Mise en œuvre

4.1 Introduction

Dans ce dernier chapitre, nous commençons par un bref aperçu des outils utilisés pour réaliser notre application, tels que MySql, Valentina Studio, Netbeans. Nous présentons ensuite l'ensemble des fonctionnalités qu'offre notre application qui vise à implémenter les algorithmes de graphe de fouille de données de la bibliothèque Neo4j Graphe Data Science (GDS) détaillés dans le chapitre précédent, sous forme de captures d'écran avec des descriptions pour bien comprendre chaque étape de la mise en place de notre travail.

4.2 Environnement de développement

Dans ce qui suit nous présentons brièvement l'environnement de développement et différents outils utilisés :

4.2.1 MySQL

MySQL est un système de gestion de bases de données relationnelles. Il fait partie des logiciels de gestion de base de données les plus utilisés au monde selon le classement DB-Engines, autant par le grand public (applications web principalement) que par des professionnels. SQL fait référence au « Structured Query Language » : le langage standard pour les traitements de bases de données.
MySQL est Open Source, c'est-à-dire, les possibilités de libre redistribution, d'accès au code source.

4.2.1.1 Caractéristiques

MySQL est un serveur de bases de données relationnelles SQL développé dans un souci de performances élevées en lecture, ce qui signifie qu'il est davantage orienté vers le service de données déjà en place que vers celui de mises à jour fréquentes et fortement sécurisées. Il est multi-thread et multi-utilisateur.
C'est un logiciel libre, open source, développé sous double licence selon qu'il est distribué avec un produit libre ou avec un produit propriétaire.

4.2.1.2 Fonctionnalités

Deux moteurs principaux sont présents dans MySQL, MyISAM et InnoDB. MyISAM, contrairement à InnoDB, ne supporte ni transactions ni intégrité automatique des tables, il n'est pas destiné aux applications dont la cohérence des données est critique ; cependant, ses performances le font adopter pour des applications ayant besoin d'une base de données simple et peu onéreuse à mettre en œuvre.
Pour les utilisateurs, phpMyAdmin est un outil web souvent disponible pour créer, remplir et utiliser des bases MySQL. [67]

4.2.1.3 Utilisation du MySQL dans notre travail

Dans notre travail nous avons utilisé MySQL comme un entrepôt temporaire pour stocker momentanément les données téléchargés du réseau « StackExchange », car MySQL offre un « Load Dump » optionnel, qui assure un chargement des gros volumes dans un temps court et sans commit.
Cette phase est très importante pour la construction de notre base de données orientée graphe.

4.2.2 Valentina Studio

Valentina Studio est une suite d'outils de gestion de base de données gratuite pour travailler avec toutes les bases de données populaires, y compris MySQL, PostgreSQL, Microsoft SQL Server, MariaDB, SQLite et la propre base de données relationnelle objet ValentinaDB de Paradigma Software. Valentina Studio peut être mis à niveau vers Valentina Studio PRO, qui ajoute SQLDIFF, la création de diagrammes, le développement de formulaires, les outils de création de rapports, Visual Query Builder, l'éditeur SQL avancé et plus encore.
Valentina Studio inclut Report Viewer pour afficher les rapports. Valentina Studio agit également en tant que client gratuit pour travailler avec Valentina Forms, un système de formulaires de données qui peut être hébergé localement ou servi à partir de Valentina Server. Les formulaires sont un puissant paradigme visuel permettant aux employés de travailler avec les données de l'entreprise et sont scriptables avec JavaScript. [34]
Valentina Studio par sans éditeur de requêtes « Query Editor » on à élaborer des tables intermédiaires par l'exécution des requêtes visuelles, ce qui nous permet d'exporter les données en format CSV pour les importer a la base de données Neo4j et construire notre base de données orienté graphe.
La figure 4.1 représente l'environnement de travail de Valentina Studio.

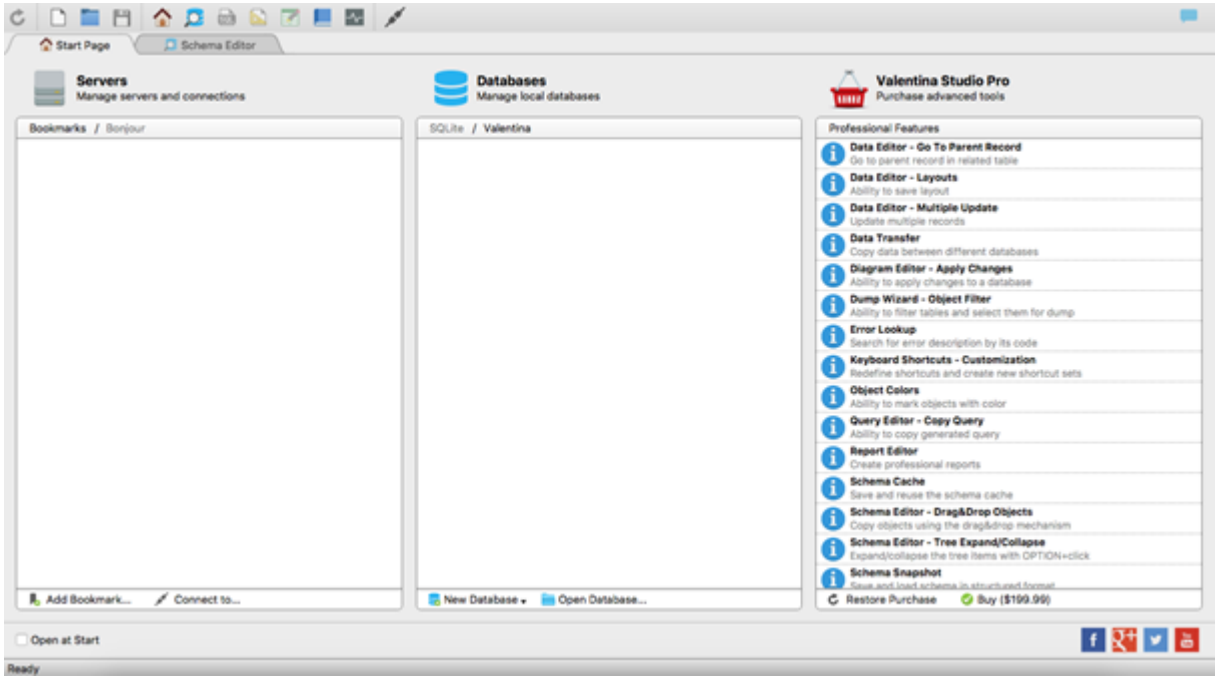

FIGURE 4.1: Environnement de travail sur valentina studio

4.2.3 Neo4j

Neo4j permet de représenter les données en tant que nœuds reliés par un ensemble d'arcs, ces objets possédant leurs propres propriétés. Les propriétés sont constituées d'un couple de clé-valeurs de type simple tel que chaînes de caractères ou numérique ; celles-ci peuvent être indexées. La modélisation est très proche des concepts métier, il n'est pas nécessaire d'utiliser de clés dans Neo4j, car les relations ont une existence propre. L'absence de modélisation rigide rend Neo4j bien adapté à la gestion de données changeantes et de schémas évoluant fréquemment.
La base de données Neo4j est construite pour être extrêmement performante pour traiter les liens entre nœuds. Ces performances sont dues au fait que Neo4j pré-calcule les jointures au moment de l'écriture des données, comparativement aux bases de données relationnelles qui calculent les jointures à la lecture en faisant appel aux Index et à la logique de clés. Ce qui fait de Neo4j une technologie adaptée à de larges ensembles de données connectées.
Les traversées utilisent le langage de requête Cypher, standard de parcours des connexions, élaboré dans le but de réaliser plus simplement que SQL les opérations de parcours ou d'analyse de proximité qui sont les plus courantes lorsqu'on traite des données connectées. Les bases de données de graphes sont des outils puissants pour répondre à des requêtes faisant intervenir des relations entre objets. La recherche du plus court chemin entre deux points du graphe permet par exemple de mettre en place des profils utilisant liens sociaux, géographie et analyse d'impact. [68]

Neo4j contient plusieurs bibliothèques, ont à utilisés :

- APOC

La bibliothèque APOC comprend de nombreuses procédures et fonctions (environ 450) pour vous aider dans de nombreuses tâches différentes dans des domaines tels que l'intégration de données, les algorithmes graphiques ou la conversion de données.

- Graph Data Science (GDS)

La bibliothèque Neo4j Graph Data Science (GDS) fournit des capacités analytiques étendues centrées sur les algorithmes de graphes. Elle comprend des algorithmes pour la détection de communauté, la centralité, la similitude des nœuds, la recherche de chemin et la prédiction de liens, ainsi que des procédures de catalogue de graphiques conçues pour prendre en charge les flux de travail de science des données et les tâches d'apprentissage automatique sur vos graphiques. Toutes les opérations sont conçues pour une échelle et une parallélisation à grande échelle, avec une API personnalisée et générale conçue pour le traitement global des graphes et des structures de données en mémoire compressées hautement optimisées.
Neo4j présente de nombreux avantages, sa réactivité dans la gestion de données, flexibilité et évolutivité, performance. . . .etc, ce qu'il nous a encouragé de l'utiliser pour stocker les données graphes et pour exécuter les requêtes de la bibliothèque Graph Data Science (GDS)
La figure 4.2 représente l'interface Neo4j version 1.2.9 de Neo4j Desktop et la version 3.5.17 pour la base de données utilisées dans notre travail.

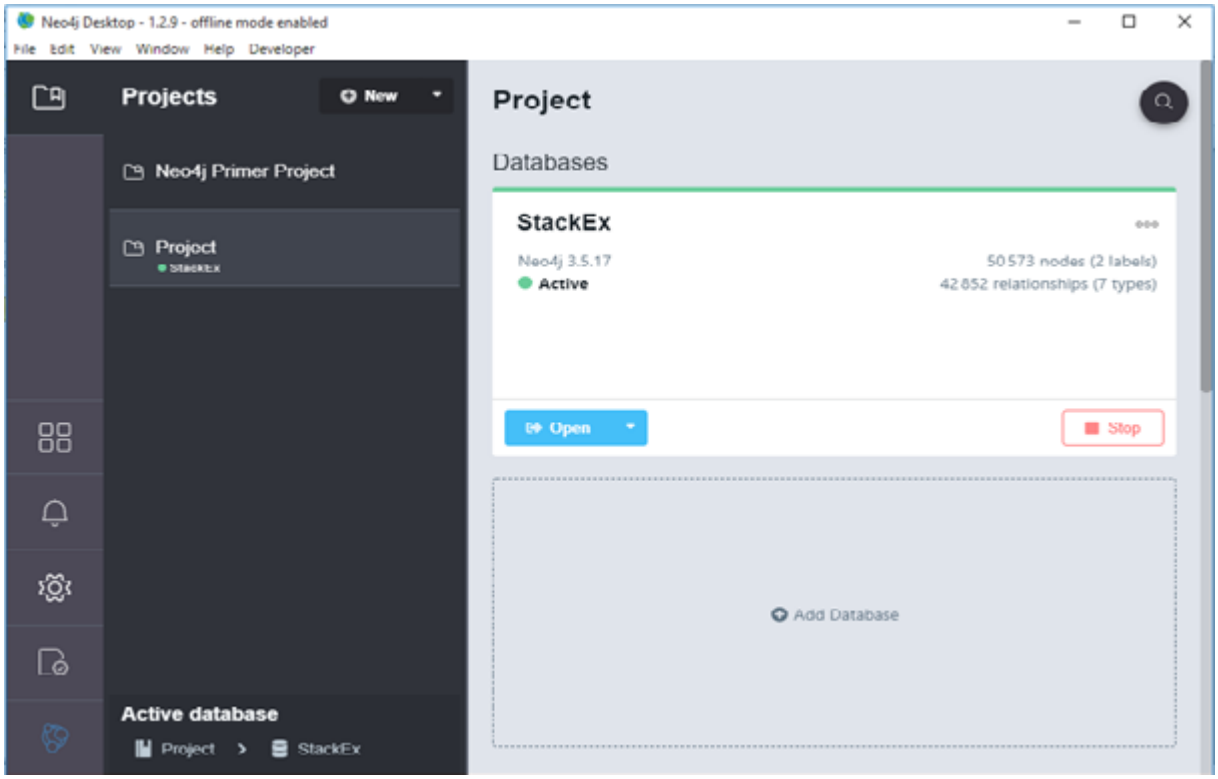

FIGURE 4.2: Interface Neo4j version 1.2.9 Desktop

4.2.4 NetBeans

NetBeans est un environnement de développement intégré (EDI), placé en open source. En plus de Java, NetBeans permet la prise en charge native de divers langages tels le C, le C++, le JavaScript, le XML, le Groovy, le PHP et le HTML, ou d'autres (dont Python et Ruby) par l'ajout de greffons. Il offre toutes les facilités d'un IDE moderne (éditeur avec coloration syntaxique, projets multi-langage, refactoring, éditeur graphique d'interfaces et de pages Web).
Compilé en Java, NetBeans est disponible sous Windows, Linux, Solaris (sur x86 et

SPARC), Mac OS X ou sous une version indépendante des systèmes d'exploitation (requérant une machine virtuelle Java).
Un environnement Java Development Kit JDK est requis pour les développements en Java. NetBeans constitue par ailleurs une plateforme qui permet le développement d'applications spécifiques (bibliothèque Swing (Java)). L'IDE NetBeans s'appuie sur cette plate forme. [69]
Nous avons utilisé NetBeans comme environnement de programmation en JAVA, qui nous a permis de développer une application destinée pas forcément à un expert mais à un utilisateur même s'il ne connait pas les technologies sous-jacentes (neo4j, les algorithmes de fouille de graphe...etc).

4.3 Jeux de données

Nous avons entamé le travail d'implémentation par la reconstruction de la source externe, contenant les donées téléchargées de StackExchange, en particulier celles du sous-réseau Data Science dédié pour la communauté des professionnels de la science des données et des spécialistes de l'apprentissage automatique.

Etape 1 : exécution de fichier téléchargé depuis Stackexchange en utilisant la fonctionnalité « load dump », cette étape se charge de la création de la base de données, les tables et l'alimentation de ces tables. La figure 4.3 présente le résultat de l'étape 1, la base de données « stackexchange » de Mysql est ouverte dans Valentina Studio.

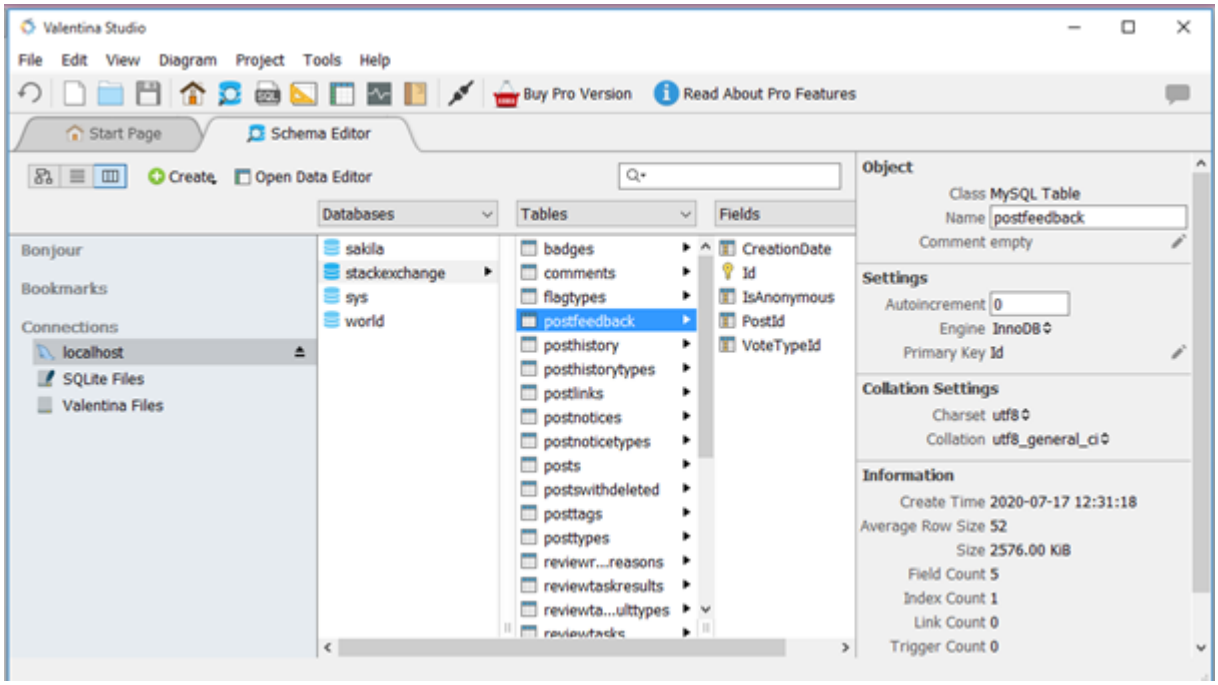

FIGURE 4.3: La base de données « stackexchange » de Mysql est ouverte dans Valentina Studio

Etape 2 : création d'une connexion vers MySQL depuis Valentina Studio, et puis l'extraction des données nécessaires pour notre travail par l'exécution des requêtes SQL, les résultats seront exportés se forme des fichiers « .CSV », pour pouvoir les importer en Neo4j. Dans la figure 4.4 nous présentons la requête visuel pour extraire les données des

commentaires des utilisateurs, suivi de requête SQL correspondante et requête SQL pour extraire les données des liens entre les utilisateurs .

- Requête SQL pour extraire les données des utilisateurs :

```
SELECT `users`.`Id`, `users`.`DisplayName`
FROM `users`
```

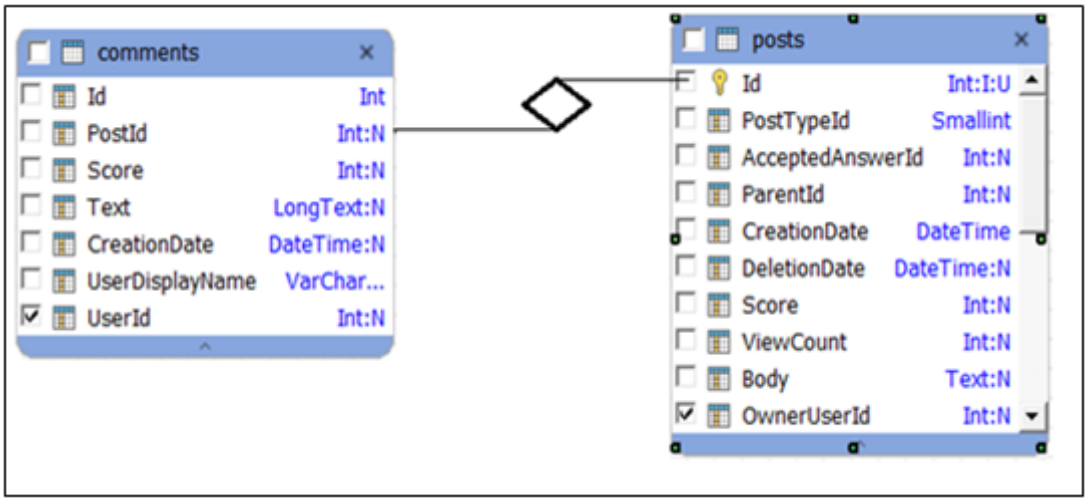

FIGURE 4.4: La requête visuel pour extraire les données de la relation User -> Comment -> User

- Requête SQL pour extraire les données des commentaires des utilisateurs :

```
SELECT `comments`.`UserId`, `posts`.`OwnerUserId`, COUNT(*)
FROM `posts`
INNER JOIN `comments` ON `posts`.`Id` = `comments`.`PostId`
GROUP BY `comments`.`UserId`, `posts`.`OwnerUserId`
```

- Requête SQL pour extraire les données des Liens entre les utilisateurs :

```
SELECT `posts`.`OwnerUserId`, `T`.`OwnerUserId`, COUNT(*)
FROM `postlinks`
INNER JOIN `posts` ON `postlinks`.`PostId` = `posts`.`Id`
INNER JOIN `posts` `T` ON `postlinks`.`RelatedPostId` = `T`.`Id`
WHERE ( `posts`.`OwnerUserId` <> `T`.`OwnerUserId` )
GROUP BY `T`.`OwnerUserId`, `posts`.`OwnerUserId`
```

Dans la figure qui suit (Figure 4.5), nous présentons le résultat de la première requête en format « .csv ».

	A	B
1	Id	DisplayName
2	84580	MementoMori
3	84581	jamal mohammadi
4	84582	Nitin Singh
5	84583	iamLalit
6	84584	Manas shukla
7	84585	mangatinanda
8	84586	Simen Isaken
9	84587	HAO CHEN
10	84588	Mark
11	84589	Kalpit
12	84590	Yaroslav Baranov
13	84591	HannahB
14	84592	river_bell
15	84593	orangepips
16	84594	tatu tuta

utilisateurs

Prêt

FIGURE 4.5: Les données des utilisateurs en format « .csv »

Etape 3 : Création d'une base de données sous Neo4j et importation des données depuis les fichiers csv. Par l'exécution des requêtes Cypher :

- Création de la relation Comment entre les utilisateurs :

```
:auto using Periodic Commit LOAD CSV with headers FROM "file:/comment.
csv" AS row FIELDTERMINATOR ';' match (n:Users {userId: coalesce(toInteg
er(row.OwnerUserId),0)}), (b:Users {userId: coalesce(toInteger(row.UserI
d),0)}) CREATE (b) -[:Comment {NbrOfComments: toInteger(row.NbrOfComment
s)}]- > (n)
```

- Création de la relation DownVote entre les utilisateurs :

```
: auto using Periodic Commit LOAD CSV with headers FROM "file:/downvot
e.csv" AS row FIELDTERMINATOR ';' match (n:Users {userId: coalesce(toInt
eger(row.OwnerUserId),0)}), (b:Users {userId: coalesce(toInteger(row.Use
rId),0)}) CREATE (b) -[:Downvote {NbrOfDownvotes: toInteger(row.NbrOfDow
nvotes)}]- > (n)
```

Le tableau 4.1 représente les caractéristiques de notre base de données utilisée, créée dans le Neo4j « StackEx ».

Noeud	
Etiquette	**Nombre**
Users	50 001
Tags	572
Relations	
Type	**Nombre**
Comment	14 901
Answer	11 950
Upvote	4 130
Downvote	591
Favoritevote	2 758
Relatedlink	653
LieeA	7 869

TABLE 4.1: Description de la base de données orientée graphe « StackEx » utilisée

Etape 4 : Installation des Plugins (les bibliothèques) cité dans 4.2.3. sur Neo4j pour pouvoir exécuter les algorithmes de neo4j.

4.4 Application développée

4.4.1 Intérêt d'application

L'intérêt de notre application est de permettre aux utilisateurs finaux de ne pas écrire les algorithmes de graphe de fouille de données eux-mêmes, et aussi d'avoir une interface conviviale pour pouvoir analyser les données de réseau StackExchange (sous-réseau "Data Science") et les réseaux similaires de Questions /Réponses.

4.4.2 L'objectif d'application

L'objectif de notre application est de préparer, manipuler et analyser les données dans l'optique de les transformer en connaissance actionnable et en outil d'aide à la décision pour l'utilisateur.

4.4.3 Schéma général et enchainement de l'application

Pour bien organiser notre travail nous avons schématisé l'application et l'enchainement de ses étapes d'exécution comme le montre la figure 4.6.

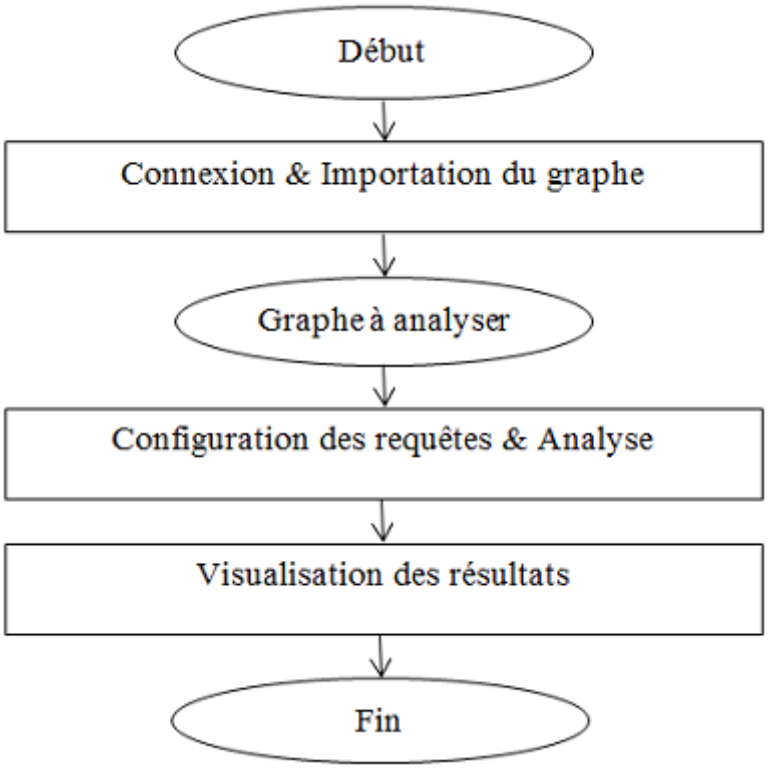

FIGURE 4.6: Enchainement des étapes d'exécution de l'application

Dans l'étape « Configuration des requêtes et Analyse » l'utilisateur doit faire le choix entre les types d'analyses (familles d'algorithmes), analyse de centralité, détection de communauté, similarité, et prédiction des liens (Figure 4.7), puis pour chaque type d'analyse doit choisir l'algorithme d'analyse (Figure 4.8), et après choix de configuration proposée (Figure 4.9).

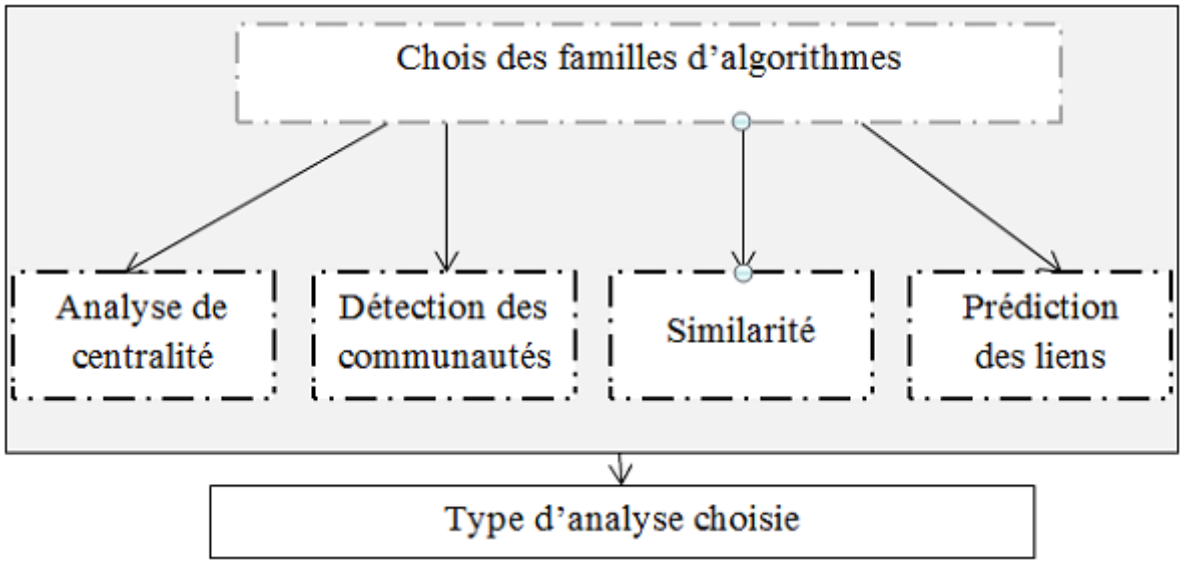

FIGURE 4.7: Choix de type d'analyse

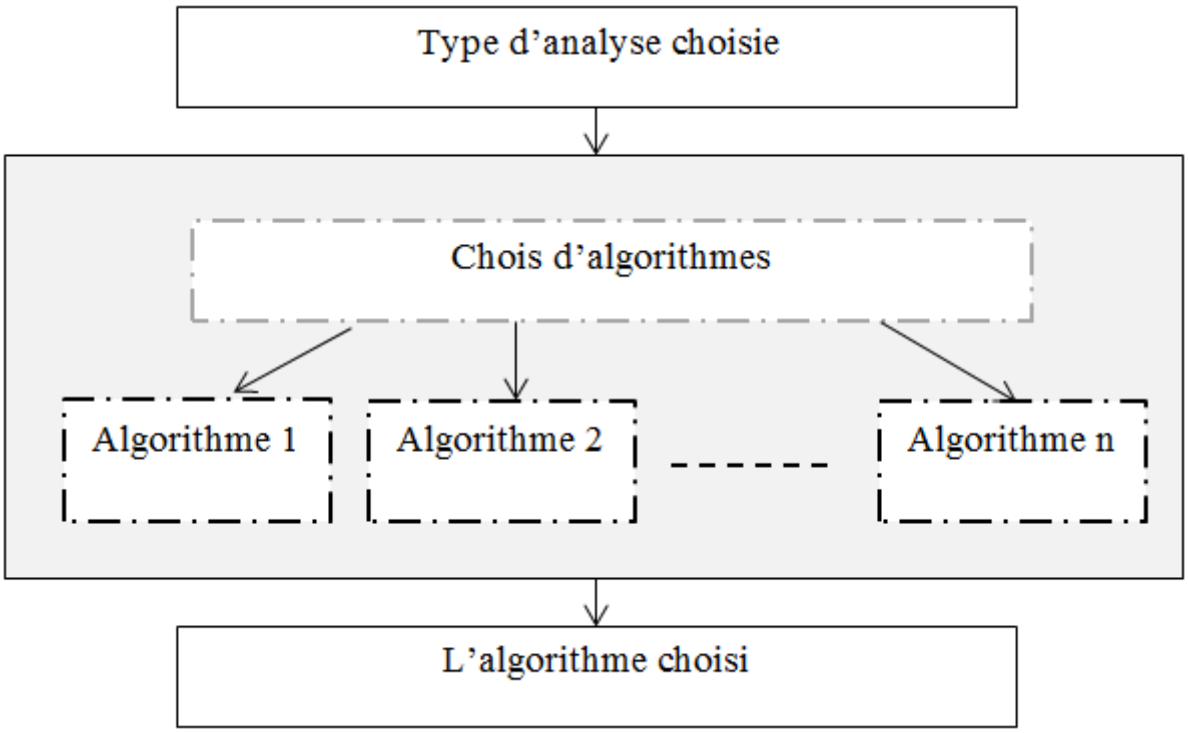

FIGURE 4.8: Choix d' algorithme d'analyse

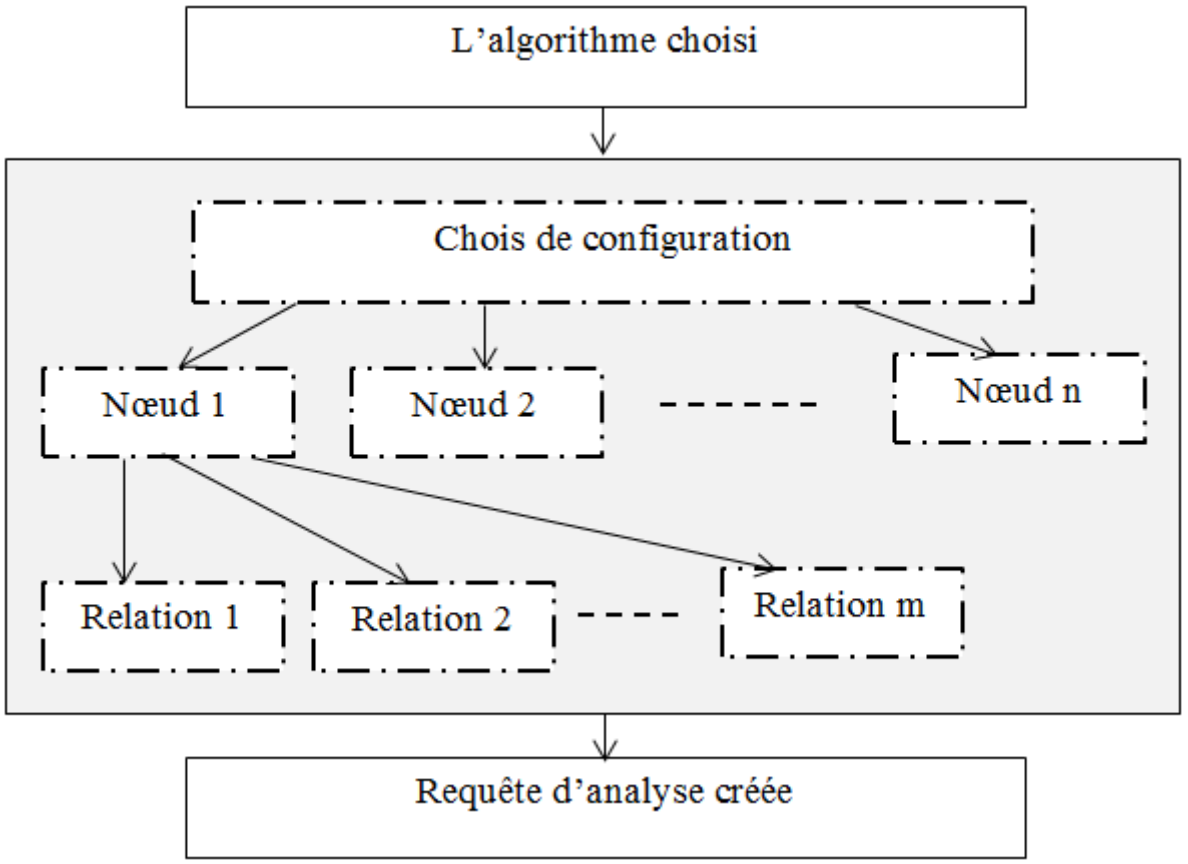

FIGURE 4.9: Choix de configuration et création de la requête Cypher d'analyse

4.4.4 Interfaces d'application

Le projet réalisé contient deux interfaces, une de connexion et l'autre d'analyse. Dans ce qui suit nous détaillons le fonctionnement et le rôle de chaque élément via la description des interfaces établits.

• Une interface de connexion qui se charge de se connecter à la base de données en entrant le lien de la base de données avec le nom de l'utilisateur (username) et le mot de passe (la base de données doit être active pour pouvoir se connecter à l'application).

FIGURE 4.10: Interface de connexion

• Une deuxième interface pour l'analyse des données de notre base de données en appliquant les algorithmes de Neo4j cités dans le chapitre précédant.

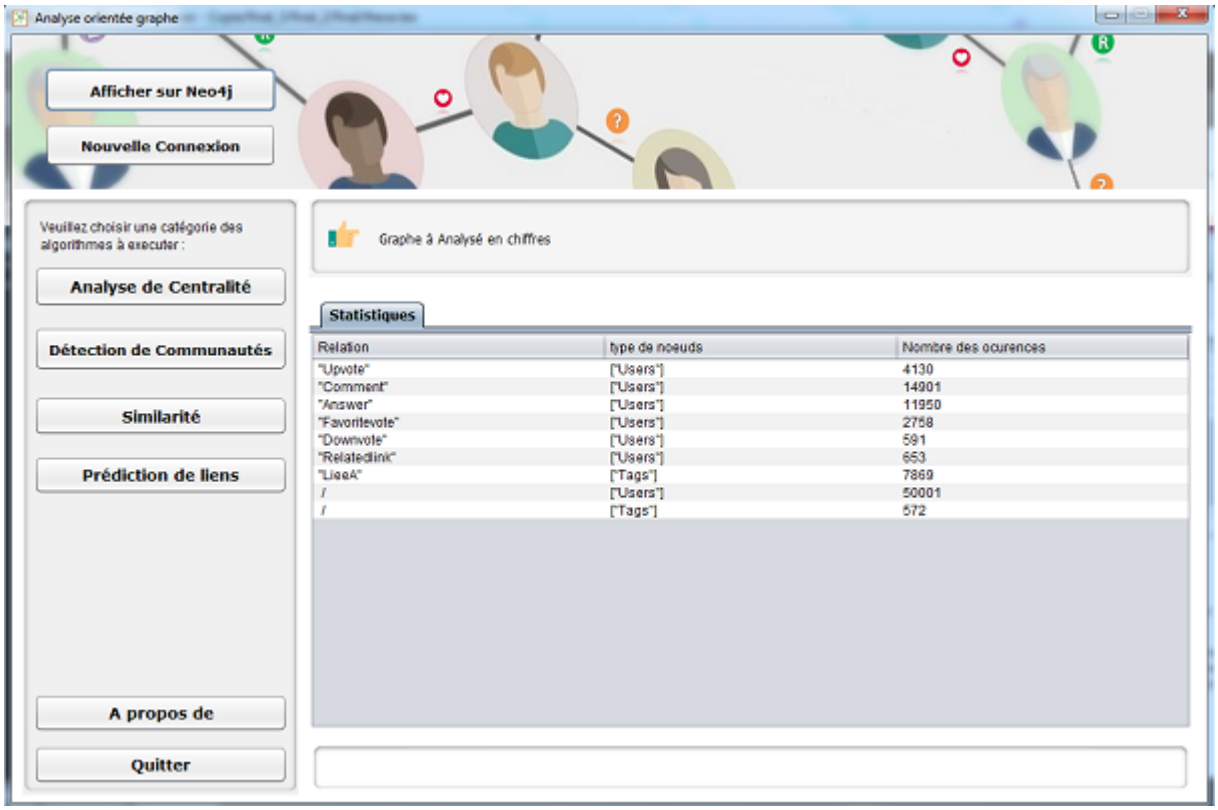

FIGURE 4.11: Interface d'analyse

4.4.5 Explication des différents composants de l'interface

Dans cette partie nous expliquons le rôle de chaque élément :

• La première partie est un panneau de contrôle (figure 4.12) qui contient les différentes familles des algorithmes de Neo4j, chaque famille contient plusieurs algorithmes sous formes d'une liste, et chaque algorithme a une description qui s'affiche lors du déplacement de la souris au-dessus.

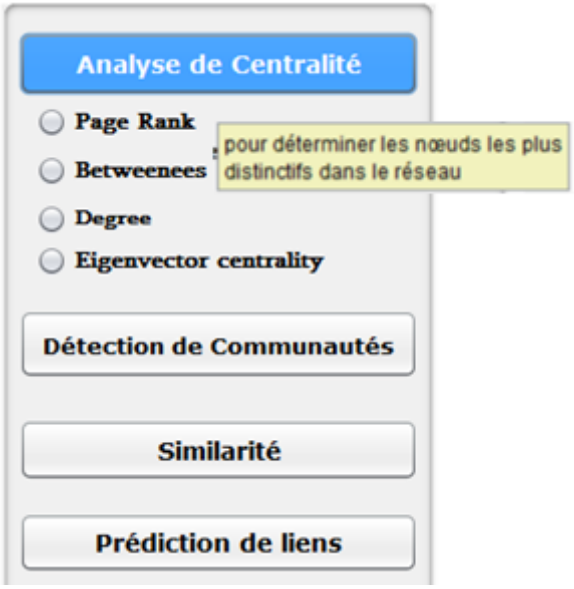

FIGURE 4.12: Panneau de contrôle

• La 2ième partie est un panneau de configuration (figure 4.13), qui affiche les choix de configuration pour chaque algorithme d'analyse (les nœuds et les relations de la base de données).

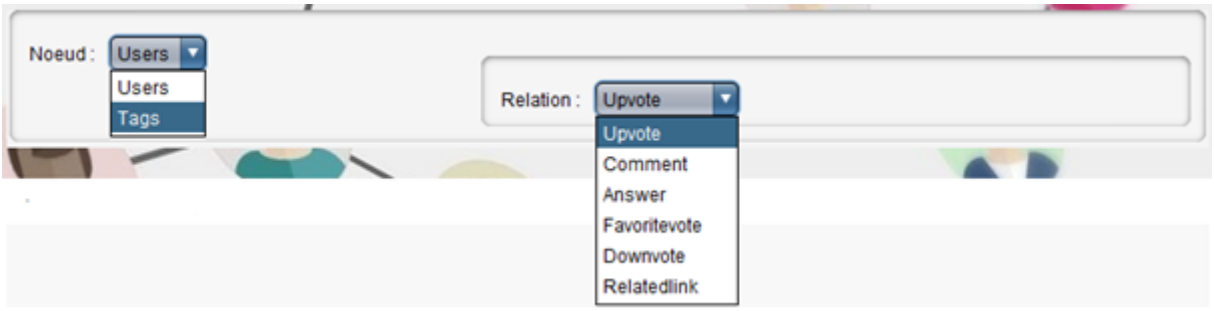

FIGURE 4.13: Panneau de configuration

• La 3ième partie est pour l'affichage des résultats de l'exécution en format d'un tableau ou bien d'un graphe (cité dans 4.4.6), on peut toujours changer l'ordre de résultats du tableau alphabétiquement selon chaque colonne en cliquant sur cette dernière.

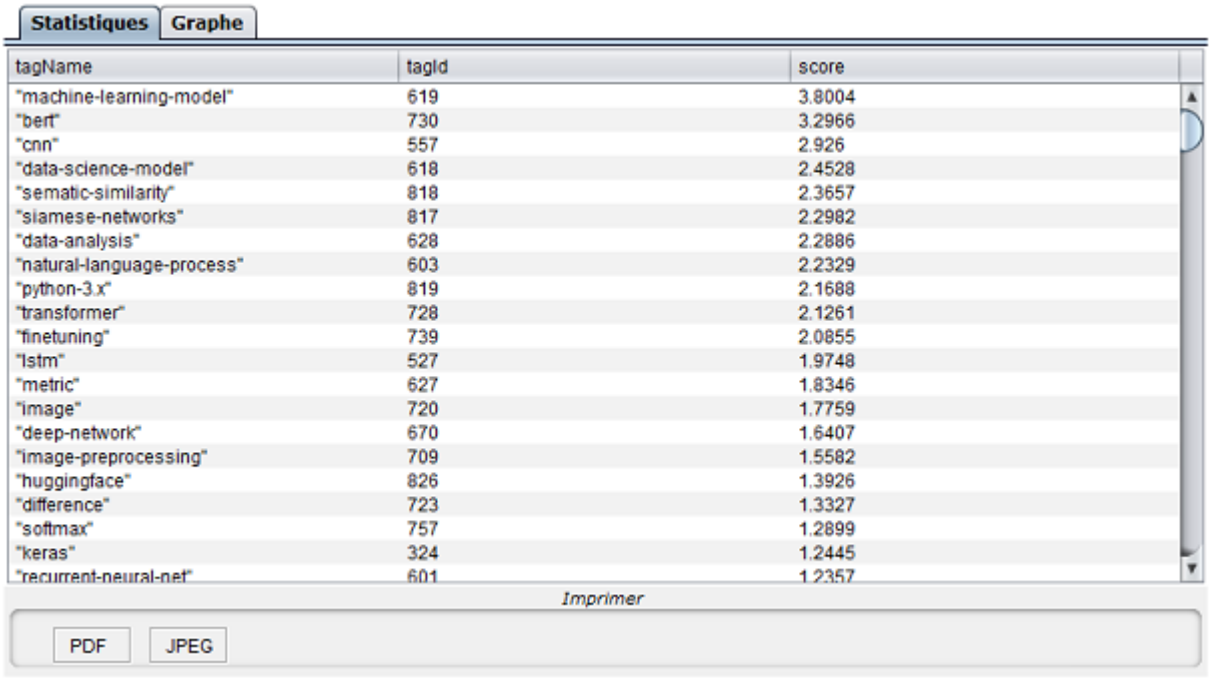

FIGURE 4.14: Panneau d'affichage des résultats

• La 4ième partie facilite l'utilisation de cette application en affichant des instructions d'aide à l'utilisateur (figure 4.15).

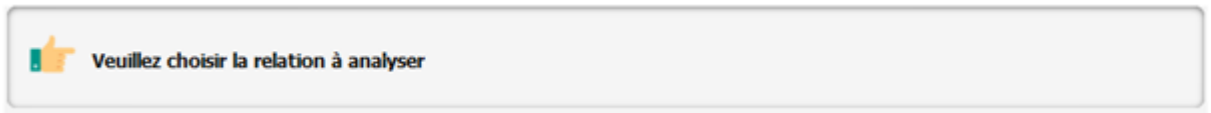

FIGURE 4.15: Panneau d'aide

• Autres boutons :

1- Afficher sur Neo4j : Un bouton qui permet à l'utilisateur d'ouvrir la base de données dans le navigateur par défaut de l'ordinateur utilisé avec un seul clic.

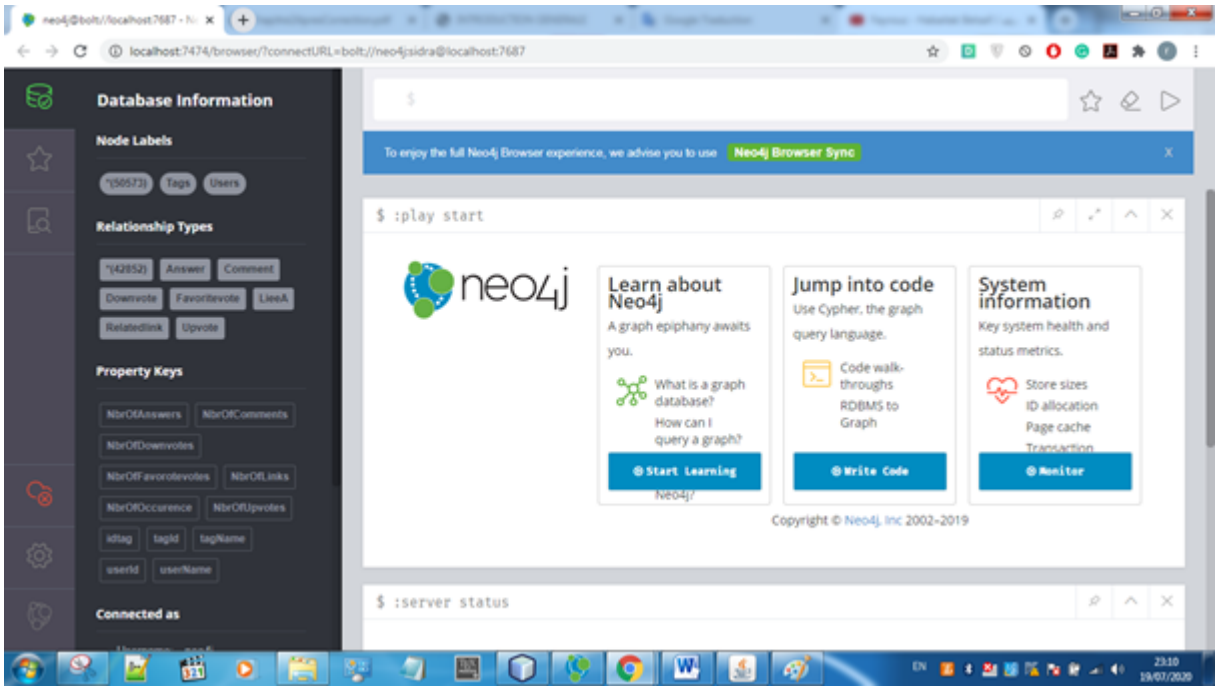

FIGURE 4.16: Base de données connectée dans un navigateur par défaut

2- A propos de : Un bouton qui qui permet d'afficher des informations des informations sur l'application.

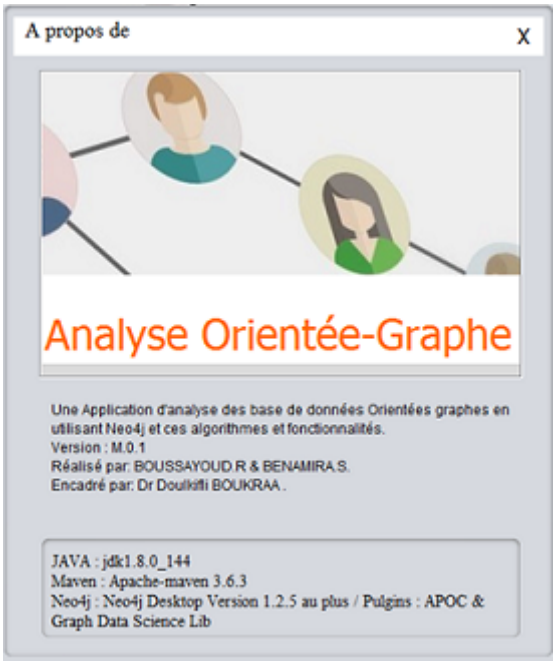

FIGURE 4.17: JFrame qui contient des informations sur l'application

3– Nouveau Graphe : Un bouton qui permet d'afficher l'interface de connexion, pour connecter à la nouvelle base de données (Figure 4.10).

- Traitements des erreurs et les cas exceptionnels :

1- Un message d'alerte qui s'affiche s'il y a un problème dans la connexion à la base de données lors de l'utilisation de l'application.

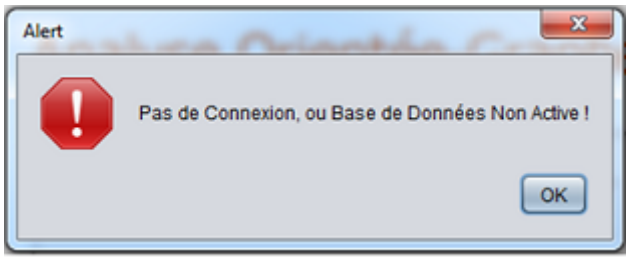

FIGURE 4.18: Message d'alerte qui s'affiche s'il y a un problème dans la connexion

2- Un message d'erreur qui s'affiche dans la première interface lors d'une fausse tentative de connexion à la base de données, si l'utilisateur a saisi des fausses informations ou bien la base de données n'est pas active.

FIGURE 4.19: Un message d'erreur qui s'affiche dans la première interface lors d'une fausse tentative de connexion à la base de données

3- Un message d'alerte qui s'affiche s'il y a un problème dans les données, certains algorithmes sont basés sur le principe d'élimination des données ; le cas de l'algorithme de similarité (les nœuds qui n'ont pas des relations entre eux).

FIGURE 4.20: Message d'alerte pour le cas des données non compatibles

4.4.6 Représentation graphique

Notre application contient plusieurs représentations graphiques qui se différent selon le type des résultats de chaque algorithme.

4.4.6.1 Diagramme en bâtons

Pour représenter les résultats des algorithmes d'analyse de centralité (les algorithmes de classification) et les algorithmes de prédiction des liens, le diagramme en batôns est utilisé pour afficher vingt nœuds qui ont les scores les plus élevés où chaque bâton illustre un nœud (figure 4.21). L'utilisateur peut comparer facilement les données à l'œil nu.

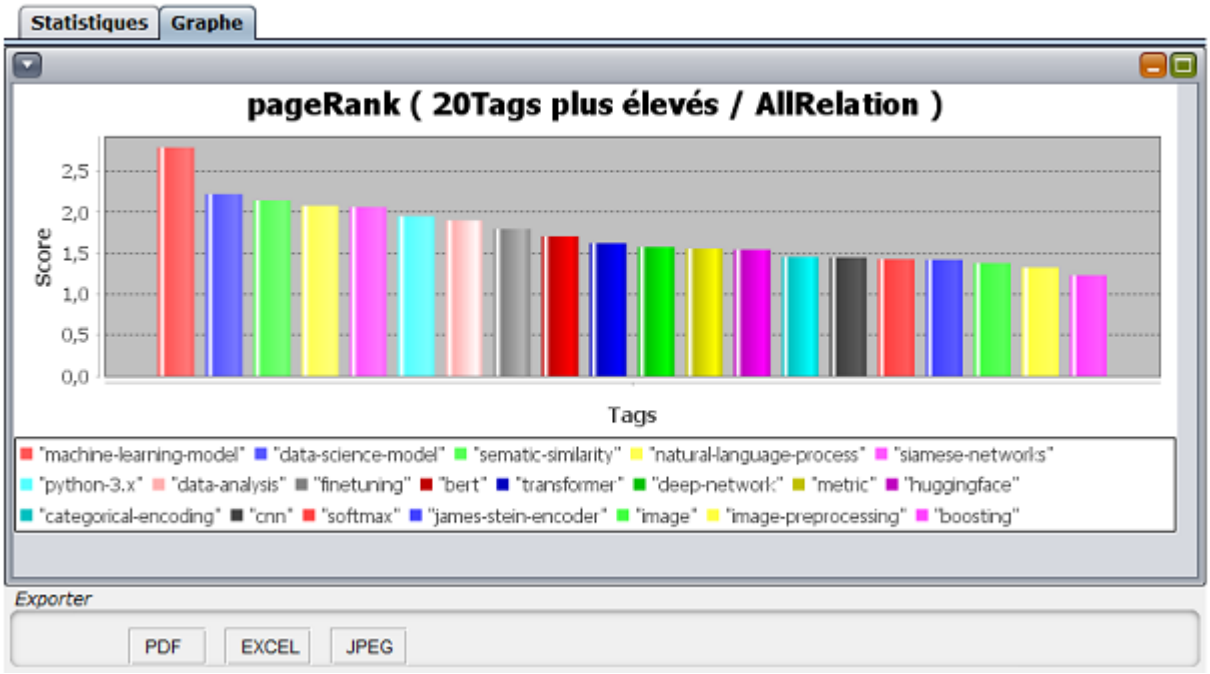

FIGURE 4.21: Représentation des résultats sous forme de diagramme en bâtons

4.4.6.2 Les cellules

Pour représenter les résultats des algorithmes de détection de communautés où chaque cellule représente une communauté avec les nœuds qui lui appartiennent, comme le montre la figure 4.22.

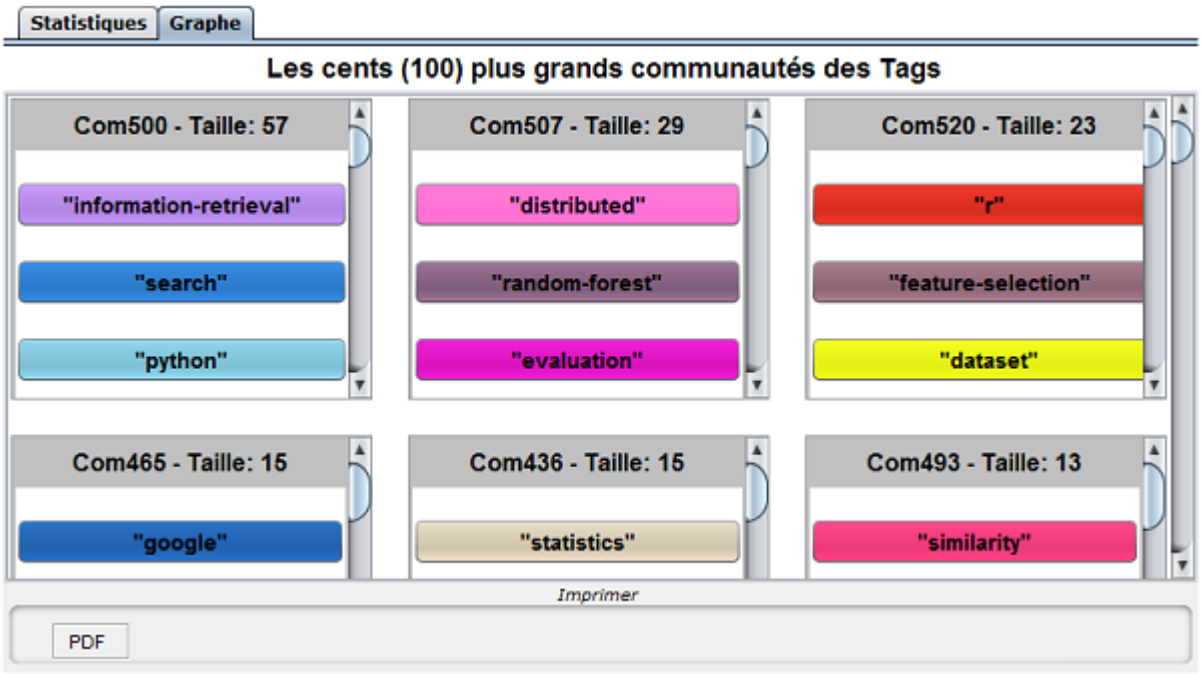

FIGURE 4.22: Représentation graphique des résultats de l'algorithme de louvain selon toutes les relations sous forme des cellules

4.4.6.3 Diagramme en secteurs (camembert)

Utilisé pour présenté les résultats des algorithmes de similarité, ils permettent d'illustrer le pourcentage de similarité et de différence entre deux nœuds (Figure 4.23).

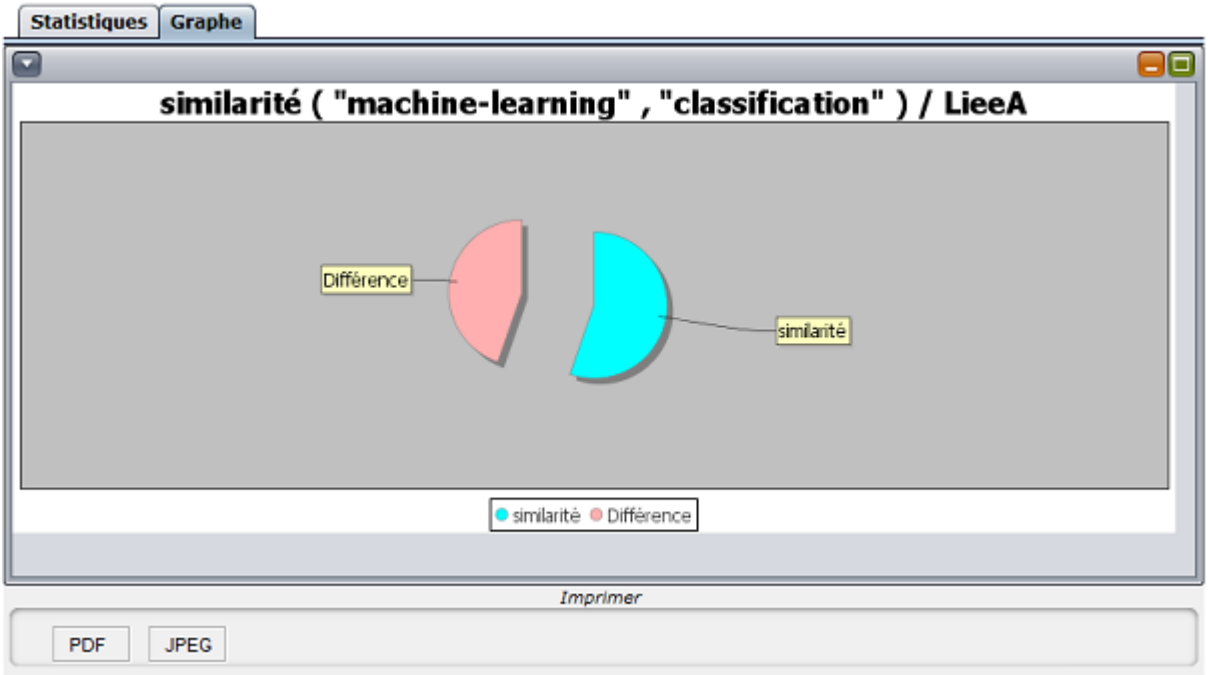

FIGURE 4.23: Représentation graphique des résultats de l'algorithme de similarité entre les tags (machine-learning) et (classification) selon la relation « LieeA »

Après avoir présenté les différents outils et les plateformes utilisées dans ce travail et l'application réalisée, dans la section suivante, nous allons expliquer quelques analyses réalisées puis nous analysons les résultats obtenus.

4.5 Résultats, interprétation, utilisation

Dans cette section, nous présentons plusieurs analyses en utilisant le modèle opérationnel décrit précédemment. Ces expériences sont faites en modifiant les valeurs des paramètres d'entrée afin d'obtenir des résultats pour des cas différents, puis analysons ces résultats pour tirer des connaissances et des conclusions sur les données étudiées. Parmi les paramètres d'entrée que l'on peut utiliser :

- Paramètres d'entrées directs :
  - type d'algorithme d'analyse,
  - type de nœuds,
  - type de relations.
- Paramètres d'entrées dont les valeurs sont définies aléatoirement et automatiquement :
  - nombre d'itérations,
  - orientation des relations,
  - facteur d'amortissement,
  - utilisation des poids.

4.5.1 Analyse de centralité

Dans ces analyses, nous considérons les valeurs fixes des paramètres présentées dans le tableau 4.2 et nous changeons la valeur de la relation à chaque fois (tableau 4.3), tandis que les paramètres de sortie sont l'identifient des nœuds et la valeur de l'analyse obtenues (score).

Paramètre	Valeur
Algorithme	Degré de centralité
	Page Rank
	Betweeness
	Eingenvector centrality
Nœud	Users

Table 4.2: Tableau des paramètres d'analyse de centralité

Les résultats obtenus sont des classements des candidats selon le score, mais nous prenons juste celui qui a la plus grande valeur.

Analyse \ Relation	Degré de centralité		Page Rank		Betweeness		Eingenvector centrality	
	userId	Score	userId	Score	userId	Score	userId	Score
Comment	836	870.0	45264	31.997	381	3742655.17	28175	55.984
Answer	836	283.0	45264	184.52	0	3460210.17	45264	21.061
Upvote	836	607.0	836	0.153	0	8522256.0	84591	0.295
Downvote	21608	9.0	21608	0.151	0	173755.0	84649	0.781
Favoritevote	227	89.0	434	19.388	723	268607.67	227	26.309
Relatedlink	97	23.0	13518	4.993	28175	14844.26	12496	8.571
Toutes relations	0	4887.0	45264	41.967	/	/	28175	112.424

TABLE 4.3: Résultats d'analyse de centralité

Après ces analyses et selon les résultats montrés dans le tableau 4.3, nous constatons que :

4.5.1.1 Selon le Degré de centralité

L'utilisateur qui a reçu le plus de commentaire c'est le «836 » avec un score de 870, ainsi que pour les réponses avec un score de 283, et des votes positifs avec un score de 607, alors que le plus grand nombre des interactions selon toutes les relations par un score de 4885.

L'utilisateur « 21608 » a le plus grand nombre des votes négatifs avec un score de 9 alors que l'utilisateur « 97 » a le meilleure score pour les votes favoris 23.

Ce type d'analyse permet de connaitre les utilisateurs qui ont plus d'interaction avec les autres ou le score est le degré de centralité qui peut être utilisé et interprété selon le sens de la relation par exemple dans le cas de la relation « Upvote » le meilleur celui qui a le plus grand score par contre « Downvote » le meilleur c'est l'utilisateur avec un score minimal.

4.5.1.2 Selon le Page Rank

L'utilisateur qui a reçu le plus de commentaires, et ses commentaires ont été commentés à leurs tours par d'autres et ainsi de suite, c'est le «45264 » avec un score de 31.997 et la même chose pour les réponses par un score de 184.52, et avec toutes relations par un score de 41.967 malgré que l'analyse selon toutes relations ne prenne pas en compte les poids des relations.

Alors que pour les relations Upvote, Downvote et Favoritevote, et Relatedlink, les utilisateurs « 836 », « 21608 », « 434 », et « 13518 » ont été les meilleurs par des score de 0.153, 0.151, 19.388, et 4.993.

C'est le classement en fonction de la fréquence de citation (et des liens entrants) pour connaitre les utilisateurs qui ont plus d'influence selon la relation où le score est le degré d'importance de l'utilisateur.

4.5.1.3 Selon la Centralité entre les deux (Betweeness)

Le nœud « 381 » a le degré maximum de 3742655.17 et est le plus susceptible d'être commenté, alors que l'utilisateur « 0 » c'est le nœud central avec un score de 3460210.17, 8522256.0, et 173755.0 successivement pour les réponses, les votes positifs et les votes négatifs et la même chose pour les utilisateurs 723 et 28175 avec les scores 268607.67 et 14844.26 selon les votes favoris et les links, en raison du fait qu'il y a plus de chemins les plus courts plus susceptibles de passer par ces nœuds centraux.

Ce classement définit les utilisateurs importants pour la diffusion d'information (intermédiaires ou centraux).

4.5.1.4 Selon la centralité de vecteur propre (Eingenvector centrality)

L'utilisateur « 28175 » avec le score le plus élevé (55.984) a commenté ou il a reçu des commentaires d'autres utilisateurs qui ont des score aussi plus élevés, i.e. connectés à d'autres nœuds qui ont des scores élevés, même chose pour les autres utilisateurs «45264», « 227 » et « 12496 » par rapport a les relations : « Answer », «Upvote », «Downvote », «Favoritrvote », et « Relatedlink ». Par contre, les utilisateurs «84591 », « 84649 » avec leurs scores 0.295 et 0.781, ils sont connectés à des utilisateurs à faible score.
La centralité de vecteur propre mesure l'influence transitive ou la connectivité des nœuds.

Sur la base de l'analyse de centralité nous pouvons déduire plusieurs résultats utiles, parmi ceux-ci, nous citons :

- Identification des experts par :
  - l'analyse **« Page Rank »** ou **« Degré de centralité»** faite pour les utilisateurs selon les relations « **Favoritevote »**, **« Upvote »** et **« Downvote»** peut aider à améliorer la qualité des services proposés.
  - Identifier les nœuds centraux et intermédiaires permet une diffusion d'informations de qualité.
- le Page Rank et le degrée de centralisation d'utilisateurs les plus élevés selon les relations **« Comment »** et **« Answer »**, identifier les acteurs importants ou plus actifs qui peut aider à rendre le site plus actif par les notifications ciblées.
- Diffusion rapide et globale d'information.
- Elle donne une représentation des nœuds à forte intermédiarité qui peuvent bloquer l'information.

4.5.2 Détection des Communautés

Dans ce cas, il s'agit d'identifier les communautés et les groupes des nœuds, Le tableau 4.4 représente les paramètres dont les valeurs sont inchangées. Tandis que le tableau 4.5 représente les résultats des analyses selon la relation «LieeA » et le tableau 4.6 les résultats selon les relations « Comment » et « Answer » (les relations sont choisies aléatoirement).

Paramètre	Valeur
Algorithme	Louvain
	Composants fortement connectés
	Propagation d'étiquettes
Nœud	Users
	Tags

Table 4.4: Tableau des paramètres de Détection des Communautés

Les résultats obtenus sont des listes des candidats et leurs groupes, nous prenons la communauté ayant la plus grande taille avec quelques exemples des nœuds de la communauté.

Algorithme	Communauté	Taille de communauté	Exemple des tags
Louvain	90	150	" definitions" , " tools", "processing", "apache-hadoop", "r", "data-cleaning",....
Composants fortement connectés	0	1	" definitions"
	15	1	" tools"
	16	1	"processing"
	17	1	"apache-hadoop"
	18	1	"r"
	19	1	"data-cleaning"
Propagation d'étiquettes	50109	479	" definitions" , " tools", "processing", "apache-hadoop", "r", "data-cleaning",....

TABLE 4.5: Résultats d'analyse de détection des communautés selon les tags

Algorithme	Relation	Communauté	Taille de communauté	Exemple des users
Louvain	Comment	3973	708	84712, 84908, 77317, 77631, 18699,18840, 18852, 19113, 1927, 19220,....
	Answer	7448	612	77945, 18869, 18888, 18984, 18992, 19040, 19095, 42495, 42578, 42895, 42985...
Composants fortement connectés	Comment	32521	1476	84626, 84712, 84717, 84756, 84762, 84872, 84891, 85110, 85143, 85147, ...
	Answer	32521	124	84626, 84989, 77900, 18843, 19161, 42926, 11, 26, 84, 97, ...
Propagation d'étiquettes	Comment	4316	238	18951, 19040, 19100, 63044, 42299, 42335, 42384, 42578, 43040,,..
	Answer	4316	241	77945, 18869, 18888, 18984, 18992, 19040, 19095, 42495, 42895, 42985, ...

TABLE 4.6: Résultats d'analyse de détection des communautés selon les utilisateurs

Dans l'ensemble des analyses du tableau 4.5, nous pouvons voir que les communautés diffèrent d'un algorithme à l'autre.

4.5.2.1 Selon Louvain

Les nœuds : " definitions", " tools", "processing", "apache-hadoop", "data-cleaning",... etc appartiennent à une seule communauté avec l'identifient est « 90 » et une taille 150 nœuds. Ces nœuds ayant de nombreuses connexions entre eux mais ayant un très faible nombre de connexions vers les nœuds d'autres communautés.

4.5.2.2 Selon Composants fortement connectés

Chacun des nœuds : " definitions", " tools", "processing", "apache-hadoop", "r", "data-cleaning", appartient à une communauté différente : « 0 », « 15 », « 16 », « 17 », « 18 », « 19 » successivement avec une taille d'un seul nœud.
Dans ce type d'analyse chaque nœud de la communauté doit être connecté à tous les autres nœuds de la même communauté.

4.5.2.3 Selon Propagation d'étiquettes

Les mêmes nœuds des exemples précédents appartiennent cette fois à une seule communauté «50109 » d'une taille de 479 nœuds (avec d'autres nœuds différents).
Dans cette analyse les nœuds : " definitions", " tools", "processing", "apache-hadoop", "r", "data-cleaning" appartiennent à la communauté dont l'identifient est l'étiquette du nœud voisin qui a le plus grands nombre des nœuds connectés.

4.5.2.4 Selon les relations

En ce qui concerne les analyses qui sont réalisées du tableau 4.6, nous concluons que les nœuds changent leurs communauté selon la relation analysée aussi.
L'utilisateur « 84626 » est fortement connecté selon la relation « Comment » avec les utilisateurs « 84712 », « 84717 », « 84756 », « 84762 », « 84872 »,... etc, il commente leurs publications, et eux aussi commentes ses publications.
Mais selon la relation « Answer » l'utilisateur « 84626 » appartient à la même communauté (la même étiquette « 32521 ») mais avec des membres différents : « 84989 », «77900 », « 18843 », « 19161 »,... etc
La détection des communautés permet :

- d'améliorer la compréhension globale de ces réseaux, de connaitre les groupements des nœuds, l'aspect d'appartenance pour chaque nœud.
- de visualiser les individus bénéficiant de la plus forte centralité des services proposés.
- d'apporter des pistes de réflexion renouvelées sur la place des leaders d'opinion, l'interactivité ou l'interdépendance entre membres.
- les politiques coopératives du site peuvent être améliorées grâce à la détection de véritables communautés actives, et pouvoir distinguer au sein de ces communautés des zones de productions.

4.5.3 Similarité

Pour ce type d'analyse nous appliquons l'algorithme de Similarité des nœuds sur les utilisateurs « Users » selon la relation « Comment ». Dans le tableau 4.7 nous présentons des échantillons des résultats obtenus.

userId1	userName1	userId2	userName2	Similarité
84682	"MrLeeh"	50406	"thanatoz"	0.055
84682	"MrLeeh"	71219	"aranglol"	0.037
84682	"MrLeeh"	65131	"Leevo"	0.017
84682	"MrLeeh"	64377	"Erwan"	0.008
84830	"guru_007"	86148	"Christina"	1.0
84830	"guru_007"	78284	"Hello Mellow"	1.0
84830	"guru_007"	70031	"Eswar"	0.333
84830	"guru_007"	58433	"user10296606"	0.125
84830	"guru_007"	29781	"aivanov"	0.076
84830	"guru_007"	84626	"Yohanes Alfredo"	0.04
84830	"guru_007"	85398	"Piotr Rarus - Reinstate Monica"	0.022
84830	"guru_007"	27432	"S van Balen"	0.022
84830	"guru_007"	0	"anonymous"	0.012
84830	"guru_007"	86148	"Christina"	1.0
84830	"guru_007"	78284	"Hello Mellow"	1.0

Table 4.7: Résultats d'analyse de similarité

D'après les résultats montrés ci-dessus (tableau 4.7), nous remarquons les valeurs de similarité de deux utilisateurs « 84682 » et « 84830 » avec d'autres utilisateurs qui sont connectés à eux (les ayant commenté ou étant commentés par eux). Notons que le degré de similarité entre chaque deux utilisateur est appartient à l'intervalle [0, 1].
Un degré similarité de 0.008 entre « 84682 » et « 64377 » désigne une faible ressemblance, un nombre bas de voisins communs (même interaction), par contre la similarité entre « 84830 » et « 86148 » avec une valeur de 1.0 est idéale et complète, elle désigne qu'ils ont les mêmes nœuds auxquels ils sont connectés (les même voisins communs).
L'analyse de similarité des nœuds que nous venons de présenter nous permet de :

- déterminer la similitude entre les membres de site.
- faire des requêtes de recommandation basées sur les pratiques des membres similaires, par exemple : des emails ciblés.
- l'étude des pratiques et usages des utilisateurs du site, et analyse des rites et usages afin de bien les exploiter.

4.5.4 Prédiction des liens

Pour ce type de fouille, nous étudions des liens et connexions déjà existant pour prédire des interactions possibles.Nous allons utiliser les paramètres présentés dans le tableau 4.8. Les résultats seront présentés dans le tableau 4.9.

Paramètre	Valeur
Algorithme	Adamic Adar
	Voisins communs
Nœud	Users

Table 4.8: Tableau des paramètres d'analyse de prédiction des liens

Algorithme	Relation	userId1	userId2	Score
Adamic Adar	**Comment**	84872	87355	0. 655
			84626	0.468
			84580	0.0
	Upvote	84626	84591	0.120
			84617	0.120
	Answer	3150	88704	0.0
Voisins communs	**Toutes les relations**	88709	0	3.0
			85027	2.0
			84580	0.0
		84626	0	23.0
			80580	9.0

Table 4.9: Résultats d'analyse de prédiction des liens

Les résultats obtenus sont des listes d'utilisateurs classés en fonction du degré de probabilité que l'utilisateur interagisse avec d'autres utilisateurs à l'avenir. Dans nos résultats (tableau 4.9) nous prenons juste quelques valeurs.

4.5.4.1 Selon Adamic Adar

D'après le tableau 4.9 l'utilisateur « 84872 » peut être commenté par l'utilisateurs «87355» et « 84626 » dans le futur par un pourcentage de 0.655 et 0.468, par contre il n'a aucune chance pour être commenté par l'utilisateur « 84580 ». Ainsi que la possibilité que l'utilisateur « 84626 » obtienne un vote positif de l'utilisateurs «84591 » et «84617» est estimée à 0.120. En même temps l'utilisateur «88704» ne va répondre sur aucune question de l'utilisateur « 88718 ».
Le score Adamic Adar est la proximité de ces deux nœuds. Une valeur de 0 indique que deux nœuds ne sont pas proches, tandis que des valeurs plus élevées indiquent que les nœuds sont plus proches par conséquence la possibilité qu'il y ait des liens entre eux à l'avenir.

4.5.4.2 Selon Voisins communs

La proximité selon toutes les relations entre les nœuds : « 88709 » et « 0 » est estimée par la valeur 3.0, entre «88709» et « 85027 » est de 2.0, ainsi qu'entre «88709» et « 84580 » il n'y a aucune proximité. Alors qu'entre « 84626 » et « 0 » la proximité est de valeur 23.0, et « 84626 » et « 80580 » et 9.0. Cette valeur est le nombre des nœuds voisins voisins en commun.
Sur la base de cette valeur en prédit la possibilité d'être connecté au future, deux étrangers qui ont un ami en commun sont plus susceptibles d'être présentés que ceux qui n'ont pas d'amis en commun.

La prédiction des liens soit par l'analyse des liens existant ou par l'utilisation des voisins en commun permet de :

- Prédire les comportements d'utilisateurs du site et améliorer les services proposés par conséquence.
- Reconstituer les pratiques et les relations entre membres
- Visualiser la confiance et les liens réciproques.

- Construction des systèmes de simulations.

4.6 Conclusion

Dans ce chapitre nous avons commencé par un bref aperçu de l'environnement de travail et les outils utilisés pour développer notre application, MySQL, Valentina Studio, neo4j, et NetBeans, nous avons présenté la manière de reconstruire la base de données SQL « stackexchange », ainsi la construction de notre base de données orientée graphe sous Neo4j « StackEx ». Ensuite nous avons présenté l'application développée qui vise à appliquer les algorithmes de graphe de fouille de données de la bibliothèque Neo4j Graphe Data Science (GDS) détaillée dans le chapitre précédent, les composants de l'application (les interfaces, les panneaux d'affichages,... etc) et l'enchainement de ces différentes taches, et enfin nous avons présenté les résultats d'analyses et leur utilité.