Utilisation de graphes de connaissances pour explorer un catalogue muséographique

Léo ORVEILLON

Table des matières

Objectif du projet

Le but de ce projet est d'utiliser un graphe de connaissances regroupant une liste de tableaux, afin de créer une exposition cohérente et adaptative dans un environnement 3D. Le sujet devait s'articuler autour de deux parties, la première étant l'utilisation d'un avis « expert » (entendez algorithmie de graphe et machine learning) afin d'avoir une répartition des œuvres cohérentes entre elles. Et une deuxième partie, rendant le cheminement de l'exposition évolutive, en fonction de l'intérêt que l'utilisateur porte à certains tableaux. Seulement la première partie a été traitée, par manque de temps.

Concepts théoriques

La majeure partie de ce projet s'articule autour d'une structure de données particulière appelée : graphe de connaissances (Knowledge Graph) Figure 1. Ces graphes ont la particularité, de par leurs simplicités, d'offrir une grande flexibilité. Ils sont constitués de la manière suivante :

- Des « nodes » (nœuds) représentant chacun, un concept ou une entité, chaque nœud peut contenir plusieurs méta-données indépendamment des autres.
- Des « edges » (liens), chaque lien reliant 2 nœuds entre eux, en précisant le contexte du lien. Par exemple, le lien entre un tableau et son peintre sera : « peint par »

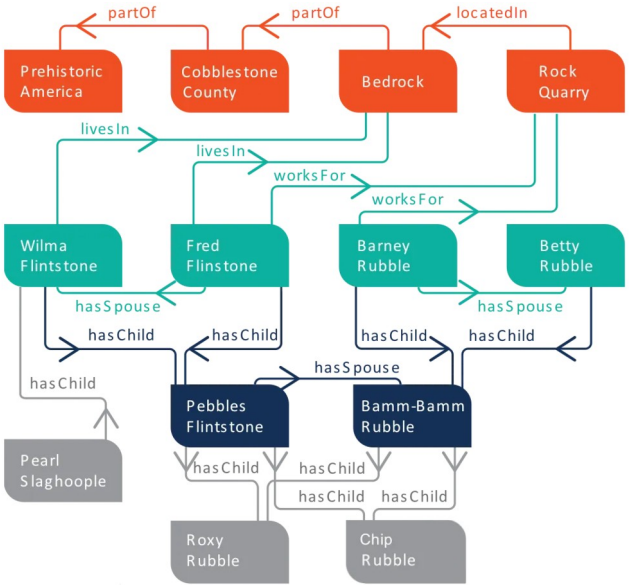

Figure 1: Exemple de graphe de connaissances

Une fois le graphe établit, l'objectif est d'utiliser différents algorithmes, afin de créer un environnement évolutif cohérent, disposant les tableaux « similaire » dans une même pièce, et en ayant des pièces qui s'enchaînent en fonction de groupe de tableaux « proches ».

Pour la deuxième partie, l'idée aurait été de pondérer les différents nœuds et liens du graphe, en fonction de l'intérêt que l'utilisateur porte à certains tableaux. Cela aurait pu, par exemple, permettre de faire ressortir certains concepts en premier, et donc de modifier l'ordre de l'exposition basée seulement sur une expertise afin d'y inclure les préférences de l'utilisateur.

Mise en œuvre

1 – Environnement 3D

Pour l'environnement 3D, nous partons d'un projet d'IEVA proposant une représentation de musée en 3D Figure 2. Celui-ci utilise Python avec la bibliothèque Flask permettant de créer des applications web, le client web utilise ensuite Babylon, une bibliothèque utilisée pour créer des environnements 3D dans un navigateur en Javascript.

Il a donc été nécessaire de trier et supprimer les parties de code du projet d'IEVA n'étant pas utile ici. Cette partie du code sert seulement à l'affichage de nos tableaux dans les salles, la liste et l'ordre des tableaux à afficher est calculé en amont par un autre programme.

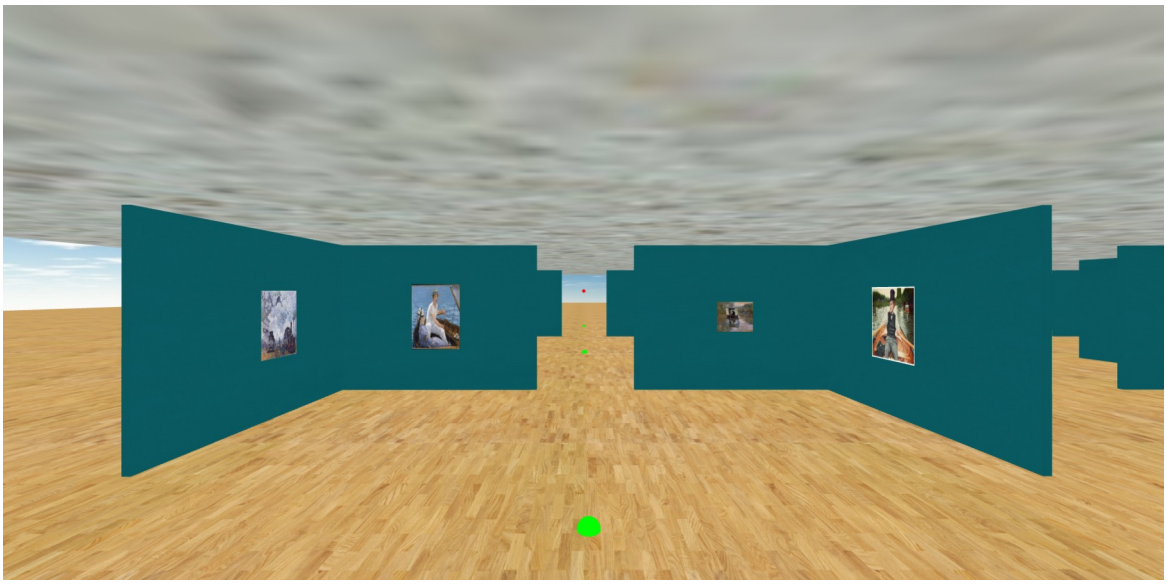

Figure 2: Musée 3D

2 – Construction du graphe

Pour constituer notre graphe, nous utilisons deux fichiers csv distincts :

- Le premier, contenant la liste des tableaux ainsi que leurs méta-données permettant de construire un nœud par tableau. Figure 3

- Le deuxième fichier, de son côté est une liste de triplets RDF (Resource Description Framework), chaque triplet représentant deux nœuds du graphe ainsi que le lien les reliant. Figure 4

```csv
id,nom,peintre,date,largeur,longueur
MAN001,Olympia,Manet,1863,130,192
MAN002,Le déjeuner sur l'herbe,Manet,1863,207,265
MON001,Le train dans la neige,Monet,1875,81,116
MON002,La gare saint-Lazare,Monet,1877,75,104
MON003,Le déjeuner sur l'herbe,Monet,1865,65,54
CEZ001,Une moderne Olympia,Cézanne,1873,46,55
TIT001,La Vénus d'Urbin,Titien,1538,124,180
GIO001,La Vénus endormie,Giorgione,1511,65,54
CAI001,La partie de bateau,Caillebotte,1877,89,116
MAN003,En bateau,Manet,1874,97,130
MON004,Le bateau atelier,Monet,1876,72,60
MON005,Régates à Argenteuil,Monet,1872,48,75
```

Figure 3: CSV tableaux

```rdf
head,relation,tail
Manet,is,peintre
MAN001,is,tableau
MAN001,isNamed,Olympia
MAN001,paintedBy,Manet
MAN001,is,nu
MAN001,datesFrom,1863
MAN001,represents,F01
F01,is,femme
F01,hasAsModel,Victorine Meurent
```

Figure 4: Triplets RDF

Ces fichiers sont chargés en graphe en utilisant les bibliothèques pandas et networkx. Une fois le graphe créé, celui-ci contenant peu d'informations, il est possible de le visualiser en choisissant le bon agencement Figure 5. Cet agencement n'influe pas sur les calculs, mais nous permet de visualiser nos données afin de vérifier de manière graphique que nos algorithmes organisent bien nos tableaux d'une manière cohérente.

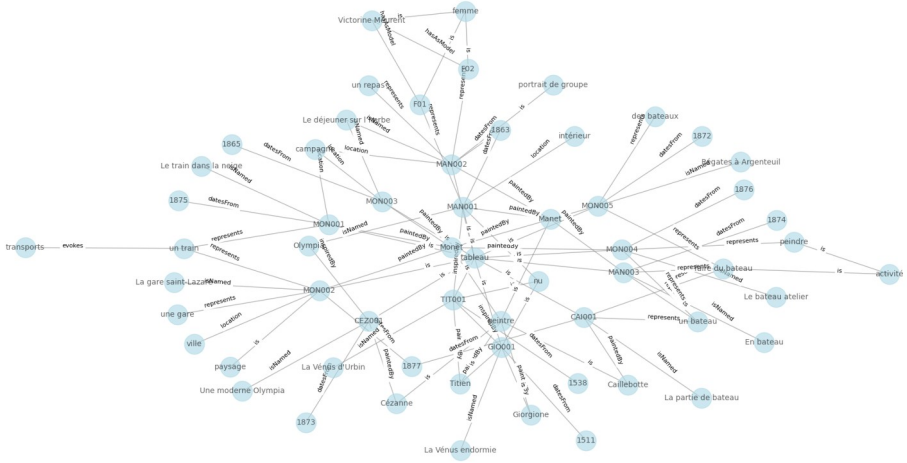

Figure 5: Représentation du graphe selon l'agencement Kamada-Kawai

3 – Calculs sur le graphe

Maintenant que notre environnement et notre graphe sont prêts, il nous faut définir une méthode de calculs pour disposer nos tableaux. La méthode proposée suit deux étapes :

- Création de « clusters » regroupant les tableaux.
    - Le but ici est de grouper les tableaux entre eux de manière à avoir une disposition de tableaux pour chaque salle, pour se faire on « vectorisera » notre graphe afin de pouvoir lui appliquer des algorithmes de machine learning.
    - Deux algorithmes ont été étudiés, le DBSCAN et le Kmeans. Le Kmeans détermine les clusters en fonction d'un nombre donné alors que le DBSCAN de son côté, détermine lui-même le nombre de clusters à extraire. Figure 7
    - Le Kmeans a été gardé, car il nous permet de choisir le nombre de clusters que l'on désire, et donc dans notre cas le nombre de salles que l'on souhaite. A contrario, si un tableau partage peu de concepts avec les autres, le DBSCAN lui attribuera un cluster pour lui seul, cela créant ainsi des salles peuplées d'un seul tableau. Figure 6

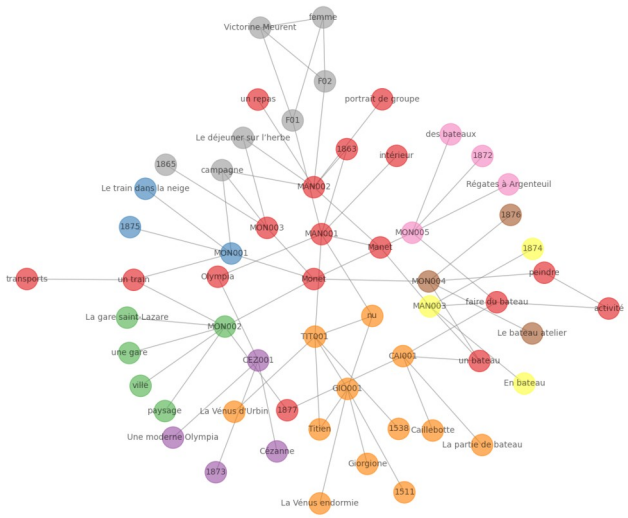

Figure 7: Groupements selon DBSCAN

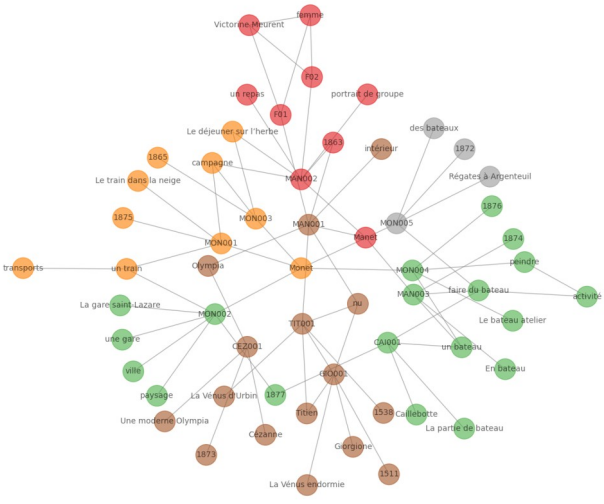

Figure 6: Groupements selon K-Means (5 Clusters)

- Création d’un parcours
    - On souhaite déterminer l’ordre de passage de nos clusters. Pour se faire, on utilise des algorithmes de parcours de graphe afin de trouver les deux tableaux les plus éloignés l’un de l’autre. (Pour avoir un point de départ et de fin)
    - Une fois notre nœud de départ trouver, on ordonne nos tableaux par ordre de proximité en cherchant le plus court chemin entre notre nœud actuel et tous les nœuds restant.

Une fois notre ordre de tableaux définit, on en déduit notre ordre de cluster en fonction de l’appartenance des tableaux aux clusters.

Annexe – Utilisation du code

Liste des étapes à suivre pour faire fonctionner le code fournit :

1. Télécharger et décompresser l'archive
2. Installation des dépendances nécessaire avec pip
    - > pip install -r requirements.txt
3. Lancement du serveur Flask
    - > cd serveur/
    - > python serveur.py
4. Ouvrir le fichier client/index.html avec votre navigateur
5. Vous pouvez désormais vous déplacer dans l'environnement avec les flèches directionnelles, quand vous entrez dans une nouvelle salle, les tableaux devraient apparaître suivant les clusters déterminés précédemment.

Références

- Figure 1 : https://ontotext.medium.com/knowledge-graphs-breaking-the-ice-2fc11e6de48b
- Code pour l'affichage de graphe ainsi que l'utilisation des algorithmes de machine learning : https://lopezyse.medium.com/knowledge-graphs-from-scratch-with-python-f3c2a05914cc
- Documentation de networkx : https://networkx.org/documentation/stable/