Passage de la langue naturelle à une requête SPARQL dans le système SWIP

Camille Pradel, Ollivier Haemmerlé, Nathalie Jane Hernandez

Passage de la langue naturelle à une requête SPARQL dans le système SWIP

Camille Pradel, Ollivier Haemmerlé, and Nathalie Hernandez

IRIT, Université de Toulouse le Mirail,
Département de Mathématiques-Informatique,
5 allées Antonio Machado, F-31058 Toulouse Cedex
pradel@irit.fr

Résumé : Notre objectif est de fournir aux utilisateurs un moyen d'interroger des bases de connaissances en utilisant des requêtes exprimées en langue naturelle. Nous souhaitons masquer la complexité liée à la formulation des requêtes dans un langage de requêtes graphes comme SPARQL. L'originalité principale de notre approche réside dans l'utilisation de patrons de requêtes. Dans cet article, nous justifions le postulat selon lequel les requêtes issues d'utilisateurs de la "vraie vie" sont des variations autour de quelques familles typiques de requêtes. Nous expliquons également comment notre approche est adaptable à différentes langues. Les premières évaluations sur le jeu de données du challenge QALD-2 montrent la pertinence de notre approche.

Mots-clés : requête langue naturelle, patrons de requêtes, SPARQL

1 Introduction

De plus en plus d'utilisateurs ont besoin d'accéder à la gigantesque masse de données disponible par le biais d'internet. Les bases de données sont cachées grâce à des formulaires qui masquent en général des requêtes SQL. Avec le développement des entrepôts de triplets RDF, un besoin d'interfaçage des moteurs SPARQL a émergé, puisqu'il est impossible pour un utilisateur final de maîtriser la complexité des schémas de ces connaissances : pour pouvoir exprimer une requête valide sur les connaissances du web sémantique, l'utilisateur doit connaître le langage SPARQL, les ontologies utilisées pour exprimer les triplets qu'il veut interroger ainsi que la "forme" des graphes RDF considérés. Plusieurs travaux concernant la génération de graphes requêtes ont été menés – dans des langages for-

mels comme SPARQL ou les graphes conceptuels, par exemple – à partir de requêtes mots-clés. Nous pensons que l'existence de programmes de reconnaissance vocale qui comprennent de mieux en mieux le langage parlé, particulièrement sur les "smartphones", impose désormais de travailler sur la reconnaissance de requêtes exprimées en langue naturelle.

Notre travail se place dans ce champ de recherche : comment pouvons-nous interpréter une requête en langue naturelle et la traduire en SPARQL ? Notre approche se distingue des approches existantes en ce que nous proposons d'améliorer la pertinence des requêtes générées et l'efficacité de la méthode en nous fondant sur l'utilisation de patrons de requêtes qui représentent des familles de requêtes. L'utilisation de patrons de requêtes nous affranchit de l'exploration de l'ontologie afin de lier les entités sémantiques identifiées dans la requête puisque les relations potentiellement pertinentes sont déjà exprimées dans les patrons.

Dans [6, 7], nous avons proposé un moyen de construire des requêtes exprimées en graphes conceptuels à partir d'une requête utilisateur exprimée sous forme de mots-clés. Dans [16], nous avons étendu le système afin qu'il prenne en considération les relations exprimées par l'utilisateur entre les mots-clés utilisés dans sa requête. Dans [17], nous avons adapté notre système aux langages du Web sémantique en lieu et place des graphes conceptuels. Dans [18], nous avons présenté de manière précise la notion de patron de requêtes.

Cet article présente le système SWIP (Semantic Web Interface with Patterns), qui prend en entrée une requête exprimée en langue naturelle et propose des requêtes SPARQL classées par ordre de pertinence en sortie. La section 2 présente les systèmes existants partageant notre objectif. La section 3 présente une vue d'ensemble du système SWIP. La section 4 introduit le langage pivot, qui est un formalisme intermédiaire entre la requête langue naturelle et la requête SPARQL. La section 5 présente la notion de patron de requête. La section 6 présente l'implémentation du système SWIP et les résultats de l'évaluation. Nous concluons en section 7.

2 Travaux connexes

2.1 Utilisabilité

La pertinence des interfaces en langue naturelle a été considérée plusieurs fois dans la littérature, sans que l'on parvienne à une conclusion claire [3, 8, 23]. Dans [12], les auteurs évaluent l'utilisabilité des interfaces permettant aux utilisateurs d'accéder à des connaissances par le biais

de la langue naturelle. Ils identifient trois problèmes principaux dans ces interfaces :

- les *ambiguïtés* de la LN sont le premier problème, de loin le plus évident ; il est dû à la variabilité linguistique, i.e. les différentes façons d'exprimer une même requête ;
- la *barrière adaptative* fait que les interfaces ayant de bonnes performances de recherche ne sont pas très portables ; ces systèmes sont la plupart du temps domaine-dépendants et donc difficiles à adapter à de nouveaux contextes ;
- le *problème de l'habitabilité* selon lequel les utilisateurs peuvent exprimer des requêtes au-delà des capacités du système quand on laisse trop de liberté dans l'expression des requêtes.

La contribution principale de [12] est inspirée par ce dernier problème : l'*hypothèse d'habitabilité* établit qu'une interface LN, pour présenter la meilleure performance d'utilisabilité, doit imposer des structures à l'utilisateur afin de l'aider durant le processus de formulation de la requête. Afin de soutenir cette hypothèse, les auteurs décrivent l'étude d'utilisabilité qu'ils ont conduite auprès de vrais utilisateurs. Pour ce faire, ils ont développé quatre interfaces de requête et les ont confrontées à 48 utilisateurs.

Les auteurs brisent la dichotomie classique établie entre les approches complètement LN et les approches formelles. Ils proposent de situer chaque système sur un *continuum* en fonction de la nature de leurs interface avec les utilisateurs. Les approches complètement LN sont à une extrémité de ce continuum, tandis que les langages formels de requête sont à l'autre extrémité. De fait, conformément à l'hypothèse d'habitabilité, l'interface la plus efficace concernant le point de vue utilisabilité devrait être située quelque part au milieu du continuum. Les systèmes développés par les auteurs dans le cadre de leur étude d'utilisabilité sont naturellement positionnés sur ce continuum : (i) *NLP-Reduce* est une interface en LN naïve, domaine-indépendante, dans laquelle les requêtes peuvent être exprimées sous forme de mots-clés, de fragments de phrases ou de phrases complètes en anglais ; (ii) *Querix* est une interface LN domaine-indépendante dans laquelle les requêtes peuvent être exprimées sous la forme de questions en anglais commençant par "Which…", "What…", "How many…", "How much…", "Give me…", ou "Does…" ; (iii) *Ginseng*est un moteur de recherche LN qui prend en entrée une phrase respectant une grammaire donnée ; (iv) *Semantic Cristal* est un outil graphique qui aide les utilisateurs à construire des requêtes formelles.

Les résultats de l'étude montrent une préférence claire des utilisateurs pour Querix et son langage de requête, ce qui tend à confirmer l'hypothèse de l'habitabilité. Pourtant, il est important de remarquer que ces retours utilisateurs et leur perception de la justesse des réponses ne représente pas la réelle justesse des systèmes analysés. De fait, NLP-Reduce obtient de meilleures performances sur des critères objectifs majeurs (temps moyen pour requêter et obtenir des réponses). Certaines interactions complémentaires avec l'utilisateur comme les pratique Querix (désambiguïsation explicite de mots, retours en LN) semblent créer une certaine confiance. Malgré ces résultats, nous considérons clairement qu'un prochain verrou scientifique sera de générer des requêtes formelles à partir de requêtes LN exprimées oralement au travers de terminaux mobiles. Les utilisateurs ont de plus en plus l'habitude de parler à leurs terminaux comme s'ils posaient une question à un ami...

2.2 Approches existantes

Dans cette section, nous présentons des approches existantes visant à formuler des graphes requêtes. Pour cela, nous collons à la classification proposée dans [12], en les présentant selon le continuum, du plus formel au plus permissif, en commençant par les langage de requêtes eux-mêmes.

A l'extrémité de ce continuum se trouvent les langages formels de requêtes graphes comme SPARQL [1]. Ils sont la cible des systèmes que nous présentons dans la section. De tels langages présentent d'évidentes contraintes d'utilisabilité ce qui les rend totalement inadaptés aux utilisateurs finals. Exprimer des requêtes formelles implique de connaître et de respecter la syntaxe du langage utilisé, de comprendre un modèle de graphes et de connaître le schéma de la base interrogée. [9] vise à étendre le langage SPARQL et son mécanisme d'interrogation afin de prendre en compte des mots-clés et des jokers quand l'utilisateur ne connaît pas exactement le schéma sur lequel il veut requêter ; une telle approche nécessite que l'utilisateur connaisse le langage SPARQL.

Très proches sont les approches qui assistent l'utilisateur pendant la formulation de la requête dans de tels langages. Des interfaces comme Flint [2] et SparQLed [3] implémentent de simples fonctionnalités comme la coloration syntaxique et l'autocomplétion. D'autres approches reposent

1. http ://www.w3.org/TR/rdf-sparql-query/
2. http ://openuplabs.tso.co.uk/demos/sparqleditor
3. http ://sindicetech.com/sindice-suite/sparqled/

sur des interfaces graphiques comme [1] pour les requêtes RQL, [19, 4] pour SPARQL ou [5] pour les graphes conceptuels. Même si ces interfaces graphiques sont utiles et rendent la formulation de requêtes bien moins laborieuse, il ne permettent pas de dépasser les limites d'utilisabilité des langages de requête graphes formels.

Sewelis [10] introduit la *recherche par facette à base de requêtes*, un paradigme combinant la recherche par facette et le requêtage, les deux paradigmes les plus populaires dans la recherche sémantique.

D'autres travaux cherchent à générer automatiquement – ou semi-automatiquement – des requêtes formelles à partir de requêtes exprimées sous forme de mots-clés ou de phrases en LN. Notre travail se situe dans cette famille d'approches. 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. Certains travaux proposent d'exprimer des requêtes formelles en différents langages comme SeREQL [14], SPARQL [25, 24, Cabrio *et al.*] ou les graphes conceptuels [6]. Dans ces systèmes, la génération de requêtes nécessite les étapes suivantes : (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, (iv) sélection de la bonne requête par l'utilisateur.

Autosparql [13] étend la catégorie précédente : après 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.

3 Vue générale du système

Dans le système SWIP, le processus conduisant à l'interprétation de la requête se compose de deux étapes principales illustrées par la Figure 1(a). La première étape, l'*interprétation de la requête en langue naturelle*, consiste à identifier les entités nommées, analyser les dépendances syntaxiques et traduire le graphe de dépendances ainsi obtenu en une nouvelle requête, appelée requête pivot. Une requête pivot représente explicitement les relations extraites entre les sous-chaînes de la requête en LN.

Dans la seconde étape, la *formalisation de la requête pivot*, des patrons de requête prédéfinis sont mis en correspondance avec la requête pivot ; nous obtenons alors une liste d'interprétations potentielles de la requête

utilisateur, qui sont ensuite classés conformément à leur pertinence estimée, et finalement proposées à l'utilisateur sous forme de phrases en langue naturelle reformulées pour tenir compte de la requête initiale de l'utilisateur. Les indicateurs de pertinence et la reformulation en LN des interprétations de requêtes suggérées sont pour nous un moyen de dépasser le problème de l'habitabilité introduit précédemment. Ils donnent de plus à l'utilisateur un retour compréhensible et, de ce fait, augmente la confiance que l'on peut avoir dans le système.

4 La requête pivot

4.1 Justification

Comme nous l'avons expliqué précédemment, le processus d'interprétation se compose de deux étapes principales, produisant un résultat intermédiaire qui est la requête utilisateur traduite dans une nouvelle structure appelée "requête pivot". Cette structure est à mi-chemin entre la requête langue naturelle et la requête formelle souhaitée, et elle vise à stocker les résultats de la première étape d'interprétation. Nous utilisons le langage pivot afin de faciliter la mise en œuvre du multilinguisme au moyen d'un formalisme intermédiaire commun : un module spécifique de traduction doit être écrit pour chaque langue, mais l'étape de formalisation de la requête pivot reste la même. Pour le moment, nous avons adapté notre système à l'anglais et au français.

4.2 Définition et syntaxe du langage pivot

La grammaire détaillée du langage pivot est présentée dans [17]. Le langage pivot est une extension d'un langage composé de mots-clés, qui peuvent être connectés par des relations plus ou moins explicitées. Le symbole optionnel "`?`" avant un mot-clé signifie que ce mot-clé est le "focus" de la requête : nous voulons obtenir des résultats spécifiquement pour ce mot-clé. Une requête pivot est composée d'une conjonction de sous-requêtes qui peuvent être :

- des sous-requêtes unaires, comme `?"actor"` qui demande la liste des acteurs de la base de connaissances ;
- des sous-requêtes binaires qui qualifient un mot-clé par un autre mot-clé, comme la requête `?"actor": "married"` qui demande la liste des acteurs mariés ;

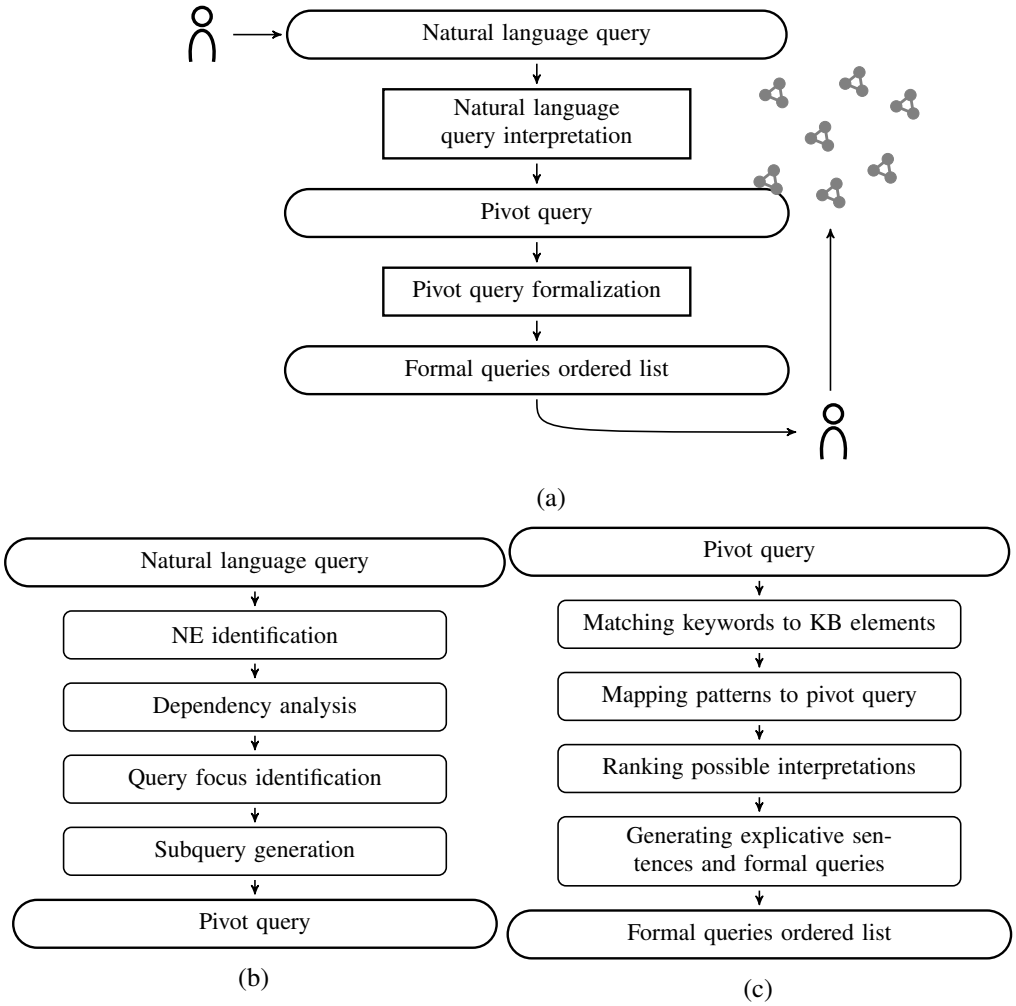


FIGURE 1 – Le processus d'interprétation de SWIP. Figure 1(a) propose une vue d'ensemble de tout le processus. Figures 1(b) et 1(c) montrent respectivement le détail des étapes d'interprétation de la requête langue naturelle et de formalisation de la requête pivot.

– des sous-requêtes ternaires qui qualifient, au moyen d'un mot-clé, la relation entre deux autres mots-clés, comme la requête `actor?"": "married to"= "Penelope Cruz"` qui demande la liste des acteurs qui sont/étaient mariés à Penelope Cruz.

4.3 De la LN à la requête pivot

La traduction d'une requête langue naturelle en une requête pivot se fonde sur l'utilisation d'un analyseur de dépendances syntaxiques qui produit un graphe dans lequel les sommets correspondent aux mots de la phrase et les arêtes aux dépendances grammaticales entre les mots. Comme illustré en Figure 1(b), la traduction se déroule en plusieurs étapes.

4.3.1 Identification des entités nommées

Avant d'analyser la requête, une première étape identifie dans la phrase les entités correspondant aux entités de la base de connaissances. Ces entités sont alors considérées comme un tout et ne sont pas séparées par l'analyseur dans l'étape suivante. Par exemple, dans la phrase "who are the actors of Underworld : Awakening", "Underworld : Awakening" sera considéré comme une entité nommée puisqu'il s'agit d'une étiquette d'une instance de Film dans la base de connaissances. Cette étape est cruciale lorsque l'on interroge des bases de connaissances contenant de longues étiquettes, comme des noms de groupes ou des titres de films composés de plusieurs mots, voire d'une phrase entière. Cette étape repose sur l'utilisation de gazetteers construits en cohérence avec la base de connaissances considérée.

Une fois que les ressources sont identifiées et le graphe de dépendances généré, un ensemble de règles sont appliquées pour construire la requête pivot.

4.3.2 Identification du focus de la requête

Tout d'abord, le focus de la requête est cherché en naviguant dans le graphe à la recherche d'un mot interrogatif. Dans le cas où un mot interrogatif est trouvé et qu'une arête le relie à un nom, le lemme correspondant au nom est considéré comme étant le focus de la requête. Par exemple pour la requête "Which films did Ted Hope produce ?", le focus de la requête sera " ?film". Si une telle arête n'existe pas, nous positionnons le focus comme étant le type de la réponse attendue par le mot interrogatif. Pour la requête "When did Charles Chaplin die ?", le focus est " ?date". Pour la requête "who played in the Artist ?",le focus est " ?person". Si aucun mot interrogatif n'est trouvé dans le graphe, nous recherchons des expressions spécifiques introduisant des requêtes listes, qui sont des requêtes qui attendent des listes de ressources remplissant certaines conditions ; de telles requêtes

commencent par “List”, “List all”, “I am looking for”, “What are”,“Give me”... Dans ce cas, le focus de la requête sera le lemme du nom identifié comme étant l'objet direct du verbe dans la clause dans laquelle l'expression a été trouvée. Par exemple dans la requête “List all the films in which Charles Chaplin acted”, le focus sera “ ?film”. Si, après ce processus, aucun focus n'a été identifié, nous considérons la requête comme dichotomique ce qui signifie qu'elle n'admet que deux réponses : “vrai” ou “faux” (ou bien “oui” ou “non”). C'est le cas de la requête “Was Jean Dujardin involved in the film The Artist ?” pour laquelle il n'y aura pas de focus défini.

4.3.3 Génération des sous-requêtes

Le graphe de dépendances est une fois de plus parcouru afin d'identifier les sous-requêtes potentielles de la requête pivot à partir des différentes propositions de la phrase. Une liste contenant les pronoms interrogatifs et les mots composant les expressions introduisant les requêtes listes est utilisée. Les sommets pour lesquels le lemme est dans la liste ou pour qui la catégorie grammaticale indique que c'est un déterminant ou une préposition sont ignorés.

Les sommets restant sont parcourus en appliquant les règles suivantes :

– si trois sommets correspondant aux sujet, verbe et complément d'objet de la proposition sont identifiés, une sous-requête ternaire est construite à partir du lemme de chaque sommet. Par exemple pour la requête “Who played in the Artist ?”, la requête ternaire correspondante est `?"person": "play"= "the Artist"`. Dans le cas où le sommet correspondant au sujet ou au complément est un pronom relatif, le mot-clé utilisé dans la requête pivot sera le lemme du sommet référencé par le pronom. Par exemple dans la requête “Who played in a film in which Jean Dujardin also played ?”, la sous-requête correspondant à la seconde proposition est `"film": "play"= "Jean Dujardin"`, la sous-requête générée pour la première est `?"person": "play"= "film"`. Remarquons que l'utilisation du même mot-clé“film” dans les deux sous-requêtes indique que le même film doit être considéré dans chacune d'entre-elles.

– si deux sommets font partie d'un syntagme nominal identifié quand leur catégorie grammaticale est “nom” et que l'arête représente une dépendance de tête ou d'expansion, une sous-requête binaire est construite à partir du lemme de chacun des sommets. Par exemple, pour la requête “Give me all the films of Jean Dujardin”, la sous-requête

générée est `?"film": "Jean Dujardin"`.

Pour chaque sommet, chaque règle est testée. Ceci signifie que pour une même proposition de la phrase initiale peuvent être générées plusieurs sous-requêtes. Par exemple pour la phrase composée d'une seule proposition "Was Jean Dujardin involved in the film The Artist ?", les deux sous-requêtes `"film": "The Artist"` et `"Jean Dujardin": "involved"= "The Artist"` sont générées.

Les règles peuvent paraître simples, mais nous avons observé que la structure des requêtes exprimées par les utilisateurs finals est généralement basique.

5 Les patrons de requêtes

Les phrases suivantes sont de simples requêtes en LN qu'un utilisateur peut poser sur le domaine du cinéma : (i) "Which actors play in the movie Biutiful ?" ; (ii) "Which thrillers were released in 2008 ?" ; (iii) "Which movies were made by a French director ?" ; (iv) "Which movies were directed by the Coen brothers ?". Ces requêtes semblent très familières. Tout le monde s'est demandé un jour ou l'autre quel acteur jouait dans tel ou tel film. C'est une observation simple qui nous a naturellement conduits à l'idée de patron.

5.1 Justification

Le postulat principal sur lequel repose notre travail est que, dans les applications réelles, les requêtes soumises sont pour l'essentiel des variations autour de quelques familles typiques de requêtes. Les auteurs de [22] analysent des journaux de requêtes d'un moteur de recherche réel du Web et discutent leurs résultats par rapport à des études similaires plus anciennes [11, 21]. Malgré des premières observations tendant à infirmer notre hypothèse, leurs conclusions nous confortent dans notre voie. Les auteurs pointent le fait que le vocabulaire (et donc potentiellement, pour ce qui nous intéresse, sa sémantique) de leurs requêtes et très varié. Sur l'ensemble de requêtes analysé, comportant 926.877 requêtes contenant au moins un mot-clé, of the 140,279 unique terms, 57,1% étaient utilisés une seule fois, 14,5% deux, et 6,7% trois fois. Cela représente un taux élevé de termes très rarement utilisés, en comparaison des ressources textuelles classiques. Ce point doit être nuancé puisque ce phénomène est en partie causé par un nombre élevé de fautes d'orthographe, de termes dans des

langues autres que l'anglais, ou de termes spécifiques au Web comme les URL.

D'un autre côté, quelques termes sont utilisés très fréquemment. Dans les requêtes analysées, les 67 mots significatifs les plus fréquents (i.e. que les termes comme "and", "of", "the" ne sont pas pris en compte) représentent seulement 0,04% of unique terms that account for 11.5% of all terms used in all queries.

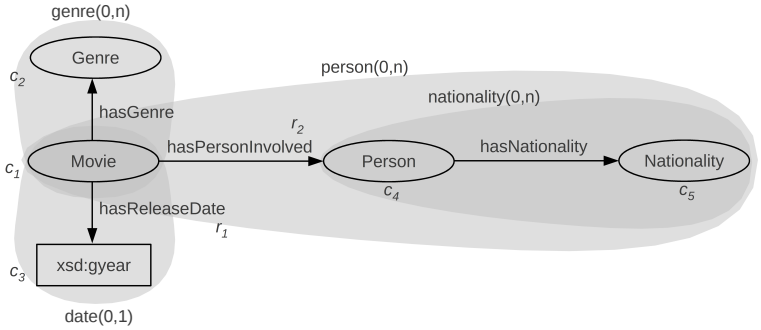
Deux analyses ont été menées pour compléter ces résultats. La première s'intéresse à la co-occurrence de termes ; en identifiant les paires de termes co-occurrant le plus fréquemment, il est possible de mettre en évidence des sujets populaires et récurrents. La dernière analyse présentée est qualitative et est la plus pertinente pour la sémantique des requêtes. Elle consiste en une classification manuelle d'un sous-ensemble de l'ensemble des requêtes. 11 catégories principales sont mises en évidence pour cette analyse, chacune correspondant à un ensemble de sujets reliés. De telles observations nous conduisent à proposer un mécanisme permettant d'exprimer les requêtes utilisateur en LN et de les traduire dans un graphe requête en adaptant des patrons de requêtes choisis en fonction des mots-clés de la requête initiale. L'utilisation de patrons était la principale différence entre notre approche et les approches développées par d'autres équipes concomitamment à la nôtre.

5.2 Les patrons de requête

Nous ne définissons pas dans cet article la notion de patron au sens où nous l'entendons, puisque cela a fait l'objet d'un article présenté lors de la conférence IC'2012 [18]. Rappelons simplement qu'un patron de requête est composé : (i) d'un graphe RDF qui représente la famille de requête considérée, (ii) : d'un sous-ensemble d'éléments du graphe appelés éléments qualifiants ; ces éléments sont considérés comme étant caractéristiques du patron et ils sont pris en compte lors de l'appariement de la requête utilisateur avec les patrons ; (iii) de "sous-patrons" qui peuvent être optionnels ou répétables et sont caractérisés par des cardinalités.

La Figure 2 présente un exemple de patron de requête. Il est composé de quatre sous-patrons nommés *genre*, *date*, *person* et *nationality*. Tous sont optionnels. Ils sont également répétables, sauf le sous-patron *date* : il est considéré qu'un film ne peut pas avoir plus d'une date de sortie. Dans le template de phrase descriptive, les parties correspondant à un sous-patron sont écrites entre crochets et sont indicées par l'identificateur, les sommets qualifiants sont soulignés et indicés en référence à l'élément du graphe

correspondant.



A movie$_{c_1}$ [of genre c_2]$_{genre}$ [that was released on c_3]$_{date}$
[has for person involved$_{r_2}$ a person$_{c_4}$ [which is c_5]$_{nationality}$]$_{person}$

FIGURE 2 – Exemple de patron de requêtes.

La façon dont un patron de requêtes est instancié, c'est à dire la façon selon laquelle il est transformé en requête SPARQL à partir de la requête pivot, n'est pas présentée dans cet article. C'est illustré par la Figure 1(c) et nous le rappelons ici très brièvement. Chaque élément de la requête utilisateur exprimé dans la requête pivot est mis en correspondance avec un ou plusieurs éléments de la base de connaissances. Les éléments de la base de connaissances peuvent être une classe, une propriété, une instance ou un type de littéral (qui peut être n'importe quel type de RDF Schema). Nous apparions alors les patrons de requêtes à ces éléments. Les différents appariements possibles sont alors présentés à l'utilisateur par le biais de phrases en langue naturelle. La phrase sélectionnée par l'utilisateur conduit à la génération de la requête SPARQL finale [18, 16, 17].

6 Implémentation et évaluation

Un prototype de notre système a été mis en œuvre afin d'évaluer son efficacité. Il est disponible à `http://swip.univ-tlse2.fr/SwipWebClient`. Il a été implémenté au moyen de services Web et utilise TreeTagger[20] pour le POS tagging et MaltParser[15] (avec le modèle pré-entraîné pour l'anglais) pour l'analyse de dépendances syntaxiques de la requête utilisateur. Le système implémente la deuxième partie du processus (traduction de la requête pivot en requête formelle) en utilisant un serveur SPARQL

fondé sur le processeur de requêtes ARQ [4], configuré ici pour utiliser LARQ [5], permettant l'utilisation des fonctionnalités d'Apache Lucene [6], comme l'indexation et le "Lucene score" (utilisé pour obtenir une mesure de similarité de chaînes de caractères).

Des expérimentations ont été menées dans le cadre de la campagne d'évaluation du challenge QALD-2 [7]. Ce challenge se fonde sur MusicBrainz, i.e. une ontologie de la musique et 50 millions de triplets associés. Nous avons construit des patrons en utilisant un ensemble de 100 requêtes d'entrainement fourni par les organisateurs du challenge sous forme de requêtes LN et leur traduction SPARQL. Nous avons ensuite évalué notre approche sur 50 nouvelles requêtes. Nous avons identifié des catégories dont l'expressivité excède celle que nous pouvons prendre en compte avec notre approche. Les exemples les plus évidents sont les requêtes nécessitant de la comparaison de chaînes comme par exemple *Give me all bands whose name starts with Play.*, des agrégations complexes (i.e. agrégations autres que celles couvertes par l'opérateur COUNT de SPARQL 1.0), telles que *Give me all artists who contributed to more than three collaborations.* ou des inférences spécifiques, comme *Which artists turn 50 on May 28, 2012 ?* (qui demande en réalité quels sont les artistes nés le 28 mai 1962). Parmi ces 50 requêtes, nous avons identifié 12 requêtes appartenant à ces catégories non prises en charge. Sur les 38 requêtes restantes, 23 ont été interprétées correctement et 15 ont échoué. Nous obtenons une précision de 0,46, qui est meilleure que tous les autres systèmes ayant participé au challenge QALD-2. Pourtant, ces résultats ne peuvent pas être comparés en détail puisque les modalités d'évaluation diffèrent sensiblement et que ces systèmes ont fait tourner leurs évaluations sur la base de connaissances DBPedia, avec des requêtes utilisateurs différentes.

Afin d'évaluer la qualité de la première étape principale de notre processus (traduction de la requête LN en requête pivot), nous avons mené une autre expérimentation, prenant en compte seulement la deuxième étape de notre processus (la formalisation de la requête pivot), considérant en entrée des requêtes pivot écrites à la main. Nous avons dénombré 31 requêtes interprétées correctement et seulement 7 erreurs. Ces résultats sont singulièrement meilleurs que les précédents, et montrent la perte de précision due à la traduction automatique en langage pivot. Pourtant, nous avons mis

4. http ://openjena.org/ARQ/
5. LARQ = Lucene + ARQ, voir http ://jena.sourceforge.net/ARQ/lucene-arq.html
6. http ://lucene.apache.org/
7. http ://greententacle.techfak.uni-bielefeld.de/ cunger/qald/2/

en évidence le fait que la majorité de ces nouvelles mauvaises interprétations sont dues à des erreurs de l'analyseur de dépendances que nous utilisons "tel quel". Donc nous considérons que ça ne discrédite pas notre approche et nous avons bon espoir d'améliorer les performances de SWIP en améliorant celles de l'analyseur (en utilisant les futures versions développées par d'autres équipes, ou en entraînant l'analyseur avec un corpus plus approprié).

7 Conclusion et perspectives

Dans cet article, nous avons présenté l'approche que nous suivons en vue de permettre à un utilisateur final de requêter des bases de connaissances fondées sur des graphes étiquetés. Cette approche est mise en œuvre dans le système SWIP ; elle est caractérisée par l'utilisation de patrons de requêtes, qui guident l'interprétation de la requête utilisateur exprimée en LN et sa traduction en requête formelle.

Bien que certaines améliorations doivent encore être apportées, la mise en place des deux étapes principales du système est quasiment opérée et les premiers résultats expérimentaux sont trsè encourageants. Nous avons l'intention de développer notre travail dans trois directions :

- expérimenter la facilité de s'adapter à différentes langues ; nous allons participer à la tâche *Multilingual question answering* du challenge QALD-3 et nous développons un partenariat avec l'IRSTEA (Institut National de Recherche en Sciences et Technologies pour l'Environnement et l'Agriculture) afin de construire une application réelle concernant des requêtes en français sur l'agriculture biologique ou raisonnée ;
- expérimenter des méthodes pour automatiser au moins partiellement la conception de patrons de requête ; nous souhaitons tout d'abord déterminer automatiquement les structures de patrons en analysant des graphes requêtes exemples, puis comparer cette approche à des approches fondées sur l'apprentissage de requêtes en LN ;
- étendre le nombre de requêtes pouvant être traités par notre système, par exemple en prenant en compte les extensions de SPARQL 1.1.

Références

[1] ATHANASIS N., CHRISTOPHIDES V. & KOTZINOS D. (2004). Generating on the fly queries for the semantic web : The ics-forth graphical rql interface (grql). In *International Semantic Web Conference*, p. 486–501.

[Cabrio *et al.*] CABRIO E., COJAN J., APROSIO A., MAGNINI B., LAVELLI A. & GANDON F. Qakis : an open domain qa system based on relational patterns.

[3] CHAKRABARTI S. (2004). Breaking through the syntax barrier : Searching with entities and relations. *Knowledge Discovery in Databases : PKDD 2004*, p. 9–16.

[4] CLEMMER A. & DAVIES S. (2011). Smeagol : a "specific-to-general" semantic web query interface paradigm for novices. In *Proceedings of the 22nd international conference on Database and expert systems applications - Volume Part I*, DEXA'11, p. 288–302, Berlin, Heidelberg : Springer-Verlag.

[5] COGUI (2009). A conceptual graph editor. Web site. http ://www.lirmm.fr/cogui/.

[6] COMPAROT C., HAEMMERLÉ O. & HERNANDEZ N. (2010a). An easy way of expressing conceptual graph queries from keywords and query patterns. In *ICCS*, p. 84–96.

[7] COMPAROT C., HAEMMERLÉ O. & HERNANDEZ N. (2010b). Expression de requêtes en graphes conceptuels à partir de mots-clés et de patrons (regular paper). In *Journées Francophones d'Ingénierie des Connaissances (IC), Nîmes, 08/06/2010-11/06/2010*, p. 81–92, http ://www.cepadues.com/ : Cépaduès Editions.

[8] DEKLEVA S. (1994). Is natural language querying practical ? *ACM SIGMIS Database*, **25**(2), 24–36.

[9] ELBASSUONI S., RAMANATH M., SCHENKEL R. & WEIKUM G. (2010). Searching rdf graphs with sparql and keywords. *IEEE Data Eng. Bull.*, **33**(1), 16–24.

[10] FERRÉ S. & HERMANN A. (2012). Reconciling faceted search and query languages for the semantic web. *International Journal of Metadata, Semantics and Ontologies*, **7**(1), 37–54.

[11] JANSEN B., SPINK A. & SARACEVIC T. (2000). Real life, real users, and real needs : a study and analysis of user queries on the web. *Information processing & management*, **36**(2), 207–227.

[12] KAUFMANN E. & BERNSTEIN A. (2010). 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.

[13] LEHMANN J. & BÜHMANN L. (2011). Autosparql : let users query your knowledge base. In *Proceedings of the 8th extended semantic web conference on The semantic web : research and applications-Volume Part I*, p. 63–79 : Springer-Verlag.

[14] LEI Y., UREN V. S. & MOTTA E. (2006). Semsearch : A search engine for the semantic web. In *EKAW*, p. 238–245.

[15] NIVRE J., HALL J. & NILSSON J. (2006). Maltparser : A data-driven

parser-generator for dependency parsing. In *Proceedings of the fifth international conference on Language Resources and Evaluation (LREC2006), May 24-26, 2006, Genoa, Italy*, p. 2216–2219 : European Language Resource Association, Paris.

[16] PRADEL C., HAEMMERLÉ O. & HERNANDEZ N. (2011). Expressing conceptual graph queries from patterns : how to take into account the relations. In *Proceedings of the 19th International Conference on Conceptual Structures, ICCS'11, Lecture Notes in Artificial Intelligence # 6828*, p. 234–247, Derby, GB : Springer.

[17] PRADEL C., HAEMMERLÉ O. & HERNANDEZ N. (2012a). A Semantic Web Interface Using Patterns : The SWIP System (regular paper). In M. CROITORU, S. RUDOLPH, N. WILSON & J. HOWSE, Eds., *IJCAI-GKR Workshop, Barcelona, Spain, 16/07/2011-16/07/2011*, number 7205 in LNAI, p. 172–187 : Springer.

[18] PRADEL C., HAEMMERLÉ O. & HERNANDEZ N. (2012b). Des patrons modulaires de requêtes sparql dans le système swip. *23es Journées Francophones d'Ingénierie des Connaissances*.

[19] RUSSELL A. & SMART P. R. (2008). Nitelight : A graphical editor for sparql queries. In *International Semantic Web Conference (Posters & Demos)*.

[20] SCHMID H. (1994). Probabilistic part-of-speech tagging using decision trees.

[21] SILVERSTEIN C., MARAIS H., HENZINGER M. & MORICZ M. (1999). Analysis of a very large web search engine query log. In *ACm SIGIR Forum*, volume 33, p. 6–12 : ACM.

[22] SPINK A., WOLFRAM D., JANSEN M. & SARACEVIC T. (2001). Searching the web : The public and their queries. *Journal of the American society for information science and technology*, **52**(3), 226–234.

[23] THOMPSON C., PAZANDAK P. & TENNANT H. (2005). Talk to your semantic web. *Internet Computing, IEEE*, **9**(6), 75–78.

[24] TRAN T., WANG H., RUDOLPH S. & CIMIANO P. (2009). Top-k exploration of query candidates for efficient keyword search on graph-shaped (rdf) data. In *ICDE*, p. 405–416.

[25] ZHOU Q., WANG C., XIONG M., WANG H. & YU Y. (2007). Spark : Adapting keyword query to semantic search. In *ISWC/ASWC*, p. 694–707.