Projet : intégration de données bio-informatiques (4 séances)

UCBL - Bases de données pour la bioinformatique - 2017 / 2018

Objectif du projet : intégrer des données de plusieurs sources pour confirmer des résultats scientifiques

Il s'agit d'étudier les interactions entre les protéines humaines et celles du virus Epstein-Barr (EBV). Ce projet s'appuie sur un article de 2007[1] et l'intégration de données dans ce projet permettra de vérifier certaines découvertes mentionnées dans cet article et de procéder à des analyses supplémentaires.
La figure 1 fournit un aperçu de l'intégration des différentes sources. Les informations (les plus utiles) que vous pouvez extraire d'une source sont indiquées en jaune, et les mappings nécessaires pour "lier" des informations entre deux sources sont indiqués en bleu.

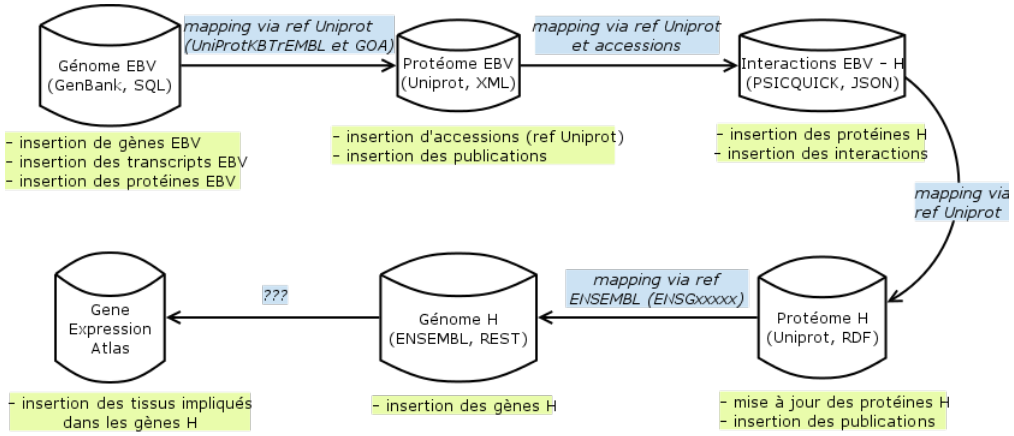

Fig. 1: Aperçu des aspects intégration du projet

1 Conditions de réalisation du projet

Le projet est noté sous forme de bonus/malus sur la note de l'examen. Vous pouvez travailler seul.e ou en binôme. Il est à rendre sur Tomuss (colonne *depotProjet*) en fin de dernière séance : vous déposerez uniquement un fichier texte (dump .sql ou .txt) de votre base de données (schéma + données). Pour l'évaluation, ce dump sera chargé avec le SGBD SQLite. Il n'y a aucune contrainte sur les aspects techniques (i.e., vous pouvez utiliser les SGBD que vous souhaitez, écrire les mappings dans les langages qui vous conviennent, etc.). Mais les conseils et pointeurs sont donnés pour le langage Python.

2 Méthodologie

Il est conseillé de lire le sujet complet avant de démarrer.

2.1 Documents à télécharger

- Présentation sur le virus EBS :
  http://liris.cnrs.fr/fabien.duchateau/ens/BDBIO/tp/presentation-virus-EBV.pdf
- Ficher SQL du TP1 (schéma) : c'est le votre, généré à partir de JMerise

1. Calderwood, Michael A. and Venkatesan, Kavitha and Xing, Li and Chase, Michael R. and Vazquez, Alexei and Holthaus, Amy M. and Ewence, Alexandra E. and Li, Ning and Hirozane-Kishikawa, Tomoko and Hill, David E. and Vidal, Marc and Kieff, Elliott and Johannsen, Eric, "Epstein–Barr virus and virus human protein interaction maps", Proceedings of the National Academy of Sciences (2007), pmid :17446270

- Wrappers en Python (pour SQLite, XML et XPath, JSON, et RDF) et les quatre sources de données : http://liris.cnrs.fr/~fduchate/ens/BDBIO/tp/wrappers-python.zip

2.2 Architecture

Les sources de données et le schéma global sont très hétérogènes (SQL, XML, JSON, RDF). Cependant les sources se recoupent assez peu quant aux éléments qu'elles décrivent (i.e., une source décrit des protéines, une autre des interactions, etc.). Parmi ces sources, il y a 4 "dumps" (fichier contenant des données) et 1 source à interroger en ligne. L'architecture la plus adaptée est l'entrepôt (ou centralisation des données dans un seul SGBD). Cela facilitera l'interrogation (pas besoin de créer des adaptateurs et d'attendre les réponses des sources, ni de devoir décomposer les requêtes et recomposer les résultats), d'autant que la majorité des sources sont des "dumps" (pas de mises à jour).

Le schéma créé lors du TP1 servira de schéma pour l'entrepôt (ou BD globale). Concrètement, vous utiliserez un langage pivot comme Python pour interroger une source (e.g., requêtes XQuery), puis traiter les données récupérées pour peupler votre BD globale (e.g., requêtes d'insertion SQL). Vous devrez probablement modifier votre schéma global et vous ferez probablement des insertions erronées ou incomplètes : il est donc conseillé d'écrire un programme qui, à chaque exécution, supprime et recrée la base de données de l'entrepôt à partir d'un script SQL, puis intègre progressivement chaque source.

3 Création du schéma de votre entrepôt

Vous disposez d'un script SQL pour créer le schéma de votre entrepôt dans SQLite. Python possède une API pour faciliter la connexion à SQLite[2]. La méthode `executescript()` permet notamment d'exécuter un script SQL, et vous pouvez donc programmer la création du schéma de votre entrepôt en Python.

4 Intégration des gènes EBV (source GenBank, format SQL)

La première source est un fichier SQL (pour SQLite) qui contient les informations sur les gènes du virus EBV. Cette source n'est pas correctement modélisée, et quelques problèmes peuvent donc survenir (une solution rapide est de perdre quelques informations). Elle vous permet de peupler votre entrepôt avec les informations sur les gènes, les transcripts et les protéines.

Pour créer les mappings, vous devez ouvrir une connexion vers cette BD du TP2, l'interroger avec une ou plusieurs requêtes SQL. Les résultats retournés seront éventuellement transformés et nettoyés, puis seront utilisés pour construire des requêtes SQL de type `INSERT` qui seront exécutées sur votre entrepôt pour le peupler. Vérifiez les données insérées avec le SGBD SQLiteStudio.

5 Intégration des protéines EBV (source Uniprot, format XML)

La seconde source est un fichier Uniprot au format XML qui contient les informations sur les protéines du virus EBV. Pour manipuler un fichier XML, le langage Python dispose de plusieurs modules, dont *etree*[3].

Dans ce fichier XML, vous pouvez compléter les informations sur les protéines en extrayant :

- la liste des accessions (références Uniprot données successivement à la protéine).
- la masse de la séquence (attribut `mass` de la balise `<sequence>`) et sa longueur (attribut `length`)

6 Intégration des interactions EBV - humain (source IntAct, format JSON)

Cette troisième source est un fichier au format JSON qui décrit les interactions entre des protéines de EBV et des protéines humaines. Pour manipuler un fichier JSON, le langage Python dispose d'un module spécifique[4].

2. Documentation Python pour SQLite, http://docs.python.org/3/library/sqlite3.html
3. Documentation Python pour XML, http://docs.python.org/3/library/xml.etree.elementtree.html
4. Documentation Python pour JSON, http://docs.python.org/3/library/json.html

Dans ce fichier, chaque sous-document du champ `data` représente une interaction, pour laquelle vous devez extraire les informations suivantes :

- un identifiant d'interaction, qu'il faudra fabriquer ou générer.
- les protéines impliquées dans cette interaction.
- la méthode de détection utilisée (champ `interactionDetectionMethod`, qui contient à la fois l'identifiant *psi:mi* suivi du nom de la méthode entre parenthèses).

Une protéine qui n'existe pas encore dans votre entrepôt doit évidemment être insérée avant d'ajouter l'interaction dans laquelle elle participe. Ces nouvelles protéines contiendront peu d'informations (une référence Uniprot, et un identifiant obligatoire à générer), mais la source suivante permettra de les compléter.

7 Intégration des protéines humaines (source Uniprot, format RDF)

Cette quatrième source est un fichier au format RDF contenant des informations sur les protéines humaines. Pour manipuler du RDF, le langage Python inclut le module RDFlib, qui permet également de requêter les données avec SPARQL[5]).

Avec des triplets, il faut reconstruire une information complète (par exemple, sur une protéine). Complétez votre table PROTÉINES avec les informations manquantes pour les protéines humaines, et notamment l'identifiant ENSEMBL du gène (prédicat `rdfs:seeAlso` avec un objet contenant *ENSG*). Attention, certaines protéines humaines issues du fichier d'interactions utilisent des accessions obsolètes !

8 Intégration des gènes humains (source ENSEMBL en REST)

Cette source est la BD ENSEMBL contenant des informations sur les protéines humaines. Cette base sera interrogée directement via une API REST décrite ici : http://rest.ensembl.org/

Les protéines humaines possèdent un identifiant ENSEMBL suite à l'intégration de la source précédente. Une façon assez simple de retrouver les informations sur ENSEMBL est de parcourir chaque identifiant ENSEMBL, et de construire une requête REST (méthode GET) avec les bons paramètres. Par exemple, pour récupérer au format JSON les identifiants vers des bases de données externes pour un identifiant ENSEMBL donné, on pourra envoyer la requête suivante[6] :
http://rest.ensembl.org/xrefs/id/ENSG00000157764?content-type=application/json

Pour chaque protéine humaine, on peut récupérer les informations suivantes sur ENSEMBL :

- chaque transcript avec son identifiant (champ `transcriptID`), son nom (champ `logic_name`), sa longueur (champ `length`). La ressource `GET overlap`[7] permet de retrouver la liste des transcripts (leurs identifiants ENSEMBL) pour un ENSEMBL ID de protéine. Ensuite, vous pouvez utiliser la méthode `GET lookup`[8] avec en paramètre un ENSEMBL ID de transcript pour obtenir les informations demandées

9 Intégration des tissus humains (source GeneAtlas)

Si vous êtes parvenu.e.s jusqu'ici, déjà bravo ! Cette dernière source, Gene Expression Atlas, permet de lister les tissus humains qui sont impliqués par les gènes : http://www.ebi.ac.uk/gxa/home

Vous êtes complètement libres sur la façon d'intégrer ces nouvelles données. Les identifiants ENSEMBL sont également utilisés par Gene Expression Atlas, et (donnée `target_protein_gene_expression profile`). Le dump fourni par Atlas est très volumineux (50 Go), il est donc plus judicieux de construire une URL avec chaque identifiant de gène ENSEMBL.

5. Documentation Python pour RDF, http://pypi.python.org/pypi/rdflib, urlhttp://rdflib.readthedocs.io/en/latest/
6. Documentation ENSEMBL `GET xrefs`, avec des exemples de code dans différents langages comme Python, http://rest.ensembl.org/documentation/info/xref_id
7. Documentation ENSEMBL GET overlap, http://rest.ensembl.org/documentation/info/overlap_id
8. Documentation ENSEMBL `GET lookup`, http://rest.ensembl.org/documentation/info/lookup