Graphes avec Neo4J

Faites une rapide présentation de Neo4J et de Cypher, quels sont les avantages/inconvénients, quels sont les cas d’usage (listez des données pouvant être organisées en graphes).

Définition d’un graphe, quelles sont les propriétés d’un graphe ?

Définissez :

* Graphes orientés vs non orientés.
* Graphes pondérés.
* Degré d’un sommet.
* Chemins et longueur de chemin.
* Graphe cyclique vs graphe acyclique.

Types de graphes utiles en text mining

* Graphe de cooccurrence.
* Graphe syntaxique.
* Graphe de connaissances .

## Activité pratique :

A partir du premier texte fourni, récupérer un graphe de relations grammaticales (comme vu dans le notebook) et insérer ces données dans une base Neo4J.

Pour s'approprier rapidement la librairie spacy, utilisez le notebook fourni (demo\_spacy.ipynb)

Voici les clauses Neo4J qui vous seront utiles : MATCH + WHERE + RETURN + CREATE + MERGE + SET  
  
Faites un bref laïus sur chacune d’entre elles (détaillez son utilité + donnez un exemple)

### Répondez aux questions métier suivantes :

* Lister tous les verbes du corpus

→ Identifier les Token dont pos = "VERB".

* Trouver les sujets d’un verbe donné

Exemple : pour le verbe “manger”, afficher les tokens reliés par :DEP {label:"nsubj"}.

* Trouver les objets directs associés aux verbes

→ relations :DEP {label:"dobj"}.

* Compter combien de fois chaque dépendance grammaticale apparaît
* Construire des couples Sujet–Verbe–Objet

Exemple : ("chat" - "mange" - "souris").

* Lister les adjectifs qui qualifient un nom donné : esprit, oiseau, train
* Mesurer les verbes les plus utilisés dans le corpus (avec une requête Cypher)

Contraintes :

N’utilisez que le SDK Python pour neo4j, appelé neo4j

Vous n’avez pas besoin de créer de fichier temporaire.

## Objectifs pédagogiques :

* Comprendre le modèle de graphe (nœuds, arêtes, orientation, propriétés).
* Extraire avec spaCy les tokens, lemmes, POS et dépendances grammaticales.
* Interroger le graphe avec Cypher (patterns, chemins, agrégations).
* (Bonus) Construire un graphe de cooccurrence et mettre à jour la base

## Planning proposé

Jour 1 : Bases théoriques, recherches et restitution sur slides

Jour 2 : Prise en main de spaCy + extraction des informations, résumer le graphe dans un array numpy.

Jour 3 - 4 : Prise en main de Neo4J, installation Docker ou utilisation cluster gratuit en ligne

Prise en main de Cypher (langage de Neo4J), insertion des données en base + réponse aux questions métier

Jour 5 - Réponse aux questions métier + Présentation des slides

## Modalités

Travail en groupe de 3 à 4 personnes, livrable : slides collectifs

## Bonus (dans l’ordre) :

1. Calculez le graph de cooccurrence sous forme d’une matrice à partir du premier texte, importez ce graph dans neo4j puis mettez à jour ce graphe de cooccurrence depuis le deuxième texte (il y a des mots en commun entre ces deux paragraphes vous vous en doutez) et mettez à jour la base neo4j.
2. Faites des recherches sur Neo4j Graph Data Science (GDS Library) et trouve runa algorithme de recherche de chemin le plus court, faites-en la description et expliquer comment l’utiliser avec cette librairie officielle

## Ressources :

<https://neo4j.com/docs/getting-started/appendix/tutorials/guide-cypher-basics/>

<https://neo4j.com/docs/getting-started/appendix/tutorials/tutorials-overview/>

<https://spacy.io/usage/>

<https://spacy.io/usage/visualizers#dep>

https://spacy.io/usage/linguistic-feature