Introduction:

Ce chapitre présente notre choix de l'approche convenable pour faire fasse aux défis et a la problématique de notre sujet . On commence par l'architecture générale , description du problème et le détail des trois étapes suivi par des exemples explicatifs extraits de l'ontologie DBpedia .

3.1. Description :

Pour un aperçu global , notre système consiste à traduire ou transférer automatiquement des requêtes mots-clés en des requêtes formelles, afin de permettre aux utilisateurs finaux de se familiariser avec l'utilisation des mots-clés et d'utiliser ces mots-clés pour effectuer des recherches sémantiques. Nous avons adaptés les idées spécialement de l'approche SPARK [WAN,XIO,YU 07] et le travail remarquable du Docteur Bellatreche Abdelkrim et son équipe [LAT, LEH, BEN 09].

La recherche sémantique est un processus complexe qui devrait être achevé sur plusieurs étapes:

- Un processus d'annotation sémantique (Mapping des mots clés aux éléments de la Base de Connaissances BC)
- La phase de traitement de la requête (Construction des graphes requêtes)
- Une phase qui consiste à classer les résultats obtenus (Classement pour évaluer la pertinence.)

Après l'opération du prétraitement qui a comme rôle l'indexation automatique des entités de la base de connaissances pour obtenir un index très utile pour les étapes de notre travaille. L'approche appeler brièvement (*méthode de construction des requêtes formelles*), prend en entrée les mots-clés, et renvoie en sortie une liste classées de requêtes conjonctives SPARQL (voir figure 3.1).

Il est important de permettre aux utilisateurs d'effectuer des recherches sémantiques simplement en entrant des requêtes à base de mots-clés. Pour adapter la recherche par mots-clés à la recherche sémantique, nous avons à sur monter les obstacles suivants :

- ✓ Le fossé vocabulaire *:* Les utilisateurs du web traditionnel généralement n'ont pas de connaissances sur le contenu et la structure de la base de connaissances (annotations et ontologies sous-jacentes).

- ✓ Le manque de relations : les relations entre les ressources de la base de connaissances sont exigées pour être explicitement énoncées dans les requêtes formelles, qui sont souvent manquantes dans les requêtes mots-clés des utilisateurs.
- ✓ Classement des Requête: En raison de l'ambigüité de la recherche par mots clés, il peut y avoir de multiples requêtes formelles produites à partir d'une requête mots-clés.

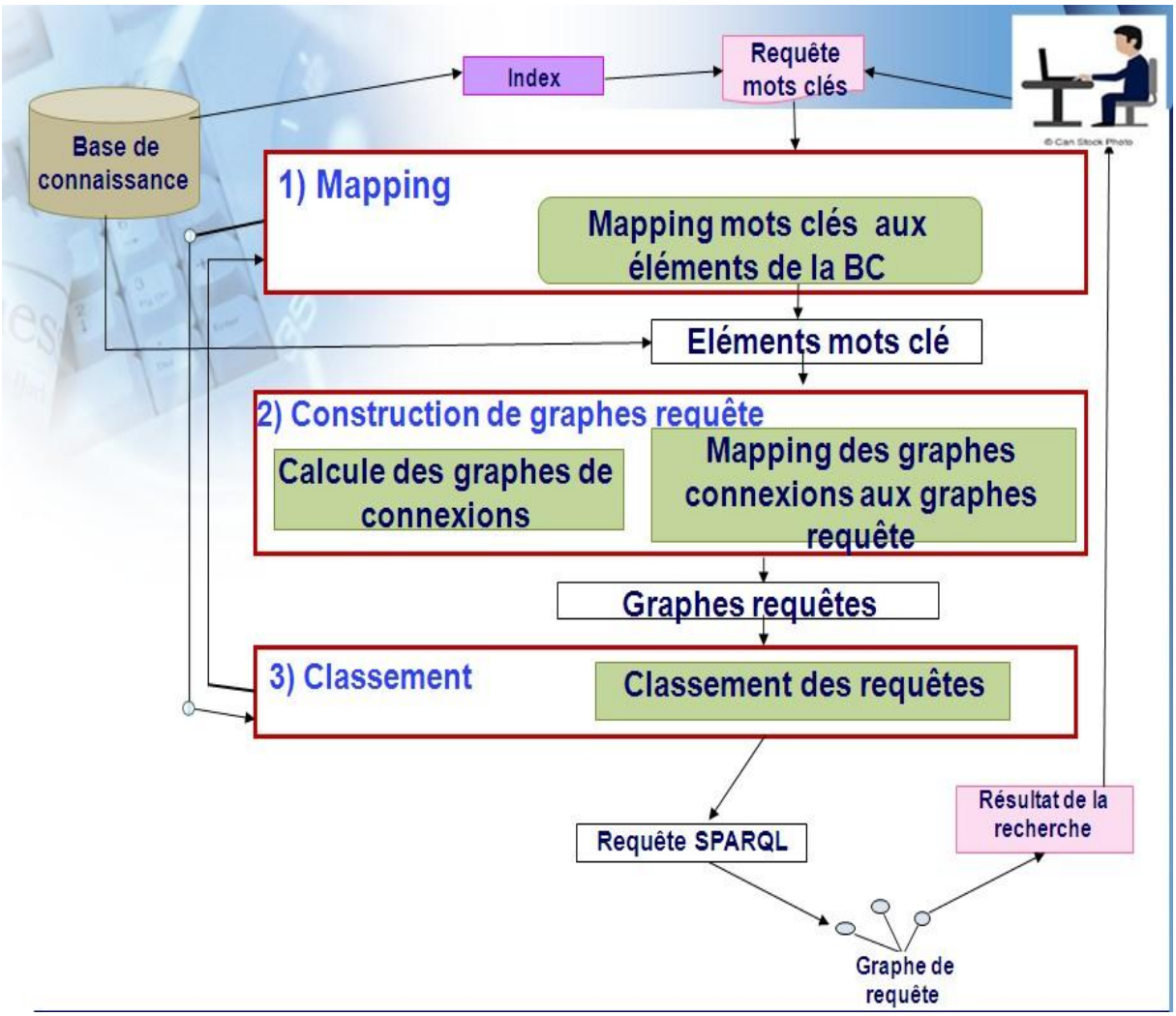


Figure 3.1 : Architecture générale de l'approche.

3.2. Présentation du problème :

Dans une base de connaissances RDF, une requête sémantique correspond à un graphe requête avec des nœuds objets et des arcs propriété. Ainsi, nous pouvons réduire le problème de la traduction des requêtes mots-clés en requêtes formelles conjonctives au

problème de la construction des graphes requêtes équivalent. Pour clarifier le problème, nous donnons la définition formelle comme suit:

- *Base de connaissances (B.C) D:(C, I, L, R, t)* est un graphe orienté *GD*

  où :

  C définit des ensembles de *concepts (classes)*

  R définit des ensembles de *relations (propriétés).*

  I définit des ensembles d'*individus* (r*essources*)

  L définit des *littéraux*.

  t une fonction $t:(C \cup I) \times (C \cup I \cup L) \rightarrow R$ définit tous les *triples* de *D.*

  En outre, nous utilisons le symbole éléments (entités) $\{e\}:\{C \cup R \cup I \cup L\}$ pour représenter toutes les concepts (classes), individus (ressources), relations (propriétés) et les littéraux.
- *Requête mots-clés : Qu*: est un ensemble de mots-clés *{k1,……..,kn}.*
- *Requête formelle : Qs: (C', R', I', L', V, t')* de *D* est un graphe *GF* subsumé par *GD.*

  où *:*

  V définit l'ensemble des *nœuds variable*.

  t' une fonction $t' (I' \cup C' \cup V) \times (I' \cup C' \cup V \cup L') \rightarrow R'$ définit l'ensemble des *triples* de *Qs.*

Exemple (1):

On navigant sur le web et spécialement on basant la recherche sur l'ontologie DBpedia , nous avons choisis une partie concernant les œuvres écrites (les livres) . Le graphe construit est représenté ci-dessous (voir figure 3.2):

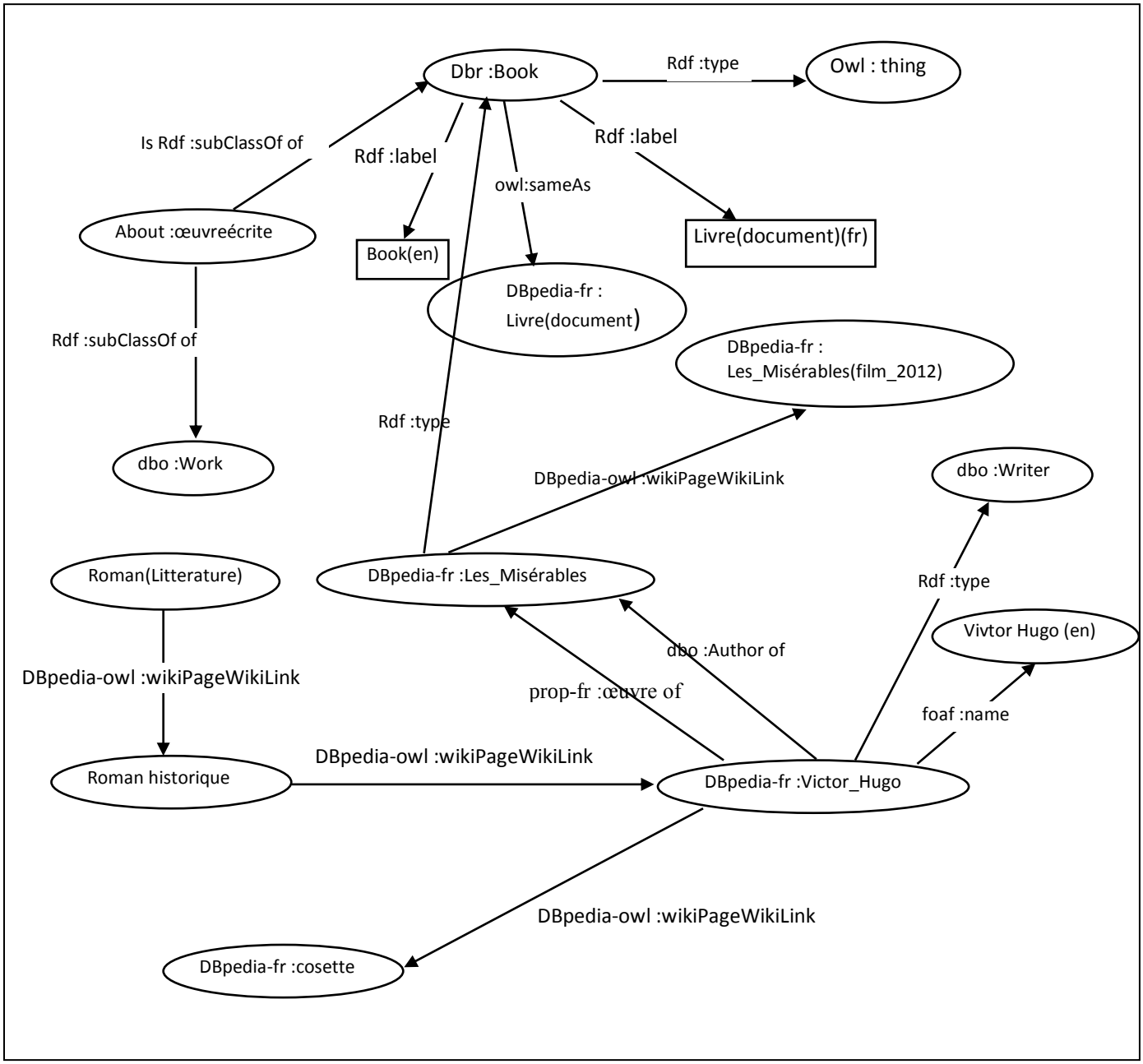


Figure 3.2 : Le Graphe Conceptuel extrait de l'ontologie DBpedia

Préfixes : (Les URIs)

Book → http://DBpedia.org/resource/Book

rdf :type → http://www.w3.org/1999/02/22-rdf-syntax-ns#type

rdfs :label → http://DBpedia.org/resource/Book

owl :thing → http://www.w3.org/2002/07/owl#Thing

DBpedia-Fr:Livre_(document) → http://fr.DBpedia.org/resource/Livre_(document)

rdf :label → http://www.w3.org/2000/01/rdf-schema#label

owl :sameAs → http://www.w3.org/2002/07/owl#sameAs

DBpedia-owl :WikiPageWikiLink → http://DBpedia.org/ontology/wikiPageWikiLink

Roman(litterature) → a#subClassOfhttp://DBpedia.org/ontology/WrittenWork

Roman(litterature)→ http://fr.DBpedia.org/resource/Roman_(litt%C3%A9rature)

Roman historique → http://fr.DBpedia.org/resource/Roman_historique

DBpedia-Fr :Victor_Hugo → http://fr.DBpedia.org/resource/Victor_Hugo

DBpedia-fr :Les_Misérables → http://fr.DBpedia.org/resource/Les_Mis%C3%A9rables

DBpedia-fr :Les_Misérables(film_2012)→

http://fr.DBpedia.org/resource/Les_Mis%C3%A9rables_(films,_1934)

Is Rdf :subClassOf of → http://www.w3.org/2000/01/rdf-schema#subClassOf

Rdfs :subClassOf → http://www.w3.org/2000/01/rdf-schema#subClassOf

About :œuvreécrite → http://DBpedia.org/ontology/WrittenWork

dbo :Work → http://DBpedia.org/ontology/Work

dbo :Writer → http://DBpedia.org/ontology/Writer

foaf :name → http://xmlns.com/foaf/0.1/name

Is prop-fr :œuvre of : → http://fr.DBpedia.org/property/oeuvre

DBpedia-fr :cosette→ http://fr.DBpedia.org/resource/Cosette

Is dbo :Author of → http://DBpedia.org/ontology/author

3.3 - Etapes du système :

3.3.1- Mappage des mots clés aux éléments de la B.C :

- **Définition** :

Le mappage *(mapping)* en informatique est l'association des données appartenant à un ensemble (modèle logique de données, base de données de production, champ source) avec les données appartenant à un autre ensemble (modèle physique de données, entrepôt de données, champ cible), de manière que les données du premier ensemble puissent se substituer à celles du second ensemble, ou encore que l'on puisse passer harmonieusement des premières aux secondes.

Les recherches montrent que le fait de comparer deux choses produit un savoir, Miner [LAT,LEH,BEN 09] écrit « *Il est manifestement impossible de comparer ce qui est identique. Des différences doivent exister ou alors nous identifions plus que nous comparons. De la même façon, si les différences sont trop grandes, la comparaison devient infaisable, les résultats logiques ou pratiques ne satisfont pas*.» Ainsi, il semble que comparaison et similarité soient étroitement liées.

Ce qui est important dans le mappage est la similarité sémantique qui exprime une liaison entre deux concepts est une capacité abstraite de l'homme, les machines n'ont pas la faculté de l'interpréter. Il est évident pour un humain que les concepts « *stylo* » et « *papier* » sont liés beaucoup plus que « *température* » et « *chaise* ». Cet état de fait, difficile à formaliser sans recours aux ressources sémantiques : les ontologies permettent de montrer les liens entre les concepts (hyperonymie, antonymie, etc.).

Le calcul des mesures de similarité entre concepts d'une ontologie, constitue aussi un autre défi auquel diverses solutions existent et qui sont classées en deux grandes catégories :

- ✓ Approches basées sur la structure de l'ontologie (edge counting): C'est une technique de calcul basée sur les liens taxonomiques « est un » « is a ». comme exemple " mesure de Wu-Palmer ".
- ✓ Approches utilisant le contenu informatif des concepts : comme exemple " mesure de Resnik" .

Dans cette étape (voir figure 3.3): nous détectons le meilleur concept pour chaque terme. Chaque terme peut avoir plusieurs sens, nous allons essayer de déterminer le meilleur sens (concept) dans le contexte (requête utilisateur) ou se trouve ce terme. Pour cela, une opération de désambigüisation est nécessaire.

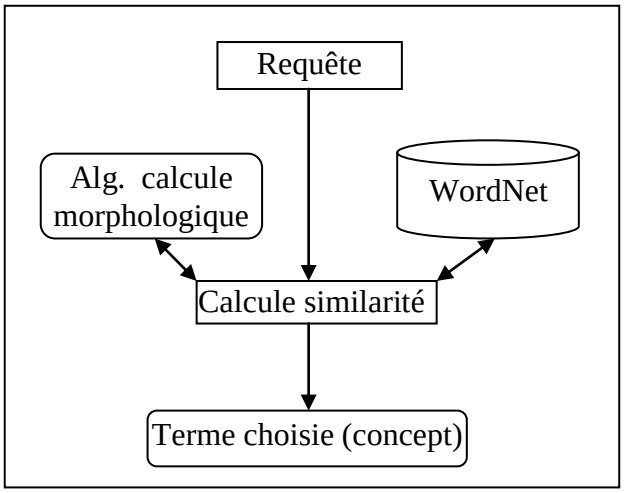


Figure 3.3 : Calcule de similarité.

- **Le Détail :**

Le mappage (*mapping*) comme étape représente " *une cartographie temporaire* " , son but est de trouver les éléments de la B.C. (concepts, individus, relations et littéraux)

correspondants à chaque mot clé de la requête de l'utilisateur. Les noms et les étiquettes (labels) des éléments de la B.C. utilisées pour le mappage.

Dans notre système, deux types de méthodes de mappage peuvent être utilisées:

- ✓ Le mappage morphologique: emploie les techniques de comparaison de chaîne de caractères, tels que le stemming, Sub-String et Levenshtein-distance pour trouver les mots morphologiquement semblables, Par exemple : « author » et « author of ».
- ✓ Le mappage sémantique: se base sur la similarité sémantique et il utilise principalement des dictionnaires généraux comme WordNet [10],, afin de trouver les mots sémantiquement approprié (les synonymes, …… etc.), Par exemple : « author » et « writer ».

Nous assignons un score prédéfini ***Sm**(e) : (Sm(e) ∈ [0,1])* à chaque méthode de mapping pour déterminer la qualité du mapping. En règle générale, le score du mapping direct est supérieur au mapping à base des synonymes. En conséquence, le mapping des mots clés aux éléments de la B.C associe à chaque terme de la requête mots-clés un ou plusieurs sens selon la base de connaissances. En particulier, le mapping peut être définie comme fonction $f : Qu \rightarrow D$qui pour chaque terme de la requête mot-clés retourne un ensemble des éléments de la B.C.

(Pour des fins pratiques le moteur de recherche Lucene [11], est effectivement utilisée comme moteur d'index et de recherche). C'est à dire, les noms et les étiquettes des éléments de la BC. sont indexées, et en utilisant la fonction de recherche de Lucene, une requête est générée pour chaque mot-clé entré. Le moteur retourne les éléments de la B.C classées en fonction de leurs similarités syntaxique aux mots clés respectifs. Ces éléments appelés « éléments mots clés » ***Dm** : Dm = {f(ki) / Qu=(k1,....., Kn)}* , seront alors introduit dans l'étape de la construction des graphes requêtes.

Suite de l'Exemple (1):

On exploite le même exemple de la Figure 3.2 , et nous prenons la phrase suivante (en anglais), qui représente une requête d'un utilisateur :" **Author Miserable book** ". Nous avons trois noms qui ont des synsets dans WordNet . T = { Author, Miserable, Book }. Chaque terme de T a plusieurs sens (synsets) dans WordNet . La figure 3.4 montre le résultat de mappage entre les éléments de T et les éléments dans le graphe de la figure 3.2. les valeur des similarités trouvées sont présenté dans le tableau 3.1.

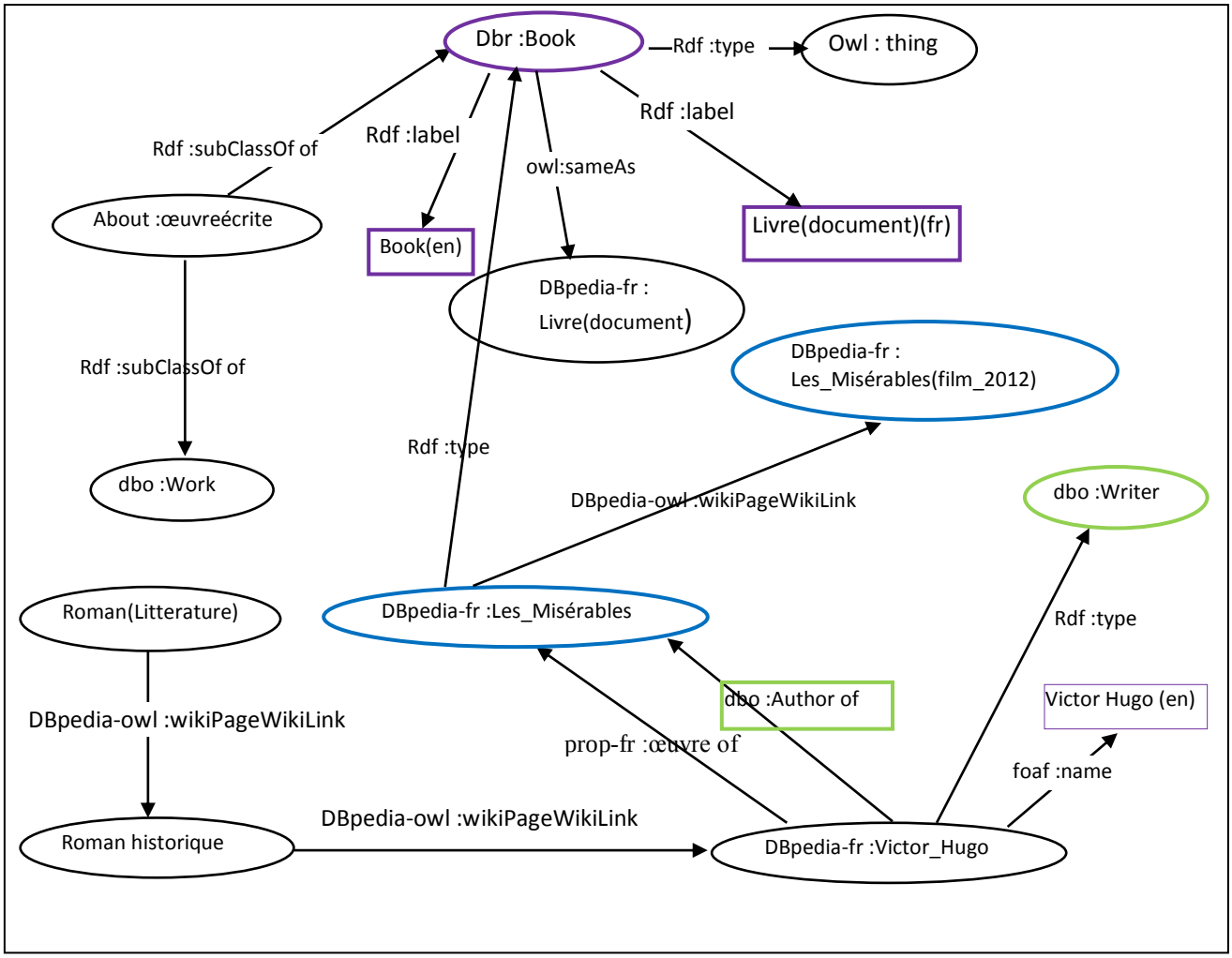


Figure 3.4 : Exemple de mapping

Mots clés	**Ressource**	**Mesure utilisées**	**Valeur de similarité**
Book	dbr :Book (c)	Levenshtein	1.00
Book	Book(en) (L)	Levenshtein	1.00
Misérable	dbfr :Les_Misérables(film_2012)(I)	TagLinkToken	0,49
Misérable	dbfr : Les_Misérables (I)	JaroWinkler	0.32
Author	dbo :Author of (P)	MongeElkan	0,66
Author	dbo :Writer (P)	WordNet	0,37

Tableau 3.1 : Les valeurs des similarités trouvées entre les mots clés et les ressources**.**

3.3.2 . Construction des Graphes Requête :

Cette opération représente la construction des graphes requête candidats avec les éléments mots clés trouvés dans l'étape précédente. Tout d'abord, les éléments mots clés sont répartis en différentes ensembles de requêtes. Le but de cette répartition est d'attribuer un sens à chaque requête formelle par l'énumération de toutes les combinaisons possibles des différents sens pour chaque terme ***ki*** de ***Qu***. Ensuite, un certain nombre d'algorithmes sont appliqués pour construire de possible graphes requêtes pour chaque ensemble de requêtes.

a- Calcul des graphes de Connexions :

A ce stade , l'exploration des relations se limite aux relations de type: (i , c) , (i_1, R, i_2) et (i , U, j) , où : I, i_1, $i_2 \in \mathcal{I}$, $j \in \mathcal{L}$, $c \in C$, $R \in \mathcal{R}$ et $U \in \mathcal{U}$

avec : $\mathcal{I}$(Individus), $\mathcal{L}$(littéraux), C (Concepts), $\mathcal{R}$ (propriétés d'objets) et $\mathcal{U}$ (propriétés de données).

Par l'utilisation de ces axiomes, nous explorons tous les éléments de la B.C liées aux éléments de ***Dm*** identifiés dans la première étape selon l'algorithme (1) (Figure 3.5)

Le processus du calcule des graphes de connexions se déroule comme suit :

- **Explorer les liens entre les éléments de la B.C :**

Elle comprend la traversée des voisins pour chacun des éléments de ***Dm***. Ensuite, selon le type de l'élément particulier ***em*** $\in$ ***Dm***, différents parcours sont effectués pour construire un graphe connectant ***em*** à tous ces voisins sur une distance spécifique ***d***. Soit un concept, tous les individus sont recherchés par l'intermédiaire de l'axiome ***(i, c)***. Compte tenu une relation, les axiomes ***(i_1, R, i_2)*** et ***(i, U, j)*** sont utilisées pour naviguer vers les individus et les littéraux, respectivement.

La Figure 3.5 montre le pseudo-code de l'algorithme pour le passage récursif d'un individu ou d'un littéral particulier à ses voisins concepts, individus et littéraux. La valeur de ***d*** est réduite de (**1**) à chaque étape de récusions pour assurer à ce que ce passage sera limité à une certaine distance. En fin de compte, nous obtenons un graphe ***g*** contenant tous les éléments de ***D*** qui ont une distance graphique qui ne dépasse pas ***d*** au moins à un des éléments de ***Dm***.

Ainsi, quelques éléments découverts peuvent vraiment ne pas être nécessaires pour connecter des éléments de ***Dm***. Par conséquent, à partir de ce graphe, on choisis seulement les chemins où le premier et le dernier sommet correspond est un élément de ***Dm***. En particulier,

Algorithme Exploration-BC (Dm, d) :

Entrée Dm ensemble des éléments mots clés et d distance du passage.
Sortie graphe contenant tous ou certaines éléments de D.
Initialisation d'un nouveau graphe vide g.
Pour e ∈ Dm
Faire Si e est un concept
Alors Pour Tous i étant instance de e
Faire Passé (i, d, 0, g)
Sinon Si e est une propriété d'objet (Individu-Individu)
Alors Pour Tous i, j avec <i, e, j>∈ D
Faire Passé (i, d, 0, g), Passé (j, d, 0, g)
Sinon Si e est une propriété de données (Individu-Littérale)
Alors Pour Tous i, j avec <i, e, j>∈ D
Faire Passé (j, d, 1, g)
Sinon Si e est un Individu
Alors Passé (e, d, 0, g)
Sinon Si e est un Littérale
Alors Passé (e, d, 1, g)
Retourné **g.**

Figure 3.5 : Algorithme (1) Exploration de la B.C [LAT,LEH, BEN 09]

L'algorithme précédant fait des appelles à l'algorithme (Passé) présenté ci-dessous **:** (Voir figure 3.6):

Algorithme Passé (i, d, t, g) :

Entrée i un individu ou un littérale à explorer, d distance du passage, g un graphe intermédiaire.

Sortie graph g mis à jour contenant les éléments connecté à i sur une distance d.

```
Si i n'est pas marqué comme visité et  d > 0
    Alors Marquer i comme visité dans D
Si t=0
  Alors Ci :={c/ i instances de c}
  Ajouter Arête (i, type, c) dans g pour tout c ∈ Ci
  P :={(i, p, j) / <i, p, j>∈ D}
  Pour Tous (i, p, j) ∈ P
     Faire Si j n'est pas marqué comme visité dans D
          Alors Ajouter nouveau Arête (i, p, j) dans g
        Si j est un individu
          Alors  Passé  (j, d-1, 0, g)
          Sinon  Passé  (j, d-1, 1, g)
Si t=1
  Alors P :={(j, p, i) / <j, p, i>∈ D}
  Pour Tous (j, p, i) ∈ P
     Faire Si j  n'est pas marqué comme visité dans D
        Alors Ajouter nouveau Arête (j, p, i) dans g
     Si  j est un individu
        Alors  Passé  (j, d-1, 0, g)
```

Figure 3.6 : Algorithme (2) Passé [LAT, LEH, BEN 09]

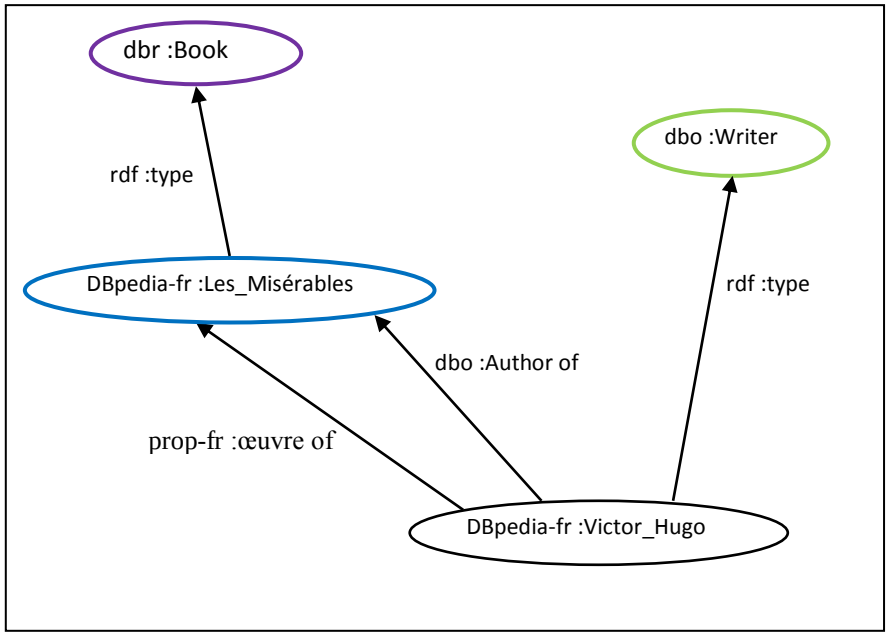


Figure 3.7 : Graphe résultat d'application de l'algorithme d'exploration sur le graphe de la Figure 3.4

Exemple : Si on utilise l'algorithme d'exploration avec comme entrées l'ensemble Dm={« dbr :Book », « dbfr : Les_Misérables », « dbo :writer » } et le graphe de la Figure 3.4 , alors on obtient le graphe présenté dans la Figure 3.7 . Par contre si on remplace « dbfr : Les_Misérables » par « dbfr :Les_Misérables (film_2012) » alors aucun graphe sera trouvé .

- **Calcul des Graphes de Connexions**:

Une requête peut être dérivée quand tous les éléments ***Dm*** identifiés dans la première étape sont connectés. En fusionnant tous les chemins ***P*** calculés dans l'étape précédente, nous obtenons toutefois un graphe qui peut contenir plusieurs différents sous-graphes connectant tous les éléments de ***Dm***. Le sous-graphe connectant les éléments de ***Dm*** sont calculés par une procédure récursive montrée dans l'algorithme de la Figure 3.8 . L'entrée de l'algorithme est l'ensemble des chemins ***P*** calculés précédemment. La récusions commence par la sélection d'un certain chemin connectant deux sommets arbitraires et entre d'autres récusions pour ajouter des sommets supplémentaires. De cette façon, tous les sous-graphes en forme d'arbre possibles connectant des éléments de ***D*** sont déterminés.

Algorithme Calcule-graphe-connexion (P, C, R, G, g) :

Entrée l'ensemble de chemins P calculer par DFS pour tous les éléments mots clés de Dm

Sortie Tous les sous-graphes connectant les sommets de Dm

```
Si G= Ø
  Alors Pour {i, j}  ⊆R
        Faire G = Nouveau Graphe
        Pour chaque chemin p entre i et j (calculer par DFS)
              Faire ajouter (i, p, j) dans G
              Calculer-graphe-connexion (P\p, C∪{i, j}, R\{i, j}, G)
  g=g ∪ G
  Sinon Pour i ∈  R
       Faire Pour j ∈ C
              Faire Pour chaque chemin p entre i et j
                    Faire ajouter (i, p, j) dans G
                    Calculer-graphe-connexion (P\p, C∪{i}, R\{i}, G)
```

Figure 3.8 : Algorithme (3) calcule des graphes de connexions [LAT,LEH, BEN 09]

Suite de l'Exemple: Le premier graphe de connexions (voir figure 3.9):

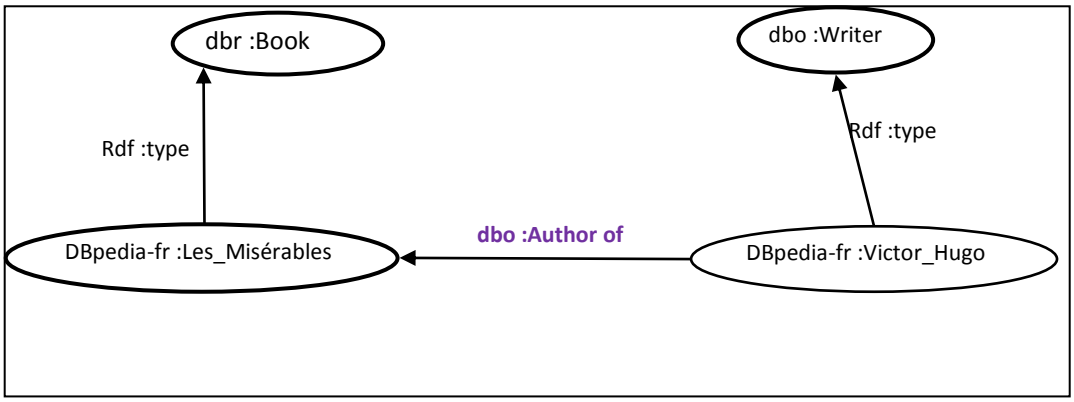


Figure 3.9 : Graphe de Connexions 1

Le deuxième graphe de connexions (voir figure 3.10) :

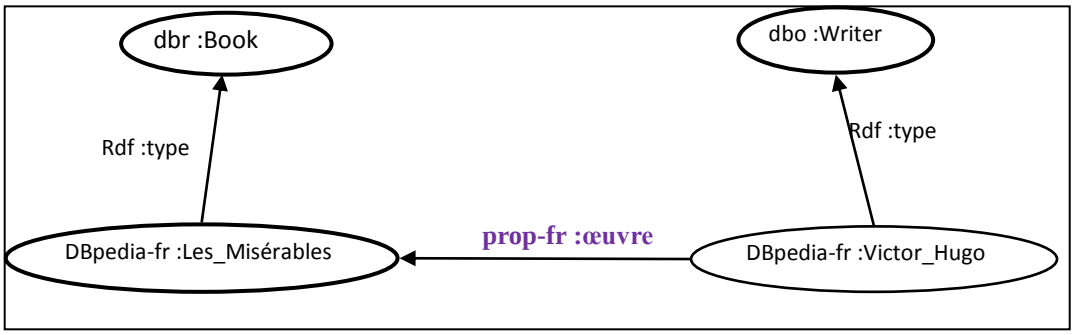


Figure 3.10 : Graphe de Connexions 2

b- Mappage des graphes de Connexions aux graphes requête :

Enfin, Chacun des graphes de connexion ***Gc*** trouvé précédemment est ensuite traduit en un graphe requête comme suit :

Une arête dans ***Gc*** de la forme *type* ***(vi, vc)*** (représentant la connexion *(i, c)*) est considérée comme arête dans le graphe requête de la forme *Type****(x, C)***,

où ***:*** ***vi*** est un sommet construit au moyen d'un individu,

vc est construit en utilisant un concept,

x est un individu ou une variable.

Lorsqu'un individu de ***vi*** correspond à certains éléments ***em*** $\in$ ***Dm***, il est considéré comme une constante, sinon une variable est utilisé pour ce terme. Comme le même individu peut être utilisé dans beaucoup d'arêtes, la même variable doit être utilisée pour le même individu. Les arêtes construit avec les axiomes de la forme ***R (vi, vj)***, où ***vi*** **(vj)** est construit en utilisant soit un individu ou un littérale (connexions (i_1, R, i_2) *et (i, U, j))*.

Ces arêtes sont considérés comme arêtes dans le graphe requête de la forme ***R(x, y)***, où un sommet construits au moyen d'un individu est associé à une variable ou une constante comme décrit ci - dessus. Lorsque ***vi*** **(vj)** est construit en utilisant un littérale, il est tout simplement considéré comme nœud associé à une constant.

Dans notre exemple de la Figure 3.7, il existe deux graphes de connexions donc deux graphes requêtes.

Le premier graphe de requete (voir figure 3.11) :

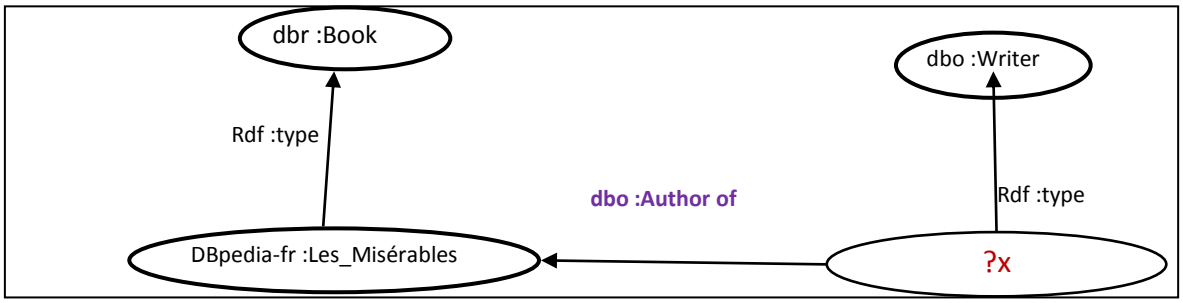


Figure 3.11 : Graphe de requête 1

Le deuxième graphe de requête (voir figure 3.12) :

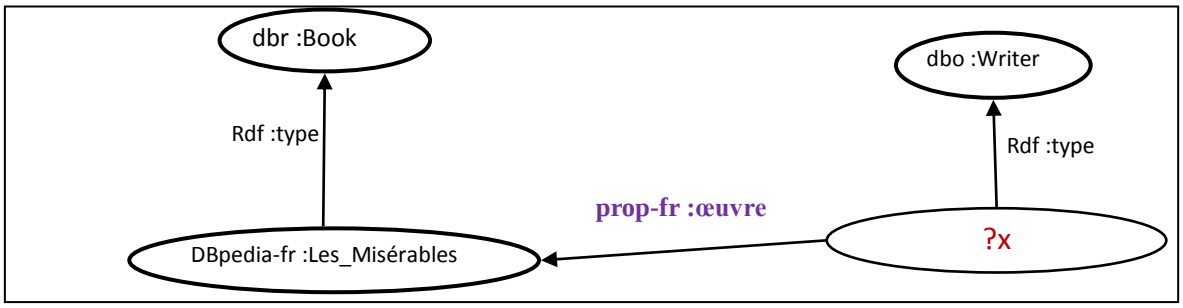


Figure 3.12 : Graphe de requête 2

Puisque SPARQL est un langage d'interrogation basé sur les motifs de graphes, il est facile de convertir un graphe requête en une requête SPARQL (voir figure 3.13 et figure 3.14)

```
SELECT    ? x
WHERE  {
  ? x   Type   dbo:Writer
  ? xdbo :Author_ofDbpedia-fr :Les_Misérables
Dbpedia-fr :Les_Misérablesrdf :type    dbr :book
 }
```

Figure 3.13 : La requête Sparql du Graphe Requête (1)

```
SELECT    ?x
WHERE  {
 ? xrdf:type   Writer
? x   oeuvreDbpedia-fr :Les_Misérables
Dbpedia-fr :Les_Misérablesrdf :type    dbr :book
  }
```

Figure 3.14 : La requête Sparql du Graphe Requête (2)

3.3.3. Classements des requêtes :

Après l'étape de la construction des graphes requêtes, plusieurs requêtes formelles candidates peuvent être produites à partir de la requête mots-clés initiale correspondant à toutes les interprétations possibles des mots clés. Un problème se pose: comment prendre la requête formelle qui correspond aux besoins de l'utilisateur? Le classement des requêtes est utilisé pour résoudre le problème (voir figure 3.15) .

Le calcule du score des réponses a été largement discuté dans le domaine de la recherche d'information. Dans le contexte des données structurées en graphe, des mesures de proximité sémantique ont été proposées dans la littérature (une douzaine) utilisant des structures de réseaux sémantiques ou hiérarchiques :

- Mesure basée sur le chemin (path based measures) entre les deux concepts à comparer telles que définies par Rada, Leacock ou Jiang en 1997.
- Mesure basée sur la notion de contenu d'information (Information Content ou IC) telle que celle définie par Wu et Palmer et Resnik.
- Mesure basée sur une combinaison du chemin et du contenu d'information (Mesure hybride) définie par D. Lin en 98.

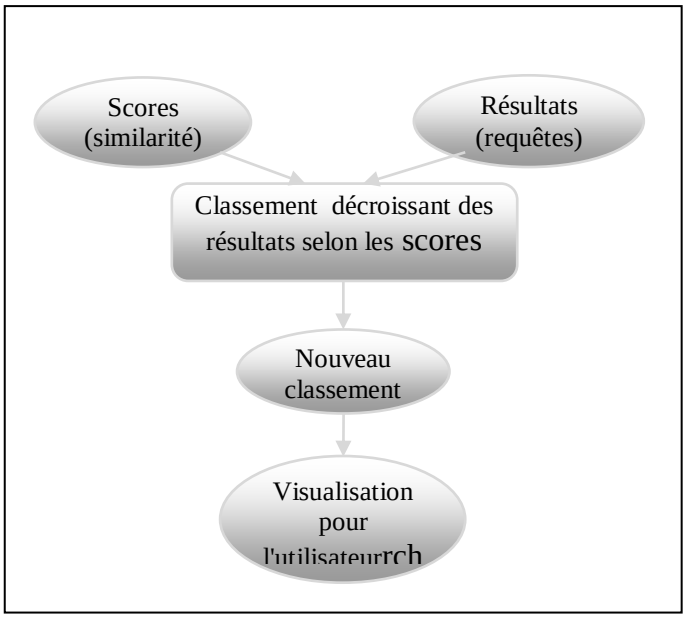


Figure 3.15 : Classement des requêtes.

A chaque graphe de requête est associé un score qui permet de le pondérer. Les graphes sont ensuite convertis en requêtes et exécutés dans l'ordre décroissant de ces scores. Ce score **S** est donné par la formule :

$$S = \sum_{i=1}^{n} Sei$$

Avec $\mathbf{Se_i}$ le score du l'élément i (valeur de similarité trouvée dans le procédure de mappage sinon il est égale à 0). La figure 3.16 présente l'algorithme de classement des requêtes.

Algorithme de Classement (R , Rc) :

Entrée R ensemble des requêtes trouvées (Ri / Ri ∈ R)

Sortie Rc liste des requêtes classées selon leurs scores.

Initialisation du score et la somme som.

Pour tous élément i ∈ Ri

Faire : som := som + Sei

positionner Ri dans Rc selon le score.

Figure 3.16 : Algorithme (4) - Classement des résultats.

Suite de l'Exemple (1) : on prenant compte des résultats de notre exemple durant l'étape (1) et (2) , on peut appliquer l'algorithme de classement pour les deux requêtes et nous aurons les résultats illustrés dans le tableau 3.2 :

Requête Sparql	**Score Similarité**	**Classement**
Requête 1 (Figure 3.13)	0,51	1
Requête 2 (Figure 3.14)	0,37	2

Tableau 3.2 : Classement des requêtes formelles selon leurs scores

3.4. Conclusion :

Dans ce chapitre on a opté pour une approche de formalisation du problème de la construction de requête formelle qui vise à traduire les mots-clés en requêtes SPARQL pour satisfaire les fins sémantiques utilisateurs.

On a détaillé les trois étapes du système (mapping - construction - classement) avec justifié et accompagné par des exemples concrets.