Université Claude Bernard Lyon 1

bioinfo@lyon
BIOINFORMATIQUE MOLÉCULAIRE: MÉTHODES & ANALYSES

Cahier des charges

Développement d'une application automatisant la recherche de jeux de données, production de statistiques et création d'une base de donnée

Projet 3

Porteurs du projet :

- Duchateau Fabien
- Launay Guillaume

Tuteur pédagogique :

- Lacroix Vincent

Étudiants :

- Thalamas Thibaut
- Moullé Pauline
- Koralewski Alexis

M1 Bioinformatique
2019-2020
Université de Lyon1

Sommaire

1 Présentation du projet

1.1 Contexte

Dans le cadre de l'UE "base de données pour la bioinformatique" enseignée en Master Bioinformatique, les étudiants réalisent un projet **d'intégration de données** de bio-informatique. L'objectif du TP consiste à vérifier les informations d'un **article scientifique** qui décrit des **interactions** entre protéines et de retrouver des informations supplémentaires (protéomes, génomes, tissus). Pour cela les étudiants doivent créer une **base de données** qu'ils peupleront grâce aux données de 5 fournisseurs. Le tableau 1 décrit les fournisseurs utilisés et les informations collectées. Les sources de données des fournisseurs ont des modèles et des **formats variés**, une organisation différente des concepts et des **incohérences** au niveau des valeurs, ce qui complique la tâche d'intégration. L'ensemble des sources de données relatif à un article est appelé ici **jeu de données.**

Actuellement, les enseignants de cette UE utilisent le même jeu de données avec des variations mineures car il est fastidieux de chercher manuellement les sources de données que les étudiants du master doivent intégrer. En effet, à partir de l'article scientifique, on doit idéalement retrouver des **informations** qui peuvent être entièrement ou partiellement vérifiées en interrogeant les 5 fournisseurs.[1]

L'an dernier, un projet de TER a été réalisé. Il a démontré qu'il est possible **d'automatiser** la récupération d'un jeu de données.

Tableau 1. Description détaillée du jeu de données

Fournisseur	Type d'information	Fichiers du jeu de données
Intact[2]	Interaction protéine-protéine	Fichier d'Interaction entre deux protéines d'espèces différentes
PubMed[3]	Article scientifique	Fichier associé à un article (auteurs, titre, résumé, etc.) qui décrit une interaction ou des protéines
Uniprot[4]	Protéique	Tous les fichiers associés aux protéines
Ensembl[5]	Génomique	Tous les fichiers associés aux gènes
GenBank[6]	Génomique	Tous les fichiers associés aux gènes non présents sur Ensembl

Malheureusement, le code créé lors de cette étude est **inutilisable** car non générique et les fonctionnalités sont trop **limitées**.

Un nouveau projet est donc proposé aux étudiants de M1 afin de développer une application de recherche de jeux de données et la production de statistiques. Les informations collectées et générées sont stockées dans une base de données.

1.2 Objectifs.

1.2.1 Objectifs principaux

Le premier objectif du projet est de créer une **application python** qui permet de générer en local un jeu de données, c'est à dire de récupérer un maximum de sources de données chez les fournisseurs mentionnés. Cela nécessite de détecter, à partir d'un ensemble de départ, les **correspondances** entre les sources de données (e.g, à partir d'un identifiant de protéine dans le fichier d'interactions, retrouver la protéine correspondance sur Uniprot). Cette information est importante car un jeu de données avec un faible nombre de correspondances entre ses sources sera peu formateur pour les étudiants, et n'aura donc que peu d'intérêt pour les utilisateurs.

Un second objectif concerne la création d'une base de données et elle contiendra au minimum :

- l'identifiant du jeu de données
- la date de création du jeu de données
- le chemin local vers chacune des sources composant le jeu de données
- Statistiques du jeu de donnée
    - La complétude du jeu de données : information générale sur les 5 fournisseurs, combien d'entre eux nous renvoie des fichiers de sortie.
    - Le nombre de correspondances entre chaque source de données et les correspondances elles-mêmes.
    - D'autres statistiques peuvent être calculées

Enfin, l'application donne la possibilité de récupérer un jeu de données grâce à son identifiant afin de transférer ses fichiers dans un répertoire différent, renseigné en paramètre.

1.2.2 Perspectives

Un système de **versionning** sera mis en place afin de gérer les évolutions dans le temps de données chez les différents fournisseurs questionnés (l'ajout ou la suppression de protéines, changement de format ou autre). La base donnée gardera en mémoire toutes les versions.

Certaines requêtes peuvent se faire en parallèle comme par exemple la recherche sur la base Uniprot pour les différentes protéines concernées par une interaction. On pourrait donc récupérer les informations simultanément pour réduire le temps de réponse et **optimiser** ainsi les **performances** du script.

Les fournisseurs proposant des formats hétérogènes, il serait utile que l'application dispose de **convertisseurs** entre formats, e.g. convertir un fichier xml en json.

Une **interface web** permettra d'interroger la base de données et de renvoyer à l'utilisateur les informations et statistiques des jeux de données.

1.3 Description de l'existant

1.3.1 Étude de faisabilité

Une étude de faisabilité réalisée par des étudiants de L3 en projet TER a montré que l'automatisation de la recherche de jeux de données est possible.
Cette étude est constituée, entre autres, d'un script python. Le script génère un jeu de données qui est sauvegardé en local.
Le script original ne sera pas utilisé mais sera **ré-implémenté** pour les besoins du projet, notamment parce que le stockage local des fichiers n'est pas optimal (nombreuses redondances).
Le rapport nous liste également les formats de sortie des différents fournisseurs : tsv, txt, fasta, xml, rdf, gb, json. La liste des formats conservés pour l'application est détaillée en section 1.3.3.

1.3.2 Logiciels

Afin de créer notre application, nous utiliserons le langage **Python**.
La manipulation des différents formats de sortie des sources sera facilitée par l'utilisation de packages python tels que ceux listés dans le Tableau 2.

Tableau 2: Packages utilisé pour le script python

Module	Utilisation
API REST, requests	Requête HTTP pour récupérer les fichiers provenant des différents fournisseurs
json[7] xml[8] rdflib [9]	Parser des fichiers json, xml et rdf
sqlite3[10]	Interagir avec une base de données SQL
docopt[11]	Gestion des paramètres en ligne de commande de l'application Python

La base de données conçue utilisera le SGBD SQLiteStudio[12].

1.3.3 Fournisseurs

Le jeu de données se base sur 5 fournisseurs différents, chaque fournisseur propose différents formats de fichier de sortie. Les formats du Tableau 3 sont ceux listés par les fournisseurs comme étant accessible par programmation.

Tableau 3: Liste des formats accessible par programmation

Fournisseur	Format des fichiers de sortie	
Intact [13]	● PSI-MITAB 2.5 à P2.8 ● PSI-MIXML 2.5.4 ● count ● biopax	● xgmml ● rdf-xml ● rdf-xml-abbrev ● rdf-n3 ● rdf-turtle
Pubmed[14]	● xml	● json
Uniprot[15]	● html ● xml ● tab ● xls ● fasta	● gff ● txt ● rdf ● list ● rss
Ensembl[16]	● json ● xml	● jsonp
Genbank[14]	● xml	● json

L'application essaiera de récupérer tous les formats ci-dessus s'ils sont disponibles lors de la requête.

1.4 Critères d'acceptabilité du produit

L'application Python sera acceptable si elle génère tous les fichiers en **local** dans une arborescence optimisée, c'est à dire, minimiser l'espace utilisé sur le système de fichier.
Elle doit aussi fournir la possibilité de **récupérer** le jeu de donnée souhaité.
La base de données, et en particulier les informations et statistiques, doivent se **mettre à jour** à chaque production d'un nouveau jeu de données par l'utilisateur.
Afin de valider les critères ci-dessus, des **tests** avec différentes requêtes seront réalisés.

2 Expression des besoins

L'application est un outil en ligne de commande. Elle comprend deux sous-commandes, l'une permettant la création d'un jeu de données et l'autre sa reconstitution.
Le tableau ci-dessous décrit les arguments de ces sous-commandes :

Tableau 4 : Description des arguments et options de chaque sous-commande de l'application

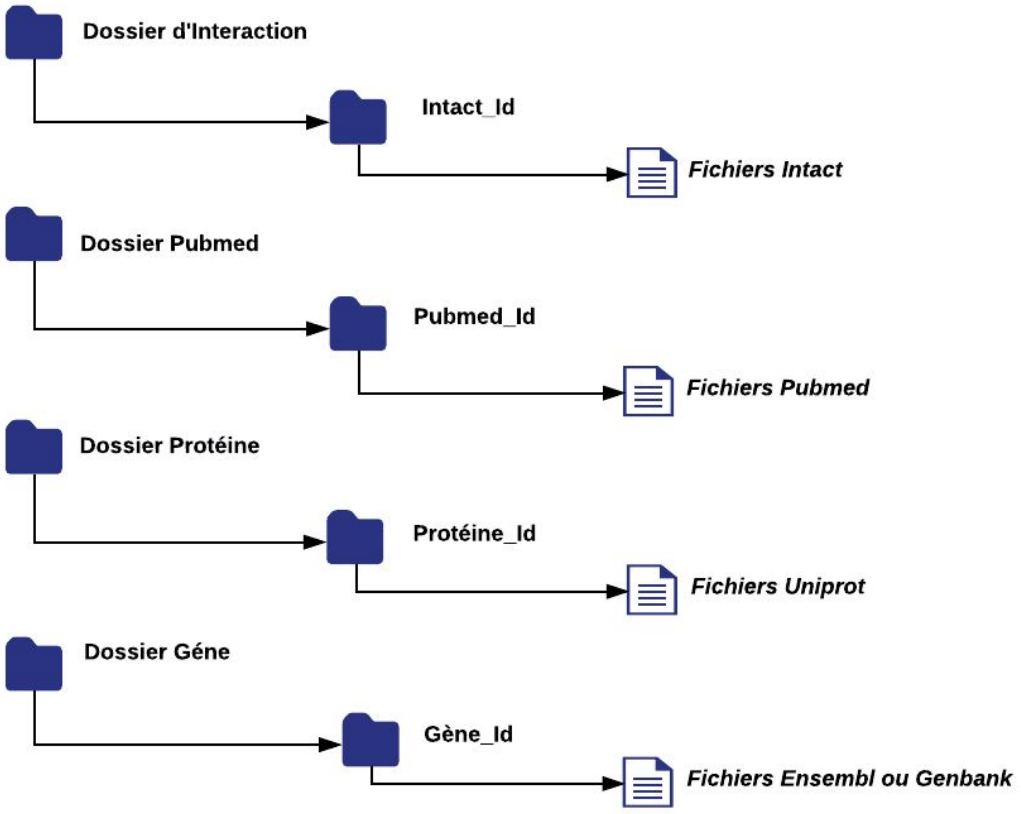
	Nom_Outil <subcommand>	
Sous Commande	**Create** [OPTIONS] < -p ID_pubmed | -1 ID_prot1 -2 ID_prot2 | -t tuple_taxo >	**Dump** [OPTION] <id jeu de données> <date> <dossier de sortie>
Arguments	-p Un ou plusieurs identifiant(s) pubmed -1 Identifiant Uniprotkb -2 Identifiant Uniprotkb -t Tuple d'identifiants taxonomique	1. id jeu de données 2. date : date de création du jeu de données 3. dossier de sortie : chemin vers un répertoire
Option	-max Nombre de protéines maximum à prendre en compte dans le fichier d'interaction -f Pour forcer le remplacement des fichiers du jeu de données -i Pour prendre en compte les d'interactions intra-espèce	-o <format> : format de sortie des fichiers du jeu de données -l <liste> : retourne la liste des jeux de données avec leurs identifiants, la date, la complétude et la correspondance

2.1 Sous commande Create

Dans un premier temps le programme interrogera la base **Intact** à partir des arguments d'entrée. Ensuite pour chaque interaction trouvée contenant au moins une protéine différente, les fournisseurs suivants seront interrogés : Uniprot, Ensembl ou Genbank et Pubmed.

Les Fichiers de sortie sont stockés dans 4 répertoires. Afin de limiter la génération de répertoire, l'arborescence de la Figure 1 sera utilisée :

Figure 1: Arborescence de stockage



De plus les informations citées en 1.2.1 seront stockées dans une base de données relationnelle.

2.2 Sous commande Dump

Cette sous commande permet à l'utilisateur de récupérer un jeu de données dans le répertoire de son choix. En effet, la hiérarchie de stockage en protéines, interactions, articles et gènes permet de limiter les redondances d'information, mais elle rend plus complexe la récupération d'un jeu de données complet .De plus, l'utilisateur peut préciser le format des fichiers qu'il souhaite récupérer.

3 Contraintes

3.1 Délais

Plusieurs échéances sont prévues tout au long du projet.
Tout d'abord la version définitive du cahier des charges est à envoyer le 5 février.
Ensuite un premier prototype de l'application python fonctionnelle est à proposer pour le 13 février.
La version finale de l'application sera à communiquer le 2 avril, avec le code source de l'application web si réalisé.
Enfin le 3 avril, une présentation orale sera effectuée et conclura le projet.

3.2 Autres contraintes

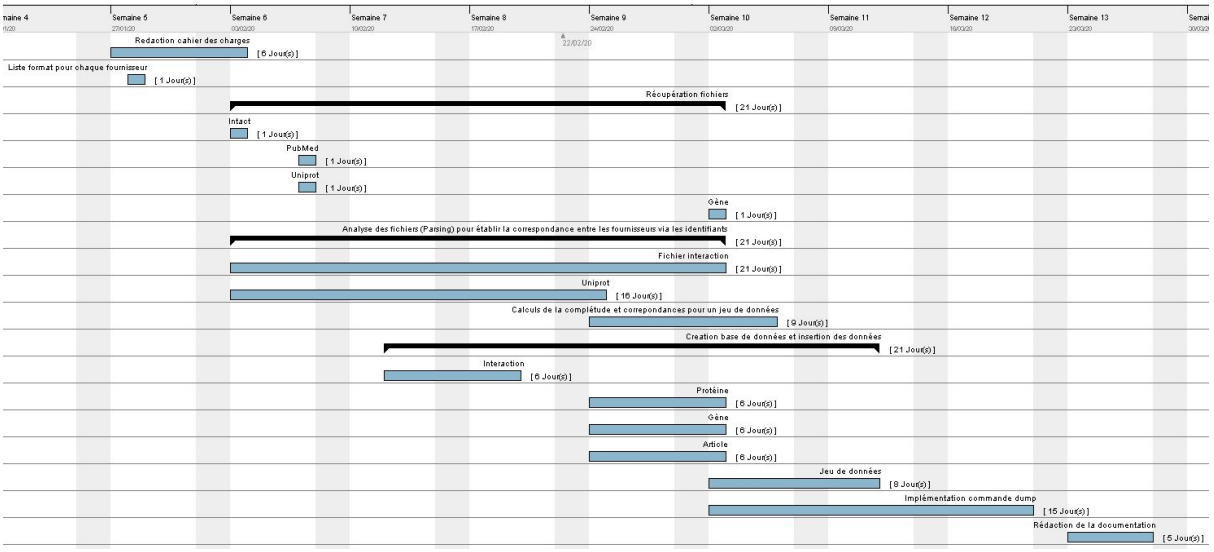
Des réunions hebdomadaires avec les porteurs du projets seront effectuées.
L'application python doit être développée sous la version Python 3 et utiliser les modules précédemment cités.

4 Déroulement du projet

4.1 Planification

Le diagramme de Gantt ci-dessous présente les grandes phases du projet durant les 5 prochaines semaines.

Figure 2 : Diagramme de Gantt du projet



Voici le déroulé détaillé du projet :

- Rédaction du cahier des charges (27/01/2020 au 03/02/2020)
- Liste des formats pour chaque fournisseur (28/01/2020)
- Récupération des fichiers pour chaque fournisseur (03/02/2020 au 02/03/2020)
- Analyse des fichiers (parsing) pour établir la correspondance entre les fournisseurs via les identifiants (03/02/2020 au 02/03/2020)
- Création de la base de données et insertion des données (12/02/2020 au 11/03/2020)
- Calculs de la complétude du jeu de données et des correspondances par fournisseur (24/02/2020 au 05/03/2020)
- Implémentation commande Dump (02/03/2020 au 20/03/2020)
- Rédaction de la documentation (23/03/2020 au 27/03/2020)

4.2 Délivrables

Le rendu final sera constitué de:

- L'application python
- La base de données (fichier .db et dump en sql) et au moins un jeu de données collecté via l'application

Le rendu final sera accompagné d'une documentation utilisateur et d'une documentation développeur afin de faciliter la reprise des outils par d'autres développeurs.

4.4 Responsabilités

Le projet développement d'une application pour l'automatisation de jeu de données est porté par Fabien DUCHATEAU , Enseignant-chercheurs à Université Lyon 1 au Laboratoire d'InfoRmatique en Image et Systèmes d'information et Guillaume LAUNAY, Enseignant-chercheurs à Université Lyon 1 à l'Institut de Biologie et Chimie des Protéines.

Afin de réaliser l'application, le travail ici présenté est soumis comme projet aux étudiants du master de bioinformatique dans la cadre de l'UE « Projet En Bioinformatique 2 » (UE-BIO1282M). Les étudiants participants sont MOULLÉ Pauline, KORALEWSKI Alexis et THALAMAS Thibaut.

5 Références

(1) Fabien Duchateau, UCBL - LIRIS https://perso.liris.cnrs.fr/fabien.duchateau/index.php?page=ens/BDBIO/bdbio.php (accessed Feb 4, 2020). [pdf projet : https://perso.liris.cnrs.fr/fabien.duchateau/ens/BDBIO/tp/projet.pdf]

(2) IntAct Molecular Interaction Database https://www.ebi.ac.uk/intact/ (accessed Feb 4, 2020).

(3) pubmeddev. Home - PubMed - NCBI https://www.ncbi.nlm.nih.gov/pubmed/ (accessed Feb 4, 2020).

(4) UniProt https://www.uniprot.org/ (accessed Feb 4, 2020).

(5) Ensembl genome browser 99 https://www.ensembl.org/index.html (accessed Feb 4, 2020).

(6) GenBank Overview https://www.ncbi.nlm.nih.gov/genbank/ (accessed Feb 4, 2020).

(7) json — JSON encoder and decoder — Python 3.8.1 documentation https://docs.python.org/3/library/json.html (accessed Feb 4, 2020).

(8) xml.etree.ElementTree — The ElementTree XML API — Python 3.8.1 documentation https://docs.python.org/3/library/xml.etree.elementtree.html (accessed Feb 4, 2020).

(9) Team, Rdfl. *Rdflib: RDFLib Is a Python Library for Working with RDF, a Simple yet Powerful Language for Representing Information.*

(10) sqlite3 — DB-API 2.0 interface for SQLite databases — Python 3.8.1 documentation https://docs.python.org/3/library/sqlite3.html (accessed Feb 4, 2020).

(11) *Docopt/Docopt*; docopt, 2020.

(12) SQLiteStudio https://sqlitestudio.pl/index.rvt (accessed Feb 4, 2020).

(13) REST · GitBook http://psicquic.github.io/PsicquicSpec_1_4_Rest.html (accessed Feb 4, 2020).

(14) E-utilities Quick Start - Entrez Programming Utilities Help - NCBI Bookshelf https://www.ncbi.nlm.nih.gov/books/NBK25500/#chapter1.Searching_a_Database (accessed Feb 4, 2020).

(15) Programmatic access - Retrieving entries via queries https://www.uniprot.org/help/api_queries (accessed Feb 4, 2020).

(16) Ensembl Rest API - GET xrefs/id/:id https://rest.ensembl.org/documentation/info/xref_id (accessed Feb 4, 2020).