

## Projet D'analyse Prédictive

# Une méthode pour transformer les ensembles de données en graphes de connaissances

### Master Data Science & Big data

---

#### Réalisé Par :

Ilyas Bazzi

Abdelhakim EL Ghayoubi

Oussama EL Hafidi

#### Examiné par :

Pr. Mohamed Ghezouani

---

Année Universitaire : 2025/2024  
Présenté le : XX/XX/2025

# Résumé

*La transformation de données en graphes de connaissances (KGs) représente une avancée majeure pour structurer et organiser des informations complexes, en reliant des entités à travers des relations sémantiques riches. Ce rapport explore différentes approches et méthodologies pour construire des KGs à partir d'informations textuelles non structurées, avec un accent particulier sur l'utilisation de modèles de langage de grande taille (LLMs) tels que Llama 3.1. À travers une analyse approfondie de l'état de l'art, une méthodologie claire est définie pour surmonter les défis liés à la conversion d'informations textuelles en graphes exploitables. Notre proposition met l'accent sur l'intégration de Neo4j pour la gestion des données en graphe et l'utilisation des LLMs pour l'extraction sémantique. Ce travail contribue à enrichir les pratiques actuelles en démontrant comment les KGs peuvent révolutionner l'exploitation des données*

# Sommaire

Introduction Générale.....	5
Chapitre 1 : Étude & Analyse du Projet.....	6
Introduction .....	6
1. Généralités Sur Graphe De Connaissances .....	7
1.1 Définition : .....	7
1.2 L'histoire :.....	7
1.3 Objectifs : .....	8
1.4 Cas d'utilisations : .....	9
1.5 Comment fonctionne un graphe de connaissances ?.....	9
1.6 Modélisation des données et conception d'ontologies .....	12
1.7 Composants d'un Graphe de Connaissances .....	12
1.8 Avantages des Graphes de Connaissances .....	13
1.9 Inconvénients des Graphes de Connaissances .....	14
2. État de l'Art.....	14
2.1 From Unstructured Text to Causal Knowledge Graphs (SpEAR) .....	14
2.2 From Text to Knowledge with Graphs: Modelling, Querying, and Exploiting Textual Content.....	15
2.3 iText2KG : Incremental Knowledge Graphs Construction Using Large Language Models Construction Incrémentielle de Graphes de Connaissances .....	15
2.4 Knowledge Graph Generation from Text (Grapher) .....	15
2.5 Fusion d'Informations Multi-Facettes via Graphes de Connaissances .....	16
2.6 LLM-based SPARQL Query Generation from Natural Language over Federated Knowledge Graphs .....	16
2.7 Tableau comparative .....	17
3. Architecture générale .....	17
3.1 Solution Proposée.....	18
4. Étude des solutions .....	18
4.1 Google Collab .....	18
4.2 Python.....	19
4.3 Ollama.....	19
4.4 Langchain .....	20
4.5 LLaMA 3.1 8b model .....	20
4.6 mxbai-embed-large model .....	21
4.7 Neo4j.....	21

Conclusion .....	22
Chapitre 2 : Etapes, Implémentation, Résultat.....	23
Introduction .....	23
1. Les Données .....	24
1.1 Résumé .....	24
1.2 Points clés .....	25
1.3 Questions posées.....	25
2. Les Etapes d'implémentation.....	26
2.1 Installation des Packages Requis .....	26
2.2 Configuration de l'API Ollama et Téléchargement du Modèle LLama .....	26
2.3 Définition des Classes pour les Nœuds, Relations et Documents .....	27
2.4 Traitement et Sérialisation des Documents en Graphes de Connaissances .....	27
2.5 Chargement et Transformation des Données JSON en GraphDocuments .....	28
2.6 Configuration de la Connexion Neo4j et Ajout des GraphDocuments .....	29
2.7 Création d'un Index en Texte Intégral dans Neo4j.....	31
2.8 Extraction d'Entités et Recherche dans Neo4j.....	31
2.9 Définition du Full Retriever et de la Chaîne de Q&A.....	32
2.10 Exécution de la Q&A.....	33
Conclusion .....	34
Conclusion et perspectives.....	35

# Liste des Figures :

Figure 1 : Graphe de connaissances en arrière-plan.....	7
Figure 2 : Flux de travail des données à la sagesse .....	8
Figure 3 : Des données brutes au graphique de connaissance.....	10
Figure 4 : Entité/Ontologie (nœud et relation) .....	11
Figure 5 : Modélisation des données et conception d'ontologies .....	12
Figure 6 : Tableau comparatif des travaux réalisés.....	17
Figure 7 : Architecture générale du projet.....	17
Figure 8 : Carte mentale des données non structurées .....	24
Figure 9 : Ontologies Extraction à partir de morceaux de données .....	27
Figure 10 : Diagramme de graphe de connaissances.....	29
Figure 11 : Interface Noe4j.....	29
Figure 12 : Entités extraites du document graphique et interrogation de toutes les entités .	30
Figure 13 : Exemple de contenu d'entités.....	30
Figure 14 : Création d'un Index en Texte Intégral dans Neo4j.....	31
Figure 15 : Extraction d'Entités et Recherche dans Neo4j .....	32
Figure 16 : Définition du Full Retriever et de la Chaîne de Q&A.....	33
Figure 17 : Exemple d'exécution des questions-réponses .....	34

# Introduction Générale

*Le présent rapport documente les travaux réalisés dans le cadre du projet d'Analyse Prédictive au sein de la Faculté des Sciences Ben M'sick, dans le cadre du Master en Data Science & Big Data. Ce projet vise à développer une méthode innovante permettant de transformer des informations non structurées, telles que des textes ou des documents, en graphes de connaissances. L'objectif principal est de faciliter l'exploration, l'analyse et l'exploitation de ces données complexes.*

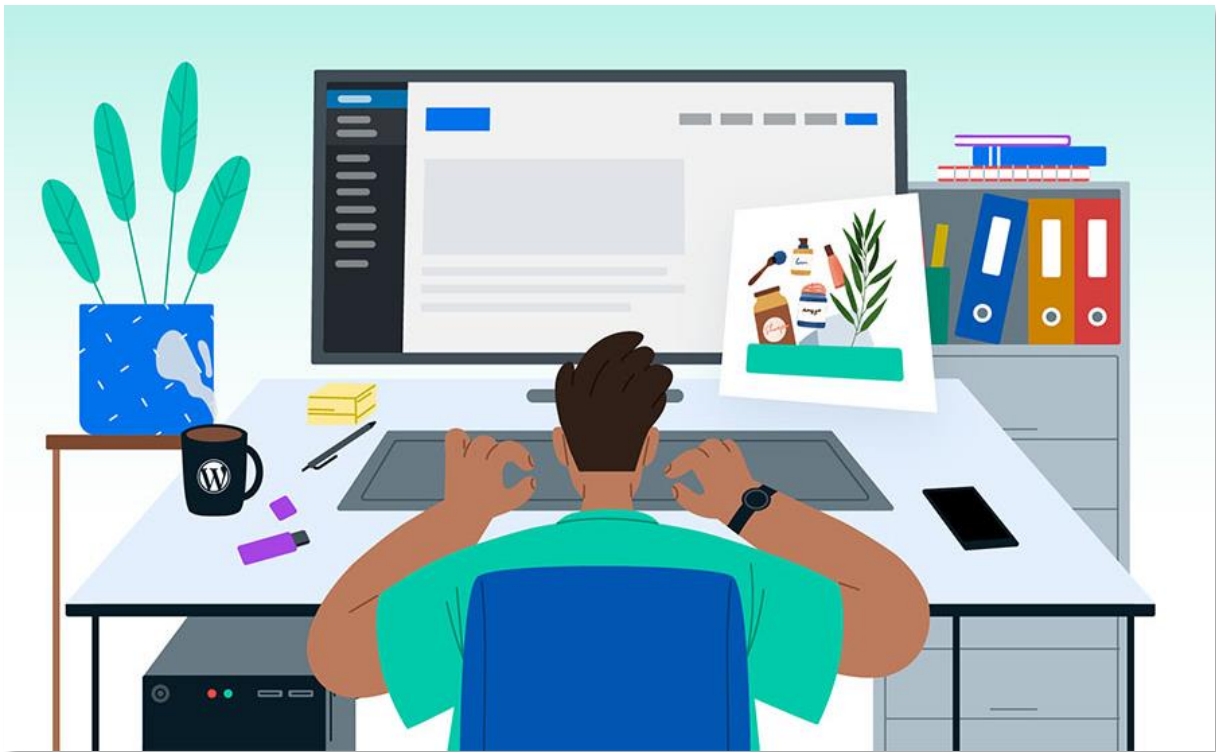
*Dans un contexte où les données textuelles prolifèrent et jouent un rôle crucial dans divers domaines, la capacité d'extraire et de structurer ces informations de manière significative est devenue indispensable. Les graphes de connaissances offrent une représentation intuitive et puissante des données, en les structurant sous forme de réseaux d'entités et de relations, ce qui permet d'interroger et d'exploiter l'information de manière efficace.*

*Ce projet explore une approche innovante basée sur la Récupération Augmentée par la Génération (RAG) pour répondre aux questions complexes. Les étapes du projet se décomposent comme suit :*

- *Transformation des informations non structurées en un graphe de connaissances, stocké dans une base de données dédiée (Neo4j).*
- *Interrogation du graphe de connaissances à l'aide du modèle de langage avancé LLaMA 3.1 pour récupérer les données pertinentes.*
- *Génération de réponses concises et informatives grâce à LLaMA 3.1, en combinant les informations extraites du graphe avec le contexte fourni par la question.*

*En combinant des techniques de traitement du langage naturel, de modélisation des connaissances et de génération de réponses, ce projet contribue à exploiter pleinement le potentiel des données textuelles, tout en ouvrant la voie à des applications pratiques dans divers secteurs.*

# Chapitre 1 : Étude & Analyse du Projet



## Introduction



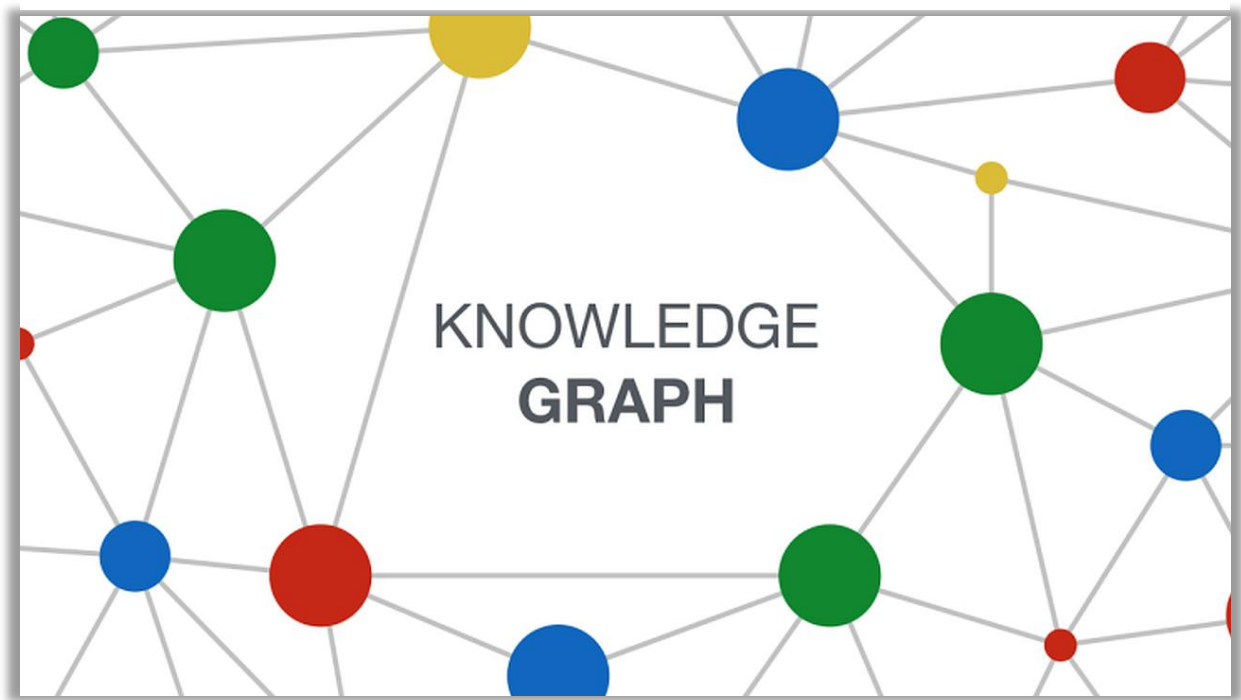
*Dans ce premier chapitre, nous abordons les fondements théoriques et pratiques nécessaires à la compréhension du projet. Nous commençons par introduire les concepts clés liés aux graphes de connaissances, en détaillant leur définition, leur histoire, leurs objectifs, ainsi que leurs applications pratiques. Nous explorons ensuite les principes de modélisation des données et les composants essentiels qui composent un graphe de connaissances. Par ailleurs, une analyse approfondie de l'état de l'art est réalisée à travers l'examen de travaux récents dans ce domaine, afin d'identifier les approches existantes et d'évaluer leurs points forts et leurs limites. Enfin, nous présentons une vue d'ensemble de l'architecture générale du système proposé, accompagnée d'une étude comparative des outils et technologies envisagés pour sa mise en œuvre.*



# 1. Généralités Sur Graphe De Connaissances

Dans un monde où la quantité d'informations disponibles croît de manière exponentielle, les organisations et les chercheurs doivent relever un défi majeur : exploiter ces données de manière structurée et efficace. La majorité des informations disponibles se présentent sous forme de texte non structuré, dispersé à travers des articles, des rapports, des bases de données textuelles, ou encore des publications en ligne. Cette nature non structurée rend l'analyse, la recherche d'insights, et la prise de décision particulièrement difficiles.

Figure 1 : Graphe de connaissances en arrière-plan



## 1.1 Définition :

Un graphe de connaissances, également connu sous le nom de réseau sémantique, représente un réseau d'entités du monde réel - telles que des objets, des événements, des situations ou des concepts - et illustre la relation qui existe entre elles. Ces informations sont généralement stockées dans une base de données de graphes et visualisées sous la forme d'une structure de graphe, d'où le terme de « graphe » de connaissances [1].

## 1.2 L'histoire :

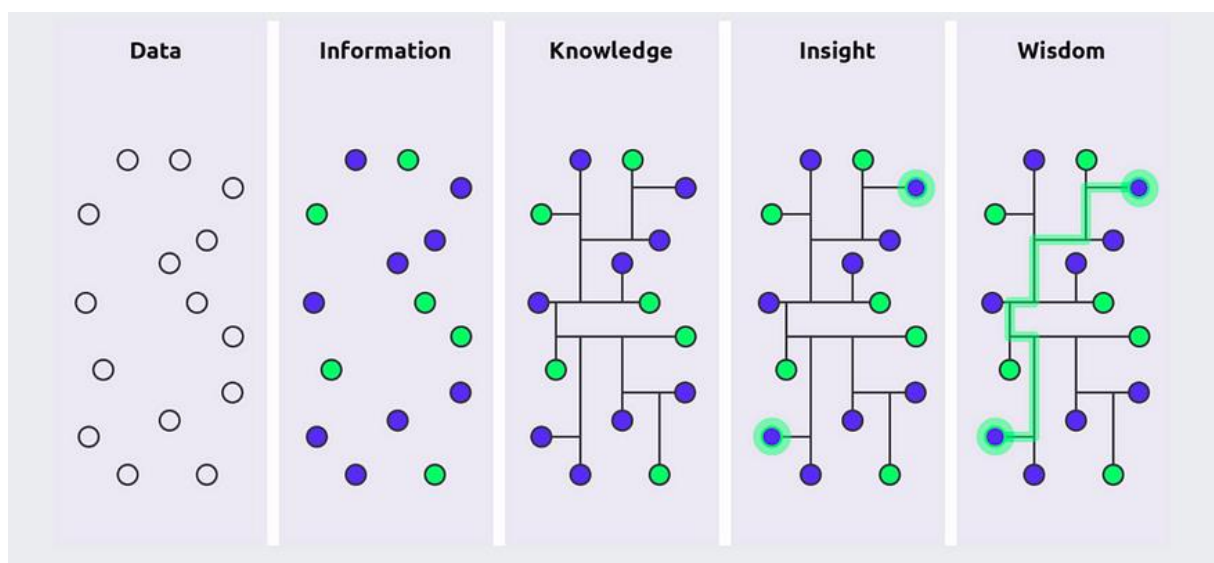
Les graphes de connaissances (KG) sont des structures sémantiques permettant d'organiser et de relier les données sous forme de réseaux de relations compréhensibles par les machines. Leur popularité a explosé avec le lancement du Knowledge Graph de Google en 2012, mais leurs bases reposent sur des standards comme RDF (Resource Description Framework) et SPARQL, utilisés pour modéliser des ontologies et exécuter des requêtes complexes. En



France, des institutions comme l'INRIA, le CNRS et des universités ont contribué à leur développement, notamment dans les domaines du web sémantique et du traitement automatique du langage naturel (TALN). Des projets comme OpenKG ou des initiatives dans la santé et l'e-commerce démontrent leur potentiel pour intégrer et exploiter des données hétérogènes. Aujourd'hui, les KG jouent un rôle clé dans les domaines de l'intelligence artificielle explicable, des systèmes de recommandation, et de l'automatisation des processus décisionnels [2].

### 1.3 Objectifs :

Figure 2 : Flux de travail des données à la sagesse



Les graphes de connaissances (KG) ont pour objectif principal d'organiser, structurer et relier des données provenant de sources variées afin de les rendre compréhensibles pour les humains et exploitables par les machines. Ils permettent de :

- **Représenter les connaissances :** Modéliser des informations complexes sous forme de relations sémantiques entre entités.
- **Améliorer la recherche et les requêtes :** Fournir des réponses précises et contextuelles dans des moteurs de recherche ou systèmes de questions-réponses.
- **Faciliter l'intégration des données :** Relier des données hétérogènes provenant de différentes bases pour créer une vue unifiée.
- **Soutenir l'intelligence artificielle :** Renforcer les systèmes de recommandation, le raisonnement automatique et les modèles d'apprentissage machine.
- **Optimiser la prise de décision :** Aider à l'analyse et à l'interprétation des données pour des applications comme la santé, la finance, et la gestion des connaissances.

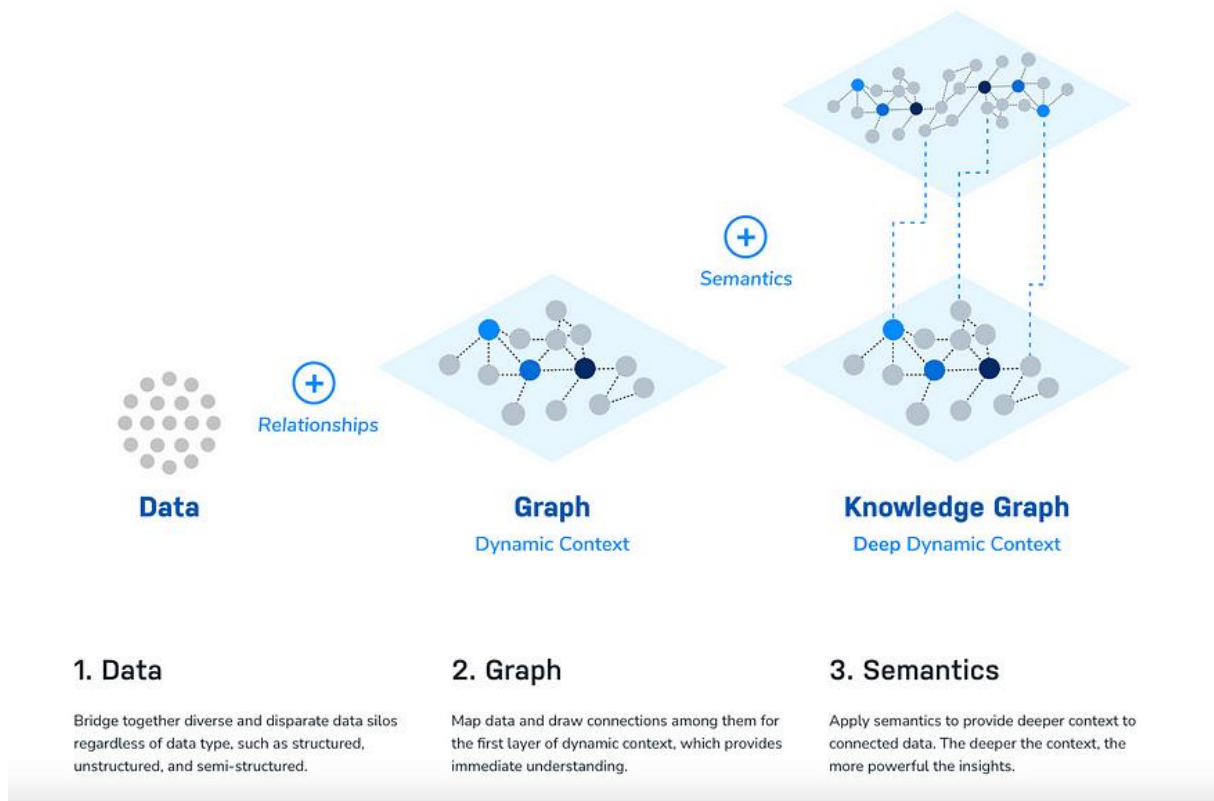
## 1.4 Cas d'utilisations :

Les graphes de connaissances (KGs) jouent un rôle clé dans de nombreux domaines en raison de leur capacité à structurer et relier des données complexes. Ils permettent de transformer des données brutes en réseaux d'information interconnectés, facilitant ainsi l'analyse, la compréhension et la prise de décision. Leur adoption croissante dans divers secteurs met en évidence leur polyvalence et leur potentiel à résoudre des défis spécifiques liés à la gestion et à l'exploitation des connaissances [1]

- **Recherche scientifique** : Ils aident à cartographier les relations entre chercheurs, découvertes et publications, soutenant l'innovation et permettant de connecter des domaines scientifiques disparates.
- **Systèmes de recommandation** : En comprenant les relations entre les préférences des utilisateurs, les produits et les comportements, les KGs améliorent la personnalisation et la pertinence des recommandations.
- **Gestion des connaissances industrielles** : Les KGs structurent les informations sur les processus industriels, les équipements et les flux de travail pour optimiser la maintenance prédictive et la prise de décision.
- **Commerce de détail** : Les graphes de connaissances sont utilisés pour des stratégies de vente incitative et croisée, en recommandant des produits basés sur le comportement d'achat individuel et les tendances populaires parmi différents groupes démographiques.
- **Divertissement** : Les graphes de connaissances alimentent les moteurs de recommandation basés sur l'intelligence artificielle pour des plateformes comme Netflix ou les réseaux sociaux. En analysant les clics et les comportements en ligne, ces systèmes recommandent aux utilisateurs de nouveaux contenus à lire ou à regarder.
- **Finance** : Les KGs sont employés dans les initiatives de connaissance client (KYC) et de lutte contre le blanchiment d'argent. Ils permettent aux institutions financières de suivre les flux d'argent, d'identifier les clients à risque et de prévenir les crimes financiers.
- **Recherche médicale** : Les graphes de connaissances organisent et catégorisent les relations dans le cadre de la recherche médicale. Ces informations aident les prestataires à valider les diagnostics et à proposer des plans de traitement adaptés aux besoins individuels.

## 1.5 Comment fonctionne un graphe de connaissances ?

Figure 3 : Des données brutes au graphique de connaissance

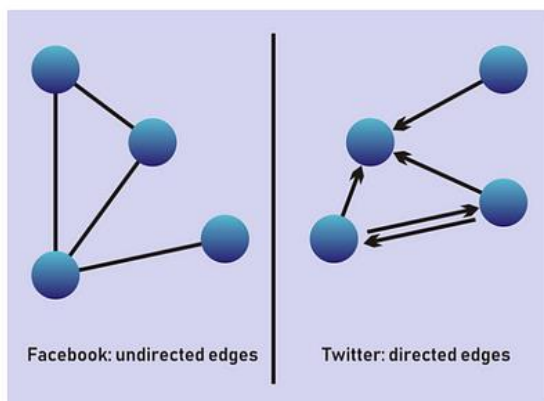


Le sens sémantique ajouté par les graphes de connaissances est écrit de manière formelle afin d'éliminer toute ambiguïté, de le rendre digeste à la fois pour les utilisateurs et les machines, et de permettre au raisonnement automatisé d'envisager un raisonnement déduit [3].

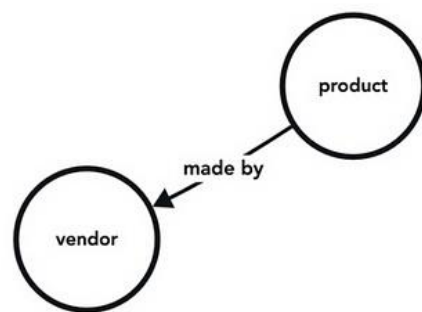
- **Ajout de sens sémantique** : Les Graphes de Connaissances ajoutent un sens sémantique aux données pour éliminer l'ambiguïté et les rendre compréhensibles à la fois pour les utilisateurs et les machines. Ils permettent également un raisonnement automatisé pour des conclusions inférées.
- **Chaque description est complète et partielle** : Les descriptions des entités et relations dans un graphe de connaissances définissent également partiellement des entités liées, créant une structure en forme de réseau. Exemple : décrire un "Chat" comme un mammifère qui chasse des rats définit également partiellement "Rat" et "Mammifère".
- **Ontologie : Un contrat** : Les **sémantiques formelles** définissent la signification des objets à l'aide d'outils logiques, avec l'**ontologie** comme fondation. L'ontologie catégorise les entités et leurs relations, assurant une compréhension partagée entre les développeurs et les utilisateurs. Elle agit comme un contrat qui permet d'avoir un consensus sur la signification des données.
- **Ontologie vs Taxonomie** :
  - Les **taxonomies** définissent des structures hiérarchiques, tandis que les **ontologies** vont plus loin en ajoutant des relations plus riches entre les entités.
  - Une ontologie peut inclure plusieurs taxonomies, fournissant ainsi un contexte plus profond.
- **RDF (Resource Description Framework)** :

- Le **RDF** est un modèle de données pour les données interconnectées, permettant des opérations CRUD sans affecter les données physiques.
- Il permet d'intégrer des données provenant de diverses sources et de les interroger globalement.
- **Les trois modèles de RDF :**
  - **Bases de données** : Permet d'interroger des données comme un seul ensemble global.
  - **Graphes** : Permet d'analyser des structures de réseau.
  - **Bases de connaissances** : Permet d'interpréter le contexte des données et d'inférer de nouveaux faits.
- **Nœuds, Arêtes et Propriétés :**
  - **Nœuds** : Représentent des entités réelles.
  - **Arêtes** : Représentent les relations entre les entités.
  - **Propriétés** : Décritent les caractéristiques des nœuds et des arêtes.
  - Ces composants peuvent représenter des actifs de données, des concepts, des services ou des utilisateurs et des relations telles que la hiérarchie, la localisation et la classification.

**Figure 4 : Entité/Ontologie (nœud et relation)**



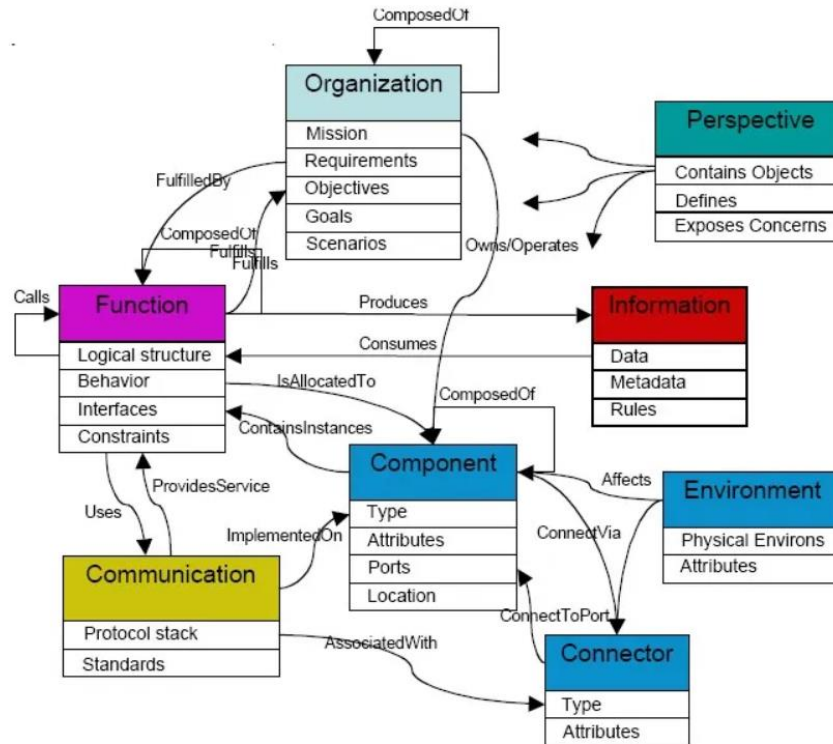
*Example of Nodes & Edges*



*A relationship designed with Nodes & Edges*

## 1.6 Modélisation des données et conception d'ontologies

Figure 5 : Modélisation des données et conception d'ontologies



- **Modèle de données** : Le graphe de connaissances repose sur un modèle de données solide, qui définit les types d'entités, leurs attributs et les relations entre les entités. Ce modèle est souvent appelé **ontologie** lorsqu'il est formalisé.
- **Développement de l'ontologie** : Lors de la conception d'un graphe de connaissances, il est essentiel d'établir une ontologie qui reflète les entités, les concepts et les relations du monde réel dans votre ensemble de données.
  - **Classes** : Elles représentent les différents types d'entités, tels que « Personne », « Organisation » ou « Produit ».
  - **Propriétés** : Attributs ou caractéristiques des entités (par exemple, l'âge, le nom).
  - **Relations** : Définir la manière dont les entités sont liées (par exemple, « travaille pour », « possède »).

## 1.7 Composants d'un Graphe de Connaissances

Un graphe de connaissances est composé de plusieurs éléments clés qui interagissent pour organiser, structurer et relier les données de manière significative. Ces composants permettent d'intégrer des données provenant de diverses sources, de définir des relations complexes et de donner un contexte aux informations [3]:

- **Jeux de données** : Un graphe de connaissances intègre des données provenant de diverses sources. Ces jeux de données peuvent fréquemment changer de structures et de relations avec d'autres actifs de données.
- **Schémas** : Les schémas sont une représentation structurelle ou un cadre du Graphe de Connaissances. Des modèles tels que FIBO, Brick, et d'autres disponibles sur [schema.org](https://schema.org) peuvent servir de bonnes structures de référence pour démarrer.
- **Identités ou Étiquettes** : Les identités définissent et classifient les nœuds dans le Graphe de Connaissances.
- **Métadonnées et Contexte** : Le contexte définit le cadre dans lequel la connaissance existe, soutenu par des métadonnées qui fournissent des informations sur et autour d'un actif de données.

## 1.8 Avantages des Graphes de Connaissances

Les graphes de connaissances offrent plusieurs avantages qui les rendent particulièrement puissants pour organiser et exploiter les données complexes et interconnectées.

- **Amélioration de la Recherche** : Les graphes de connaissances permettent d'améliorer les résultats de recherche en fournissant des réponses plus précises basées sur la compréhension sémantique des requêtes, et non seulement sur des correspondances de mots-clés.
- **Intégration de Données** : Ils facilitent l'intégration de données provenant de sources multiples et hétérogènes, en reliant des informations de manière significative, ce qui permet d'obtenir une vue plus complète et cohérente des données.
- **Raisonnement et Inférence** : Grâce à la capacité d'inférer des relations et des faits non explicitement déclarés, les graphes de connaissances permettent un raisonnement automatisé qui aide à tirer des conclusions et à découvrir des informations cachées.
- **Flexibilité et Évolutivité** : Les graphes de connaissances sont extrêmement flexibles et peuvent évoluer facilement au fur et à mesure que de nouvelles données ou entités sont ajoutées, sans nécessiter une réorganisation complète des données.
- **Amélioration de la Décision** : En organisant les informations de manière logique et en fournissant des connexions entre les différents éléments de données, les graphes de connaissances permettent une meilleure prise de décision basée sur une compréhension complète du contexte et des relations entre les données.
- **Compréhension du Contexte** : Ils aident à donner du sens aux données en ajoutant un contexte aux informations, ce qui améliore l'interprétation et l'analyse des données dans divers domaines (ex. : entreprises, santé, finance).
- **Automatisation des Processus** : Les graphes de connaissances permettent d'automatiser des processus complexes, comme l'assignation de catégories ou la recommandation de produits, en utilisant les relations et les liens définis dans le graph.
- **Amélioration de la Collaboration** : En permettant à différents utilisateurs d'explorer et de comprendre les mêmes données, les graphes de connaissances favorisent une collaboration plus efficace entre équipes, départements et entreprises.

## 1.9 Inconvénients des Graphes de Connaissances

Bien que les graphes de connaissances offrent de nombreux avantages, leur mise en place et leur gestion présentent également certains inconvénients.

- **Complexité de la Modélisation** : La création et la gestion d'un graphe de connaissances peuvent être complexes, surtout lorsqu'il s'agit de définir des ontologies, des relations et des entités. La conception de schémas appropriés nécessite une expertise spécialisée.
- **Coût de Mise en Place** : La construction et l'entretien d'un graphe de connaissances peuvent être coûteux en termes de ressources humaines et techniques. Cela inclut la collecte, l'intégration et l'entretien des données provenant de multiples sources.
- **Problèmes de Qualité des Données** : Les graphes de connaissances dépendent fortement de la qualité des données. Si les données intégrées sont incomplètes, incorrectes ou incohérentes, cela peut compromettre la précision des inférences et des recherches.
- **Courbe d'Apprentissage** : La gestion des graphes de connaissances et l'utilisation de langages de requête comme SPARQL ou Cypher peuvent représenter une courbe d'apprentissage importante pour les utilisateurs non techniques.
- **Maintien de la Cohérence** : Lorsque de nouvelles données ou entités sont ajoutées à un graphe de connaissances, il est important de maintenir la cohérence et la validité des relations et des inférences. Cela peut nécessiter des efforts constants en matière de validation et de nettoyage des données.

## 2. État de l'Art

### 2.1 From Unstructured Text to Causal Knowledge Graphs (SpEAR)

[4] Cet article explore l'extraction des **relations causales** dans des textes non structurés à l'aide d'un modèle appelé **SpEAR**. L'objectif est de créer des **graphes de connaissances causaux (KGs)** pour une analyse automatisée des relations de cause à effet. La méthodologie s'appuie sur trois étapes principales : (1) l'extraction des entités et des relations via **SciBERT**, adapté au vocabulaire scientifique, (2) la **prédiction d'attributs** contextuels, comme la certitude épistémique, et (3) la **désambiguïsation sémantique** avec WordNet, pour clarifier les sens des termes et éviter les erreurs d'interprétation. SpEAR a montré des résultats prometteurs avec un **F1-score de 91,85 %**, et ses applications incluent la recherche scientifique et les récits ethnographiques. Cependant, des limites subsistent, telles que le besoin de jeux de données plus larges et la gestion de l'ambiguïté linguistique.



## 2.2 From Text to Knowledge with Graphs: Modelling, Querying, and Exploiting Textual Content

[5] Cet article traite de la **modélisation de données textuelles** sous forme de graphes de connaissances, en mettant l'accent sur l'interrogation et l'analyse avancées. La méthode repose sur deux étapes principales : l'**annotation et l'extraction** des entités et relations avec des pipelines NLP pour créer des triples RDF, et l'utilisation de **SPARQL** pour effectuer des requêtes complexes et des analyses graphiques, comme la détection de communautés. Les graphes générés sont utilisés dans des domaines comme la **santé**, pour explorer les relations entre symptômes et traitements, et la gestion des connaissances scientifiques. Bien que les graphes permettent une structuration cohérente, des améliorations sont nécessaires pour développer des mécanismes de requêtes plus sophistiqués et des ontologies évolutives.

## 2.3 iText2KG : Incremental Knowledge Graphs Construction Using Large Language Models Construction Incrémentielle de Graphes de Connaissances

[6] Cet article propose une méthode appelée **iText2KG**, qui vise à construire des **graphes de connaissances (KGs)** de manière incrémentielle à partir de texte non structuré. L'objectif est de surmonter les limitations des approches classiques en utilisant des **modèles de langage de grande taille (LLMs)** comme **GPT-4** pour extraire les entités, les relations, et les attributs pertinents. La méthodologie repose sur quatre composants principaux : (1) le **Document Distiller**, qui segmente les documents en blocs sémantiques ; (2) l'**Incremental Entity Extractor (iEntities)** pour identifier les entités, (3) l'**Incremental Relation Extractor (iRelation)** pour extraire les relations entre entités, et (4) un module d'**intégration et de visualisation dans Neo4j**. Cette approche a été testée sur des cas comme les CVs, les sites web et les articles scientifiques, obtenant des scores élevés de cohérence (0,97 à 0,98). Cependant, elle nécessite des techniques d'apprentissage automatique avancées pour améliorer l'automatisation et maintenir des graphes dynamiques dans des environnements en évolution rapide.

## 2.4 Knowledge Graph Generation from Text (Grapher)

[7] **Grapher** est un système conçu pour générer des **graphes de connaissances** à partir de texte, en répondant aux limites des approches traditionnelles grâce à l'utilisation de modèles de langage pré-entraînés (PLMs) comme **T5**. La méthodologie est divisée en deux étapes : la génération des **nœuds** du graphe via des PLMs pour extraire les entités, et la **construction des arêtes** entre ces nœuds à l'aide d'architectures séquentielles comme GRU/LSTM et des modèles de classification. Grapher a été évalué sur des ensembles de données variés, tels que **WebNLG 2020**, **NYT**, et **TEKGEN**, avec des performances remarquables en termes de précision,



rappel et F1-score. Les applications incluent l'organisation des données journalistiques (e.g., New York Times) et la recherche scientifique. Cependant, il existe encore des défis à relever, notamment l'amélioration des structures des graphes et l'exploration des approches multilingues.

## **2.5 Fusion d'Informations Multi-Facettes via Graphes de Connaissances**

[8] Ce travail propose une approche de fusion d'informations soutenue par un graphe de connaissances pour modéliser des concepts multi-facettes, particulièrement adaptée aux systèmes industriels. La méthodologie inclut : (1) la construction de graphes conceptuels (CKG) pour structurer les données de diagnostic, (2) la transformation en graphes temporels pour capturer les dépendances temporelles, et (3) l'analyse via des réseaux convolutionnels graphiques (GCN) et temporels (TCN). Cette méthode a été appliquée à la maintenance prédictive et au contrôle qualité industriel, montrant une amélioration significative de la précision des diagnostics. Cependant, elle nécessite des ressources de calcul importantes et une expertise approfondie pour gérer la complexité des données hétérogènes.

## **2.6 LLM-based SPARQL Query Generation from Natural Language over Federated Knowledge Graphs**

[9] Cet article propose une approche innovante pour générer des requêtes SPARQL à partir de questions en langage naturel, facilitant l'accès aux graphes de connaissances (KGs) pour les non-experts. La méthodologie repose sur la Récupération Augmentée par la Génération (RAG) : les paires question/requête sont indexées dans une base vectorielle, puis intégrées dans des prompts pour générer les requêtes. La validation automatique garantit la conformité des requêtes générées. Cette méthode a été appliquée avec succès à des graphes bioinformatiques et pourrait être étendue à d'autres domaines, comme la finance. Les limites incluent la gestion des hallucinations des LLMs et l'amélioration des indexations des entités.

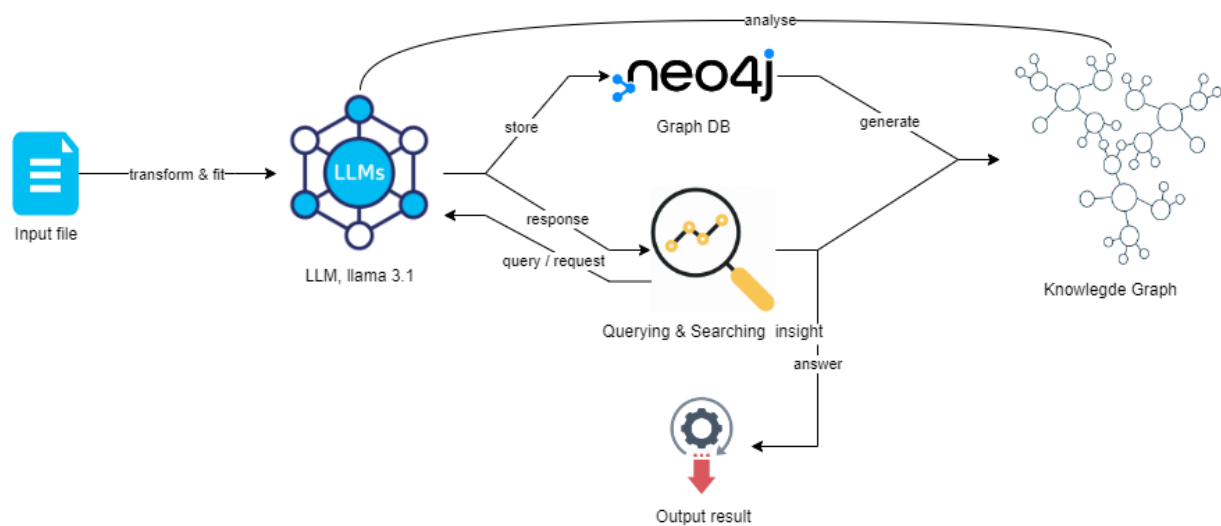
## 2.7 Tableau comparative

Figure 6 : Tableau comparatif des travaux réalisés

Tableau comparative						
Article	Objectif	Méthodologie et Technologies	Applications	Résultats	Avantages	Limitations
iText2KG	Développer des graphes de connaissances (KGs) incrémentiels à partir de texte non structuré à l'aide de LLMs.	Segmentation des documents, extraction d'entités avec GPT-4, extraction de relations, intégration dans Neo4j.	Traitement des CVs, extraction web, recherche académique.	Cohérence des schémas : 0,97-0,98 sur les ensembles de données testés.	Modulaire, adaptable à plusieurs sources, précision élevée.	Nécessite des techniques avancées d'apprentissage automatique et des mises à jour dynamiques.
Grapher	Générer efficacement des KGs à partir de texte en utilisant des modèles de langage pré-entraînés.	Génération de nœuds avec T5, construction d'arêtes avec GRU/LSTM et classificateurs.	Journalisme (e.g., New York Times), extraction de concepts scientifiques.	F1-score élevé sur WebNLG 2020, NYT, TEKGEN.	Combine PLMs et classification pour une génération robuste des KGs.	Amélioration nécessaire de la précision structurelle, manque de support multilingue.
SpEAR	Extraire les relations causales pour construire des KGs causaux à partir de texte.	Extraction d'entités et relations avec SciBERT, prédiction d'attributs, désambiguïsation avec WordNet.	Analyse scientifique, récits ethnographiques.	F1-score : 91,85 % pour les facteurs causaux.	Axié sur le raisonnement causal, forte clarté sémantique.	Limité par l'ambiguïté du langage naturel et les petits jeux de données disponibles.
Text2KG	Modéliser les textes sous forme de KGs pour faciliter l'interrogation et l'inférence.	Pipelines NLP pour l'annotation et l'extraction, SPARQL pour les requêtes complexes et l'analyse des graphes.	Santé, gestion des connaissances scientifiques.	Structuration cohérente des données textuelles, traitement avancé des requêtes.	Exploite les ontologies pour l'évolution des données.	Nécessite des mécanismes de requêtes avancés et une adaptabilité des ontologies.
Fusion Multi-Facettes	Fusionner des données multi-sources pour la modélisation conceptuelle à l'aide de KGs.	Construction de graphes conceptuels (CKG), transformation en graphes temporels, analyse avec GCN et TCN.	Maintenance prédictive industrielle, contrôle qualité.	Amélioration de la précision du diagnostic dans les contextes industriels.	Supporte la fusion temporelle et structurelle pour des insights approfondis.	Complexité élevée, demande en ressources informatiques importantes.
Génération SPARQL avec LLMs	Générer des requêtes SPARQL à partir de langage naturel pour interroger des KGs.	Récupération augmentée par la génération (RAG), indexation des embeddings, validation des requêtes.	Bioinformatique, finance, recherche pharmaceutique.	Génération précise de requêtes validées selon les schémas des KGs.	Rend les KGs accessibles aux non-experts.	Problèmes d'hallucinations des LLMs, indexation des entités perfectible.

## 3. Architecture générale

Figure 7 : Architecture générale du projet



### 3.1 Solution Proposée

Pour répondre aux défis de transformation d'informations textuelles non structurées en graphes de connaissances (KGs), nous proposons une méthodologie basée sur l'utilisation exclusive du modèle de langage **Llama 3.1 (8B)** couplé à la base de graphes **Neo4j**. Cette approche s'inspire de ressources pertinentes telles que le projet GitHub *GraphRAG-with-Llama-3.1* et met l'accent sur une solution intégrée et performante pour l'extraction, la structuration et l'interrogation des connaissances.

L'**extraction des entités et des relations** est réalisée en exploitant les capacités avancées de Llama 3.1 (8B), qui comprend le contexte linguistique de manière profonde. Chaque segment de texte est analysé pour identifier les entités clés (comme les personnes, lieux, concepts, etc.) et les relations qui les lient (par exemple, "travaille pour", "réside à", "cause de"). Les résultats sont enrichis avec des attributs tels que la confiance ou la causalité pour fournir une représentation sémantique riche.

Une fois les entités et relations extraites, elles sont structurées et intégrées dans **Neo4j**, une base de graphes puissante et flexible. Les triplets générés (entité-relation-entité) sont stockés de manière incrémentielle, permettant une mise à jour continue des graphes au fur et à mesure que de nouvelles données sont analysées. Neo4j fournit également des outils de visualisation intuitifs pour explorer les connexions entre les concepts extraits.

Enfin, notre solution permet l'**interrogation des graphes de connaissances** en langage naturel grâce à Llama 3.1. Les requêtes sont formulées par l'utilisateur en langage simple (par exemple, "Qui travaille pour l'entreprise X ?") et traduites dynamiquement en requêtes Cypher pour interroger Neo4j. Cette combinaison permet non seulement de répondre à des questions complexes mais aussi de découvrir des relations inattendues entre les données.

cette méthodologie, centrée sur Llama 3.1 (8B) et Neo4j, offre une solution robuste et évolutive pour traiter des données non structurées, les transformer en graphes exploitables, et les interroger efficacement. Elle se distingue par sa capacité à capturer des nuances contextuelles complexes tout en offrant une flexibilité graphique et une scalabilité adaptée à des volumes de données variés. Cette approche s'adresse à des applications dans divers domaines tels que la santé, la recherche scientifique, et les systèmes de recommandation.

## 4. Étude des solutions

### 4.1 Google Collab

- **Description** : Google Colaboratory est un environnement de développement en ligne gratuit qui permet d'exécuter du code Python dans un navigateur. Il est particulièrement utile pour l'apprentissage automatique et la science des données.
- **Fonctionnalités clés** :
  - **Accès à des GPUs**: Accès gratuit à des accélérateurs matériels pour accélérer l'entraînement des modèles.

- **Intégration avec Google Drive:** Stockage et partage de notebooks.
  - **Collaboration:** Possibilité de collaborer avec d'autres utilisateurs en temps réel.
- **Utilisation dans le projet :**
  - **Expérimentation:** Expérimentation rapide avec de nouvelles idées et modèles.
  - **Prototypage:** Création de prototypes d'applications.
  - **Enseignement:** Création de tutoriels et de cours interactifs.

## 4.2 Python

- **Description :** Python est un langage de programmation de haut niveau, interprété, et orienté objet. Il est apprécié pour sa syntaxe simple et lisible, ce qui le rend accessible aux débutants et aux experts.
- **Fonctionnalités clés :**
  - **Polyvalence:** Utilisé dans de nombreux domaines, tels que le développement web, l'analyse de données, la science des données, l'apprentissage automatique et l'automatisation.
  - **Grand écosystème de bibliothèques:** Une vaste collection de bibliothèques (Pandas, NumPy, Matplotlib, Scikit-learn, TensorFlow, etc.) offre des outils puissants pour diverses tâches.
  - **Communauté active:** Une communauté importante contribue à son développement et fournit une abondance de ressources et de tutoriels.
- **Utilisation dans le projet :**
  - **Traitement des données:** Nettoyage, transformation et exploration des données.
  - **Modélisation:** Développement et évaluation de modèles d'apprentissage automatique.
  - **Visualisation:** Création de graphiques et de visualisations pour comprendre les données.
  - **Automatisation:** Création de scripts pour automatiser des tâches répétitives.

## 4.3 Ollama

- **Description :** Ollama est une API qui permet de servir et d'interagir avec des modèles de langage. Elle offre une infrastructure pour déployer et exécuter des modèles de manière efficace et scalable.
- **Fonctionnalités clés :**
  - **Déploiement de modèles:** Facilite le déploiement de modèles de langage sur différentes plateformes.
  - **API RESTful:** Fournit une interface simple pour interagir avec les modèles.
  - **Scalabilité:** Capable de gérer un grand nombre de requêtes.
- **Utilisation dans le projet :**
  - **Hébergement de modèles:** Hébergement des modèles de langage utilisés dans l'application.

- **Exposition d'une API:** Mise à disposition d'une API pour que d'autres applications puissent utiliser les modèles.

## 4.4 Langchain

- **Description :** Langchain est une bibliothèque Python qui facilite la création d'applications basées sur les modèles de langage (LLM). Elle permet de connecter les LLM à des sources de données externes, de créer des chaînes de traitement de texte et de construire des agents conversationnels.
- **Fonctionnalités clés :**
  - **Modularité:** Structure modulaire pour une grande flexibilité dans la construction d'applications.
  - **Intégration avec les LLM:** Compatible avec de nombreux modèles de langage populaires (GPT-3, Hugging Face Transformers, etc.).
  - **Connecteurs de données:** Permet de connecter les LLM à des bases de données, des API et d'autres sources d'informations.
- **Utilisation dans le projet :**
  - **Création de chatbots:** Développement de chatbots intelligents capables de répondre à des questions complexes.
  - **Recherche d'informations:** Recherche d'informations dans de grandes quantités de texte.
  - **Génération de texte:** Création de contenu textuel, comme des résumés, des traductions ou des emails.

## 4.5 LLaMA 3.1 8b model

- **Description :** LLaMA 3.1 8b est un modèle de langage de grande taille développé par Meta AI. Il est capable de générer du texte, de traduire des langues, de créer du code et bien plus encore.
- **Fonctionnalités clés :**
  - **Génération de texte de haute qualité:** Capacité de générer du texte cohérent et contextuellement pertinent.
  - **Polyvalence:** Peut être utilisé pour diverses tâches de traitement du langage naturel.
- **Utilisation dans le projet :**
  - **Moteur de conversation:** Fournit les capacités de compréhension et de génération de texte nécessaires pour créer des chatbots.
  - **Traduction automatique:** Effectue des traductions entre différentes langues.

## 4.6 mxbai-embed-large model

- **Description :** mxbai-embed-large est un modèle d'encodage qui convertit du texte en représentations numériques (embeddings). Ces embeddings capturent les relations sémantiques entre les mots et les phrases.
- **Fonctionnalités clés :**
  - **Encodage sémantique:** Représentation vectorielle des mots et des phrases.
  - **Similarité sémantique:** Calcul de la similarité entre les textes.
- **Utilisation dans le projet :**
  - **Recherche sémantique:** Recherche de documents similaires.
  - **Classification de texte:** Classification de textes en catégories.

## 4.7 Neo4j

- **Description :** Neo4j est une base de données NoSQL spécialisée dans les graphes. Elle permet de représenter et de stocker des données sous forme de nœuds (entités) et de relations entre ces nœuds.
- **Fonctionnalités clés :**
  - **Modélisation de données relationnelles:** Représentation intuitive des données complexes et interconnectées.
  - **Requêtes puissantes:** Langage de requête Cypher pour naviguer et interroger les graphes de manière efficace.
  - **Scalabilité:** Capable de gérer des graphes de grande taille.
- **Utilisation dans le projet :**
  - **Représentation des connaissances:** Création de graphes de connaissances pour stocker et interroger des informations sémantiques.
  - **Analyse de réseaux:** Analyse des relations entre les entités.
  - **Recommandation:** Recommandation de produits ou de contenus en fonction des préférences de l'utilisateur.
- **Utilisation dans le projet :**
  - **Recherche sémantique:** Recherche de documents similaires.
  - **Classification de texte:** Classification de textes en catégories.

## Conclusion

*Ce chapitre a permis de poser les bases conceptuelles et méthodologiques du projet en décrivant les caractéristiques essentielles des graphes de connaissances et leur utilité dans divers contextes. L'analyse des travaux de recherche existants et l'étude des solutions technologiques offrent une perspective claire des défis à relever et des opportunités à saisir. Ces éléments fondamentaux serviront de guide pour les étapes suivantes, en orientant le développement et l'implémentation du système proposé.*

## Chapitre 2 : Etapes, Implémentation, Résultat



### Introduction



*Dans ce chapitre, nous présentons les étapes pratiques du projet, de la préparation des données à la génération des résultats finaux. Nous commençons par décrire les caractéristiques des données utilisées et les questions posées dans le cadre de l'analyse. Ensuite, nous détaillons chaque phase d'implémentation, notamment l'installation des outils nécessaires, la configuration des modèles et des API, ainsi que la construction et l'exploitation des graphes de connaissances dans une base Neo4j. Enfin, nous expliquons comment ces éléments ont été intégrés pour développer un système de question-réponse performant basé sur la technique de Récupération Augmentée par la Génération (RAG).*





Tout le code source, les données utilisées, et les scripts associés au projet sont disponibles dans le répertoire GitHub dédié. Ce répertoire offre une documentation complète pour comprendre, reproduire et étendre les travaux réalisés. Vous y trouverez également des instructions détaillées pour l'installation, l'exécution des modèles, et la manipulation des graphes de connaissances [10].

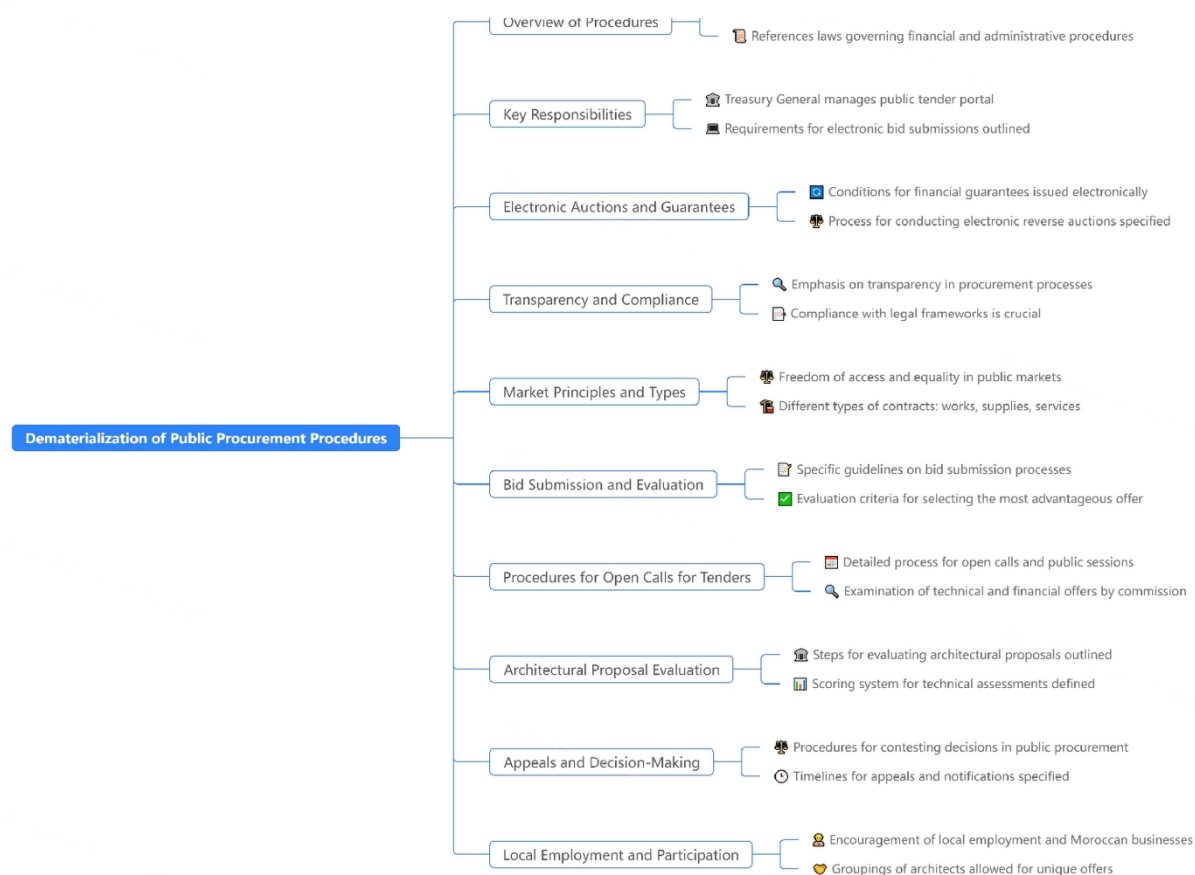
-  Lien vers le répertoire : [GitHub Repository](#)

## 1. Les Données

### 1.1 Résumé

Ces données constituent un bulletin officiel présentant les procédures de dématérialisation des documents et des processus liés aux marchés publics. Il fait référence aux lois et règlements pertinents, mettant l'accent sur l'utilisation des signatures électroniques et des plateformes en ligne pour les appels d'offres publics. Le bulletin définit également les rôles des différents acteurs, y compris la Trésorerie générale du Royaume, dans la gestion des portails des marchés publics. Il détaille les exigences relatives à la soumission électronique des offres, à leur évaluation ainsi qu'aux enchères inversées électroniques, tout en insistant sur l'importance de la transparence, de la sécurité et de la conformité légale dans les marchés publics. Il inclut également les critères d'éligibilité et d'évaluation des offres, le processus des appels d'offres ouverts et restreints, ainsi que des lignes directrices pour les propositions architecturales, garantissant ainsi un processus d'approvisionnement juste et équitable.

Figure 8 : Carte mentale des données non structurées



## 1.2 Points clés

- Le document met en lumière la transition vers des processus d'approvisionnement numériques, améliorant l'efficacité et la transparence.
- Il établit des responsabilités claires pour les différentes parties prenantes impliquées dans les marchés publics, assurant ainsi la responsabilité.
- L'accent mis sur les soumissions électroniques et les enchères inversées reflète une approche moderne de l'approvisionnement, réduisant les formalités administratives et simplifiant les processus.
- Des critères stricts d'éligibilité et des processus d'évaluation sont essentiels pour garantir que les marchés publics soient compétitifs et équitables.
- Le document décrit les garanties procédurales nécessaires pour protéger les fonds publics et maintenir l'intégrité des décisions d'approvisionnement.

## 1.3 Questions posées

- **Quels sont les principaux objectifs des procédures de dématérialisation présentées dans le bulletin ?**
  - Les objectifs sont d'améliorer la transparence, d'améliorer l'efficacité des marchés publics et d'assurer la conformité avec les cadres juridiques à travers des processus numériques.
- **Qui est responsable de la gestion du portail des marchés publics selon le bulletin ?**
  - La Trésorerie Générale du Royaume est responsable de la gestion du portail des marchés publics, supervisant l'ensemble du processus d'approvisionnement électronique.
- **Quels critères sont utilisés pour évaluer les offres dans les marchés publics ?**
  - Les offres sont évaluées en fonction de critères techniques et financiers, garantissant que la proposition la plus avantageuse économiquement soit sélectionnée.
- **Comment le bulletin assure-t-il une concurrence équitable entre les soumissionnaires ?**
  - Le bulletin définit des critères d'éligibilité stricts, assure la transparence du processus d'appel d'offres et impose des sessions publiques lors de l'ouverture des offres pour maintenir l'équité.

## 2. Les Etapes d'implémentation

### 2.1 Installation des Packages Requis

Pour garantir le bon fonctionnement du projet, plusieurs bibliothèques et outils ont été installés. Ces outils facilitent la gestion des graphes de connaissances, l'interaction avec les modèles de langage, ainsi que le traitement des données.

Les bibliothèques principales installées incluent :

- **LangChain et ses extensions** : pour interagir avec les modèles de langage et gérer les flux de données.
- **Neo4j** : pour la gestion des graphes de connaissances.
- **tiktoken** : pour gérer les jetons des modèles de langage.
- **yfiles\_jupyter\_graphs** : pour visualiser les graphes dans Jupyter.
- **python-dotenv** : pour gérer les variables d'environnement.
- **json-repair** : pour réparer les structures JSON endommagées.

En outre, des outils supplémentaires ont été installés, tels que **pciutils** (pour interagir avec les périphériques PCI) et **Ollama** (pour servir et interagir avec les modèles de langage).

```
%pip install --upgrade --quiet langchain langchain-community langchain-ollama
langchain-experimental neo4j tiktoken yfiles_jupyter_graphs python-dotenv
json-repair langchain-openai langchain_core
!sudo apt-get install -y pciutils
!curl -fsSL <https://ollama.com/install.sh> | sh
```

### 2.2 Configuration de l'API Ollama et Téléchargement du Modèle LLama

Cette étape décrit la configuration du serveur API Ollama pour interagir avec le modèle de langage LLama 3.1.

La fonction `ollama` a été créée pour :

- Configurer les variables d'environnement nécessaires.
- Démarrer le serveur API avec la commande `ollama serve`, exécutée en arrière-plan via un thread séparé.

Ensuite, le modèle LLama 3.1 a été téléchargé en utilisant la commande `ollama pull llama3.1`. Ce modèle est utilisé pour diverses tâches de traitement du langage naturel.

```
from IPython.display import clear_output
import os
import threading
import subprocess
```

```
def ollama():
    os.environ['OLLAMA_HOST'] = '0.0.0.0:11434'
    os.environ['OLLAMA_ORIGINS'] = '*'
    subprocess.Popen(["ollama", "serve"])

ollama_thread = threading.Thread(target=ollama)
ollama_thread.start()
```

## 2.3 Définition des Classes pour les Nœuds, Relations et Documents

Cette section définit les classes Python nécessaires pour organiser et manipuler les données des graphes de connaissances. Ces classes incluent `Node`, `Relationship`, `Document`, et `GraphDocument` :

- **Node** : Représente un nœud dans le graphe avec un identifiant, un type et des propriétés.
- **Relationship** : Représente une relation entre deux nœuds, avec des références aux nœuds source et cible, ainsi que le type et les propriétés de la relation.
- **Document** : Contient des métadonnées et le contenu textuel d'un document source.
- **GraphDocument** : Regroupe des nœuds et des relations dans un même objet, avec la référence au document source.

## 2.4 Traitement et Sérialisation des Documents en Graphes de Connaissances

1. **Initialisation du Modèle de Langage** : Un modèle de langage, `ChatOllama`, est utilisé avec le modèle `llama3.1` pour convertir des documents en graphes de connaissances via le transformateur `LLMGraphTransformer`. Le modèle est configuré pour produire des sorties en format JSON, permettant ainsi une manipulation plus facile des données.
2. **Sérialisation des Objets** : La fonction `serialize_document` convertit des objets `Document` ou `GraphDocument` en un format sérialisable en vérifiant la présence d'une méthode `to_dict` ou en itérant sur les attributs de l'objet. Elle prend en charge la sérialisation des listes, dictionnaires et types primitifs.
3. **Traitement des Documents** : Les documents sont transformés en graphes de connaissances à l'aide du transformateur. Les résultats sont ensuite sérialisés et sauvegardés dans un fichier JSON (`progress.json`) après chaque transformation. Cela permet de suivre les progrès du traitement et de garantir la persistance des données.
4. **Chargement et Reconstitution des Objets** : Une fois tous les documents traités, les données sérialisées sont lues depuis le fichier `progress.json` et les objets `GraphDocument` sont reconstitués à partir des données JSON.

Figure 9 : Ontologies Extraction à partir de morceaux de données

```

925 {
926   "nodes": [
927     {
928       "id": "Commission Nationale de la Commande Publique",
929       "type": "Organization",
930       "properties": {}
931     },
932     {
933       "id": "avis \u00e0 l'attention d'elle",
934       "type": "Publication",
935       "properties": {}
936     }
937   ],
938   "relationships": [
939     {
940       "source": {
941         "id": "Commission Nationale de la Commande Publique",
942         "type": "Organization",
943         "properties": {}
944       },
945       "target": {
946         "id": "avis \u00e0 l'attention d'elle",
947         "type": "Publication",
948         "properties": {}
949       },
950       "type": "PUBLISHED BY",
951       "properties": {}
952     },
953     {
954       "source": {
955         "id": null,
956         "metadata": {
957           "source": "data2.pdf",
958           "page": 4
959         },
960         "page_content": "c) la commission nationale de la commande publique est charg\u00e9e de la publication des avis \u00e0 l'attention d'elle ; \u00d9 Les administrations habi",
961         "type": "Document"
962       }
963     }
964   ]
965 }

```

## 2.5 Chargement et Transformation des Données JSON en GraphDocuments

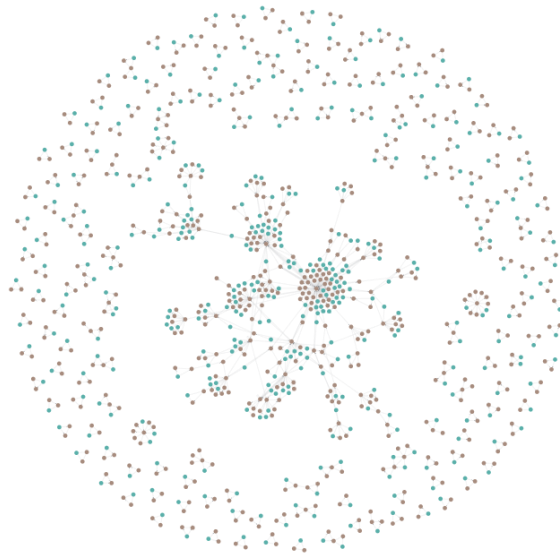
Cette étape décrit le processus de chargement des données JSON à partir d'un fichier et de leur transformation en objets `GraphDocument`. Ces objets représentent des graphes de connaissances, contenant des nœuds, des relations et des documents sources, ce qui facilite leur gestion et leur intégration dans des systèmes comme Neo4j.

1. **Chargement des Données JSON** : Les données JSON sont chargées depuis un fichier `progress.json` à l'aide de la bibliothèque `json`. Ces données contiennent des informations sur les nœuds, les relations et les métadonnées associées aux documents.
2. **Transformation en Objets `GraphDocument`** : La fonction `transform_json_to_graph` prend les données JSON et les transforme en objets `GraphDocument`. Cela permet de créer un modèle structuré de graphes de connaissances.

Pour chaque élément des données JSON :

- a. **Création des Nœuds** : Les nœuds sont créés à partir des informations d'identifiant, de type et de propriétés.
  - b. **Création des Relations** : Les relations sont créées en associant les nœuds source et cible, en précisant leur type et leurs propriétés.
  - c. **Création du Document Source** : Un objet `Document` est généré à partir des métadonnées et du contenu de la page du document source.
3. **Gestion des Données Incomplètes** : Si un élément ne contient ni nœuds ni relations, il est ignoré afin de garantir que seuls les éléments complets soient traités.
  4. **Enregistrement des Graphes** : Après avoir créé les objets `GraphDocument`, ces derniers sont ajoutés à une liste qui peut être utilisée pour une visualisation ultérieure ou intégrée dans une base de données Neo4j.

Figure 10 : Diagramme de graphe de connaissances

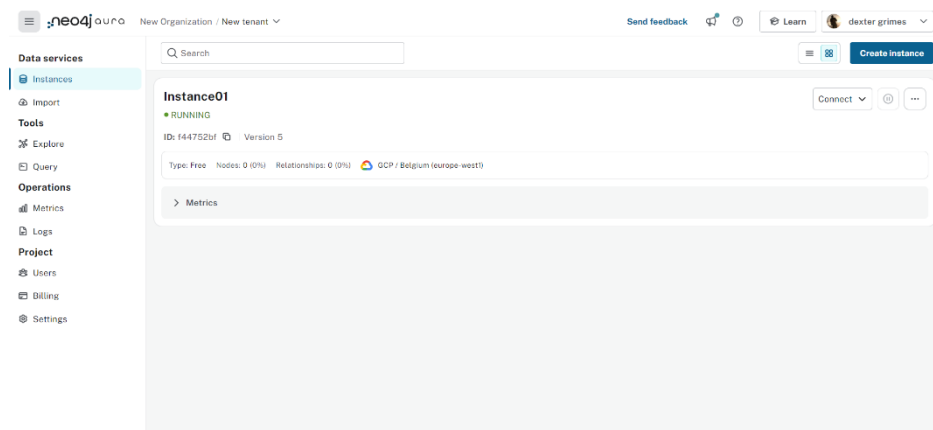


## 2.6 Configuration de la Connexion Neo4j et Ajout des GraphDocuments

Dans cette section, nous abordons la configuration de la connexion à la base de données Neo4j et l'ajout des objets `GraphDocument` transformés dans cette base. Neo4j est une base de données orientée graphes, particulièrement adaptée à la gestion de données complexes et interconnectées, comme les graphes de connaissances.

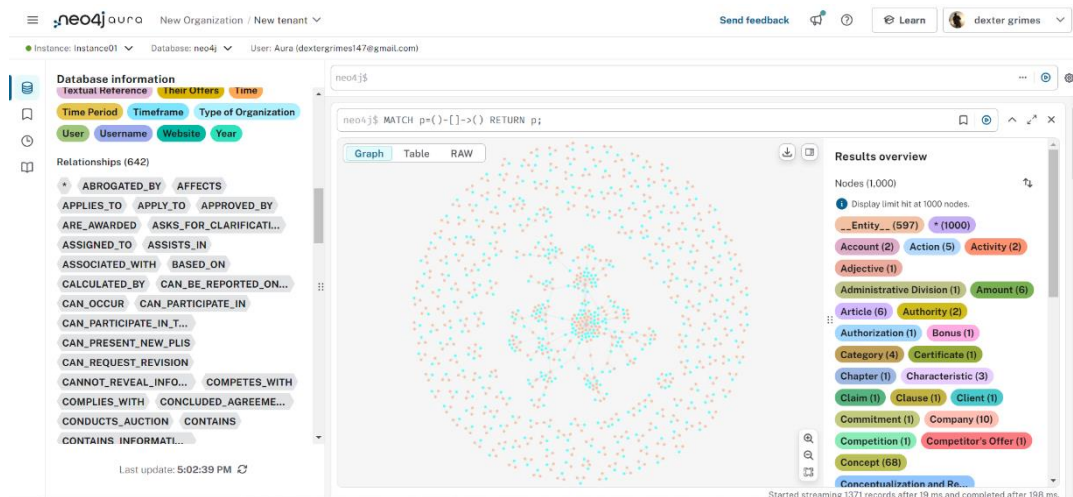
- 1. Connexion à Neo4j :** La connexion à Neo4j est établie en utilisant les informations d'authentification (URI, nom d'utilisateur et mot de passe) récupérées à partir des variables d'environnement. Cela garantit que les informations sensibles ne sont pas codées en dur dans le script.

Figure 11 : Interface Noe4j



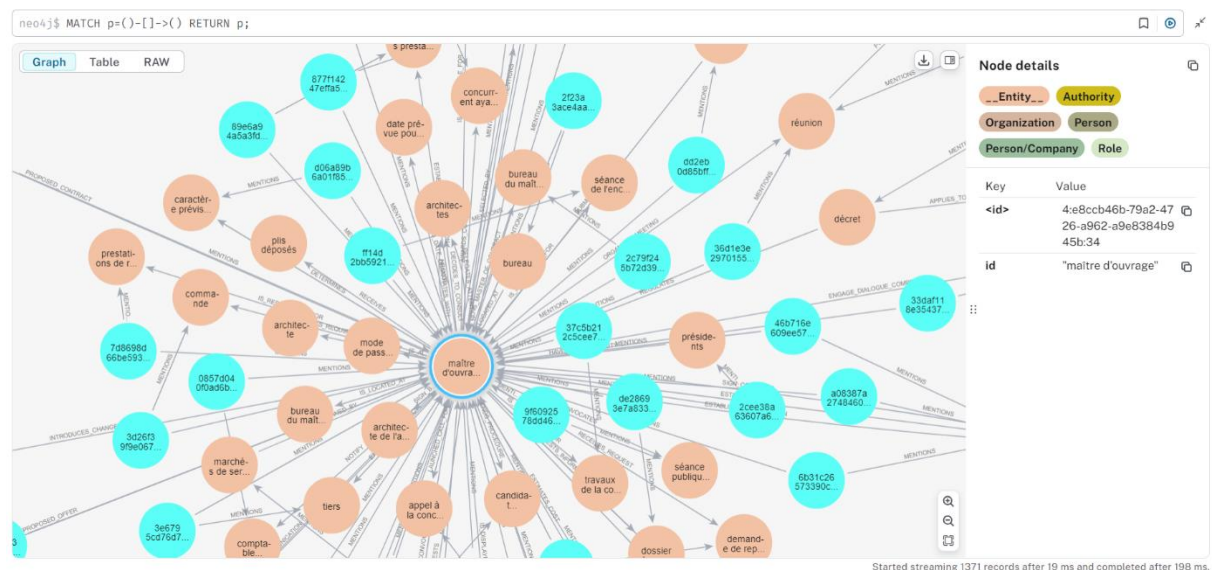
2. **Manipulation des Graphes avec Neo4jGraph** : Une fois la connexion établie, un objet Neo4jGraph est créé. Cet objet permet d'interagir facilement avec la base de données Neo4j et d'ajouter des graphes de connaissances sous forme de GraphDocument.
3. **Ajout des GraphDocuments** :
  1. Les objets GraphDocument, qui contiennent des nœuds, des relations et des informations documentaires, sont ajoutés à Neo4j en utilisant la méthode add\_graph\_documents de l'objet Neo4jGraph.

**Figure 12 : Entités extraites du document graphique et interrogation de toutes les entités**



2. Les paramètres `baseEntityLabel=True` et `include_source=True` assurent que les entités de base sont étiquetées correctement et que les informations source (métadonnées et contenu du document) sont incluses dans le graphe.

**Figure 13 : Exemple de contenu d'entités**





## 2.7 Création d'un Index en Texte Intégral dans Neo4j

1. **Contexte** : Un index en texte intégral dans Neo4j permet de rechercher efficacement des propriétés textuelles des nœuds du graphe. Cet index est essentiel pour accélérer les recherches sur des données complexes, en particulier lorsque ces données contiennent du texte libre ou des identifiants qui doivent être rapidement accessibles.
2. **Définition de l'Index** : L'index est créé en utilisant la syntaxe Cypher dans Neo4j, ciblant les propriétés des nœuds de type `__Entity__` avec un index basé sur la propriété `id`. Ce type d'index permet d'optimiser les requêtes de recherche en texte intégral, qui sont utilisées pour trouver des nœuds ou des relations de manière rapide et efficace.
3. **Création et Vérification de l'Index** : Une fonction `create_index` est définie pour créer l'index en texte intégral dans la base de données Neo4j. Cette fonction exécute la requête Cypher dans une session Neo4j pour créer l'index. Si l'index existe déjà, la fonction capture et ignore les erreurs, garantissant ainsi que l'index soit créé une seule fois.
4. **Fermeture de la Connexion** : Après la création de l'index, il est important de fermer la connexion à la base de données pour libérer les ressources.

Figure 14 : Création d'un Index en Texte Intégral dans Neo4j

```
driver = GraphDatabase.driver(
    uri = os.environ.get('NEO4J_URI'),
    auth = (os.environ.get('NEO4J_USERNAME'), os.environ.get('NEO4J_PASSWORD')))

def create_fulltext_index(tx):
    query = """
    CREATE FULLTEXT INDEX `fulltext_entity_id`
    FOR (n: __Entity__)
    ON EACH [n.id];
    """
    tx.run(query)

# Function to execute the query
def create_index():
    with driver.session() as session:
        session.execute_write(create_fulltext_index)
        print("Fulltext index created successfully.")

# Call the function to create the index
try:
    create_index()
except:
    pass

# Close the driver connection
driver.close()
```

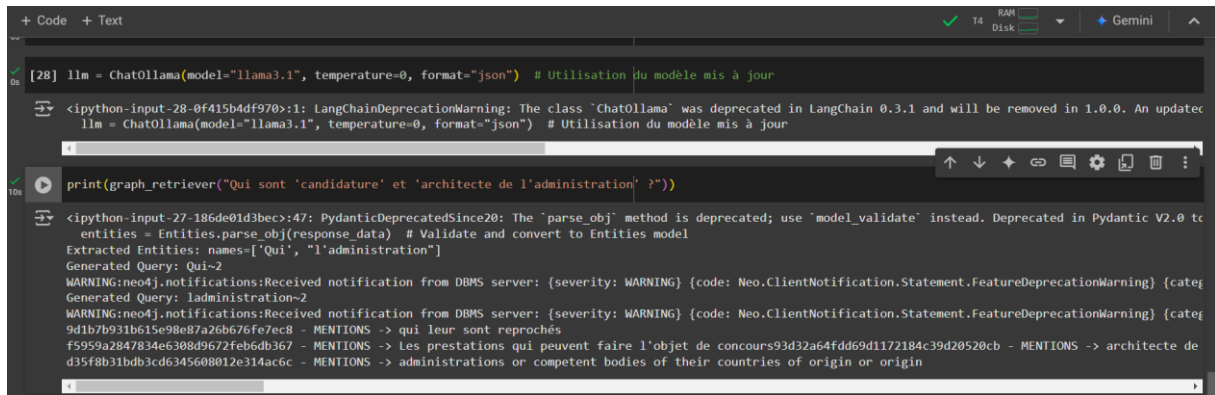
## 2.8 Extraction d'Entités et Recherche dans Neo4j

1. **Extraction des Entités** : Les entités importantes, telles que les personnes, organisations et entreprises, sont extraites du texte en utilisant un modèle de langage. Cela permet de structurer le texte et de convertir les mentions en entités exploitables dans un graphe de connaissances.
2. **Création de la Requête de Recherche** : Une fois les entités extraites, des requêtes en texte intégral sont générées pour rechercher ces entités dans le graphe de Neo4j. Ces requêtes sont conçues pour récupérer les relations entre les entités et d'autres nœuds associés dans le graphe.



3. **Récupération des Données du Graphe** : Pour chaque entité extraite, une requête est envoyée à la base de données Neo4j pour récupérer les relations pertinentes entre les nœuds. Les résultats sont formatés et retournés sous forme de chaîne.

Figure 15 : Extraction d'Entités et Recherche dans Neo4j



```
[28] llm = ChatOllama(model="llama3.1", temperature=0, format="json") # Utilisation du modèle mis à jour

<ipython-input-28-0f415b4df970>:1: LangChainDeprecationWarning: The class 'ChatOllama' was deprecated in LangChain 0.3.1 and will be removed in 1.0.0. An updated
llm = ChatOllama(model="llama3.1", temperature=0, format="json") # Utilisation du modèle mis à jour

print(graph_retriever("Qui sont 'candidature' et 'architecte de l'administration' ?"))

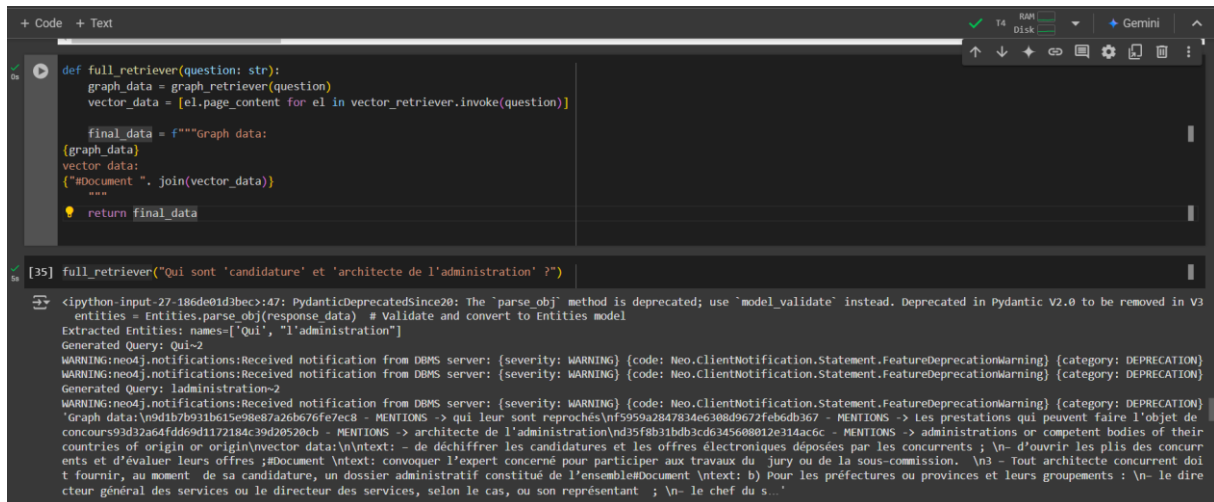
<ipython-input-27-186de01d3bec>:47: PydanticDeprecationWarning: The 'parse_obj' method is deprecated; use 'model_validate' instead. Deprecated in Pydantic V2.0 to
entities = Entities.parse_obj(response_data) # Validate and convert to Entities model
Extracted Entities: names=['Qui', "l'administration"]
Generated Query: Qui~2
WARNING:neo4j.notifications:Received notification from DBMS server: {severity: WARNING} {code: Neo.ClientNotification.Statement.FeatureDeprecationWarning} {categ
Generated Query: ladministration~2
WARNING:neo4j.notifications:Received notification from DBMS server: {severity: WARNING} {code: Neo.ClientNotification.Statement.FeatureDeprecationWarning} {categ
9d1b7b931b615e98e87a26b676fe7ec8 - MENTIONS -> qui leur sont reprochés
f5959a2847834e6308d9672feb6db367 - MENTIONS -> Les prestations qui peuvent faire l'objet de concours93d32a64fdd69d1172184c39d20520cb - MENTIONS -> architecte de
d35f8b31bdb3cd6345608012e314ac6c - MENTIONS -> administrations or competent bodies of their countries of origin or origin
```

## 2.9 Définition du Full Retriever et de la Chaîne de Q&A

1. **Objectif** : Cette section du code vise à créer une chaîne de traitement qui combine les données de graphes et les données vectorielles pour fournir des réponses contextuelles et précises à des questions spécifiques. Le Full Retriever joue un rôle crucial en récupérant les données nécessaires à partir des deux sources de données (graphes et embeddings) et en les combinant pour offrir un contexte complet pour la réponse.
2. **Fonction `full_retriever`** : La fonction `full_retriever` combine les résultats provenant du `graph_retriever` et du `vector_retriever`. Elle récupère d'abord les données de graphes pertinentes pour une question donnée, puis obtient les données vectorielles via le `vector_retriever`. Ces données sont ensuite intégrées dans une chaîne de caractères, formatée pour être utilisée dans la suite du processus.
3. **Template de Prompt** : Le modèle de prompt est conçu pour formuler des réponses en utilisant les données combinées. Il inclut une instruction claire demandant au modèle de langage de répondre de manière concise et naturelle, tout en précisant si l'information provient des "Graph data" ou des "Vector data". Cela aide à tracer la provenance des informations et à fournir des références précises.

4. **Création de la Chaîne de Q&A** : La chaîne de Q&A est ensuite formée en utilisant plusieurs composants. Elle commence par la combinaison du contexte (données de graphes et vectorielles) et de la question via `full_retriever`. Ensuite, le modèle de prompt est utilisé pour formuler la réponse, qui est générée par le modèle de langage (`llm`). Enfin, un analyseur (`StrOutputParser`) est utilisé pour traiter et extraire la réponse finale.

Figure 16 : Définition du Full Retriever et de la Chaîne de Q&A



```
def full_retriever(question: str):
    graph_data = graph_retriever(question)
    vector_data = [el.page_content for el in vector_retriever.invoke(question)]

    final_data = f"""Graph data:
{graph_data}
vector data:
{"#Document ".join(vector_data)}
"""
    return final_data
```

[35] full\_retriever("Qui sont 'candidature' et 'architecte de l'administration' ?")

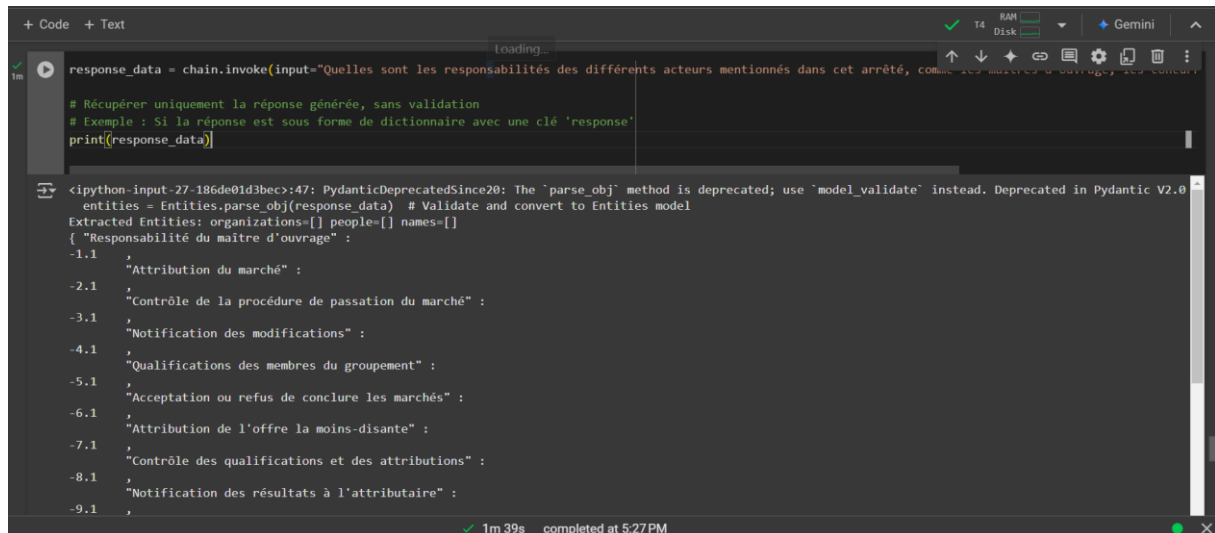
<ipython-input-27-186de01d3bec>:47: PydanticDeprecatedSince20: The 'parse\_obj' method is deprecated; use 'model\_validate' instead. Deprecated in Pydantic V2.0 to be removed in V3  
entities = Entities.parse\_obj(response\_data) # Validate and convert to Entities model  
Extracted Entities: names=['Qui', 'l'administration']  
Generated Query: Qui-2  
WARNING:neo4j.notifications:Received notification from DBMS server: {severity: WARNING} {code: Neo.ClientNotification.Statement.FeatureDeprecationWarning} {category: DEPRECATION}  
WARNING:neo4j.notifications:Received notification from DBMS server: {severity: WARNING} {code: Neo.ClientNotification.Statement.FeatureDeprecationWarning} {category: DEPRECATION}  
Generated Query: Administration-2  
WARNING:neo4j.notifications:Received notification from DBMS server: {severity: WARNING} {code: Neo.ClientNotification.Statement.FeatureDeprecationWarning} {category: DEPRECATION}  
Graph data:\n0d1b7b931b615e98e87a26b676fe7ec8 - MENTIONS -> qui leur sont reprochés\nf5959a2847834e6308d9672feb6db367 - MENTIONS -> Les prestations qui peuvent faire l'objet de concours93d32a64fdd69d1172184c39d20520cb - MENTIONS -> architecte de l'administration\nd35f8b31bdb3cd6345608012e314ac6c - MENTIONS -> administrations or competent bodies of their countries of origin or origin\nvector data:\nntext: - de déchiffrer les candidatures et les offres électroniques déposées par les concurrents ; \n- d'ouvrir les plis des concurrents et d'évaluer leurs offres ;\nDocument \ntext: convoquer l'expert concerné pour participer aux travaux du jury ou de la sous-commission. \n3 - Tout architecte concurrent doit fournir, au moment de sa candidature, un dossier administratif constitué de l'ensemble\nDocument \ntext: b) Pour les préfectures ou provinces et leurs groupements : \n- le directeur général des services ou le directeur des services, selon le cas, ou son représentant ; \n- le chef du s...

## 2.10 Exécution de la Q&A

1. **Objectif** : Cette étape montre comment exécuter la chaîne de Q&A pour répondre à une question spécifique en utilisant les données de graphes et les embeddings. Cela démontre l'efficacité de la chaîne pour générer des réponses contextualisées et précises en se basant sur des données structurées et non structurées.
2. **Invocation de la Chaîne de Q&A** : La question "Quelles sont les responsabilités des différents acteurs mentionnés dans cet arrêté, comme les maîtres d'ouvrage, les concurrents et les agents de contrôle ?" est passée à la chaîne de Q&A. Cette question sert d'entrée pour la chaîne qui combine les différentes sources de données et produit une réponse appropriée.
3. **Traitement de la Question** : Le Full Retriever est utilisé pour récupérer les données pertinentes pour la question à partir du graphe et des embeddings. Ces données sont ensuite passées au modèle de langage qui génère une réponse basée sur le contexte complet. Le modèle est guidé par le template de prompt pour s'assurer que la réponse est concise et fournit des références claires à la source des informations.

4. **Affichage de la Réponse** : La réponse générée par la chaîne de Q&A est stockée dans la variable `response_data` et affichée avec `print`. La réponse contient des informations détaillées et contextuelles sur les responsabilités des acteurs, en citant les sources des données utilisées pour formuler la réponse.

Figure 17 : Exemple d'exécution des questions-réponses



```
response_data = chain.invoke(input="Quelles sont les responsabilités des différents acteurs mentionnés dans cet arrêté, comme les maîtres d'ouvrage, les concou...")

# Récupérer uniquement la réponse générée, sans validation
# Exemple : Si la réponse est sous forme de dictionnaire avec une clé 'response'
print(response_data)
```

```
<ipython-input-27-186de01d3bec>:47: PydanticDeprecatedSince20: The `parse_obj` method is deprecated; use `model_validate` instead. Deprecated in Pydantic V2.0
entities = Entities.parse_obj(response_data) # Validate and convert to Entities model
Extracted Entities: organizations=[] people=[] names=[]
{ "Responsabilité du maître d'ouvrage" :
-1.1 ,
  "Attribution du marché" :
-2.1 ,
  "Contrôle de la procédure de passation du marché" :
-3.1 ,
  "Notification des modifications" :
-4.1 ,
  "Qualifications des membres du groupement" :
-5.1 ,
  "Acceptation ou refus de conclure les marchés" :
-6.1 ,
  "Attribution de l'offre la moins-disante" :
-7.1 ,
  "Contrôle des qualifications et des attributions" :
-8.1 ,
  "Notification des résultats à l'attributaire" :
-9.1 ,
```

## Conclusion

*Ce chapitre a permis de documenter l'ensemble du processus d'implémentation, depuis l'extraction et la structuration des données jusqu'à la génération de réponses contextuelles via le système développé. Les résultats obtenus illustrent la robustesse et l'efficacité de l'approche adoptée. Ces acquis seront discutés et mis en perspective dans la conclusion générale, avec des propositions pour des améliorations futures et des pistes de recherche complémentaires.*

## Conclusion et perspectives

*Ce projet a abouti au développement d'un système de question-réponse innovant reposant sur un graphe de connaissances. Plusieurs défis majeurs ont été relevés, notamment l'extraction d'entités et de relations à partir de textes non structurés, la construction et le stockage du graphe de connaissances dans une base de données Neo4j, ainsi que l'intégration de la technique de Récupération Augmentée par la Génération (RAG) pour améliorer la pertinence des réponses. Grâce à l'utilisation du modèle de langage LLaMA 3.1, le système a pu générer des réponses cohérentes, fluides et en langage naturel. Les résultats obtenus valident la faisabilité et le potentiel de cette approche pour une gestion plus intelligente et efficace de l'information.*

*Les perspectives pour l'avenir sont nombreuses et prometteuses :*

- **Utilisation de Modèles de Langage Plus Avancés** : Explorer des modèles de langage de nouvelle génération pour améliorer encore la qualité et la précision des réponses générées.
- **Amélioration des Méthodes d'Évaluation** : Développer des métriques plus robustes et des méthodologies pour évaluer la pertinence et l'exactitude des réponses.
- **Comparaison des Performances** : Étudier les performances du système par rapport à d'autres solutions existantes, afin d'identifier les points forts et les axes d'amélioration.
- **Intégration de Techniques Avancées de Raisonnement** : Incorporer des mécanismes sophistiqués pour le raisonnement sur le graphe, afin de traiter des questions nécessitant des inférences complexes ou une analyse contextuelle approfondie.

*Ces évolutions permettront de construire des systèmes de question-réponse encore plus performants, robustes et adaptés aux besoins variés des utilisateurs. En capitalisant sur les avancées technologiques et les nouvelles méthodologies, ce travail ouvre la voie à des applications concrètes et innovantes dans de nombreux domaines, tels que la recherche documentaire, la gestion des connaissances, et l'aide à la prise de décision.*

# Bibliographie

- [1] “What Is a Knowledge Graph? | IBM.” Accessed: Dec. 26, 2024. [Online]. Available: <https://www.ibm.com/think/topics/knowledge-graph>
- [2] “Knowledge graph,” *Wikipedia*. Dec. 25, 2024. Accessed: Dec. 26, 2024. [Online]. Available: [https://en.wikipedia.org/w/index.php?title=Knowledge\\_graph&oldid=1265199555](https://en.wikipedia.org/w/index.php?title=Knowledge_graph&oldid=1265199555)
- [3] S. Ghosh, “Knowledge Graphs — What, Why, and How,” Medium. Accessed: Dec. 26, 2024. [Online]. Available: <https://samadritaghosh.medium.com/knowledge-graphs-what-why-and-how-84f920316ca5>
- [4] S. Friedman, I. Magnusson, V. Sarathy, and S. Schmer-Galunder, “From Unstructured Text to Causal Knowledge Graphs: A Transformer-Based Approach,” Feb. 23, 2022, *arXiv*: arXiv:2202.11768. doi: 10.48550/arXiv.2202.11768.
- [5] G. Vargas-Solar, M. H. F. Alves, and A.-L. M. Forst, “From Text to Knowledge with Graphs: modelling, querying and exploiting textual content,” Oct. 09, 2023, *arXiv*: arXiv:2310.06122. doi: 10.48550/arXiv.2310.06122.
- [6] Y. Lairgi, L. Moncla, R. Cazabet, K. Benabdeslem, and P. Cléau, “iText2KG: Incremental Knowledge Graphs Construction Using Large Language Models,” Sep. 05, 2024, *arXiv*: arXiv:2409.03284. doi: 10.48550/arXiv.2409.03284.
- [7] I. Melnyk, P. Dognin, and P. Das, “Knowledge Graph Generation From Text,” Nov. 18, 2022, *arXiv*: arXiv:2211.10511. doi: 10.48550/arXiv.2211.10511.
- [8] Z. Chen, Y. Wan, Y. Liu, and A. Valera-Medina, “A knowledge graph-supported information fusion approach for multi-faceted conceptual modelling,” *Inf. Fusion*, vol. 101, p. 101985, Jan. 2024, doi: 10.1016/j.inffus.2023.101985.
- [9] V. Emonet, J. Bolleman, S. Duvaud, T. M. de Farias, and A. C. Sima, “LLM-based SPARQL Query Generation from Natural Language over Federated Knowledge Graphs,” Oct. 21, 2024, *arXiv*: arXiv:2410.06062. doi: 10.48550/arXiv.2410.06062.
- [10] “Abdelhakim-gh/PA\_information\_to\_knowledge\_graph.” Accessed: Dec. 27, 2024. [Online]. Available: [https://github.com/Abdelhakim-gh/PA\\_information\\_to\\_knowledge\\_graph](https://github.com/Abdelhakim-gh/PA_information_to_knowledge_graph)