Rapport du projet de cours de l'Inalco

Objectif du projet

L'objectif du projet, c'est de faire la conversion de la source du tableau d'Excel en bases de données au format TriG. Les bases de données en format TriG qui contiennent des informations des cours à l'Inalco. Avec ces bases de données, nous pouvons compléter le site de LaCAS.
LaCAS est une plateforme d'agrégation, d'organisation et de diffusion des données générées par les chercheurs et équipes de l'Inalco, de l'Université Paris Cité et de leurs partenaires en langues et sociétés du monde.

Description de la source (Excel)

Le tableau Excel « Feuille 1.xlsx » comporte 42 colonnes et 4805 lignes. Il contient des informations sur 4804 cours, tels que le code du cours, les noms d'enseignants du cours, la description du cours, les heures de cours, etc. Parmi les 42 colonnes, certaines colonnes ne comportent pas d'informations, telles que « SPECIALITE », « COURS PRES/HYBR/DIST. », etc. Pour certains cours, les informations sont incomplètes. Lorsqu'une cellule ne contient aucune information, elle est parfois vide et parfois contient le symbole « - ». Certaines de ces cellules contiennent des informations incorrectes, par exemple la colonne « #REF! » et « #NA ». La figure 1 montre une partie d'information d'un cours du tableau d'Excel.

1	CODE	NIVEAU	ELP - Composante	DPT/FIL/THD	DIPLÔME NATIONAL	DIPLÔME ETAB	RATTACHEMENT	Libellé long	ELP - Nature	Cours suspendu sur Apogée (OUI/NON)	ECTS	IP 2020-2021	IP 2021-2022	Langue utilisée	SPECIALITE
1530	ESTA310B	L3	ECO	Europe	Licence LLCER	-	Estonien	Version estonienne 1	EP Langue	N	4	2	2	-	-

Figure 1. exemple du cours du tableau d'Excel

Processus du travail

Le but du projet est de convertir les informations des cours du fichier Excel aux fichiers en format TriG. Le processus de conversions a été réalisé par un programme Python qui est composé de quatre parties principales : le nettoyage de la source, le langage régulier, les requêtes de SPARQL et l'étape de la conversion.

1. Nettoyage de la source

Puisque des bases de données TriG sont convertis directement du contenu du tableau Excel, donc la qualité des données de la source détermine la qualité des données générées. Comme tous les contenus du tableau d'Excel sont saisis manuellement, il est inévitable qu'une certaine erreur humaine se produise. Le

travail du nettoyage des données est une partie très importante du processus. Le travail principal de nettoyage consiste en la suppression des espaces superflus en début et fin de chaîne de caractère, en la suppression des cellules qui n'ont que des espaces, en la division des cellules fusionnées, en la correction des saisies incorrectes (les formulaires) et en la correction des doublons de code des cours.

2. Langage régulier

Le langage régulier est utilisé dans *def find_last_first_name*. La requête de SPARQL demande de saisir séparément le nom de famille et le prénom d'enseignants afin de trouver l'URI d'enseignant. Mais beaucoup d'informations de noms de l'enseignant ont été saisies sans respecter les règles de saisie. Les noms de l'enseignant sont présentés sous au moins 7 différentes formes. Elles sont listées ci-dessous :

1. Prénom (première lettre majuscule) +Nom de famille(première lettre majuscule), par exemple, « Michel Blanchard »
2. Nom de famille (première lettre majuscule) + Prénom(première lettre majuscule), par exemple, « Ville Jean-Luc »
3. Initiale + Nom de famille(première lettre majuscule), par exemple, « L. Pourchez », « J.-L. Ville », « S.S. Hnin Tun »
4. Nom de famille en majuscule + Prénom(première lettre majuscule), par exemple, « KELEDJIAN Mélanie »
5. Nom de famille, par exemple, « Germanos », « Massoud », « SILBERZTEIN »
6. Plusieurs noms d'enseignants, par exemple, « Fathi/ Massoud »
7. Erreur entre nom et prénom, par exemple, « LIWEN Chen »

3. SPARQL

Dans le programme, il y a trois requêtes de SPARQL. Elles sont *def find_uri, def find_individual* et *def find_individual_lastname*. La première requête demande des informations d'URIs de classe « cible » et des valeurs des propriétés. Avec cette requête, nous pouvons trouver tous les URIs de propriétés, sauf les URIs des enseignants, contrairement aux autres propriétés qui peuvent être interrogées directement sur leurs contenus. Lors de l'interrogation de l'URI d'enseignant, il est nécessaire de diviser d'abord les informations textuelles de l'enseignant en nom et prénom. Dans *def find_individual* nous cherchons les URIs selon le nom et prénom. Avec cette requête, nous pouvons trouver la majorité d'URIs. Cependant, sur certains cours qui n'ont que l'information de nom de famille d'enseignant, afin de résoudre ce problème, nous avons ajouté une autre requête *def find_individual_lastname.*

4. Conversion

Dans cette étape, nous avons utilisé le module *rdflib* de Python. D'abord, nous avons créé une base de données vide avec la fonction *rdflib.Dataset()*. Ensuite, le module *Graph.add(triplet)* nous permet d'ajouter des nouveaux triplets dans cette

base de données. Dans les triplets, nous pouvons trouver les URIs des cours en tant que sujet du triplet. Le prédicat est l'URI de la relation entre la propriété et le cours. L'objet est l'URI qui est correspondant de la valeur littérale de la propriété ou la valeur littérale.
Contrairement aux autres propriétés où l'information textuelle est directement ajoutée dans le triplet lorsque leur URI correspondant n'existe pas, l'ajout objet de propriété de l'enseignant est plus complexe. D'abord, il faut interroger les requêtes SPARQL avec les valeurs littérales du nom de l'enseignant. Après avoir confirmé qu'il n'existe pas d'URI ou qu'il en existe plusieurs, nous allons prendre quelques étapes supplémentaires pour traiter les deux cas. Dans *def find_individual* et *def find_individual_lastname* nous exigeons que le résultat final renvoie « None » et « duplicate », s'il n'a pas d'URI pour le nom de l'enseignant ou s'il y en a plus d'un. Si le résultat est « None », ça veut dire que cet enseignant n'existe pas sur LaCAS. Dans ce cas-là, d'abord nous stockons les noms dans une liste, ensuite nous créons leur URI. Les URIs créés sont sans accent, espace et symboles, etc. Pour les enseignants dont leurs noms nous donnent le résultat « duplicate », nous les stockons dans une autre liste. Comme il n'est pas possible de confirmer si un enseignant a plus d'une URI ou s'il y a un homonyme, donc, nous avons besoin d'une personne qui connait les informations des enseignants pour confirmer les résultats.

Difficultés

Pendant les processus de conversion, nous avons eu des difficultés qui ont affecté le déroulement de nos travaux. Nous les classons en quatre points suivants :

1. Source

Comme nous l'avons déjà mentionné, étant donné que les données TriG sont générées directement à partir du tableau d'Excel, la qualité des données originales détermine la qualité des données générées. Cependant, la qualité du tableau d'Excel est discutable. Car le travail d'enregistrement ne suit pas une norme lors de la saisie des informations. Certaines mêmes informations sont saisies dans différents textes, ce qui rend leur exploitation ultérieure beaucoup plus difficile. Par exemple, dans la propriété « PREREQUIS DE LANGUE », nous pouvons trouver la même information « prérequis » sous les différents textes, tels que « OUI », « souhaité », « Une bonne connaissance », « Prérequis : connaissance approfondie de la langue arabe », etc. Il est difficile de lister tous les éléments textuels lorsque notre programme fait la requête de SPARQL. En particulier, le format des informations sur les noms des enseignants prêtait à confusion, ce qui ajoutait beaucoup de travail au travail préliminaire. Il y avait également des incohérences entre les informations textuelles d'Excel et les valeurs des labs dans l'ontologie, ce qui a entraîné de nombreuses difficultés pour obtenir l'URI. Nous avons également

constaté l'erreur de plusieurs valeurs partageant une même URI. Ce type d'erreur est parfois difficile à détecter dans les bases de données TriG. Il faut vérifier l'ontologie. En plus, des mauvais formulaires ont provoqué des soucis de contenu du tableau. Par exemple, « #REF! », « #NA » et « 0 » . Ce sont les résultats qui sont produits par les mauvais formulaires. Le programme les convertit aux valeurs « None » qui sont des informations inutiles. À la fin, il faut noter que les codes en doublons, en plus d'entraîner l'omission de certains cours, peuvent également causer l'apparition de plusieurs cours dans le même triplet.

2. REGEX des noms d'enseignants

Nous avons mentionné que les noms des enseignants étaient saisis sous différentes formes en raison de l'absence d'une forme de saisie. Après avoir analysé les données, nous avons classé sept formes de noms qui sont listées ci-dessus.
Afin d' extraire séparément le nom et prénom avec précision, nous avons écrit 4 patterns de langage régulier qui peuvent réaliser le travail d'extraction. Ils sont destinés à extraire respectivement le nom de famille en lettres majuscules, l'initiale du prénom, les cas qui ne comportent que le nom de famille et le dernier pattern qui peut traiter les cas restants. Au début du processus d'extraction, nous avons eu des difficultés avec des noms composés. En plus, de nouvelles formes de noms étaient constamment découvertes, nous avons toujours modifié des patterns qui existaient ou ajouté des nouveaux.

3. SPARQL

Ayant peu de connaissances préalables dans le domaine de SPARQL, il m'a fallu un certain temps pour réaliser les requêtes. Dans les requêtes, d'abord,il faut que nous trouvions les triplets de l'objet de requête. Ensuite, nous ajoutons des conditions de filtre afin de trouver l'URI correspondant à la valeur. Puisque le type de data de valeur souvent sont des chaînes de caractères, donc une grande attention doit être portée à l'utilisation des symboles de l'interprète, par exemple « \ » .

Description des résultats (TriG)

Après avoir résolu les difficultés mentionnées, nous avons généré deux bases de données selon le niveau d'étude. La première est la base de données qui ne concerne que les informations des cours de master. La seconde comporte les informations des cours de tous les niveaux. Il faut souligner que les informations mentionnées ici se rapportent au contenu des 29 colonnes traitées. Parmi les 42 colonnes, il y a 13 colonnes dont nous n'avons pas besoin de leurs informations pour le moment.
La base de données de master contient 1363 triplets. C'est-à-dire, nous avons généré 1363 de cours au niveau de master. Un exemple est montré ci-dessous.

Figure 2. Un exemple d'un triplet du fichier TriG

Dans cet exemple, nous pouvons trouver que la préfix « ns1: » avec la valeur de la propriété « CODE » est le sujet du triplet. Les trois premiers objets partagent le même prédicat qui est l'abréviation de l'URI de type. Les objets restants sont leur propre prédicat. Nous remarquons que les objets du triplet n'ont pas le même type de données. Puisque certains objets n'ont pas leur propre URIs, nous avons donc gardé leur valeur du tableau d'Excel, les chaînes de caractères. Si les valeurs ont leur URIs, nous les récupérerons comme objet du triplet. Nous pouvons trouver deux formats d'URI dans ce triplet qui sont strictement équivalents. C'est le serializer de *rdflib* qui décide quelle forme va être utilisée. Il peut varier sans que nous sachions trop quelle est l'heuristique utilisée. Le long format est de la forme <http://www.campus-AAR.fr/halauthorid/683210>. Le format court avec un préfixe. Un exemple de la forme courte : ns2:ontology_428999666. Il faut noter qu'il y a 19 cours qui manquent dans le fichier TriG par rapport au nombre du cours du tableau. Parce que le programme traite les informations de cours selon leur code du cours, donc un code égale un cours. Mais dans le tableau Excel, 38 cours partagent 19 codes.
L'autre base de données TriG comporte les informations de tous les cours. Le tableau Excel actuel contient 4805 cours. Mais le fichier TriG n'a que 4769 triplets, c'est-à-dire qu'il y a 35 cours qui manquent, à part les 19 cours mentionnés ci-dessus. Nous avons également identifié 32 autres cours, 16 binômes. Ils ont un code identique à chaque binôme.

Amélioration pour le futur

Pour l'instant, nous avons poussé une version de base de données TriG sur le site LaCAS. Les triplets sont bien interprétés. Toutefois certains points peuvent être améliorés. Du côté de la source, malgré le fait que nous avons fait beaucoup de travail de nettoyage, mais il y a toujours des points pour lesquels nous n'avons pas trouvé de solution. Dans la propriété « VOL. HORAIRE SEMESTRE », les valeurs de cette propriété devaient être des chiffres, mais nous avons trouvé le type « Date

». En plus, nous ne pouvons pas les modifier, parce qu'elles sont protégées. Nous pouvons trouver une capture d'Excel ci-dessous.

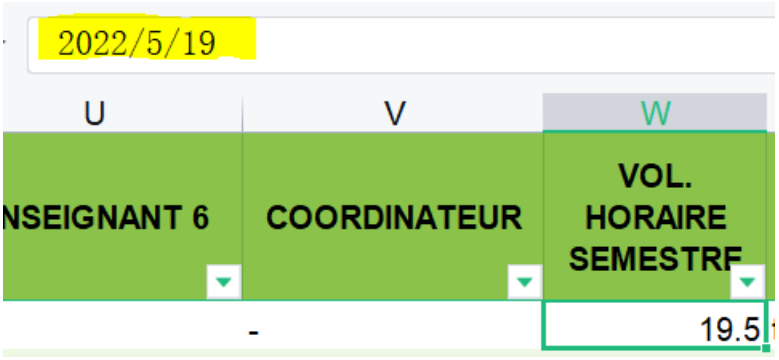

Figure 3. Une capture de l'incohérence de la valeur du propriété « VOL. HORAIRE SEMESTRE »

Ensuite, nous avons trouvé que 35 cours ont les mêmes codes que les 35 autres. Ces doublons font que le programme convertit 35 cours de moins. Car le programme définit un cours par son code. À la fin, ce sont des incohérences des informations textuelles dans le tableau. Afin de résoudre ces problèmes, des travaux d'enregistrement, il est également nécessaire d'établir une norme pour le travail de la saisie.

Du côté du programme, des efforts restent à faire dans la partie des requêtes de SPARQL. Par exemple, nous pouvons trouver la manière d'unir les deux requêtes de noms d'enseignants. En plus, les valeurs étant peu nombreuses et les labels différents entre l'excel et les valeurs présentes sur Okapi, il faut utiliser un dictionnaire statique pour faire le mapping entre les deux plutôt qu'une requête SPARQL.