N°d'ordre:

REPUBLIQUE ALGERIENNE DEMOCRATIQUE & POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR & DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE DJILLALI LIABES - SIDI BEL ABBES
FACULTE DES SCIENCES EXACTES
DEPARTEMENT D'INFORMATIQUE

THESE DE DOCTORAT EN SCIENCE

Spécialité : **INFORMATIQUE**

Présentée et soutenue par
LATRECHE Abdelkrim

Interrogation à base d'annotations sémantiques : Construction basée sur le contexte de requêtes formelles à partir de requêtes mots-clés

Soutenue, le 12/04/2018
devant le jury composé de :

M. ELBERRICHI Zakaria	Professeur à l'UDL de Sidi bel abbès	Président
M. ADJOUJ Reda	M.C.A. à l'UDL de Sidi Bel Abbes	Examinateur
M. KESKES Nabil	M.C.A. à l'E.S.I. de Sidi bel abbès	Examinateur
M. HAMOU Reda Mohamed	M.C.A. à l'U.T.M. de Saida	Examinateur
M. BOUCHIHA Djelloul	Professeur au centre universitaire de Naama	Examinateur
M. LEHIRECHE Ahmed	Professeur à l'UDL de Sidi bel abbès	Directeur de thèse

Année universitaire : 2017-2018

A ma grande et petite famille

Remerciements

Cette thèse est le fruit des années d'efforts incessants, et travail qui n'aurait pas pu aboutir sans la participation précieuse de personnes qui partagent le même dévouement et passion pour la recherche scientifique. C'est avec un énorme plaisir que je remercie tous ceux qui m'ont soutenue durant ces années de travail pour faire aboutir cette thèse.

J'aimerais tout d'abord exprimer ma gratitude et mes sincères appréciations : A mon directeur de thèse le professeur **Lehireche ahmed** pour sa disponibilité, ses critiques et ses conseils toujours opportuns ainsi que pour sa grande gentillesse et de m'avoir donné toute cette confiance et d'avoir cru à mes compétences.

Aux membres de jury pour leur disponibilité et acceptation d'examiner et de rapporter mon travail.

A toutes les personnes qui m'ont aidée durant mon processus de recherche et d'écriture de cette thèse.

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. En principe, la recherche sémantique promet de fournir des résultats plus précis que la traditionnelle recherche par mots-clés, mais sa progression a été retardée en raison de la complexité de ses langages de requêtes.

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 construit automatiquement des requêtes formelles structurées à partir de requêtes mots-clés. Nous proposons un nouveau processus où nous allons introduire une nouvelle fonctionnalité d'auto-complétion de requêtes basée sur le contexte, afin d'aider l'utilisateur à formuler sa requête mots-clés, en suggérant des complétions de requêtes à partir de préfixe entré par l'utilisateur. Nous abordons aussi le problème de la génération basée sur le contexte des requêtes formelles structurées en utilisant l'historique des requêtes des utilisateurs, où les informations sur les requêtes précédentes peuvent être utilisées en tant qu'informations contextuelles pour influencer la génération d'une nouvelle requête.

Avec les premiers tests, notre approche réalise des résultats encourageants.

Keywords: Auto-Complétion, Recherche par mot-clé, Contexte de requête, RDF Graphes RDF, Annotation sémantique, Recherche sémantique.

Abstract

Traditional information search approaches do not explicitly capture the meaning of a keyword query, but provide a comfortable way for the user to express his or her information needs based on the keywords. In principle, semantic search aims to produce better results than traditional keyword search, but its progression has been retarded because of to the complexity of its query languages.

In this document, we present an approach to adapt keyword queries to querying the semantic web based on semantic annotations : the approach automatically construct structured formal queries from keywords queries. We propose a new process where we introduce a novel context-based query auto-completion feature to help the users to construct their keywords query by suggesting queries given prefixes. We also address the problem of context-based generating formal queries by exploiting user's query history, where previous queries can be used as contextual information for generating a new query.

With the first tests, our approach achieved encouraging results.

Keywords: Auto-Completion, Keyword Search, Query Context, RDF Graphs, Semantic Annotation, Semantic Search

ملخص

إن المانهج الحالية لاسترجاع المعلومات لا تستحوذ رسميا على المعنى الصريح لاستعلام الكلمات المفتاحية ، ولكنها توفر مسارا مريحا للمستخدم الذي يحدد متطلباته للمعلومات استنادا إلى الكلمات المفتاحية. من حيث المبدأ، البحث الدلالي يعد بتقديم نتائج أكثر دقة من البحث التقليدي عن طريق الكلمات المفتاحية ، ولكن تطورها قد تأخر بسبب تعقيد لغات الاستعلام.

في هذه الوثيقة ، نستكشف نهجا جديدا لتكييف استعلامات الكلمات المفتاحية لتكون قادرة على الاستعلام عن الويب الدلالي استنادا إلى الشرح الدلالي: يقوم النهج تلقائيا ببناء استعلامات رسمية منظمة من الكلمات المفتاحية. نقترح عملية جديدة حيث سنقدم ميزة جديدة للإكمال التلقائي لاستعلام استنادا إلى استعمال السياق، لمساعدة المستخدم في صياغة استعلام الكلمات المفتاحية. كما نتعامل مع مشكلة إنشاء استعلامات رسمية منظمة استنادا إلى السياق، وذلك باستخدام سجل طلبات بحث المستخدم كمعلومات السياقية للتأثير على انتاج استعلام جديد.

مع الاختبارات ألأولى, نهجنا يحقق نتائج مشجعة.

الكلمات المفتاحية : الإكمال التلقائي، سياق استعلام، الشرح الدلالي، البحث الدلالي، رسم بياني RDF

Table des matières

INTRODUCTION GENERALE **13**

1. CONTEXTE ET PROBLEMATIQUES 13
2. CONTRIBUTIONS 16
3. DEROULEMENT DE LA THESE 17

PARTIE I : ETAT DE L'ART

CHAPITRE 1. RECHERCHE D'INFORMATION : CONCEPTS DE BASE, RI SUR LE WEB ET AUTO-COMPLETION DE REQUETES **19**

1.1. INTRODUCTION 19
1.2. LA RECHERCHE D'INFORMATION 19
1.2.1. Définitions 19
1.2.2. Processus de base de la RI 20
1.2.3. Les modèles de RI 21
1.3. RECHERCHE D'INFORMATION SUR LE WEB 22
1.3.1. Outils de recherche d'information 22
1.3.2. Algorithme des moteurs de recherche d'information 23
1.4. L'AUTO-COMPLETION DE REQUETES 25
1.4.1. Introduction 25
1.4.2. Formulation du problème 26
1.4.3. Taxonomie des approches pour l'auto-complétion de requêtes 28
1.4.4. Evaluation des systèmes d'auto-complétion de requêtes 30
1.5. CONCLUSION 32

CHAPITRE 2. ETAT DE L'ART AUTOUR DE L'ANNOTATION SEMANTIQUE **33**

2.1. INTRODUCTION 33
2.2. LE WEB SEMANTIQUE 33
2.2.1. Généralités 33
2.2.2. L'architecture du web sémantique 34
2.2.3. Les Ressources Terminologiques ou Ontologiques 36
2.3. L'ANNOTATION ET LE WEB SEMANTIQUE 39
2.3.1. Définitions : *Annotations, métadonnées, annotation sémantique* 39
2.3.2. Les caractéristiques de l'annotation sémantique 41
2.3.3. L'annotation sémantique et les RTO 42
2.3.4. Les Langages de l'annotation sémantique 43
2.3.5. Les Langages d'interrogation des annotations sémantiques 46
2.3.6. Framework et outils de support pour les annotations sémantiques 50
2.4. CONCLUSION 51

CHAPITRE 3. EXPLOITATION DES ANNOTATIONS SEMANTIQUES POUR LA RECHERCHE D'INFORMATIONS DANS LE WEB SEMANTIQUE **52**

3.1. INTRODUCTION 52
3.2. LA RECHERCHE SEMANTIQUE 52
3.2.1. La recherche d'ontologies 52
3.2.2. La recherche sémantique basée sur le contexte 52

3.2.3.Découverte d'associations sémantiques 55
3.3. LES APPROCHES VISANT A FORMULER DES REQUETES FORMELLES 56
3.3.1.Langages formels 56
3.3.2.Constructeurs de requêtes par auto-complétion 56
3.3.3.Constructeurs de requêtes graphiques 57
3.3.4.Construction de requêtes formelles à partir de mots-clés ou en LN 57
3.3.5.Recherche par mots clés sur les bases de données relationnelles 60
3.4. MODELES DE DONNEES 61
3.5. STRUCTURE DES REPONSES 61
3.6. ALGORITHMES D'EXPLORATION 62
3.7. CLASSEMENT / CALCUL DU SCORE DES REPONSES 63
3.8. METHODOLOGIES D'EVALUATION 64
3.8.1.Jeux de données de test 64
3.8.2.Méthodologie d'évaluation de la recherche sémantique par mots clés 65
3.9. SYNTHESE 67
3.10. CONCLUSION 69

PARTIE II : CONTRIBUTIONS

CHAPITRE 4.CONSTRUCTION BASEE SUR LE CONTEXTE DE REQUETES FORMELLES A PARTIR DE REQUETES MOTS-CLES 71

4.1. INTRODUCTION 71
4.2. DEFINITION DU PROBLEME 73
4.2.1.Modèles de données 74
4.2.2.Modèle de requêtes 75
4.2.3.Aperçu de l'approche 75
4.3. PRETRAITEMENT 77
4.3.1.Construction de l'index de données 77
4.3.2.Construction de l'index structure 77
4.3.3.Calcul des scores des éléments 78
4.4. PROCESSUS D'AUTO-COMPLETION DE REQUETES 79
4.4.1.Définitions 79
4.4.2.Construction et utilisation 81
4.5. CONSTRUCTION DES REQUETES FORMELLES 84
4.5.1.Construction des k-meilleurs graphes requêtes 85
4.5.2.Mapping des graphes requêtes 88
4.6. MODELE DE PONDERATION 88
4.7. CONCLUSION 89

CHAPITRE 5. IMPLEMENTATION ET EVALUATION 90

5.1. INTRODUCTION 90
5.2. IMPLEMENTATION 90
5.3. EXPERIMENTATION ET EVALUATION 91
5.3.1.L'environnement de développement 92
5.3.2.Le jeu de données test 92
5.3.3.Métriques 92
5.3.4.Evaluation de l'effectivité 94
5.3.5.Evaluation de l'efficacité 95
5.3.6.L'évaluation de l'auto-complétion de requêtes 96
5.4. CONCLUSION 97

CONCLUSIONS GENERALE ET PERSPECTIVES 105
REFERENCES BIBLIOGRAPHIQUES 108

Liste des figures

FIGURE 1.1. RECHERCHE D'INFORMATION EN REPONSE A UNE REQUETE [BELEW, 2000]. 20
FIGURE 1.2. ILLUSTRATION DU PAGERANK. 24
FIGURE 1.3. EXEMPLE DES PAGES PIVOTS (HUBS) ET AUTORITES (AUTHORITIES). 25
FIGURE 1.4. ILLUSTRATION DE L'AUTO-COMPLETION DE REQUETE POUR LA REQUETE «INFORMATION RETRIEVAL». 26
FIGURE 1.5. L'INFRASTRUCTURE D'AUTO-COMPLETION DE REQUETES DE BASE 27
FIGURE 2.1. ARCHITECTURE PYRAMIDALE DU WEB SEMANTIQUE 34
FIGURE 2.2. EXTRAIT D'UNE TAXONOMIE SUR LA REPRESENTATION SIMPLIFIEE DES INSECTES. 36
FIGURE 2.3. LES DIFFERENTES RELATIONS QUI COMPOSENT UN THESAURUS. 37
FIGURE 2.4. DEFINITION FORMELLE D'UNE ONTOLOGIE DONNEE PAR [HANDSCHUH, 2005] 38
FIGURE 2.5. EXEMPLE D'UNE ONTOLOGIE DANS LE DOMAINE DE LA PRESSE «PEOPLE». 39
FIGURE 2.6. UTILISATION DE L'ANNOTATION SEMANTIQUE 41
FIGURE 2.7. EXEMPLE D'UNE ANNOTATION SEMANTIQUE ORCHESTREE PAR UNE ONTOLOGIE 42
FIGURE 2.8. EXEMPLE D'ANNOTATION SEMANTIQUE EN RDF (NOTATION GRAPHIQUE A GAUCHE ET XML A DROITE) 44
FIGURE 2.9. EXEMPLE D'ANNOTATION SEMANTIQUE BASEE SUR UN SCHEMA RDFS 45
FIGURE 2.10. EXEMPLE D'ANNOTATION SEMANTIQUE EN OWL LITE 46
FIGURE 2.11. SYNTAXE DE REQUETE RDQL 47
FIGURE 2.12. EXEMPLE DE DOCUMENT RDF. 49
FIGURE 2.13. EXEMPLE DE REQUETE SPARQL 49
FIGURE 3.1. L'ARCHITECTURE COMMUNE DE LA RECHERCHE SEMANTIQUE PAR MOT CLE. 60
FIGURE 3.2. METHODES D'EVALUATION DE L'EFFICACITE DU CLASSEMENT. 66
FIGURE 4.1. A) EXEMPLE DE GRAPHES DE DONNEES ET SCHEMA B) ESPACE D'INTERPRETATION C) GRAPHE REQUETE D) MAPPING DU GRAPHE REQUETE 73
FIGURE 4.2. A) EXEMPLES DE REQUETES B) ARCHITECTURE DE NOTRE SYSTEME 76
FIGURE 4.3. ILLUSTRATION DU PROCESSUS D'AUTO-COMPLETION DE REQUETES. 82
FIGURE 5.1. NDCG DES REQUETES POUR DIFFERENTES VALEURS DE QAS. 95
FIGURE 5.2. PRECISION DES REQUETES POUR DIFFERENTES VALEURS DE QAS. 95
FIGURE 5.3. EVALUATION DE L'EFFICACITE POUR DIFFERENTES VALEURS DE QAS. 96
FIGURE 5.4. LE MRR POUR LES PREFIXES DE 2-5 CARACTERES. 97

Liste des tableaux

TABLEAU 1.1. LES APPROCHES D'AUTO-COMPLETION DE REQUETES REPRESENTATIVES DANS LA LITTERATURE. ... 28
TABLEAU 1.2. EXEMPLE D'UNE SESSION DE RECHERCHE DANS L'ENSEMBLE DE DONNEES DU LOG AOL. 31
TABLEAU 5.1. REQUETES MOTS-CLES AVEC LEURS DESCRIPTIONS EN LANGAGE NATUREL (LN)..... 92

Introduction Générale

Introduction Générale

1. Contexte et Problématiques

La quantité d'information disponible sur internet est aujourd'hui gigantesque et sa croissance est exponentielle. Mais la spécificité de telles sources d'informations les rend difficilement exploitables et leur évolution constante rend complexe les techniques de recherche d'informations et devient difficile à appréhender pour un être humain. La raison principale est que, les documents sont hétérogènes, fragmentés, dispersés et sont souvent très peu structurés. Pour ce faire, des techniques, des méthodes et des outils permettant de partager, manipuler et rechercher au sein de tels documents ont étés proposés [Abrouk et al., 2006]. Dans le but d'améliorer la pertinence de la RI dans les systèmes de recherche d'information (SRI), plusieurs travaux ont été faits à divers niveaux. Alors, il y a eu proposition de plusieurs modèles de RI (*modèle booléen*, *modèle vectoriel*, *modèle probabiliste*, *modèle connexionniste*, *LSI, etc.)*. D'autres fonctionnalisées ont été introduites pour améliorer d'avantage la pertinence de la recherche d'information (RI), tel que la fonctionnalité d'auto-complétion de requêtes qui fait partie de la famille de reformulation de requête qui a pour tache d'aider l'utilisateur à formuler sans peine sa requête en saisissant uniquement un préfixe (par exemple, quelques caractères). De plus, plusieurs moteurs de recherche ont vu le jour (Google[1], Yahoo[2], etc.). Bien, que ces systèmes répondent à une bonne partie des besoins des utilisateurs, présentent quelques problèmes critiques : 1) la grande quantité des documents restitués ; 2) la variabilité des langages utilisés sur le web et la non structuration des documents; 3) La plupart des approches de RI considèrent le contenu d'un document comme un sac de mots sans syntaxe et sans sémantique [Khelif, 2006].

Cependant, grâce aux efforts de la communauté du Web sémantique (W3C), une deuxième génération est établie en ajoutant des annotations sémantiques au contenu Web. Le Web Sémantique est une extension du Web actuel dont la vision initiée en 1998 par Sir Tim Berners-Lee [Berners-Lee, 1998] a pour objectif de structurer les informations disponibles sur le Web, où 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 [Fensel et al., 2003]. 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 [Uren et al, 2006]. Les standards

[1] https://www.google.com/
[2] https://www.yahoo.com

du W3C, tels que XML (eXtensible Markup Language) [Bray et al., 1998], RDF (Ressource Description Framework) [Lassila et Swick, 1999], les schémas RDF (RDFS) et OWL [Ding et al., 2005] ont reçu un succès croissant en offrant un format uniforme pour la description et l'échange du contenu du Web [Latreche et al., 2009].

La recherche sémantique par l'exploitation des annotations sémantiques est devenue une voie très explorée et promet de fournir des résultats plus précis que la traditionnelle recherche par mots-clés. Cela permet de dépasser de nombreux problèmes liés aux comparaisons terme à terme entre documents ou entre documents et requêtes. Dans cette optique, plusieurs systèmes de recherche d'informations à base d'annotations et ontologies ont été proposés, les plus important et le plus populaires de ces systèmes sont ceux qui emploient de petites ontologies dans un domaine spécifique pour améliorer les performances du processus de recherche d'information. 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. L'objectif ultime de ces systèmes est l'amélioration des performances des moteurs de recherche traditionnels (particulièrement en ce qui concerne les mesures de précision et de rappel), en utilisant l'information de contexte (représentée par les ontologies de domaine et les annotations). La qualité de leurs résultats dépend fortement des propriétés qualitatives et quantitatives des ontologies utilisées. Le plus gros problème de ces moteurs de recherche est qu'ils sont limités à un certains contexte [Latreche et al., 2009].

Ces dernières années, la quantité de données structurées disponibles sur le Web a augmenté rapidement. Actuellement, il y'a des milliards de triplets RDF[3] stockées dans de multiples sources de données Web de différents domaines est reliés entre elles par des millions des liens RDF. Cette évolution ouvre de nouvelles voies pour répondre aux besoins en information plus complexes. De même, de nombreux langages de requêtes sémantiques (par exemple, RQL [Karvounarakis et al., 2002], RDQL[4], SquishQL [Miller et al., 2002] , SPARQL[5], etc.) ont été proposés pour interroger ces annotations sémantiques. Toutefois, pour pouvoir utiliser ces langages sémantiques, les utilisateurs doivent maîtriser les représentations complexes de la logique formelle et être familier avec les ontologies sous-adjacentes. Ceci devient un fossé critique entre la recherche sémantique et les utilisateurs finaux. Pendant ce temps, la plupart des utilisateurs se sont habitués pendant des années à faire de la recherche simplement par mots-clés. Par conséquent, il est important de permettre aux utilisateurs d'effectuer des recherches sémantiques simplement en entrant des requêtes à base de mots clés. Au cours des dernières années, il ya eu un intérêt croissant dans l'application des requêtes mots clés aux données structurées telles que le XML, RDF. Ainsi, il semble important de développer des approches qui peuvent interpréter les mots-clés pour pouvoir interroger une base d'annotation par une requête à base de mots clés [Latreche et al., 2009].

[3] http://www.w3.org/RDF/
[4] http://www.w3.org/Submission/2004/SUBM-RDQL-20040109/
[5] http://www.w3.org/TR/rdf-sparql-query/

Toutefois pour adapter la recherche par mots clés à la recherche sémantique, il faut surmonter les obstacles suivants [Latreche et al., 2009]:

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), de sorte que les termes de leurs requêtes peuvent être très différents de celles de la base de connaissances qu'ils utilisent.

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. La découverte automatique de ces relations manquantes devient un grand problème.

3) *Classement des Requête*s : 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.

Face à l'écart entre la requête sémantique fondée sur la logique formelle et les utilisateurs finaux, certaines communautés ont proposé différentes solutions pour le surmonter. Dans ce contexte, il existe plusieurs travaux sur la traduction des requêtes mots-clés en requêtes sémantiques. La recherche à base de graphe contribue également dans cette voie, par la construction des graphes requêtes par le biais de la navigation et de sélection sur l'ontologie [Tran et al., 2011 ; Wang et al., 2008 ; Zhou et al., 2007]. D'autres systèmes trouvent tous les chemins qui peuvent être dérivés à partir d'un graphe RDF, pour générer les graphes requêtes [Tran et al., 2009 ; Fu et al., 2011 ; Teng et Zhu, 2015 ; Le et al., 2014 ; Latreche et al., 2009]. Dans la plupart de ces systèmes, la génération de requêtes nécessite les étapes suivantes : (i) Mise en correspondance ou appariement des mots-clés de la requête aux éléments de la base de connaissances ; (ii) construction de graphes requêtes liant les éléments détectés précédemment ; (iii) classement des requêtes formelles / graphes requêtes construites, (iv) sélection de la bonne requête par l'utilisateur.

Chaque approche s'est focalisée sur une ou plusieurs étapes de ce processus.

Dans les approches existantes, notamment, les approches qui entre dans le contexte de notre travail c.à.d. les approches de recherche par mots-clés sur les données structurées en graphe RDF, supposent que l'utilisateur exprime son besoin en information de façon intuitive, sans avoir à connaître le langage de requêtes ou bien le formalisme de représentation de connaissances utilisé dans le système, c.-à.d., il n'a aucunes connaissances préalables sur le vocabulaire, la syntaxe et la structure (schéma) de la base interrogée, ceci va engendrer un certain nombre de problèmes critiques :

1) L'utilisateur trouve beaucoup de difficultés de formuler correctement sa requête mots-clés pour quelle puisse répondre à son besoin en informations, puisqu'il n'a aucunes connaissances préalables sur le vocabulaire, la syntaxe et la structure (schéma) de la base interrogée (annotations et ontologies sous-jacentes).

2) Vu que certaines requêtes sont plus ambigus que d'autres, la phase d'appariement des mots-clés aux éléments de la base de connaissances peut ne pas générer d'entités, comme elle peut générer pour chaque mot-clé un nombre important d'entités (utilisation des techniques de RI, telles l'appariement syntaxique imprécis, l'appariement sémantique, etc.), ce qui implique un nombre important de requêtes formelles construites à cause de toutes les combinaisons possibles des entités de la BC correspondant aux mots-clés.

3) Le calcul de toutes les combinaisons possibles des entités correspondantes aux mots-clés générés durant la première étape est un problème qui n'est pas abordé dans les travaux de recherche par mot-clé en général en raison de sa complexité en temps très élevé. La plupart des travaux ignorent simplement ce problème. D'autres, optent pour le prendre en compte dans la conception des algorithmes de recherche par mots clés, mais l'ignorent dans l'implémentation respective. Nous croyons que cette limitation est très restrictive.

4) La plupart de ces approches produisent des interprétations de requêtes (requêtes formelles) les plus populaires (en se basant sur le calcul des scores de popularité des entités de la BC), mais, ces interprétations de requêtes peuvent ne pas être celles désirés ou attendues par l'utilisateur.

Tous ces problèmes vont contribuer à rendre ces systèmes moins performants.

2. Contributions

Le travail présenté dans ce mémoire se situe dans ce contexte. L'approche que nous proposons [Latreche et al., 2017] veut fournir aux utilisateurs un moyen d'interrogation des bases d'annotations sémantiques à l'aide de requêtes mots-clés. L'approche proposée permet la construction automatique en se basant sur le contexte de requêtes formelles à partir de requêtes mots-clés entrées par les utilisateurs, et ce dans le but d'éviter à ces utilisateurs de se confronter à la complexité de la formulation d'une requête formelle dans un langage du Web Sémantique (tel que SPARQL). De cette façon, les utilisateurs peuvent garder l'habitude de saisir les mots-clés pour interroger le Web sémantique d'une manière transparente, ce qui augmentent l'utilisation sociale de la recherche sémantique.

Pour relever ce défi nous avons proposé les contributions suivantes:

1. Proposition d'un nouveau processus de construction de requêtes formelles à partir de requêtes mots-clés. Ce processus est constitue des étapes suivantes :
   a) Auto-complétion de requêtes,
   b) Construction des k-meilleurs graphes requêtes,
   c) Mapping des graphes requêtes (traduction des graphes requêtes en requêtes formelles conjonctives),
   d) Modèle de pondération (Mise à jour des scores des éléments de la requête sélectionnée).

2. Introduction et formalisation la fonctionnalité d'auto-complétion de requêtes qui a pour tache d'aider l'utilisateur à formuler sans peine sa requête mots-clés qui sera constituée uniquement de termes (éléments de la base de connaissances correspondant aux mots-clés), en lui présentant de possibles complétions de requêtes les plus pertinentes en se basant sur l'historique de recherche.
3. Proposition et implémentation d'un modèle de pondération des scores qui est utilisé pour capturer de façon concise les caractéristiques essentielles de l'historique des requêtes d'un utilisateur.
4. Exploitation des informations sur les requêtes précédentes (historique de recherche) pour les utilisées en tant qu'informations contextuelles pour influencer la génération d'une nouvelle requête.
5. Conception d'un algorithme efficace de construction des graphes requêtes qui étend les algorithmes existant.

3. Déroulement de la thèse

Le plan de cette thèse s'organise autour d'une progression de la réflexion, partant de la problématique telle qu'elle est traitée dans l'état de l'art (cf. partie 1) vers les solutions opérationnelles qui ont été conçues, en passant par la description de notre contribution, les expérimentations et les résultats. (cf. partie 2).

La première partie dénommée «état de l'art » comprend, les chapitres 1, 2 et 3.

Le chapitre 1 commence par présenter les concepts de base de la Recherche d'Information (RI). La deuxième partie sera consacrée à la RI sur le web en présentant les outils de recherche d'informations sur le web, les algorithmes des moteurs de recherche d'information. Enfin nous présentons un état de lieux sur l'auto-complétion de requêtes qui est une nouvelle fonctionnalité qui a été introduite dans les SRI et qui fait partie de la famille de reformulation de requête.

Le chapitre 2 dresse un état de l'art sur l'annotation sémantique, à savoir qu'est-ce qu'une annotation sémantique dans le cadre du web sémantique, quels sont les langages et ressources disponibles pour l'annotation sémantique et quels sont les outils existants.

Le chapitre 3 présente une revue de littérature sur l'exploitation des annotations pour la recherche d'information dans le web sémantiques tentant de couvrir la majorité des approches connexes à notre travail de recherche.

La deuxième partie dénommée «contributions » comprend les chapitres 4 et 5.

Le chapitre 4 présente en détails les contributions que nous avons proposés et nous présenterons en détaille les différentes étapes de notre approche.

Le chapitre 5 est consacré à l'expérimentation de notre approche ainsi que l'évaluation des résultats.

Nous concluons notre mémoire par une conclusion générale où nous présentons les perspectives de ce travail.

Partie I
Etat de l'art

Chapitre 1. Recherche d'Information : Concepts de base, RI sur le Web et Auto-complétion de requêtes

1.1. Introduction

La Recherche d'Information (RI) peut être définie comme une activité dont la finalité est de localiser et de délivrer un ensemble de documents à un utilisateur en fonction de son besoin en informations. Donc, Le plus grand défit à relever est que, parmi le volume important de documents disponibles de trouver ceux qui répondent au mieux au besoin en information de l'utilisateur. La mise en œuvre de la RI est réalisée par des outils informatiques appelés Systèmes de Recherche d'Information (SRI), qui ont pour but de retourner les documents pertinents à la requête de l'utilisateur, il s'agit, donc, de mettre en correspondance une représentation du besoin de l'utilisateur (requête) avec une représentation du contenu des documents au moyen d'une fonction de correspondance. Avec l'avènement du Web, la Recherche d'Information est confronté à de nouveaux défis d'accès à l'information, il s'agit cette fois de retrouver une information pertinente dans un espace d'informations diversifié et qui augmente rapidement. Ces difficultés ont donné naissance à de nouvelle discipline appelée Recherche d'Information sur le Web, qui est à son tour complété par d'autres fonctionnalités telles que l'auto-complétion de requêtes.

Dans ce chapitre, nous allons présentés, les trois éléments indissociables qui sont la Recherche d'Information, la Recherche d'Information sur le Web et l'auto-complétion de requêtes.

1.2. La Recherche d'Information

La recherche d'information RI (IR Information Retrieval en anglais) est un domaine historiquement lié aux sciences de l'information et à la bibliothéconomie. qui ont toujours eu le souci d'établir des représentations des documents dans le but d'en récupérer des informations à travers la construction d'index. L'informatique a permis le développement d'outils pour traiter l'information et établir la représentation des documents au moment de leur indexation, ainsi que pour rechercher l'information.

1.2.1. Définitions

Une des premières définitions de la RI la plus crédible à été donnée par Salton : « *La recherche d'information est un domaine, qui a pour objectif, la représentation, l'analyse, l'organisation, le stockage et l'accès à l'information* » [Salton et Mcgill, 1984]. Plusieurs autres définitions de la recherche d'information ont étés données ces dernières années, citons quelques unes :

- « *La recherche d'information est une activité dont la finalité est de localiser et de délivrer des granules documentaires à un utilisateur en fonction de son besoin en informations* » [Hernandez, 2005].
- « *La recherche d'information est une discipline de recherche qui intègre des modèles et des techniques dont le but est de faciliter l'accès à l'information pertinente pour un utilisateur ayant un besoin en information* » [Daoud, 2009].

Toutes ces définition partagent l'idée que la RI à pour objective d'extraire d'un document ou d'un ensemble de documents les informations pertinentes qui reflètent un besoin d'information [Bouramoul, 2011].

1.2.2. Processus de base de la RI

Dans son livre « FOA-Finding Out About » [Belew, 2000], R.K. Belew, résume le processus entier de Recherche d'Information (RI) en trois (3) sous-processus (Figure 1.1) : (i) « *injection d'une Requête* » (requête), (ii) « *Construction d'une Réponse* » (liste des documents pertinents), (iii) « *Evaluation de la Réponse* » (jugement des documents restitués).

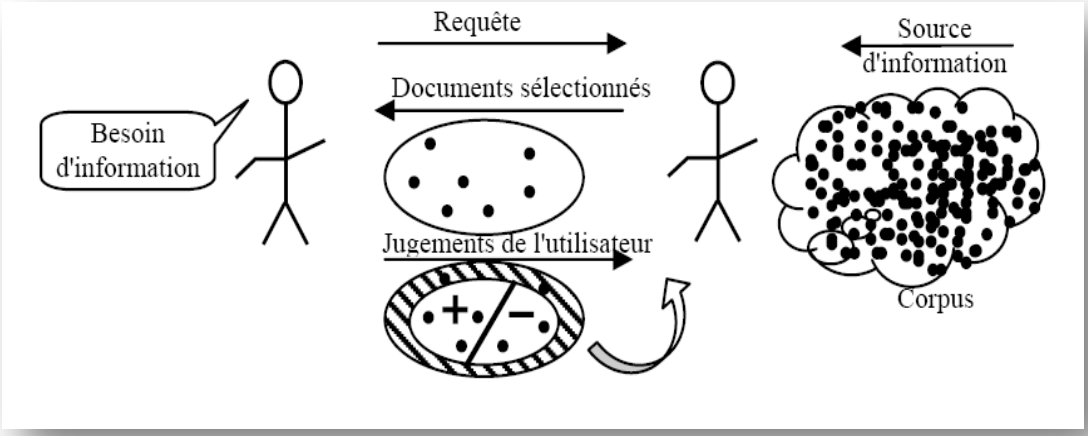


Figure 1.1. Recherche d'Information en réponse à une requête [Belew, 2000].

1.2.2.1. Notion de requête

Formuler clairement une requête correspondant à un besoin en information est connu pour être la plus difficile partie de sa « Réponse ». Un utilisateur peut ou non être capable de définir complètement les caractéristiques de la « Réponse » (besoin en information). Souvent il regarde s'il sait de quelle information il a besoin. Cet état cognitif mal défini est transformé par la suite en une expression externe dans le langage d'interrogation des systèmes de recherche d'information est nommé la « Requête ».

1.2.2.2. Le résultat de la recherche

La tâche de « Construire une Réponse » est de la responsabilité du « Répondeur » (Answerer) qui est dans le cas de l'internet, le « Système de Recherche d'Information » (SRI). Les problèmes liés aux machines font que cette tâche devient plus difficile pour un SRI que

pour l'être humain. L'uns de ces problèmes est le manque d'intelligence pour comprendre le besoin de l'utilisateur, pour palier les difficultés dues à l'ambiguïté du langage naturel et chercher des solutions ainsi que le manque de connaissances de fond pour répondre de façon compréhensible et avec les détails adéquats à l'utilisateur.

1.2.2.3. L'évaluation des résultats de la recherche

La phase de « Construction de la Réponse » induit que l'utilisateur doit évaluer "mentalement" par la suite les réponses qu'on lui a proposées pour décider jusqu'à quel point elles sont pertinentes. Le processus peut idéalement s'arrêter à ce niveau si l'utilisateur est complètement satisfait, ou alors, l'évaluation des réponses peut mener l'utilisateur lui-même à repenser et redéfinir le besoin en information. Ce processus se produit hors du système de RI.

1.2.3. Les modèles de RI

Un modèle de RI a pour rôle de fournir une formalisation du processus de recherche d'information et de fournir un cadre théorique pour la modélisation de la notion de pertinence. Ils s'appuient sur des cadres théoriques différents (Théorie des ensembles, algèbre, probabilité, etc.). De façon générale, on distingue trois principales catégories:

1) *les modèles booléens* : où l'on peut distinguer :
   a) *le modèle booléen pur* [Salton, 1970], est basé sur la théorie des ensembles. Dans ce modèle, chaque document est représenté par une conjonction logique des termes. Une requête est une expression booléenne dont les termes sont reliés par des opérateurs logiques (OR, AND, NOT). La fonction de correspondance est basée sur l'hypothèse de présence/absence des termes de la requête dans le document.
   b) *le modèle booléen étendu* [Salton et al., 1983] tient compte de l'importance des termes dans la représentation des documents et dans la requête, en affectant des poids à chaque terme du document et de la requête.
   c) *le modèle basé sur les ensembles flous* [Zadeh, 1965], qui une autre extension du modèle booléen et qui est basée sur la théorie des ensembles flous pour surpasser les inconvénients de la théorie classique.
2) *les modèles vectoriels* : où l'on peut distinguer :
   a) *le modèle vectoriel (vector model)* [Salton et Mcgill, 1984], est une représentation mathématique du contenu d'un document, selon une approche algébrique. les requêtes et les documents sont représentés dans l'espace vectoriel engendré par les termes d'indexation. Chaque document et requête est représenté par un vecteur. Le mécanisme de recherche consiste à retrouver les vecteurs documents qui s'approchent le plus du vecteur requête. Les principales mesures de similarité utilisées sont : Produit scalaire, Mesure de Jaccard, Mesure cosinus, etc.
   b) *Latent Semantic Indexing (LSI)* [Dumais, 1994] est une technique qui tend à implanter partiellement la recherche sémantique ou orientée concepts. Elle permet d'établir des relations entre un ensemble de documents et les termes qu'ils contiennent, en

construisant des « concepts » liés aux documents et aux termes. Dans ce modèle, les documents sont représentés dans un espace de dimension réduite issu de l'espace initial des termes d'indexation. Ceci permet de sélectionner des documents pertinents même s'ils ne contiennent aucun mot de la requête.

c) *Modèle connexionniste* se base sur le formalisme des réseaux de neurones [Mothe, 1994]. Un réseau est construit à partir des représentations initiales des documents et des informations descriptives associées (termes, auteurs, mots clés).

3) *modèles probabilistes,* utilisent un modèle mathématique fondé sur la théorie de la probabilité [Salton et Mcgill, 1984]. Le processus de recherche se traduit par le calcul du degré ou probabilité de pertinence d'un document relativement à une requête.

1.3. Recherche d'information sur le Web

1.3.1. Outils de recherche d'information

Il existe aujourd'hui une panoplie de systèmes ou outils de recherche d'information sur le Web, qui se spécialisent en fonction des services utilisés, du type d'information qu'ils traitent et du mode d'indexation utilisé. Ceci, induit des usages et des technologies très différentes. Nous pouvons, donc, distingués trois principales catégories d'outils de RI sur le web : les *moteurs de recherche*, *les annuaires* et *les méta-moteurs*. Un moteur de recherche, permet d'obtenir des résultats à partir du contenu des pages web (ex. Google). L'annuaire de recherche (ex. Yahoo.com) permettent également des recherches par mots-clés, mais ils sont surtout construits sur le modèle d'une arborescence thématique (thèmes, sous-thèmes, rubriques, sous-rubriques). Un méta-moteur, est un moteur intégrant plusieurs moteurs et permettant donc de lancer une recherche dans plusieurs directions en même temps (ex. Copernic). Généralement, les utilisateurs du Web, considèrent, toute interface d'interrogation comme moteur de recherche. En effet, l'annuaire réalise le premier repérage des ressources dans un domaine défini par l'organisation arborescente proposée, alors qu'un moteur de recherche permettra de trouver un document très précis. Enfin les méta-moteurs permettent d'interroger en une seule fois différents outils de recherche, qu'ils soient de type annuaire ou de type moteur [Bouramoul, 2011].

Dans ce qui suit, nous mettons l'accent en particulier sur les moteurs de recherche.

1.3.1.1. Moteur de recherche

Un moteur de recherche[6] est définit comme « *une application web permettant de trouver des ressources pages web, images, vidéo, fichiers, etc. à partir d'une requête sous forme de mots* ». Les moteurs de recherche sont constitués de « robots », qui parcourent les sites à intervalles réguliers et de façon automatique en suivant les liens hypertextes rencontrés sur chaque page atteinte pour découvrir de nouvelles adresses (URL). Chaque page identifiée est alors indexée dans une base de données, qui sera ensuite, accessible par les utilisateurs à partir de requêtes sous forme de mots-clés.

Le fonctionnement d'un moteur de recherche est résumé comme suit :

[6] https://fr.wikipedia.org/wiki/Moteur_de_recherche

1) *L'exploration du web* : durant cette phase, le web est exploré à intervalles réguliers et de façon automatique par un robot d'indexation suivant récursivement tous les hyperliens qu'il trouve et collecte les informations sur chaque page rencontrée.
2) *L'indexation* : qui consiste à extraire les informations considérées comme significatifs des documents explorés, puis les enregistrer dans une base de données organisée.
3) *La recherche :* correspond à la partie requêtes du moteur. Durant cette étape et à partir des mots clés de la requête de l'utilisateur, tout en appliquant des algorithmes appropriés il va rechercher les documents qui correspondent le mieux aux mots clés de la requête utilisateur, et les présenter par ordre de pertinence.

D'autres modules sont associés avec les trois briques de bases du moteur de recherche. Les plus importants sont :

- *Le correcteur orthographique* : il permet de corriger les erreurs introduites dans les mots de la requête de l'utilisateur.
- *L'anti dictionnaire* : utilisé pour supprimer les mots vides (tels que "le", "de", "la", etc.) dans l'index et dans les requêtes.
- *Le lemmatiseur* : il permet de lemmatiser les mots recherchés pour étendre leur portée de recherche.

Bien que, tous les moteurs de recherche passent par les mêmes étapes, néanmoins, ils ont tous leurs différences. Nous pouvons distingués, quelques unes :

- La façon d'explorer le web et quelles sont les informations considérés comme pertinente qui seront collectés des sites visités.
- La façon de construire et de rechercher l'index.
- La façon de présenter les résultats à l'utilisateur.

1.3.2. Algorithme des moteurs de recherche d'information

Les moteurs de recherche sur Internet doivent répondre à des millions de requêtes par jour alors que la quantité de documents qu'ils doivent analyser est gigantesque. Cette problématique est bien illustrée par Sergey Brin PageRank [Brin et Page, 1998] : « *Le Web est une vaste collection de documents hétérogènes complètement incontrôlés* ». Pour toutes ces raisons, différentes études ont suggéré de tenir compte de la popularité des documents afin d'améliorer les performances de la recherche d'information. Le PageRank de Google et le HITS [Kleinberg, 1999] de Kleinberg sont deux algorithmes fondamentaux qui utilisent les liens hypertextes pour classer les résultats d'une requête. Généralement, ces algorithmes fonctionnent en deux temps : Dans un premier temps, un moteur de recherche retourne une liste de documents répondant à la requête de l'utilisateur, en suite, dans un second temps, ils classent ces documents en tenant compte des liens hypertextes.

1.3.2.1. L'algorithme PageRank

Le moteur de recherche Google, utilise ce mode de classement des résultats. Les pages Web sont ordonnées selon leur popularité, une page qui est la cible d'un très grand nombre de liens est probablement non seulement une page validée, mais aussi une page détenant un contenu utile à un grand nombre d'utilisateurs. L'idée principale de cette méthode est de simuler le comportement d'un internaute naviguant de manière aléatoire sur Internet. La

probabilité qu'il visite une page donnée est d'autant plus grande que cette page est pointée par beaucoup d'autres pages au travers de leurs liens hypertextes. En considérant qu'une page confère une certaine autorité à une autre page en établissant un lien vers elle, la probabilité de passage de cet internaute aléatoire sur une page indique le degré de pertinence de ce document. La mesure de PageRank (PR) proposée par [Brin, 1998] mesure en effet la probabilité PR, pour un utilisateur navigant au hasard, d'atteindre une page donnée. Elle considère qu'un un lien émis par une page X vers une page Y peut être assimilé à un vote de X pour Y. Une page est considérée comme importante, lorsqu'elle reçoit plus de votes. Le PageRank est calculé comme suit : Soient, n pages ($P_1, P_2, \ldots, P_n$), citant la page X. Notons $PR(P_k)$ le PageRank de la page P_k, $S(P_k)$ le nombre de liens sortants de la page P_k, et $\alpha \in [0,1]$ un facteur, (généralement $\alpha = 0.85$), qui représente la probabilité de suivre effectivement les liens pour atteindre la page X, tandis que $(1-\alpha)$ représente la probabilité d'atteindre la page X sans suivre de liens. Le PageRank de la page X se calcule par : $PR(X) = (1-\alpha) + \alpha \frac{PR(T)}{S(T)}$.

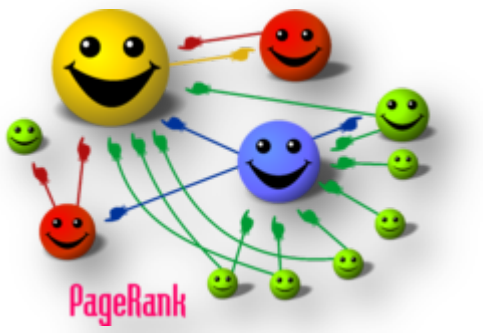


Figure 1.2. Illustration du PageRank.

1.3.2.2. L'algorithme HITS (Hyperlink-Induced Topic Search)

Kleinberg fut un des premiers à s'intéresser aux propriétés de connectivité du graphe représentatif d'Internet et de son apport dans la détection de la pertinence d'une page à une requête [Kleinberg, 1999]. Dans cette approche, deux types de page sont identifiés en fonction de la nature de leurs connexions avec les autres documents. On distingue ainsi les pages *autorités* ayant un grand nombre de liens entrants et les pages *hubs* ayant un grand nombre de liens sortants et regroupant les autorités d'un même sujet. Le but de l'algorithme HITS est de déterminer les *hubs* et les *autorités* qui renforcent leurs relations mutuellement sur un sujet donné. Ainsi Kleinberg dénombre les bons hubs comme des pages pointant vers beaucoup de bonnes autorités et les bonnes autorités comme des pages pointées par beaucoup de bons hubs.

Supposant W la matrice d'adjacence du graphe orienté G. Soit X un vecteur colonne pivot de dimension $(n * 1)$ et soit Y un vecteur colonne autorité de dimension $(n * 1)$. Les vecteurs X et Y contiennent les poids pivots et autorités correspondant à chaque nœud du graphe G. Les poids pivots et autorités sont calculés un processus itératif de la façon suivante: $X = W^T Y$ et $Y = WX$.

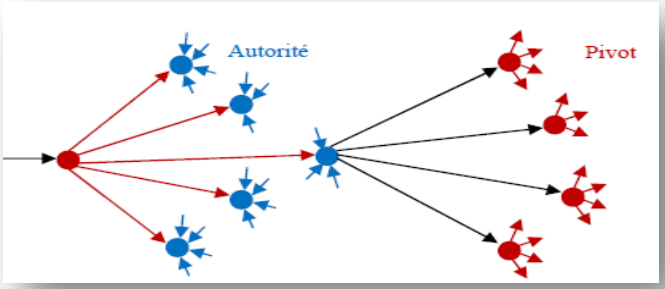


Figure 1.3. Exemple des pages pivots (Hubs) et autorités (Authorities).

1.4. L'auto-complétion de requêtes

1.4.1. Introduction

L'auto-complétion de requêtes (ACR) qui fait partie de la famille de reformulation de requête est une nouvelle fonctionnalité qui a été introduite dans les systèmes de recherche d'information (SRI). L'auto-complétion de requêtes a pour tache d'aider l'utilisateur à formuler sa requête en saisissant uniquement un préfixe (par exemple, quelques caractères), et donc de prédire la requête attendue par l'utilisateur en lui présentant de possibles complétions de requêtes classées sur la base de différents scores de pertinence. Cela permet d'éviter d'éventuelles fautes d'orthographe, en particulier sur les appareils à petits écrans et réduire la durée de recherche des utilisateurs ce qui implique une charge plus faible sur le moteur de recherche [Bar-Yossef et Kraus, 2011].

L'auto-complétion de requête est devenue une éminente caractéristique des principaux moteurs de recherche d'aujourd'hui, par exemple, Bing, Google, Yahoo, etc. La Figure 1.4 illustre un exemple d'auto-complétion de requête, telle implémenté dans un moteur de recherche commerciale pour la requête « *information retrieval* ».

La recherche indique que l'utilisation des complétions de requêtes peut grandement améliorer la satisfaction des utilisateurs, en particulier pour les requêtes informationnelles [Song et al., 2011].

L'entrée incomplète de l'utilisateur est souvent appelée *préfixe de requête* et les requêtes proposées sont souvent appelées *complétions de requêtes*.

Figure 1.4. Illustration de l'auto-complétion de requête pour La requête «information retrieval».

1.4.2. Formulation du problème

Le public général a été exposé à l'auto-complétion de requêtes vers 2004, lorsque Google a commencé à offrir un service nommé «Google Suggest», qui a fourni aux utilisateurs des complétions de requêtes dans la zone de recherche sous forme de menu déroulant pendant leur saisie en temps réel. Les complétions ont été basées sur l'exploration de ce que les autres ont recherchés. En 2011, [Bar-Yossef et Kraus, 2011] fait référence à cette fonctionnalité par « Query Auto Complétion (QAC) » et propose une approche simple qui est connue sous le nom de « *most popular completion* » qui génère des complétions de requête sur la base de la popularité des requêtes correspondant au préfixe entré par l'utilisateur.

De nombreux modèles d'ACR ont été proposés au cours de ces dernières années, la plus part de ces approches réduisent le problème d'auto-complétion de requêtes au problème de classement de requêtes. Ces modèles classent les requêtes suggérées pour chaque préfixe entré par l'utilisateur en se basant sur différents scores de pertinence. Étant donné un préfixe, les complétions de requêtes possibles sont classées selon un critère prédéfini, puis certains sont retournés à l'utilisateur. Généralement, un système d'auto-complétion stocke les associations entre les préfixes et les complétions de requête dans une structure de données efficace, comme par exemple un arbre de préfixes, qui permet des recherches plus efficaces par correspondance aux préfixes. Cet index est similaire à une table inversée stockant un mappage des requêtes aux documents dans un système de recherche d'informations. La Figure 1.5 montre l'infrastructure de base de l'auto-complétion de requêtes. Comme les requêtes et les interactions des utilisateurs peuvent être enregistrées par les moteurs de recherche, ce type de données est utilisé pour générer une table d'index en offline; Il capture les relations entre les préfixes et les requêtes. Lorsqu'un utilisateur saisit un préfixe dans le champ de recherche, en fonction de l'index préconstruit [Kastrinakis et Tzitzikas, 2010, Hsu et Ottaviano, 2013], une liste de complétions de requêtes est retrouvée. Sur la base d'un reclassement en utilisant des signaux déterminés au moment de la requête,

par exemple, l'heure, l'emplacement et le comportement de l'utilisateur, l'utilisateur recevra une liste finale des complétions de requêtes.

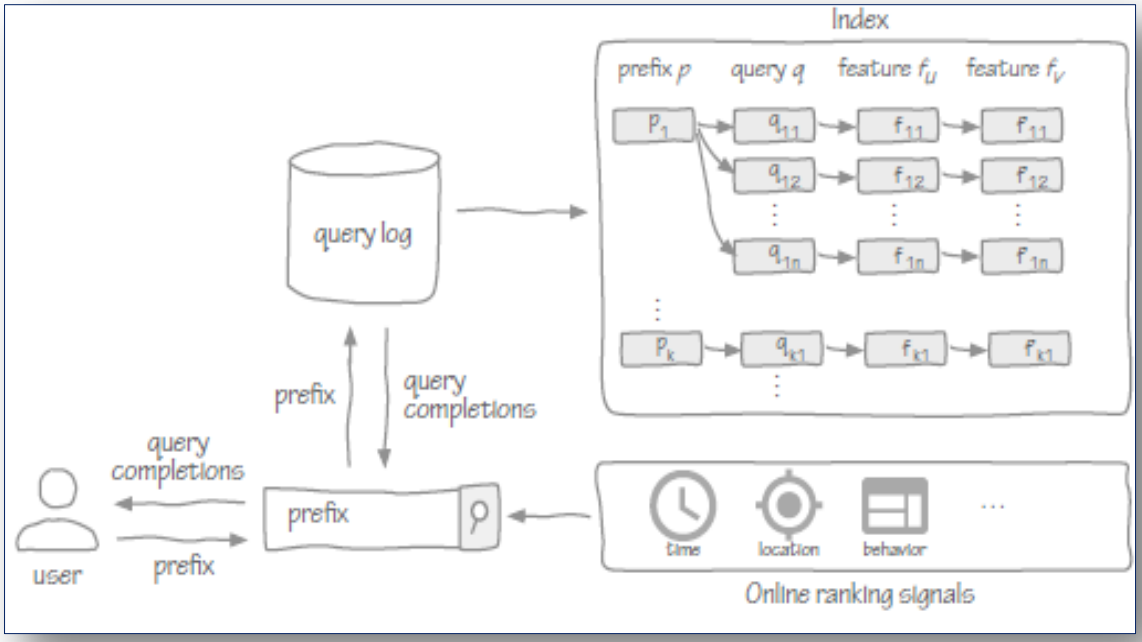


Figure 1.5. L'infrastructure d'auto-complétion de requêtes de base [Cai et de Rijke, 2016].

Dan ce qui suit, nous présentons le problème d'auto-complétion de requête de manière plus formelle.

Soit p un préfixe (c'est-à-dire une chaîne de caractères) entré par un utilisateur u. Soit $Q_I(p)$ désigne un ensemble de requêtes complètes qui étendent p; Il est utile de penser à $Q_I(p)$ comme un ensemble de complétions initialement retrouvés. Le problème de l'auto-complétion de requête consiste de trouver un classement $Q_S(p) \subseteq Q_I(p)$ de requêtes, avec $|Q_S(p)| = N$, où $N > 0$ est une valeur indiquant le nombre d'éléments classés au premier rang de $Q_I(p)$ à inclure dans $Q_S(p)$, de sorte que la fonction de coût $C(Q_S(p))$ pour générer le classement $Q_S(p)$ est optimisé:

$$C(Q_S(p)) = \sum_{q \in Q_S(p) \subset Q_I(p)} c(q) \qquad 1.1$$

Où $c(q)$ indique le coût de la requête q. Généralement, pour la tâche d'auto-complétion de requêtes, nous voulons classer la requête attendue par l'utilisateur en tête de la liste classée des complétions de requêtes. En conséquence, nous modélisons la fonction de coût à maximiser comme suit:

$$C(Q_S(p)) = \sum_{q \in Q_S(p) \subset Q_I(p)} \emptyset(q) \qquad 1.2$$

Où

$$\emptyset(q) = \begin{cases} \frac{1}{rank(q) in\ Q_S(p)}, & if\ q = q' \\ 0 & otherwise \end{cases} \qquad 1.3$$

Où q' est la requête qui est finalement soumise par l'utilisateur u. Les solutions algorithmiques au problème du ACR visent à prédire la requête attendue par l'utilisateur,

puis la retourner tôt dans la liste des candidats $Q_S(p)$. Nous désignons les éléments de la liste $Q_S(p)$ en tant que complétions de requêtes ou simplement complétions.

1.4.3. Taxonomie des approches pour l'auto-complétion de requêtes

Dans la littérature sur l'auto-complétion de requêtes, plusieurs approches ont été proposées pour cette tâche. Une comparaison brève des récentes approches de classement des complétions de requête est fournie par [Di Santo et al., 2015]. Dans [Cai1 et de Rijke, 2016], les auteurs classent les approches existantes en deux grandes catégories, à savoir les modèles heuristiques (heuristic models) et les modèles basés sur l'apprentissage (learning-based models). Les approches heuristiques cherchent à calculer un score directement en considérant différentes sources pour chaque complétion de requête possible indiquant à quel point il est probable que cette requête serait émise. En revanche, les approches basées sur l'apprentissage, qui sont basées sur des algorithmes d'apprentissage, visent à extraire des dizaines de fonctionnalités raisonnables pour capturer les caractéristiques de chaque complétion de requête. Ces deux catégories d'approches d'auto-complétion de requêtes peuvent être divisées en trois groupes: i) *time-sensitive*, ii) *contextual based* et iii) *demographic-based,* Voir le tableau 1.1.

Les approches d'ACR basées sur le contexte et la démographiques sont principalement axées sur l'historique de recherche d'un utilisateur et sur les informations sur le profil d'un utilisateur, telles que l'âge et le genre, respectivement. Comme les caractéristiques démographiques sont souvent difficiles à accéder, les travaux publiés sur le sujet sont limités.

Comme expliqué précédemment, les approches d'ACR traitent le problème de reclassement [Bar-Yossef et Kraus, 2011; Shokouhi, 2013; Cai et al., 2014; Jiang et al., 2014; Cai et de Rijke, 2016a].

Tableau 1.1. Les approches d'auto-complétion de requêtes représentatives dans la littérature.

		User-centered	
	Time-sensitive	**Contextual-based**	**Demographic-based**
Heuristic	[Shokouhi et Radinsky, 2012; Cai et al., 2014; Whiting et Jose, 2014]	[Bar-Yossef et Kraus, 2011; Mitra et al., 2014; Li et al., 2015]	
Learning-based	[Cai et de Rijke, 2016a]	[Jiang et al., 2014; Li et al., 2014 ; Mitra, 2015]	[Shokouhi, 2013]

La popularité d'une requête est un critère évident pour classer les complétions de requêtes, mais le temps peut affecter la popularité d'une requête. C'est pour cela que les aspects liés au temps ont été largement étudiés ces dernières années. Par exemple, la popularité des requêtes observée dans les dernières tendances [Whiting et Jose, 2014] ou dans les phénomènes périodiques [Cai et al., 2014] sont fréquemment considérés dans les approches d'ACR en fonction de la popularité des requêtes. La popularité de la requête future prévue

en fonction de l'analyse des séries chronologiques [Shokouhi et Radinsky, 2012] est un autre élément important des approches d'ACR liées au temps. En ce qui concerne les informations personnelles d'un utilisateur particulier, par exemple, le contexte de recherche consistant en des requêtes antérieures dans leur session courante [Bar-Yossef et Kraus, 2011], leur comportement de reformulation de requête, comme l'ajout de termes [Jiang et al., 2014], et leur contexte de profil tel que l'âge et le genre [Shokouhi, 2013], peuvent tous être utilisés pour en déduire leur intérêt spécifique et leur intention de recherche.

Les modèles d'ACR centrés sur l'utilisateur, c'est-à-dire les approches d'ACR personnalisées, où les informations provenant du contexte de recherche d'un utilisateur et les informations sur leurs interactions avec un moteur de recherche sont exploitées pour la tache d'auto-complétion de requête. Dans la plupart des travaux mentionnés jusqu'ici, les compléments de requêtes sont calculés globalement, c'est-à-dire, pour un préfixe donné, tous les utilisateurs reçoivent la même liste de complétions candidates. Mais l'exploitation du contexte personnel de l'utilisateur entraine une augmentation de l'efficacité des modèles d'ACR [Bar-Yossef et Kraus, 2011; Shokouhi, 2013]. Les modèles d'ACR centrés sur l'utilisateur sont principalement basés sur le contexte de recherche [Bar-Yossef et Kraus, 2011; Cai et al., 2014; Cai et de Rijke, 2016a] ou sur les interactions avec les utilisateurs [Li et al., 2015 ; Zhang et al., 2015] pour estimer la probabilité de soumettre les complétions de requête.

Malgré leur intuitivité, les approches heuristiques sont toujours surpassées par des méthodes probabilistes relativement simples telles que le modèle MPC [Bar-Yossef et Kraus, 2011], indépendamment de leur choix quant à la façon de calculer les scores de classement pour les complétions de requêtes. Les résultats expérimentaux de [Li et al., 2015; Zhang et al., 2015] montrent que l'incorporation de données détaillées d'interaction des utilisateur, par exemple, des feedback négatifs implicites sur les complétions de requêtes, peut considérablement et constamment augmenter la qualité de classement des complétions de requête.

Les approches d'ACR basées sur l'apprentissage explorent les fonctionnalités à partir des caractéristiques liées au temps et des caractéristiques spécifiques à l'utilisateur. Jusqu'à présent, les approches d'ACR centrées sur l'utilisateur ont reçu plus d'attention que les méthodes liées au temps. Nous croyons que cela est dû aux données riches enregistrées, selon lesquelles l'intention de recherche d'un utilisateur peut souvent être correctement prédite, ce qui entraîne de bonnes performances des ces modèles.

Un autre classement des approche d'ACR existantes a été proposé dans [Zhang et al., 2015]. Etant donné que la plupart des modèles d'ACR produisent les complétions de requêtes qui sont généralement basés sur l'historique de recherche (requêtes précédentes), bien que certains systèmes sont capables de construire de nouvelles requêtes en se basant sur différents scores de pertinence, tels que :

1) ***Popularity-based QAC*** : utilisant les données sur les fréquences des requêtes de l'historique de recherche (requêtes précédentes) [Bar-Yossef et Kraus, 2011]. La popularité de la requête est un critère évident pour classer les complétions de requêtes,

2) ***Personalized QAC*** : utilisant les informations du profil des utilisateurs [Bar-Yossef et Kraus, 2011; Cai et al.,2014; Shokouhi, 2013; Cai et de Rijke, 2016],
3) ***Context-based QAC*** : utilisant les informations sur les requêtes précédentes des utilisateurs [Bar-Yossef et Kraus, 2011; Cai et al., 2014; Mitra, 2015],
4) ***Time-based QAC*** : utilisant l'information temporelles [Shokouhi et Radinsky, 2012; Whiting et Jose, 2014; Cai et al., 2014; Miyanishi et Sakai, 2013], où certaines technologies d'analyse des séries chronologiques sont empruntées pour prédire la popularité future de la requête pour générer une liste classifiée des complétions de requêtes,
5) ***user behavior QAC*** : le comportement des utilisateurs [Li et al., 2014; Li et al., 2015],
6) ***tolerate errors QAC*** : la tolérance aux erreurs [Xiao et al., 2013; Mitra et Craswell, 2015]

1.4.4. Evaluation des systèmes d'auto-complétion de requêtes

L'évaluation des systèmes d'ACR constitue une étape importante dans l'élaboration d'un modèle d'ACR. En effet, elle permet de caractériser le modèle et de fournir des éléments de comparaison entre modèles. Les suggestions de mesures et les techniques d'évaluation d'auto-complétion de requêtes, se sont multipliées ces dernières d'années. D'une façon générale, un système d'auto-complétion de requêtes idéal a pour objectif de renvoyer une bonne liste de complétions de requêtes. Nous considérons une bonne liste de complétions de requêtes est celle qui contient la requête attendue par l'utilisateur en tète de liste, même lorsque l'entrée de l'utilisateur est courte, composée de très peu de caractères. Cet objectif est évalué par des mesures que nous définissons par la suite.

1.4.4.1. Collections de test

Beaucoup de recherche sur l'auto-complétion de requêtes prend le journal de requêtes (query logs) des moteurs de recherches comme point de départ. Chaque enregistrement dans un tel journal contient au moins trois composants principaux: une requête soumise, un ID utilisateur (ou une ID de session) et un horodatage. L'horodatage et l'ID utilisateur (ou l'ID de session) sont utilisés pour extraire les caractéristiques sensibles au temps et spécifiques à l'utilisateur, alors que la requête elle même peut être utilisée pour générer les préfixes de requête et pour générer sa fréquence dans les journaux de requêtes. Certaines informations supplémentaires, par exemple, les URL cliquées et leurs rangs, peuvent également être disponibles. Ces données peuvent être utilisées pour inférer les intentions de recherche des utilisateurs. Le tableau 1.2 montre un exemple d'une session de recherche dans le jeu de données du journal de requêtes AOL [Pass et al., 2006]. Dans cet exemple particulier, l'utilisateur (avec l'ID 2722) a soumis trois requêtes au total. Les deux premières requêtes, "*goodyear*" et "*hyundaiusa*", génèrent toutes les deux un URL cliqué, classé au rang 1 dans les listes d'URL renvoyées. La dernière requête, "*order hyundai parts*", comporte trois URL cliquées, classées en position 2, 9 et 6, respectivement.

Tableau 1.2. Exemple d'une session de recherche dans l'ensemble de données du log AOL.

User ID	Query	Timestamp	Rank	URL
2722	goodyear	2006-05-21 09:10:29	1	http://www.goodyear.com
2722	hyundaiusa	2006-05-21 09:28:54	1	http://www.hyundaiusa.com
2722	order hyundai parts	2006-05-21 09:51:02	2	http://www.hypartswholesale.com
2722	order hyundai parts	2006-05-21 09:51:02	9	http://www.the-best-source.com
2722	order hyundai parts	2006-05-21 09:51:02	6	http://www.racepages.com

Pour autant que nous le sachions, il existe trois journaux de requêtes (query logs) accessibles au public pour soutenir l'évaluation des systèmes d'ACR: le jeu de données AOL [Pass et al., 2006], le jeu de données MSN [Craswell et al., 2009] et le jeu de données SogouQ. Par exemple, l'ensemble de données AOL est couramment utilisé dans les approches d'ACR sensibles au temps [Cai et al., 2014 ; Whiting et Jose, 2014 ;Cai et de Rijke, 2016a], et dans les approches d'ACR personnalisé [Bar-Yossef et Kraus, 2011 ; Shokouhi, 2013]. Le jeu de données MSN est préféré dans [Cai et de Rijke, 2016a ; Cai et al., 2016b] pour personnaliser l'auto-complétion de requêtes. Le jeu de données SogouQ a été utilisé par [Whiting and Jose, 2014] et son utilisation est encouragée dans d'autres tâches. D'autres le jeu de données des journaux de requêtes des moteurs de recherche commerciaux modernes, qui ont enregistré plus de détails de recherche mais ne sont pas accessibles au public pour des raisons de confidentialité ou de compétitivité, ont également été appliqués à la recherche [Shokouhi et Radinsky, 2012 ; Jiang et al., 2014 ; Li et al., 2014, 2015 ; Zhang et al., 2015]. Bien que ces journaux facilitent la recherche sur l'auto-complétion de requête, ils sont limités dans le sens où ils ne contiennent pas d'informations sur les interactions détaillées de l'utilisateur avec le moteur de complétion de requêtes.

1.4.4.2. Mesures d'évaluation

Pour évaluer l'efficacité du classement des modèles d'ACR, le « *Mean Reciprocal Rank* » (MRR) est devenu une mesure standard. Pour un préfixe p d'une requête q de l'ensemble de requêtes Q, une liste de complétions de requêtes candidats correspondant $Q_I(p)$ et la requête finale q' soumise par l'utilisateur sont données. Ensuite, le « *Reciprocal Rank* » (RR) pour ce préfixe est calculé comme suit:

$$RR = \begin{cases} \frac{1}{rank\ of\ q' in\ Q_I(p)}, & if\ q' \in Q_I(p) \\ 0, & otherwise \end{cases} \qquad 1.4$$

Enfin, le score MRR est calculé comme la moyenne de tous les scores RR pour les préfixes de requêtes testés dans Q. À partir de cette formule, il est clair que MRR est une métrique axée sur la précision.

Le choix de MRR en tant que métrique de performance est commun dans les approches qui concernent la recherche d'une seule solution connue. Cependant, en raison des problèmes rapportés dans [Hofmann et al., 2014] sur la formulation de MRR (moyenne sur une échelle non proportionnelle), une métrique moins problématique, c'est-à-dire le taux de réussite au topK ($SR@K$), est étant également utilisé pour l'évaluation des modèles d'ACR, ce qui

indique le ratio moyen de la requête réelle qui peut être trouvée dans les topK de complétions de requête candidats sur les données de test.

MRR traite tous les préfixes du test de manière égale, c'est-à-dire qu'il est calculé en faisant la moyenne de tous les scores RR. Bar-Yossef et Kraus (2011) présentent un MRR pondéré pour évaluer la performance des approches d'ACR, ce qui est obtenu en pondérant les scores RR en fonction du nombre de complétions disponibles pour le préfixe.

En outre, si nous supposons que l'utilisateur soumettrait une requête en cliquant sur une complétion une fois qu'elle est classé dans la position supérieure de la liste des complétions de requêtes, nous pouvons utiliser la métrique « $Minimal\ Keystrokes\ (MKS)$ » pour indiquer le nombre minimal de touches que l'utilisateur doit taper avant qu'une requête soit soumise en cliquant sur la complétion de la requête.

1.5. Conclusion

Dans ce chapitre, nous avons présenté les principales notions et concepts de la recherche d'information. Nous avons aussi présentés les principales notions et outils de recherche sur le Web. Enfin un état de lieux sur l'auto-complétion de requêtes. A travers les différentes sections que nous avons présentées, dans ce chapitre, nous concluons que la recherche d'information, s'attache à définir des modèles et des systèmes pour faciliter l'accès à un ensemble de documents pertinents se trouvant dans des bases documentaires ou sur le web dont le contenu répond aux besoin en information des utilisateurs. Plusieurs moteurs de recherche ont vu le jour ces dernières années (Google, Altavista, etc.). Ces outils, bien qu'ils répondent à une bonne partie des besoins des utilisateurs, présentent quelques problèmes critiques : 1) la masse énorme des documents retournés. 2) la sensibilité au vocabulaire utilisé dans la requête. 3) la variabilité des langages utilisés sur le web et la non structuration des documents 4) La plupart des approches de RI considèrent le contenu d'un document comme un sac de mots sans syntaxe et sans sémantique. Cependant, grâce aux efforts de la communauté du Web sémantique (W3C), une deuxième génération est établie en ajoutant des annotations sémantiques au contenu Web, que les logiciels peuvent comprendre et dont les humains peuvent bénéficier.

Dans le chapitre suivant, nous dressons un état de l'art sur l'annotation sémantique.

Chapitre 2. Etat de l'art autour de l'annotation sémantique

2.1. Introduction

La plupart des informations du web actuel sont lisibles par les hommes mais elles sont difficilement interprétables par les machines. Ceci est visiblement constaté à travers des exemples de recherche d'information par les moteurs de recherche basé essentiellement sur les mots-clés ; tel que Google. L'imprécision de la recherche est due principalement à la nature de ces informations qui est encore limitée et non exploitable d'une manière plus « intelligente » par les machines. Le manque de normes et de sémantique formelle des informations du Web actuel empêche également le partage et la réutilisation d'information entre les individus et les organisations. Par conséquent, l'expression de Web Sémantique, énoncée par Tim Berners-Lee [Berners-Lee et al., 2001] fait d'abord référence à la vision du Web du futur comme « un vaste espace d'échange de ressources supportant la collaboration entre humains et machines en vue d'exploiter plus efficacement de grands volumes d'informations et de services variés disponibles sur le Web ».

L'objectif du Web sémantique est de rendre explicite le contenu sémantique des ressources dans le Web (documents, pages web, services, etc.). Pour cela, les ressources, textuelles ou multimédias, doivent être sémantiquement annotées par des métadonnées afin que les agents logiciels puissent les exploiter. Ces annotations sont alors exploitables par les utilisateurs finaux d'une application donnée pour rechercher, partager, accéder, publier des documents, des métadonnées ou même de la connaissance [Luong, 2007].

2.2. Le Web Sémantique

2.2.1. Généralités

Le Web sémantique est une idée de l'inventeur de World Wide Web « Tim Berners-Lee » en 1989, que le Web dans son ensemble peut être rendu plus intelligent et peut-être même intuitif sur la façon de servir les besoins d'un utilisateur. Berners-Lee observe que bien que les moteurs de recherche indexent une grande partie du contenu du Web, ils ont peu de capacité de sélectionner les pages qu'un utilisateur veut vraiment ou a besoin. Il prévoit un certain nombre de façons dont les développeurs et les auteurs, seuls ou en collaboration, peuvent utiliser des auto-descriptions et d'autres techniques pour que les programmes de compréhension du contexte puissent trouver sélectivement ce que veulent les utilisateurs.

Le Web sémantique est un Web qui comprend des documents ou des parties de documents décrivant des relations explicites entre les choses et contenant des informations sémantiques destinées au traitement automatisé par nos machines. Il fonctionne sur le principe des données partagées. Il y aura de nombreuses couches sur le Web sémantique (cf.

Figure 2.1).

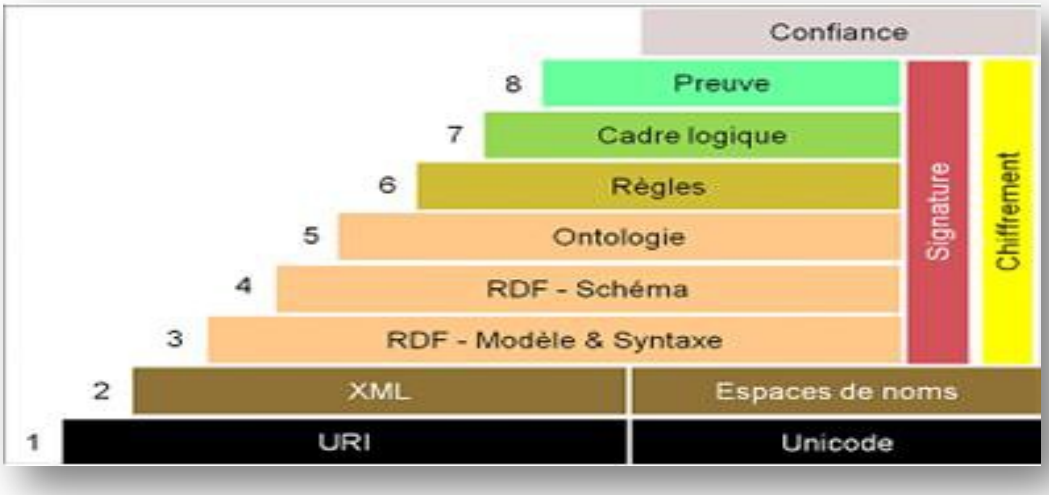


Figure 2.1. Architecture pyramidale du web sémantique [W3C][7]

2.2.2. L'architecture du web sémantique

2.2.2.1. L'URI

L'URI (Uniform Ressource Identifier, identifiant uniforme de ressource) suit les caractéristiques importantes de la WWW existante. C'est une chaîne d'un formulaire standardisé qui permet d'identifier uniquement les ressources (par exemple, les documents). Un sous-ensemble d'URI est Uniform Resource Locator (URL), qui contient le mécanisme d'accès et un emplacement (réseau) d'un document comme http://www.example.org/. Un autre sous-ensemble d'URI est URN qui permet d'identifier une ressource sans impliquer son emplacement et les moyens de déréférencement. L'utilisation de l'URI est importante pour un système Internet distribué, car il permet une identification compréhensible de toutes les ressources. Une variante internationale de l'URI est l'identificateur de ressource internationalisé (IRI) qui permet l'utilisation de caractères Unicode dans l'identifiant et pour lesquels un mappage à URI est défini [Davidson, 2010].

2.2.2.2. Unicode

Unicode est un standard d'encodage de jeux de caractères internationaux et permet à toutes les langues humaines d'être utilisées (écrites et lues) sur le web à l'aide d'un formulaire standardisé

2.2.2.3. XML

La couche XML (Extensible Markup Language) avec XML namespace et XML schema , assure qu'il existe une syntaxe commune utilisée dans le Web sémantique. XML est un langage de balisage général pour les documents contenant des informations structurées. Un document XML contient des éléments qui peuvent être imbriqués et qui peuvent contenir des attributs et des contenus. Les espaces de nommage XML permettent de spécifier

[7] https://semantiquee.wordpress.com/2014/12/07/architecture-web-semantique

différents vocabulaires de balisage dans un document XML. Le schéma XML sert à exprimer le schéma d'un ensemble particulier de documents XML [Daconta et al., 2003].

2.2.2.4. Les couches RDF Model & Syntax et RDF Schéma

RDF (Resource Description Framework), un format de représentation de données de base pour le Web sémantique, est un cadre pour représenter l'information sur les ressources sous forme de graphique. Il était principalement destiné à représenter des métadonnées sur les ressources WWW, telles que le titre, l'auteur et la date de modification d'une page Web, mais il peut être utilisé pour stocker d'autres données. Il est basé sur des triples sujet-prédicat- objet qui forment un graphique de données. Toutes les données du Web sémantique utilisent RDF comme langage de représentation primaire. La syntaxe normative pour la sérialisation de RDF est XML dans la forme RDF / XML. La sémantique formelle de RDF est également définie [Daconta et al., 2003].

RDF lui-même sert de description d'un graphe formé par triples. N'importe qui peut définir le vocabulaire des termes utilisés pour une description plus détaillée. Pour permettre la description normalisée des taxonomies et d'autres constructions ontologiques, un schéma RDF (RDFS) a été créé avec sa sémantique formelle au sein de RDF. RDFS peut être utilisé pour décrire des taxonomies de classes et de propriétés et les utiliser pour créer des ontologies légères.

2.2.2.5. La couche ontologie

Apporte une évolution car elle assure la description de sources d'information hétérogènes. Ces sources peuvent d'ailleurs formaliser une conceptualisation de choses existantes partagée par plusieurs personnes, voir par toute une communauté. Le rôle de l'ontologie est donc d'aider l'humain et la machine à communiquer, en priorisant sur l'échange de sémantique des informations plutôt que la syntaxe et en utilisant des règles précises [Daconta et al., 2003].

2.2.2.6. La couche Rules (règles)

Offre les moyens de l'intégration, de la dérivation, et de la transformation de données provenant de sources multiples.

2.2.2.7. La couche Logique

Se trouve au-dessus de la couche Ontologie. Certains les considèrent comme étant au même niveau.

2.2.2.8. Les couches Proof (Preuve) et Trust (Confiance)

Permettent de vérifier des déclarations effectuées dans le web sémantique. Si l'on part du web sémantique, qui est le but en soit, on peut voir qu'il dépend des différents agents (logiciels ou machines), qui eux même dépendent d'un service de requête (Query Service dépendant du langage XML), mais aussi de la couche confiance (trust), elle-même dépendant des couches Preuve (Proof) Sécurité (Security) et des règles associées au langage de requête (Rules). Les agents dépendent également de cette couche Règles, qui est basée sur la

définition des ontologies, elle-même existant par le biais des seules métadonnées. On retrouve donc ici clairement la démarche stratégique associée au web sémantique.

2.2.3. Les Ressources Terminologiques ou Ontologiques

Bourigault et al. ont défini la notion de Ressources Terminologiques ou Ontologiques (RTO) à la croisée des domaines de la Terminologie et l'ingénierie des connaissances [Bourigault et al., 2004]. Nous allons nous intéresser aux trois principales RTO permettant de représenter et de modéliser la connaissance d'un domaine : *les taxonomies, les thesaurus, et les ontologies*.

2.2.3.1. Les Taxonomies

L'utilisation des taxonomies est l'un des moyens pour conceptualiser les objets et les classifier hiérarchiquement. Une taxonomie est définie comme : « *La classification des entités d'information sous la forme d'une hiérarchie, selon les relations présumées des entités réelles qu'elles représentent* » [Charlet, 2002]. Une taxonomie est une hiérarchie sémantique dans laquelle les entités d'information sont liées par la sous-classification de la relation. Une taxonomie est généralement représentée avec la racine de la taxonomie en haut et chaque nœud de la taxonomie représente une entité du monde réel. Chaque lien entre nœuds représente une relation spéciale appelée la sous-classification de la relation (si la flèche du lien est pointant vers le nœud parent) ou est une super-classification de (si la flèche pointe vers le nœud enfant). Un exemple de taxonomie est montré dans la Figure 2.2.

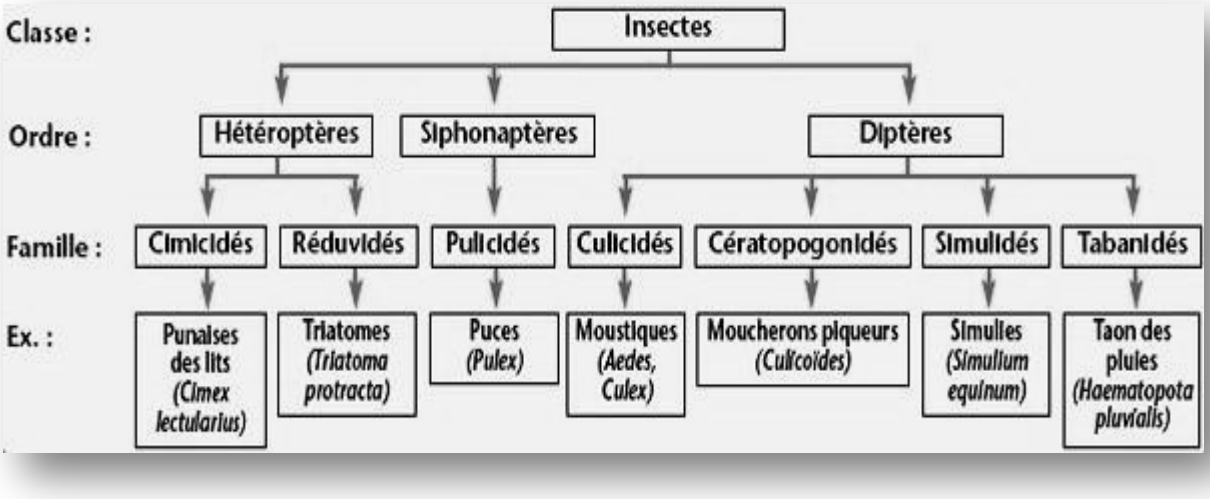


Figure 2.2. Extrait d'une taxonomie sur la représentation simplifiée des insectes[8]

2.2.3.2. Les Thesaurus

Charlet [Charlet, 2002] a défini un thesaurus comme « *un ensemble de termes normalisés fondé sur une structuration hiérarchisée. Les termes y sont organisés de manière conceptuelle et reliés entre eux par des relations sémantiques. Organisé alphabétiquement, il forme un répertoire alphabétique de termes normalisés pour l'analyse de contenu, le classement et donc l'indexation de documents d'information*». Bourigault & al. [Bourigault et al., 2004]

[8] http://www.jim.fr/e-docs/00/01/B3/F8

définissent un thesaurus comme « *un langage documentaire fondé sur une structuration hiérarchisée* », sachant qu'un langage documentaire est un « *ensemble organisé de termes normalisés, utilisé pour représenter le contenu des documents à des fins de mémorisation pour une recherche ultérieure* ». Un thesaurus est donc considéré comme un vocabulaire contrôlé et structuré dans lequel les relations entre les termes du domaine considéré sont clairement spécifiées formant ainsi un réseau terminologique. La structuration hiérarchisée correspond à la relation d'hyperonymie déjà vue pour les taxonomies sauf qu'elle ne structure plus des concepts mais les termes du vocabulaire. Dans la figure 2.3, le terme « Véhicule » a un sens plus général que « voiture »

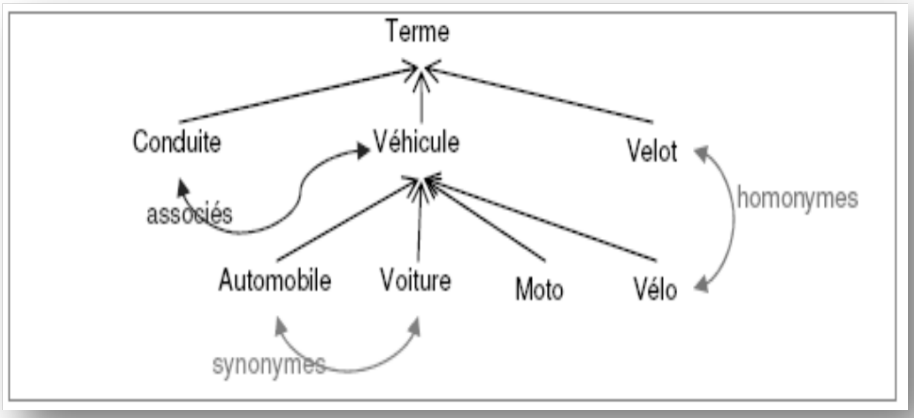


Figure 2.3. Les différentes relations qui composent un thesaurus [Amardeilh, 2007].

- Entre « Voiture » et « Automobile », il existe une relation de Synonymie.
- Entre « Velot » et « Vélo », il existe une relation d'homonymie.

Comme mentionné dans sa définition, l'objectif premier d'un thesaurus est de faciliter la recherche de document et de rendre l'indexation de documents consistante à l'aide des termes, aussi appelés descripteurs dans ce contexte [Amardeilh, 2007].

2.2.3.3. Les Ontologies

2.2.3.3.1. Définitions

Le dictionnaire nous a donné les définitions suivantes : « *Une branche de la métaphysique concernée par la nature et les relations de l'être* » ou « *Une théorie particulière sur la nature de l'être ou sur les genres d'existants* ». Ces définitions indique que le terme provient de la philosophie, partie de la métaphysique qui est l'étude systématique des principes sous-sujet, le plus souvent la nature de l'être et la nature de l'expérience [Charlet, 2002]. Souvent la distinction est faite entre «grand O» Ontologie et "Petite o" ontologie. Grand « O » L'Ontologie est la discipline philosophique. Petit « o », est la discipline de l'ingénierie des technologies de l'information qui émergée au cours des dernières années.

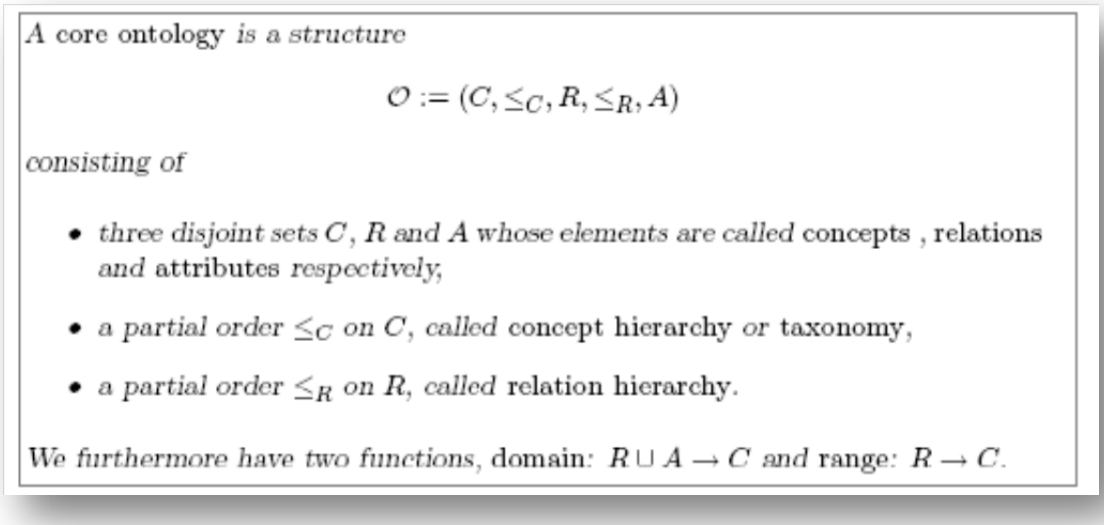
Une des premières définitions de l'ontologie communément admise en Intelligence Artificielle a été énoncée par Gruber [Gruber, 1993] comme la « *spécification explicite d'une conceptualisation* ». Cette définition de l'ontologie a ensuite été affinée *par* R. Studer et al.

[Studer et al., 1998] comme « *spécification formelle et explicite d'une conceptualisation partagée* » :

- *Formelle* : l'ontologie doit être lisible par une machine, ce qui exclut le langage naturel.
- *Explicite :* la définition explicite des concepts utilisés et des contraintes de leur utilisation.
- *Conceptualisation :* le modèle abstrait d'un phénomène du monde réel par identification des *concepts* clefs de ce phénomène.
- *Partagée :* l'ontologie n'est pas la propriété d'un individu, mais elle représente un consensus *accepté* par une communauté d'utilisateurs.

Une ontologie nous fournit les moyens d'exprimer les concepts d'un domaine en les organisant hiérarchiquement et en définissant leurs propriétés sémantiques dans un langage de représentation des connaissances formel favorisant le partage d'une vue consensuelle sur ce domaine entre les applications informatiques qui en font usage [Bourigault et al., 2004].

Une définition formelle d'une ontologie est présentée par Handschuh [Handschuh, 2005] (Figure 2.4) :



A core ontology *is a structure*

$$\mathcal{O} := (C, \leq_C, R, \leq_R, A)$$

consisting of

- *three disjoint sets* C, R *and* A *whose elements are called* concepts , relations *and* attributes *respectively,*
- *a partial order* $\leq_C$ *on* C, *called* concept hierarchy *or* taxonomy,
- *a partial order* $\leq_R$ *on* R, *called* relation hierarchy.

We furthermore have two functions, domain: $R \cup A \rightarrow C$ *and* range: $R \rightarrow C$.

Figure 2.4. Définition formelle d'une ontologie donnée par [Handschuh, 2005]

Les concepts (ou « classes ») représentent les objets, abstraits ou concrets, réels ou fictifs, élémentaires ou composites, du monde réel. Ces concepts sont organisés en taxonomie, par l'utilisation de la relation de subsumption. Dans la Figure 2.5, le concept « Film » est une sous-classe de « œuvre artistique ».

Les relations représentent des interactions entre concepts permettant de construire des représentations complexes de la connaissance du domaine [Charlet et al., 2004]. Dans la figure 2.5, une relation sémantique « réalise » existe entre les concepts « Personnalité » et « Film ».

Les attributs sont des caractéristiques particulières permettent de le définir un concept de manière unique dans le domaine [Charlet et al., 2004]. Leurs valeurs sont littérales, i.e. comme une chaîne de caractère ou un nombre entier. Par exemple, dans la figure 2.5, le concept « Film », peut avoir les attributs suivants : un « titre », une « date de production », etc.

Les instances de concepts ou les individus font partie de la base de connaissance [Handschuh, 2005]. Ils permettent de stocker les instances des concepts et de relations et les valeurs des propriétés selon le domaine de l'ontologie. Dans la figure 2.5, « le parrain » est une instance du concept (Film).

Les ontologies représentent une composante importante du web sémantique puisque son objectif est la modélisation des ressources web pour les rendre accessibles par tous. Les ontologies donnent des représentations conceptuelles du monde réel, qui un point important dans la démarche proposée par Berner Lee. L'ontologie en tant que résultat d'une conceptualisation d'un domaine, offre une solution idéale pour relier les réseaux sémantiques associés à des ressources web et donc faciliter l'accès à des informations et des applications.

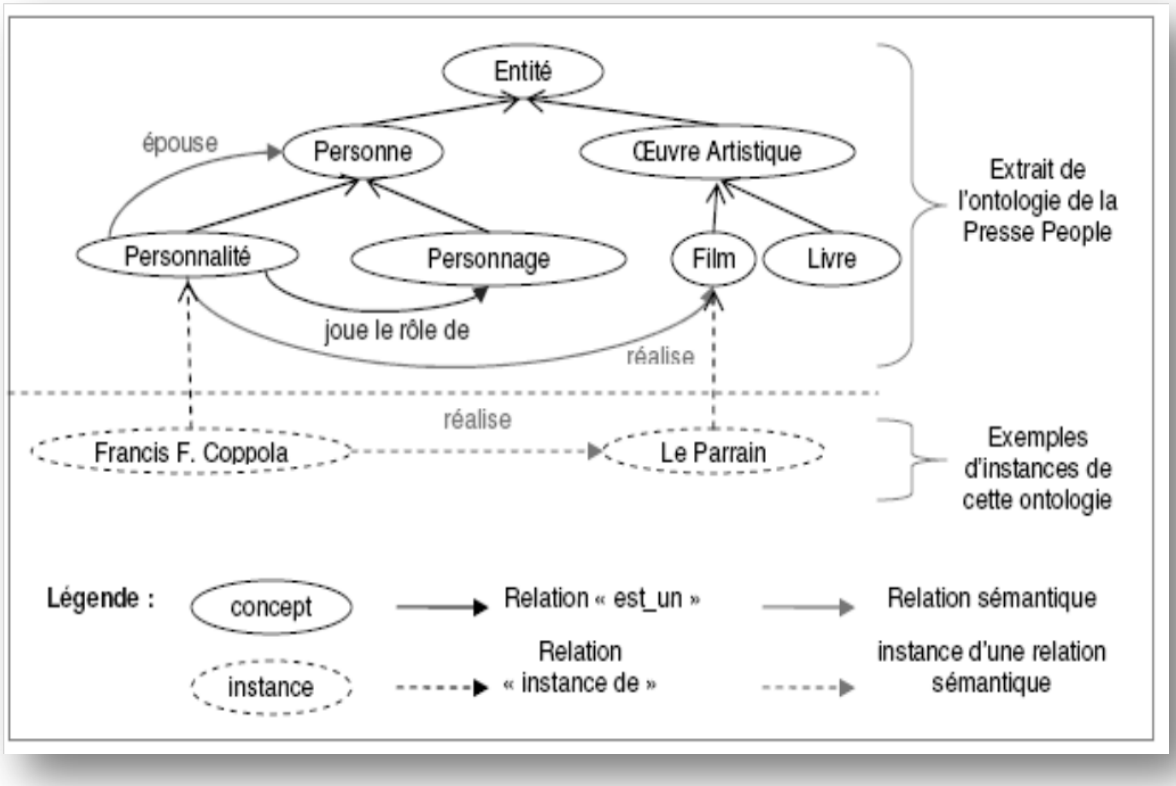


Figure 2.5. Exemple d'une ontologie dans le domaine de la presse «People» [Amardeilh, 2007].

2.3. L'annotation et le Web Sémantique

L'annotation pour le Web Sémantique consiste donc à prendre en entrée une ressource documentaire et fournir en sortie le même contenu enrichi par des annotations sémantiques basées sur des représentations de la connaissance plus ou moins formelles. [Amardeilh, 2007]

2.3.1. Définitions : *Annotations, métadonnées, annotation sémantique*

Le dictionnaire le **Petit Robert** définit le terme *annotation* comme une « *note critique ou explicative qui accompagne un texte* ». Le verbe *annoter* est quant à lui défini comme « *accompagner un texte de notes, de remarques, de commentaires* ».

En informatique, selon le W3C[9], « *une annotation est un commentaire, une note, une explication ou toute autre remarque externe qui peut être attaché à un document web ou à une partie de celui-ci* ».

Ainsi, Une *annotation* peut être considérée d'une manière simple comme « *une information graphique ou textuelle attachée à un document et le plus souvent placée dans ce document* » [Desmontils et Jacquin, 2002]. Normalement, une annotation est toujours associée à l'objet qui a été annoté.

Les *métadonnées* sont des informations structurées qui décrivent, expliquent, localisent ou facilitent la récupération, l'utilisation ou la gestion d'une ressource d'information [Daconta et al., 2003]. Les métadonnées sont souvent appelées données sur données ou des informations sur l'information. Le terme de métadonnées est utilisé différemment dans différentes communautés. Certains l'utilisent pour se référer à des informations compréhensibles par machine, tandis que d'autres l'utilisent uniquement pour les enregistrements qui décrivent des ressources électroniques [Daconta et al., 2003]. La métadonnée est une information interprétable par machine sur des ressources d'information du Web.

Prié et Garlatti [Prié et Garlatti, 2004] considèrent qu'une annotation descriptive, lorsqu'elle s'intéresse à la structure logique du contenu d'un document, est le plus souvent appelée *annotation sémantique*. Selon la définition de Kiryakov [Kiryakov et al., 2004], « *l'annotation sémantique consiste en la génération de méta-données ayant pour objectif de rendre possible de nouvelles méthodes d'accéder aux informations, telles que l'indexation et la recherche d'informations, la catégorisation, ou la génération de méta-données avancées* ». Amardeilh [Amardeilh, 2007] définit l'annotation sémantique 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* ». L'objectif de l'annotation sémantique est d'exprimer la "*sémantique*" du contenu d'une ressource. En termes de documentation, les annotations sémantiques décrivent le lien entre les entités se trouvant dans le document et leurs descriptions sémantiques représentées dans l'ontologie. La Figure 2.6 montre un exemple de l'utilisation de l'annotation sémantique sur le Web sémantique.

[9] http://www.w3.org (juillet 2009)

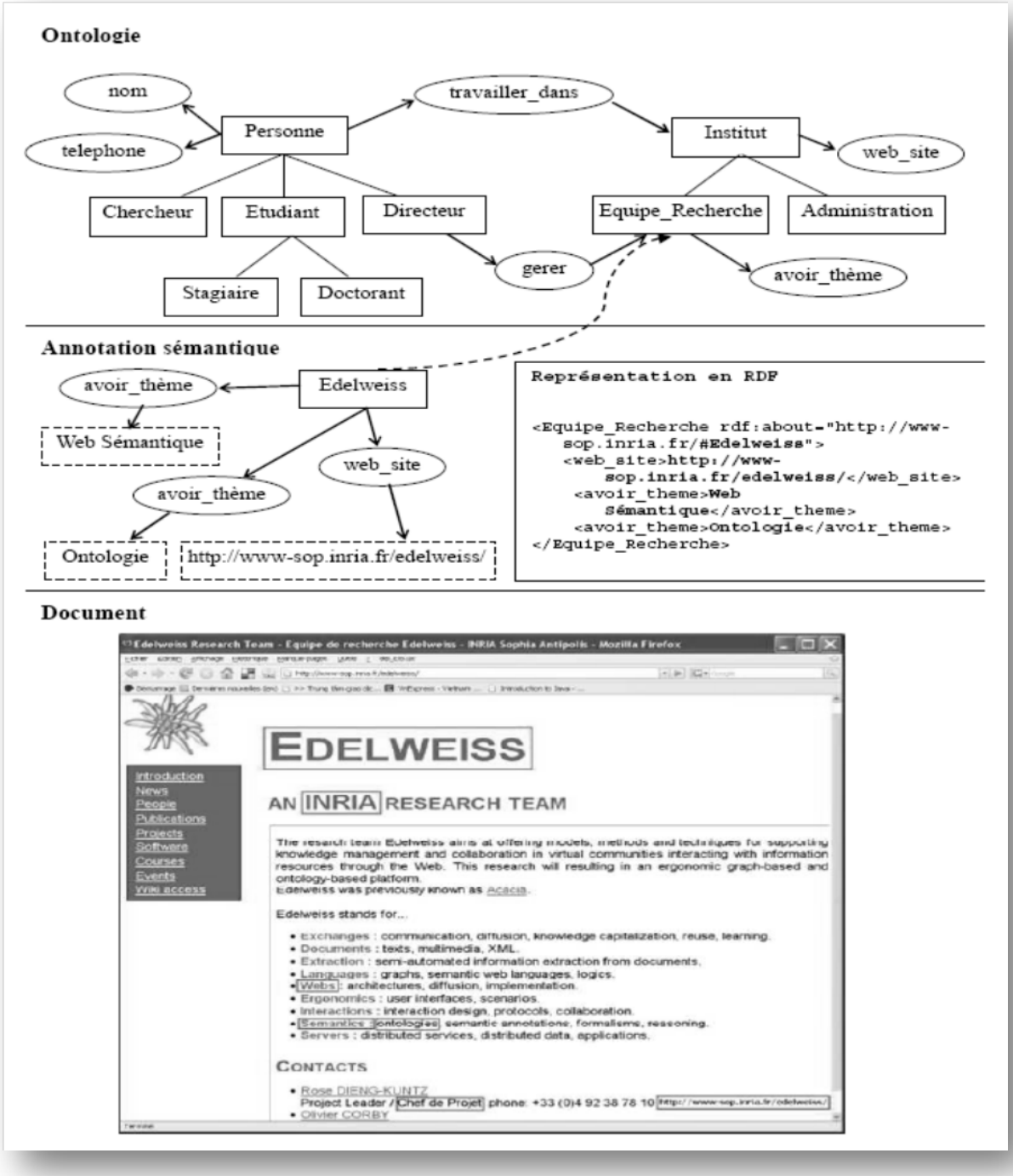


Figure 2.6. Utilisation de l'annotation sémantique [Luong, 2007].

2.3.2. Les caractéristiques de l'annotation sémantique

Pour caractériser la spécificité des annotations de ressources documentaires numériques, plusieurs caractéristiques ont étés distingués [Prié et Garlatti, 2004] :

- Une ressource peut être un document entier ou à un fragment de celui-ci
- Une ressource peut contenir des informations de natures différentes (du texte, des images, du son, de la vidéo, etc.).

- D'autre part, même un texte peut être plus ou moins structuré.

- Pour la création des annotations sémantiques le traitement peut être : manuel, automatisé ou semi-automatiques.

- Les modèles formels utilisés pour l'annotation sémantique peuvent être plus ou moins structurés. La Figure 2.7. montre un exemple d'annotation dans le contexte des ontologies.

- lorsqu'une annotation est stockée à l'extérieur du document source, elle est dite «*débarquée*» vis-à-vis de la ressource documentaire source, lorsqu'elle est ajoutée directement au contenu du document d'origine et est dite «*embarquées*».

- L'annotation sémantique permet de nombreuses applications (comme la recherche d'information sémantique, la catégorisation, la composition de documents, etc.) [Prié et Garlatti, 2004].

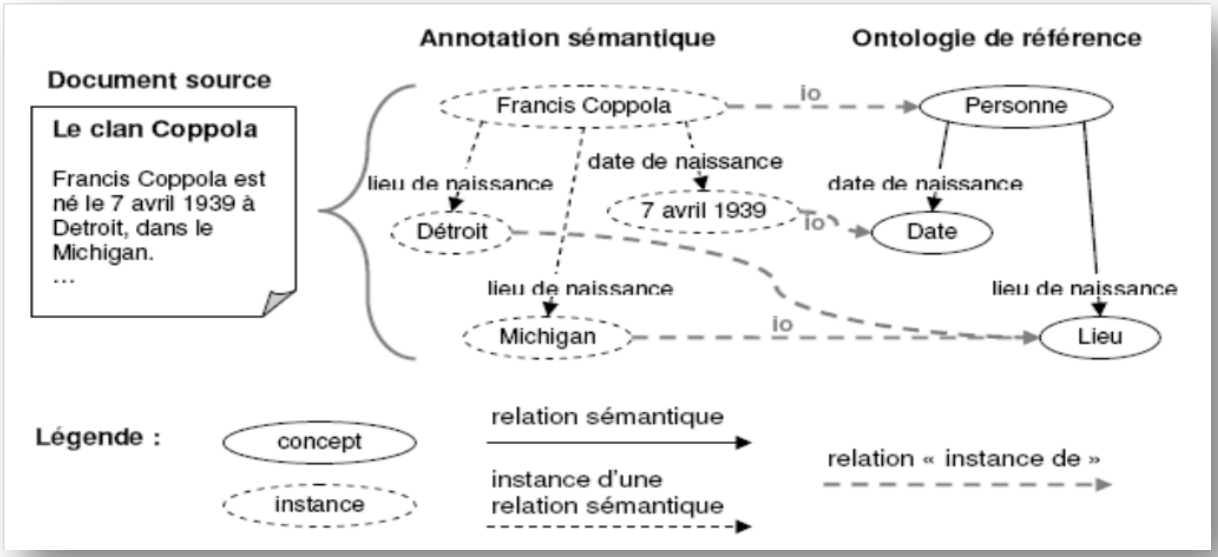


Figure 2.7. Exemple d'une annotation sémantique orchestrée par une ontologie [Amardeilh, 2007].

2.3.3. L'annotation sémantique et les RTO

Les RTO n'offrent pas toutes le même niveau de représentation de la connaissance : les taxonomies étant moins expressives que les thesaurus, eux-mêmes moins formels et précis qu'une ontologie. [Amardeilh, 2007].

Les thesaurus permettent de constituer un premier niveau d'annotation sémantique par leur capacité à exprimer un vocabulaire contrôlé ainsi qu'un ensemble de relations lexicales entre les termes du thesaurus. Les annotations dans ce contexte, consistent en de simples pointeurs vers les termes du thesaurus, appelés aussi descripteurs.

Les annotations sémantiques créées à partir d'une ontologie améliorent la recherche d'information, car elles peuvent contribuer à faire jouer des mécanismes d'inférence et de

raisonnement. Il est possible qu'une même ressource puisse être annotée par différentes ontologies en même temps, offrant ainsi différents points de vue sur un même contenu.

Les ontologies et les thesaurus peuvent être considérés comme complémentaires dans le contexte de l'annotation sémantique, car, ils offrent des accès aux contenus de façons différentes : celui de la conceptualisation d'un domaine pour l'ontologie et celui du vocabulaire pour les thesaurus. Les thesaurus et autres vocabulaires contrôlés peuvent être combinés avec les annotations sémantiques basées sur l'ontologie du domaine. [Amardeilh, 2007].

2.3.4. Les Langages de l'annotation sémantique

Plusieurs langages de représentation de la connaissance, et d'ontologies, ont été définis par les chercheurs du domaine de l'Intelligence Artificielle, comme *OntoLingua*, *LOOM, OCML* et *F-Logic*, qui sont issus de théories de la logique comme les logiques de description, les prédicats du premier et second ordre, les frames, etc. [Baget et al., 2004]. Puis, avec la naissante du Web Sémantique, de nouveaux langages pour la spécification d'ontologies et d'annotations sémantiques ont étés développés, tels que RDF, RDFS et OWL, que nous allons présentés dans cette section.

2.3.4.1. RDF : Resource Description Framework

RDF est un modèle standard pour l'échange de données sur le Web. RDF possède des fonctionnalités qui facilitent la fusion de données même si les schémas sous-jacents diffèrent, et il prend en charge l'évolution des schémas au fil du temps sans exiger que tous les consommateurs de données soient modifiés. La première recommandation de RDF[10] est publiée par W3C en 1999. Le premier objectif de RDF est la description de ressources [Baget et al., 2004]. RDF étend la structure de liaison du Web pour utiliser les URIs pour nommer la relation entre les choses ainsi que les deux extrémités du lien (cela est généralement appelé un « triple »). En utilisant ce modèle simple, il permet aux données structurées et semi-structurées d'être mélangées, exposées et partagées entre différentes applications [Daconta et al., 2003]. Au niveau le plus simple, le RDF est un Langage pour décrire les ressources.. Alors que les documents XML attachent des méta-données à des parties d'un document, une utilisation de RDF consiste à créer des métadonnées sur le document comme entité autonome. En d'autres termes, au lieu de marquer les éléments internes d'un document, RDF capture des métadonnées sur les "externes" d'un document.

Un document RDF peut contenir plusieurs descriptions, chaque une peut correspond à un ensemble d'énoncés (ou « statements ») au sujet d'une ressource. Un énoncé RDF est aussi appelé triplet car il est composé de trois éléments, *sujet- prédicat-objet*, où :

1) *le sujet* représente la ressource décrite, i.e. tout document accessible sur le Web comme les pages HTML, les documents textuels (PDF, Ms Word) ou multimédias (images, vidéo), etc.,

2) *le prédicat* représente la propriété descriptive, i.e. une caractéristique spécifique, un attribut ou une relation, utilisée pour décrire une ressource.

[10] http://www.w3.org/RDF/

3) *l'objet* représente la valeur de cette propriété, soit une valeur littérale, comme un nombre entier ou une chaîne de caractère, soit une autre ressource accessible par son URI. Par contre, une valeur littérale ne peut en aucun cas être le sujet d'un énoncé.

Un triplet peut s'écrire « prédicat (sujet, objet) ».

D'autres notations peuvent être utilisées pour afficher des données RDF. Les énoncés RDF peuvent être représentés par les graphes orientés étiquetés puisque le modèle RDF a été influencé par les réseaux sémantiques [Baget et al., 2004]. Dans ces graphes, un nœud représente un sujet ou un objet et l'arc un prédicat dont l'origine est le sujet et la destination l'objet de l'énoncé (cf. Figure 2.8). RDF peut aussi être exprimé en XML.

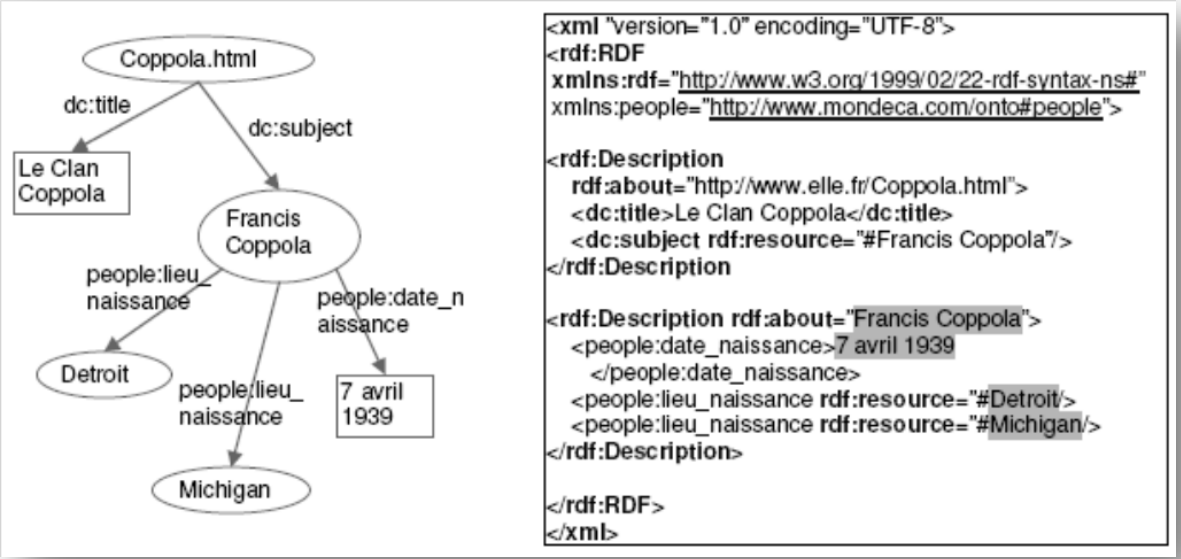


Figure 2.8. Exemple d'annotation sémantique en RDF (notation graphique à gauche et XML à droite).

2.3.4.2. RDFS : Resource Description Framework Schema

RDFS[11] est un langage utilisé pour la définition de schémas RDF. Alors qu'RDF exprime les relations sémantiques au niveau des instances sous la forme de triplets, RDFS exprime donc les relations au niveau des classes et des propriétés (les prédicats RDF), contraignant ainsi les instances possibles dans les triplets RDF [Baget et al., 2004]. RDFS est construit au dessus de RDF, c'est-à-dire que ce langage réutilise la syntaxe des triplets RDF et l'étend avec de nouveaux éléments pour définir les classes, *rdfs:class*, et les propriétés, *rdfs:property*. L'élément *rdfs:subClassOf* permet de spécifier la taxonomie des classes.

Les définitions de propriétés sont restreintes par deux contraintes :

1. *rdfs:domain* qui représente le domaine de la propriété indiquant la classe à laquelle cette propriété s'applique et,
2. *rdfs:range* qui représente la portée de la propriété indiquant la classe dont les instances seront les valeurs autorisées pour cette propriété.

[11] http://www.w3.org/TR/rdf-schema/

Dans la Figure 2.9., «Personnalité » étant une sous-classe de la classe « Personne ». Nous définissons aussi la propriété « Lieu_Naissance » , qui a pour domine la classe « Personne » et « Lieu » comme sa portée.

Les Schémas RDF apportent des caractéristiques importantes à l'annotation sémantique en RDF par la définition d'une ontologie relativement simple à travers une taxonomie de classes, de leurs propriétés ainsi que la restriction du domaine et de la portée des propriétés. [Amardeilh, 2007].

```
<xml "version="1.0" encoding="UTF-8" xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">

<rdf:Description rdf:about="http://www.elle.fr/Coppola.html">
   <rdf:type rdf:about="#Article"/>
   <rdfs:label>Le Clan Coppola</rdfs:label>
   <dc:subject rdf:resourceRef="#FFCoppola"/>
</rdf:Description>

<rdfs:Class rdf:ID="#Lieu">
<rdfs:Class rdf:ID="#Personne">
<rdfs:Class rdf:ID="#Personnalité">
   <rdfs:subClassOf rdf:resource ="#Personne"/>
</rdfs:Class>

<rdf:Property rdf:about="lieu_naissance">
   <rdfs:domain rdf:resource="#Personne"/>
   <rdfs:range rdf:resource="#Lieu"/>
</rdf:Property>
<rdf:Property rdf:about="date_naissance">
   <rdfs:domain rdf:resource="#Personne"/>
   <rdfs:range rdf:resource="http://www.w3.org/2000/10/XMLSchema#string"/>
</rdf:Property>

<Personnalité rdf:about="FFCoppola">
   <rdfs:label>Francis Coppola</rdfs:label>
   <lieu_naissance rdf:resource="#Detroit"/>
   <lieu_naissance rdf:resource="#Michigan"/>
   <date_naissance>7 avril 1939</date_naissance>
</Personnalité>
</xml>
```

Figure 2.9. Exemple d'annotation sémantique basée sur un schéma RDFS [Amardeilh, 2007].

2.3.4.3. OWL : Web Ontology Language

OWL[12], est une recommandation du W3C en février 2004, est le plus expressif des langages ontologiques pour le Web [Baget et al., 2004]. OWL bénéficié d'une base théorique solide en logique et d'une volonté de la part de ses concepteurs pour créer un langage approprié à une utilisation dans le cadre du Web Sémantique.

Le langage OWL fournit des mécanismes pour créer tous les composants d'une ontologie : classes, instances, propriétés et axiomes. OWL repose également sur la syntaxe des triplets RDF et réutilise certaines des constructions RDFS. Comme en RDFS, les classes peuvent avoir des sous-classes, fournissant ainsi un mécanisme pour le raisonnement et l'héritage des propriétés. Par contre, en OWL, on distingue : les propriétés objet (*object*

[12] http://www.w3.org/2004/OWL/

property), et les propriétés type de données (*datatype property*).

Les axiomes fournissent de l'information au sujet des classes et des propriétés, spécifiant par exemple l'équivalence entre deux classes.

Le langage OWL se compose de trois sous-langages qui symbolisent une expressivité croissante : OWL-Lite, OWL-DL, OWL-Full [Baget et al., 2004].

Les ontologies OWL sont des fichiers texte dont l'extension peut être owl ou rdf puisque le langage OWL repose sur RDF et RDFS, en y ajoutant notamment des nouvelles balises pour gagner en précision.

La Figure 2.10. Illustre un exemple d'annotation sémantique en OW Lite.

```
<xml "version="1.0" encoding="UTF-8" xmlns:owl="http://www.w3.org/2002/07/owl#">
<rdf:RDF>
<owl:Ontology rdf:about="http://www.mondeca.com/onto#people">
   <dc:title>People</dc:title>
</owl:Ontology>

<owl:Class rdf:ID="Lieu">
<owl:Class rdf:ID="Personne">
<ow:Class rdf:ID="Personnalité">
   <rdfs:subClassOf rdf:resource ="#Personne"/>
</owl:Class>

<owl:ObjectProperty rdf:ID="lieu_naissance">
   <rdfs:domain rdf:resource="#Personne"/>
   <rdfs:range rdf:resource="#Lieu"/>
</owl:ObjectProperty>
<owl:DatatypeProperty rdf:about="date_naissance">
   <rdfs:domain rdf:resource="#Personne"/>
   <rdfs:range rdf:resource="http://www.w3.org/2000/10/XMLSchema#string"/>
</owl:DatatypeProperty>

<owlThing rdf:ID="FFCoppola">
   <rdf:type rdf :resource="#Personnalité"/>
   <rdfs:label>Francis Coppola</rdfs:label>
   <lieu_naissance rdf:resource="#Detroit"/>
   <lieu_naissance rdf:resource="#Michigan"/>
   <date_naissance>7 avril 1939</date_naissance>
</Personnalité>

<rdf:Description rdf:about="http://www.elle.fr/Coppola.html">
   <rdf:type rdf:about="#Article"/>
   <rdfs:label>Le Clan Coppola</rdfs:label>
   <dc:subject rdf:resourceRef="#FFCoppola"/>
</rdf:Description>

</rdf:RDF>
</xml>
```

Figure 2.10. Exemple d'annotation sémantique en OW Lite [Amardeilh, 2007].

2.3.5. Les Langages d'interrogation des annotations sémantiques

La représentation de connaissances dans la nouvelle génération du Web sémantique est importante mais la capacité de faire des requêtes de ces connaissances joue aussi un rôle crucial. Actuellement ils existent un certains nombre de langages de requêtes connus tels que, RQL [Karvounarakis et al., 2002], OntoQL [Jean et al., 2006], RDQL[13], SquishQL[14] [Miller et al., 2002] et SPARQL[15]. Dans ce qui suit, nous allons présentés ces langages de requêtes,

[13] http://www.w3.org/Submission/2004/SUBM-RDQL-20040109/
[14] http://swordfish.rdfweb.org/rdfquery/
[15] http://www.w3.org/TR/rdf-sparql-query/

en mettant l'accent, particulièrement sur le langage SPARQL.

2.3.5.1. RQL

RQL [Karvounarakis et al., 2002} a été l'un des premiers langages défini. Il est basé sur un modèle de données formel ayant la particularité de distinguer les différents niveaux manipulés: données (instances), ontologies (concepts ontologiques) et modèles d'ontologies.

2.3.5.2. OntoQL

OntoQL (Jean et al., 2006) est un langage de requête pour l'exploitation des bases de donnés à base ontologique. Implémenté sur la BDBO OntoDB, il permet de définir, modifier et interroger les ontologies et ses données et d'étendre, si besoin, le modèle d'ontologie utilisé pour la définition des ontologies. Ainsi, OntoQL permet la manipulation directe des modèles et des concepts ontologiques tout en cachant les représentations internes.

2.3.5.3. RDQL

RDQL (RDF Data Query Language) dérivé des travaux de Libby Miller sur sur SquishQL, RDQL traite RDF comme une base de données et permet de faire des requêtes en utilisant des modèles et des contraintes sur les triplets d'un modèle de données RDF. Andy Seaborne note que RDQL est destiné à être utilisé dans des scripts et pour des expériences au niveau des langages de modélisation. Ce langage n'est pas standardisé, il existe de nombreuses implémentations. Sa syntaxe est très proche de SQL (Structured Query Language).

```
SELECT variable [, variable]*
FROM documents_rdf [, documents_rdf]*
WHERE modele_de_triplets
AND restrictions_booléennes
USING définition_des_raccourcis
```

Figure 2.11. Syntaxe de requête RDQL

2.3.5.4. SPARQL

Le langage SPARQL (SPARQL Protocol and RDF Query Language), est un langage de requête défini par le W3C pour la première fois en 2008, puis étendu en 2013. Ce langage est un langage d'interrogation de bases de connaissances, à comparer avec le langage d'interrogation des bases de données relationnelles SQL adapté pour la structure sous forme de triplets des bases RDF. La dernière norme SPARQL 1.1 permet l'interrogation, nommée SPARQL [Harris et Seaborne, 2013], l'ajout et la suppression, désignés par SPARQL Update [Gearon et al., 2013], des données d'une base RDF. Une base qui peut être interrogée par des requêtes SPARQL via Internet propose un point d'accès – appelé endpoint SPARQL – où envoyer les requêtes via HTTP. SPARQL est largement utilisé dans le domaine de recherche et d'extraction d'informations [Prud'hommeaux et Seaborne, 2007]. Tout comme le SQL, SPARQL est un langage de requête qui se doit de respecter une certaine syntaxe. Il permet

d'interroger et de construire un graphe RDF. Le langage SPARQL est basé sur la correspondance des patrons de graphe. Le patron de graphe le plus simple est le patron de triplets (comme un triplet en RDF) mais il possède la capacité d'exprimer des variables de requête dans les positions du sujet, de la propriété ou de l'objet d'un triplet. D'autre part, SPARQL intègre également des balises (tags) spécifiques telles que le patron de graphe optionnel, l'union et l'intersection des patrons, le filtrage, les opérateurs de comparaison des valeurs, etc. permettant d'effectuer des requêtes plus efficaces et flexibles.

La recommandation SPARQL du W3C fournit trois briques de base pour construire une telle solution :

- un langage de requête permettant de décrire les graphes RDF recherchés sous forme de triplets (arcs du graphe) et de contraintes sur ces triplets (filtres) ;
- un format de réponse en XML donnant les valeurs des variables sélectionnées dans la requête pour chaque graphe trouvé ;
- un protocole permettant d'échanger les requêtes et leurs réponses avec un serveur distant, notamment en utilisant une architecture de services web.

Une requête SPARQL possède un entête et, le plus souvent, un corps de requête.

L'entête contient le mot-clé de la requête « SELECT, ASK, INSERT, etc. » avec les éléments demandés en résultat de la requête. Le corps de requête contient un ensemble de « triplets de recherche ». Dans les triplets de recherche, un élément sujet, relation ou objet peut être remplacé par une variable, sous la forme d'un nom de variable préfixé du caractère « ? ». Un ensemble de triplets de recherche forme un motif de graphe. On appelle solution d'un motif de graphe un ensemble de données RDF qui forme un morphisme du motif. Ce morphisme substitue les variables du motif par des ressources de façon à ce que le motif de graphe et l'ensemble de données RDF soient identiques. Le corps d'une requête commence par le mot clé WHERE suivi d'un motif de graphe entre accolades ou de plusieurs motifs organisés par des opérateurs.

Interrogation

Les requêtes d'interrogation SPARQL sont de 4 sortes différentes :

- Une requête ***SELECT*** extrait des résultats sous forme d'un tableau, contenant les ressources solutions à un motif de graphe donné et correspondantes aux variables présentes dans l'entête.
- Une requête ***ASK*** retourne une valeur booléenne s'il existe au moins une solution à un motif de graphe donné.
- Une requête ***CONSTRUCT*** extrait les solutions à un motif de graphe donné sous la forme d'un nouveau graphe RDF.
- Une requête ***DESCRIBE*** extrait tous les triplets de la base contenant les ressources données.

Les requêtes SPARQL disposent d'opérateurs et de filtres pour composer le corps des requêtes :

- Les 2 principaux opérateurs de disjonction :
  - L'opérateur *UNION* permet de définir une disjonction entre deux motifs.

- L'opérateur *OPTIONAL* permet de définir un sous-motif de graphe dont la résolution est optionnelle pour la satisfaction de la requête.

– Les filtres, définis par *FILTER*, vérifient que les variables satisfont certaines conditions exprimées par des des expressions telles que :
  - Les opérateurs (<, >, ! =, <=, >=) définissent des conditions sur les valeurs des variables.
  - La fonction *EXISTS* retourne un booléen en fonction de l'existence d'un motif de graphe.

Les filtres peuvent être organisés par les opérateurs logiques de conjonction (&&) et de disjonction (||).

Voici un exemple simple de requête SPARQL : Considérons le schéma RDF suivant issu d'une image annotée (Figure 2.12).

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:foaf='http://xmlns.com/foaf/spec/'
   xmlns:dc="http://purl.org/dc/elements/1.1/">
  <rdf:Description rdf:about='/media/home/youl/photo128.jpg'>
    <dc:subject>Vacances Pâques 2007</dc:subject>
    <dc:title>Julien à Athènes</dc:title>
    <dc:description>Julien devant le Parthenon</dc:description>
    <foaf:fullName>Julien Pierre de la Brière</foaf:fullName>
  </rdf:Description>
</rdf:RDF>
```

Figure 2.12. Exemple de document RDF.

Considérons à présent que l'on cherche toutes les photos dont le titre contient «Julien». Pour ce faire, la requête SPARQL doit tout d'abord déclarer les espaces de noms utilisés comme suit (Figure 2.13).

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?about
FROM <index.rdf>
WHERE {
  ?photo rdf:about ?about .
  ?photo dc:title ?title .
  FILTER regex(?title, "^Julien")
}
```

Figure 2.13. Exemple de requête SPARQL

2.3.6. Framework et outils de support pour les annotations sémantiques

Un système, un outil, ou un framework d'annotation sémantique est un outil logiciel permettant de créer, d'insérer et de gérer des annotations sémantiques liées à une ressource documentaire donnée et il a pour objectif principal, de faciliter et d'alléger le lourd travail que demande l'annotation manuelle dans les pages Web. La majorité d'entre eux ont essaie d'être de plus en plus automatisés, notamment, grâce aux méthodes issues de l'Extraction d'Information et des Systèmes d'Apprentissage [Kahan et al., 2001]. Un comparatif et une classification des outils d'annotations, a été donné par Amardeilh [Amardeilh, 2007]. L'ensemble des outils étudiés, ont étés réparties en trois grandes classes : 1) les outils consacrés à l'annotation sémantique dans le contexte du développement du Web Sémantique, tels que : *Annotea*[16], *COHSE Annotator*[17], *OntoMat-Annotize*[18] ; 2) les outils utilisant les annotations afin de peupler une ontologie, tels que : *MnM*[19], *Melita*[20], *Armadillo*[21] ; 3) Les outils récents qui sont considéré comme intéressants comme des pistes de recherche futur, mais, pas suffisamment mature, tels que ; *OntoText*[22], *SmartWeb*[23], h-*TechSight*[24].

[16] http://www.w3.org/2001/Annotea/
[17] http://cohse.semanticweb.org
[18] http://annotation.semanticweb.org/ontomat/index.html
[19] http://kmi.open.ac.uk/projects/akt/MnM/
[20] http://nlp.shef.ac.uk/melita/
[21] http://nlp.shef.ac.uk/wig/armadillo_home.html
[22] http://tcc.itc.it/projects/ontotext/
[23] http://smartweb.dfki.de/main_pro_en.pl?infotext_en.html
[24] http://www.h-techsight.org/

2.4. Conclusion

Dans ce chapitre nous avons dressés le portrait de l'annotation sémantique. Nous avons tout d'abord donnés une brève introduction du web sémantique. Ensuite, nous avons soulignés les différentes définitions et caractéristiques de l'annotation sémantique. Puis, nous avons présentés le lien existant entre l'annotation sémantique et les ressources terminologiques ou ontologiques existantes. Ensuite, nous avons présentés quelques langages actuels du web sémantique pour la représentation de ces ressources et leur impact quand à l'écriture des annotations sémantiques. En fin, nous avons présentés quelques langages d'interrogation et outils de construction des annotations sémantiques.

Nous avons vu que l'annotation sémantique est intrinsèquement liée à la modélisation d'une ontologie. En effet, cette ontologie va représenter les concepts, attributs et relations d'un domaine à l'aide d'un langage de représentation des connaissances orienté Web. Les annotations sémantiques sont structurées à l'aide de cette ontologie et leurs valeurs pointent vers les instances de l'ontologie de référence ou, dans certains cas, directement vers les concepts eux-mêmes. L'annotation sémantique à partir des ontologies est la méthode la plus pertinente et la plus prometteuse.

L'utilisation des annotations sémantiques, en recherche d'information est devenue une voie très explorée. Dans ce contexte que dans le chapitre suivant, nous dressons un état de l'art sur l'exploitation des annotations pour la recherche d'information dans le Web Sémantique.

Chapitre 3. Exploitation des annotations sémantiques pour la recherche d'informations dans le Web Sémantique

3.1. Introduction

La recherche d'information sur le Web Sémantique partage les mêmes objectifs qu'une recherche d'information classique. Le but est de retrouver des résultats pertinents en réponse à une requête posée par un utilisateur. Dans le cadre du Web Sémantique, on aura accès au *contenu sémantique* des documents, notamment, grâce aux *annotations*. L'importance de la recherche dans ce contexte est clair pour les raisons suivantes : elle complète de recherche d'informations classiques en fournissant des services de recherche centré sur les entités, les relations et les connaissances et le développement du web sémantique exige également les paradigmes améliorés de recherche en vue de faciliter l'acquisition, le traitement, le stockage et la recherche de l'information sémantique. L'utilisation des annotations sémantiques, en recherche d'information est devenue une voie très explorée ces dernières années. Cela permet de dépasser de nombreux problèmes liés aux comparaisons terme à terme entre documents ou entre documents et requêtes. Dans cette optique, plusieurs systèmes de recherche d'informations à base d'annotations et ontologies ont été proposés. Ce type de moteurs de recherche du web sémantique vise à améliorer les résultats de la recherche en utilisant les normes et les langages du Web sémantique [Latreche et al., 2009].

Dans ce chapitre, nous allons passer en revue, les différentes approches existantes, en mettant l'accent particulièrement sur les approches liées à notre approche.

3.2 La recherche sémantique

Les moteurs de recherche du web sémantique visent à améliorer les résultats de la recherche en utilisant les normes et les langages du Web sémantique. Il est possible de classer ces types de moteurs de recherche en trois groupes :

3.2.1 La recherche d'ontologies

Les moteurs de recherche d'ontologie sont des systèmes qui recherchent l'information sous forme d'ontologies. Les résultats de recherche sont des ontologies entières, des parties d'ontologie ou des éléments d'ontologie, voici quelques échantillons : Swoogle [Ding et al., 2004], OntoSelect [Buitelaar et al., 2004], OntoSearch2 [Pan et al., 2006] , Watson [dAquin et al., 2007].

3.2.2 La recherche sémantique basée sur le contexte

L'objectif ultime de ces moteurs est l'amélioration des performances des moteurs de recherche traditionnels (particulièrement en ce qui concerne les mesures de précision et de

rappel), en utilisant l'information de contexte (représentée par les ontologies de domaine et les annotations). La qualité de leurs résultats dépend fortement des propriétés qualitatives et quantitatives des annotations sémantiques utilisées. Le plus gros problème de ces moteurs de recherche est qu'ils sont limités à un certains contextes, en voici quelques échantillons :

SEAL [Stojanovic et al., 2001] et son mécanisme sémantique pour l'acquisition, la structuration et le partage de l'information communautaire entre l'homme et / ou agents de la machine. Les ontologies constituent le fondement de SEAL. L'épine dorsale du système consiste en un entrepôt de connaissances (c'est-à-dire l'ontologie et la base de connaissances), et le système Ontobroker, c'est-à-dire le principal mécanisme d'inférence.

SCORE « The Semantic Content Organization and Retrieval Engine » [Sheth et al. 2002] utilise les techniques de classification et d'extraction d'informations pour extraire les métadonnées de sources textuelles. Ces métadonnées sont utilisées plus tard pour la recherche sémantique.

QuizRDF [Davies et al., 2002] combine la recherche par mots clés avec la recherche et la navigation sur les annotations RDF(S). QuizRDF peut être utilisé comme un moteur de recherche Internet classique en entrant un ensemble de termes de recherche ou une requête en langage naturel et produire une liste de liens vers les pages Web en se basant sur des algorithmes statistiques de façon habituelle.

[Zhu et al., 2002] ont proposé une approche de recherche sémantique basée sur la correspondance des graphs RDF. Lorsque l'utilisateur émit une requête en entrée, elle est interprétée en graphe requête RDF. Puis arrive l'étape du calcul de la similarité entre le graphe requête et chaque graphe ressources candidat.

OWLIR « Ontology Web Language and Information Retrieval » [Mayfield & Finin, 2003], le système veut combiner les techniques de recherche d'information efficaces et les mécanismes d'inférence du Web Sémantique. Les requêtes peuvent alors être soit classiques dans le sens RI (avec possibilité d'intégrer des balises), soit des faits à prouver.

TAP [Guha et al., 2003] effectue une recherche à base de graphes sur les graphes RDF du Web. Il commence à un ou plusieurs nœuds ancre dans le graphe RDF, qui doivent être mis en correspondance avec les termes de la requête. Une recherche BFS (breath first search) est réalisée sur le graphe RDF, recouvrant une quantité prédéfinie de triplets.

DOSE « a distributed open semantic elaboration platform » [Bonino et al., 2003] est organisée comme une collection de services faiblement couplés. Une des caractéristiques notables du système est que les méthodes d'annotation et de recherche sont basées sur des fragments de document. Le moteur accepte les requêtes exprimées en tant que mots-clés, textes courts ou des concepts et interagit avec le mappeur sémantique créant un ensemble pondérée de concepts sémantiquement liés à la requête d'origine.

BioPatentMiner [Bamba and Mukherjea, 2004] ont présenté une approche pour le classement des résultats des requêtes du Web sémantique. L'ensemble des triples retournés par une requête RDQL est classé en fonction de divers facteurs. Un aspect central pour la

pondération des nœuds dans les graphes résultats est une version adaptée de l'algorithme de Kleinberg HITS.

*KIM « Knowledge and Information Management » [*Kiryakov et al., 2004] repose sur l'extraction d'informations et associé les mots des documents avec les concepts d'une ontologie. Avant l'indexation, des documents sont enrichis avec des identificateurs des concepts ontologiques des mots représentés dans le document. Des requêtes sont formées des concepts et des relations de l'ontologie.

InWis [Priebe et al., 2004] ont présenté un moteur de recherche sur des métadonnées RDF mis en œuvre dans le portail de connaissances. InWiss qui prévoit des requêtes approximatives en utilisant une approche basé sur la similarité connu dans les modèles classiques de la recherche d'information.

[Contreras et al., 2004], L'approche comprend un annotateur automatique basée sur les ontologies, un moteur de recherche sémantique avec une interface en langage naturel, un outil de publication web permettant la navigation sémantique, et un composant de visualisation en 3D. Il existe deux types d'interface pour le moteur de recherche. La première est basée sur des formulaires représentant le domaine des concepts et des relations existantes. Le deuxième type d'interface accepte en entrée une requête écrite en langage naturel qui est ensuite transformée en une requête RDQL.

[Song et al., 2005] ont présenté un modèle de recherche d'information à base d'ontologie pour le web sémantique. En utilisant OWL Lite comme langage d'ontologie standard, l'ontologie est générée par le biais de la translation et l'intégration des ontologies de domaine. Les termes définis dans l'ontologie sont utilisés comme métadonnées pour marquer le contenu du Web. En outre, la requête devrait être représentée en utilisant des termes d'indexation sémantique.

[Vallet et al 2005] ont proposé un modèle pour l'exploitation des bases de connaissances à base d'ontologie pour améliorer la recherche sur de grands entrepôts de documents. L'approche inclut un système à base d'ontologie pour l'annotation semi-automatique des documents, et un système de recherche à base de ces annotations. La recherche sémantique (requête RDQL) est combinée avec la recherche par mots-clés pour parvenir à la tolérance incomplète des BC.

[Zhang et al., 2005] ont présenté un modèle amélioré pour la recherche dans des portails sémantiques. Ils intègrent la recherche à base de texte avec une version flou de la logique de description ALC. L'ensemble de résultats de la requête est représentée comme une classe dans la base de connaissances.

CORESE [Corby *et al.*, 2004 ; Corby et al., 2006] est un moteur de recherche pour le Web sémantique fondé sur une ontologie. Il permet de retrouver des ressources web annotées en RDF(S). L'ontologie est représentée en graphe conceptuel au dessus de RDFS. Le moteur de recherche prend en considération la subsomption des concepts et des relations pour calculer un appariement entre la requête et les annotations. La requête est traduite en graphe

conceptuel et l'appariement correspond à la projection de ce graphe sur les graphes conceptuels relatifs aux documents.

*OntoSearch [*Jiang and Tan, 2006] ont présenté un moteur de recherche plein texte pour la recherche des documents dans le Web sémantique. La procédure de traitement des requêtes du système est similaire à celle d'un moteur de recherche traditionnel. Un utilisateur soumet des requêtes composées de mots clés au système, qui retourne une liste initiale de documents. De ces documents des métadonnées sémantiques sont extraites et utilisés pour la recherche par propagation d'activation dans l'ontologie. Le processus de la propagation d'activation infère les concepts qui sont sémantiquement liés à l'ensemble initial de concepts.

MESH [Castells et al. 2007] combine la recherche basée sur SPARQL avec la recherche plein texte. Pour classer les résultats de la requête SPARQL ils pondèrent les annotations de documents avec les concepts d'une ontologie en utilisant tf*idf comme mesure.

K-Search [Bhagdev et al. 2008] un system qui permet une recherche hybride, une méthode de recherche qui supporte la recherche de documents et de la connaissance par l'intermédiaire de la combinaison flexible de la recherche à base d'ontologie et la correspondance à base de mots clés.

3.2.3 Découverte d'associations sémantiques

L'objectif est de trouver des relations sémantiques entre les termes saisi (généralement deux) et puis classer les résultats on se basant sur les métriques de distances sémantiques. :

PISTA « Passenger Identification, Screening, and Threat » [Aleman-Meza et al., 2003] ont présenté une approche pour le classement des associations sémantiques dans le Web sémantique. A partir de deux entités d'un graphe RDF ils visent à trouver des associations sémantiques entre eux et de classer ces associations sémantiques en fonction de leur importance.

[Rocha et al., 2004] ont présenté une approche hybride pour la recherche dans le web sémantique qui combine la recherche plein texte avec la recherche par propagation d'activation (spread activation) dans une ontologie. La recherche commence avec une requête sur la base de mot-clé. Les résultats de la recherche textuelle sont des instances d'une ontologie. Ces instances sont utilisées pour lancer une recherche par propagation d'activation en ontologie pour trouver des instances supplémentaires.

[Bangyong et al., 2005] ont présenté une approche préliminaire à la recherche d'association dans le Web sémantique. Ils proposent de produire un réseau bayésien d'une ontologie et de l'employer pour trouver des instances non trouvés par la requête initiale. La recherche initiale est faite par les techniques traditionnelles de recherche d'informations sur le Web qui renvoie un ensemble d'instances.

[Choudhury et Phon-Amnuaisuk, 2006] ont présenté un système de recherche similaire à celui présenté par (Rocha et al., 2004). Ils implémentent un sous-ensemble de fonctionnalité de pondération du système présenté dans (Rocha et al., 2004) et testent leur système sur un plus

petit ensemble de données.

SemDis [Halaschek et al. 2004] leurs objectifs est de trouver et classer les associations sémantiques. Techniquement parlant, après la phase de découverte, il y a souvent de nombreuses associations sémantiques, donc une politique de classement qui doit être utilisé. Dans *SemDis* , certains de ces critères ont été introduites par les algorithmes de classement : *Contexte*, *Subsomption, Longueur du chemin, Confiance*. Un tel système permet l'extraction automatique des métadonnées résultant des annotations sémantiques des entités du Web via RDF, nous permet d'utiliser les ontologies et divers bases de connaissances.

3.3 Les approches visant à formuler des requêtes formelles

Nous présentons dans cette section un aperçu des différents types d'approches existants visant à formuler ou à générer des requêtes formelles/graphes requêtes. Pour cela, ces approches sont présentées le long du continuum de formalisation, du plus formel au plus permissif.

3.3.1 Langages formels

A la tête de continuum se trouvent les langages de requêtes formels comme SPARQL[25]. De tels langages sont totalement inadaptés aux utilisateurs finals, puisque, exprimer des requêtes formelles, suppose que les utilisateurs doivent connaître et respecter la syntaxe du langage utilisé, et le plus contraignant, de connaître le formalisme de représentation de connaissances utilisé, c.-à.d., le vocabulaire, la syntaxe et la structure (schéma) de la base à interrogée. Dans [Elbassuoni et al., 2010], les auteurs étend le langage SPARQL et son mécanisme d'interrogation pour prendre en compte les mots-clés, surtout, quand les utilisateurs ne connaissent pas exactement le schéma à interroger; de telle approche suppose que les utilisateurs maitrisent le langage formelle SPARQL.

3.3.2 Constructeurs de requêtes par auto-complétion

En suite, nous trouvons les approches qui assistent et aident les utilisateurs à formuler leur requêtes dans de tels langages. Des interfaces comme *SparQLed*[26] *et Flint*[27] implémentent des fonctionnalités simples comme l'auto-complétion et la coloration syntaxique. Ces constructeurs ne créent pas la requête à la place des utilisateurs; mais, ils les aideront à l'écrire et ils sont destinés spécialement aux utilisateurs qui connaissent l'écriture de requêtes SPARQL. L'idée générale est de présenter à l'utilisateur des complétions de requêtes et de prédire la prochaine écriture. Il est à noter que ces types de constructeurs de requêtes ne soient pas visuellement riches, mais, ils fournissent un contrôle en profondeur de la requête et sont également plus rapides en termes de temps de création de requête. Les auteurs, dans [Smits et al., 2014] ont utilisés une stratégie d'auto-complétion pour construire des requêtes SPARQL par un suivi stricte d'une certaine grammaire.

[25] http://www.w3.org/TR/rdf-sparql-query/
[26] http://sindicetech.com/sindice-suite/sparqled/
[27] http://openuplabs.tso.co.uk/demos/sparqleditor

3.3.3 Constructeurs de requêtes graphiques

Ce genre d'approches utilisent des interfaces graphiques pour construire des requêtes formelles, comme [Russell et Smart, 2008; Ambrus et al., 2010; Clemmer et Davies, 2011] pour les requêtes SPARQL , [Athanasis et al., 2004] pour les requêtes RQL, ou [Team, 2007] pour les graphes conceptuels.

Dans [Russell et Smart, 2008], les utilisateurs peuvent exprimer leurs requêtes en utilisant un langage de requête visuel, appelé vSPARQL, qui fournit un formalisme graphique pour spécifier des requêtes SPARQL. Ce type d'interface permet de configurer, avec des assistants ou librement, des formulaires et sous-formulaires de saisie, de trier les résultats, limiter leur nombre, les regrouper selon divers critères, ou utiliser différentes fonctions d'aggrégation (SUM, COUNT, AVG).

L'autre forme consiste à construire des requêtes en se basant sur le remplissage des formulaires de ce que l'utilisateur veut que la requête réalise. Ces constructeurs de requêtes sont similaires à un assistant ou ont une interface qui commence à construire la requête comme un arbre en commençant à partir du type de la requête en tant que racine, puis restreindre les termes et ressources possibles à choisir à partir de chaque choix effectué sur les branches [Ambrus and al., 2010].

Si ces outils rendent le travail de formulation de requêtes formelles moins fastidieux, mais, ils nécessitent toujours de connaître le langage de requête utilisé, et il n'est pas envisageable de demander à l'utilisateur quelconque de saisir des requêtes dans un langage qu'il ne maîtrise pas.

3.3.4 Construction de requêtes formelles à partir de requêtes mots-clés ou en langage naturel

À l'autre extrémité du continuum de formalisation, se trouvent les approches qui visent à générer ou à construire automatiquement ou semi-automatiquement des requêtes formelles à partir de requêtes exprimées sous forme de mots-clés ou de phrases en langage naturel (LN). Les utilisateurs expriment leurs besoins en information de façon intuitive, sans avoir à connaître le langage de requêtes ou bien le formalisme de représentation de connaissances utilisé dans ces systèmes. Certains travaux ont proposé de traduire des requêtes exprimées en mots-clés ou en phrases en langage naturel en requêtes formelles dans un langage comme SeREQL [Lei et al., 2006] ou SPARQL [Zhou et al., 2007; Tran et al., 2009; Cabrio et al., 2012].

Dans le contexte du Web sémantique il existe plusieurs travaux sur la traduction des requêtes mots-clés en requêtes sémantiques, ce qui est très important pour la recherche sémantique. Face à l'écart entre la requête sémantique fondée sur la logique formelle et les utilisateurs finaux, certaines communautés ont proposé différentes solutions pour le surmonter : Dans [Royo et al., 2005], les auteurs proposent un mapping des mots-clés à leurs correspondant synsets de WordNet. Dans [Bernstein et al., 2005], ils explorent les langages contrôlés fourni et les interfaces guidées par le langage naturel [Stojanovic et al., 2003]. Du point de vue de l'affinement des requêtes [Stojanovic et al., 2004 ; Carlos et al., 2006], 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 [Athanasis et al., 2004] contribue également dans cette voie, par la construction des graphes requêtes par le biais de la navigation et de sélection sur l'ontologie. SemSearch [Lei et al., 2006], dans ce système, l'entrée est appariée à des entités sémantiques au moyen d'index, puis un ensemble de modèles prédéfinis sont utilisés pour interpréter les requêtes dans le language SeRQL. Quick [Zenz et al., 2009] utilise également les modèles de requêtes prédéfinis pour construire un ensemble de possibles requêtes formelles sémantiques dans un domaine donné. Le processus de construction de requêtes formelles est identique à celui de Semsearch. Avatar Semantic Search [Kandogan et al., 2006] 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 [Li et al., 2007] est un prototype de moteur de recherche à base de relation. OntoLook évoqué la faiblesse des requêtes mots clés dans le contexte du web sémantique et infère de possibles relations entre les mots-clés pour améliorer la précision de la recherche. Dans [Tran et al., 2007], 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 [Bobed & Mena, 2016], Ils présentent un système qui effectue l'interprétation sémantique des mots clés sur différents sources de données, découvrent la signification des mots-clés entrés en consultant un ensemble générique d'ontologies. Autres systèmes pertinents dans le domaine de la recherche sémantique sont [Tran et al., 2011; Wang et al., 2008; Zhou et al., 2007]. Ces systèmes trouvent tous les chemins qui peuvent être dérivés à partir d'un graphe, jusqu'à une profondeur prédéfinie, pour générer les requêtes. Dans le même contexte, d'autres systèmes plus pertinents [Tran et al., 2009; Fu et al., 2011; Teng & Zhu, 2015; Le et al., 2014; Latreche et al., 2009] ont été réalisés, où les auteurs présentent des approches de recherche par mots-clés sur les données structurées en graphe, en mettant l'accent en particulier sur le modèle de données RDF. Ces techniques ont été améliorées et étendues de plusieurs manières différentes.

Dans le même contexte d'autres travaux ont étés réalisés :

Autosparql [Lehmann and Bühmann, 2011; Unger et al., 2012] commençant par une interprétation sommaire de la requête utilisateur exprimée en LN, le système échange avec l'utilisateur, lui demandant des exemples positifs et négatifs de réponses, afin de raffiner l'interprétation initiale qui avait été faite de la requête.

e-learning, CHESt [Linckels and Meinel, 2005]. Dans une version, les requêtes sont exprimées sous forme de mots-clés ; dans l'autre, sont exprimées en langue naturelle.

Querix [Kaufmann et al., 2006] est une interface indépendante du domaine, dans laquelle les requêtes peuvent être exprimées sous la forme de questions en langue naturelle commençant par l'un des débuts de question autorisés (Which, What, How many, How much, Give me, ou Does).

NLP-Reduce [Kaufmann et al., 2007] est une interface indépendante du domaine, et dans laquelle les requêtes peuvent être exprimées sous forme de phrases complètes en langue naturelle, de fragments de phrases ou de mots-clés.

FREYA [Damljanovic et al., 2010] utilise les techniques du TAL pour reformuler une requête de langage naturel dans une requête SPARQL.

La plupart des systèmes discutés ci-dessus et notamment les systèmes de recherche par mots clés sur les données structurée en graphes, partagé une architecture commune telle que représentée dans la figure 3.1. Ils se composent de 2 modules principaux: module de prétraitement et module de construction de requêtes formelles. Le module de prétraitement consiste à accélérer la performance lors de la génération de requêtes formelles. Ce module implique l'indexation des entités de base de connaissances. Le module de construction de requêtes formelles est l'objet de la recherche sémantique par mot-clé. Normalement, il se compose des composants suivants :

(i) Mise en correspondance ou appariement des mots de la requête aux éléments de la base de connaissances ;

(ii) génération des requêtes formelles liant les éléments détectées précédemment;

(iii) classement des requêtes formelles construites et sélection par l'utilisateur de la bonne requête, qui satisfait son besoin en information.

Le composant appariement des mots de la requête aux éléments sémantiques de la base de connaissances mappe les mots clés aux entités de base de connaissances indexées. Le composant générant des requêtes formelles construira alors des requêtes formelles à partir de l'ensemble des entités détectées à l'étape précédente. Les techniques de construction de requêtes formelles sont différentes dans chaque système. Certains utilisent la technique de graphe requêtes, tandis que d'autres utilisent la technique de modèle. Ce composant produira toutes les requêtes formelles possibles en interprétant les significations saisies par la sémantique (par exemple, une classe, des données et des propriétés d'objet, un littéral). Le composant de classement des requêtes formelles classera et sélectionnera la requête la plus pertinente (ou l'ensemble des requêtes pertinentes) qui correspond aux mots-clés entrés par l'utilisateur. Dans certains systèmes, l'ensemble des requêtes classées est directement transmis aux utilisateurs. Cependant, dans d'autres systèmes, les requêtes formelles seront exécutées et les résultats (réponses) sont transmis à l'utilisateur.

Les approches se sont pour l'instant focalisées sur des problèmes précis : optimiser l'étape d'appariement en exploitant des ressources externes comme Wordnet ou Wikipedia (Lei et al., 2006; Tran et al., 2009; Wang et al., 2008; Latreche et al., 2009), optimiser l'indexation et l'exploration de la base de connaissances pour la construction de la requête graphe (Zhou et al., 2007; Tran et al., 2009; Fu et al., 2011; Le et al., 2015), améliorer le classement des requêtes candidates (Zhou et al., 2007; Tran et al., 2009; Wang et al., 2008), exploiter le contexte de recherche de l'utilisateur pour focaliser l'ensemble du processus de recherche (Fu et al., 2011).

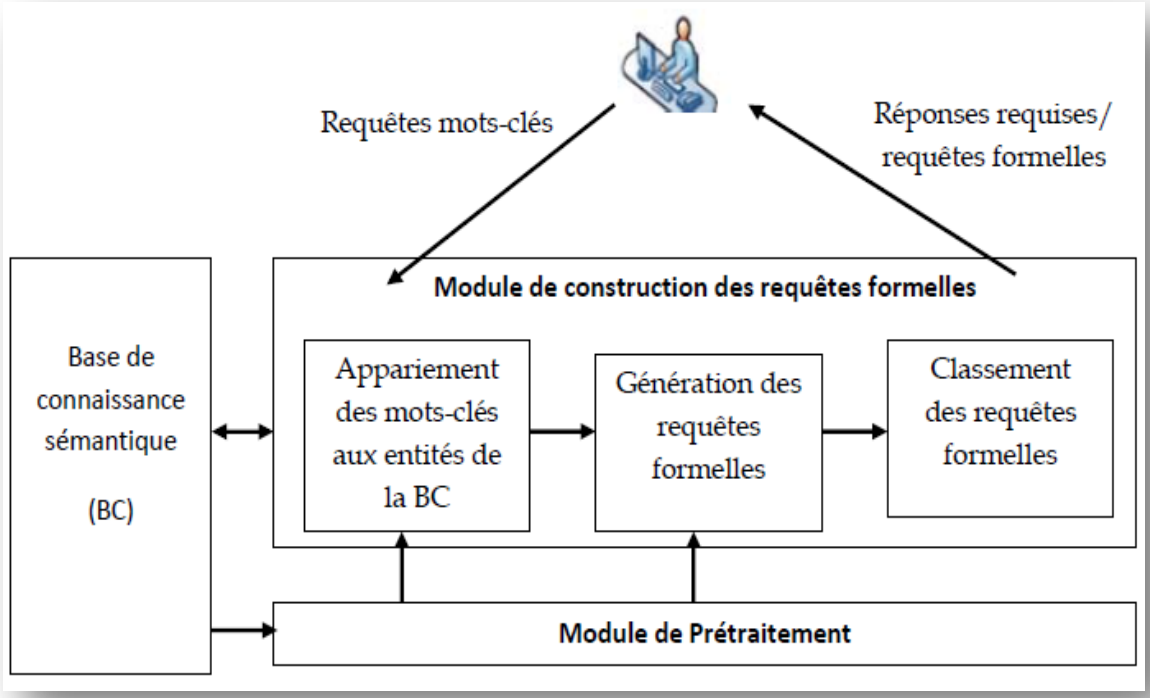


Figure 3.1. L'architecture commune de la recherche sémantique par mot clé.

3.3.5 Recherche par mots clés sur les bases de données relationnelles

Il faut mentionner, aussi qu'un grand nombre de travaux soutenant la recherche par mots clés sur les bases de données relationnelles ont été réalisés. Ces approches permettent de traduire un ensemble de mots clés en requêtes SQL. BANKS [Aditya, et al. 2002 ; Bhalotia et al. 2002], DBXplorer [Agrawal et al., 2002] et DISCOVER [Hristidis et al., 2002], Keymantic [Bergamaschi et al., 2011], [He et al., 2007; Kandogan et al., 2006 ; Liu et al. , 2006]. Ils se concentrent sur le comment effectuer efficacement des recherches par mots-clés sur les données relationnelles, en surmontant le problème de la répartition des données pertinentes entre plusieurs tables. De plus, comme souligné dans [Bergamaschi et al., 2011], la plupart de ces travaux ne reposent que sur les connaissances extensives obtenues en appliquant des techniques de recherche (RI), et ne considèrent pas la connaissance intensive (la connaissance structurelle) ou la sémantique des mots-clés entrées. Dans ce qui suit nous discuterons quelques uns des ces approches.

L'approche basée sur les graphes adoptée par BANKS [Aditya, et al. 2002 ; Bhalotia et al. 2002] a fortement influencé d'autres travaux tels que SPARK ou Q2Semantic. Dans ce système, la base de données est modélisée sous la forme d'un graphe orienté, chaque tuple étant un nœud dans ce graphe. Les liens clés-primaire-clé-étrangère sont des arêtes entre les différents nœuds. Avec ce graphe construit en mémoire, BANKS utilise les indexes résidents sur disque pour rechercher les nœuds qui correspondent exactement aux mots clés. Une fois qu'il les a obtenus, il crée des arborescences qui relient tous les nœuds appariés. Ces arbres sont enracinés dans un nœud d'information, qui peut être restreint pour former l'ensemble sélectionné de nœuds du graphe.

Considérant également la base de données comme un graphe de tuples interconnectés, DBXplorer [Agrawal et al., 2002] et DISCOVER [Hristidis et al., 2002] fonctionnent avec des structures de données spécialement conçues stockées dans la même base de données. Le premier, construit une table de symboles qui indexe la base de données en associant les mots clés à leurs emplacements. Ceci permet de déterminer où les mots-clés de la requête apparaissent de manière efficace (c'est-à-dire les tableaux, les colonnes ou les lignes, en fonction du niveau de granularité choisi). Ensuite, il travail tous les sous-ensembles de tables qui, lorsqu'ils sont joints, peuvent contenir des lignes avec tous les mots-clés. Enfin, pour chacune de ces combinaisons, ils créent une instruction SQL pour récupérer les lignes avec tous les mots clés. DISCOVER repose également sur un index mots clés (Master Index) à granularité de ligne pour effectuer la recherche par mots clés initiale sur la base de données. Ensuite, il travail au niveau des lignes pour créer des réseaux candidats, des ensembles de tuples qui couvrent les mots clés entrées.

Dans, Keymantic [Bergamaschi et al., 2011], les auteurs se concentrent sur le mappage des mots-clés aux entités du schéma relationnel d'une base de données et les interprètent comme une requête SQL pour permettre une recherche par mots clés sur des bases de données sans avoir à traiter les données extensives, ce qui constitue une amélioration importante lorsque nous avons seulement accès au schéma de la base de données.

3.4 Modèles de données

Pour les données structurées, et en particulier pour les données RDF, le modèle de données est tout simplement un graphe. Par conséquent, toutes les approches de la recherche par mot-clé emploient un modèle de données graphique, non orienté ou orienté pour représenter les données sous-jacentes et leurs interrelations. Il est également possible que des poids puissent être associés à chaque élément (c'est-à-dire nœud ou arêtes) du graphe, pour refléter son importance par rapport aux autres éléments du graphe. Cependant, la différenciation clé de ces approches est de savoir si elles tiennent compte du schéma des données sous-jacentes ou non. Dans le cas du modèle RDF, c'est le schéma RDF (RDFS) qui est utilisé.

Les approches axées sur le schéma modèlent les données sous-jacentes en tant que graphe en fonction de leur schéma. Dans le modèle RDF, cette approche mappe les classes RDF aux nœuds, et les propriétés qui doivent apparaître entre les instances de deux classes sont mappées aux arêtes. Les arêtes sont étiquetées avec le nom de la propriété [Lei, et al., 2006 ; Tran, et al., 2007 ; Tran et al., 2009]. Ces approches impliquent fréquemment deux phases pour la génération des réponses à partir d'une requête mots-clés. Tout d'abord, ils prennent comme entrée le graphe schéma et les éléments correspondant aux mots-clés de la requête, puis tentent de dériver le schéma de chaque réponse possible. Deuxièmement, compte tenu de ces schémas, les requêtes appropriées sont construites pour récupérer les entités.

3.5 Structure des réponses

Actuellement, posant une requête mots-clés sur un moteur de recherche Web, le résultat est un ensemble classé de pages Web qui contiennent ces mots-clés. En comparant

cette approche avec les approches adoptées pour traiter les requêtes mots clés sur des données structurées et semi-structurées, le résultat est généralement un élément structuré à refléter, à part les informations simples contenues dans les mots-clés, la manière dont cette information est connectée, définissant ainsi, une interprétation possible de la réponse. Par conséquent, un tel élément pourrait être un arbre, un graphe ou simplement un nœud / entité de ces structures. D'autres approches, en particulier dans le modèle RDF, s'arrêtent au stade du calcul de la requête permettant à l'utilisateur de sélectionner la requête en fonction de ses besoins d'information [Tran et al., 2007 ; Tran at al., 2009]. Ils calculent les requêtes correspondant à toutes les interprétations possibles de la requête mots-clés de l'utilisateur, puis les présentent à l'utilisateur dans une structure graphique. Enfin, il existe des travaux visant à fournir l'interprétation d'une réponse structurée en langage naturel [Koutrika et al., 2010].

3.6 Algorithmes d'exploration

De nombreuses approches de recherche par mot-clé qui génèrent des réponses structurées en arbre ou en graphe répondent au problème de trouver des sous-structures reliant les éléments contenant les mots-clés de la requête. Les approches qui s'appuient sur les données structurées en arbre et en graphe sont principalement : *Backward Expansion* [Hulgeri et al., 2001, Bhalotia et al., 2002], *Bidirectional Search* [Kacholia et al., 2005] et leurs extensions [He et al., 2007, Tran et al., 2009]. Dans ce qui suit, nous présentons les algorithmes : *backward expansion* et *bidirectional search*.

a) **Backward Expansion :** L'algorithme [Hulgeri et al., 2001, Bhalotia et al., 2002], en premier lieu, identifie les nœuds du graphe contenant les mots-clés de la requête, appelés *keyword nodes,* puis effectue une traversée itérative le long des arêtes entrantes des nœuds visités, Jusqu'à ce que le nœud appelé *answer root* soit trouvé, avec la propriété spéciale d'être un nœud connecté à tous les nœuds mots-clés. Cette traversée itérative est réalisée par l'algortithme Dijkstra du plus court chemin pour chaque nœud mot-clé. Si un mot-clé est contenu dans plus d'un nœud, ces nœuds forment un cluster pour lequel une seule copie de l'algorithme Dijkstra est instanciée. Ce cluster est appelé *Initial Keyword Cluster*, et est désigné comme $IKCi$, en raison de la contenance de nœuds correspondant au i^{ème} mot-clé.

   A chaque itération de l'algorithme, il existe deux stratégies pour choisir la direction (c'est-à-dire le nœud) à suivre (c'est-à-dire à visiter) par la suite. Avant de procéder à leur présentation, certaines terminologies sont introduites. La requête mots-clés est un ensemble de mots clés, $K = \{k1, k2, \dots, kn\}$, *Keyword Cluster* KCi est l'ensemble des nœuds qui ont été visités et peuvent atteindre le mot clé ki, c'est-à-dire $IKCi \cap KCi = IKCi$. Initialement, chacun de ces clusters contient tous les nœuds mots-clés, pour chaque ki, c'est-à-dire $KCi = IKCi$.. Les stratégies d'expansion sont les suivantes : *Equidistance expansion in each keyword cluster et Distance-balanced expansion across keyword clusters*

b) **Bidirectional Expansion :** Les mêmes auteurs ont proposé l'algorithme «*Bidirectional Expansion*» [Kacholia et al., 2005] pour surmonter la mauvaise performance que reflète l'algorithme «*Backward Expansion* » dans certains types de graphes. Cet algorithme a la possibilité d'explorer le graphe en parcourant aussi les arêtes en avant. La raison est d'identifier un nœud racine beaucoup plus rapidement. Pour contrôler l'ordre

d'expansion, l'algorithme donne la priorité aux nœuds par des facteurs d'activation heuristique, ce qui permet d'estimer de manière intuitive les nœuds susceptibles de répondre aux racines. Bien que cette stratégie se déroule bien dans plusieurs scénarios, il est difficile de fournir une garantie de performance pessimiste. La raison en est que les facteurs d'activation sont des mesures heuristiques dérivées de la topologie graphique générale et des parties du graphe déjà visitées; Ils peuvent ne pas refléter avec précision la probabilité d'atteindre les nœuds mots clés à travers une région inexplorée du graphe à une distance raisonnable.

3.7 Classement / Calcul du score des réponses

La recherche par mots-clés est intrinsèquement ambiguë et tous les résultats de la requête ne sont pas tous pertinents pour un utilisateur. Différents schémas de classement ont été proposés pour classer les résultats que les utilisateurs puissent se focaliser sur les meilleurs. Différents schémas de classement utilisés dans les travaux existants, qui considèrent à la fois les propriétés des nœuds de données (ex., TF/IDF et les mesures complexes adoptées par RI, poids des nœud / arêtes, classement dans le style de Page Rank) et les propriétés de l'ensemble du résultat de la requête (ex., longueur du chemin, nombre de nœuds / arêtes, poids des nœuds / arêtes, normalisation des tailles) [Konstantin et al., 2008 ; He et al., 2007 ; Li et al., 2007, Liu et al., 2006 ; Luo et al., 2007 ; Hulgeri et al, 2001 ; Bhalotia et al., 2002 ; Tran et al., 2009 ; Wang et al., 2009].

En particulier, dans [Hulgeri et al., 2001, Bhalotia et al., 2002], les réponses sont classées à l'aide de la notion de proximité couplée avec la notion de prestige des nœuds basée sur des liens entrants, semblables à des techniques développées pour la recherche sur le Web. En outre, le schéma de classement général comprend la pondération des arêtes et la combinaison additive ou multiplicative des scores des arêtes et des nœuds. Représentant le score global des nœuds et des arêtes avec $Nscore$ et $Escore$ respectivement, alors le score global d'une réponse est donné comme suit:

$$Additive : (1-\gamma) * E_{score} + \gamma * N_{score}$$
$$Multiplicative : E_{score} * N_{score}^{\gamma} \quad (3.1)$$

Considérant les réponses en arbres, [Hulgeri et al., 2001, Bhalotia et al., 2002] calculent le score global du nœud en tant que moyenne des scores des nœuds racine et feuille, c'est-à-dire en omettant les nœuds intermédiaires. Le score d'un nœud u est déterminé par le nombre d'arcs entrants Pr (u), ce qui traduit ainsi la notion de prestige. Ce score est normalisé en utilisant le score de prestige maximal des nœuds du graphe sous-jacent:

$$Score(u) = \frac{\Pr(u)}{\max_{u \in V(G)}\{\Pr(u)\}} \quad (3.2)$$

Des travaux tels que [Hristidis et al., 2003 ; Luo et al., 2007, Liu et al., 2006 ; Li et al., 2008 ; Tran et al., 2009] utilisent un certain nombre de mesures IR pour classer les réponses. Dans [Hristidis et al., 2003 ; Luo et al., 2007], ils exploitent les techniques IR dans les RDBM, tandis que dans [Liu et al., 2006] ils les étendent en proposant des mesures de notation nouvelles et plus fines (normalisation des arbres, normalisation de la longueur du document, normalisation de la fréquence des documents, normalisation des fréquences inter-document).

Dans [Li et al., 2008], ils proposent un nouvel index qui matérialise les classements basés sur TF/IDF. En revanche, le travail [Tran et al., 2009] emploie Lucene Index pour le classement textuel.

Outre le classement des réponses basées sur la similarité textuelle, d'autres facteurs sont également pris en compte, tels que la compacité structurelle, le facteur de complétude et les mesures de popularité (c'est-à-dire des variantes simplifiées du classement Page Rank). Tous ces facteurs peuvent être utilisés indépendamment ou combinés d'une manière additive ou multiplicative comme mentionné précédemment. La compacité structurelle des réponses est déterminée par la somme des longueurs de chemin présentes dans la réponse comme dans [Tran et al., 2009] ou une combinaison plus sophistiquée des longueurs de chemin entre deux nœuds n_i; n_j de la réponse comme dans [Li et al., 2008]:

$$Score(n_i, n_j) = \sum_{P_{ij}} \frac{1}{(|P_{ij}|+1)^2} \qquad 3.3$$

Où P_{ij} est un chemin entre les nœuds n_i et n_j, et $|P_{ij}|$ est la longueur d'un tel chemin.

Les auteurs de [Luo et al., 2007], pour quantifier le facteur du nombre de mots-clés de requêtes correspondants, ils utilisent le facteur de complétude, qui a été reconnu comme significatif par les chercheurs en RI face à de requêtes courtes.

La plupart des approches utilisent des fonctions monotones, dans lesquelles le score d'une réponse reflète une fonction d'agrégation sur les scores de ses composantes. Des approches telles que [Luo et al., 2007 ; Liu et al., 2006] utilisent des fonctions non monotoniques, dans lesquelles le score d'une réponse est indépendant de ses composants. Surtout pour les approches de calcul des top-k requêtes, les fonctions non monotoniques doivent être associées à une fonction de délimitation supérieure monotone respective pour déterminer efficacement les top-k réponses.

Ontologer [Stojanovic et al., 2003] ont construit un mécanisme de requête en enregistrant les comportements de l'utilisateur en ontologie et le rappelé pour le classement. SemRank [Anyanwu et al., 2005] utilise la théorie de gain d'informations pour classer les chemins découverts entre deux ressources. [Zhou et al., 2007] ils ont incorporé deux modèles probabiliste appelé *KQM (keyword query model*) et *KBM (knowledge base model)* pour le classement des requêtes a fin de choisir les requêtes les plus pertinentes.[Wang et al., 2008] ont propose trois métriques qui sont : *path length, popularité score, keyword matching score*, pour calculer le score les requêtes formelles les plus pertinentes.

3.8 Méthodologies d'évaluation

3.8.1 Jeux de données de test

Au début, il n'existait aucune méthode formelle et objective permettant d'évaluer les interfaces entre langue naturelle ou mots-clés et bases de connaissances. Les concepteurs de ce type d'interface menaient leurs propres évaluations. Puis, le domaine de recherche gagnant en popularité et le nombre d'approches proposées augmentant régulièrement, il devenait urgent de mettre en place un cadre d'évaluation commun. Un certain nombre d'initiatives ont été lancées dans ce contexte.

Avec le développement des fonctions de classement viens la nécessité d'évaluer la qualité des résultats. Pour cela, un certain nombre d'ensemble de jeu de test ont été employés. Les plus largement utilisées dans le contexte de recherche par mots-clés sur les données structurés sont la base de données DBLP[28] des publications scientifiques pour les utilisées dans le contexte des bases de données, XML[29], RDF[30]. Les requêtes sont généralement formulées par les auteurs eux-mêmes et ne sont pas réutilisés entre les expériences et les expérimentateurs, ce qui rend la comparaison des résultats, extrêmement difficile, à moins que le chercheur ne fournisse cette comparaison directement en réimplantant ou réutilisant les approches existantes comme lignes de base, qui est une pratique importante qui est suivie par les chercheurs lorsqu'il n'existe pas de compagne d'évaluation. En outre, les ensembles de requêtes sont souvent assez petits, rarement plus de 20 par jeu de données. Comparer aux ensembles de requêtes utilisés dans les collections TREC. Dans le même style suit l'évaluation de la pertinence des résultats, qui est en général interprété par les auteurs eux-mêmes ou par leurs collègues.

Un certain nombre d'initiatives ont été lancées dans ce contexte, comme, La campagne d'évaluation *INEX Linked Data Track*[31] a vu le jour en 2012, dont le but est de retourner une liste d'entités correspondant à un besoin en information exprimé de différentes manières (en langue naturelle, sous forme de mots-clés et sous forme de requêtes semi-structurées inspirées de SPARQL) et Le *Semantic Search Challenge*[32] évalue la pertinence des entités retournées par un système à partir de requêtes réellement exprimées par des utilisateurs sous forme de mots-clés.

3.8.2 Méthodologie d'évaluation de la recherche sémantique par mots clés

Dans, cette section nous discuterons de la méthodologie d'évaluation pour les approches de recherche sémantiques par mot-clé. Les évaluations concernent 3 caractéristiques principales: l'effectivité (effectiveness), l'efficacité (efficiency) et l'utilisabilité (usability) des systèmes.

3.8.2.1 L'effectivité

La figure 3.2 illustre la méthode d'évaluation du classement. Pour l'évaluation, un ensemble de mots-clés (c'est-à-dire les entrées de système) est fourni. Les mots-clés peuvent être extraits de scénarios problématiques ou de questions fournies par des experts / participants dans le domaine. Les requêtes sémantiques manuelles correspondant aux questions sont créées en standard. Les requêtes sémantiques construites des systèmes sont évaluées avec les requêtes manuelles standards. Si la requête sémantique construite est équivalente sémantiquement à la requête standard, elle est «acceptable».

La métrique largement utilisée pour évaluer la capacité de classement est le « Mean Reciprocal Rank » (MRR). Le Reciprocal Rank (RR) est l'inverse multiplicatif du rang de la

[28] http://dblp.uni-trier.de/db/
[29] http://dblp.uni-trier.de/xml/
[30] http://dblp.l3s.de/dblp++.php
[31] https://inex.mmci.uni-saarland.de/tracks/lod/
[32] http://semsearch.yahoo.com/

première requête sémantique acceptable. Par conséquent, le MRR est la moyenne des RR pour toutes les requêtes sémantiques dans le jeu de test.

$$MRR = \frac{1}{N_{KP}} \sum_{i=1}^{N_{KP}} \frac{1}{rank_i}$$

Où N_{KP}, est le nombre de requêtes sémantiques dans l'ensemble testé, $rank_i$ = ordre de la requête sémantique acceptable.

De même, d'autres approches introduisent une autre nouvelle métrique nommée «Target Query Position » (TQP). Il a également utilisé pour évaluer l'efficacité du classement des requêtes sémantiques. A savoir, $TQP = 11 - P_{target}$, où P_{target} désigne la position de la requête acceptable dans la liste classée. Si le rang d'une requête est supérieur à dix, son TQP est tout simplement 0. Ainsi, le score TQP d'une requête varie de 0 à 10.

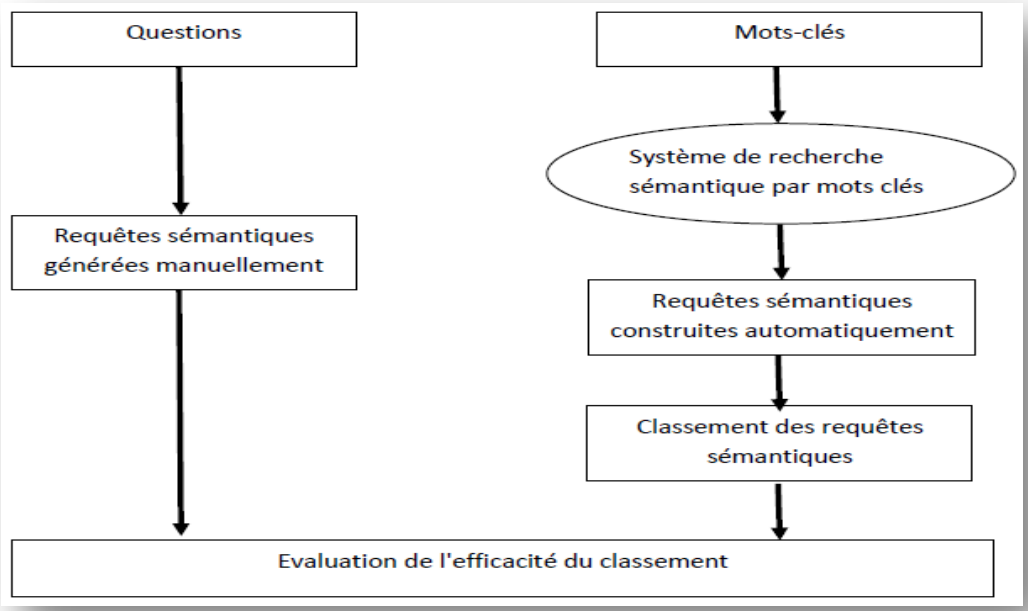


Figure 3.2. Méthode d'évaluation du classement.

3.8.2.2 Efficacité

L'évaluation de l'efficacité vise à mieux comprendre les performances des systèmes. Par exemple, Q2semantic a étudié l'impact de la taille de l'espace de recherche (graphe RDF) sur l'index du graphe cluster. Tran at al. [Tran at al., 2009] pour évaluer l'efficacité ont étudiés l'impact du paramètre k sur la performance de la recherche. On constate que le temps de recherche augmente linéairement lorsque k devient plus grand. L'évaluation de SKengine était concernée par l'impact sur son temps de calcul de deux paramètres: (i) le nombre de mots-clés et (ii) le nombre de relations entre concepts dans l'ontologie. Le nombre de mots-clés est inversement lié à la performance du temps de SKengine. En outre, le nombre plus élevé de relations dans l'ontologie provoque des performances plus faibles. En général, l'évaluation de l'efficacité est mesurée par le temps de performance moyen.

3.8.2.3 Utilisabilité

L'évaluation de l'utilisabilité étudie comment les systèmes de recherche sémantiques par mots clés sont utiles du point de vue des utilisateurs. Fondamentalement, l'interface utilisateur du système est au centre de l'étude. La présentation de requêtes sémantiques de manière simple aux utilisateurs est nécessaire et c'est un défi a relevé. Il y a eu une étude sur l'utilisabilité de la recherche sémantique utilisant une interface de langage naturel. Kaufmann et Bernstein [Kuafmann et Bernstein, 2010] présentent quatre interfaces permettant chacune un langage de requête différent et présentent une étude sur l'utilisabilité en comparant ces interfaces. Les méthodes qualitatives telles que les entrevues et les questionnaires ont été utilisées pour saisir la préférence des utilisateurs pour chaque interface. Il a été constaté que chaque interface avait différents points faibles / forts dans les opinions des utilisateurs.

3.9 Synthèse

Nous avons présenté un certain nombre de systèmes de recherche d'informations sémantiques existants, et nous les avons classées en plusieurs catégories en fonction de leurs fonctionnalités. Malheureusement, la plupart de ces moteurs de recherche ne sont pas disponibles pour le test et l'évaluation. Dans d'autre part en raison de leurs divergences avec les moteurs de recherche traditionnels, il n'est pas possible de les comparés à l'aide d'un unique cadre d'évaluation.

1) ***Moteurs de recherche d'ontologie :*** Afin d'évaluer la performance de ce type de moteurs de recherche, il n'existe pas de collection de test, mais nous pouvons tout simplement les tester par la recherche d'ontologies en utilisant le nom d'ontologies, les classes et les propriétés et de juger leurs résultats selon la mesure de précision.
2) ***Moteurs de recherche sémantique basée sur le contexte*** : La principale étrangeté de ces moteurs est leur simplicité. En fait, car ils ont essayé d'être aussi simple que les moteurs de recherche traditionnels (comme Google), ils sont les plus populaires moteurs de recherche dans le web sémantique. Ici la qualité des résultats dépend fortement sur la puissance de son module d'annotation. Le plus grand problème de ces moteurs de recherche est qu'ils sont limités à des contextes particuliers. Il est beaucoup mieux si nous pouvons développer des moteurs de recherche sémantique multi-contexte. Heureusement, nous pouvons appliquer des mesures standard (c'est-à-dire de précision et de rappel) pour évaluer ce type de moteurs de recherche sémantique.
3) ***Moteurs de découverte d'association sémantiques*** : Comparé à d'autres catégories, les moteurs de découverte d'association sémantiques sont liés à couches la plus élevées de la pyramide du Web sémantique (logique et preuve). Les résultats de ces moteurs sont très dépendants de leur ontologie référentielle.
4) ***Construction de requêtes formelles :***
    a) ***Constructeurs de requêtes par auto-complétion*** : Ces constructeurs sont destinés aux utilisateurs qui connaissent l'écriture des requêtes formelles (tel que, SPARQL). Ils ne créent pas la requête à la place des utilisateurs, mais, ils les aideront à l'écrire.

b) ***Constructeurs de requêtes graphiques*** : Ce type d'approches reposent sur des interfaces graphiques pour construire des requêtes formelles. Si ces outils rendent le travail de formulation moins fastidieux, ils nécessitent toujours de connaître le langage de requête utilisé, et il n'est pas envisageable de demander à l'utilisateur quelconque de saisir des requêtes dans un langage qu'il ne maîtrise pas.

c) ***Construction de requêtes formelles à partir de requêtes mots-clés*** : Dans ce type d'approches, notamment, les approches de recherche par mots-clés sur les données structurées en graphe RDF, supposent que l'utilisateur exprime son besoin en information de façon intuitive, sans avoir à connaître le langage de requêtes ou bien le formalisme de représentation de connaissances utilisé dans le système, c.-à.d., il n'a aucunes connaissances préalable sur le vocabulaire, la syntaxe et la structure (schéma) de la base interrogée, ceci va engendrer un certain nombre de problèmes :

- l'utilisateur trouve beaucoup de difficultés de formuler correctement sa requête mots-clés pour quelle puisse répondre à son besoin en informations ;
- vu que certaines requêtes sont plus ambigus que d'autres, la phase d'appariement des mots-clés aux entités de la base de connaissances peut ne pas générer d'entités, comme elle peut générer pour chaque mot-clé un nombre important d'entités (utilisation des techniques de RI, telles l'appariement syntaxique imprécis), ce qui implique un nombre important de requêtes formelles construites à cause du nombre important de toutes les combinaisons possibles des entités correspondant aux mots-clés ;
- Le calcul de toutes les combinaisons possibles des entités correspondant aux mots-clés générés durant la première étape est un problème qui n'est pas abordé dans les travaux de recherche par mots clés en général en raison de sa complexité en temps très élevé. La plupart des travaux ignorent simplement ce problème. D'autres, optent pour le prendre en compte dans la conception des algorithmes de recherche par mots clés, mais l'ignorent dans l'implémentation respective. Nous croyons que cette limitation est très restrictive ;
- La plupart de ces approches produisent des interprétations de requêtes les plus populaires, mais qui peuvent ne pas être celles désirés par l'utilisateur.

Toutes ces problèmes vont contribuer à rendre moins performant ces systèmes.

Afin d'évaluer la performance de ce type de systèmes, il n'existe pas de collection de test, mais nous pouvons tout simplement appliquer des mesures standard (c'est-à-dire de précision, de rappel, le MRR, etc.) de la RI traditionnelles et de construire des collections de test spécifique pour évaluer ce type de systèmes.

d) ***Construction de requêtes formelles à partir de requêtes en LN*** : Plusieurs fois dans la littérature, la pertinence des interfaces en langue naturelle a été mise en cause. Par exemple, dans les approches soutenant les requêtes exprimées sous forme de mots-clés ou en langue naturelle, les utilisateurs déclarent en majorité (76%) préférer la

version mots-clés que la version langue naturelle [Linckels et Meinel, 2005]. Dans ce type d'interface, trois grands problèmes sont identifiés :

- l'*ambiguïté* de la langue naturelle, une même expression en langue naturelle peut être interprétée de différentes façons ou encore une même idée peut s'exprimer de différentes façons en langue naturelle.
- la *barrière adaptative* rend peu portables que les interfaces présentent de bonnes performances de recherche ; ces systèmes sont la plupart du temps spécifiques à un domaine particulier et, par conséquent, difficiles des les adaptées à de nouveaux contextes.
- le *problème d'habitabilité* selon lequel l'utilisateur final peut se sentir perdu face à une trop grande liberté dans l'expression de sa requête. La plupart des interfaces en langue naturelle n'expliquent pas comment exprimer ses requêtes ou quel type d'information peut être demandé.

3.10 Conclusion

Dans la majorité des approches mentionnées dans ce chapitre, notamment, les approches de recherche par mots-clés sur les données structurées en graphe RDF, les graphes requêtes finales sont obtenues en explorant la base de connaissances et en reliant ses entités qui ont été identifiées dans la requête utilisateur. Ces approches supposent que l'utilisateur exprime son besoin en information de façon intuitive, sans avoir à connaître le langage de requêtes ou bien le formalisme de représentation de connaissances utilisé dans le système, c.-à.d., il n'a aucunes connaissances préalables sur le vocabulaire, la syntaxe et la structure (schéma) de la base interrogée. Ceci va engendrer un certain nombre de problèmes : (1) Difficulté pour un l'utilisateur de formuler correctement sa requête mots-clés pour quelle puisse répondre à son besoin en informations ; (2) L'ambiguïté de la requête mots-clés de l'utilisateur, peut ne pas engendrer de graphes requêtes ou engendrer un nombre important de requêtes formelles ; (3) La plupart de ces approches produisent des interprétations de requêtes les plus populaires, mais qui peuvent ne pas être désirés par l'utilisateur. Tous ces problèmes vont contribuer à la diminution des performances de ces systèmes.

Alors, pour faire face à ces problèmes constatés, nous proposons dans le chapitre suivant une nouvelle approche qui se situe dans le contexte de la construction automatique de requêtes formelles à partir de requêtes mots-clés. L'approche que nous proposons veut fournir aux utilisateurs un moyen d'interrogation basée sur le contexte des bases d'annotations sémantiques, à l'aide de requêtes mots-clés, et ce dans le but d'éviter à ces utilisateurs de se confronter à la complexité de la formulation d'une requête formelle dans un langage du Web Sémantique (tel que SPARQL).

Partie II
Contributions

Chapitre 4. Construction basée sur le contexte de requêtes formelles à partir de requêtes mots-clés

4.1. Introduction

Dans la majorité des approches visant à formuler des requêtes formelles à partir de requêtes mots-clés et notamment, les approches de recherche par mots-clés sur les données structurées en graphe (RDF), supposent que l'utilisateur exprime son besoin en information de façon intuitive, sans avoir à connaître le langage de requêtes ou bien le formalisme de représentation de connaissances utilisé dans le système, c.-à.d., que l'utilisateur n'a aucunes connaissances préalables sur le vocabulaire, la syntaxe et la structure (schéma) de la base interrogée, ceci va engendrer un certain nombre de problèmes critiques :

(1) L'utilisateur trouve beaucoup de difficultés à construire correctement sa requête mots-clés pour quelle puisse répondre à son besoin en informations ;

(2) Vu que certaines requêtes sont plus ambigus que d'autres, la phase d'appariement des mots-clés aux entités de la base de connaissances peut ne pas générer d'entités, comme elle peut générer pour chaque mot-clé un nombre important d'entités (utilisation des techniques de RI, telles l'appariement syntaxique imprécis), ce qui implique un nombre important de requêtes formelles construites à cause du nombre important de toutes les combinaisons des entités correspondant aux mots-clés ;

(3) Le calcul de toutes les combinaisons possibles des entités correspondant aux mots-clés générés durant la première étape est un problème qui n'est pas abordé dans les travaux de recherche par mot-clé en général en raison de sa complexité en temps très élevé. La plupart des travaux ignorent simplement ce problème. D'autres, optent pour le prendre en compte dans la conception des algorithmes de recherche par mots clés, mais l'ignorent dans l'implémentation respective. Nous croyons que cette limitation est très restrictive ;

(4) La plupart de ces approches produisent des interprétations de requêtes (requêtes formelles) les plus populaires, mais qui peuvent ne pas être celles désirées par l'utilisateur.

Touts ces problèmes vont contribuer à diminuer les performances de ces systèmes. Alors, pour faire face à ces problèmes constatées, nous avons proposés une nouvelle approche [Latreche et al., 2017], qui se situe dans le contexte de la construction automatique de requêtes formelles à partir de requêtes mots-clés. L'approche que nous proposons veut fournir aux utilisateurs un moyen d'interrogation basée sur le contexte des bases

d'annotations sémantiques, à l'aide de requêtes mots-clés, et ce dans le but d'éviter à ces utilisateurs de se confronter à la complexité de la formulation d'une requête formelle dans un langage du Web Sémantique (tel que SPARQL).

Pour relever ce défi nous avons proposé les contributions suivantes:

(1) Proposition d'un nouveau processus de construction basé sur le contexte de requêtes formelles à partir des mots-clés, cela nécessite les étapes suivantes : (i) processus d'auto-complétion de requêtes, (ii) construction des graphes requêtes, (iii) classement des requêtes construites, (iv) sélection de la bonne requête par l'utilisateur, v) pondération des scores des éléments de la requête sélectionnées.

(2) Introduction et formalisation de la fonctionnalité d'auto-complétion de requêtes qui a pour tache d'aider l'utilisateur à formuler sans peine sa requête mots-clés (constituée uniquement de termes (éléments) de la base de connaissances), en lui présentant de possibles complétions de requêtes ou de termes les plus pertinentes en se basant sur l'historique de recherche.

(3) Proposition et implémentation d'un modèle de pondération des scores qui est utilisé pour capturer de façon concise les caractéristiques essentielles de l'historique des requêtes d'un utilisateur.

(4) Exploitation des informations sur les requêtes précédentes (historique de recherche) pour les utilisées en tant qu'informations contextuelles pour influencer la génération d'une nouvelle requête.

(5) Conception d'un algorithme efficace de construction des graphes requêtes qui étend l'algorithme existant.

(6) Présentation d'une évaluation complète de notre approche et la démonstration que l'approche proposée surpasse celles de l'état de l'art en qualité de performance.

Nous allons dans ce chapitre exposer en détaille notre proposition.

Ce chapitre est donc organisé comme suit : section 2 définit le problème de la construction des requêtes formelles, où nous présentons le modèle de données et de requêtes et l'architecture de notre système. Nous discuterons plus en détail des aspects spécifiques des données et du traitement des requêtes: prétraitement et indexation dans la section 3. Dans la section 4, nous détaillerons le processus d'auto-complétions de requêtes, traduction des mots-clés en requêtes formelles dans la section 5, la fonction de pondération dans la section 6. Nous donnons la conclusion dans la dernière section.

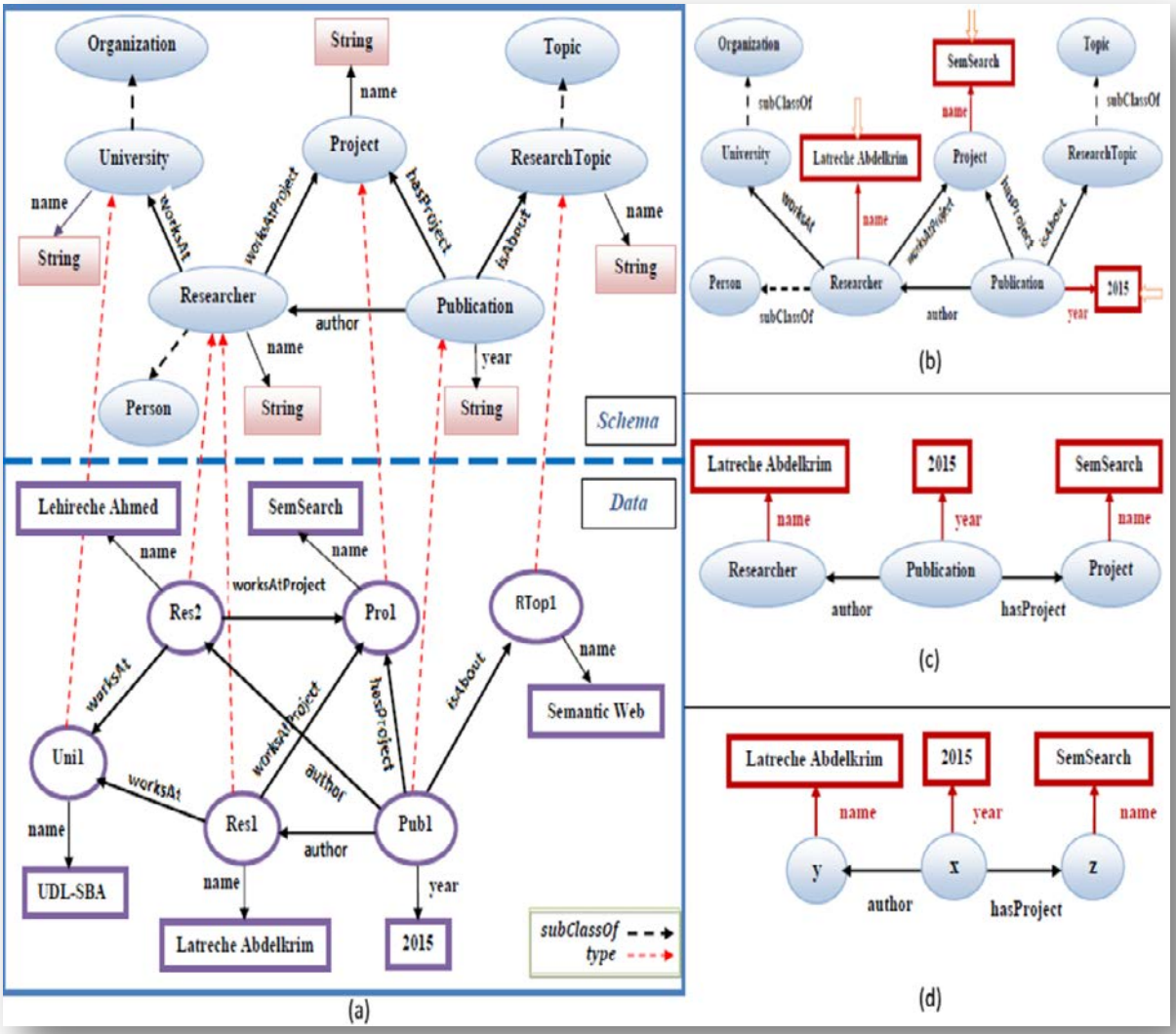


Figure 4.1. a) Exemple de graphes de données et schéma b) Espace d'interprétation c) Graphe requête d) Mapping du graphe requête

4.2. Définition du problème

Nous supposons que les utilisateurs assistés par un système d'auto-complétion de requête formulent leurs besoins en informations en utilisant un certain langage de requête utilisateur Q_u. Le moteur de recherche prend en charge des requêtes spécifiées dans un langage de requêtes système Q_S. Lorsque ces deux langages sont différents, une transformation est nécessaire pour que la requête utilisateur puisse être traitée par le moteur de recherche du système. Nous traitons avec de tels scénarios où le moteur de recherche prend en charge uniquement les requêtes conjonctives. Ainsi, nous pouvons réduire le problème de l'interprétation des requêtes mots-clés au problème de la construction des graphes requêtes qui seront traduites en requêtes formelles conjonctives. Pour clarifier le problème, nous donnons quelques définitions formelles.

4.2.1. Modèles de données

Un modèle de données RDF est une collection de triplets (sujet-propriété-objet) liant les ressources RDF. Nous pouvons aussi le considéré comme un graphe, où les sujets et les objets sont représentés par des sommets et chaque triplet est représenté par une arête orientée du sujet vers l'objet. On peut distinguer, trois types de graphes différents, chacun fournissant une vue différente sur les données sous-jacentes. Tous au long de ce document, nous donnerons les définitions formelles de ces graphes.

Définition 1 **:** Un graphe de données est un n-uplet $G^D = (V^D, L^D, E^D)$, où

1. $V^D = V_C \uplus V_E \uplus V_L$: un ensemble fini de sommets qui correspond à tous les sujets et objets dans un graphe RDF, où

- V_C : ensemble de toutes les classes.
- V_E : ensemble de toutes les entités (ressources).
- V_L : ensemble de valeurs littérales (valeurs de données).

2. L^D : un ensemble fini d'étiquettes (Labels) d'arêtes. Etant donné une arête $e \in L^D$, son étiquette est son correspondant prédicat (propriété). Avec $L^D = L_R \uplus L_A \uplus \{type\}$ où :

- L_R : ensemble de propriétés d'objet (relation).
- L_A : ensemble de propriétés de données (attribut).

3. E^D est un ensemble fini d'arêtes orientées de la forme $e(v_1, v_2)$ avec $v_1, v_2 \in V^D$ et $e \in L^D$ remplissant les conditions suivantes :

- $e \in L_R$ si et seulement si $v_1, v_2 \in V_E$;
- $e \in L_A$ si et seulement si $v_1 \in V_E$ et $v_2 \in V_L$;
- e = *type* si et seulement si $v_1 \in V_E$ et $v_2 \in V_C$.
  Ce type prédéfini capture l'appartenance d'une entité à une classe.

Définition 2 *:* Un graphe schéma est un n-uplet $G^S = (V^S, L^S, E^S)$, où

1. $V^S = V_C \uplus V_D$: un ensemble fini de sommets qui correspond à tous les sujets et objets, où :

- V_C : ensemble de toutes les classes.
- V_D : ensemble de tous les types de données

2. L^S : un ensemble fini de labels d'arêtes. Etant donné une arête $e \in L^S$, son label est son correspondant prédicat (propriété). $L^S = L_R \uplus L_A \uplus \{subclass\ of\}$ où :

- L_R : ensemble de propriétés de classe (relation).
- L_A : ensemble de propriétés de types de données (attribut).

3. E^S : est un ensemble fini d'arêtes orientées de la forme $e(v_1, v_2)$ avec $v_1, v_2 \in V^S$ et $e \in L^S$, remplissant les conditions suivantes :

- $e \in L_R$ si et seulement si $v_1, v_2 \in V_C$;
- $e \in L_A$ si et seulement si $v_1 \in V_C$ et $v_2 \in V_D$;
- e = *subclass of* si et seulement si $v_1, v_2 \in V_C$.
  Ce type prédéfini est utilisée pour définir la hiérarchie des classes.

La Fig. 4.1 (a) montre un exemple de graphe de données et graphe schéma.

4.2.2. Modèle de requêtes

Dans notre approche, nous distinguons entre la notion de requête utilisateur et requête système (ou formelle) :

Définition 3 *:* Une requête utilisateur est définie par : $Q_u = \{t_1, \dots, t_n\}$ où t_i : est un *terme* ou *élément-mot-clé* (éléments de la base de connaissance correspondant aux mots-clés). Les requêtes système (ou formelle) Q_S sont des requêtes conjonctives et ils ont une grande importance pratique, car de nombreuses requêtes SPARQL peuvent être écrites sous forme de requêtes conjonctives. Une requête conjonctive est définie comme suit :

Définition 4 *:* Une requête conjonctive est une expression de la forme : $(x_1, \dots, x_k).\exists\, x_{k+1}, \dots, x_m.A_1 \dots A_n$, où $x_1, \dots, x_m$ sont des variables et $A_1 \dots A_n$, sont des requêtes atomes. Ces atomes sont de la forme $P(v_1, v_2)$ où P est appelé prédicat, v_1, v_2 sont des variables ou constantes. Alors que les variables et les constantes correspondent aux identifiants des sommets, les prédicats correspondent aux labels des arêtes dans le graphe de données considéré.

Une requête conjonctive q peut être considérée comme un graphe motif construit à partir d'un ensemble de motifs triples $P(v_1, v_2)$ dans lesquels zéro ou plusieurs variables pourraient apparaître.

La figure 4.1 (d) montre un exemple de cette requête.

4.2.3. Aperçu de l'approche

La figure 4.2 illustre l'architecture conceptuelle de notre système. Dans cette architecture, nous pouvons faire une distinction entre les composants qui supporte le traitement en *offline* et *online*.

Traitement Offline : Durant le traitement offline, différentes informations sont extraites à partir du graphe de données et sont stockées dans des structures de données spécifiques sous formes d'indexes internes. Tout d'abord, les étiquettes (labels) des éléments du graphe de données sont extraites. Une analyse lexicale standard, y compris stemming, la suppression de mots vides et l'expansion des termes à l'aide de ressources lexicales (ex. WordNet) est effectuée sur ces étiquettes, ce qui entraîne un ensemble de termes. Ces termes sont stockés dans un index spécifique appelé *index de données*. Nous construisons, ensuite un graphe schéma réduit pour notre graphe de données. Le graphe schéma est stocké dans un index spécifique appelé *index structure* qui contient seulement les éléments structurels (schéma). Pour le classement, des scores sont calculés et associés aux éléments de l'index de données et structure. Les indexes internes sont utilisé durant le processus de construction des requêtes formelles.

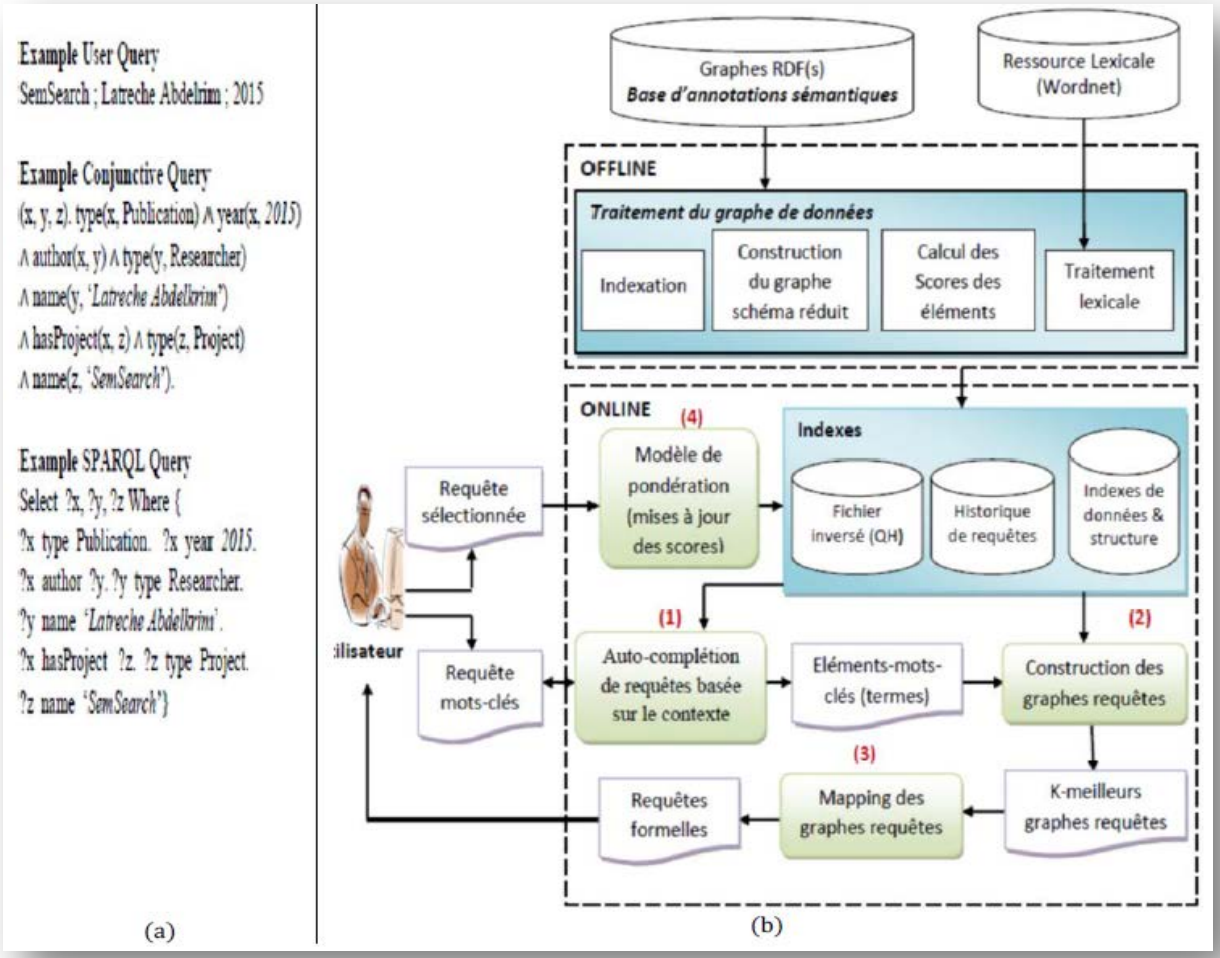


Figure 4.2. a) Exemples de requêtes b) Architecture de notre système

Traitement Online : Nous commençons par un aperçu des différentes étapes de notre processus de recherche par mot clé, qui est représenté sur la figure 4.2 (b). Le défi crucial auquel nous nous adressons est comment aider l'utilisateur à formuler sa requête mots-clés, et d'inférer des requêtes formelles à partir de cette requête. Dans notre travail, nous nous appuyons sur le graphe schéma pour déduire les connexions manquantes. Nous détaillons les différentes étapes de l'approche ci-dessous.

Pour aider l'utilisateur à exprimer son besoin en informations sous forme de requête éléments-mots-clés (ex. «SemSearch ; Latreche Abdelrim ; 2015»), constituée uniquement de termes ou éléments-mots-clés (éléments de la base de connaissance correspondant aux mots clés). Une première phase est exécuté qui consiste en une phase d'auto-complétion de requêtes, qui à partir de quelques caractères entrés par l'utilisateur, en lui présentant de possibles complétions de requêtes les plus pertinentes en se basant sur l'historique de recherche. Une fois cette phase terminée, l'ensemble de termes sera présenté à la phase de construction des graphes requêtes. Ces termes sont combinés avec le graphe schéma réduit extrait de l'index structure pour construire un *espace d'interprétation* c'est-à-dire le graphe schéma réduit augmenté avec les termes de la requête utilisateur. Pendant la recherche des

k-meilleurs graphes requêtes, cet espace d'interprétation est exploré pour trouver ces graphes, c.-à-d. les sous-graphes qui relient tous les termes. Selon un plan de classement, les graphes requêtes générés sont classés, puis pour chaque graphe requête calculés, une requête conjonctive est dérivée par un appariement des éléments du graphe requête aux éléments de la requête conjonctive. Ensuite ces requêtes conjonctives sont présentées à l'utilisateur pour la sélection. En fin, une fois que l'utilisateur a sélectionné la requête formelle qui répond à son besoin en informations, les scores des éléments constituants cette requête sont mis-à-jours pour leurs donnés un poids supplémentaire (en augmentant leurs scores) et de les favorisés durant les sessions de recherches suivantes.

4.3. Prétraitement

Cette section décrit le processus où le graphe de données est prétraité et stocké dans des structures de données spécifiques.

4.3.1. Construction de l'index de données

L'index données qui est couramment utilisé pour la recherche par mots-clés [Tran et al., 2009; Kacholia et al., 2005; He et al., 2007]. Dans notre approche, du point de vue du graphe des données, les mots-clés pourraient se référer à V_C (classes), V_E (entités) ou V_L (valeurs de données) et les arêtes E^D. L'index de données est implémenté comme un index inversé. Pour soutenir un appariement efficace des mots-clés aux éléments du graphe de données, une analyse lexicale (ex stemming , la suppression des mots vides) qui est pris en charge par un moteur de RI classique (tel que. Lucene[33]) est réalisée sur les éléments extrait du graphe de données pour obtenir une liste de termes à indexer pour un élément du graphe particulier. La distance de Levenshtein est aussi utilisée pour supporter une correspondance imprécise des mots-clés aux termes sur la base de la similarité syntaxique. En outre, les termes sont expansés avec les entrées sémantiquement extraites d'une ressource lexicale (tel que WordNet[34]) pour supporter une correspondance basée sur la similarité sémantique (synonymes, hyponymes, etc.). En raison de cette correspondance imprécise, chaque élément retrouvé pour cet index est associés un score indiquant le degré de correspondance.

4.3.2. Construction de l'index structure

Nous utilisons l'index de structure pour effectuer une exploration efficace des sous-structures qui relient les termes (éléments-mots-clés). L'index de structure est essentiellement un graphe schéma réduit. Dans le cas ou le graphe schéma existe, alors le graphe schéma réduit est défini comme suit :

Définition 5 : Un graphe schéma réduit G_S , d'un graphe schéma G^S est un n-uplet (V_S, L_S, E_S), où

1. $V_S = V_C \uplus \{\text{Resource}\}$: est l'ensemble de toutes les classes.
2. $L_S = L_R \uplus \{\text{subclass of}\}$: un ensemble fini de labels d'arêtes où : L_R est l'ensemble de propriétés de classes (relation).

[33] http://lucene.apache.org
[34] http://www.cogsci.princeton.edu/wn/

3. E_S : est un ensemble fini d'arêtes orientées de la forme $e(v_1, v_2)$ avec $v_1, v_2 \in V_S$ et $e \in L_S$ remplissant la condition suivante : $e \in L_S$ si et seulement si $v_1, v_2 \in V_S$.

Dans la pratique, un graphe schéma est souvent non disponible ou incomplet, dans ce cas, des techniques sont employés, pour dérivé un graphe schéma réduit d'un graphe de données $G^D = (V^D, L^D, E^D)$. La procédure de construction d'un graphe schéma réduit se résume comme suit :

1. Supprimer tous les sommets V_L et les arêtes L_A
2. Chaque $v_{ei} \in V_E$ qui est associé à un $v_c \in V_C$, c-à.-d. il existe une arête type (v_{ei}, v_c) , est supprimé et v_c hérite de toutes les L_R de v_{ei}. De ce fait, toutes les relations spécifiées pour v_{ei} sont capturés par leur classe v_C.
3. Tous les autres $v_{ei} \in V_E$ qui n'ont pas de classe explicite sont associés à un sommet prédéfini *Resource*. Aussi, *Resource* hérite de tous les L_R c.-à-d. toutes les relations présentées par v_{ei} qui sont capturés par cette classe prédéfinie.

Essentiellement, nous essayons de dériver les relations entre les classes indiquées dans le graphe des données. L'utilisation de la classe *Resource*, est utilisée quand il n'y a pas de classe explicite dans le graphe de données. En conséquence, le graphe schéma réduit contient uniquement des éléments structurels (schéma), tels que des classes et les propriétés entre les classes.

4.3.3. Calcul des scores des éléments

Durant le processus de construction des graphes requêtes, plusieurs requêtes candidates peuvent être produites correspondant à toutes les interprétations possibles de la requête utilisateur. Un problème se pose: comment classer les meilleures requêtes formelles qui seront présentés à l'utilisateur et qui correspond à son besoins en informations?

Les scores indiquant la pertinence des éléments dans le graphe sont essentiels pour le classement des requêtes. Le mécanisme de calcul des scores devrait permettre des estimations adéquates de la popularité d'un élément du graphe. Dans le contexte des données structurées en graphe, les métriques standard souvent utilisés sont TF /IDF et « PageRank » [Brin et Page, 1998] et le plus court chemin (pour calculer le score des chemins).

Il est intéressant de noter que, bien qu'en parle du calcul des scores, les fonctions de calcul des scores utilisées sont en fait des fonctions de coût, c'est-à-dire qu'elles mesurent la mauvaise qualité (coût élevé, score bas) et non la bonne qualité (faible coût, score élevé) des réponses. Un autre point à noter, est que les requêtes sont considérées comme des graphes. Les graphes sont construits à partir d'un ensemble de chemins P. Le score d'un tel graphe est défini par : $C_G = \sum_{pi \in P} C_{pi}$ où C_{pi} dénote le coût d'un chemin qui est calculé à partir du coût de ses éléments, c'est-à-dire $C_{pi} = \sum_{n \in pi} c(n)$.

Nous avons adoptées dans notre approche la « *fonction de popularité* » comme métrique pour estimer le cout d'un graphe requête. La popularité a pour but de mesurer la popularité d'un graph en tenant compte de la popularité de ses éléments.

Donc, le score d'un graphe requête est calculée par la formule (4.1) :

$$C_G = \sum_{pi \in P} \sum_{n \in pi} C_p(n) \quad (4.1)$$

Alors le coût pour un élément du graphe peut être calculé par l'expression :

$$C_p(n) = \begin{cases} 1 - \frac{|n_{ag}|}{|V|} & , if\ n \in V_C \\ 1 - \frac{|n_{in}|}{|V|} & , if\ n \in V_L \\ 1 - \frac{|n_{ar}|}{|E|} & , if\ n \in L_R \\ 1 - \frac{|n_{aa}|}{|E|} & , if\ n \in L_A \end{cases} \quad (4.2)$$

Où, $|V|$ est le nombre de sommets, and $|E|$ est le nombre d'arêtes dans le graphe données, $|n_{ar}|$ est le nombre des éléments de L_R qui ont été regroupés en sommet n de L_R du graphe schéma réduit, $|n_{aa}|$ est le nombre de l'élément n de L_A dans le graphe données, $|n_{ag}|$ est le nombre des éléments de V_E qui ont été regroupés en un élément n de V_C, et $|n_{in}|$ est le nombre d'arêtes entrantes pour le sommet n.

4.4. Processus d'auto-complétion de requêtes

L'auto-complétion de requêtes (ACR) est une nouvelle fonctionnalité qui a été introduite dans les système de recherche d'information (SRI) et qui a pour tache d'aider l'utilisateur à formuler sans peine sa requête en saisissant uniquement un préfixe (quelques caractères), et donc de prédire la requête attendue par l'utilisateur en lui présentant de possibles complétions de requêtes classées sur la base de différents scores de pertinence. De nombreux modèles ACR ont été proposés ces dernières années comme mentionné dans l'état de l'art. La plus part de ces approches réduisent le problème d'ACR au problème de classement de requêtes. Dans le contexte de la recherche par mots-clés sur les données structurés en graphes, nous introduisons une nouvelle fonctionnalité d'auto-complétion de requêtes dans le processus de construction des requêtes formelles. Dans notre approche on ne considère pas la requêtes utilisateur comme une chaine de caractères formée d'un ensemble de mots, mais plus tôt comme un ensemble de termes (ou éléments-mots-clés) et L'auto-complétion de requêtes se réalise à travers les termes constituant cette requête. Dans ce qui suit nous allons définir et formaliser cette fonctionnalité.

4.4.1. Définitions

a-) L'historique de requêtes (Query History « QH ») : est une structure qui contient les requêtes antérieures soumises par un utilisateur. Cette structure est représenté par la structure $[Id^q, q, \omega^q]$, où :

- Id^q : est l'identificateur de la requête,
- $q = \{t_i / 1 \leq i \leq m\}$: est la requête formée d'un ensemble de termes t_i (élément-mot-clé),
- ω^q : est le poids de la requête q.

Nous considérons que les requêtes sont identiques si elles sont formées de mêmes termes. Le poids de la requête est calculé à partir des poids de ses termes. Si $q_1, \dots, q_k$ sont les requêtes soumises par l'utilisateur, qui sont stockés dans QH, avec q_k la requête la plus récente. Alors, le poids $\omega^q(q_i)$ de la requête q_i, est défini par la formule ci-dessous :

$$\omega^q(q_i) = \sum_{t_j \in q_i} \frac{\omega^t(t_j)}{|q_i|} \quad (4.3)$$

Avec $|q_i|$ et le nombre de termes dans la requête q_i, et $\omega^t(t_j) = \frac{1}{k}\sum_{i=1}^{k} \frac{c(t_j,q_i)}{\alpha^{k-i}}$: est le poids d'un terme t_j qui apparait dans la requête q_i , où $c(t_j, q_i) = C_p(t_j)$, $\alpha > 1$ est un entier positif pour réduire l'importance des termes des requêtes passées et $k - i$ est la distance entre une requête donnée et la requête la plus récente.

La figure 4.3 (b) montre un exemple de cette structure.

b-) L'index inversé (Inverted Index « *InvQH* ») : l'index inversé est une structure de données utilisée pour indexer QH. Cette structure est représentée par $[Id^t, t, \{Id_i^q\}, C_p(t)]$, où :

- Id^t : est l'identifiant d'un terme t,
- t : est un terme donné,
- $\{Id_i^q\}$: l'ensemble des identificateurs des requêtes de QH qui contiennent le terme t,
- $C_p(t)$: est le score du terme t.

L'historique de requêtes et son index inversé sont mis à jour après chaque cession de recherche.

La figure 4.3 (c) montre un exemple de cette structure.

c-) Graphe de complétions de termes (*Term Completion Tree* «T_{TC} ») : Le graph de complétions de termes T_{TC} est un graphe orienté qui permet de stocker l'ensemble des termes de l'index inversé ($InvQH$). Ce graphe se compose de trois types de nœuds :

- un nœud racine unique avec aucunes arêtes en entrée qui représente le début de chaque terme,
- des nœuds intermédiaires qui contiennent une structure de données représentant un seul caractère d'un terme donné,
- et des nœuds de fin de termes avec aucune arêtes en sorties qui marquent la fin des termes.

La structure de données qui représente le caractère d'un termes est formé de $[c, \{Id_i^t\}]$, où

- c : un des caractères du terme,
- $\{Id_i^t\}$: est l'ensemble des identificateurs de termes qui sont stockés dans le l'index inversé, tel que chaque terme identifié par Id_i^t a comme préfixe la chaine de caractères qui est construite à partir du graphe de complétions en partant de racine jusqu'au caractère c.

Chaque chemin du graphe de complétions de termes commençant par un nœud racine et se termine par un nœud feuille représentent un terme.

La figure 4.3 (a) montre un exemple de cette structure, où le préfixe entré par l'utilisateur est $p =$ '***Sem***', qui est un préfixe du terme $t_1 =$ '***Sem****antic Web*' identifié par $Id_1^t = 4$, et du terme $t_2 =$ '***Sem****Search*' identifié par $Id_2^t = 5$.

d-) Requête utilisateur (*User Query* « $\boldsymbol{Q_u}$ »): $Q_u = \{t_1, \dots, t_n\}$ où $\{t_1, \dots, t_n\}$: est l'ensemble de termes (éléments-mots-clés), où $t_i \in V_C \uplus V_L \uplus L_R \uplus L_A$.

e-) Préfixe ($\boldsymbol{p}$): est une chaine de caractères, que l'utilisateur est entrain de saisir. Il est a noté qu'un terme en construction ou en modification est considéré comme un préfixe.

4.4.2. Construction et utilisation

Étant donné une requête utilisateur Q_u composée de n termes ($n \geq 0$) et un préfixe p saisi par un utilisateur. Notre système génère une liste classée de complétions de requêtes les plus pertinentes qui seront présentés à l'utilisateur pour une éventuelle sélection. A la fin du processus d'auto-complétion de requêtes, une requête d'utilisateur Q_u constituée uniquement de termes est générée pour être utilisée dans l'étape suivante. Le processus se compose de deux phases :

4.4.2.1. Construction

Durant cette phase et en exploitant le journal des requêtes, on construit les structures de données nécessaires au fonctionnement de notre système d'auto-complétion de requêtes, à savoir :

- L'historique de requêtes ($\boldsymbol{QH}$), qui est trier par ordre décroissant selon le poids des ces requêtes,
- l'index inversé ($\boldsymbol{InvQH}$), qui est trier par ordre alphabétique selon ses termes.
- et en fin le graphe de complétions de termes ($\boldsymbol{T_{TC}}$).

Ces structures sont mises à jour, après chaque session de recherche.

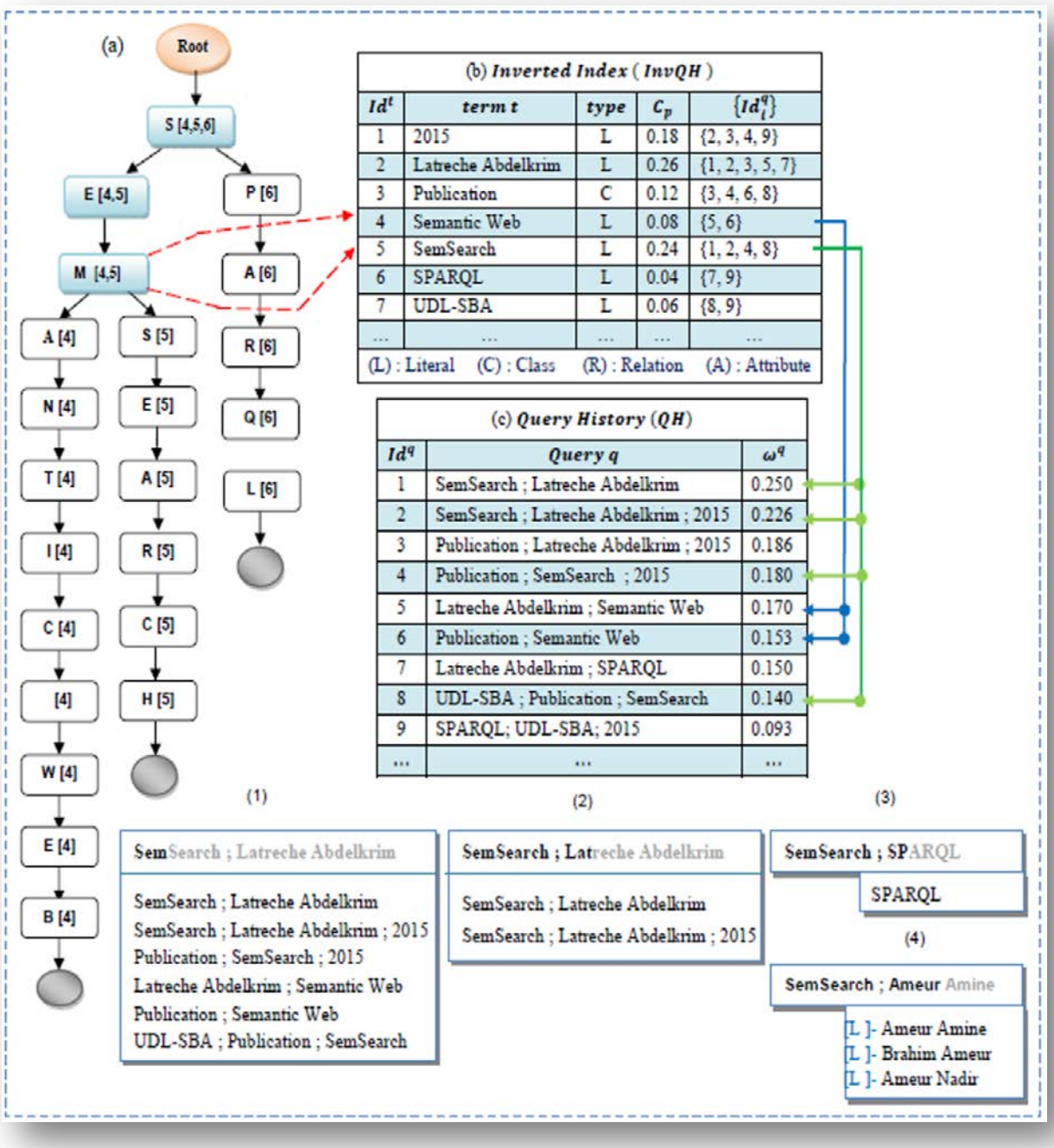


Figure 4.3. Illustration du processus d'auto-complétion de requêtes.

4.4.2.2. Utilisation

Etant donné un entier k : qui représente le nombre de complétions de requêtes ou de termes qui seront présentés à l'utilisateur et une requête utilisateur Q_u. A chaque fois que le préfixe p change (soit par ajout, modification, ou suppression de caractères), alors l'algorithme 1 est exécuté. Tous d'abord, pour chaque terme t_i de la requête Q_u, la fonction $CompQueyTerme(t_i)$ génère une liste de requêtes contenant le terme t_i. Une fois tous les termes traités, alors CQ_T, doit contenir l'ensemble de requêtes résultant de l'intersection de tous les ensemble de requêtes trouvées précédemment. Pour le préfixe p de la requête Q_u, la

fonction $CompQueryPrefix(p)$ génère une liste CQ_P de requêtes contenant les termes qui ont pour préfixe p. En fin, la liste finale de complétions de requêtes $CL = CQ_T \cap CQ_P$ est générée et présentée à l'utilisateur pour sélection.

Plusieurs cas peuvent se présentés : 1) Si la liste CL n'est pas vide ($CL \neq \emptyset$), alors la fonction $k\_ranked(CL)$ renvoie une liste classée des k-meilleurs requêtes, qui sera présentée à l'utilisateur qui sélectionne la requête qui répond à son besoin en information qui sera ensuite stocké dans Q_u. Si au contraire, la liste CL est vide ($CL = \emptyset$) , cela veut dire qu'il n'existe pas de requêtes contenant à la fois les termes de la requête utilisateur et les termes qui ont p comme préfixe, alors dans ce cas la d'autres alternatives se présentent : 2) Si la liste CQ_P est vide ($CQ_P = \emptyset$) , cela veut dire, qu'il n'existe pas de termes qui ont pour préfixe p dans $InvQH$, alors, par un accès direct à *l'index de données* par le préfixe p , la fonction $k\_ranked(CompTermePrefix(p))$ génère une liste classée CT des k-meilleurs termes qui sont similaires (syntaxiquement ou sémantiquement) au préfixe p qui sera présentée à l'utilisateur pour sélection ; une fois le terme sélectionné, il sera ajouté à la requête utilisateur Q_u . Dans le cas contraire ($CQ_P \neq \emptyset$) , c'est-à-dire, il existe des requêtes contenant les termes qui ont p comme préfixe, dans ce cas la d'autres cas se présentes : 3) Si la liste CQ_T est vide ($CQ_T = \emptyset$), ce la veut dire, qu'il n'existe pas de termes appartenant à $InvQH$, alors fonction $k\_ranked(CQ_P)$ renvoi la liste classée CQ des k-meilleurs requêtes contenant les termes qui ont p comme préfixe et qui sera présentée à l'utilisateur qui sélectionne une requête qui sera ensuite stocké dans Q_u . En fin, si $CQ_T \neq \emptyset$ c'est-à-dire, il existe des termes de la requête utilisateur dans $InvQH$, la fonction $k\_ranked(CompTerme(p))$ génère une liste classée CT des k-meilleurs termes qui ont comme préfixe p qui sera présentée à l'utilisateur pour sélection ; une fois le terme sélectionné, il sera ajouté à la requête utilisateur Q_u.

La figure 4.3 illustre un exemple de ce processus.

Dans ce qui suit, nous détaillerons chaque fonction utilisée dans notre algorithme :

1) $CompQueryPrefix(p)$: est une fonction qui permet de générer l'ensemble des requêtes $q_i \in QH$ qui contiennent l'ensemble de termes t_i, qui ont p comme préfixe. La solution consiste à parcourir le chemin correspondant au préfixe p dans l'arbre des complétions T_{TC} par l'algorithme DFS (*Depth First Search)*. A tout moment du parcours et à partir de la structure de données représentant le caractère en cours de saisie, en extrait l'ensemble des identificateurs de termes $\{Id_i^t\}$ qui sont stockés dans cette structure. Chaque identificateur $\{Id_i^t\}$ est utilisé pour accéder à l'index inversé ($InvQH$), pour en déduire l'ensemble de requêtes contenant le terme t_i. En fin une réunion de tous les ensemble de requêtes, obtenus pour chaque terme est réalisé pour obtenir une liste finale de requêtes qui sont constitués de termes qui ont p comme préfixe. Cette fonction est formalisée comme suit :

$$\text{CompQueryPrefix(p)} = \bigcup_{\text{t}_\text{i} \in \text{InvQH}} \{\text{q}_\text{j}\}, \text{ où } \text{t}_\text{i} \in \text{q}_\text{j} \text{ et p est un préfixe de } \text{t}_\text{i}$$

2) $CompQueyTerme(t_i)$: est une fonction qui permet de générer l'ensemble des requêtes $q_i \in QH$ qui contiennent le terme t_i. Ceci est réaliser tout simplement par accès direct par le terme t_i à l'index inversé ($InvQH$) pour obtenir la liste des requêtes contenant le terme t_i.

3) $CompTerme\ (p)$: est une fonction qui permet de générer l'ensemble de termes $t_i \in InvQH$ qui ont p comme préfixe. Ceci est réaliser tout simplement par accès par le préfixe p à l'index inversé ($InvQH$).

4) $CompTermePrefix\ (p)$: est une fonction qui permet d'accéder à l'index de données par le préfixe p, pour trouver les termes de la base de connaissance (c.-à-d. classes, littéraux, relations et attributs) correspondants à ce préfixe. Deux types de mapping peuvent être utilisées: le mapping syntaxique, qui emploie les techniques de comparaison de chaîne de caractères (tels que le stemming, Sub-String et Levenshtein Distance) et le mapping sémantique (tels que la synonymie) qui utilise principalement une ressource lexicale comme WordNet, afin de trouver les termes, qui sont syntaxiquement ou sémantiquement similaire au préfixe p. Nous assignons un score $St(t) = Sm(t) * C_p(t)$ à chaque terme trouvé, où $Sm(t)$ le score de similarité d'un terme t and $C_p(t)$ le score de popularité d'un terme t. Le score $St\ (t)$ est utilisé pour classer les termes générés.

Algorithm 1 : ComputeUserQuery

Input : Q_u , k , p
Output : Q_u or Ø
$CQ_T = \bigcap_{t_i \in Q_u} CompQueyTerme(t_i)$
$CQ_P = CompQueryPrefix(p)$
$CL = CQ_T \cap CQ_P$
If $CL \neq \emptyset$ **Then**
  $CQ = k\_ranked(CL)$
  $Q_u = UserSelection(CQ)$
Else
  If $CQ_P = \emptyset$ **Then**
    $CT = k\_ranked(CompTermePrefix(p))$
    $Q_u = Q_u \cup UserSelection(CT)$
  Else
    If $CQ_T = \emptyset$ **Then**
      $CQ = k\_ranked(CQ_P)$
      $Q_u = Q_u \cup UserSelection(CQ)$
    Else
      $CT = k\_ranked(CompTerme(p))$
      $Q_u = Q_u \cup UserSelection(CT)$
    End
  End
End
Return Q_u

4.5. Construction des requêtes formelles

Dans cette section, nous décrivons tous en donnant des définitions formelles le processus de construction des requêtes formelles conjonctives à partir d'un ensemble de termes entrés par l'utilisateur. Ce processus est constitue de deux étapes principales: (1) *Construction des k-meilleurs graphes requêtes*, et (2) *Mapping des graphes requêtes*.

4.5.1. Construction des k-meilleurs graphes requêtes

Avant d'explorer les k-meilleurs graphes requêtes, nous devons tous d'abord construire la structure qui doit contenir tous les éléments nécessaires pour la construction de possibles graphes requêtes. Cette structure est appelée « *espace d'interprétation* » composé de deux parties essentielles :

- les termes qui constitues la requête utilisateur pour explorer les constantes des graphes requêtes et
- le schéma (les éléments structurels) pour générer les prédicats des graphes requêtes.

Pour obtenir l'espace d'interprétation il suffit tout simplement de combiner le graphe schémas réduit et les termes de la requête utilisateur.

Formellement un espace d'interprétation est défini comme suit :

Définition 6 : Étant donné K un ensemble de termes (éléments-mots-clés), l'espace d'interprétation est un graphe G_S^R, qui consiste en un graphe schéma réduit G_S contenant en outre :

- Si le terme est un sommet $v_v^k \in V_L$ avec ses éléments adjacents tel que l'arête $e_a \in L_A$ et les sommets $v_{c1}, \dots, v_{cn} \in V_C$, alors les arêtes $e_a(v_v^k, v_{c1}, \dots, v_{cn})$ seront ajoutés pour connecter v_v^k aux sommets $v_{c1}, \dots, v_{cn}$.
- Si le terme est une arête $e_a^k \in L_A$ avec ses éléments adjacents tels que les sommets $v_{c1}, \dots, v_{cn} \in V_C$ alors l'arrête $e_a^k(value, v_{c1}, \dots, v_{cn})$ sera ajoutée au graphe. Notez que le sommet *value* est un élément artificiel.
- Dans le cas contraire, le terme doit être un sommet $v_c^k \in V_C$ ou une arête $e_r^k \in L_R$. Dans ce cas, aucun autre élément ne doit être ajouté au graphe qui contient déjà cet élément.

4.5.1.1. Exploration des k-meilleurs graphes

La tâche suivante consiste en la recherche des graphes requêtes minimales dans l'espace d'interprétation donné. Tout d'abord, nous commençant par donner la définition formelle d'un graphe requête :

Définition 7 : Soit $G_S^R = (V_S^R, E_S^R)$ est l'espace d'interprétation. Un graph requête est un sous-graphe de G_S^R défini par $QG = (V_S^q, E_S^q)$ où $V_S^q \subseteq V_S^R$ et $E_S^q \subseteq E_S^R$, tel que $\forall\, v_i, v_j \in QG$, il existe un chemin reliant $v_i \; et \; v_j$. Un sous-graphe requête QG_iest minimal s'il n'existe pas un autre graphe QG_j de G_S^R de telle sorte que score$(QG_i) <$ Score(QG_j).

Dans la plus part des approches de recherche par mots-clés et notamment dans [Tran et al., 2011], la phase d'appariement des mots-clés aux éléments de la base de connaissances peut générer un nombre important de termes en raison de l'ambigüité de la requête utilisateur, ce qui implique un nombre important de requêtes formelles construites à cause de toute les combinaisons possible de ces termes. Il est intéressant de noter que la conception de l'algorithme de recherche par mots clés qui impose l'ajout de tous les termes dans le graphe synthèse, va augmenter considérablement l'espace d'interprétation. Nous croyons

fermement qu'une telle approche est irrationnelle en tenant compte de la conception de cet algorithme, qui exige d'explorer tous les termes et de combiner ces termes pour construire d'éventuelles requêtes formelles candidates, ceci va augmenter considérablement la complexité en temps et en espace de cette algorithme. Pour surmonter ce problème nous avons proposé un algorithme efficace qui améliore les performances de cet algorithme en qualité de complexité en temps et en espace.

Exploration de l'espace d'interprétation : Cette phase consiste en la génération des k-meilleurs graphes requêtes par l'exploration de l'espace d'interprétation, comme montrée dans l'algorithme 2. L'algorithme a en entrée l'ensemble de termes $K = \{e_1, \ldots, e_n\}$, constituant la requête utilisateur et le graphe schéma réduit, G_s, et un nombre k désigne le nombre de sous-graphes à construire. La valeur $dmax$ désigne la profondeur de l'exploration. Afin de suivre les chemins visités pendant l'exploration, le concept de curseur est utilisé. Un curseur est représenté par $c\ (n, e, p, d, w)$, où n est l'élément du graphe visité, e est un terme représentant l'origine du chemin capturé par c et p est le curseur parent de c . En outre, le coût w et la distance d sont stockés pour le chemin. Afin de suivre les informations relatives à un élément du graphe n et les différents chemins découverts pour n au cours de l'exploration, une structure de données de la forme $n(w, (C_1, \ldots, C_n))$ est employé, où w est le coût de n tel que discuté plus haut et C_i est une liste triée de curseurs représentant des chemins de e_i à n.

La première étape de l'algorithme est la construction de l'espace d'interprétation. Ce graphe est construit au moment de la requête, augmentant les informations présentes dans le graphe schéma réduit. Le processus d'exploration utilise les termes générées durant la phase d'auto-complétion comme éléments de départ $K = \{e_1, \ldots, e_n\}$, et en construisant également leurs curseurs respectifs (lignes 1-2). Tous les curseurs générés sont maintenus dans des files d'attente de priorité, $Q_i \in LQ$. Chaque files Q_i maintient les curseurs ayant comme origine le terme e_i. Chaque élément de départ forme un point d'exploration indépendante du graphe. En effet, au cours de l'exploration et pour chaque élément de départ, de nombreux chemins différents sont explorés et pour chacun d'eux un curseur différent est utilisé. A chaque étape du processus d'exploration, un curseur (c'est-à-dire un chemin) avec le coût le plus bas est choisi pour une expansion à partir d'une file d'attente des curseurs (ligne 4). Si l'élément actuel visité a également été exploré par d'autres curseurs émanant de tous les autres éléments de départ, un sous-graphe est trouvé et l'élément actuel visité constitue un élément de connexion (ligne 24). Ce sous-graphe, qui est produit par la combinaison des chemins émanant de tous les éléments de départ et se termine à l'élément de connexion, est inséré dans une liste de sous-graphes candidats (ligne 26), SG, à partir de laquelle seul les k-meilleurs seront sélectionné à la fin. Le processus d'exploration se termine lorsque l'une des conditions suivantes est remplie:

1. Tous les k-meilleur sous-graphes ont été générés.
2. La profondeur d'exploration atteint une limite supérieure, $dmax$ (ligne 7)
3. Tous les éléments du graphe ont été explorés.

Nous avons amélioré l'algorithme existant de la manière suivante:

- Nous avons introduit une autre condition qui met fin au processus d'exploration d'un curseur émanant d'un élément de départ spécifique. Ainsi, lorsque le processus d'exploration est sur le point d'étendre un curseur à un élément qui est lui-même un élément de départ, il est visité, mais un curseur n'est pas produit pour cet élément (ligne 10).
- En introduisant la phase d'auto-complétion de requêtes qui génère une seule requête utilisateur, constitué uniquement de termes, ce qui implique qu'on peut éliminer la partie combinaison des termes dans notre algorithme.

Ces améliorations peuvent contribuer à réduire à la fois le temps et l'espace d'exploration.

Algorithm 2 : Explore graph

```
   Input : k, d_max, G_s , K = {e_1 , .... e_n }
   Output :Top-k subgraphs
   // For each element add a cursor to Q_i ∈ LQ
1  foreach  e ∈ K do
2      | Q_i.add(new Cursor(e, e, Ø, 0, e.w));
3  end
4  while  not all queues Q_i ∈ LQ are empty do
5      c ← minCostCursor(LQ);
6      n ← c.n;
7      if  c.d < d_max  then
8          n.addCursor(c);
9          // do not expand keyword terms further
10         If  n ∉ K then
11             // get all neighbors except parent element of  c
12             nbrs ← neighbors(n)\(c.p).n;
13             if  nbrs ≠ Ø then
14                 foreach  n ∈ nbrs do
15                     // check for cyclic path when n already visited by c
16                     If  n ∉  parents(c) then
17                         // add new cursor to respective queue
18                         Q_i.add(new Cursor(n, c.e, c.n, c.d + 1, c.w + n.w));
19                     end
20                 end
21             end
22         end
23         Q_i.pop(c);
24         If  n is a connecting element then
25             // generate possible subgraphs
26             SG.add(gensubgraphs(n));
27         end
28         SG_best  ← k-best(SG);
29         lowestCost ← minCostCursor(LQ).w;
30         highestCost ← k-ranked(SG_best);
31         if  highestCost < lowestCost then
32             // top-k subgraphs have been computed
33             topk ← SG_best;
34         end
35         if  topk ≠ Ø then ;
36             return  topk;
37         end
38     end
39 end
40 //Return the current best, where the top-k results have not been calculated
41 return  SG
```

4.5.2. Mapping des graphes requêtes

Au cours de cette étape, les graphes requêtes QG_i générés au cours de la phase d'exploration sont traduits en requêtes formelles conjonctives. Fondamentalement, les sommets des graphes requêtes correspondent aux variables et aux constantes de la requête conjonctive, alors que les arêtes correspondent aux prédicats de la requête conjonctive. La Figure 4.1(c) conjointement avec la figure 4.1(d) illustrent ces correspondances. Dans ce qui suit nous détaillerons le processus de traduction.

a-) Traitement des sommets

Les étiquettes des sommets peuvent désigner des constantes de la requête ou représenter des variables. Deux fonctions sont définies, $var(n)$ qui renvoi une variable et $const(n)$ qui renvoi l'étiquette du sommet.

b-) **Traitement des arêtes relation**

L'arête $e(v_1, v_2)$, où $e \in L_R$ sont traduits en trois prédicats de requêtes de la forme :
type $(var(v_1), const(v_1))$, type $(var(v_2), const(v_2))$ et $e(var(v_1), var(v_2))$

Par exemple, $hasproject\ (Publication,\ Project)$ est traduit en :

$type(x, Publication)$, $type\ (z, Project), et\ hasproject(x, z)$.

b-) **Traitement des arêtes attribut**

L'arête $e(v_1, v_2)$, où $e \in L_A$ et $v_2 \rightarrow valeur$ sont traduits en deux prédicats de requête de la forme :
$type\ (var(v_1), const\ (v_1))$, et $e(var(v_1), const\ (v_2))$.

Par exemple, $name\ (Project,\ 'SemSearch')$ est traduit en :

$type(z, Project)\ et\ name(z, 'SemSearch')$.

La requête résultante est simplement une conjonction de tous les prédicats générés pour un graphe requête. De plus les requêtes conjonctives représentent un fragment de SPARQL. La figure 5.2(a) montre un exemple de requête conjonctive et requête SPARQL.

4.6. Modèle de pondération

Le modèle de pondération consiste en l'introduction d'une fonction de pondération qui attribue des poids supplémentaires aux éléments du graphe de données pour les rendre pertinents dans le contexte de recherche en cours. Soit e un élément du graphe de données et supposons qu'à l'instant t, le graphe requête QG_t pour la requête mots-clés Q_t a été générée et sélectionné par l'utilisateur. La fonction de pondération est définie en tant qu'une fonction monotone décroissante de la distance graphique entre e et QG_t, et elle est définie par l'expression (4.4) :

$$wf(e, QG_t) = \frac{1}{\beta^{\alpha + \min_{m \in QG_t} dist(e,m)}} \qquad 4.4$$

sachant que : $(wf(e, QG_t) = 0$ si $min_{m \in QG_t}\, dist(e, m) \geq \varepsilon)$, ε et est une constante, $\beta > 0$ est un entier positif, $\alpha \geq 0$ est un facteur de décroissement et $dist(e, m)$ est la distance entre e et m qui est un élément du sous-graphe QG_t.

Donc à l'instant T, le score d'un élément e est donné par :

$$C_p(e,T) = C_p(e,T-1) + wf(e,QG_T) \quad 4.5$$

C'est-à-dire que, connaissant le score de e à l'instant $T-1$, alors le score de e à l'instant T est tout simplement donné par la formule 4.5.

Dans la pratique, nous utilisons $\beta > 1$, $\varepsilon = 2$ et $\alpha \geq 0$. Alors, seuls les scores d'un élément e et ses voisins (sommets et arêtes) peuvent être mis à jours.

4.7. Conclusion

Dans ce chapitre, nous avons formalisé le problème de la construction basée sur le contexte de requêtes formelle à partir de requêtes mots-clés et présenté une approche efficace pour le résoudre. Notre système vise à traduire les requêtes mots-clés en requêtes formelles conjonctives en utilisant les connaissances disponibles dans une base d'annotation pour réduire l'écart entre la recherche sémantique basée sur la logique formelle et les utilisateurs habitués à utiliser les moteurs de recherche classiques basés sur les mots-clés. Notre approche peut se résumé ainsi : (i) processus d'auto-complétion de requêtes ; (ii) construction de requêtes formelles conjonctives liant les entités détectées à l'étape précédente; (iii) classement des requêtes construites, (iv) sélection de la bonne requête par l'utilisateur, v) Pondération des scores des entités de la requête sélectionnées ainsi que leurs voisins les plus proches.

Nous avons définis le problème de construction des requêtes formelles, où nous avons présenté le modèle de données et de requêtes et l'architecture de notre système. Nous avons aussi discuté plus en détail des aspects spécifiques des données et du traitement des requêtes: prétraitement et indexation. Nous avons introduit et formalisé la fonctionnalité d'auto-complétion de requêtes qui a pour tâche essentielle d'aider l'utilisateur à formuler sans peine sa requête mots-clés constitue uniquement de termes (éléments de la base de connaissances). Nous avons exploité les informations sur les requêtes précédentes (historique de recherche) pour les utiliser en tant qu'informations contextuelles pour influencer la génération d'une nouvelle requête.

La principale contribution de ce document est l'exploitation des annotations sémantiques dans la recherche d'information sur le Web. De plus, nous devons noter que certaines fonctionnalités du prototype sont actuellement implémentées de façon relativement simplifiée et en conséquence nous demeurons convaincus, compte tenu de l'intérêt de cette approche, que ce prototype doit être soutenu et poursuivi afin de lui apporter toutes les améliorations nécessaires.

Dans le chapitre suivant, nous allons présenter une évaluation complète de notre approche et nous allons démontrer que l'approche proposée donne des résultats encourageants.

Chapitre 5. Implémentation et Evaluation

5.1. Introduction

Dans ce chapitre nous allons présenter en détaille l'implémentation et l'expérimentation de notre système, afin de valider certains éléments du cadre méthodologique de l'approche proposée. L'objectif principal de ces expérimentations est d'abord, de prouver l'applicabilité de notre approche, ensuite de comparer, tester et valider chacune de nos contributions.

5.2. Implémentation

Pour la mise en œuvre et l'évaluation de notre approche proposée, nous avons développé un outil nommé "ISA" qui est implémenté en Java et Jena[35] API, et qui permet en se basant sur le contexte de recherche de construire automatiquement des requêtes formelles (SPARQL) à partir de requêtes mots-clés entrées par un utilisateur. Notre système permet par un processus d'auto-complétion de requêtes d'aider l'utilisateur à formuler sa requête mots-clés qui est formée uniquement d'un ensemble de termes (entités de la base de connaissance), ensuite de construire automatiquement des requêtes formelles (SPARQL) à partir de ces mots-clés. La requêté formelle (SPARQL) sélectionnée par l'utilisateur est ainsi directement envoyée à un moteur de requête SPARQL *ARQ*[36] pour avoir les ressources relatives à sa requête.

Dans ce qui suit, nous présentons l'architecture et le fonctionnement du système, ainsi que les différents composants du système et leurs interactions.

A l'instar de l'approche présentée dans ce travail, le système se divise en deux modules principaux. Chaque module est chargé d'une des deux grandes étapes du processus d'interprétation d'une requête utilisateur, comme illustré dans la figure 4.2(a). Le premier module « *prétraitement du graphe de données* » permet de charger la base d'annotation RDF, puis différentes informations sont extraites à partir du graphe de données et sont stockées dans des structures de données spécifiques sous formes d'indexes internes. Le second module « *construction des requêtes formelles* » assiste l'utilisateur pour l'aider à construire sa requête mots-clés, ensuite construit automatiquement les requêtes formelles (SPARQL) nécessaires à partir de ces mots-clés, et en fin mettre à jour les scores des termes de la requête sélectionné par utilisateur, ainsi que leurs voisins les plus poches.

Le premier module est architecturé de façon assez classique. Tout d'abord, les étiquettes des éléments du graphe de données sont extraites. Une analyse lexicale standard (y compris stemming, la suppression de mots vides, etc.) qui est pris en charge par un

[35] http://jena.sourceforge.net/
[36] http://jena.apache.org/documentation/query/

moteurs de RI classique (tel que. Lucene[37]) est réalisée sur les éléments extrait du graphe de données pour obtenir une liste de termes à indexer pour un élément du graphe particulier. En outre, les termes obtenue sont expansés avec les entrées sémantiques extraites d'une ressource lexicale (tel que WordNet[38]) pour supporter une correspondance basée sur la similarité sémantique (synonymes, hyponymes, etc.). Ces termes sont stockés dans une structure spécifique appelée « index de données ». Ensuite un graphe schéma réduit est construit pour le graphe de données. Le graphe schéma est stocké dans une structure appelée « index structure » qui contient seulement les éléments structurels (schéma). Pour le classement, des scores sont calculés et associés aux éléments de l'index de données et l'index de structure. Les indexes internes sont utilisés durant le module de *construction des requêtes formelles*.

L'implémentation du second module est plus originale et présente selon nous une réelle nouveauté. En effet, le composant « *auto-complétion de requêtes* » aide l'utilisateur à exprimer son besoin en informations sous forme de requête mots-clés, et qu'à partir de quelques caractères entrés par l'utilisateur, lui présente de possibles complétions de requêtes les plus pertinentes en se basant sur trois structure de données construites préalablement, qui sont : *l'historique de requêtes, l'index inversé et l'arbre des complétions de termes*. Ensuite, l'ensemble des éléments-mots-clés générés par le premier composant sera présenté au composant « *construction des graphes requêtes* » pour construire d'éventuels graphes requêtes. Ces éléments-mots-clés sont combinés avec le graphe schéma réduit extrait de l'index structure pour construire un *espace d'interprétation* (graphe réduit étendu). Pendant la recherche des k-meilleurs graphes requêtes, cet espace de recherche est exploré pour trouver ces graphes, c.-à-d. les sous-graphes qui relient tous les éléments-mots-clés. Pour chaque graphe requêtes généré, le composant « *mapping des graphes requêtes* » génère une requête conjonctive qui est dérivée par un appariement des éléments du graphe requête aux éléments de la requête conjonctive. Ensuite ces requêtes conjonctives sont présentées à l'utilisateur pour la sélection. Enfin, une fois que l'utilisateur a sélectionné la requête formelle qui répond à son besoin en informations, les scores des éléments constituants cette requête sont mis à jours par le composant « *modèle de pondération*» pour leurs donner un poids supplémentaire et de les favoriser durant les sessions de recherche suivantes, ensuite cette requête est traduite en requête SPARQL et est ainsi directement envoyée à un moteur de requête SPARQL *ARQ*[39] pour avoir les ressources relatives à sa requête.

5.3. Expérimentation et évaluation

Dans cette section, nous discutons les expériences que nous avons réalisé pour évaluer les performances de notre système en qualité d'effectivité (c'est-à-dire l'évaluation de la qualité des interprétations) et d'efficacité (c'est-à-dire le temps de réponse). Pour bien testé les performances de notre approche, nous avons comparé notre système (nommée ISA) avec un système de référence (que nous appelons TKQ), et qui est constitué des étapes suivantes:

[37] http://lucene.apache.org.
[38] http://www.cogsci.princeton.edu/wn/.
[39] http://jena.apache.org/documentation/query/

(i) appariement des mots de la requête aux entités sémantiques de la base de connaissances ;
(ii) construction de graphes requêtes liant les entités détectées à l'étape précédente ;
(iii) classement des requêtes construites.

5.3.1. L'environnement de développement

Les expériences ont été menées sur une machine avec un processeur Intel duel Core 2.0 GHz et 4 Go de mémoire fonctionnant sous Windows 7 Professionnel.

5.3.2. Le jeu de données test

Dans notre travail, nous avons dû faire face à l'absence de cadre d'évaluation pour tester notre système. En conséquence, pour évaluer notre approche, nous avons utilisés un ensemble de données réelles, tels que DBLP, DBpedia, semanticWeb.org et le portail AIFB, tous sont publiquement disponibles en RDF. Nous avons construit une base d'annotation sur le domaine du publication des documents scientifiques que nous avons peuplée de façon à obtenir une base de connaissances contenant la réponse aux requêtes recueillies. Afin d'évaluer notre approche et avec la collaboration de nos collègues, nous avons donc construit notre propre cadre d'évaluation. Un ensemble de requêtes mots-clés avec leurs descriptions en Langage Naturel (LN) ont été proposées et nous nous sommes limités aux requêtes qui, en principe, devraient donner de réponses compte tenu des données décrites ci-dessus. Des exemples de requête sont illustrés dans le tableau 5.1.

Tableau 5.1. Requêtes mots-clés avec leurs descriptions en Langage Naturel (LN).

N°	Requête mots-clés	Description en LN
1	Publications Smart Web Hitzler 2002	retrouver toutes les publications publiées par Pascal Hitzler dans SmartWeb en 2002
2	Jeans X-Media publications	retrouver les publications de l'auteur Jeans Davies qui sont associées au projet X-Media
3	algorithm 1999	retrouver tous les papiers concernant les algorithmes publiés en 1999
4	Projects Blohm	retrouver tous les projets sur lesquelles Sebastian Blohm travaille
5	Latreche Semantic Web	retrouver toutes les publications publiées par Latreche Abdelkrim dans le domaine du Web Semantic
…	…	…

5.3.3. Métriques

L'effectivité de notre système (ISA) et du système (TKQ) est mesurée à l'aide des métriques d'évaluation standard en RI, qui sont : la *Normalized discounted cumulative gain (NDCG)*, la *précision (P@K) et Mean Reciprocal Rank (MRR).* Dans ce qui suit, nous définissons ces mesures, ainsi que les justifications pour leurs utilisations.

Notre choix d'une métrique d'évaluation des interprétations de requête est d'évaluer comment les k-meilleurs interprétations générées par une approche sont considérées comme

l'interprétation souhaitée. En outre, elle évalue la qualité de classement des interprétations avec respect de leur pertinence relative par rapport à l'interprétation souhaitée.

Plus précisément, nous adoptons une métrique d'évaluation standard en RI appelée « *Normalized Discounted Cumulative Gain* $(NDCG)$ » avec une fonction de pertinence affinée. Cette mesure est conçue pour des situations non binaires, notions de pertinence (c'est-à-dire dans [0, 1]). Tous d'abord nous définissons le « *Discounted Cumulative Gain* » (DCG_K) , qui est donné par :

$$DCG_K = \sum_{i=1}^{K} \frac{2^{rel_i}-1}{log_2(1+i)} \quad 5.1$$

Ou

$$DCG_K = rel_1 + \sum_{i=2}^{K} \frac{rel_i}{log_2(i)} \quad 5.2$$

Où K est le nombre des k-meilleurs interprétations générées, rel_i est la pertinence de l'interprétation résultante classée à la position i. Dans notre approche, la pertinence entre une interprétation résultante QI et une interprétation souhaitée QI_D pour une requête mot-clé donnée Q ne peut pas être simplement caractérisée par une correspondance ou non. QI et QI_D sont tous les deux des sous-graphes de l'espace d'interprétation et pourraient avoir un certain chevauchement, ce qui signifie que $rel_i \in [0,1]$. Dans cette expérience, nous définissons la pertinence entre une interprétation candidate QI et l'interprétation souhaitée QI_D par :

$$rel_i = \frac{|QI_i \cap QI_D|}{|QI_i \cup QI_D|} \quad 5.3$$

où QI_i est l'interprétation placée à la position i.

Le DCG peut être normalisé en triant la liste des résultats par pertinence, produisant un idéal DCG à la position k, appelé $IDCG$. Pour une requête, le DCG normalisé à K est calculé par:

$$NDCG_K = \frac{DCG_K}{IDCG_K} \quad 5.4$$

avec $NDCG_K \in [0,1]$.

Notez que dans un parfait algorithme de classement, le DCG sera le même que $IDCG$, produisant un $NDCG$ de *1,0*.

D'autre part, nous utilisons une autre métrique de précision pour évaluer les résultats. La précision d'une liste des k-meilleurs interprétations candidates est donné par :

$$P@K = \frac{|Interprétations\ pertinentes\ dans\ TOPK|}{|TOPK|} \quad 5.5$$

$P@K$ est la proportion des interprétations pertinentes qui sont générées dans TOPK. Parce que, parfois, lorsque les votes des utilisateurs sont répartis uniformément, l'Interprétation du consensus ne peut représenter la réponse la plus attendue, notre évaluation basée sur NDCG peut ne pas être convaincante. La métrique de précision peut surmonter cette limitation en considérant les interprétations candidates que plus de 10% des gens ont choisi les interprétations souhaitées.

Afin de caractériser l'ambiguïté des requêtes de l'utilisateur, nous définissons un concept appelé : « *Query Ambiguity* » (QA) qui défini par :

$$QA(Q_u) = log_2(1 + \prod_{w_i \in Qu} |[\![w_i]\!]|) \quad 5.6$$

où $[\![w_i]\!] = \{e_j / e_j \text{ correspond au mot clé } w_i, \}$, et $\prod_{w_i \in Qu} |[\![w_i]\!]|$ est le nombre de toutes les combinaisons possibles des termes correspondant aux mots-clés constituant la requête utilisateur Q_u.

$QA(Q_u)$, est une métrique utilisée en tant que mesure de performance dans notre approche.

Pour évaluer l'efficacité du classement du système d'auto-complétion de requêtes, la métrique « *Mean Reciprocal Rank* » (MRR) est utilisée. Pour un préfixe p d'une requête q de l'ensemble de requêtes Q, une liste de complétions de requêtes candidates correspondantes $Q_I(p)$ et la requête finale q' soumise par l'utilisateur sont données. Ensuite, le « *Reciprocal Rank* » (RR) pour ce préfixe est calculé comme suit:

$$RR = \begin{cases} \frac{1}{rank\ of\ q' in\ Q_I(p)}, & if\ q' \in Q_I(p) \\ 0, & sinon \end{cases} \quad 5.7$$

Enfin, le score MRR est calculé comme la moyenne de tous les scores RR pour les préfixes de requêtes testés dans Q. A partir de cette formule, il est clair que MRR est une métrique axée sur la précision.

Le choix de MRR en tant que métrique de performance est commun dans les paramètres qui concernent la recherche d'une seule solution connue. MRR traite tous les préfixes du test de manière égale, c'est-à-dire qu'il est calculé en faisant la moyenne de tous les scores RR.

5.3.4. Evaluation de l'effectivité

Nous avons comparé les résultats obtenus par notre approche « ISA » et les résultats obtenus par l'approche de référence « TKQ ». Pour chaque approche et pour chaque requête mots-clés, une liste de 10-meilleurs requêtes est générée. Les résultats montrés dans les figure 5.1 et 5.2, illustre la qualité des interprétations de requêtes avec des valeurs variantes de QA. On peut observer clairement que notre approche surpasse nettement l'approche TKQ, en qualité de génération des interprétations de bonnes qualités. Ces résultats sont justifiés par les raisons suivantes :

(i) dans notre approche, les requêtes mots-clés sont non ambigües (avec $QA = 1$) car ces requêtes sont formées uniquement de termes (éléments de la base de connaissance), c'est-à-dire que notre approche génère un nombre très réduit de requête formelles (cela dépend de la densité de l'espace d'interprétation) , par contre, on peut observer que TKQ ne génère pas des interprétations de bonne qualité pour les requêtes ambigües (avec QA très élevées), c'est-à-dire que l'approche TQK génère un nombre très important de requête formelles.

(ii) dans l'approche TKQ, ils préfèrent les interprétations de requêtes populaires (utilisation des scores de popularité pour le classement des requêtes) qui sont bien classées, mais qui peuvent ne pas être celles désirées par l'utilisateur, par contre dans notre approche, avec le

l'introduction du modèle de pondération et l'exploitation de l'historique de recherche, améliore la génération des interprétations de bonne qualité, car dans ce scénario, les requêtes précédentes peuvent être utilisées en tant qu'informations contextuelles pour influencer la génération d'une nouvelle requête.

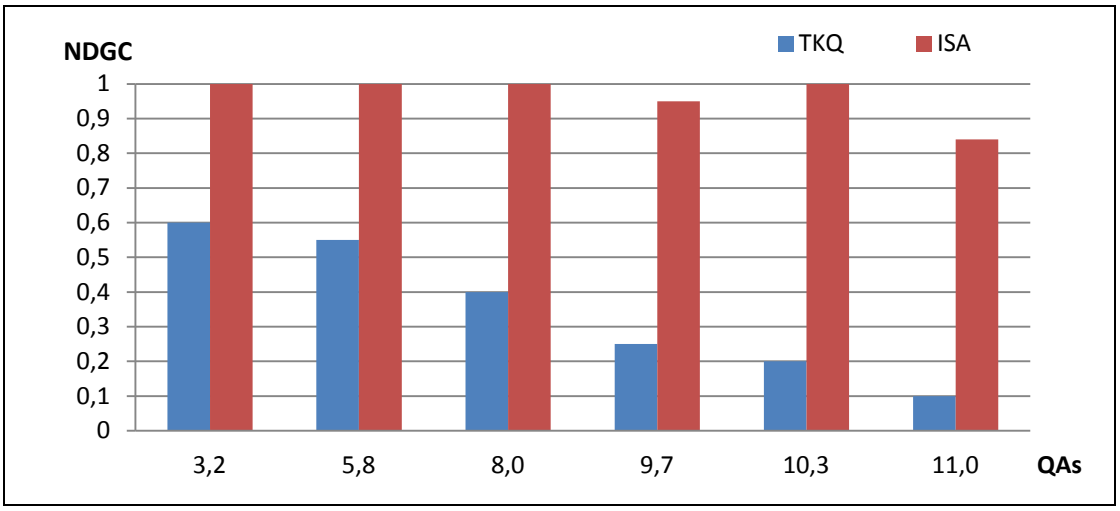


Figure 5.1. NDCG des requêtes pour différentes valeurs de QAs.

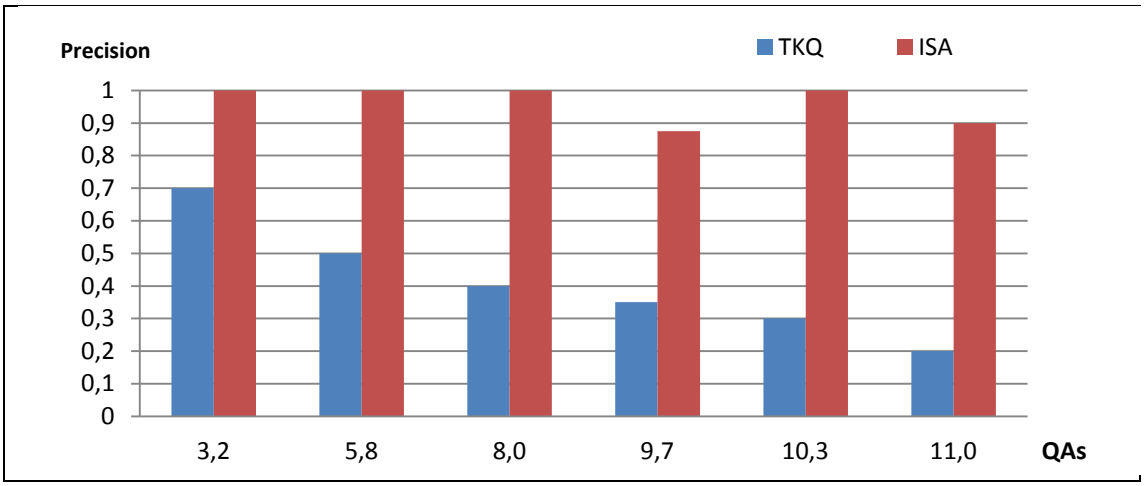


Figure 5.2. Précision des requêtes pour différentes valeurs de QAs.

5.3.5. Evaluation de l'efficacité

A partir des résultats de l'évaluation de l'efficacité de la figure 5.3, nous pouvons constater que notre approche surpasse nettement TKQ en particulier :

(i) lorsque l'ambigüité des requêtes *QA* est très élevée. Donc, le gain en performance est dû essentiellement à la réduction de l'espace d'interprétation dans notre approche par rapport à l'approche TKQ,

(ii) Le gain en performance est dû aussi au temps global de construction de requêtes formelles, qui est réduit dans notre approche, par rapport à l'approche TKQ, car ce temps dans TKQ est composé de : « *temps d'appariement des mots de la requête aux entités sémantiques*

de la base de connaissances », plus le « *temps de construction des k-meilleurs requêtes formelles* », par contre dans notre approche le temps global de construction des requêtes formelles est composé seulement du « *temps de construction des k-meilleurs requêtes formelles* »,

(iii) Le gain en performance est dû aussi en l'absence dans notre approche de l'étape de calcul des combinaisons possibles des éléments-mots-clés et qui est une phase importante dans TKQ.

On peut conclure que l'approche proposée surpasse nettement celles de l'état de l'art en qualité de performance et que la seule caractéristique qui a un impact négatif sur les performances de notre système, c'est la densité de l'espace d'interprétation. Ces résultats sont dus essentiellement à l'introduction dans notre approche de la fonctionnalité d'auto-complétion de requêtes et du modèle de pondération.

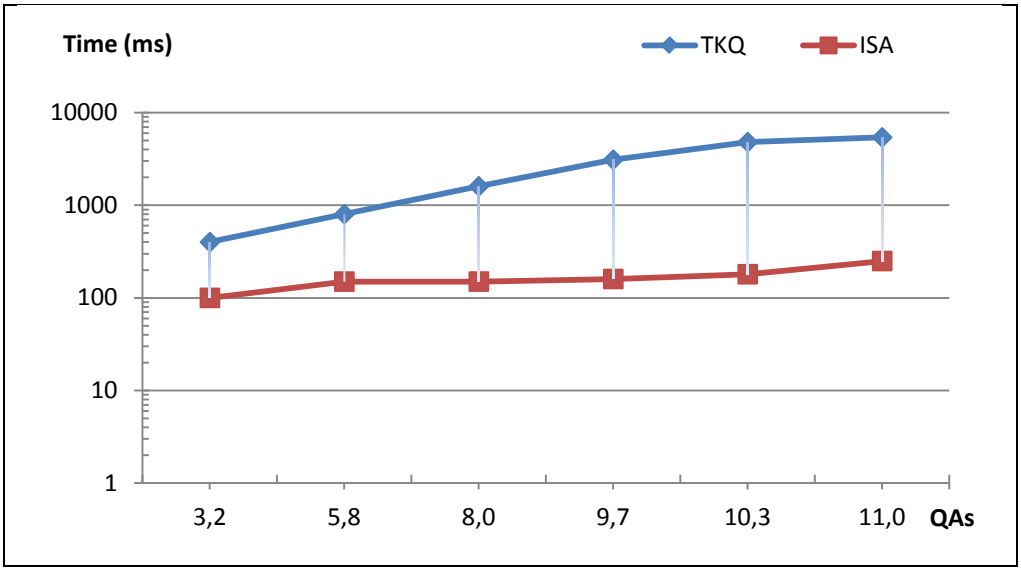


Figure 5.3. Evaluation de l'efficacité pour différentes valeurs de QAs.

5.3.6. L'évaluation de l'auto-complétion de requêtes

Dans cette section, nous présentons les résultats les plus intéressants qui valident expérimentalement l'approche d'auto-complétion de requêtes proposée. Nous avons mené un nombre d'expériences pour tester l'approche avec des préfixes de 2-5 caractères.

Les résultats de la figure 5.4, montre que notre système d'auto-complétion de requêtes génère de bons résultats. Le plus évident, est que l'auto-complétion est considérablement plus efficace avec un préfixe plus long (c'est-à-dire plus spécifique). Ce n'est pas surprenant, étant donné que chaque caractère supplémentaire dans le préfixe réduit considérablement l'espace des suggestions de complétions possibles. Ces résultats sont dus essentiellement à l'exploitation de l'historique de recherche, car dans ce scénario, les requêtes précédentes peuvent être utilisées en tant qu'informations contextuelles pour influencer la génération d'une nouvelle requête.

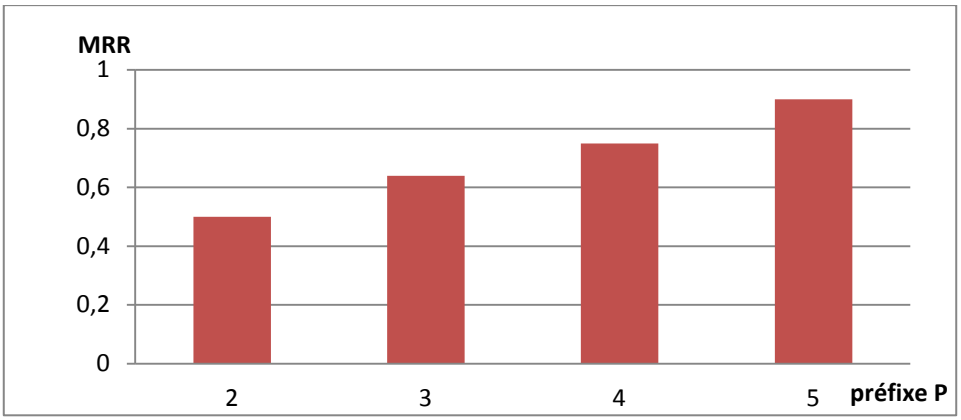


Figure 5.4. Le MRR pour les préfixes de 2-5 caractères.

5.4. Conclusion

Dans ce chapitre, nous avons présenté une implémentation et une expérimentation de notre système nommé « ISA » qui a pour but de valider certains éléments du cadre méthodologique de l'approche proposée. Les expérimentations décrites dans ce chapitre avaient pour objectifs de valider la pertinence de l'idée d'intégrer particulièrement dans ce système les modules d'auto-complétion de requêtes et de pondération des scores et de comparer les performances de notre approche à celle des autres systèmes. Nous avons présenté une évaluation complète de notre approche et nous avons démontré que l'approche proposée surpasse en performance d'autres systèmes. De plus, nous devons souligner que certaines des fonctionnalités de notre système sont pour le moment implémentées de façon relativement simplifiée et en conséquence et compte tenu de l'intérêt que porte cette approche, nous demeurons convaincus que ce système doit être soutenu et poursuivi afin de lui apporter toutes les améliorations nécessaires.

Conclusions Générale et perspectives

Nous avons présenté dans cette thèse le fruit de nos travaux de recherche portant sur : «Interrogation à base d'annotations sémantiques : Construction basée sur le contexte de requêtes formelles à partir des mots-clés ».

L'objectif principal de cette thèse était d'apporter des contributions sur l'exploitation des annotations sémantiques dans la recherche d'information sur le Web, et plus précisément, ce mémoire de thèse détaille l'approche que nous avons proposée dans le but de traduire automatiquement des requêtes mots-clés en requêtes formelles, en se basant sur le contexte de recherche d'un utilisateur. Elle s'appuie sur les théories, les méthodes et les techniques développées dans le domaine du Web Sémantique (WS).

La principale originalité de cette approche est la proposition d'un nouveau processus de construction des requêtes formelle à partir de mots-clés, qui contient deux phases essentielles, à savoir l'auto-complétion de requêtes qui a pour tache principale d'aider l'utilisateur à formuler sans peine sa requête mots-clés en exploitant l'historique de recherche et le modèle de pondération des scores qui est utilisé pour capturer de façon concise les caractéristiques essentielles de l'historique des requêtes d'un utilisateur pour influencer la génération d'une nouvelle requête. Des détails sont donnés sur l'implémentation et une évaluation rigoureuse a montré la pertinence de l'approche.

Le mémoire commence par présenter les principales notions et concepts de la recherche d'information. Nous avons aussi présenté les principaux notions et outils de recherche sur le Web. Enfin un état de lieu sur l'auto-complétion de requêtes. Dans le second chapitre nous avons présenté et défini l'annotation sémantique dans le contexte du Web Sémantique et nous avons vu qu'il existe une gamme assez importante d'outils d'annotation sémantique ayant chacun des caractéristiques propres mais dont le but est de plus en plus orienté vers l'assistance des annotateurs humains à la création des annotations. Ensuite dans le chapitre suivant, nous avons vu que l'utilisation des annotations sémantiques, en recherche d'information est devenue une voie très explorée. Dans cette optique, plusieurs systèmes de recherche d'information sémantique ont été proposés. Dans ce chapitre nous avons présenté un certain nombre de systèmes existants, et nous les avons classé en plusieurs catégories en fonction leurs fonctionnalités. Dans le quatrième chapitre, nous avons formalisé le problème de construction des requêtes formelles et présenté une approche efficace pour le résoudre. Notre système vise à traduire les requêtes mots-clés en requêtes formelles SPARQL en se basant sur le contexte et 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 classiques basés sur les mots clé. Nous avons présentés le modèle de données et de requêtes et l'architecture de notre système. Nous avons aussi discuté plus en détail des aspects spécifiques des données et du traitement des requêtes: prétraitement et indexation. Nous avons introduit et formalisé la fonctionnalité d'auto-complétion de requêtes qui a pour tâche essentielle d'aider l'utilisateur à formuler sans peine sa requête mots-clés constitue uniquement de termes (éléments de la base de connaissances). Nous avons exploité les

informations sur les requêtes précédentes (historique de recherche) pour les utiliser en tant qu'informations contextuelles pour influencer la génération d'une nouvelle requête. Dans le chapitre cinq, nous avons présenté une implémentation et une expérimentation de notre système, qui a pour but de valider certains éléments du cadre méthodologique de l'approche que nous avons proposée. Nous avons présenté une évaluation complète de notre approche et nous avons démontré que l'approche proposée surpasse en performance d'autres systèmes existants.

Les contributions principales de ce mémoire sont :

1. Proposition d'un nouveau processus de construction de requêtes formelles à partir de requêtes mots-clés. Ce processus est constitue des étapes suivantes : (a) Auto-complétion de requêtes, (b) Construction des k-meilleurs graphes requêtes, (c) Mapping des graphes requêtes, (d) Modèle de pondération.
2. Introduction et formalisation la fonctionnalité d'auto-complétion de requêtes
3. Introduction d'un modèle de pondération des scores
4. Exploitation des informations de l'historique de recherche pour les utiliser en tant qu'informations contextuelles pour influencer la génération d'une nouvelle requête.
5. Conception d'un algorithme efficace de construction des graphes requêtes qui étend l'algorithme existant.

Conscients des limites que présentent nos travaux et soucieux de la continuité des travaux qui peut être mise en œuvre. Dans les travaux futurs, nous estimons améliorer notre approche de plusieurs façons :

- Elargir 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.
- Etendre notre approche par l'intégration de différentes sources de données Web hétérogènes et reparti sur le Web pour répondre aux besoins en informations plus complexes.

Il est clair qu'atteindre tous ces objectifs laissés en perspectives nécessitera un travail de longue haleine. Cependant, nous demeurons convaincus, compte tenu de son intérêt, que ce travail doit être soutenu et poursuivi.

Références bibliographiques

[Abrouk et al., 2006] – L. Abrouk, A. Gouaich, Chedy Raïssi, « Annotation automatique de documents », INFORSID'06, Hammamet, Tunisie, pp.483-497, (2006).

[Aditya, et al. 2002] - B. Aditya, G. Bhalotia, S. Chakrabarti, A. Hulgeri, C. Nakhe, Parag, S. Sudarshan, "BANKS: browsing and keyword searching in relational databases", in: Proceedings of, VLDB'02, Morgan Kauffman, China, (2002).

[Aleman-Meza, et al. 2003] - B. Aleman-Meza, C. Halaschek-Wiener, I. B. Arpinar, and A. Sheth, "Context-aware semantic association ranking," in First International Workshop on Semantic Web and Databases, Berlin, Germany, pp. 33–50, (2003).

[Agrawal et al., 2002] - S. Agrawal, S. Chaudhuri, G. Das, "DBXplorer: a system for keyword-based search over relational databases", in: Proceedings of the 18th International Conference on Data Engineering, ICDE'02, IEEE, USA, (2002).

[Amardeilh, 2007] – F. Amardeilh, « Web Sémantique et Informatique Linguistique : propositions méthodologiques et réalisation d'une plateforme logicielle ». Thèse de doctorat. Discipline : Informatique. Université Paris X – Nanterre, Mai 2007.

[Anyanwu et al., 2005] – K. Anyanwu, A. Maduko, A. Sheth, " SemRank: ranking complex relationship search results on the semantic web". In: Proceedings of WWW'2005, New York, NY, USA, ACM Press (2005) 117–127, (2005).

[Athanasis et al., 2004] – 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).

[Baget et al., 2004] – J. F. Baget, E. Canaud, J. Euzenat , M. S. Hacid, « Les langages du Web Sémantique », in Le Web sémantique, Hors série de la Revue Information - Interaction - Intelligence (I3), 4(1), Cépaduès, Toulouse , pp. 21-43, (2004).

[Bamba and Mukherjea, 2004] - B. Bamba, S. Mukherjea, "Utilizing resource importance for ranking semantic web query results", In Semantic Web and Databases, Second Int. Workshop, SWDB 2004, (2004).

[Bangyong et al., 2005] - L. Bangyong, T. Jie, Juanzi, "Association search in semantic web: search + inference", In WWW '05: Special interest tracks and posters of the 14th int. conf. on World Wide Web, (2005).

[Bar-Yossef et Kraus., 2011] - Z. Bar-Yossef, N. Kraus, "Context-sensitive query auto-completion", In WWW '11, pages 107–116, New York, NY, USA ACM, (2011)..

[Belew, 2000] - R. K. Belew, "Finding Out About: A Cognitive Perspective on Search Engine Technology and the WWW". New York: Cambridge University Press, 2000. Review published in Information Retrieval, Vol. 5, Issue 2-3, April-July (2002).

[Bergamaschi et al., 2011] - S. Bergamaschi, E. Domnori, F. Guerra, R. TrilloLado, Y. Velegrakis, "Keyword search over relational databases: a metadata approach", in: Proceedings of SIGMOD'11, ACM, Greece, (2011).

[Berners-Lee, 1998] – T. Berners-Lee, "Weaving the Web", Harper Eds, San Francisco, 226, (1998).

[Berners-Lee et al., 2001] – T. Berners-Lee, J. Hendler, O. Lassila, "The Semantic Web", In Scientific American , p35-43, May 2001.

[Bernstein et al., 2005] – A. Bernstein, E. Kaufmann, A. Gohring,, C. Kiefer, "Querying ontologies: A controlled English interface for end-users", In: Proceedings of ISWC'2005. (2005).

[Bhagdev et al. 2008] – R. Bhagdev, S. Chapman, F. Ciravegna, V. Lanfranchi, D. Petrelli, "Hybrid Search: Effectively Combining Keywords and Semantic Searches"., (2008).

[Bhalotia et al. 2002] – G. Bhalotia, A. Hulgeri, C. Nakhe, S. Chakrabarti, S. Sudarshan, "Keyword searching and browsing in databases using banks", . In ICDE [DBL02], pages 431–440, (2002).

[Bobed & Mena, 2016] - C. Bobed, E. Mena, "QueryGen: Semantic interpretation of keyword queries over heterogeneous information systems", Inf. Sci. 329: 412-433 (2016).

[Bonino et al., 2003] - D. Bonino, F. Corno, L. Farinetti, "DOSE: a distributed open semantic elaboration platform," in ICTAI 2003, California, (2003).

[Bouramoul, 2011] – A. Bouramoul, «Recherche d'information contextuelle et sémantique sur le web », thèse de doctorat en informatique, Université MENTOURI de Constantine, (2011).

[Bourigault et al., 2004] – D. Bourigault, N. Aussenac-Gilles, J. Charlet J., « Construction de ressources terminologiques ou ontologiques à partir de textes : un cadre unificateur pour trois études de cas », Numéro Spécial de la RIA, 18(1), Hermès, Paris, pp. 87-110, (2004).

[Bray et al., 1998] – T. Bray, J. Paoli, et C.M. Sperberg-McQueen, "Extensible Markup Language (XML) 1.0", W3C Recommendation, February. (1998).

[Brin et Page, 1998] - S. Brin et L. Page, « The anatomy of a large-scale hypertextual web search engine", In Proc. of seventh international conference on WW 7, pages 107–117, Australia, (1998).

[Buitelaar et al., 2004] - P. Buitelaar, Th. Eigner, Th. Declerck, "Ontoselect: A dynamic ontology library with support for ontology selection", In, Proc. of the Demo Session at the Int. Semantic Web Conf., (2004).

[Cabrio et al., 2012] – E. Cabrio, J. Cojan, A. P. Aprosio, B. Magnini, A. Lavelli, F. Gandon, « QAKiS: an open domain QA system based on relational patterns", In International Semantic Web Conference (Posters & Demos), volume 914, (2012).

[Cai et al., 2014] – F. Cai, S. Liang, M. de Rijke, « Time-sensitive personalized query auto-completion", In Proc. of the 23rd ACM CIKM '14, pages 1599–1608, New York, USA, (2014)..

[Cai et de Rijke, 2016] - F. Cai, M. de Rijke, « A Survey of Query Auto Completion", in Information Retrieval, Foundations and Trends in Information Retrieval, Vol. 10, No. 4, p.1-92. ISSN 1554-0669, (2016).

[Cai et de Rijke, 2016a] – F. Cai, M. de Rijke, « Learning from homologous queries and semantically related terms for query auto completion", Information processing and Management, 52(4):628–643, (2016).

[Carlos et al., 2006] – A. Carlos, A.P. Hurtado, P.T. Wood, "A Relaxed Approach to RDF Querying", In: Proc. of ISWC'2006. (2006).

[Castells et al. 2007] - P. Castells, M. Fernndez, D. Vallet, « An adaptation of the vector-space model for ontology-based information retrieval", IEEE Trans. Knowl. Data Eng., 19:261–272, (2007).

[Charlet, 2002] – J. Charlet, « L'Ingénierie des connaissances : développements, résultats et perspectives pour la gestion des connaissances médicales », Habilitation à diriger des recherches, Université Paris VI, 2002, 127 p, (2002).

[Choudhury et Phon-Amnuaisuk, 2006] - S. Choudhury, S. Phon-Amnuaisuk, "Semantic retrieval with spreading activation », In Proc. of the MMU Int. Symposium on Information and Communications Technologies M2USIC 2006, (2006).

[Clemmer et Davies, 2011] – A. Clemmer, S. Davies, « Smeagol: A "Specific-to-General" semantic web query interface paradigm for novices". In Database and Expert Systems Applications, page 288–302, (2011).

[Konstantin et al., 2008] - K.onstantin Golenberg, Benny Kimelfeld, and Yehoshua Sagiv. Keyword proximity search in complex data graphs. In Wang [Wan08], pages 927–940.

[Contreras et al., 2004] - J. Contreras, V. R. Benjamins, M. Blzquez, S. Losada, R. Salla, J. Sevilla, D. Navarro, J. Casillas, A. Momp, D. Patn, L. Rodrigo, P. Tena, I. Martos, "International Affairs Portal: A semantic web application," in ECAI Workshop on Application of Semantic Web Technologies to Web Communities, (2004).

[Corby et al., 2004] - O. Corby, R. Dieng-Kuntz, C. Faron-Zucker, "Querying the semantic web with the corese search engine," in Proc. 15th ECAI/PAIS, Valencia, Spain, (2004).

[Corby et al., 2006] - O. Corby, R. Dieng-Kuntz, C. Faron- Zucker, F. Gandon, "Searching the semantic web: Approximate query processing based on ontologies", IEEE Intelligent Systems, 21(1):20–27, (2006).

[Craswell et al., 2009] - N. Craswell, R. Jones, G. Dupret, E. Viegas, editors. WSCD '09: Proceedings 2009 Workshop on Web Search Click Data, 2009.

[Daconta et al., 2003] - M. C. Daconta, L. J. Obrst, K. T. Smith, « The semantic web: a guide to the future of XML, web services, and knowledge management". John Wiley & Sons. (2003).

[Damljanovic et al., 2010] – D. Damljanovic, M. Agatonovic, H. Cunningham, "Natural language interfaces to ontologies: Combining syntactic analysis and ontology-based lookup through the user interaction", In The Semantic Web: Research and Applications, page 106–120. Springer, (2010).

[Daoud, 2009] – M. Daoud, "Accès personnalisé à l'information : approche basée sur l'utilisation d'un profil utilisateur sémantique dérivé d'une ontologie de domaines à travers l'historique des sessions de recherche", thèse de doctorat en informatique, Université Paul Sabatier. (2009).

[dAquin et al., 2007] - M. dAquin, M. Sabou, M. Dzbor, C. Baldassarre, L. Gridinoc, S. Angeletou, E. Motta, « Watson: A gateway for the semantic web", In European Semantic Web Conf., ESWC 2007 (poster), (2007).

[Davidson, 2010] – P. Davidson, "Designing uri sets for the uk public sector", UK Chief Technology Officer Council, (2010).

[Davies et al., 2002] - J. Davies, U. Krohn, R. Weeks, "Quizrdf: search technology for the semantic web", In WWW2002 workshop on RDF & Semantic Web Applications, (2002).

[Desmontils et Jacquin, 2002] - E. Desmontils et C. Jacquin, "Indexing a web site with a terminology oriented ontology", The Emerging Semantic Web, I.F. Cruz, S. Decker, J. Euzenat and D. L. McGuinness Ed, pages 181–197, (2002).

[Di Santo et al., 2015] – G. Di Santo, R. McCreadie, C. Macdonald, I. Ounis, "Comparing approaches for query autocompletion", In Proc. of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '15, pages 775–778, New York, USA, (2015).

[Ding et al., 2004] - L. Ding, T. Finin, A. Joshi, R. Pan, R. Scott Cost, Y. Peng, P. Reddivari, V. C. Doshi, J. Sachs, "Swoogle: A Search and Metadata Engine for the Semantic Web", In Proc. of the 13thACM Conf. on Information and Knowledge Management, (2004).

[Dumais, 1994] - S. Dumais, "Latent Semantic Indexing (LSI) ", in Proc. of TREC-3, (1994).

[Elbassuoni et al., 2010] – S. Elbassuoni, M. Ramanath, R. Schenkel, G. Weikum, "Searching rdf graphs with SPARQL and keywords", IEEE Data Engineering Bulletin, 33(1), (2010).

[Esmaili et Abolhassani, 2006] - K. S. Esmaili et H. Abolhassani, « A categorization scheme for semantic web search engines". In 4th ACS/IEEE Int. Conf. on Computer Systems and Applications (AICCSA-06), (2006).

[Fensel et al., 2003] – D. Fensel, et al eds. 2003, "Spinning the Semantic Web:Bringing the World Wide Web to Its Full Potential". Cambridge, Mass.: MIT Press. (2003).

[Fu et al., 2011] - Fu, K. Anyanwu," Effectively interpreting keyword queries on RDF databases with a rear view", in: Proc. of the 10th International Semantic Web Conference, ISWC'11, Springer, Germany, (2011).

[Gruber, 1993] – T. Gruber, "A translation approach to portable ontology specifications", in Knowledge Acquisition Journal, 5(2), Academic Press, 1993, pp. 199-220, (1993).

[Guha et al., 2003] - R. Guha, R. McCool, E. Miller, "Semantic search", in WWW '03: Proc. of the 12th int. conf. on World Wide Web, (2003).

[Halaschek et al. 2004] - C. Halaschek, B. Aleman-Meza, I. Arpinar, A. Sheth, "Discovering and ranking semantic associations over a large RDF metabase," in 30th International Conference on Very Large Data Bases(VLDB), Toronto, Canada, (2004).

[Handschuh , 2005] – S. Handschuh, "Creating Ontology-based Metadata by Annotation for the Semantic Web", Thèse de doctorat, Université de Karlsruhe, 225 p, (2005).

[Harris et Seaborne, 2013]- S. Harris, et A. Seaborne, "SPARQL 1.1 Query Language", W3C Recommendation, W3C. (2013).

[He et al., 2007] - H. He, H. Wang, J. Yang, et P. S. Yu, "Blinks: ranked keyword searches on graphs," in SIGMOD Conference, pp. 305–316, (2007).

[Hernandez, 2005] – N. Hernandez, « Ontologies de domaine pour la modélisation du contexte en Recherche d'information », Thèse de doctorat, Université Paul Sabaltier de Toulouse, (2005).

[Hristidis et al., 2002] - V. Hristidis, Y. Papakonstantinou, "DISCOVER: keyword search in relational databases", in Proc.of 28th International Conference on Very Large DataBases, VLDB'02, China, (2002).

[Hsu et Ottaviano, 2013] – P. Hsu et G. Ottaviano, "Space-efficient data structures for top-k completion". In Proc. of the 22nd International World Wide Web Conference, WWW '13, pages 583–594, New York, USA, (2013).

[Hulgeri et al., 2001] – A. Hulgeri, G. Bhalotia, C. Nakhe, S. Chakrabarti, et S. Sudarshan, « Keeyword search in databases", IEEE Data Eng. Bull., 24(3):22–32, (2001).

[Jean et al., 2006] – J.S. Ait-Ameur, Y. Pierra , "Querying Ontology Based Database Using OntoQL (an Ontology Query Language)", in Proc. of On the Move to Meaningful Internet Systems (2006).

[Jiang and Tan, 2006] - X. Jiang et A.H. Tan, "Ontosearch: A full-text search engine for the semantic web", in Proc. of the 21st National Conf. on Artificial Intelligence and the 18th Innovative Applications of Artificial Intelligence Conf., (2006).

[Jiang et al., 2014] – J.Y. Jiang, Y.Y. Ke, P.Y. Chien, et P.J. Cheng, "Learning user reformulation behavior for query auto-completion", in Proc. of the 37th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '14, pages 445–454, New York, USA, (2014).

[Kacholia et al., 2005] – V. Kacholia, S. Pandit, S. Chakrabarti, S. Sudarshan, R. Desai, et H. Karambelkar, « Bidirectional expansion for keyword search on graph databases", VLDB, pages 505–516. ACM, (2005).

[Kahan et al., 2001] – J. Kahan, M.R. Koivunen, E. Prud'hommeaux, et R. Swick, "Annotea: An Open RDF Infrastructure for Shared Web Annotations", in Proc. of the 10th International World Wide Web Conference (WWW'01), ACM Press, Hong-Kong, pp. 623-632, (2001).

[Kandogan et al., 2006] – E. Kandogan, R. Krishnamurthy, S. Raghavan, S. Vaithyanathan, H. Zhu, " Avatar Semantic Search: A Database Approach to Information Retrieval", in: Proc. of SIGMOD'06, New York, USA, ACM Press 790–792, (2006).

[Karvounarakis et al., 2002] – G. Karvounarakis, S. Alexaki, V. Christophides, D. Plexousakis, M. Scholl, "RQL: A Declarative Query Language for RDF", in Proc.of WWW'02, New York, USA, ACM Press 592–603, (2002).

[Kastrinakis et Tzitzikas, 2010] – D. Kastrinakis et Y. Tzitzikas, "Advancing search query autocompletion services with more and better suggestions", in Proc. of the 10th International Conference on Web Engineering, ICWE'10, pages 35–49, Berlin, Heidelberg, (2010).

[Kaufmann et al., 2006] – E. Kaufmann, A. Bernstein, et R. Zumstein, "Querix: A natural language interface to query ontologies based on clarification dialogs". In 5th ISWC, (2006).

[Kaufmann et al., 2007] – E. Kaufmann, A. Bernstein, et L. Fischer, "NLP-Reduce: a "naıve" but domain-independent natural language interface for querying ontologies", (2007).

[Kaufmann et Bernstein, 2010] – E. Kaufmann et A. Bernstein, "Evaluating the usability of natural language query languages and interfaces to semantic web knowledge bases", Web Semantics: Science, Services and Agents on the World Wide Web, 8(4):377–393, (2010).

[Khelif, 2006] - M.K. Khelif. "Web sémantique et mémoire d'expériences pour l'analyse du transcriptome". Thèse de doctorat en informatique, Université de Nice-Sophia Antipolis-UFR sciences, pages 7-16, Avril (2006).

[Kiryakov et al., 2004] - A. Kiryakov, B. Popov, I. Terziev, D. Manov, et D. Ognyanoff, "Semantic annotation, indexing, and retrieval", Journal of Web Semantics: Science, Services and Agents on the WWW, 2:49–79, (2004).

[Kleinberg, 1999] J. M. Kleinberg, "Authoritative sources in a hyperlinked environment". J. ACM, 46(5):604–632, (1999).

[Koutrika et al., 2010] – G. Koutrika, Z. Mohammadi Zadeh, et H. GarciaMolina, "Data clouds: summarizing keyword search results over structured data" in Proc. of the 12th International Conference on Extending Database Technology: Advances in Database Technology, EDBT '09, pages 391–402, New York, USA, (2009).

[Lassila et Swick, 1999] - O. Lassila, et R. Swick, "Resource Description Framework (RDF) Model and Syntax Specification", W3C Recommendation, (1999).

[Latreche et al., 2009] – A. Latreche, A. Lehireche, K. Benyahia, "Interrogation à base d'annotation sémantique", CIIA'09 - Conférence Internationale sur l'Informatique et ses Applications, SAIDA, (2009).

[Latreche et al., 2017] – A. Latreche, A. Lehireche, K. Benyahia , "Interrogation Based on Semantic Annotations: Context-Based Construction of Formal Queries from Keywords", International Journal of Web Portals (IJWP), Volume 9, Issue 2 (2017).pages : 47-67.

[Le et al., 2014] - W. Le, F. Li, A. Kementsietsidis, S. Duan, Scalable keyword search on large RDF data, IEEE Trans. Knowl. Data Eng. 26(11)(2014)2774–2788.

[Lehmann and Bühmann, 2011] – J. Lehmann et L. Bühmann, "AutoSPARQL: let users query your knowledge base", in The Semantic Web: Research and Applications, page 63–79. Springer, 2011.

[Lei et al., 2006] – Y. Lei, V. Uren, et E. Motta, "Semsearch: A search engine for the semantic web", in Managing Knowledge in a World of Networks, page 238–245. Springer, 2006.

[Li et al., 2007] - Li, Y., Wang, Y., Huang, X, "A Relation-Based Search Engine in Semantic Web", . Proc. of IEEE Transactions on Knowledge and Data Engineering 19(2) 273–282, (2007).

[Li et al., 2014] - Yanen Li, Anlei Dong, Hongning Wang, Hongbo Deng, Yi Chang, and ChengXiang Zhai. A two-dimensional click model for query auto-completion. In Proceedings of the 37th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '14, pages 455–464, New York, NY, USA, 2014. ACM.

[Li et al., 2015] – L. Li, H. Deng, A. Dong, Y. Chang, H. Zha, et R. Baeza-Yates, "Analyzing user's sequential behavior in query auto-completion via Markov processes", in Proc. of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '15, pages 123–132, New York, USA, 2015.

[Linckels et Meinel, 2005] – S. Linckels et C. Meinel, "A simple solution for an intelligent librarian system", I IADIS AC, page 495–503, (2005).

[Liu et al. , 2006] - F. Liu, C. T. Yu, W. Meng, and A. Chowdhury, "Effective keyword search in relational databases," in SIGMOD Conference, pp. 563–574, (2006).

[Luo et al., 2007] – Y. Luo, X. Lin, W. Wang et X. Zhou, "Spark: topk keyword query in relational databases", in Chan et al. (COZ07), pages 115–126, (2007).

[Luong, 2007] – P. H. Luong, "Gestion de l'évolution d'un web sémantique d'entreprise ». Thèse de doctorat de l'Ecole des Mines de Paris, (2007).

[Mayfield & Finin, 2005] - T. Finin, J. Mayfield, C. Fink, A. Joshi, et R. S. Cost, "Information retrieval and the semantic web," in Proc. of the 38th International Conference on System Sciences, Hawaii, United States of America, (2005).

[Miller et al., 2002] – L. Miller, A. Seaborne, A. Reggiori, "Three Implementations of SquishQL, a Simple RDF Query Language", in Proc. of ISWC'2002. (2002).

[Mitra et al., 2014] – B. Mitra, M. Shokouhi, F. Radlinski, et K. Hofmann, "On user interactions with query auto-completion", in Proc. of the 37th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '14, pages 1055–1058, New York, USA, (2014).

[Mitra, 2015] – B. Mitra, "Exploring session context using distributed representations of queries and reformulations", in Proc. of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '15, pages 3–12, New York, USA, (2015).

[Mitra et Craswell, 2015] - B. Mitra, et N. Craswell, "Query auto-Completion for rare prefixes", in 24th ACM Conference on Information and Knowledge Management , CIKM '15, pages 1755-1758, (2015).

[Miyanishi et Sakai, 2013] - T. Miyanishi et T. Sakai, "Time-aware structured query suggestion", in In SIGIR, (2013).

[Mothe , 1994] - J. Mothe, "Modèle Connexionniste pour la Recherche d'Information, Expansion dirigée de requêtes et apprentissage», Thèse de Doctorat de l'Université Paul Sabatier, Toulouse France, (1994).

[Pan et al., 2006] - J. Z. Pan, E. Thomas, et D. Sleeman, "Ontosearch2: Searching and querying web ontologies", in Proc. of the IADIS Int. Conf. WWW/Internet (2006).

[Pass et al., 2006] - G. Pass, A. Chowdhury, et C. Torgeson, "A picture of search", in InfoScale '06, pages 1–7, New York, USA, (2006. ACM).

[Prié et Garlatti, 2004] – Y. Prié, et S. Garlatti, "Méta-données et annotations dans le Web sémantique », Hors Série 2004 - Web Sémantique. Revue en Sciences du Traitement de l'Information, I3 (Information - Interaction – Intelligence), Cépaduès, Toulouse, pp. 45-68. (2004).

[Priebe et al., 2004] - T. Priebe, Ch. Schlager, et G. Pernul, "A search engine for rdf metadata", in DEXA '04: Proc. Of the Database and Expert Systems Applications, 15th Int. Workshop on (DEXA'04), (2004).

[Prud'hommeaux et Seaborne, 2007] – E. Prud'hommeaux et A. Seaborne, "SPARQL Query Language for RDF", W3C Candidate Recommendation 14 June 2007.

[Rocha et al., 2004] - C. Rocha, D. Schwabe, et M. Poggi de Arag˜ao, "A hybrid approach for searching in the semantic web", in Proc. of the 13th int. conf. on World Wide Web (WWW), (2004).

[Royo et al., 2005] – J. Royo, E. Mena, J. Bernard, A. Illarramendi, " Searching the web: From keywords to semantic queries", in Proc. of the Third International Conference on Information Technology and Applications (ICITA'05), IEEE Computer Society 244–249, (2005).

[Russell et Smart, 2008] – A. Russell et P. Smart, "Nitelight: A graphical editor for sparql queries", (2008).

[Salton, 1970] - G. Salton, "The SMART retrieval system : Experiments in automatic document processing", Prentice Hall, (1970).

[Salton et al., 1983] – G. Salton, E.A. Fox, et H. Wu, "Extended Boolean information retrieval system", CACM 26(11), pp. 1022-1036, (1983).

[Salton et Mcgill, 1984] - G. Salton et J. Mcgill, "Introduction to modern information retrieval", Mcgraw Hill, New York (1984).

[Sheth et al. 2002] - A. Sheth, C. Bertram, D. Avant, B. Hammond, K. Kochut, et Y. Warke, "Managing semantic content for the web", IEEE Internet Computing, 6:80–87, (2002).

[Shokouhi et Radinsky, 2012] – M. Shokouhi et K. Radinsky, "Time-sensitive query auto-completion", in Proc. of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '12, pages 601–610, New York, USA, 2012.

[Shokouhi, 2013] – M. Shokouhi, "Learning to personalize query auto-completion", in Proc. of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '13, pages 103–112, New York, USA, (2013).

[Smits et al., 2014] G. Smits, O. Pivert, H. Jaudoin, et F. Paulus, "AGGREGO SEARCH: Interactive keyword query construction (demo)", in Int. Conf. Extending Database Technology, pages 636-639, (2014).

[Song et al. 2005] - J. Song, W. Zhang, W. Xiao, G. Li, et Z. Xu, "Ontology-based information retrieval model for the semantic web", in IEEE Int. Conf. on e-Technology, e-Commerce, and e-Services (EEE), (2005).

[Song et al. 2011] – Y. Song, D. Zhou, et L.W. He, "Post-ranking query suggestion by diversifying search results", in Proc. of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '11, pages 815–824, New York, USA, (2011).

[Stojanovic et al., 2001] - N. Stojanovic, A. Maedche, S. Staab, R. Studer, et Y. Sure, "Seal: a framework for developing semantic portals", in Proc. of the 1st Int. Conf. on Knowledge Capture (K-CAP), (2001).

[Stojanovic et al., 2003] – N. Stojanovic, J. Gonzalez, L. Stojanovic, "Ontologer: A System for Usage-Driven Management of Ontology-Based Information Portals", in Proc. of L-CAP'2003. (2003).

[Stojanovic et al., 2004] – N. Stojanovic, L. Stojanovic, "A Logic-based Approach for Query Refinement in Ontology-based Information Retrieval Systems", in Proc. of the 16th IEEE Int. Conf. on Tools with Artifical Intelligence. (2004).

[Studer et al., 1998] – R. Studer, V.R. Benjamins et D. Fensel ,"Knowledge engineering: principles and methods", in IEEE Transactions on Data and Knowledge Engineering, 25(1&2), pp.161-197, (1998).

[Teng et Zhu, 2015] - M. Teng, G. Zhu, "Interactive search over Web scale RDF data using predicates as constraints", J.Intell. Inf. Syst. 44(3) 381–395, (2015).

[Team, 2007] – C. Team. CoGui, 2007. URL http://www.lirmm.fr/cogui/.

[Tran et al., 2007] - T. Tran, P. Cimiano, S. Rudolph, and R. Studer, "Ontology-based interpretation of keywords for semantic search," in ISWC/ASWC, pp. 523–536, (2007).

[Tran et al., 2011] - T. Tran, D.M. Herzig, G. Ladwig, "SemSearchPro: using semantics throughout the search process", Web Semant. :Sci. Serv. Agents World Wide Web9(4) 349–364, (2011).

[Tran et al., 2009] - T. Tran, H. Wang, S. Rudolph, and P. Cimiano, "Top-k exploration of query candidates for efficient keyword search on graph-shaped (rdf) data", in Data Engineering, ICDE'09. IEEE 25th International Conference on, page 405–416, (2009).

[Unger et al., 2012] – C. Unger, L. Bühmann, J. Lehmann, Axel-Cyrille Ngonga Ngomo, D. Gerber, et P. Cimiano, "Template-based question answering over RDF data", in Proc. of the 21st international conference on World Wide Web, page 639–648, (2012).

[Uren et al, 2006] – V. Uren, P. Cimiano, S. Handschuh , M. Vargas-Vera ,E. Motta E. et F. Ciravegna, "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, pp.14-26, (2006).

[Vallet et al 2005] – D. Vallet, M. Fernández, et P. Castells, "An Ontology-Based Information Retrieval Model", (2005).

[Whiting et Jose, 2014] – S. Whiting et J.M. Jose, "Recent and robust query auto-completion", in Proc. of the 23rd International World Wide Web Conference, WWW '14, pages 971–982, New York, USA, (2014).

[Wang et al., 2008] - H. Wang, K. Zhang, Q. Liu, T. Tran, et Y. Yu, "Q2semantic: A lightweight keyword interface to semantic search," in ESWC, pp. 584–598, (2008).

[Wang et al., 2009] – H. Wang, Q. Liu, T. Penin, L. Fu, L. Zhang, T. Tran, Y. Yu, et Y. Pan, « Semplore: A scalable ir approach to search the web of data", . J. Web Sem., 7(3):177–188, (2009).

[Xiao et al., 2013] - C. Xiao, J. Qin, W. Wang, Y. Ishikawa, K. Tsuda, et K. Sadakane, "Efficient error-tolerant query autocompletion », VLDB, 6(6), (2013).

[Zhang et al. 2005] - L. Zhang, Y. Yu, J. Zhou, Ch. Lin, et Y. Yang, "An enhanced model for searching in semantic portals", in WWW '05: Proc. of the 14th int. conf. on World Wide Web, (2005).

[Zhang et al. 2015] - A. Zhang, A. Goyal,W. Kong, H. Deng, A. Dong, Y. Chang, C. A. Gunter, et J. Han, "adaqac: Adaptive query auto-completion via implicit negative feedback", in SIGIR '15, pages 143–152, New York, USA, (2015).

[Zhu et al., 2002] - H. Zhu, J. Zhong, J. Li et Yong Yu ,"An Approach for Semantic Search by Matching RDF Graphs", (2002).

[Zhou et al., 2007] – Q. Zhou, C. Wang, M. Xiong, H. Wang, et Y. Yu, « SPARK: adapting keyword query to semantic search", in The Semantic Web, page 694–707. Springer, (2007).

[Zenz et al., 2009] - Zenz, G., et al., "From keywords to semantic queries incremental query construction on semantic web", Web Semantics: Science, Services and Agents on the World Wide Web, 2009. 7: p. 166-176. (2009).

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. En principe, la recherche sémantique promet de fournir des résultats plus précis que la traditionnelle recherche par mots-clés, mais sa progression a été retardée en raison de la complexité de ses langages de requêtes. 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 construit automatiquement des requêtes formelles structurées à partir de requêtes mots-clés.

Nous proposons un nouveau processus où nous allons introduire une nouvelle fonctionnalité d'auto-complétion de requêtes basée sur le contexte, afin d'aider l'utilisateur à formuler sa requête mots-clés, en suggérant des complétions de requêtes à partir de préfixe entré par l'utilisateur. Nous abordons aussi le problème de la génération basée sur le contexte des requêtes formelles structurées en utilisant l'historique des requêtes des utilisateurs, où les informations sur les requêtes précédentes peuvent être utilisées en tant qu'informations contextuelles pour influencer la génération d'une nouvelle requête.

Avec les premiers tests, notre approche réalise des résultats encourageants.

Keywords: Auto-Complétion, Recherche par mot-clé, Contexte de requête, RDF Graphes RDF, Annotation sémantique, Recherche sémantique.

Abstract

Traditional information search approaches do not explicitly capture the meaning of a keyword query, but provide a comfortable way for the user to express his or her information needs based on the keywords. In principle, semantic search aims to produce better results than traditional keyword search, but its progression has been retarded because of to the complexity of its query languages. In this document, we present an approach to adapt keyword queries to querying the semantic web based on semantic annotations: the approach automatically construct structured formal queries from keywords queries.

We propose a new process where we introduce a novel context-based query auto-completion feature to help the users to construct their keywords query by suggesting queries given prefixes. We also address the problem of context-based generating formal queries by exploiting user's query history, where previous queries can be used as contextual information for generating a new query.

With the first tests, our approach achieved encouraging results.

Keywords: Auto-Completion, Keyword Search, Query Context, RDF Graphs, Semantic Annotation, Semantic Search

ملخص

إن المانهج الحالية لاسترجاع المعلومات لا تستحوذ رسميا على المعنى الصريح لاستعلام الكلمات المفتاحية ، ولكنها توفر مسارا مريحا للمستخدم الذي يحدد متطلباته للمعلومات استنادا إلى الكلمات المفتاحية. من حيث المبدأ، البحث الدلالي يعد بتقديم نتائج أكثر دقة من البحث التقليدي عن طريق الكلمات المفتاحية ، ولكن تطورها قد تأخر بسبب تعقيد لغات الاستعلام .في هذه الوثيقة ، نستكشف نهجا جديدا لتكييف استعلامات الكلمات المفتاحية لتكون قادرة على الاستعلام عن الويب الدلالي استنادا إلى الشرح الدلالي: يقوم النهج تلقائيا ببناء استعلامات رسمية منظمة من الكلمات المفتاحية. نقترح عملية جديدة حيث سنقدم ميزة جديدة للإكمال التلقائي لاستعلام استنادا إلى استعمال السياق، لمساعدة المستخدم في صياغة استعلام الكلمات المفتاحية. كما نتعامل مع مشكلة إنشاء استعلامات رسمية منظمة استنادا إلى السياق، وذلك باستخدام سجل طلبات بحث المستخدم كمعلومات السياقية للتأثير على انتاج استعلام جديد.

مع الاختبارات ألأولى, نهجنا يحقق نتائج مشجعة.

الكلمات المفتاحية : الإكمال التلقائي، سياق استعلام، الشرح الدلالي، البحث الدلالي، رسم بياني RDF