

TP Advanced Data Mining

Graphes et Applications

JY Ramel – 2019

Le compte-rendu de TP (fichier zip avec sources html et js et fichier contenant les réponses aux questions) à renvoyer par email à ramel@univ-tours.fr ou déposer sur Celene avant le 21 octobre (bien préciser les noms constituant le binôme)

1. Données étudiées

Ce TP va vous permettre d'exploiter une suite logicielle intitulée **Linkurious**¹ pour développer une petite application web permettant de visualiser, manipuler et analyser des données stockées sous forme de graphe.

Le jeu de données utilisé est 'Marvel Universe'² représentant les personnages fictifs de l'univers de Marvel ainsi que les comics dans lesquels ces personnages apparaissent.

Les données originales sont composées de :

- une liste de 6486 personnages
- une liste de 12 942 comics
- une liste de liens (personnage, comic) indiquant la présence d'un personnage dans un comic donné.

Question 1. Si on considère un graphe G pour lequel :

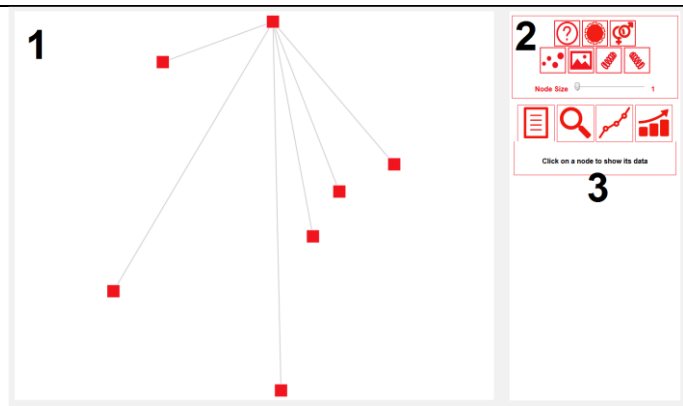
1. les nœuds sont les personnages et les comics
2. les arêtes sont les liens (personnage, comic)

A quoi ressemble ce graphe ? Dessiner le grossièrement. Comment qualifie-t-on ce type de graphe ?

Comment transformer G pour créer un graphe type 'réseau social' à partir de ce jeu de données ? Appelons ce nouveau graphe G*, quel poids peut-on associer aux arêtes ?

2. Prise en main du site MUGEN : Marvel Universe Graph Exploration

Pour visualiser les graphes, nous allons utiliser, développer une plateforme web, MUGEN (squelette à télécharger sur Celene), qui s'appuie sur la librairie JavaScript Ogma³ associée à Linkurious. La Figure 1 présente l'interface de cette plateforme web.

 <p>Figure 1 : Interface de MUGEN</p>	<p>L'interface est constituée des régions suivantes :</p> <ol style="list-style-type: none">1. correspond au canvas ou le graphe sera dessiné2. correspond aux boutons de contrôle. Dans le TP, vous aurez à coder des fonctions javascript associées à ces boutons3. correspond à un espace où des informations seront affichées (images, plus court chemin, statistiques, etc.).
--	--

¹ <https://linkurio.us/>

² Alberich et al. 'Marvel Universe looks almost like a real social network' (2002)


³ <https://doc.linkurio.us/ogma/latest/>

Parcourez les sources de cette plateforme et la documentation Linkurious. Pour répondre à la plupart des questions, il sera nécessaire / possible de vous inspirer du code fourni pour produire le code manquant. Cela vous permettra, par exemple de voir comment :


- Parcourir les nœuds et les arcs du graphe chargés et accéder à leurs propriétés (`sig.graph.edges().forEach(function(e)`)
- Associer des événements aux actions sur le graphe affiché (`function initListener()`)
- Modifier le code HTML de la page affichée depuis le code javascript (`document.getElementById("showHideImagesImage").src = "images/hide_image.png";`)


Question 2. Par défaut, le site charge le graphe `xxxxxxxx.json`. Modifier `mugen.js` pour que le graphe chargé par défaut soit le graphe stocké dans le fichier `mu_128.json`.


Question 3. Dans `help.html`, compléter la section `About the students`. Rajouter au minimum vos noms, prénoms, numéros étudiant.

Question 4. Dans `index.html`, associer comme événement sur le bouton  la méthode JavaScript `filLayout()` qui est déjà implémentée.

3. Ajout de fonctionnalités de visualisation et d'exploration

Question 5. Compléter la fonction JavaScript `changeNodeStyles()` qui change la forme et la couleur de tous les nœuds (`type= "circle"` et `color="#000"`). Cette fonction est associée au bouton .

Question 6. Compléter la fonction JavaScript `showHideEdges()` qui permet d'afficher ou non les arêtes. Cette fonction est associée au bouton . Quel peut-être l'intérêt d'une telle fonctionnalité ?

Question 7. Ecrire la fonction JavaScript `maleFemaleHighlight()` qui colorie les nœuds du graphe selon le critère homme/femme. Cette fonction est associée au bouton .

Question 8. Compléter la fonction JavaScript `getNeighbours()` utilisé dans `selectNode()` pour afficher des informations sur un nœud dans le panneau `nodeTabContent` de droite. Les informations à afficher sont les vignettes des voisins du nœud courant. Un clic sur une de ces vignettes correspond à la sélection d'un nouveau nœud d'intérêt, et donc à la mise à jour dans le graphe et dans l'onglet `Current node`.

5. Ajout de fonctionnalités de fouille de graphe

Question 9. Nous souhaitons étudier les héros les plus "importants". Pour ce faire, on choisit de ne garder que les héros qui ont de nombreuses connexions (arcs de poids supérieur ou égal à 50). Ecrire le pseudocode de l'algorithme qui effectuerait ce premier filtrage des arcs.

Question 10. Pour filtrer le graphe, écrire la fonction JavaScript `pruneGraph()` qui supprime les arêtes dont le poids est inférieur à un paramètre global `MIN_EDGE_WEIGHT`. Faire appel à cette fonction dans la méthode `on load()` avant l'affichage du graphe.

Aide : (`Suppression d'arc = sig.graph.dropEdge(edge.id);`)

Question 11. Avec `MIN_EDGE_WEIGHT = 50`, lancer l'algorithme de dessin de graphe Fruchterman-Reingold. Que remarquez-vous ? Compléter la fonction `pruneGraph()` pour éventuellement supprimer les éléments en trop.

Question 12. Regarder et expliquer la fonction JavaScript ``hitsAlgorithm()`` qui fait appel à l'algorithme HITS 1, déjà implémenté. Qui sont, dans l'ordre, les 5 personnages les plus importants, selon le critère HITS ?

Aide : https://en.wikipedia.org/wiki/HITS_algorithm

Question 13. En vous basant sur l'algorithme précédent, compléter la fonction JavaScript ``highestDegreeNodes()`` qui colorie en rouge les personnages qui ont le plus de liens. Le reste des nœuds sera colorie en noir. Qui sont, dans l'ordre, les 10 personnages les plus importants, selon le critère des degrés élevés ?

Question 14. Regarder et expliquer (partie 1 et partie 2) la fonction JavaScript ``louvainClustering()`` qui fait appel à l'algorithme de Louvain, déjà implémenté.

Aide : <https://github.com/Linkurious/linkurious.js/tree/linkurious-version/plugins/sigma.statistics.louvain>

Question 15. Proposer le pseudo-code d'un algorithme et d'une signature de nœuds qui permettrait de mesurer une similarité ou dissimilarité entre **2 nœuds** dans le graphe. Justifier et expliquer l'intérêt d'une telle fonction.

Question 16. Proposer le pseudo-code d'un algorithme qui permettrait de mesurer une similarité ou dissimilarité **entre 2 sous-graphes** en renvoyant éventuellement aussi un appariement (**matching**) entre les éléments des 2 sous-graphes. Justifier et expliquer l'intérêt d'une telle fonction.

Questions optionnelles

Question 17. Regarder et expliquer la fonction JavaScript ``computeDensities()`` qui calcule les densités intra- et inter-cluster d'un partitionnement donnée. Indiquer les expressions mathématiques des différentes valeurs intéressantes calculées par l'algorithmes. Quelles valeurs obtient-on pour le partitionnement avec l'algorithme de Louvain ?

Question 18. Ecrire la fonction JavaScript ``shortestPath()`` qui permet de trouver le plus court chemin entre deux nœuds. On affichera la longueur du chemin et le chemin dans l'onglet ``shortestPathTabContent``.

Aide : <https://github.com/Linkurious/linkurious.js/tree/linkurious-version/plugins/sigma.pathfinding.astar>

Question 19. Compléter la fonction JavaScript ``newmanGirvanClustering()`` qui implémente l'algorithme de partitionnement de Newman-Girvan.

Aide : <https://github.com/Linkurious/linkurious.js/blob/linkurious-version/examples/plugin-pathfinding-astar.html>

Hypothèses :

- Vous utiliserez une approche 'simple' pour calculer la betweenness centrality,
- Vous contrôlerez l'arrêt de l'algorithme avec MAX ITERATIONS, paramètre global, et non pas avec la modularité comme propose dans l'algorithme original.
- Après la 1ere itération, quelle est l'arête qui doit être supprimée ? Quels sont les nœuds associés ? Cela vous paraît-il cohérent ?
- Recalculer les valeurs des densités intra- et inter-cluster pour le partitionnement obtenu avec l'algorithme de Newman-Girvan.