Université des Sciences de Montpellier II
Département Informatiques
Master 1 Informatique pour les sciences (IPS)

Alignements terminologiques et conceptuels dans le domaine de la culture.

- Alignements de concerts prévus aux concerts réalisés de la Philharmonie de Paris

Réalisé par :

Mr KOCHED Hichem
Mme MENZOU Kahina
Mme MESSADI Radia
Mr OUAIL Md Amine

Encadrant : Mr Konstantin TODOROV

Année universitaire : 2018-2019

Remerciements

Nous remercions tout particulièrement notre Encadrant Monsieur ***Konstantin TODOROV*** *pour sa patience, sa générosité, disponibilité et judicieux conseils.*

Nos vifs remerciements à Madame ***Cécile CECCONI,*** *Chef de projet, responsable catalogue et normes de la Philharmonie de Paris, pour le temps qu'elle a consacré pour les précieuses informations fournies tout au long du projet.*

Table des matières

Liste des figures

INTRODUCTION

I. Contexte du domaine applicatif

La clarification des différentes données, leur lecture, leur classification mais aussi les liens formés entre elles font l'objet d'une étude pratique qui permettra de traiter des données particulières de l'institution musicale de la philharmonie de Paris.

Ce projet s'inscrit dans le cadre du master I Informatique pour les sciences (IPS), plus précisément dans l'unité d'enseignement intitulée Travail d'Etude et Recherche (TER), celle qui offre aux étudiants l'opportunité de mise en œuvre (exécution) de leur capacités et compétences dans le domaine pratique tout en tenant compte de la phase recherche, conception, l'élaboration et le partenariat .

L'objectif consiste à réaliser un alignement et une mise en correspondance des différents concerts (prévus et réalisés) de l'institution.

Les experts de l'institution ont en premier lieu organisé leurs données dans deux bases particulières, nommée EUTERPE qui regroupe les données des concerts prévus et une autre nommée ALOES ou PP regroupant à son tour les concerts réalisés dans des fichiers TURTLE structurés sous la forme de graphe grâce au formalisme RDF.

Ce qui est à retenir pour chaque base est :

Les informations concernant les concerts prévus (base EUTERPE) figurent dans la classe M26. Les informations concernant les concerts réalisés (base ALOES ou PP) figurent dans les classes F31 et F29

- 80% des concerts réalisés sont enregistrés en audio.
- 10 à 20% des concerts réalisés sont enregistrés en vidéo.
- Les concerts réalisés peuvent être enregistré en audio et en vidéo (décrits 2 fois).
- Certains concerts sont enregistrés par Radio France.

II. Objectif informatique

L'objectif consiste à créer et explorer des méthodes de liage et de rapprochement sur les deux bases afin d'aligner chaque information présente sur la base EUTERPE avec celle qui lui correspond sur celle d'ALOES.

Afin de manipuler et décrire ces données, on fait recours au web sémantique en employant des vocabulaires représentants toutes ces données et en utilisant des technologies diverses.

ETAT DE L'EXISTANT

Cette phase de documentation approfondie est nécessaire au travail dans le but d'une bonne compréhension des principes fondamentaux existants et à partir des livres, articles et travaux antérieurs afin de nous familiariser avec les disciplines visées, une étape qui nous aidera à définir la portée du travail d'alignement qui représente notre perspective principale dans ce projet.

I Le Web sémantique

Le web sémantique est une dérivée du web, permettant une communication fluide et facile mais surtout intelligentes entre les machines durant l'interrogation et la manipulation des différentes données du web.

La différence entre le web actuel et le web sémantique peut être représentée comme suit :

Web actuel	Web sémantique
Ensemble de documents	Ensemble de connaissances
Basé essentiellement sur HTML	Basé sur XML et RDF(S)
Recherche par mots clé	Recherche par concepts
Utilisable par l'humain	Utilisable par la machine

1. Dispositif langagier de base du web sémantique

C'est un dispositif langagier normalisé par le W3C, qui montre l'agencement des différentes briques technologiques du Web sémantique. Les fonctions et les relations des composants peuvent être résumées comme suit :

• **URI/IRI** : c'est une adresse web (chaine de caractères) qui permet d'accéder à la donnée.
Plusieurs d'entre elles commencent par « http:// » comme l'illustre l'exemple ci-dessous : http://data.doremus.org/performance/4be8bdc3-4ee5-38cc-b18a-71f9adde037e/8

• **RDF :** c'est un langage qui permet de décrire la donnée, dans lequel toute ressource est identifiée par une URI et cet ensemble de ressources composent un graphe.

Les nœuds : représentent des ressources.

Les arcs : représentent des relations entre ces ressource

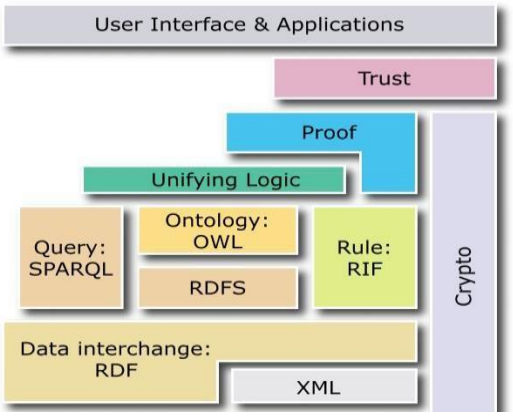

Figure 1 : La sémantique web stack (https://fr.wikipedia.org)

2. Structure des données RDF

Un triplet RDF, est une association de type {**sujet, prédicat, objet**}.

- Le *sujet* représente la ressource à décrire.
- Le *prédicat* représente un type de propriété applicable à cette ressource.
- L'*objet* représente une donnée ou une autre ressource : représente la valeur de la propriété.

La structure des données RDF est synthétisée sur l'exemple schématisé ci-dessous afin de mieux la comprendre :

Tim Berners-Lee est une personne

URI 1 URI 2

<http://www.w3.org.People/Berners-Le/card#i> <http://www.w3.org/1999/02/22-rdf-

URI 3

syntax-ns#type> <http://xmlns.com/foaf/0.1/Person>.

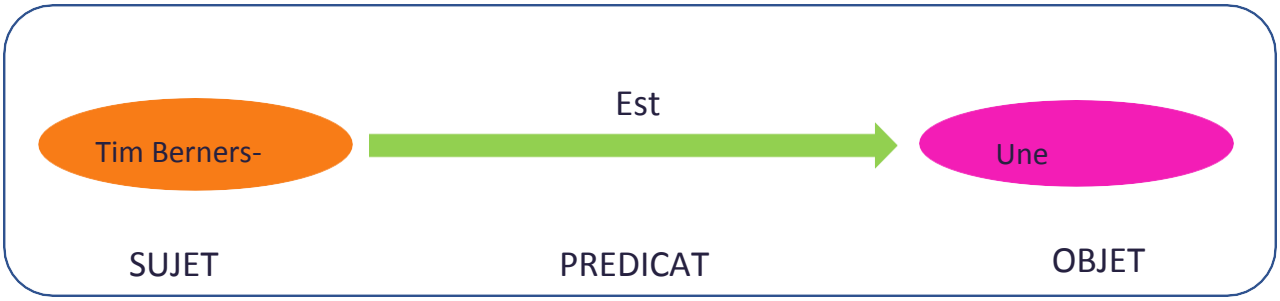

Figure 2 : Schéma RDF Simplifié
Source : Auteurs

- **XML :** est un langage de balisage extensible, il s'agit plutôt d'un métalangage qui nous permet de définir des propres balises pour nos documents.
- **SPARQL :** est un langage de requêtes applicable sur des triplets RDF. Il permet d'extraire les informations sous forme d'URI, de nœuds vides ou de littéraux et construire de nouveaux RDF à partir des informations disponibles dans le web à travers l'internet.

II L'API Jena

Apache Jena est un framework adapté à Java et composé de différentes API permettant la manipulation multiple des données RDF et la conception d'applications liées au Web sémantique

Ces graphes sont représentés sous forme de modèle qui lui, peut-être crée à partir des données provenant d'un ensemble de fichiers, de bases de données, d'URL ou encore d'une combinaison de ceux-ci. Ainsi, un modèle peut également être interrogé par des requêtes SPARQL.
Il est à noter aussi que Jena prend en charge le langage OWL (Web Ontology Language).

III L'outil SILK

Silk est un framework open source pour l'intégration de sources de données hétérogènes. Il est basé sur le modèle des données liées, et se repose sur deux idées fondamentales et simples :

Premièrement, le RDF fournit un modèle de données expressives pour la représentation d'informations structurées.

Deuxièmement, les liens RDF sont établis entre des entités de différentes sources de données. L'outil Silk est utilisé pour spécifier les types de liens RDF qui doivent être découverts entre les sources

de données (source et target), ainsi que les conditions que doivent remplir les éléments de données pour être interconnectés.

Ce framwork accède aux sources de données qui doivent être interconnectées via le protocole SPARQL pour qu'elles soient utilisables. Les spécifications de lien peuvent être créées à l'aide de l'interface utilisateur graphique de Silk Workbench ou manuellement en créant un fichier de configuration sous format XML où on fait appel à un fichier source et un fichier target, d'un type de lien, un type de ressources à comparer (SPARQL), d'une propriété à comparer ainsi qu'un output pour les résultats d'alignement et la mesure de similarité (voir ultérieurement dans la section **V. les mesures de similarité**) est associée à ces derniers suivant le seuil accordé. L'exécution de fichier de configuration se fait sur un terminal.

IV Langage de requête SPARQL

SPARQL est un langage de requête et un protocole pour interroger des triplets RDF, Il est adapté à la structure spécifique de triplets qui les constituent. Ce langage permet d'extraire les informations sous forme d'URI, de nœuds vides ou de littéraux, il construit de nouveaux RDF à partir des informations disponibles dans le web et retourne les informations extraites sous la forme d'un ensemble de liaisons ou d'un graphe RDF.

Le modèle standard ou global de requête SPARQL se représente sous la syntaxe suivante :**Extract1**

PREFIX *prefix atribut1 : < uri relatif >*
PREFIX *prefix atribut2 : < uri relatif >*

PREFIX permet l'utilisation des URIs relatifs.
Ces URIs relatifs évitent de réécrire la même racine de certains URIs.

SELECT *?attribut, ?attribut1 , ?attribut2*

La clause SELECT identifie les variables à faire apparaître dans la réponse de la requête.

WHERE{

La clause WHERE définit des les conditions qui devront être respectées

?attribut a Nom de la classe1 ;
prefix atribut1 : Nom sous-classe 1 ?attribut1 ;
prefix atribut2 : Nom sous-classe 2 ?attribut2.

Triplets

figure 3: Modèle de requête SPARQL global
Source : Auteurs

On peut tout de même ajouter des **FILTER** pour filtrer les résultats des requêtes, des **GROUP BY** pour les regrouper et des **ORDER BY** pour les ordonner.

Parmi les préfixes les plus courants qu'on aura à exploiter ultérieurement :

- **rdf**: http://www.w3.org/1999/02/22-rdf-syntax-ns#
- **rdfs**: http://www.w3.org/2000/01/rdf-schema#
- **ecrm**: http://erlangen-crm.org/current
- **xsd**: http://www.w3.org/2001/XMLSchema#
- **foaf**: http://xmlns.com/foaf/0.1/

- **mus:** http://data.doremus.org/ontology#
- **efrbroo**: http://erlangen-crm.org/efrbroo

V Les mesures de similarité

On entend dire par similarité dans le domaine de l'informatique, toute distinction entre deux groupes d'objets, entre des valeurs numériques mais aussi la reconnaissance ou pas de certaines données.

Une mesure se joint à cette dernière nommée mesure de similarité ou autrement appelée une mesure de distance entre mots, qui est une métrique qui mesure la distance entre deux chaînes de caractères et la comparaison des ces dernières. Le résultat fourni par une métrique est un nombre qui représente une indication sur la distance, et peut varier d'un algorithme à l'autre.

Nombreuses sont les mesures de similarité existantes, on site parmi elles :

- **Distance Levenshtein** : Elle a pour but de comparer entre chaque caractère présent dans deux ‘‘chaines de caractère ’’, le résultat est le nombre minimal de caractère qu'il faut supprimer, insérer ou remplacer pour passer d'une chaîne à l'autre dans un seuil qui varie entre 0= deux chaines complétement différentes et 1= deux chaines identiques.

- **SMOA similarity** : elle permet de donner la différence entre la longueur des sous-chaînes communes et la longueur des sous-chaînes non appariées restantes.

- **Distance de Jaro-Winkler** : sa particularité c'est qu'elle détecte les doublons et elle est mieux utilisée pour la comparaison entre deux chaines de caractères courtes.

- **Co-synonymy similarity** : c'est le résultat de similarité entre la signification de deux termes et non par chaines de caractère.

MISE EN ŒUVRE

I Démarche globale

Le projet répond aux besoins de l'institution de la Philharmonie de Paris qui s'intéresse particulièrement au traitement de neuf (9) cas de figures donnés par les experts de l'institution. Dans l'ensemble des cas, le critère de comparaison primaire est : le titre et sa date respectif pour chaque évènement présent dans les deux bases. La démarche globale pour atteindre l'objectif s'est présentée selon le scénario suivant :

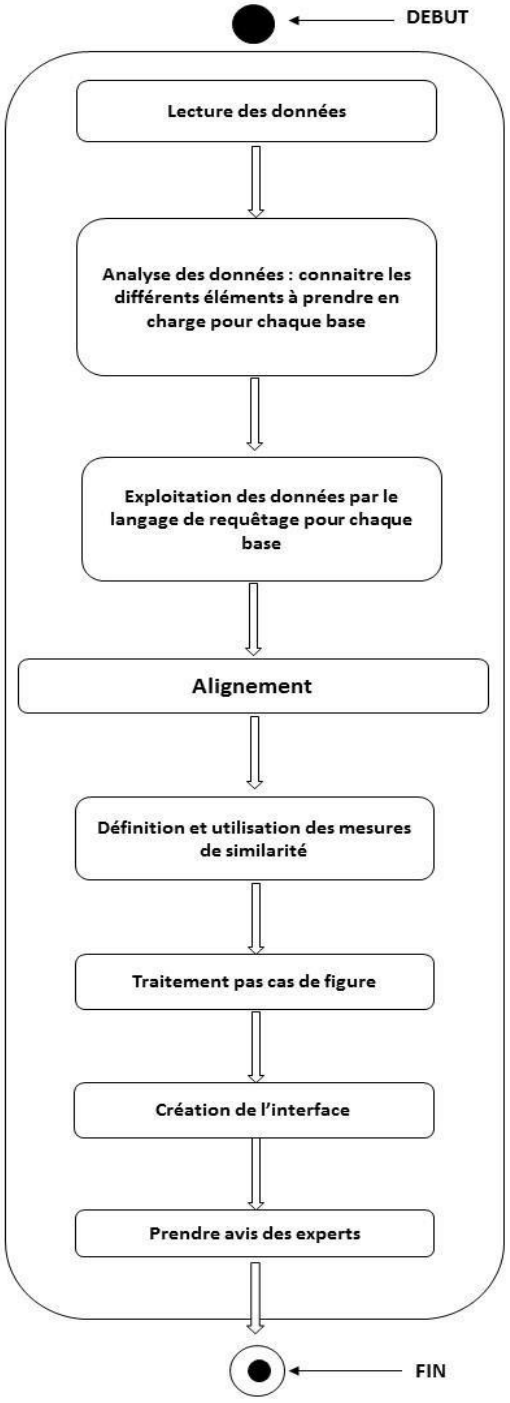

Figure 4 : schéma explicatif des différentes étapes du projet

II. Analyse des données

La première phase s'est portée sur la lecture et l'analyse des classes principales et leurs sous- classes respectives. Le parcours des fichiers de la base de données EUTERPE a défini le constat suivant :

Concerts prévus :

- La base de données EUTERPE contient 3834 fichiers et chaque fichier peut contenir plusieurs concerts prévus.
- mus:M26_Foreseen_Performance => Classe pour les concerts prévus
- La sous-classe relative au titre : ecrm:P102_has_title.
- La sous-classe relative à la date : mus:U8_foresees_time_span.
- La sous-classe relative aux artiste :ecrm :P69_has_association_with
- La base de données ALOES ou PP contient 7322 fichiers.

Concerts réalisés :

- L'ensemble des informations concernant les concerts réalisés dans la base de données ALOES ou PP se trouvent dans deux classes distinctes :
- efrbroo:F31_Performance => pour les concerts réalisés (ALOES ou PP).
- La sous-classe relative au titre : ecrm:P102_has_title.
- efrbroo:F29_Recording_Event => pour les concerts réalisés (ALOES ou PP).
- La sous-classe relative à la date : ecrm:P4_has_time-span.
- Les sous-classes relatives aux artistes : ecrm :P9_consists_of et ecrm :P14_carried_out_by
- La sous-classe relative au type d'enregistrement du concert : ecrm:P32_used_general_technique.

Il est à noter que le croisement des concerts prévus et réalisés se forme à partir du concert du 2008-09-18 ayant le titre « Heureux le peuple qui chante » jusqu'au concert du 2017-03-12 ayant le titre : « les amazones d'Afrique ».

III. Extraction et exploitation des données

Afin de nous permettre la manipulation des données il faut tout d'abord les extraire, ce qui a induit à l'utilisation du langage de requête SPARQL pour une première manipulation des données. En s'appuyant sur le SPARQL Endpoint du http://data.doremus.org/, l'exploitation des données « en ligne » de l'institution était plus efficace et pratique pour utiliser ces requêtes par la suite dans un programme Java.

Le modèle de requête SPARQL adopté se présente sous la syntaxe suivante :

- Modèle de requête pour extraire les concerts prévus selon le modèle global :**Extract1**

```
SELECT ?ConcertPrevu,?date, ?titre
WHERE{
?ConcertPrevu a mus:M26_Foreseen_Performance;
//extraire
mus:U8_foresees_time_span
?date;
ecrm:P102_has_title?
?title.
FILTER(year(?date) AND month(?date) AND day(?date))
}ORDER BY ?date
```

Figure 5: Requête d'extraction (concerts prévus)

- Modèle de requête pour extraire les concerts réalisés selon le modèle global **Extract1** :

```
SELECT ?ConcertRealisé?, title, ?date
WHERE{
?ConcertRealisé efrbroo:F31_Performance;
ecrm:P102_has_title? ?title;
?recording  efrbroo:F29_Recording_Event;
ecrm:P4_has_time-span?date;
FILTER(year(?date) AND month(?date) AND day(?date))
  }ORDER BY ?date
```

Figure 6:Requete d'extraction (concerts réalisés)

L'étape suivante consiste à appliquer les requêtes dans deux codes dans un programme Java : un pour la lecture et le parcours de la totalité des fichiers des deux bases, et un autre pour l'extraction des données en résultats.

PseudoCode1 « lecture des données ».

```
{
Parcourir les fichiers des dossiers en définissant leur chemin
For (parcourir récursivement tous les fichiers existants un par
un)
If le dossier indiqué en chemin contient une liste de fichiers {
Appliquer la requête du pseudoCode2 en extrayant la valeur de la
sous-classe }
```

Figure 7: PseudoCode1: lecture des données

PseudoCode2 « extraction des résultats »

```
   {
 Insertion des préfixes
//Insertion de la requête => Requête d'extraction (concerts prévus) ou
Requête d'extraction (concerts réalisés).
Appeler l'exécution de la requête
While  (premier résultat trouvé, passé au résultat suivant
récursivement et l'afficher ());
 Valeur URI du ConcertPrevu sous forme de chaine de caractère, passé
au résultat suivant et l'afficher ();
 Valeur titre sous forme de chaine de caractère, passé au résultat
suivant et l'afficher () ;
 Valeur date sous forme de chaine de caractère, passé au résultat
suivant et l'afficher ();
Résultat: ''URI ConcertPrevu''+''titre''+''date'' ;
Sortie résultat en fichier.dot}
```

Figure 8:PseudoCode2: extraction des résultats

Vu le nombre important de concerts en sortie, un traitement par saison nous a été imposé. Nous avons donc filtré l'appel de la requête de la saison 2008-2009 à la saison 2016-2017 avec des fichiers résultat **saison.dot** cependant comme dans la saison 2008-2009, le premier concert croisé été prévu et réalisé le septembre 2008 nous avons implémenter le filtre suivant : **FILTER (?date >= "2008-09"^^xsd:gYear AND xsd:gMonth AND ?date < "2010"^^xsd:gYear))** Ensuite, tous les filtres ont eu la même syntaxe : **FILTER (?date >= "YYYY"^^xsd:gYearg AND ?date < "YYYY"^^xsd:gYear)**)
Le modèle de requêtage final se présente comme suit :

```
SELECT ?ConcertPrevu,?date, ?titre
WHERE{
?ConcertPrevu a mus:M26_Foreseen_Performance;
//extraire
mus:U8_foresees_time_span  ?date;
ecrm:P102_has_title? ?title.
FILTER (?date >= "YYYY"^^xsd:gYearg AND ?date < "YYYY"^^xsd:gYear))
}ORDER BY ?date
```

Figure 9 : modèle Extract1 final

IV. Alignement et similarité par cas de figures

Cette phase est considérée comme phase principale du projet, son objectif comme déjà mentionné auparavant est de lier chaque attribut résultant de la requête d'un concert prévu avec celui qui lui correspond dans les classes des concerts réalisés en créant des **couples** de concert.
UriConcertPrévu avec UriConcertRéalisé ;
TitreConcertPrévu avec TitreConcertRéalisé. ;
DateConcertPrévu avec DateConcertRéalisé.
Il faut tout de même mentionner que la condition pour le traitement des cas de figure est que pour chaque cas, la **date** est la **même**.

Pour le traitement des différents cas de figure donnés par les experts de la Philharmonie de Paris, un travail de similarité ou de distance doit être établi pour chaque cas.

Le choix s'est porté sur la mesure de Levenstein (expliquée auparavant dans **état de l'existant V.Les mesures de similarité**).
Lors du traitement par cas de figure, nous avons remarqué des incohérences par rapport à ce qui a été donné par les experts dans 2 des 9 cas : le même cas de figure initié 2 fois et un cas de figure où ils demandent de comparer entre deux concerts qui se sont déroulés le même jour à la même heure or que, les horaires des concerts réalisés n'y figurent pas dans nos bases de données.

a. Cas A : Alignement et Similarité

Nous avons tout d'abord créé une classe qui initialise tous les attributs dont nous aurons besoin tout au long de cette phase (date, titre, artsite et type d'enregistrement) ensuite on a commencé le traitement par « sous-cas » de figure :

Les Cas de figure => attributs utilisés

a.1 Evènement avec même date et même Titre (EMTD)=> date et titre

```
"Ensemble Modern Orchestra - Peter Eötvös"  "2010-11-06  "http://data.doremus.org/performance/f7495f30-3a46-396d-9ec7-05e1b5dd2620"
"Ensemble Modern Orchestra - Peter Eötvös"  "2010-11-06  "http://data.doremus.org/event/e09e8d6f-8995-3bd7-818b-83663215bf38"
```

a.2 Evènement avec même date et titre très proche (EMTDP) =>date et titre

```
"Bach, concertos pour piano - Orchestre de chambre de Lausanne"  "2013-10-21  "http://data.doremus.org/performance/40339280-5e67-3bb9-bcde-640d89bafe0f"
"Bach, concertos pour clavier, Orchestre de chambre de Lausanne"  "2013-10-21  "http://data.doremus.org/event/4066264d-513c-3788-99e0-68a4363dc4f7"
```

a.3 Evènement avec même date et titre légèrement différent (EMTrD)=> date et titre

```
"Ensemble Orchestral de Paris / Accentus"  "2011-02-26  "http://data.doremus.org/performance/fa94cc55-e460-3e3d-ab38-23489ba61887"
"Le rêve américain : Ensemble Orchestral de Paris : Accentus"  "2011-02-26  "http://data.doremus.org/event/428d15af-3573-3e72-803f-cd1f50c3de2a"
```

a.4 Evènement avec même date et titre complétement différent (EMD) => date, titre et critère de comparaison en commun est le nom d'artiste pour dire que l'évènement est le même

```
"Alain Planès - Claude Debussy I""  2008-10-11  "http://data.doremus.org/performance/663e3194-80e5-3795-b108-a07e9700fde3"
"Beethoven / Debussy : Intégrale de l'oeuvre pour piano de Claude Debussy""  2008-10-11  "http://data.doremus.org/performance/8d5c9c30-1f23-3ee7-a3f4-eb445f6df5d7"
 measure  = 0,284
 state  = Différents
```

a.5 Evènement avec même date et même titre enregistré en vidéo ou audio (EMTDE) date, titre, type d'enregistrement.

```
"Rising Stars : Apollon Musagete Quartet"  "2011-01-13  "http://data.doremus.org/performance/041ce907-9fa9-38fe-8cf7-2e1e8a24345a"
"Rising Stars : Apollon Musagete Quartet"  "2011-01-13  "http://data.doremus.org/event/2b461d89-5794-3d52-a5ca-60a909867052"  "audio""
```

PseudoCodeClasseParcours : modèle Type de la classe d'alignement

Etape1 : la lecture des fichiers résultat des requêtes

Création <liste des concerts prévus> Création<liste des Concerts réalisés>

Définition des chemins des dossier contenants les fichiers entrés ***Saison*** *des concerts prévus et réalisés ;*
Définition du chemin du dossier contenant les fichiers résultants (sortis) de l'alignement ;

// La lecture

If *(la liste des fichiers des concerts prévus ou réalisés = null)*

For (*chaque fichier dans* **Fichier** présent *dans la liste des fichiers des concerts prévus ou réalisés)*

{

//Lecture par ligne courante

While *(ligne courante= lecture des lignes des concerts prévus ou réalisés une Par une)*

{

La syntaxe de la ligne courante est sous forme de 3 **groupes** *de chaines de caractères séparés par des (,)*

Ajouter le positionnement attribué à chaque **groupe** *des 2 concerts tout en supprimant la partie qui commence par un T(horaires) dans la position correspondant à la date des concert prévus pour avoir une syntaxe homogène pour la date des concerts réalisés*

}

Fin de lecture}

Figure 10: PseudoCode étape1 :la Lecture

Etape 2 : valeur et statut de la similarité

```
   Définition des noms des catégories de seuils de similarité
 (clé) et leurs (valeurs) respectives
For( i élément dans la liste des concerts prévus, lire ces i éléments)
{
For(j élément dans la liste des concerts réalisé, lire ces j éléments)
{
If ( i élémentDate du ConcertPrévu = j élémentDate du concert réalisé)
{
//vérifier la similarité
   Obtenir le score de similarité pour (i élémentTitre du
concertPrévu avec j élémentTitre du concertRéalisé)
  Initialisation newValue correspondant à la valeur de similarité
if (la valeur de similarité = 0.9){
newValue= valeur de similarité +0.1
// sim varie entre 0.9 et 1 et correspond au cas de figure Cas A : a.1 :
(EMTD)=>Même date, même titre
}
if ( la valeur de similarité = 0.8){ newValue= valeur de similarité +0.09
// sim varie entre 0.8 et 0.899 et correspond au cas de figure Cas A :
a.2 : (EMTDP)=>Même date et titre très proche
}
if (la valeur de similarité = 0.6){ newValue= valeur de similarité +0.19
// sim varie entre 0.9 et 1 et correspond au cas de figure Cas A : a.3 :
(EMTrD) :Meme date et titre légèrement différent
}
 {
  if(le score de similarité correspond à la valeur de seuil de similarité
défini )
{
Afficher ( UriConcertPrévu de i élémentUri du concert prévu+ titre
de i élémentTitre du concert prévu+ date de i élémentDate du
concert prévu)

Afficher ( UriConcertRéalisé de j élémentUri du concert réalisé+
titre de j élémentTitre du concert réalisé+ date de j élémentDate
du concert réalisé)
Fin de l'écriture
}
```

Figure 11:Etape2 Statut et valeur de similarité

Nous avons intégré un code pour le calcul de similarité selon la mesure de Levenstein où nous avons défini les fonctions et les conditions de mesure.

Pour le cas de figure a.5 Evènement avec même date et même titre enregistré en vidéo ou audio (EMTDE) : nous avons introduit la sous-classe ecrm:P32_used_general_technique dans la requête **Extract1** pour la classe F31 seulement, car les enregistrements sont mentionnés que lors de la réalisation des concerts et avons refait le **PseudoCodeParcours** avec l'attribut d'enregistrement initié.

b. Cas B : Alignement par Requêtage

Cas1 : Evènement aussi présent dans les données de Radio France (ERF)

Ce cas n'est pas achevé à 100% en raison de manque de données fournies par les experts.
Le seul attribut qui met en correspondance les données de la Philharmonie de Paris et Radio France c'est la note dont la sous-classe est : ecrm :P3_has_note ou rdfs :comment dans la classe efrbroo:F29_Recording_Event.
Ce cas, a été traité que par la requête **Extract1 avec modèle de requête (concerts réalisés)** en rajoutant que la sous-classe correspondante**,** donc, il n'a pas besoin d'être aligné.

V. Interface de validation par les experts

L'interface est conçue dans le but de communiquer avec les experts de la Philharmonie de Paris à propos des résultats obtenus de chaque alignement pour la validation ou non validation de ces derniers pour l'amélioration des seuils de similarité afin de ressortir avec un alignement précis.

Notre interface est composée de3 parties

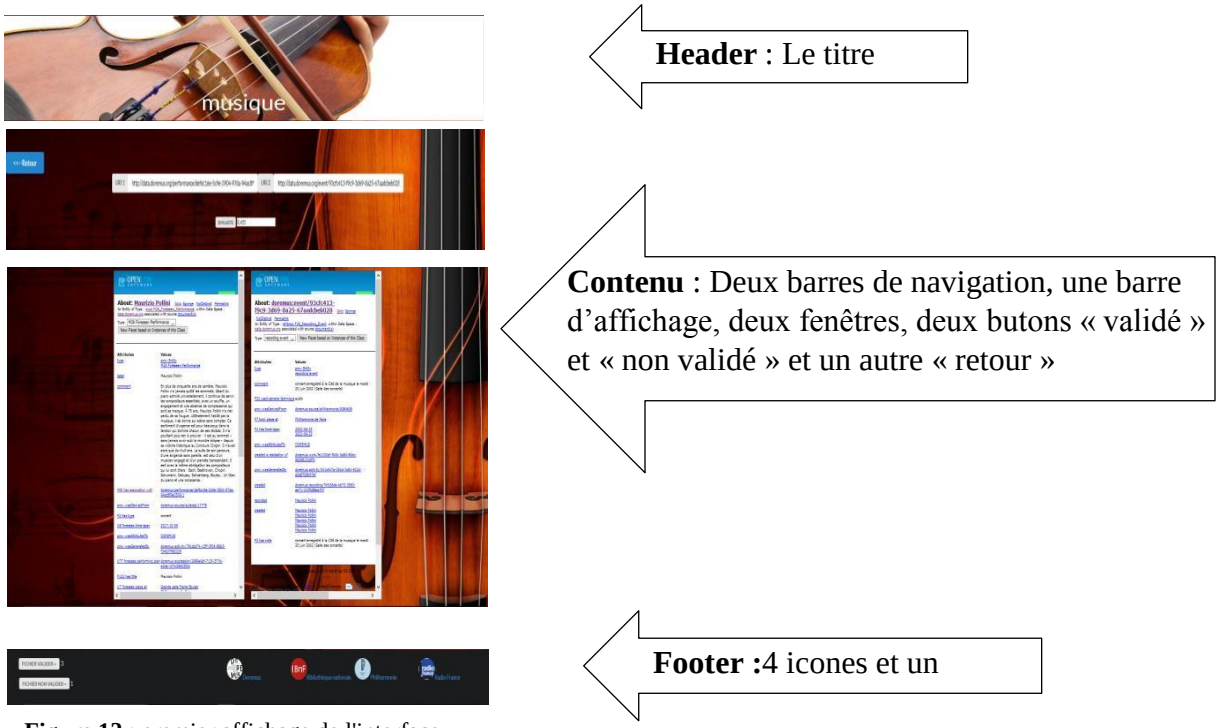

Figure 12 : premier affichage de l'interface

1- **Header** : qui est la partie haute de la page, elle contient un grand titre.

2- **Contenu** : c'est la partie corps de notre interface, elle est composée de :

- Un premier script qui consiste à
  - ✓ Traiter notre fichier source et mettre toutes les lignes dans un tableau associatif « tab »
  - ✓ Nous avons créé 3 variables de sessions : $a qui représente les uri1, $b représente les uri2, $c représente la similarité.

- ✓ Afin de parcourir notre tableau, nous avons créé une autre variable $i qui sera incrémenter à chaque clique sur le bouton « valide » ou « non valide » de notre formulaire. Etant donné que chaque ligne de URI1 est suivie par une ligne de URI2 et une autre de LA SIMILARITE, nous avons utilisé la formule suivante :

```
$a=$tab[$i].
$b=$tab[$i+1].
$c=$tab[$i+2].
```

- ➢ Deux fenêtres qui nous servent à afficher les deux URI, pour cela nous avons utilisé un « iframe »
- ➢ Un formulaire qui contient :
  - ✓ Trois zones pour uri1, uri2 et similarité, ces derniers prendront les valeur des variables $a, $b , $c.
  - ✓ Bouton (YES) pour résultat valider le et (NO) pour non valide.
- ➢ Un deuxième script avec la fonction « window.onload » qui consiste à afficher automatiquement les contenus des deux variables uri1 et uri2 dans les deux fenêtres après chaque clique sur un bouton.
- ➢ Un bouton « retour » pour retourner aux pages précédents.
- ➢ Un script php qui consiste à récupérer les donnes du formulaire, ouvrir un fichier texte et écrire ces données dans le fichier après chaque validation ou non validation.

3- **Footer** qui contient

- ➢ Un compteur intégré avec « include », il consiste à compter les triplets validés et les non validés, pour cela, nous avons créé deux fichiers texte un pour les triplets validés et l'autre pour non validés, chacun est incrémenté de +1 à chaque clique sur valide ou non valide, et pour afficher les valeurs il lit chaque fichier et l'affiche.

- ➢ Une image qui mène vers le site de Doremus
- ➢ Une image qui mène vers le site de Radio France
- ➢ Une image qui mène vers le site de bibliothèque national
- ➢ Une image qui mène vers le site de philharmonie

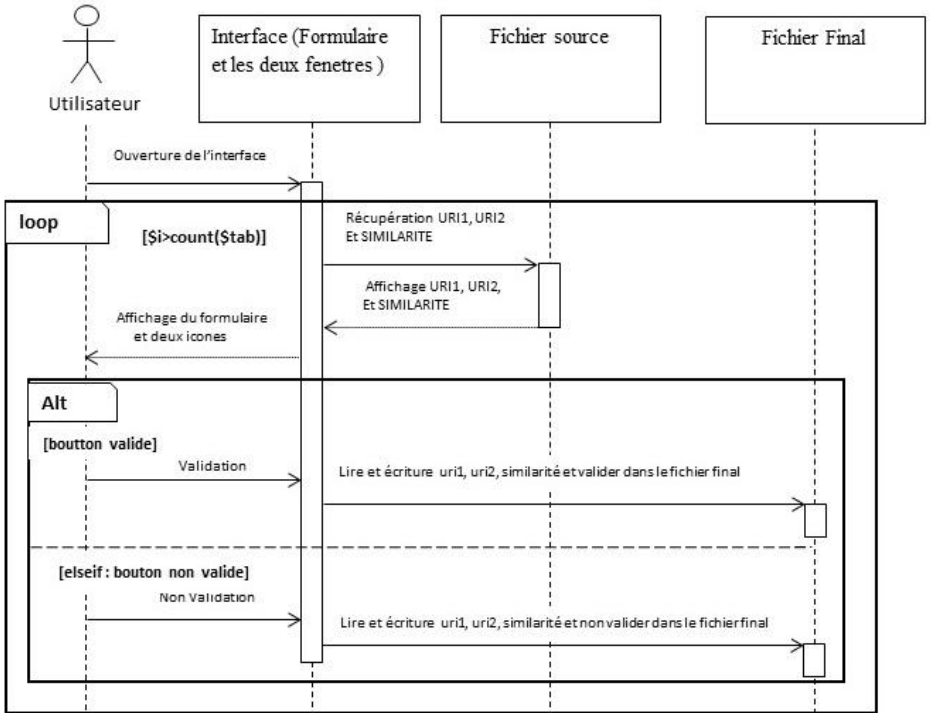

Figure 13: diagramme de séquence

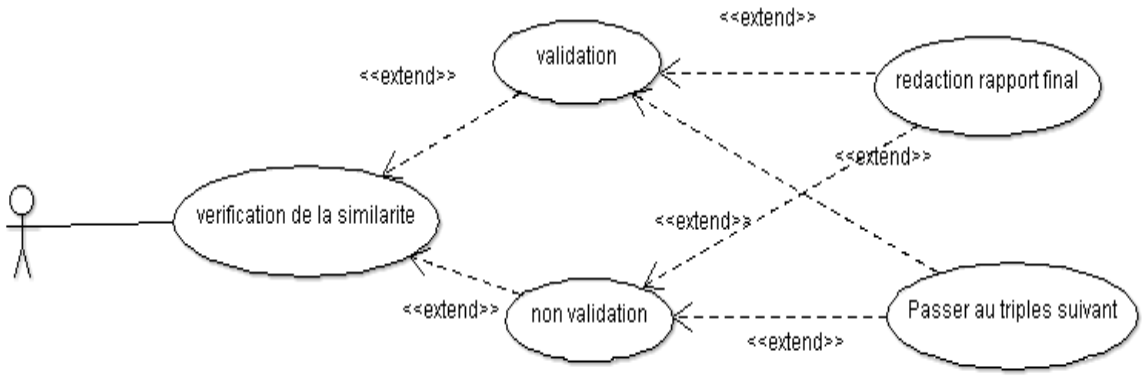

Figure 14: Diagramme de cas d'utilisation

VI. Gestion de projet

a. Organisation interne

L'organisation interne s'est effectuée à travers des réunions hebdomadaires avec l'encadrant du projet, et avec des réunions quasi-journalière avec les membres du groupe.

b. Echange et retour avec les experts

Nous avons tout au long du projet échangé des e-mails avec la *Chef de projet, responsable catalogue et normes* de la Philharmonie de Paris pour poser des questions et avoir d'éventuels éclaircissements concernant le projet, mais aussi pour valider ou non les résultats obtenus.

c. Difficultés rencontrées.

Tout au long du projet nous avons fait face à deux contraintes majeures :

La première : Après l'extraction des données cibles (titre et date) des concerts prévus et réalisés et lors de la phase d'alignement, la démarche à suivre n'était pas assez claire, nous avions le choix entre deux démarches : aligner les résultats par le biais d'un programme ou bien utilisé le framwork SILK. Nous avons opté en premier lieu pour l'alignement par SILK (mécanisme expliqué dans **état de l'existant / IV.Outil SILK**), cependant cet outil prend en charge l'alignement d'attribut appartenant au même type : aligne titre avec titre ou date avec date seulement et ça ne répond pas à l'objectif du projet et donc, nous avons fait recours à l'alignement par le biais d'un programme en Java.

La **deuxième** : Présence des doublons.

Nous avons remarqué dans certains résultats d'alignement qu'ils existent plusieurs dou blons pour le même résultat, cela est expliqué par la présence de plusieurs dates iden- tiques mais avec des horaires différents pour le même concert dans la classe M26, car dans certains cas, le même concert se joue plusieurs fois par jour et le programme prend à chaque fois cette date. Exemple :

"Ateliers-Concerts pour les tout-petits","2016-05-15T**10:00:00**"

"Ateliers-Concerts pour les tout-petits","2016-05-15T**11:00:00**"

Le programme prend les deux titres et les deux dates même si c'est le même évènement. Cette contrainte n'est toujours pas résolue et s'inscrit dans les perspectives futures.

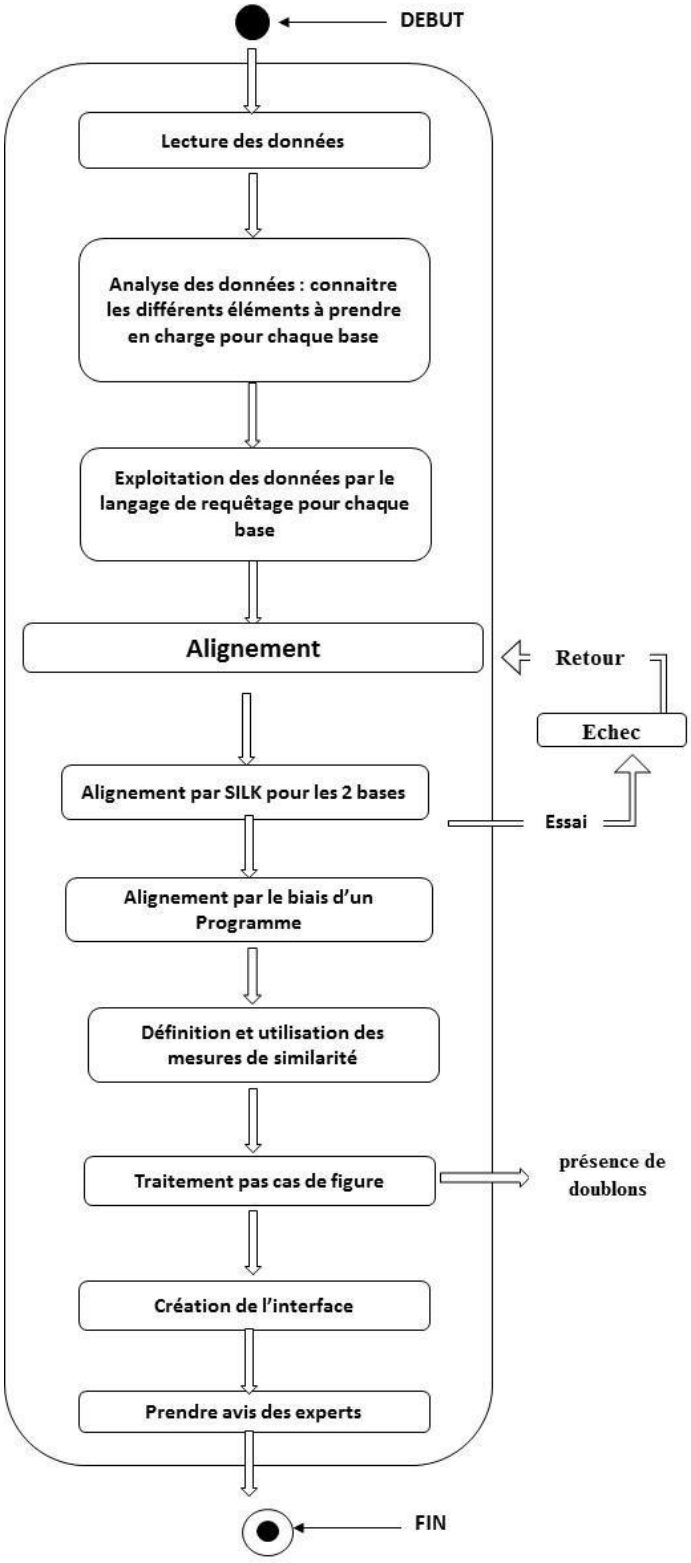

Figure 15 :Diagramme d'état-transition

d. Diagramme de Gantt

Dans la gestion de projet on utilise un certain nombre de techniques traditionnelles de représentation graphique comme par exemple le digramme développé par Henry Gantt, dit, diagramme de Gantt. Ce digramme représente des tâches de projets en fonction du temps, il s'exprime par des barres horizontales et affiche toutes les taches mises en relation avec le chemin critique. Son objectif c'est de définir la durée d'un projet, les taches critiques ainsi que les marges qu'on peut avoir par rapport à toutes les taches.

La date de début des travaux a été fixée pour le 23/01/2019. Le tableau suivant (tab1) représente les différentes taches abordées dans note projet :

	Nom	Durée	Début	Fin
1	**Projet TER**	**118 jours?**	**24/01/19 08:00**	**21/05/19 17:00**
2	étude bibliograhique	22 jours?	24/01/19 08:00	14/02/19 17:00
3	Extraction et exploitation des données	42 jours?	15/02/19 08:00	28/03/19 17:00
4	**Alignements des termes communs**	**28 jours?**	**09/04/19 08:00**	**06/05/19 17:00**
5	Alignement avec SILK	16 jours?	09/04/19 08:00	24/04/19 17:00
6	Alignement avec code	13 jours?	24/04/19 08:00	06/05/19 17:00
7	Echange avec les expert	1 jour?	06/05/19 07:00	06/05/19 17:00
8	Interface	39 jours?	07/04/19 07:00	15/05/19 17:00
9	Validation par l'expert	3 jours?	15/05/19 07:00	17/05/19 17:00
10	Rédaction du rapport	4 jours?	18/05/19 08:00	21/05/19 17:00

Figure 16 : Tableau explicatif des taches et leurs durées

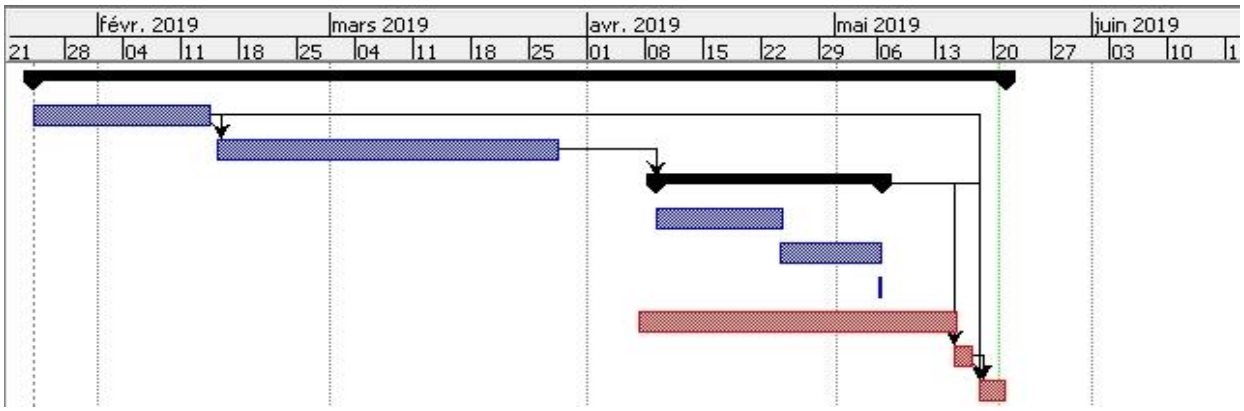

Figure 17 : Diagramme de Gant

CONCLUSION

Ce projet est loin d'être une utopie, il représente une opportunité qui nous a permis de contribuer à la réalisation d'une tâche très importante au sein d'une grande institution qui est la philharmonie de paris. Une participation unique et enrichissante permettant de nous former au monde professionnel et au monde de la web sémantique.

Même si nous n'avons pas pu accomplir toutes les tâches (5/7 cas de figure accompli) mais nous pouvons inscrire quelques tâches dans la section des perspectives ou travaux à venir :

- **Cas de figure** : *un concert qui doit se jouer plusieurs jours de suite mais qui n'a été enregistré qu'une seule fois (plusieurs dates associées à M26, une seule à F31 / titre légèrement différent).*

Pour ce cas, nous avons pu extraire grâce aux requêtes les résultats pour chaque base néanmoins, notre programme d'alignement fonctionne « par couple » c'est-à-dire que la sortie accepte pour chaque attribut (titre) un seul attribut(date) et non plusieurs en même temps, or que dans le résultat d'alignement de ce cas, les dates doivent apparaitre plusieurs fois pour un seul titre pour la classe M26.

- **Cas de figure :** *concert également présent dans les données de Radio France***.**

Il est conseillé d'explorer les bases de données de Radio France pour ressortir avec un résultat plus fiable grâce à la classe F22 de la base de Radio France qui est liée à la classe F31 de la base Philharmonie de Paris via la sous-classe R66.

- Demander aux experts d'améliorer et de corriger le tableau des cas de figure comme décrit auparavant dans : V. Alignement et similarité.

REFERENCE BIBLIOGRAPHIQUE

Algorithme Implémentation/Strings/Levenshtein Distance [en ligne]:

https://en.wikibooks.org/wiki/Algorithm_Implementation/Strings/Levenshtein_distance

https://stackoverflow.com/questions/39332845/iterative-version-of-damerau-levenshtein-distance

https://www.developpez.net/forums/d1514493/general-developpement/algorithme-mathematiques/algorithmes-structures-donnees/distance-levenshtein-tableau-chaines/

http://rosettacode.org/wiki/Levenshtein_distance

https://blog.madadipouya.com/2015/04/07/finding-similarity-percentage-between-strings/

https://fr.wikipedia.org/wiki/Mesure_de_similarité.

Données et les schémas sur le Web?,1er Janvier 2012

https://web-semantique.developpez.com/faq/?page=gen

https://fr.wikipedia.org/wiki/Web_s%C3%A9mantique

Fabien Gandon, Catherine Faron Zucker et Olivier Corby. Le Web sémantique : comment lier les

Konstantin Todorov, Ontology Matching and Data Linking, Février 2018.

Le tutoriel SPARQL. 27 Avril 2011.

https://websemantique.developpez.com/tutoriels/jena/arq/introduction-sparql

Manel Achichi & Konstantin Todorov, Interconnexion de Données du Web avec SILK, 2017.

Silk the Linked Data Integration Framework:

http://silkframework.org/

SPARQL Query Language for RDF

https://www.w3.org/TR/rdf-sparql-query/

https://web-semantique.developpez.com/tutoriels/jena/arq/introduction-sparql/

Liens GitHub :

https://github.com/Koched92/Terminology-and-conceptual-alignments

https://github.com/kahinaMenzou?tab=repositories

https://github.com/radiaips?tab=repositories