Interrogation à base d'annotation sémantique

LATRECHE Abdelkrim[1], LEHIRECHE Ahmed[1], BENYAHIA Kadda[1]

[1]Laboratoire EEDIS, UDL de Sidi Bel Abbes, ALGERIE.

Résumé. Les approches de la recherche d'information (RI) actuelles ne saisissent pas formellement la signification explicite d'une requête à base de mots-clés mais fournissent une voie confortable pour l'utilisateur qui spécifie ces besoins en informations sur la base des mots-clés. La recherche sémantique promet de fournir des résultats plus précis que la traditionnelle recherche par mots-clés. Toutefois, les progrès de la recherche sémantique ont été retardés en raison de la complexité de ses langages de requête. Dans ce document, nous explorons une nouvelle approche pour l'adaptation des requêtes mots-clés pour pouvoir interroger le web sémantique en se basant sur les annotations sémantiques: l'approche traduit automatiquement des requêtes mots clés en des requêtes formelles, afin de permettre aux utilisateurs finaux familiarisés avec l'utilisation des mots clés d'utiliser ces mots-clés pour effectuer des recherches sémantiques. Un prototype de système a été mis en œuvre sur la lumière de cette approche. Etant donné une requête mots clés, notre système donne en résultats une liste classé de requêtes SPARQL en tant que résultat de la traduction. La traduction dans notre système se compose de trois grandes étapes: *Mapping des mots clés aux éléments de la B.C.*, *Construction des graphes requêtes* et C*lassement des requêtes*. Avec les premiers tests notre système a réalisé des résultats de traduction encourageant.

Mots-Clés : Web Sémantique, Recherche Sémantique, Annotation, Ontologie, RDF, SPARQL.

1 Introduction

La quantité d'information disponible sur internet est aujourd'hui gigantesque et sa croissance est exponentielle, le nombre d'usagers d'internet double chaque année. On estime que la taille du Web couvert par les moteurs de recherche est estimée à au moins 24 milliards de pages[1]. Mais la spécificité de telles sources d'informations les rend difficilement exploitables. La raison principale est que les documents sont fragmentés, dispersés, hétérogènes et sont souvent très peu structurés. Cependant, grâce aux efforts de la communauté du Web sémantique (W3C), une deuxième

[1] http://www.worldwidewebsize.com/

génération est établie dont la vision initiée en 1998 par Sir Tim Berners-Lee [2] a pour objectif de structurer les informations disponibles sur le Web. Pour cela, les ressources, textuelles ou multimédias, doivent être sémantiquement annotées par des métadonnées afin que les agents logiciels puissent les exploiter. La représentation explicite des contenus des ressources documentaires du Web est rendue possible grâce notamment aux ontologies qui proposent une compréhension commune et partagée d'un domaine, tant au niveau des utilisateurs humains qu'au niveau des applications logicielles [3]. Dans [9] l'annotation sémantique est définit comme "une représentation formelle d'un contenu, exprimée à l'aide de concepts, relations et instances décrits dans une ontologie, et reliée à la ressource documentaire source". Pour ce faire, le Web Sémantique fournit un ensemble de langages et de technologies pour la modélisation des ontologies et l'annotation sémantique des contenus documentaires en fonction de ces ontologies [4]. Les standards du W3C, tels que XML [5], RDF [6] et les schémas RDF (RDFS) offrent un format uniforme pour la description et l'échange du contenu du Web. D'autres efforts accomplis par des communautés commerciales et/ou académiques tels que DAML+OIL [7], OWL [8]. L'utilisation des annotations sémantiques, en recherche d'information est devenue une voie très explorée. Dans cette optique, plusieurs systèmes de recherche d'informations à base d'annotations et d'ontologies ont été proposés. Dans ce type d'outils de recherche, de manière générale, les pages Web sont (peut-être incomplète) des instances de certaines ontologies de domaine, et ils contiennent des données sémantiquement annotées selon les sous-tendent ontologie de domaine, pour plus de détails voir [11,12].

De même, que de nombreux langages de requêtes sémantiques (par exemple, RQL [16], RDQL[2], SquishQL [17] et SPARQL[3]) ont été proposés pour interroger ces annotations. En particulier le langage SPARQL, qui une recommandation du W3C dont la standardisation finale datte du 15 Janvier 2008. Toutefois, pour pouvoir utiliser ces langages sémantiques, les utilisateurs doivent maîtriser les représentations complexes de la logique formelle et être familier aves les ontologies sous-adjacentes. Ceci devient un fossé critique entre la recherche sémantique et les utilisateurs finaux. Par conséquent, 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 surmonter les obstacles suivants [24]: 1) *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). *2) 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. 3) *Classement des Requête*: En raison de l'ambiguïté de la recherche par mots clés, il peut y avoir de multiples requêtes formelles produites à partir d'une requête mots-clés.

Dans le contexte de Web sémantique il existe des travaux sur la traduction des requêtes mots-clés en requêtes sémantiques. Royo et al. [18] propose un mapping des

[2] http://www.w3.org/Submission/2004/SUBM-RDQL-20040109/

[3] http://www.w3.org/TR/rdf-sparql-query/

mots-clés à leurs correspondant synsets de WordNet.. Bernstein et al. [19] explore les langages contrôlés fourni et les interfaces guidées par le langage naturel [26]. Du point de vue de l'affinement des requêtes [20] [21], l'écart entre les besoins en information des utilisateurs et leurs requêtes sémantique est quantifié par la mesure de plusieurs types d'ambiguïtés des requêtes à travers les interactions progressives. La recherche à base de graphe [22] contribue également dans cette voie, par la construction des graphes requêtes par le biais de la navigation et de sélection sur l'ontologie. SemSearch [14], a une petite interface structurée de requête mots-clés pour cacher la complexité de la recherche sémantique. Avatar Semantic Search [23] est un prototype de moteur de recherche qui exploite les annotations dans le contexte de la recherche par mot-clé classique. Une autre application représentative de la recherche sémantique à base de mot clé est OntoLook [28]: un prototype de moteur de recherche à base de relation. Dans [10], les auteurs ont proposés une approche pour traduire les requêtes mots-clés en requêtes conjonctives DL en utilisons les connaissances disponibles dans une ontologie. Dans [24] la construction des graphes requêtes se base sur l'application de l'algorithme MST (Minimum Spanning Tree). Thanh et al. [25] présentent une approche de recherche par mot-clé sur les données structurées en graphe, en mettant l'accent en particulier sur le modèle de données RDF. Beaucoup de méthodologies pour décider des meilleures requêtes ont été proposés [26, 27, 24, 25].

Sur la base des travaux présentés ci-dessus, nous présentons une approche qui permet de traduire automatiquement des requêtes mots-clés en requêtes formelle SPARQL en vertu de la base de connaissances (base d'annotations sémantiques). Les étapes principales de traduction sont: 1) Le mapping des mots clés aux éléments de la BC, 2) Construction des graphes requêtes et 3) Classement des requêtes.

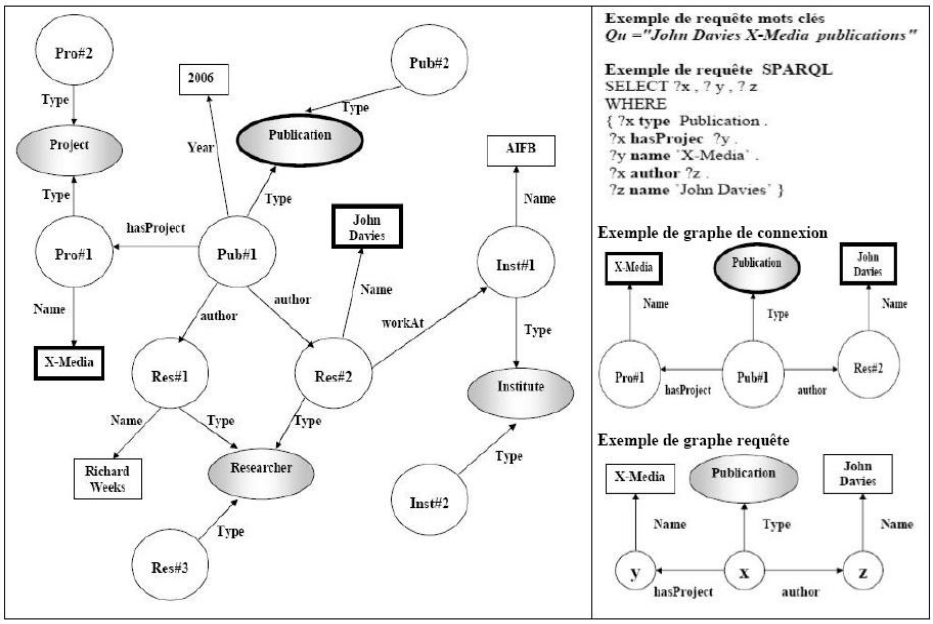

Fig. 1. Exemple de graphe de données RDF, graphe de connexion, graphe requête

De cette façon, les utilisateurs peuvent garder l'habitude de saisir les mots-clés pour interroger le Web sémantiques sur la base d'annotation sémantique d'une manière transparente, ce qui augmentent l'utilisation sociale de la recherche sémantique.

Le reste de ce document est organisé comme suit: section 2 définit le problème de la construction des requêtes formelles. Section 3 décrit les principales étapes de construction de requête formelle SPARQL. L'implémentation et les résultats expérimentaux sont présentés dans la section 4. Nous donnons la conclusion et les travaux futurs dans la dernière section.

2 Définition 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)* $D:(C, I, L, R, \tau)$ est un graphe orienté G_D où : *C et R* définis respectivement des ensemble de *concepts (classes)* et de *relations (propriétés). I* et *L* définis respectivement des ensembles *d'individus* (r*essources*) et de *littéraux*. La *fonction* $\tau:(C \cup I) X (C \cup I \cup L) \rightarrow R$ 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* Q_u est un ensemble de mots-clés $\{k_1,\ldots,k_n\}$.
- *Requête formelle* $Q_s:(C', R', I', L', V, \tau')$ de *D* est un graphe G_F subsumé par G_D. *V* est l'ensemble des nœuds variable. $\tau': (I' \cup C' \cup V) X (I' \cup C' \cup V \cup L') \rightarrow R'$ définit l'ensemble des triples de Q_s.

Dans la section suivante, nous illustrerons notre approche en détaille.

3 Aperçu de l'approche

L'infrastructure de notre système se compose de deux modules (fig. 2) : *Le module prétraitement* a pour rôle d'indexer automatiquement les entités de la base de connaissances pour obtenir un *index* qui sera utilisé dans l'étape de Mapping des mots clés aux éléments de la B.C. Alors que *Le module de construction des requêtes formelles* prend en entrée les mots-clés, et renvoie en sortie une liste classées de requêtes conjonctives SPARQL. Dans les sections suivantes, nous détaillerons ces déférentes étapes.

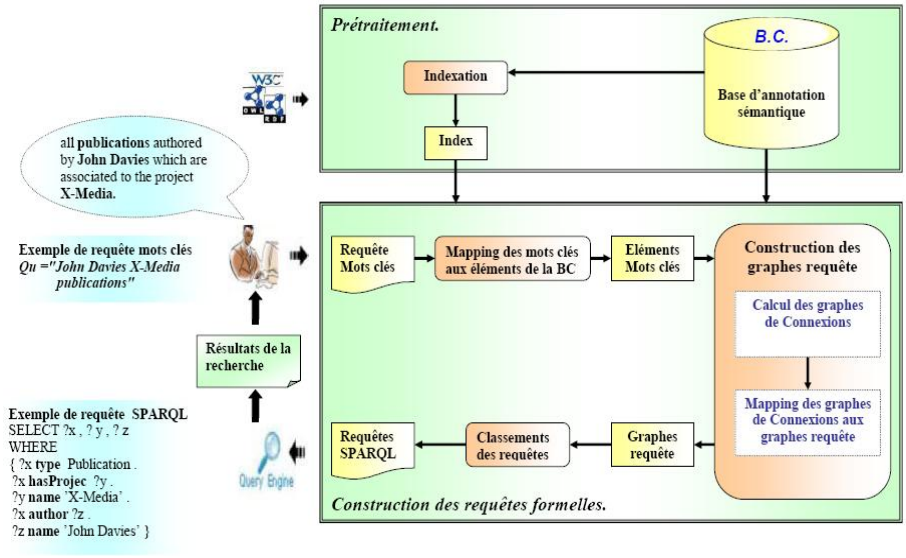

Fig. 2. Architecture générale de l'approche proposée

3.1 Mapping des mots clés aux éléments de la B.C.

Le but de ce mapping est de trouver les éléments de la B.C. (c'est-à-dire 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. sont utilisées pour le mapping. Dans notre approche, deux types de méthodes de mapping peuvent être utilisées: 1) *le mapping 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; 2) *Le mapping sémantique* utilise principalement des dictionnaires généraux comme WordNet [1], afin de trouver les mots sémantiquement approprié (par exemple, les synonymes, etc.). Nous assignons un score prédéfini *Sm(e) (Sm(e)* $\in$ *[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 BC 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 $f: Q_u \rightarrow D$ 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[4] est effectivement utilisée comme moteur d'index et de recherche. C'est à dire, les noms et les étiquettes des éléments de la B.C. 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 » $D_m = \{f(ki) / Q_u=(k1,..., K_n)\}$ seront alors introduit dans l'étape de la construction des graphes requêtes.

[4] Voir http://lucene.apache.org/java/docs/

Algorithme Exploration-BC (D_m, d)

1 **Entrée** D_m ensemble des éléments mots clés et **d** distance de la traversée.
2 **Sortie** graphe contenant tous ou certaines éléments de **D.**
3 Initialisation d'un nouveau graphe vide **g.**
4 **Pour e** $\in$ D_m
5 **Faire Si e** est un concept
6 **Alors Pour** Tous **i** étant instance de **e**
7 **Faire** Traversé (**i**, **d**, **0**, **g**)
8 **Sinon Si e** est une propriété d'objet (Individu-Individu)
9 **Alors Pour** Tous **i**, **j** avec <**i**, **e**, **j**> $\in$ **D**
10 **Faire** Traversé (**i**, **d**, **0**, **g**)
11 Traversé (**j**, **d**, **0**, **g**)
12 **Sinon Si e** est une propriété de données (Individu-Littérale)
13 **Alors Pour** Tous **i**, **j** avec <**i**, **e**, **j**> $\in$ **D**
14 **Faire** Traversé (**j**, **d**, **1**, **g**)
15 **Sinon Si e** est un Individu
16 **Alors** Traversé (**e**, **d**, **0**, **g**)
17 **Sinon Si e** es un Littérale
18 **Alors** Traversé (**e**, **d**, **1**, **g**)
19 Retourné **g**

Fig. 3. Algorithme d'exploration de la B.C.

3.2 Construction des Graphes Requête

Ce processus construit les 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 k_i de Q_u. Ensuite, un certain nombre d'algorithmes sont appliqués pour construire de possible graphes requêtes pour chaque ensemble de requêtes.

3.2.1 Calcul des graphes de Connexions

Dans notre approche, l'exploration des relatons se limite aux relations de type : (i, c), (i_1, R, i_2) et (i, U, j) où $i, i_1, i_2 \in I$, $j \in L$, $c \in C$, $R \in R$, $U \in U$ avec I (Individus), L (littéraux), C (Concepts), R (propriétés d'objets) et U (propriétés de données) Par l'utilisation de ces axiomes, nous explorons touts les éléments de la B.C. liées aux éléments de D_m identifiés dans la première étape selon l'algorithme montré à la Fig. 3. Le processus du calcule des graphes de connexions se déroule comme suit :

1. *Explorer les liens entre les éléments de la B.C* :.Fondamentalement, l'exploration comprend la traversée des voisins pour chacun des éléments de D_m. Ensuite, selon le type de l'élément particulier $e_m \in D_m$, différents parcours sont effectués pour construire un graphe connectant e_m 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 4, montre le pseudocode de l'algorithme pour la traversée récursif d'un individu ou d'un littéral particulier à ses voisins concepts, individus et littéraux. La valeur de d est réduite de un à chaque étape de récusions pour assurer à ce que cette traversée sera limitée à une certaine distance. En fin de compte, nous obtenons

un graphe *g* contenant touts les éléments de *D* qui ont une distance graphique qui ne dépasse pas *d* au moins à un des éléments de D_m. Ainsi, quelques éléments découverts peuvent vraiment ne pas être nécessaires pour connecter des éléments de D_m. Par conséquent, à partir de ce graphe, ont choisis seulement les chemins où le premier et le dernier sommet correspond est un élément de D_m. En particulier, une version modifiée de la procédure DFS (Depth First Search) sur les graphes est utilisé pour le calcul de tous les chemins $p \in P$ pour chaque paire possible tel que $(a, b) \in D_m$ tel que $p= (a, e_1, ..., e_n, b)$, et aucun des sommets n'est visité plus d'une fois. Ces chemins seront introduits dans l'étape suivante.

2. *Le calcul des Connexions* : Une requête peut être dérivée quand tous les éléments D_m 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 D_m. Le sous-graphe connectant les éléments de D_m sont calculés par une procédure récursive montrée dans l'algorithme de la fig.5. L'entrée de l'algorithme est l'ensemble des chemins *P* calculée 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 Traversé (i, d, t, g)

```
1  Entrée  i un individu ou un littérale à explorer, d distance de la traversée,  g un graphe
2           intermédiaire.
3  Sortie   graph g mis à jour contenant les éléments  connecté à i  sur une distance d.
4   Si   i   n'est pas marqué comme visité et d>0
5   Alors  Marquer i comme visité dans D
6          Si t=0
7          Alors  Ci :={c / i instances de c}
8                 Ajouter Arête (i, type, c) dans  g pour tout c ∈ Ci
9                 P :={(i, p, j) / <i, p, j> ∈ D}
10                Pour Tous  (i, p, j) ∈ P
11                Faire  Si  j n'est pas marqué comme visité dans D
12                       Alors  Ajouter nouveau Arête (i, p, j) dans  g
13                              Si  j est un individu
14                              Alors  Traversé (j, d-1, 0,  g)
15                              Sinon  Traversé (j, d-1, 1, g)
16         Si t=1
17         Alors  P :={(j, p, i) / <j, p, i> ∈ D}
18                Pour Tous  (j, p, i) ∈ P
19                Faire  Si  j n'est pas marqué comme visité dans D
20                       Alors  Ajouter nouveau Arête (j, p, i) dans  g
21                              Si  j est un individu
22                              Alors  Traversé (j, d-1, 0,  g)
```

Fig. 4. Algorithme d'exploration des individus et des littéraux

3.2.2 Mapping 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* (v_i, v_c)

(représentant la connexion *(i, c)*) est considéré comme arête dans le graphe requête de la forme *Type(x, C)*, où v_i, est un sommet construit au moyen d'un individu, v_c est construit en utilisant un concept, et *x* est un individu ou une variable. Lorsqu'un individu de v_i correspond à certains éléments $e_m \in D_m$, 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* (v_i, v_j), où v_i (v_j) 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 v_i (v_j) est construit en utilisant un littérale, il est tout simplement considéré comme nœud associé à une constant. Dans notre exemple de la Fig.1, il existe un seul graphe de connexion donc un seul graphe requête.

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.

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

Entrée l'ensemble de chemins **P** calculer par DFS pour tous les éléments mots clés de $\mathbf{D_m}$
Sortie Tous les sous-graphes connectant les sommets de $\mathbf{D_m}$
Si G= ϕ
Alors Pour {**i**, **j**} $\subseteq$ **R**
    Faire G=NouveauGraphe
        Pour chaque chemin **p** entre **i** et **j** (calculer par DFS)
        Faire ajouter (**i**, **p**, **j**) dans **G**
            Calculer-graphe-connexion (**P\p**, **C** $\cup$ {**i**, **j**}, **R**\{**i**, **j**}, **G**)
        g=**g** $\cup$ **G**
Sinon Pour i $\in$ **R**
    Faire Pour j $\in$ **C**
        Faire Pour chaque chemin **p** entre **i** et **j**
            Faire ajouter (**i**, **p**, **j**) dans **G**
                Calculer-graphe-connexion (**P\p**, **C** $\cup$ {**i**}, **R**\{**i**}, **G**)

Fig. 5. Algorithme pour calculer les graphes de connexions

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 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, Les

métriques standard souvent utilisés sont "PageRank" (pour calculer le score des sommets) et le plus court chemin (pour calculer le score des chemins). Les graphes sont construits à partir d'un ensemble de chemins *P*. Le score d'un tel graphe est défini comme agrégation monotonique des scores de ses chemins. En particulier, $C_G = \sum_{pi \in P} C_{pi}$ est utilisé, où C_{pi} et C_G dénotent des coûts. En général, le coût d'un chemin est calculé à partir du coût de ses éléments, c'est-à-dire $C_{pi} = \sum_{n \in pi} c(n)$. Sur la base des modèles proposés dans [62] pour attribuer des scores aux graphes requêtes, nous allons présenter la métrique que nous allons adoptées dans notre approche. Cette métrique représente à la fois le score de popularité et le score du mapping des éléments d'un graphe requête. $C_G = \sum_{pi \in P} \sum_{n \in pi} \frac{c(n)}{s_m(n)}$, où $s_m(n)$ est le score du mapping d'un élément *n* et reflète à la fois la similarité syntaxique et sémantique de l'élément *n*. *c(n)* est une fonction de coût spécifique des éléments. En particulier, on définit $c(v)=1-\frac{|v_{agg}|}{|V|}$ pour les sommets *v* et $c(e)=1-\frac{|e_{agg}|}{|E|}$ pour les arêtes, où $|V|$ est le nombre total de sommets dans le graphe d'exploration, $|v_{agg}|$ est le nombre des individus qui sont relies au sommet *v*, $|E|$ est le nombre total d'arêtes et $|e_{agg}|$ est le nombre d'arêtes correspondant à *e* dans le graphe d'exploration.

4 Implémentation et expérimentation

Notre système est implémenté en Java et Jena[5] API. Un utilisateur saisi sa requête mots-clés et il obtient en revanche une liste classée des requêtes SPARQL. Ces requêtes SPARQL traduites peuvent ainsi être directement envoyées à moteur de recherche ARQ[6] pour avoir les ressources relatives à sa requête.

Nous allons maintenant discuter les expériences que nous avons réalisées pour évaluer l'efficacité et le rendement de notre approche. Pour évaluer notre approche, nous utilisons une base d'annotation RDF au sujet des publications scientifiques sur l'informatique. Notre expérience a été réalisée sur un PC avec un CPU Pentium de 3,2 GHz et 2 Go de mémoire. Afin d'évaluer l'efficacité de notre approche, nous avons demandé à des collègues de nous fournir des requêtes mots-clés avec leurs descriptions en Langage Naturel (LN). 20 requêtes différentes ont été proposées. Un exemple de requête est : "*publications SmartWeb Pascal Hitzler 2002*" (*retrouver toutes les publications publié par Pascal Hitzler dans SmartWeb en 2002*).

Nous avons pris deux métriques pour l'évaluation: ***Rappel*** et ***MRR*** : Le rappel est défini comme le nombre des requêtes mots-clés correctement traduites divisées par toutes les requêtes mots-clés de l'ensemble d'évaluation (c.-à-d. 20 dans notre cas). Tandis que MRR, se concentre sur les performances globales du système. Pour

[5] http://jena.sourceforge.net/

[6] http://jena.sourceforge.net/ARQ/

évaluer l'efficacité de la génération des requêtes et leur classement, une métrique standard en RI appelé *Reciprocal Rank* (*RR*), définie comme *RR =1/r* est utilisé, où *r* est le rang de la bonne requête. Selon notre définition du problème, une requête est correcte si elle correspond au besoin en information (la description LN fournie). Si aucune des requêtes générées ne correspondent à la description LN, *RR* est égale à 0. Nous avons calculé *MRR* qui est la moyenne de *RR* de toutes les requêtes mots-clés de notre test. Nous avons obtenus un rappel de 0.825 et un *MRR* de 0.745 (Fig. 6)

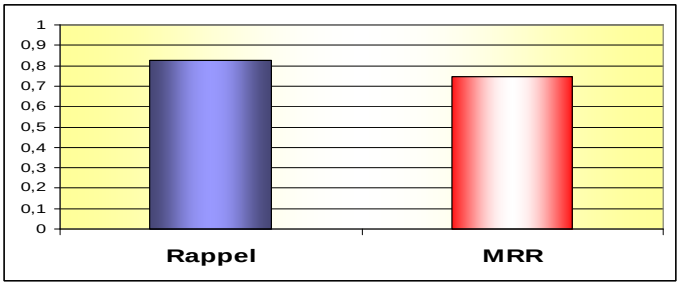

Fig. 6. Résultats des expériences

5 Conclusion et travaux futurs

Dans ce document, nous avons formalisé le problème de la construction des requêtes formelle et présenter une approche efficace pour le résoudre. Notre système vise à traduire les requêtes mots-clés en requêtes SPARQL en utilisant les connaissances disponibles dans une base de d'annotation pour réduire l'écart entre la recherche sémantique basé sur la logique formelle et les utilisateurs finaux habitué à utiliser les moteur de recherche classique basé sur les mots clé. Notre approche peut se résumé ainsi : Une fois un utilisateur entre une requête mots-clés, la première étape qui est réalisée est le mapping des mots clés aux éléments de la BC, cette étape utilise un ensemble de méthodes de mapping pour trouver les éléments correspondants dans la base de connaissances en fonction des mots-clés de l'utilisateur. Ensuite, l'étape de construction des graphes requête énumère toutes les combinaisons possibles de requête et applique un certain nombre d'algorithmes pour construire les graphes requêtes avec différents sens des éléments mots clés. Enfin, l'étape de classement des requête évalue les requêtes formelles construites à en utilisant un modèle probabiliste de classement des requêtes. Une liste de requêtes SPARQL classée sera remis à l'utilisateur final. Avec les premiers tests notre système a réalisé des résultats de traduction encourageant. La principale contribution de ce document est l'exploitation des annotations sémantiques dans la recherche d'information sur le Web.

Dans les travaux futurs, nous estimons améliorer notre approche de plusieurs façons: 1) élargir la portée des requêtes par l'introduction de certains opérateurs structurés (par exemple, NOT, OR, les filtres etc.) et en améliorant le support d'interaction humaine pour traduire les besoins en informations plus complexes. 2) Les travaux futurs seront axés sur le renforcement de la performance. Nous

envisageons d'exploiter la connaissance disponible dans l'ontologie pour une «exploration guidée » des liens entre les entités d'ontologie pour réduire le nombre de requêtes. 3) étendre notre approche sur une base d'annotation plus consistante et reparti sur le Web.

Bibliographie

1. Miller, G.A.: Wordnet: a lexical database for english. Commun. ACM 38(11) (1995) 39–41
2. BERNERS-LEE T., Weaving the Web, Harper Eds, San Francisco, 1998, 226 p.
3. Fensel, D. et al eds. 2003. *Spinning the Semantic Web:Bringing the World Wide Web to Its Full Potential*. Cambridge, Mass.: MIT Press.
4. UREN V., CIMIANO P., HANDSCHUH S., VARGAS-VERA M., MOTTA E. & CIRAVEGNA F., Semantic annotation for knowledge management: requirements and a survey of the state of the art, in Journal of Web Semantics, Science, Services and Agents on the World Wide Web, 4(1), Elsevier, 2006, pp.14-26.
5. Bray, T., J. Paoli, and C.M. Sperberg-McQueen (1998). *Extensible Markup Language (XML) 1.0*. W3C Recommendation, February
6. Lassila, O., and R. Swick, (1999). *Resource Description Framework (RDF) Model and Syntax Specification*, W3C Recommendation.
7. Harmelen, F., P.F. Patel-Schneider, and I. Horrocks (2001). *Reference Description of the DAML+OIL*. Ontology markup language.html.
8. Ding, L., P. Kolari, T. Finin, A. Joshi, Y. Peng, and Y. Yesha (2005). *On Homeland Securityand the Semantic Web: A Provenance and Trust Aware Inference Framework.* AAAI Spring Symposium on AI Technologies for Homeland Security, Stanford University, CA.
9. Amardeilh, F. (2007). Web Sémantique et Informatique Linguistique :propositions méthodologiques et réalisation d'une plateforme logicielle. Thèse de doctorat. Discipline : Informatique . Université Paris X – Nanterre, Mai 2007
10. T. Tran, P. Cimiano, S. Rudolph, and R. Studer, "Ontology-based interpretation of keywords for semantic search," in *ISWC/ASWC*, 2007, pp. 523–536.
11. K. S. Esmaili and H. Abolhassani. A categorization scheme for semantic web search engines. In 4th ACS/IEEE Int. Conf. on Computer Systems and Applications (AICCSA-06), 2006.
12. Peter Scheir, Viktoria Pammer, Stefanie N. Lindstaedt,Information Retrieval on the Semantic Web - Does it exist?, 2007
13. Y Lei, V S Uren and E Motta, Avatar: A search engine for the semantic web. EKAW 2006, pp. 238–245.
14. Lei, Y., Uren, V., Motta, E.: Semsearch: A search engine for the semantic web. Lecture Notes in Computer Science : Managing Knowledge in a World of Networks (2006) 238–245
15. Victoria Uren, Yuangui Lei, Enrico Motta , SemSearch: Refining Semantic Search ,2008
16. Karvounarakis, G., Alexaki, S., Christophides, V., Plexousakis, D., Scholl, M.: RQL: A Declarative Query Language for RDF. In: Proceedings of WWW'02, New York, NY, USA, ACM Press (2002) 592–603
17. Miller, L., Seaborne, A., Reggiori, A.: Three Implementations of SquishQL, a Simple RDF Query Language . In: Proceedings of ISWC'2002. (2002)
18. Royo, J., Mena, E., Bernard, J., Illarramendi, A.: Searching the web: From keywords to semantic queries. In: Proceedings of the Third International Conference on Information Technology and Applications (ICITA'05), IEEE Computer Society (2005) 244–249

19. Bernstein, A., Kaufmann, E., Gohring, A., Kiefer, C.: Querying ontologies: A controlled English interface for end-users. In: Proceedings of ISWC'2005. (2005)
20. Stojanovic, N., Stojanovic, L.: A Logic-based Approach for Query Refinement in Ontology-based Information Retrieval Systems. In: Proceedings of the 16th IEEE Int. Conf. on Tools with Artifical Intelligence. (2004)
21. Carlos A. Hurtado, A.P., Wood, P.T.: A Relaxed Approach to RDF Querying. In: Proceedings of ISWC'2006. (2006)
22. N Athanasis, V.C., Kotzinos, D.: Generating On the Fly Queries for the Semantic Web: The ICS-FORTH Graphical RQL Interface (GRQL). In: Proceedings of ISWC'2004. (2004)
23. Kandogan, E., Krishnamurthy, R., Raghavan, S., Vaithyanathan, S., Zhu, H.: Avatar Semantic Search: A Database Approach to Information Retrieval. In: Proceedings of SIGMOD'06, New York, NY, USA, ACM Press (2006) 790–792
24. Zhou, Q., Wang, C., Xiong, M., Wang, H., Yu, Y.: SPARK :Adapting keyword query to semantic search. In: ISWC/ASWC. (2007) 694-707
25. Thanh Tran , Haofen Wang , Sebastian Rudolph , Philipp Cimiano , Top-k Exploration of Query Graph Candidates for Efficient Keyword Search on RDF, 2008
26. Stojanovic, N., Gonzalez, J., Stojanovic, L.: Ontologer: A System for Usage-Driven Management of Ontology-Based Information Portals. In: Proceedings of L-CAP'2003. (2003)
27. Anyanwu, K., Maduko, A., Sheth, A.: SemRank: ranking complex relationship search results on the semantic web. In: Proceedings of WWW'2005, New York, NY, USA, ACM Press (2005) 117–127
28. Li, Y., Wang, Y., Huang, X.: A Relation-Based Search Engine in Semantic Web. Proceedings of IEEE Transactions on Knowledge and Data Engineering 19(2) (2007) 273–282.
29. Toumouh, A. Lehireche, D. Widdows, M. Malki. Adapting WordNet to the Medical Domain using Lexicosyntactic Patterns in the Ohsumed Corpus: 4th ACS/IEEE International Conference on Computer Systems and Applications (AICCSA-06), , Dubai/Sharjah, UAE. 2006.
30. D. Widdows, A. Toumouh, B. Dorow, A. Lehireche. Ongoing Developments in Automatically Adapting Lexical Resources to the Biomedical Domain, International Conference On Language Resources And Evaluation, , Italy, LREC 2006.