Méthodes big data pour l'analyse de très grands graphes de protéines

Contexte :

On dispose d'une base de données qui représente un graphe de similarité entre protéines. Un nœud est un identifiant de protéine, un arc relie deux protéines avec un poids de similarité (allant de 80% à 100% de similarité). On peut aussi connaitre, pour un petit sous ensemble des protéines, leurs propriétés fonctionnelles décrites par une liste de labels. Ces annotations fonctionnelles sont incomplètes car la plupart des protéines ne sont pas annotées. Les annotations ne se réfèrent pas toutes à un unique vocabulaire, en conséquence des protéines très similaires ayant la même propriété fonctionnelle peuvent avoir des labels différents.

De plus, les jeux de données à analyser sont très volumineux avec un graphe contenant plus de 20 milliards d'arcs. La taille pose un défi pour l'analyse car les outils existants nécessitant de charger les données en mémoire ne peuvent pas traiter des graphes aussi grands que celui étudié. Par ailleurs, les plateformes distribuées sur plusieurs machines dédiées à l'analyse de données à grande échelle ne sont pas efficaces pour exécuter les analyses envisagées dans ce stage : on peut constater que la quantité de données échangée entre les machines est grande et que cela ralentit fortement les analyses.

Objectif

Deux types d'analyse sont envisagés :

- Prédiction de label : Attribuer un label aux protéines qui n'en ont pas,
- Fusion de label : Déterminer quels labels représentent très probablement la même propriété fonctionnelle et peuvent être fusionnés.

L'objectif du PLDAC est de proposer une solution efficace pour effectuer ces analyses sur un très grand graphe de protéines.

Travail à faire

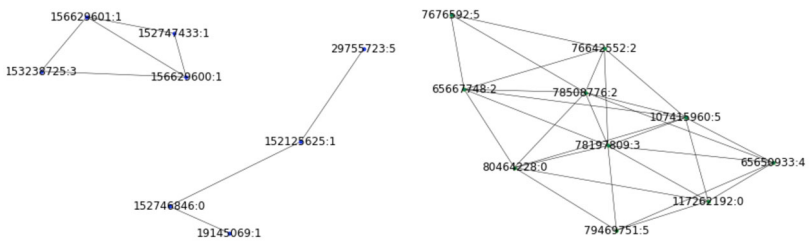
Etape 1 : Calcul parallèle des composantes connexes

Pour faire face à la très grande taille des données, l'approche proposée consiste à réduire le graphe en combinant deux méthodes :

- Appliquer un seuillage sur le poids des arcs,
- Détecter les composantes connexes du graphe pour les analyser séparément.

Commencer par étudier les solutions implantées dans MadLib [1] GraphX [2] et GraphFrames[3] pour calculer les composantes connexes d'un graphe. Expliquer quelles sont leurs limites pour le cas d'usage considéré.

Réduire la taille des données en ne considérant que les protéines 100% similaires avec les autres. On obtient un premier graphe appelé G100 contenant seulement les arcs dont le poids de similarité vaut 100%. Proposer une méthode de calcul parallèle pour déterminer les composantes connexes de ce graphe. Le résultat attendu contient par exemple les composantes suivantes : deux composantes avec 4 nœuds et une composante de 10 nœuds.



Etape 2 : Prédiction de label

On suppose connu le label (*i.e.* l'annotation fonctionnelle) de certains nœuds. A partir des labels connus dans chaque composante, proposer une méthode permettant d'attribuer un label aux nœuds sans label. Définir le taux de confiance pouvant être accordé à l'attribution d'un label.

Etape 3 : Calcul hiérarchique incrémental des composantes connexes

L'objectif de cette étape est d'étendre la solution proposée à l'étape 1 pour pouvoir traiter efficacement la totalité du graphe. Il s'agit de concevoir une méthode de **calcul hiérarchique et incrémental** des composantes connexes d'un graphe. Proposer une approche hiérarchique basée sur une partition du graphe en fonction du poids des arcs. Il est notamment possible de définir le graphe G(i) contenant seulement les arcs dont le poids est supérieur à i. L'idée est de calculer de manière incrémentale les composantes des graphes G(i) successifs en réutilisant les composantes déjà calculées.

Etape 4 : Validation expérimentale

Mesurer le gain en performance de la solution proposée à la fois pour le calcul des composantes connexes et pour la prédiction de labels. Proposer une méthode pour déterminer les paramètres optimaux (seuil, degré de parallélisme) en fonction des ressources mémoire et CPU dont on dispose.

Références

[1] The MADlib Analytics Library or MAD Skills, the SQL, VLDB 2012, https://vldb.org/pvldb/vol5/p1700_joehellerstein_vldb2012.pdf et madlib.apache.org

[2] Apache Spark GraphX https://spark.apache.org/graphx/

[3] GraphFrames : https://graphframes.github.io/graphframes/docs/_site/user-guide.html et https://cedric.cnam.fr/vertigo/Cours/RCP216/tpCheminsCentralites.html