-06-04	build/args/all.gn

FIGURE 5.4 – Extrait Dataframe après nettoyage de l'historique des commits.

5.3.2 Création du fichier electron.mat

Premièrement, à partir de la dataframe précédente, on regroupe l'historique des commits par année, ensuite, pour chaque année on crée une liste de fichiers modifiés par chaque auteur, les noms des auteurs sonts uniques.

```
regrouped_data_filtered=[[auteur1,fichier_modifie1,fichier_modifie2
    ,fichier_modifi 3...],[auteur2,fichier_modifie1,
    fichier_modifie2,fichier_modifi 3...],....]
```

Listing 5.3 – Liste de fichiers modifiés par chaque auteur

On exploite cette liste pour créer notre matrice d'adjacence à l'aide de la bibliothèque numpy et la fonction *interaction* qui prend en argument deux listes, et retourne leurs valeur d'intersection .

```
def intersection(lst1,lst2):  #cette fonction
    R = list(set(lst1) & set(lst2))
    I = len(R)
    return I
```

Listing 5.4 – La fonction interaction

La matrice electron.mat est sous la forme suivante

	林伟强	Haojian Wu	Andy	Kevin Sawicki	Samuel Attard	x-yao	Zeke Sikelianos	abouthiroppy	Drew Chandler	Thorben Egberts	...	Ming Luo	大和屋貴仁	Joe Naha	Wei Wang	Krzysztof	Christopher Dieringer	ZhangYueqiu	Omri Litov	foxy	Levin Rickert
林伟强	0.0	0.0	0.0	0.0	0.0	1.0	2.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0
Haojian Wu	0.0	0.0	0.0	18.0	2.0	0.0	4.0	0.0	1.0	0.0	...	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0
Andy	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Kevin Sawicki	0.0	18.0	1.0	0.0	84.0	0.0	182.0	1.0	1.0	0.0	...	0.0	7.0	3.0	1.0	1.0	1.0	0.0	7.0	1.0	1.0
Samuel Attard	0.0	2.0	0.0	84.0	0.0	0.0	53.0	0.0	1.0	0.0	...	2.0	0.0	0.0	1.0	0.0	1.0	0.0	5.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
Christopher Dieringer	0.0	0.0	0.0	1.0	1.0	0.0	1.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
ZhangYueqiu	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Omri Litov	0.0	0.0	0.0	7.0	5.0	0.0	1.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
foxy	0.0	1.0	0.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Levin Rickert	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

341 rows × 341 columns

FIGURE 5.5 – La matrice electron.mat.

5.4 Analyse préliminaire des données du projet Electron

En utilisant la dataframe de la section 5.3.1 et la bibliothèque plotly de python, nous avons effectué une étude préliminaire des données du projet electron afin d'avoir un aperçu sur les collaborateurs avec le plus grand nombre de commits ainsi que l'évolution du projet au fil du temps. Pour enfin, détérminer l'année où le projet Github a connu le plus d'activité

Contributeurs avec le plus grand nombre de commits dans le projet electron.

Nous remarquons que dans ce projet, il y a des collaborateurs plus actifs et indisponsables pour l'avancement du projet. Ci-dessous est le top 30 des contributeurs avec le plus grand nombre de commits.

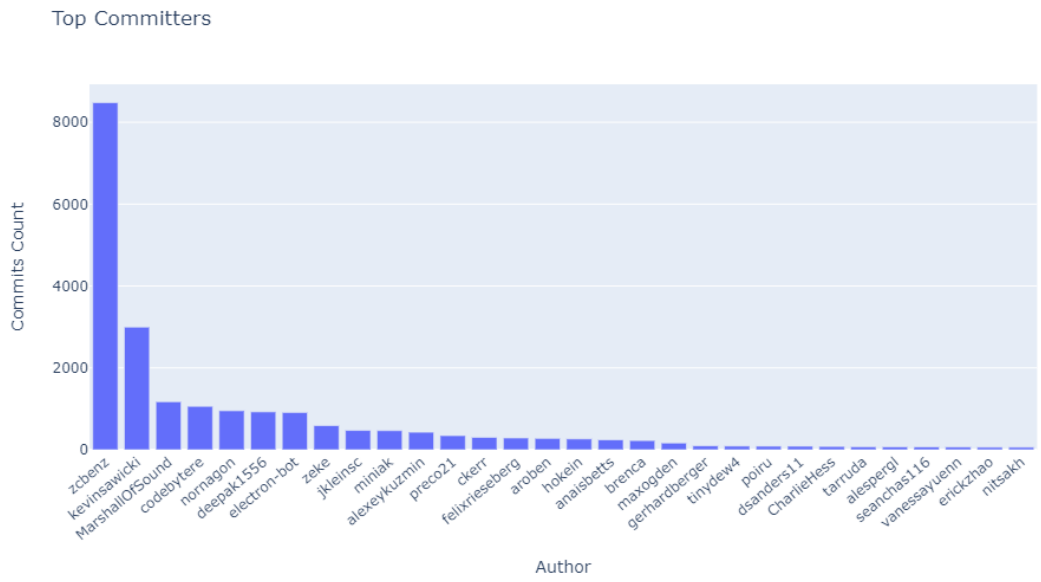

FIGURE 5.6 – Les contributeurs avec le plus grand nombre de commits.

Evolution du projet electron au fil de temps

Il s'avère que le répositoire Electron a connu un pic d'activité au cours de l'année 2016-2017.

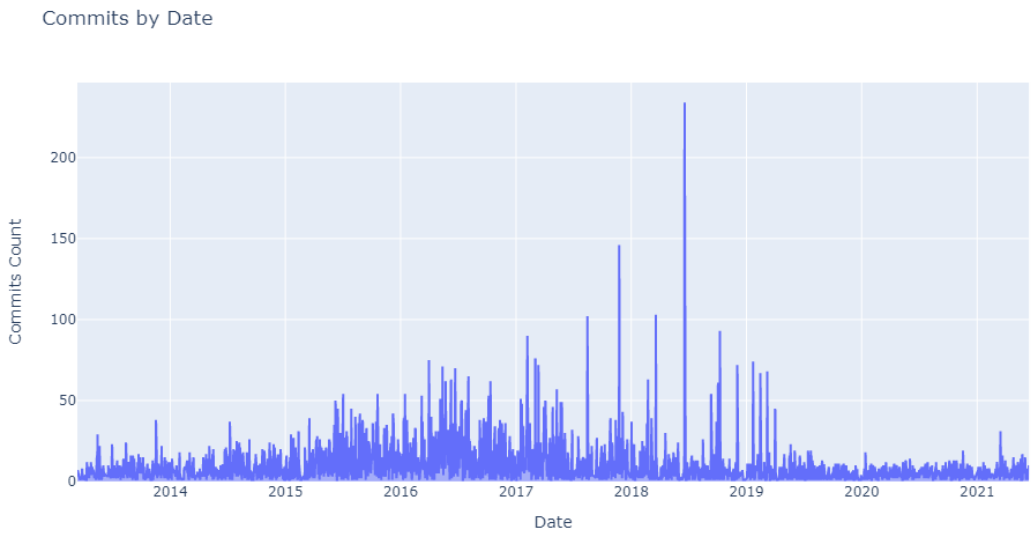

FIGURE 5.7 – Evolution de l'activité du projet electron.

Le graphique ci-dessous montre clairement que le répositoire a atteint son apogée en 2016 et que l'activité a ensuite diminué au fil de temps jusqu'à atteindre 4 commits par jour en 2020.

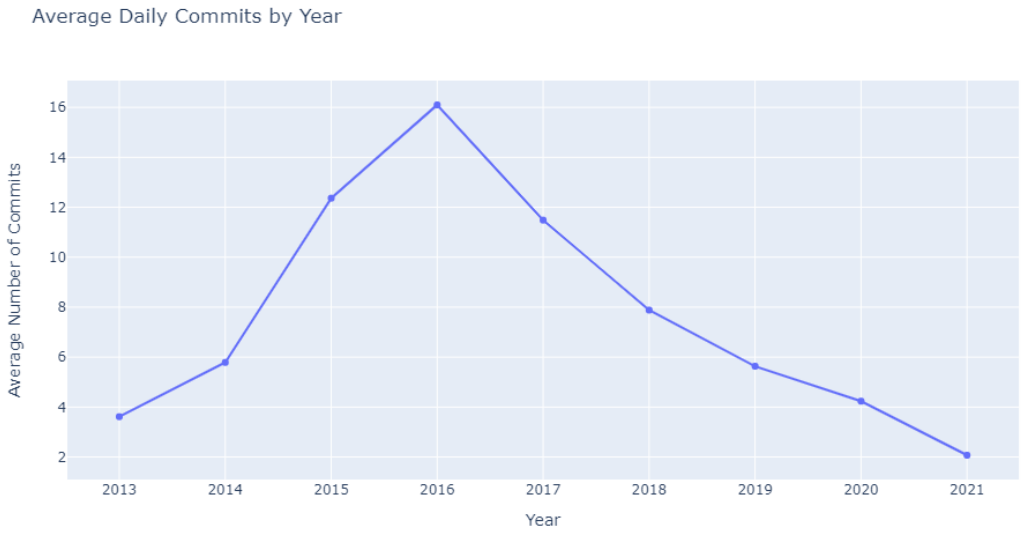

FIGURE 5.8 – Moyenne quotidienne de commits par année.

Nous allons donc nous concentré sur l'étude de l'année 2016, qui a vu le plus d'activité sur le projet Electron.

5.5 Métriques SNA du projet Electron

Le plugin AmatReader de l'outil cytoscape permet de générer les graphes de collaboration du projet electron à partir des fichiers .mat créer précédement (section 4.3.2). Par suite, on calcule les attributs globales du réseau electron ainsi que les mesures de centralités des noeuds par NetworkAnalyser et networkx.

5.5.1 Métriques SNA golables du projet Electron en 2016

Le tableau suivant énumère les attributs d'un graphe non orienté non pondéré créé à partir des collaborations des participants dans le projet electron au cours l'année qui a vu le plus d'activité.

Attribut	Valeur
Nombre de noeuds	341
Nombre d'arêtes	3723
Degré moyen	22.7012
Diamètre	4
Rayon	1
Coefficient de clustering	0.821
le plus court chemin moyen	2.041
Composantes connexes	15
Plus grande composantes connexe	326
Noeuds isolés	13

Le réseau "electron" comporte une composante géante contenant 95% de tous les nœuds. La composante suivante est nettement plus petite, environ 0.5%. La longueur moyenne des chemins, 2.041, qui est le double du diamètre, 4. En effet, le diamètre reflète le chemin le plus long à travers tous les composantes et ignore les chemins manquants entre les nœuds non connectés. Le coefficient de clustering est élevé, 0,821. L'ensemble de ces attributs indique que le réseau de collaboration est un réseau sans échelle, à petit monde, qui peut être analysé de la même manière que les autres réseaux de collaboration.

Le tableau ci-dessous fournit les scores de centralité globale du graphe de collaboration .

Attribut	Valeur
La centralité de degré	0.138
Centralité d'intermédiarité	0.009
Centralité de proximité	0.495
La centralité du vecteur propre.	0.036

En génrale , une valeur proche de 1 indique que le réseau est étroitement organisé autour d'un nombre relativement faible de nœuds. Bien qu'aucune des valeurs ne soit très proche de 1, le score de proximité suggère la possibilité d'existence de nœuds structurellement importants.

5.5.2 Etude de la centralité des noeuds du réseau électron en 2016

Ici, les mesures de centralité des participants au projet peuvent être considérées comme de bons indicateurs de leur contribution au projet ainsi que leur collaboration avec les autres développeurs .

Visualisaion selon la centralité de degré

Le degré de centralité - la mesure du nombre total d'arêtes connectées à un nœud particulier - est un bon indicateur de la capacité d'un développeur à collaborer directement avec les autres membres du réseau.

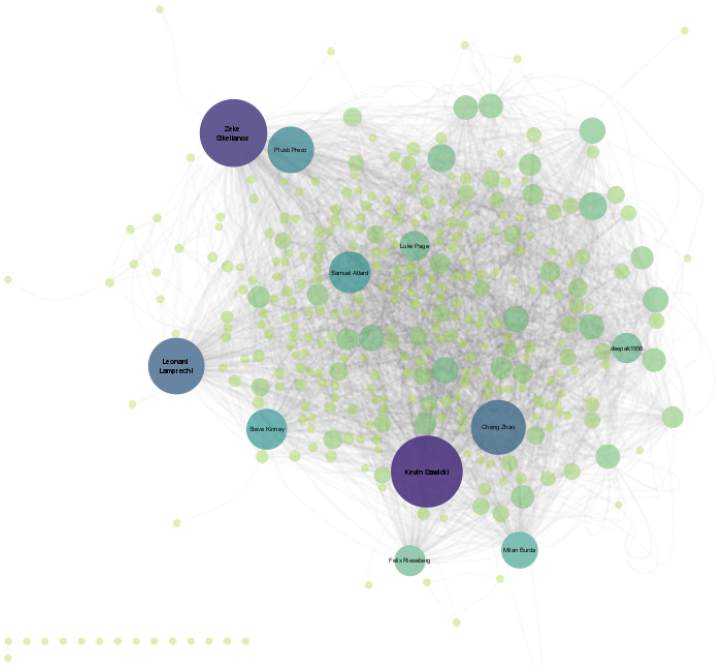

FIGURE 5.9 – Graphe du réseau de collaboration github du projet electron tel que la taille et la couleur des noeuds sont déterminées par leurs degrés de centralité .

FIGURE 5.10 – Collaborateurs ayant le degré de centralité le plus élevé.

Visualisation selon la centralité d'intermédiarité

Cette mesure peut représenter le contrôle d'un collaborateur sur les connexions ou la capacité d'un collaborateur à restreindre les connexions dans le réseau.Un nœud élevé en intermédiarité a le pouvoir de connecter des groupes déconnectés, de servir de courtier d'opinion et de contrôler le flux d'informations. Ici, on remarque que la plus part des noeuds ont une intermédiarité faible ou nulle.

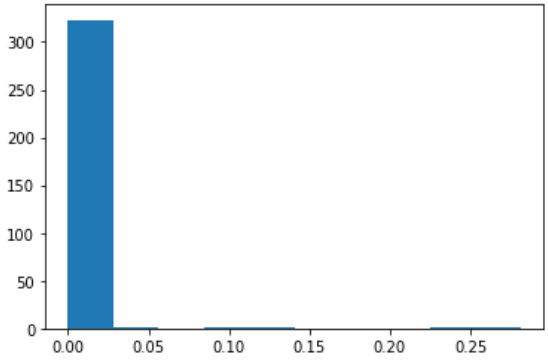

FIGURE 5.11 – Distrubition de la centralité d'intermédiarité selon les noeuds.

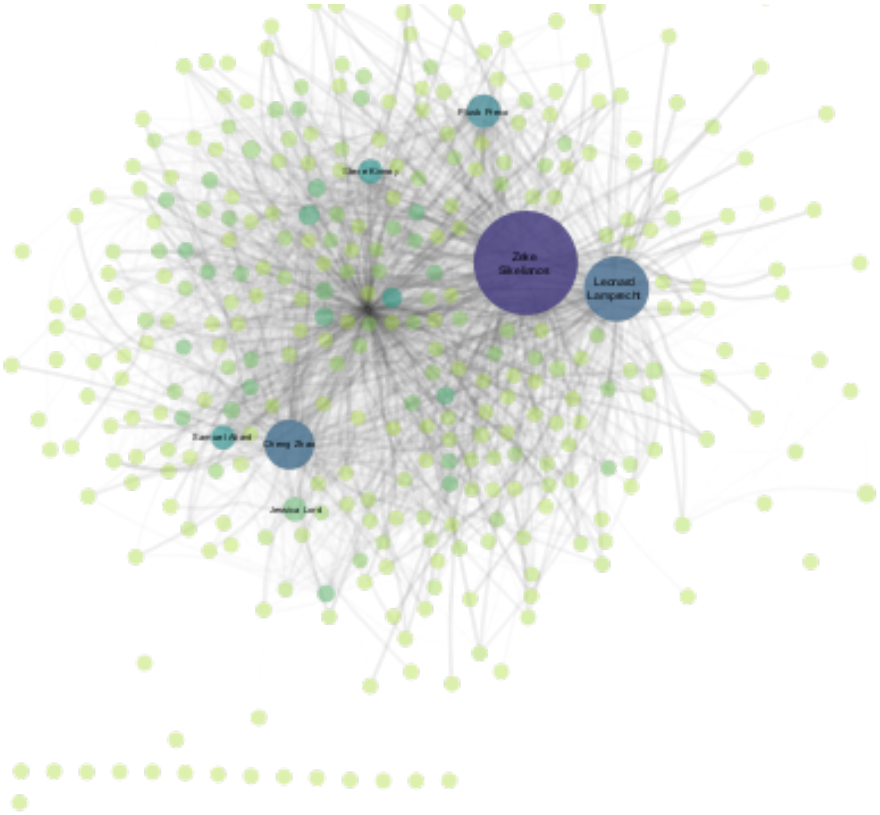

FIGURE 5.12 – Graphe du réseau de collaboration github du projet electron tel que la taille et la couleur des noeuds sont déterminées par leurs intermédiarités.

Ainsi, les noeuds les plus influents selon ce critère se présentent comme suit :

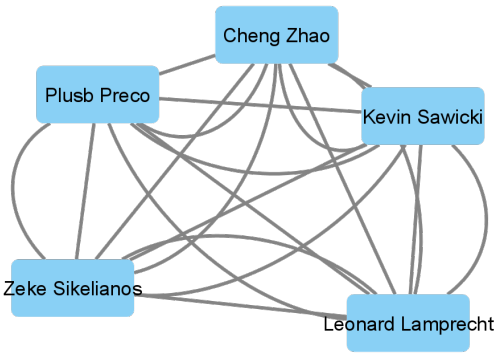

FIGURE 5.13 – Collaborateurs ayant l'intermédiarité la plus élevée.

Visualisation selon la centralité de proximité

Un nœud dont la position est la plus proche (en moyenne) de tous les autres peut obtenir le plus efficacement des informations et les diffuser rapidement dans le réseau. Ainsi, la centralité de proximité est un indicateur de l'indépendance et de l'efficacité de la communication avec les autres nœuds du réseau.

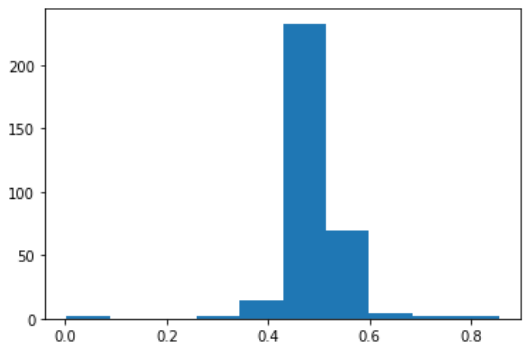

FIGURE 5.14 – Distrubition de la centralité de proximité selon les noeuds.

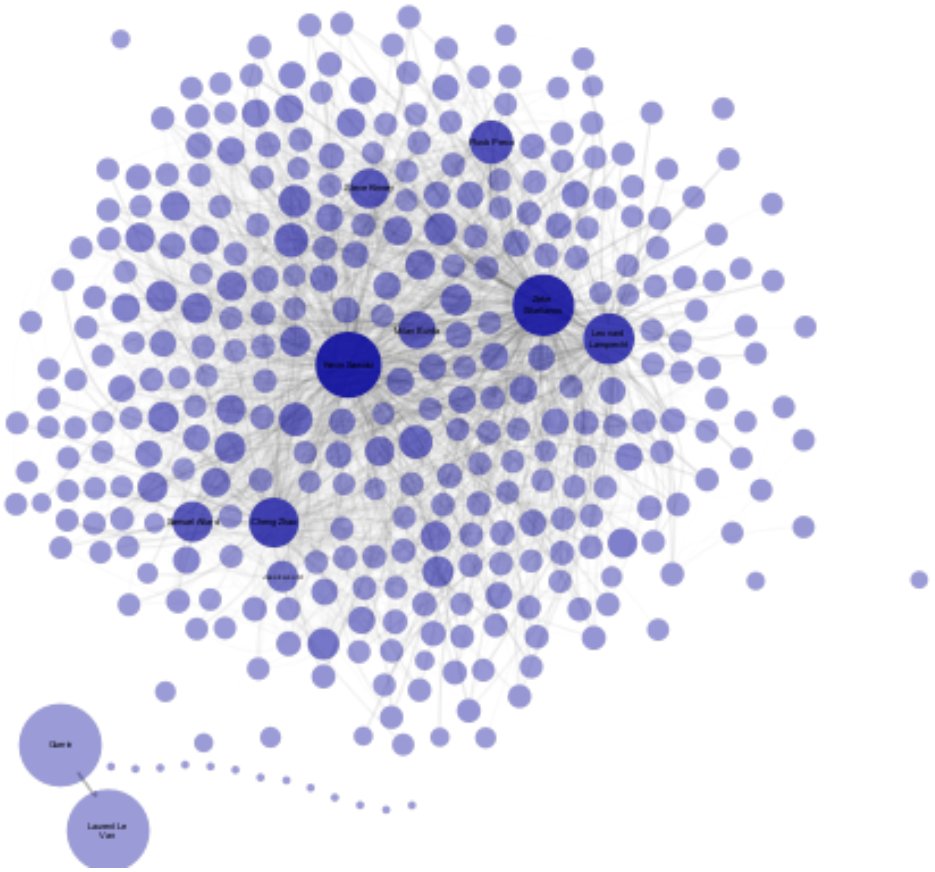

FIGURE 5.15 – Graphe du réseau de collaboration github du projet electron tel que la taille et la couleur des noeuds sont déterminées par leurs proximité.

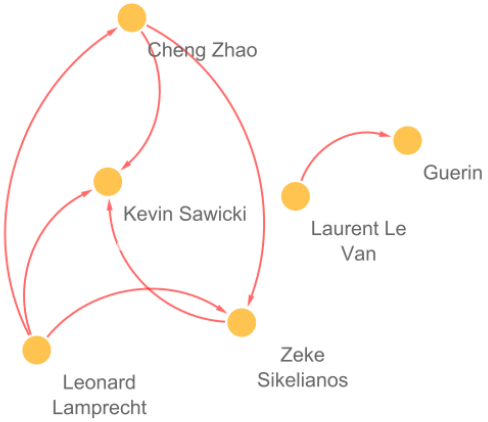

FIGURE 5.16 – Collaborateurs ayant la centralité de proximité la plus élevée.

5.5.3 La plus grande communauté des collaborateurs du réseau Electron

Génralement, une clique est un sous-ensemble d'un réseau dans lequel les acteurs sont plus étroitement et intensément liés les uns aux autres par rapport autres membres du réseau. Dans le cas étudié, les cliques représentent les plus grandes communautés de collaborateurs sur réseau. Le réseau de collaboration du projet électron contient 422 cliques tel que la plus grande clique se compose de 40 noeuds.

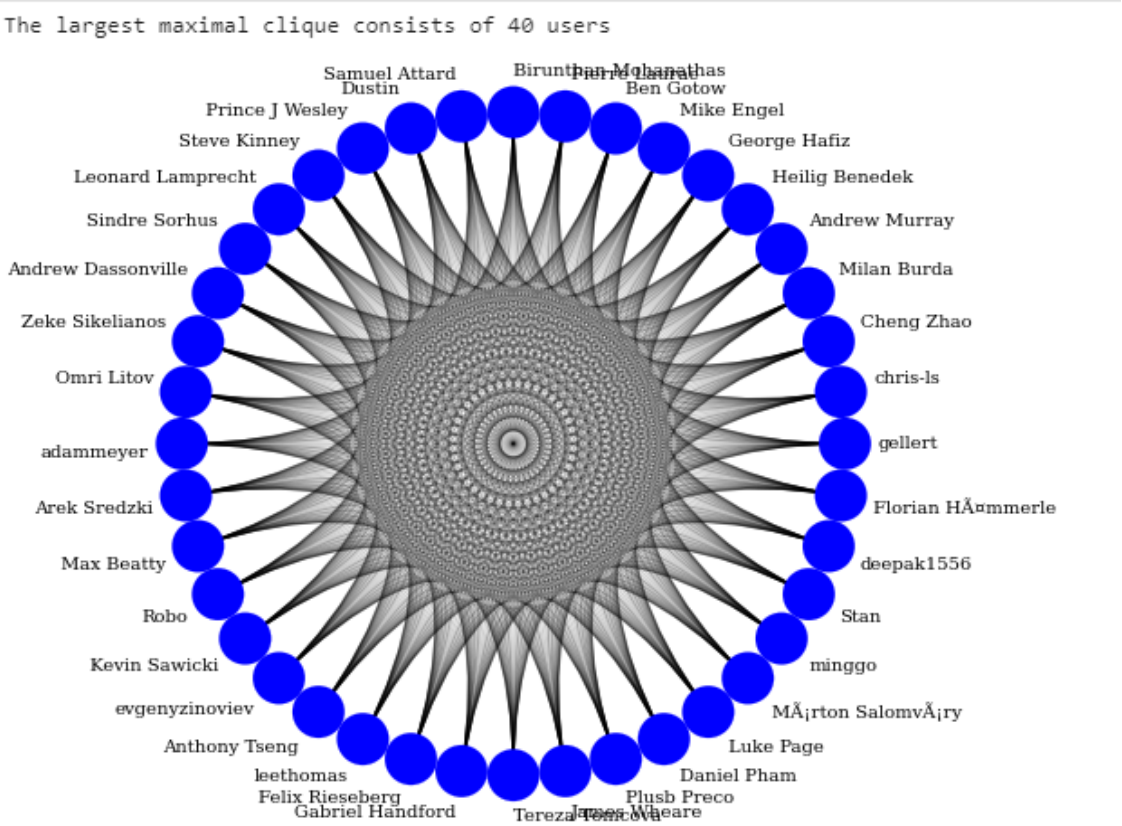

FIGURE 5.17 – La plus grande clique du réseau.

	name	Degree	BetweennessCentrality	EigenvectorCentrality	ClosenessCentrality	ClusteringCoefficient
3	Kevin Sawicki	273	0.284663	0.167577	0.859788	0.087535
6	Zeke Sikelianos	255	0.234840	0.011259	0.822785	0.097175
186	Leonard Lamprecht	208	0.132300	0.007151	0.733634	0.131503
35	Cheng Zhao	201	0.091252	0.159791	0.722222	0.145423
92	Plusb Preco	166	0.048449	0.042222	0.670103	0.186053
4	Samuel Attard	146	0.023349	0.102010	0.643564	0.247709
217	Steve Kinney	142	0.023630	0.027525	0.638507	0.242733
96	Milan Burda	125	0.013090	0.066026	0.617871	0.313032
117	Felix Rieseberg	101	0.013948	0.018908	0.589837	0.381188
13	deepak1556	98	0.010142	0.176068	0.586643	0.434462

FIGURE 5.18 – Caractéristiques SNA des top 10 noeuds classés selon le degré.

5.5.4 Classification du graphe

En utilisant le plugin Clustermaker2 de Cytoscape, nous avons appliqué l'algorithme K-means. Cet algorithme vise le partitionnement de données en deux groupes homogènes appelées clusters en se basant sur les mesures de :*degré, Closeness Centrality, Betweeness Centrality, et l'excentricité*. Nous avons donc obtenus les clusters suivant :

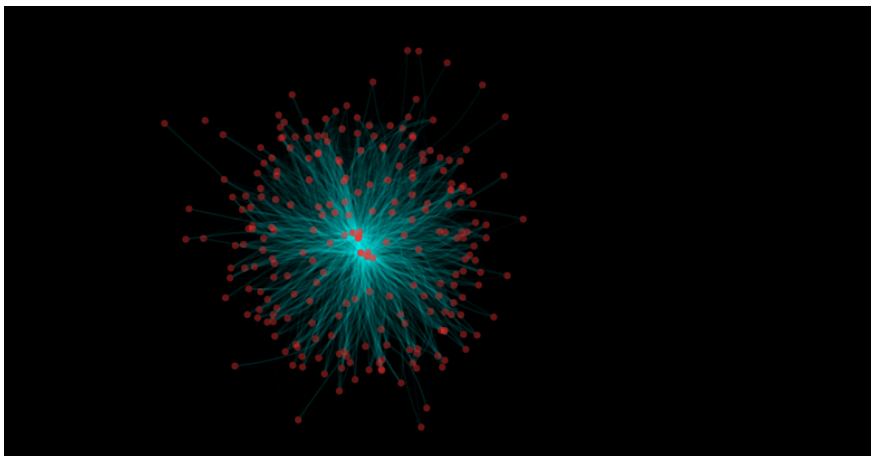

FIGURE 5.19 – Premier Cluster du réseau collaboratif avec : Nombre de noeuds : 237 et densité : 0.118

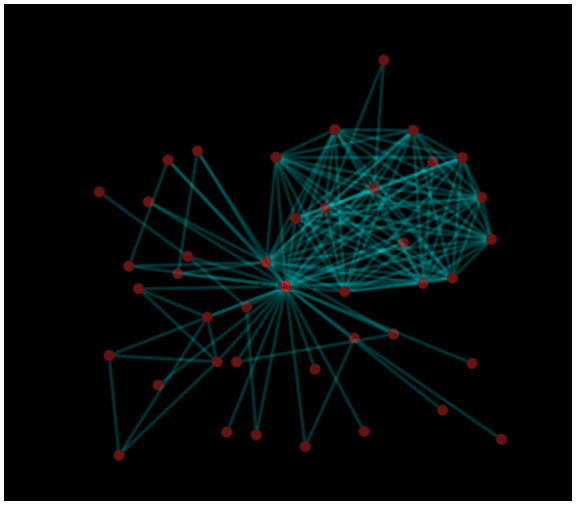

FIGURE 5.20 – Deuxième Cluster du réseau collaboratif avec : Nombre de noeuds : 42 et densité : 0.163

6 | Conclusion

Notre projet consiste à faire une Analyse SNA du réseau collaboratif d'un projet Github et par la suite révéler des contributeurs les plus actifs dans ce projet. Pour atteindre ces objectifs, on a commencé par une extraction des données du projet Github par la commande *git log* puis une préparation et traitement des données à l'aide des bibliothèques de Python : *Pandas, Numpy,* et *Collections* sous forme d'une matrice adjacence .Par l'outil Cytoscape ,on a visualisé les interactions entre les collaborateurs et par la suite fait une analyse des métriques globales et locales du graphes pour enfin classifier les graphes en deux paquets en appliquant l'algorithme K-means sous Cytoscape.

Ce projet a été une propice occasion pour développer notre aptitude de travail en équipe. Il nous a permis de pratiquer nos connaissances théoriques acquises durant cette année dans les domaines de la théorie de graphe et la programmation. C'était également une ouverture sur le domaine de Business Analytics à savoir SNA , Python.

Ce que nous avons présenté sont des efforts qui demeurent ouverts à toute amélioration ou optimisation.

Bibliographie

[1] *Analyse des réseaux sociaux*. URL : https://fr.wikipedia.org/wiki/Analyse_des_rÃľseaux_sociaux.

[2] Laurcitent BEAUGUITTE. *L'analyse des réseaux sociaux*. URL : https://groupefmr.hypotheses.org/3724.

[3] Oskar JARCZYK et al. « GitHub Projects. Quality Analysis of Open-Source Software ». In : (nov. 2014), p. 80-94. DOI : 10.1007/978-3-319-13734-6_6.

[4] Maria MERCANTI-GUÉRIN. « Analyse des réseaux sociaux et communautés en ligne : quelles applications en marketing ? » In : *Management Avenir* (2010), p. 132-153. DOI : 10.3917/mav.032.0132.

[5] Frédéric PENNERATH. *ANALYSE DE RÉSEAUX*. URL : http://www.metz.supelec.fr/metz/personnel/pennerath/Cours/DataScience/7-Analyse%20de%20reseaux.pdf.

[6] *Réseau social*. URL : https://fr.wikipedia.org/wiki/RÃľseau_social.

[7] Shazia TABASSUM et al. « Social network analysis: An overview ». In : *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 8 (avr. 2018), e1256. DOI : 10.1002/widm.1256.

[8] Morgane TUAL. *Qu'est-ce que GitHub, la plate-forme que Microsoft vient de racheter ?* URL : https://www.lemonde.fr/pixels/article/2018/06/04/qu-est-ce-que-github-la-plate-forme-que-microsoft-vient-de-racheter_5309488_4408996.html.