

Rapport de Projet Neo4j

Sorbonne Data Analytics

Étudiants : Tom Le Corre, Rishikaran Karunakaran

15 janvier 2026

Table des matières

1	Introduction et Présentation du Projet Neo4j	3
1.1	Contexte Général	3
1.2	Objectifs du Projet	3
1.3	Source de Données : Le Yelp Open Dataset	3
1.4	Modélisation et Architecture du Graphe	4
2	Analyse préliminaire et exploration de la donnée	4
3	Création des nœuds Business et relationships	4
4	Création des nœuds User et relationships FRIENDS	5
5	Création des REVIEW User et relationships REVIEWS et WROTE	6
5.1	Validation du schéma et visualisation d'un chemin complet	7
6	Appliquer les algorithmes GDS pour la recommandation	7
6.1	Approche A : La popularité brute (Standard Cypher)	7
6.2	Approche B : L'influence structurelle (Algorithme GDS PageRank)	8
7	Justification du choix final pour le moteur de recommandation	9
8	Conclusion	10

1 Introduction et Présentation du Projet Neo4j

1.1 Contexte Général

Dans l'ère actuelle du Big Data, la capacité à analyser non seulement les données brutes, mais surtout les relations entre ces données, est devenue un enjeu crucial pour les entreprises. Si les bases de données relationnelles traditionnelles (SQL) ou les fichiers plats excellent dans le stockage de données structurées, elles montrent rapidement leurs limites lorsqu'il s'agit d'explorer des réseaux complexes d'interconnexions, comme les réseaux sociaux ou les systèmes de recommandation.

C'est dans ce cadre que s'inscrit ce projet, réalisé au sein du DU Sorbonne Data Analytics. L'objectif principal est de maîtriser les concepts des bases de données orientées graphe à travers l'outil leader du marché : Neo4j. Contrairement aux bases classiques qui "pensent" en termes de tables et de lignes, Neo4j modélise l'information sous forme de nœuds et de relations, offrant une représentation beaucoup plus fidèle et intuitive de la réalité des interactions humaines et commerciales.

1.2 Objectifs du Projet

Ce projet a pour but de concevoir, implémenter et exploiter une base de données graphe à partir de données brutes. La mission consiste à transformer un ensemble de fichiers disparates en un réseau d'informations cohérent capable de répondre à des questions métier complexes, telles que la recommandation de produits.

Les objectifs techniques spécifiques sont les suivants :

- **Modélisation de données** : Passer d'un schéma de données plat (fichiers JSON) à un modèle de graphe optimisé.
- Utiliser le langage Cypher et la librairie APOC pour nettoyer et importer des volumes conséquents de données.
- **Analyse et Recommandation** : Exploiter la structure du graphe pour identifier des motifs comportementaux et suggérer des recommandations pertinentes (Filtrage Collaboratif).

1.3 Source de Données : Le Yelp Open Dataset

Le projet s'appuie sur le célèbre Yelp Open Dataset, une référence dans le domaine de l'analyse de données. Ces données, fournies initialement au format JSON, contiennent une richesse d'informations sur l'activité économique locale et les avis des consommateurs.

Pour les besoins de cette étude, nous nous sommes concentrés sur trois entités majeures :

- **Business (Commerces)** : Les établissements (restaurants, services, etc.) caractérisés par leur nom et leurs catégories.
- **Users (Utilisateurs)** : Les membres de la communauté Yelp, qui tissent entre eux un réseau d'amitié.
- **Reviews (Avis)** : Le cœur du système, représentant l'interaction textuelle et la notation (étoiles) laissée par un utilisateur sur un commerce.

1.4 Modélisation et Architecture du Graphe

La force d'un projet Neo4j réside dans la pertinence de son modèle de données. Nous avons structuré notre base autour de quatre types de nœuds interconnectés :

- (User) : Représente l'individu.
- (Review) : Représente l'avis déposé.
- (Business) : Représente le lieu visité.
- (Category) : Permet de classer les commerces (ex : "Restaurant", "Italien").

Ces nœuds sont liés par des relations sémantiques fortes qui permettent de naviguer dans le graphe :

- Un utilisateur ÉCRIT (:WROTE) un avis.
- Un avis CONCERNE (:REVIEWS) un commerce.
- Un commerce EST DANS (:IN_CATEGORY) une catégorie.
- Un utilisateur est AMI (:FRIENDS) avec d'autres utilisateurs.

[cite_start]

2 Analyse préliminaire et exploration de la donnée

Avant d'entamer toute procédure d'importation massive, il était impératif de procéder à une analyse structurelle des fichiers sources fournis au format JSON. Cette étape d'exploration nous a permis de comprendre l'architecture interne des données brutes et d'anticiper les transformations nécessaires pour le modèle de graphe.

```
CALL apoc.load.json('file:///yelp_academic_dataset_business.json')
YIELD value
RETURN value LIMIT 1
```

Listing 1 – Exploration de la structure JSON

En isolant un échantillon unitaire via la commande d'exploration, nous avons pu identifier les propriétés essentielles telles que les identifiants uniques ou les noms, tout en repérant les attributs superflus pour notre analyse, comme les horaires d'ouverture détaillés ou les attributs de stationnement. Plus important encore, cette phase a révélé que certaines données cruciales, notamment les catégories des commerces et les listes d'amis des utilisateurs, étaient stockées sous forme de chaînes de caractères ou de tableaux. Ce constat a directement orienté notre stratégie technique vers l'utilisation de fonctions de découpage (split) et de dépliage de listes (UNWIND) pour garantir une modélisation atomique des nœuds.

3 Création des nœuds Business et relationships

Pour cette première étape structurante, nous avons initié le chargement des données brutes via la procédure `apoc.load.json` ciblant le fichier des commerces. Compte tenu des contraintes de ressources de notre environnement local, nous avons immédiatement appliqué une restriction à 5 000 entrées (`LIMIT 5000`), ce qui nous a permis de travailler sur un échantillon représentatif afin de valider le modèle sans risquer de saturer la mémoire vive.

La création des nœuds « Business » a été sécurisée par l'instruction **MERGE** basée sur l'identifiant unique, garantissant ainsi l'absence de doublons dans la base dès l'insertion. Dans une logique d'optimisation du stockage, nous avons procédé au nettoyage des données brutes grâce à la fonction `apoc.map.clean`. Cette opération nous a permis d'injecter dynamiquement les propriétés du JSON dans le nœud tout en filtrant automatiquement les champs jugés non pertinents pour notre analyse, tels que les horaires d'ouverture ou les attributs complexes.

Enfin, nous avons porté une attention particulière à la classification des commerces. Le champ des catégories étant fourni sous forme d'une chaîne de caractères unique, nous avons utilisé `apoc.text.split` pour le découper, puis la commande **UNWIND** pour traiter chaque catégorie individuellement. Cette transformation a rendu possible la création de nœuds « Category » distincts et l'établissement de la relation sémantique : **IN_CATEGORY**, connectant ainsi proprement chaque commerce à son secteur d'activité.

```
CALL apoc.load.json('file:///yelp_academic_dataset_business.json')
YIELD value
LIMIT 5000
WITH value
MERGE(b:Business{id:value.business_id})
SET b += apoc.map.clean(value, ['attributes','hours','business_id','categories','address','postal_code'], [])
WITH b, apoc.text.split(value.categories, ',\\s*') as categories
UNWIND categories as category
MERGE(c:Category{id:category})
MERGE(b)-[:IN_CATEGORY]->(c)
```

Listing 2 – Import des Business et création des relations Category

4 Création des nœuds User et relationships FRIENDS

L'intégration des utilisateurs et de leur graphe social a représenté un défi technique particulier en raison de la volumétrie importante du fichier source et de la densité des connexions. Pour pallier les risques de saturation mémoire (Java Heap Space) inhérents à notre environnement local, nous avons pris la décision stratégique de restreindre l'importation à un échantillon de 1 000 utilisateurs via la clause **LIMIT 1000**. Cette approche nous a permis de construire un squelette de réseau social fonctionnel sans compromettre la stabilité de la base de données.

La création des nœuds utilisateurs a suivi une logique rigoureuse : nous avons d'abord instancié l'utilisateur principal via une commande **MERGE** sur son identifiant unique. L'étape de nettoyage, réalisée avec `apoc.map.clean`, a ici une double fonction : elle permet d'injecter les propriétés simples de l'utilisateur tout en excluant volontairement le champ « friends ».

En effet, conserver la liste d'amis sous forme de propriété textuelle aurait été contraire à la philosophie du graphe ; il était impératif de transformer cette donnée en relations structurelles. C'est pourquoi nous avons traité le champ des amis en le découpant d'abord avec `apoc.text.split`, puis en utilisant la commande **UNWIND** pour itérer sur chaque identifiant d'ami. Pour chaque relation, nous avons assuré l'intégrité du graphe en utilisant **MERGE** sur le nœud de l'ami (`u1`). Cette méthode garantit que le nœud de l'ami existe,

qu'il fasse partie ou non de notre échantillon initial de 1 000 utilisateurs, avant de tisser le lien d'amitié via la relation : FRIENDS.

```
CALL apoc.load.json('file:///yelp_academic_dataset_user.json')
YIELD value
LIMIT 1000
WITH value
MERGE(u:User{id:value.user_id})
SET u += apoc.map.clean(value, ['friends','user_id'], [])
WITH u, apoc.text.split(value.friends, ',\\s*') as friends
UNWIND friends as friend
MERGE(u1:User{id:friend})
MERGE(u) -[:FRIENDS]->(u1)
```

Listing 3 – Import des Users et relations Friends

5 Création des REVIEW User et relationships REVIEWS et WROTE

La phase d'intégration des avis (« Reviews ») constitue la clé de voûte de notre architecture, car c'est elle qui opère la jonction entre le graphe social des utilisateurs et le graphe économique des commerces. Pour cette étape critique, nous avons traité un volume conséquent de 50 000 avis via la clause LIMIT 50000, un échantillon suffisamment large pour densifier le réseau et faire émerger des motifs comportementaux pertinents pour l'analyse.

La logique d'importation que nous avons développée repose sur une garantie forte d'intégrité référentielle. Pour chaque avis traité, nous avons systématiquement utilisé la commande MERGE sur les nœuds « Business » et « User » correspondants. Cette approche est fondamentale : elle assure que l'avis sera toujours rattaché à une entité existante. Si l'utilisateur ou le commerce faisait déjà partie de notre échantillon initial, le lien est simplement établi ; dans le cas contraire, le nœud est instancié dynamiquement à partir de son identifiant, évitant ainsi toute perte d'information ou la création de relations orphelines.

Une fois ces points d'ancrage validés, nous avons procédé à la création du nœud « Review » lui-même et à l'établissement de la double relationnelle : la relation :WROTE partant de l'utilisateur vers l'avis, et la relation :REVIEWS partant de l'avis vers le commerce. Enfin, nous avons finalisé l'opération par un nettoyage des propriétés via apoc.map.clean, en supprimant les clés étrangères (business_id, user_id) devenues obsolètes puisque l'information est désormais portée par la structure même du graphe.

```
CALL apoc.load.json('file:///yelp_academic_dataset_review.json')
YIELD value
LIMIT 50000
WITH value
MERGE(b:Business{id:value.business_id})
MERGE(u:User{id:value.user_id})
MERGE(r:Review{id:value.review_id})
MERGE(u) -[:WROTE]->(r)
MERGE(r) -[:REVIEWS]->(b)
SET r += apoc.map.clean(value, ['business_id','user_id','review_id'], [])
```

Listing 4 – Création des relations WROTE et REVIEWS

5.1 Validation du schéma et visualisation d'un chemin complet

Afin de certifier la bonne exécution de nos scripts d'importation et l'intégrité relationnelle de notre base de données, nous avons soumis le graphe à une requête de contrôle stricte. L'objectif était d'extraire un chemin complet traversant l'intégralité de notre modèle de données, depuis l'utilisateur jusqu'à la catégorie du commerce visité.

```
MATCH p=(u:User)-[:WROTE]->(r:Review)-[:REVIEWS]->(b:Business)-[:IN_CATEGORY]->(c:Category)
RETURN p LIMIT 1
```

Comme en témoigne la capture d'écran ci-dessous, le résultat obtenu valide parfaitement notre architecture. Nous observons une chaîne ininterrompue où l'utilisateur (nœud vert) est relié par la relation `:WROTE` à un avis (nœud bleu), lui-même connecté via `:REVIEWS` au commerce « Target » (nœud beige), qui est enfin classifié par sa catégorie (nœud jaune). Cette visualisation confirme non seulement que les données ont été correctement nettoyées et ingérées, mais surtout que les liens sémantiques entre les entités hétérogènes (Social, Avis, Business) sont fonctionnels et prêts à être exploités pour l'analyse.

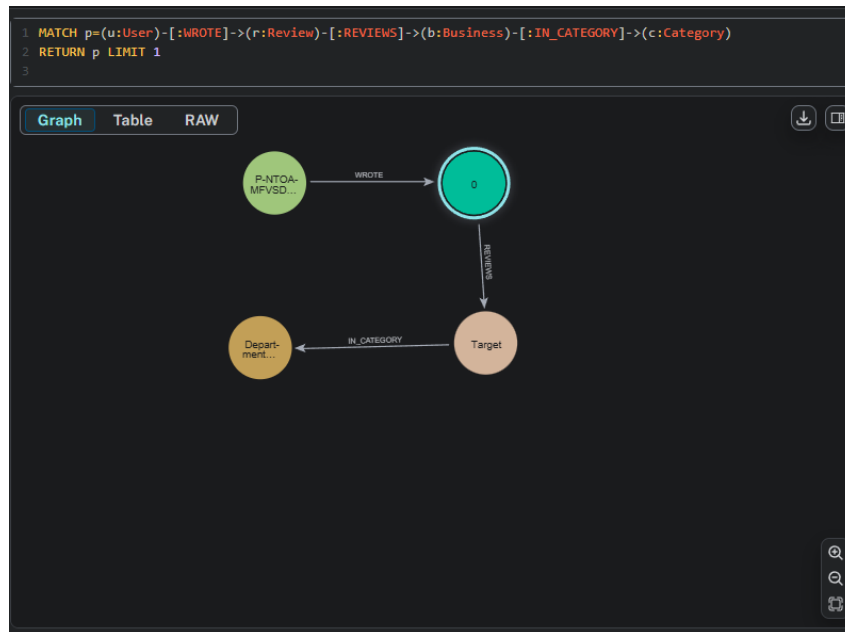


FIGURE 1 – Visualisation d'un chemin complet validant le schéma

6 Appliquer les algorithmes GDS pour la recommandation

6.1 Approche A : La popularité brute (Standard Cypher)

Dans un premier temps, nous avons cherché à identifier les commerces les plus "visibles" du réseau. Pour cela, nous avons utilisé une requête d'agrégation classique calculant la centralité de degré (Degree Centrality). Concrètement, il s'agit de comptabiliser le nombre de relations `:REVIEWS` entrantes pour chaque commerce.

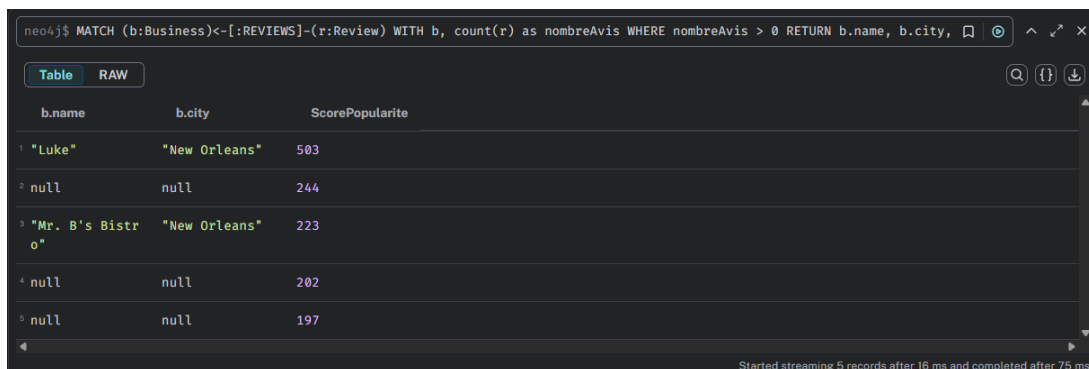
```

MATCH (b:Business)<-[:REVIEWS]-(r:Review)
WITH b, count(r) as nombreAvis
WHERE nombreAvis > 0
RETURN b.name, b.city, nombreAvis as ScorePopularite
ORDER BY ScorePopularite DESC
LIMIT 5

```

Listing 5 – Calcul de popularité simple

Cette méthode présente l'avantage d'être immédiate et ne nécessite aucune installation de plugin. Elle nous fournit un indicateur de volume : quels sont les lieux qui génèrent le plus de bruit ? Cependant, cette métrique reste limitée car elle considère que tous les avis se valent, qu'ils viennent d'un utilisateur novice ou d'un expert de la plateforme.



b.name	b.city	ScorePopularite
"Luke"	"New Orleans"	503
null	null	244
"Mr. B's Bistr o"	"New Orleans"	223
null	null	202
null	null	197

FIGURE 2 – Résultats basés sur le nombre d'avis

6.2 Approche B : L'influence structurelle (Algorithme GDS PageRank)

Afin de pallier les limites de la simple popularité, nous avons activé la librairie Neo4j GDS pour implémenter l'algorithme de PageRank. Contrairement au comptage précédent, le PageRank analyse la structure globale du graphe. Il part du principe qu'un commerce est "important" non pas seulement s'il a beaucoup d'avis, mais s'il est lié à des nœuds eux-mêmes importants dans le réseau.

Pour ce faire, nous avons procédé en trois étapes techniques : la projection d'un graphe virtuel en mémoire (pour isoler la structure topologique), l'exécution de l'algorithme en mode write (pour écrire le score directement sur les nœuds), et enfin l'interrogation des résultats.

```

// 1. Projection en m moire
CALL gds.graph.project(
  'recoGraph',
  ['User', 'Review', 'Business'],
  ['WROTE', 'REVIEWS']
);

// 2. Calcul et criture du score
CALL gds.pageRank.write(
  'recoGraph',
  {
    writeProperty: 'pagerankScore',

```

```

        maxIterations: 20,
        dampingFactor: 0.85
    }
)
YIELD nodePropertiesWritten, ranIterations;

```


Listing 6 – Projection et calcul PageRank

```

MATCH (b:Business)
WHERE b.pagerankScore IS NOT NULL
RETURN b.name, b.pagerankScore, b.city
ORDER BY b.pagerankScore DESC
LIMIT 5;

```

Listing 7 – Requête de Recommandation finale



The screenshot shows a Neo4j Cypher query interface with the following query: `neo4j$ MATCH (b:Business) WHERE b.pagerankScore IS NOT NULL AND b.name IS NOT NULL RETURN b.name, b.pagerankScore, b.city`. The results are displayed in a table view with columns `b.name`, `b.pagerankScore`, and `b.city`. The table contains 5 rows of data, sorted by `b.pagerankScore` in descending order. The first row is for "Luke" in "New Orleans" with a score of 113.47313839285687. The second row is for "Mr. B's Bistr o" in "New Orleans" with a score of 50.69280624999993. The third row is for "District Donut s Sliders Brew" in "New Orleans" with a score of 41.48284374999996. The fourth row is for "Santa Barbara Shellfish Compa ny" in "Santa Barbara" with a score of 38.203437499999964. The fifth row is for "Prep & Pastry" in "Tucson" with a score of 37.98456249999997. At the bottom, a status bar indicates "Started streaming 5 records after 76 ms and completed after 120 ms".

	b.name	b.pagerankScore	b.city
1	"Luke"	113.47313839285687	"New Orleans"
2	"Mr. B's Bistr o"	50.69280624999993	"New Orleans"
3	"District Donut s Sliders Brew"	41.48284374999996	"New Orleans"
4	"Santa Barbara Shellfish Compa ny"	38.203437499999964	"Santa Barbara"
5	"Prep & Pastry"	37.98456249999997	"Tucson"

FIGURE 3 – Résultats basés sur le score PageRank (Influence)

7 Justification du choix final pour le moteur de recommandation

Au terme de cette étude comparative entre une approche quantitative (nombre d'avis via Cypher) et une approche structurelle (PageRank via GDS), notre arbitrage final s'est porté sans équivoque en faveur de l'utilisation de la librairie Graph Data Science. Ce choix est motivé par la nécessité de fournir à l'utilisateur une recommandation basée sur la pertinence et l'autorité plutôt que sur le simple volume.

L'approche initiale par la « Centralité de Degré » (comptage brut des avis) présente un biais majeur : elle favorise systématiquement les commerces les plus anciens ou les chaînes de restauration massivement fréquentées, au détriment de la qualité réelle. Dans ce modèle, un avis déposé par un utilisateur expert a le même poids qu'un avis laissé par un compte inactif ou potentiellement suspect. Ce système est donc vulnérable aux biais de volume et ne reflète pas la dynamique sociale réelle de la plateforme Yelp.

À l'inverse, l'algorithme PageRank, que nous avons implémenté via GDS, exploite pleinement la topologie du graphe. Il introduit une notion de récursivité fondamentale : un

commerce gagne en score non seulement parce qu'il reçoit des avis, mais surtout parce que ces avis proviennent d'utilisateurs eux-mêmes centraux dans le réseau ou parce qu'il est lié à d'autres entités influentes. En pondérant l'importance des nœuds en fonction de la qualité de leurs connexions, PageRank agit comme un filtre naturel contre le bruit et les comportements artificiels.

En conclusion, le choix de l'algorithme PageRank s'inscrit dans la logique même de l'utilisation d'une base de données orientée graphe comme Neo4j. Il permet de transformer des relations brutes en une métrique d'influence qualifiée, offrant ainsi des recommandations plus fines, plus fiables et capable de faire émerger des établissements qui bénéficient d'une véritable validation communautaire. C'est cette "intelligence du lien" qui constitue la valeur ajoutée de notre projet par rapport à une analyse statistique traditionnelle.

8 Conclusion

Bilan Technique et Méthodologique

La transition d'un format de stockage traditionnel (fichiers JSON plats) vers une modélisation en graphe sous Neo4j a mis en lumière la puissance du langage Cypher pour représenter des réalités complexes. La structuration de notre base autour du triptyque (*User*) – (*Review*) – (*Business*) s'est révélée particulièrement pertinente pour capturer la nature interconnectée des données Yelp. Ce travail a également été une épreuve technique formatrice. La gestion de la volumétrie des données nous a contraints à adopter des stratégies d'optimisation rigoureuses : utilisation de la librairie APOC pour le nettoyage, gestion des transactions par lots, et échantillonnage intelligent pour contourner les limitations de mémoire. Ces contraintes nous ont poussés à ne pas simplement "exécuter du code", mais à comprendre l'architecture sous-jacente de la base de données.

Apport Analytique

Au-delà de l'ingénierie des données, nous avons démontré la supériorité de l'approche graphe pour les problématiques de recommandation. Là où une base SQL aurait nécessité des jointures multiples et coûteuses pour relier des amis d'amis à des commerces, Neo4j nous a permis de naviguer naturellement dans le réseau. La comparaison finale entre une métrique de popularité simple (volume d'avis) et l'algorithme de PageRank (via la librairie GDS) a confirmé que l'analyse de la structure du réseau permet des recommandations plus qualitatives et robustes.

Perspectives

Si ce projet a permis de valider un prototype fonctionnel sur un échantillon significatif, plusieurs pistes d'évolution sont envisageables. Une infrastructure cloud permettrait de lever les verrous de mémoire pour traiter l'intégralité du dataset. De plus, l'enrichissement du modèle par de nouvelles relations (par exemple, la similarité de texte dans les avis via du NLP) permettrait d'affiner encore davantage la précision du moteur de recommandation.

En définitive, ce projet confirme que la technologie de graphe n'est pas seulement un outil de stockage, mais un véritable levier d'analyse pour comprendre et valoriser les interactions sociales et économiques.