Une approche de recherche d'attributs pertinents pour l'agrégation d'information

Ines Krichen, Arlind Kopliku, Karen Pinel-Sauvagnat, Mohand Boughanem

Université de Toulouse, IRIT UMR 5505 CNRS
118 route de Narbonne, F-31062 Toulouse cedex 9
{krichen,kopliku,sauvagna,boughanem}@irit.fr

RÉSUMÉ. *La recherche d'information agrégée permet, en réponse à une requête, d'agréger des granules d'information provenant de plusieurs sources et de renvoyer à l'utilisateur un ensemble d'informations bien organisées. Le résultat agrégé est une alternative à la traditionnelle liste de documents répondant chacun à une partie du besoin utilisateur. Nous nous intéressons particulièrement dans cet article à la recherche agrégée relationnelle, qui se focalise sur les granules d'information classe - instance - attributs. Nous présentons une approche d'agrégation des résultats répondant à une requête de type classe, basée sur la détection d'attributs pertinents et présentant les résultats sous forme de tableau. Notre approche s'appuie sur 3 étapes que sont la sélection des instances et attributs pertinents à la classe, leur filtrage et le tri des attributs pertinents. Afin de tester cette approche, nous avons évalué la qualité des tableaux retournés à partir du dataset DBpedia. L'évaluation montre que le score des attributs est influencé par les instances sélectionnées, et que la fonction de pondération utilisée impacte significativement la qualité des résultats renvoyés.*

ABSTRACT. *Given a query, aggregated search aims at aggregating information nuggets retrieved from differents sources. The aggregated result is a coherent answer that can be seen as an alternative to the traditional ranked list of documents. We focus in this paper on relational aggregated search where information nuggets are mainly classes, instances and attributes. We present a result aggregation approach to answer to class queries. Our approach is based on relevant attributes detection and uses tables to present results. It is composed of three steps: instances and attributes selection, filtering and attributes ranking. In order to evaluate our approach we assess the quality of tables constructed from DBpedia. The evaluation shows that attributes scoring depends on the selected instances, and that the weighting function can affect significantly the quality of tabular results.*

MOTS-CLÉS : *Recherche d'information, Agrégation de résultats , Recherche d'attributs*

KEYWORDS: *Information Retrieval, Aggregated search, Attribute search*

DOI:10.3166/DN.22.1.1-?? © 2012 Lavoisier

1. Introduction

Les Systèmes de Recherche d'Information (SRI) renvoient en réponse à une requête une liste de documents potentiellement pertinents. L'information pertinente recherchée peut se retrouver entièrement dans un document ou être éparpillée dans plusieurs documents (Boughanem, Savoy, 2008). L'utilisateur doit alors parcourir la liste des documents retournés et regrouper et sélectionner les parties (ou granules d'information) qu'il juge pertinentes. Un granule d'information peut être une définition, une image, un texte descriptif, une vidéo, un tableau, ou même un attribut avec sa valeur (par exemple une adresse, un téléphone, etc). La réponse pertinente idéale serait donc composée de tous les granules sélectionnés par l'utilisateur.

Plutôt que de laisser l'utilisateur construire lui-même son résultat idéal, une alternative serait d'extraire et de regrouper automatiquement les granules d'information nécessaires. Ce genre d'approche est ce que l'on appelle la *recherche agrégée*. L'agrégation d'information à partir de plusieurs sources peut aider à satisfaire de nombreux besoins d'information. Par exemple, pour les requêtes de type instance (*"Toulouse"*) ou classe (*"villes de France"*), qui représentent 71% des requêtes posées sur le Web d'après (Kato *et al.*, 2009), la liste n'est vraiment ni la meilleure ni la plus naturelle des réponses.

Ainsi, pour la requête *"restaurant chinois à Paris"*, retourner tous les documents qui contiennent les termes *"restaurant"* *"chinois"* et *"Paris"* risque de couper l'appétit à l'utilisateur. La construction d'un résultat agrégé (voir la figure 1) semble être une meilleure solution.

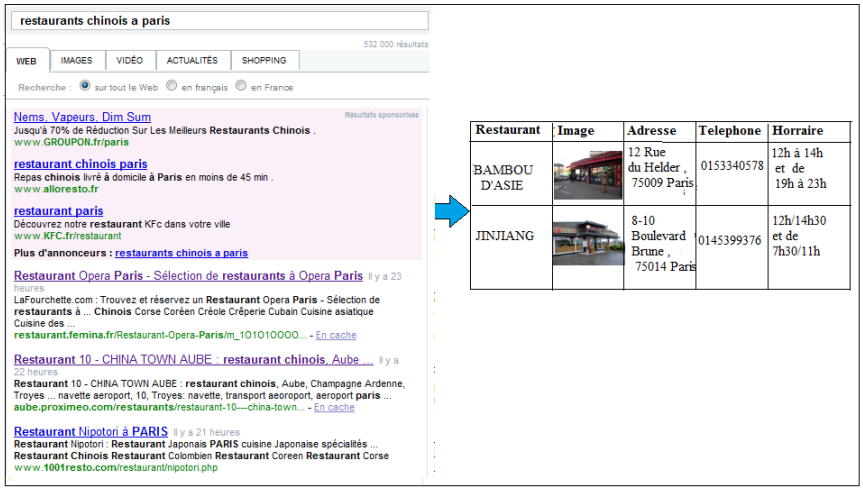

FIGURE *1. Exemple de requête et de résultat agrégé*

Nous nous proposons dans cet article de nous focaliser sur la *recherche agrégée relationnelle*, construite autour de granules d'information spécifiques, que sont les *classes*, *instances* et *attributs*. A partir d'une extraction d'instances, notre approche a pour but de sélectionner les plus représentatives et de trier leurs attributs afin de répondre à une requête de type classe (*"villes de France", "albums de Nirvana"*,...). Pour représenter les résultats, nous utilisons l'une des formes les plus répandues dans la recherche agrégée, celle des tableaux. Ces tableaux résultats contiennent un ensemble d'instances représentatives de la classe, et les attributs associés.

Afin de construire ces tableaux, plusieurs étapes sont nécessaires:

– rechercher les instances de la classe,

– rechercher les attributs (nom-valeur) de la classe,

– agréger ces granules documentaires afin de former le tableau, en privilégiant les instances et attributs les plus représentatifs.

Nous nous intéressons ici particulièrement à la dernière étape, celle de l'agrégation des résultats. Afin d'obtenir les instances et attributs correspondant à la classe, nous utilisons comme source de données Wikipedia, et plus particulièrement le *dataset* DBPedia.

La suite de l'article est organisée comme suit. Nous présentons le cadre de la recherche agrégée ainsi que l'état de l'art associé dans la section 2. Notre approche pour l'agrégation d'information est détaillée dans la section 3. Les évaluations que nous avons conduites (mise en place du système, protocole expérimental et résultats) sont décrites en section 4 et nous concluons et énonçons quelques perspectives en section 5.

2. Etat de l'art et problématique

Alors que la recherche d'information "traditionnelle" peut être taxée de savoir mal gérer la dispersion des données et l'ambiguïté de certaines requêtes, de manque de focus ou à l'inverse de manque de diversité dans ses résultats, la recherche agrégée, nouveau paradigme de la recherche d'information, s'impose comme une de ses alternatives les plus prometteuses.

2.1. Recherche agrégée : contexte et problématique

L'expression *"recherche agrégée"* a été définie pour la première fois dans un atelier à SIGIR 2008 (Murdock, Lalmas, 2008): *c'est une tâche cherchant à rassembler des informations provenant de sources différentes, et à les présenter dans une seule interface.* En d'autres termes, la recherche agrégée tente d'identifier le contenu nécessaire, de l'organiser et de le présenter à l'utilisateur de manière à faciliter sa recherche d'information. Un moteur de recherche agrégé est construit en surcouche d'un ou plusieurs autres moteurs de recherche (sources). Des informations de différents types (image, vidéo, etc.) et de différentes granularités (passage de texte, entités, attributs,

etc. . .) sont reliées et parfois même combinées par une ou plusieurs relations logiques (association, groupe, ordre, etc.) afin de composer un résultat agrégé (Kopliku, 2009).

Le processus de recherche d'information agrégée peut se décomposer en trois étapes principales (Kopliku, 2011), illustrées sur la figure 2:

– ***dispatching* de la requête**: cette étape, précédant la recherche elle-même, consiste à interpréter la requête et à sélectionner la ou les sources à interroger. Le terme source ici réfère à *un moteur de recherche d'information indexant au moins une collection et utilisant un algorithme de recherche donné*. La notion de source est importante, car les systèmes de recherche d'information peuvent être classifiés en systèmes mono-source et multi-source. Quand plusieurs sources sont présentes, cette étape permet donc de sélectionner les sources à utiliser.

– **recherche de *nuggets***: la recherche agrégée permet de retrouver du contenu de différentes granularités (phrase, passage, document) et de différents types (texte, image, video,...). Ces contenus sont appelés *nuggets*, où *un nugget est un granule d'information de taille quelconque satisfaisant un besoin d'information*. Cela peut-être un document entier ou une partie de document, mais aussi un simple mot (comme par exemple *"Paris"* permettant de répondre à la requête *"capitale de la France"*). La recherche de nuggets correspond à l'identification de contenus pertinents, mais n'implique par leur assemblage.

– **l'agrégation de résultats**: étant donnés un ensemble de nuggets pertinents, l'agrégation de résultats consiste à les assembler pour former la réponse finale. En fonction des systèmes de recherche agrégée, il peut s'agir de résumer, grouper, fusionner, trier, clusteriser,...

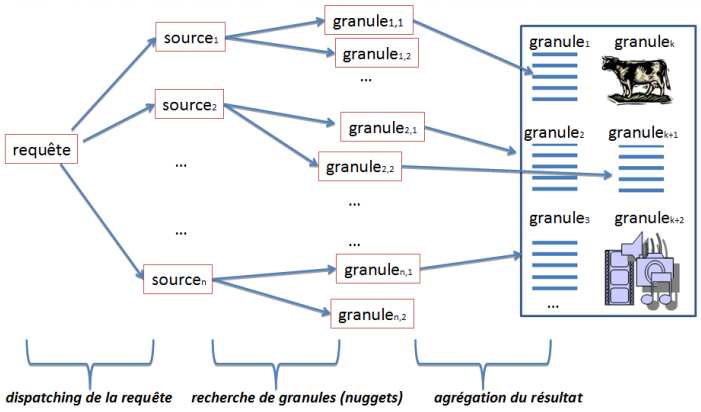

FIGURE 2. *Processus de recherche d'information agrégée*

De nombreux domaines de recherche sont liés à la recherche agrégée, parmi lesquels nous pouvons citer le Question-Réponse, la Génération de Langage Naturel, la

recherche d'information distribuée, la méta-recherche, la fusion de données ou encore l'extraction d'informations. La recherche agrégée a également donné lieu à de nouveaux domaines de recherche spécifiques comme la recherche agrégée relationnelle ou encore l'agrégation de recherches verticales, ces deux dernières se basant sur des solutions proposées par les domaines de recherche précédents.

Nous avons choisi dans les sections suivantes d'illustrer les différentes facettes de la recherche agrégée via trois grands groupes d'approches, qui sont aujourd'hui les plus visibles dans la littérature : la génération de langage naturel (Paris *et al.*, 2010)(Sauper, Barzilay, 2009), l'agrégation de recherches verticales (Ji *et al.*, 2009), ou encore la recherche agrégée relationnelle (Haas *et al.*, 2011) (Cafarella *et al.*, 2006) (Elmeleegy *et al.*, 2009).
Nous accorderons une attention particulière à la recherche agrégée relationnelle, servant de cadre aux travaux présentés dans cet article.

2.2. Génération de langage naturel et recherche agrégée inter-verticale

Les approches s'appuyant sur des **techniques de génération de langage naturel** cherchent à générer des réponses cohérentes et compréhensibles, formulées en langage naturel, en réponse à un besoin d'information généralement exprimé sous forme de question. Parmi elles nous pouvons citer l'approche de (Paris *et al.*, 2010) dans laquelle les auteurs se sont appuyés sur des techniques de traitement de langue pour agréger des résultats de recherche afin de construire un rapport de surveillance, planifier un programme pour des voyages ou produire une brochure pour des organisations. Par exemple, pour aboutir à un rapport de surveillance dans le transport maritime, une approche de planification de discours capable d'agréger des informations de sources hétérogènes (base des mouvements des navires, bulletins météorologiques) a été utilisée. Les auteurs ont élaboré un processus capable de générer automatiquement la requête appropriée pour chaque source de données et par la suite de récupérer les passages pertinents au niveau de chaque source pour les agréger finalement dans un document cohérent. Ce document contient une liste qui présente l'index du rapport produit, des tableaux informatifs sur l'état météorologique de la région en question et du texte explicatif.
D'autres auteurs comme (Sauper, Barzilay, 2009) ont cherché à construire automatiquement des articles médicaux pour Wikipedia à partir de modèles qu'ils ont eux-mêmes générés. Ils ont montré que l'intégration d'informations structurées provenant d'articles déjà existants dans Wikipedia dans le processus d'apprentissage d'extracteurs de contenus améliore la qualité des articles construits. Les résultats de l'évaluation confirment les avantages d'intégrer des informations structurées dans le processus de sélection des contenus par rapport aux approches qui n'ont pas un modèle de structure pour chaque thème vu. Malheureusement une telle approche n'est pas applicable dans le cas où le modèle approprié n'est pas disponible.
Ces deux exemples s'intègrent bien dans le cadre général de la recherche d'information agrégée: plusieurs sources doivent être interrogées pour retrouver les nuggets pertinents, et une phase d'agrégation des résultats est ensuite nécessaire.

Le but de la **recherche agrégée inter-verticale** est d'agréger des résultats de recherche provenant de différents moteurs de recherche verticaux, la plupart du temps dans un contexte de recherche Web. Les différentes recherches verticales renvoient des informations d'un seul type de support (image, vidéo, etc...) ou de contenu (news, livre, etc...). Ces informations sont ensuite triées et agrégées pour être présentées à l'utilisateur. L'agrégation de différentes recherches verticales est l'interprétation la plus commune et la plus évidente de la recherche agrégée. En raison de l'augmentation des différents types de support et de contenu d'information dans la recherche d'information, il est devenu inévitable de les intégrer dans une même interface pour pouvoir les exploiter d'avantage. La majorité des moteurs de recherche adoptent déjà ce type d'agrégation. Parmi eux, nous pouvons citer Google Universal [1](voir exemple sur la figure 3), ou encore Bing [2].

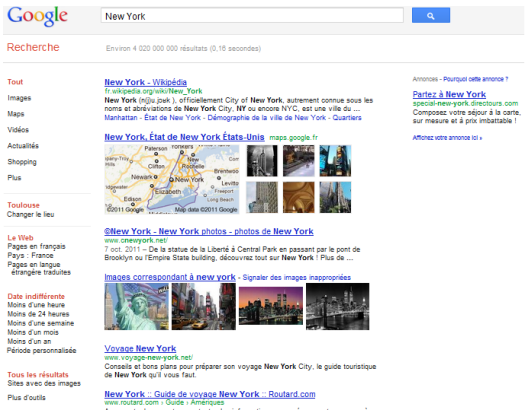

FIGURE *3. Résultats de la requête "New York" pour le moteur de recherche Google Universal. La liste des résultats contient des images, un plan et des pages web.*

D'autres approches utilisent l'agrégation de plusieurs sources pour construire des résumés de recherche compréhensibles par l'utilisateur. Nous pouvons par exemple citer les auteurs de (Haas *et al.*, 2011) qui ont présenté une nouvelle façon de générer des résumés de recherche basée sur des données structurées ou sur des métadonnées associées aux documents Web. En exploitant ces données, ils ont proposé des résultats de recherche résumé qui intègrent des images, des liens et des éléments interactifs (voir figure 4). Pour ce faire, un robot d'extraction a été implémenté, afin de soutenir l'extraction d'informations à partir de documents Web et recueillir des données du Web sémantique nécessaires pour générer des résultats améliorés. L'évaluation du

1. http://www.google.fr/
2. http://www.bing.com/

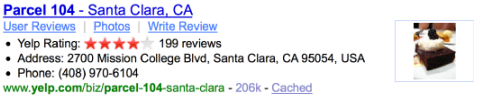

FIGURE *4. Exemple de résultat de recherche pour l'approche de (Haas et al., 2011)*

modèle auprès des utilisateurs a montré que les utilisateurs préfèrent largement les résultats de recherches améliorés aux résultats de recherches traditionnels.

Pour conclure, la recherche agrégée inter-verticale permet de donner plus de visibilité aux moteurs de recherche verticaux, dont elle intègre les avantages au sein de la recherche Web "traditionnelle". L'étape de dispatching de la requête permet de sélectionner les moteurs de recherche verticaux potentiellement pertinents, la recherche de nuggets est faite par les différents moteurs verticaux, et l'étape d'agrégation des résultats permet de trier les résultats les uns par rapport aux autres.

2.3. Recherche agrégée relationnelle

La recherche agrégée relationnelle est une généralisation de la recherche relationnelle (Cafarella *et al.*, 2006) et de la recherche d'entités (Balog *et al.*, 2009). Elle peut être vue comme un paradigme de recherche basé sur les relations entre les différents nuggets. En plus de la recherche de nuggets, la recherche agrégée relationnelle doit trouver des relations, nécessaires ensuite pour la phase d'agrégation des résultats. Les problématiques spécifiques à la recherche agrégée relationnelle sont donc les suivantes:

– **la recherche de nuggets**: n'importe quel type de nugget peut être utilisé pour la recherche agrégée relationnelle. Il est possible de retrouver et mettre en relation des images, des vidéos, etc. Il y a cependant 3 types de nuggets au centre de la recherche agrégée relationnelle:

- les *classes* (par exemple *"pays"*, *"vins français"*, *"présidents des Etats-Unis"*),
- les *instances* [3] (par exemple *"France"*, *"Angleterre"*, *"Italie"*)
- les *attributs* (*"capitale: Paris"*, *"superficie: 674 843 km2"*, ...)

– **la recherche de relations**: toutes les relations nugget-nugget peuvent être intéressantes pour la recherche agrégée relationnelle. Parmi elles, la relation instance-nugget met en relation une instance avec n'importe quel type de contenu (par exemple *"l'image X est une image d'Elvis Presley"*). D'autre part, les nuggets classes, ins-

3. Le terme instance est synonyme d'entité nommée, mais il est préféré dans cet article car sa sémantique permet une relation directe avec le terme classe.

tances et attributs sont particulièrement intéressants car les relations entre eux sont communes et implicites. Par exemple *"capitale: Paris"* est un attribut de l'instance *"France"*, où *"France"* est une instance de la classe *"Pays"*. Nous pouvons donc définir ici trois relations spécifiques:

- relation instance-classe: cette relation existe entre une instance et sa classe (par exemple *"Everest - montagne"*),

- relation instance-instance: cette relation existe entre deux instances: cela peut être un verbe comme par exemple *"John Travolta joue dans Pulp Fiction"* mais aussi une relation plus large comme *"est de la même classe que"*.

- relation instance-attribut: cette relation existe entre une instance et ses attributs (par exemple *"Everest - hauteur"*)

Les relations sont dans la plupart des cas extraites à l'aide de patterns sur le texte (Paris *et al.*, 2010) (Etzioni *et al.*, 2005), ou bien de la structure de certains documents (Elmeleegy *et al.*, 2009) (Cafarella, Halevy, Zhang *et al.*, 2008).

– **le dispatching de requêtes**. D'après (Cafarella *et al.*, 2006), nous pouvons distinguer trois types de besoins d'information pour lesquels les bénéfices de la recherche agrégée relationnelle sont évidents:

- les *requêtes attribut* (*"PIB de l'angleterre"*, *"adresse de l'hôtel Gambetta"*)
- les *requêtes instance* (*"Samsung GalaxyS"*, *"William Shakespeare", "Grèce"*)
- les *requêtes classe* (*"portables Apple"*, *"écrivains anglais*")

– **l'agrégation de résultats**: les relations permettent d'assembler les résultats de façon spécifique. Quand la requête est un attribut, le meilleur résultat est de retourner sa valeur. Quand la requête est une instance, le meilleur choix pourrait être un résumé des différents attributs. Quand la requête est une classe, le résultat pourrait être un tableau comparatif des instances de cette classe et de leurs attributs.

Parmi les approches effectuant de la recherche agrégée relationnelle, nous pouvons citer le travail de (Cafarella *et al.*, 2006) qui a mis en place un moteur de recherche intitulé TextRunner. Ce moteur essaie de répondre à des requêtes complexes qui exigent comme résultats une liste d'instances ou bien un tableau en exploitant un graphe d'instances-relations. En premier lieu, l'approche proposée cherche à traiter la liste d'instances. Pour ce faire, un robot d'exploration standard est d'abord utilisé pour récupérer des pages Web. Par la suite, les auteurs appliquent un mécanisme d'extraction d'information pour chaque phrase sur chaque page récupérée. La plupart des informations extraites sont soit des objets (instances) soit des prédicats (relations). Ces derniers vont être reliés pour y faire un graphe G. Afin de faciliter le traitement des requêtes, TextRunner calcule un index inversé sur le texte des triplets qui forment le graphe d'extraction G. L'évaluation a prouvé l'efficacité de cette méthode à retrouver les bonnes instances par rapport aux systèmes d'extraction d'information qui se basent sur des patterns classiques.

Les tableaux dans les pages Web sont connus pour être des sources pertinentes d'extraction. Par exemple, dans (Cafarella, Halevy, Zhang *et al.*, 2008)(Cafarella, Ha-

levy, Wang *et al.*, 2008), les auteurs cherchent à détecter, classifier et filtrer les tables relationnelles du Web afin d'agréger les données retournées dans un document final. L'inconvénient de cette approche est que la détection des tables relationnelles dans le Web n'est pas toujours évidente: tables relationnelles et non relationnelles sont souvent mélangées sur la même page, il n'est donc pas facile de distinguer les 1,1% de bonnes relations du reste.

Dans (Kopliku *et al.*, 2011), nous avons proposé une approche utilisant les tableaux dans les pages Web, qui détecte et trie des attributs pertinents pour une requête instance ou classe. Pour ce faire, nous avons mis en place trois niveaux de sélection afin de détecter les tableaux relationnels et leurs attributs pertinents. Les bons attributs sont ensuite ordonnés selon un score calculé en fonction de la requête. Ce score combine le degré d'appartenance de l'attribut aux tableaux pertinents, le degré de pertinence des documents Web dont sont extraits les tables et des caractéristiques externes provenant de DBpedia, Wikipedia et des résultats de la recherche. La phase d'évaluation a montré que l'approche arrive à détecter de bons attributs pour une requête de type instance ou classe avec une précision à 10 de respectivement 0.83 et 0.95. Ces résultats concernent cependant seulement les noms des attributs. En ce qui concerne les valeurs des attributs, quelques règles à suivre pour les récupérer ont été proposées, mais les résultats restent insuffisants (66% des attributs pertinents retournés ont une ou plusieurs valeurs).

De son côté, GoogleLabs [4] a lancé il y a 2 ans un outil expérimental, intitulé Google Squared, qui permettait de générer un tableau descriptif pour une requête donnée de type classe (voir exemple sur la figure 5). Google Squared offrait plusieurs fonctionnalités, comme la possibilité de modifier le tableau en retirant les lignes et les colonnes inutiles, ou d'ajouter de nouvelles lignes et colonnes en lui laissant rechercher les faits pertinents. Malgré son originalité, cette dernière approche retournait souvent des tableaux troués et renvoyait parfois des attributs avec des valeurs erronées.

Un autre système, ListExtract, a été proposé par (Elmeleegy *et al.*, 2009). C'est une technique d'extraction de tableaux à partir de listes. Tout d'abord, de multiples sources d'informations sont utilisées pour segmenter les différentes lignes d'une liste en de multiples champs. Ensuite, selon le nombre de segments obtenus dans la majorité des lignes, le système décide du nombre de colonnes à générer dans le tableau et réorganise les mauvais segments. Finalement, en exploitant les tableaux du Web, le système affine le tableau obtenu en remplissant les cases vides par des valeurs correspondantes. Pour chaque table extraite, les auteurs calculent un score d'extraction qui reflète la qualité de la table produite. La technique est indépendante du domaine étudié sauf qu'elle ne s'appuie que sur des listes.

4. http://www.googlelabs.com: Le site GoogleLabs de Google est actuellement fermé, et il n'est malheureusement plus possible d'accéder à GoogleSquared.

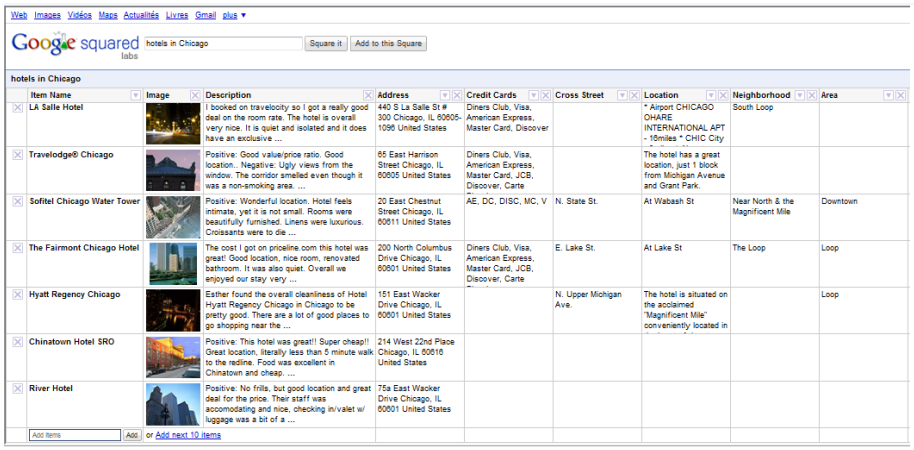

FIGURE *5. Résultats de la requête "hotels in Chicago" pour le moteur de recherche Google Squared.*

2.4. Problématique des travaux

Comme nous venons de le voir, les approches de l'état de l'art en recherche agrégée relationnelle s'intéressent principalement à la problématique de la recherche de nuggets pertinents (classes, instances, attributs, et leurs relations), mais peu d'approches (hors celle de Google Squared) ont considéré le problème de l'agrégation de ces nuggets : comment doivent-ils être présentés à l'utilisateur?

Dans cet article, nous nous intéressons à cette dernière problématique: plus particulièrement, nous nous intéressons à l'assemblage des instances et des attributs pour répondre à des requêtes relationnelles, en considérant plus particulièrement les requêtes classe. L'agrégation de résultats est plus ardue dans ce cas. Pour les requêtes attribut, il "suffit" de renvoyer des valeurs candidates triées par pertinence. Pour les requêtes instance, il faut retourner des attributs pertinents, mais aussi sélectionner quels sont les attributs les plus représentatifs. Pour les requêtes classe le problème est encore plus complexe, puisqu'il faut sélectionner et assembler les instances et attributs importants. L'ordre utilisé pour présenter les instances et les attributs importants peut affecter la qualité du résultat agrégé. Ce dernier peut prendre différentes formes, la plus intuitive étant celle du tableau, que nous utiliserons dans le reste de l'article.

Afin d'illustrer le problème de l'agrégation de résultats pour des requêtes classe, nous avons construit quelques exemples présentés sur la figure 6. Ils correspondent à des résultats possibles pour la requête *"villes françaises"*. Chaque ligne correspond à une instance et chaque colonne à un attribut.

Les 3 premiers tableaux contiennent des résultats non pertinents :

	Pays	Cartons jaunes	Joueurs de golf	Email
Paris	France	2	2000	paris@france.fr
Marseille	France	3	1541	
Roscoff	France	NA	21	
Mulhouse	France	5	56	

A

B

	Population	Région	Superficie	Maire
Londres	7 825 000	Londres	1 572 km2	Boris Johnson
Roscoff	3 705		6,19 km2	Joseph Seité
Castres	44 823	Midi-Pyrénées	457 km2	Pascal Bugis
Mulhouse	110 514	Alsace	22,18 km2	Jean Roffner

	Utilisateurs metro	Aéroport	Superficie	Surperficie metro
Paris	11 899 544	CDG, Paris orly	2 723 km2	14 518 km2
Roscoff	NA		6,19 km2	
Castres			457 km2	
Mulhouse			22,18 km2	

C

D

	Population	Région	Superficie	Maire
Paris	2 211 297	Île de France	2 723 km2	Bertrand Delanoë
Marseille	851 420	Provence	240 km2	Jean-Claude Gaudin
Nantes	283 025	Pays de la Loire	65 km2	Jean-Marc Ayrault
Toulouse	437 715	Midi-Pyrénées	118 km2	Pierre Cohen

FIGURE 6. *Exemples de résultats pour des requêtes classe*

– le tableau A a des attributs non pertinents (par exemple: *"email"*) et des attributs peu pertinents (*"joueurs de golf"*, *"cartons jaunes"*);

– le tableau B contient une instance non pertinente (*"Londres"*) et des villes peu représentatives (*"Mulhouse, Roscoff"*);

– les problèmes dans le tableau C viennent des valeurs des attributs. La plupart d'entre elles sont manquantes.

Le tableau D semble être le plus pertinent: il contient des instances (représentatives) de la classe et des attributs importants et pertinents, avec leurs valeurs. La qualité des résultats de recherche agrégée dépend de la qualité des instances, des attributs et de leurs valeurs. Idéalement, les instances et les attributs ne doivent pas être seulement pertinents mais aussi représentatifs pour la requête (ici la classe), et les valeurs d'attributs doivent être présentes et correctes.

Dans cet article, nous proposons une approche permettant de sélectionner des instances représentatives et de trier des attributs pertinents en réponse à une requête classe. Nous nous limitons aux requêtes de type classe en considérant que nous pouvons ramener les requêtes instance à la forme d'une classe en déterminant celle qui leur ressemble le plus. Notre approche permet de calculer différents scores de pertinence pour les attributs. Plusieurs méthodes de scores ont été évaluées, afin d'analyser leur impact sur le résultat final. Afin de limiter le nombre d'instances et d'attributs non pertinents et de se focaliser sur leur tri, nous utilisons le Web de Connaissance (et

plus particulièrement DBPedia) pour l'extraction des nuggets pertinents (instances et attributs).

3. Une approche pour l'agrégation d'information: comment construire des résultats tabulaires pertinents

Afin d'agréger les résultats d'une requête classe, une première étape est de rechercher les instances et les attributs liés à la classe. Il est ensuite nécessaire de *(i)* sélectionner les instances représentatives de la classe, et *(ii)* trier ses attributs. Nous nous focalisons ici sur ces deux derniers points.

La première de ce deux dernières étapes permet d'identifier les instances qui seront utilisées pour la construction du résultat final et le tri des attributs, et la seconde permet d'affecter un score aux attributs afin de ne conserver que les plus pertinents. Dans ce qui suit, nous considérons qu'étant donné une requête classe, nous disposons d'un ensemble d'instances de la classe, ainsi que d'un ensemble d'attributs (nom-valeur) pour chacune des instances de la classe.

3.1. Sélection des instances

Afin de classer les attributs les plus représentatifs d'une classe, nous partons de l'hypothèse suivante: les attributs représentatifs doivent être présents dans les instances "importantes". Le tri des attributs est donc très lié à celui des instances: qu'est ce qu'une instance "importante" ? Dans le but d'analyser l'impact des instances sur le classement des attributs, nous avons mis en place trois algorithmes de sélection d'instances.

Le premier, intitulé *Shorty*, sélectionne les n premières instances qui ont le moins d'attributs parmi celles qui ont au moins un attribut (voir algorithme 1). L'intérêt de cette sélection est de favoriser les attributs des instances qui ont peu d'attributs. L'hypothèse derrière ce choix est que le peu d'attributs présents pour ces instances sont forcément pertinents.

Le deuxième algorithme, intitulé *Maxy*, sélectionne les n premières instances qui ont le plus d'attributs. L'intérêt de cette sélection est de favoriser les attributs des instances surchargées et d'analyser leur impact sur la présentation des résultats. Nous adoptons ce choix car dans une instance qui contient beaucoup d'attributs, nous trouverons forcement des attributs représentatifs de la classe en question. L'algorithme est similaire à *Shorty*, il suffit de remplacer $\arg\min_{i_j \in C^*} (|i_j|)$ par $\arg\max_{i_j \in C} (|i_j|)$ ($|i_j|$ étant le nombre d'attributs de l'instance i_j).

Finalement le troisième algorithme, intitulé *Fantasy*, part de l'hypothèse que les instances bien référencées (donc populaires) sont importantes, et utilise certains résultats de l'état de l'art sur la recherche d'entités nommées (Balog *et al.*, 2009) (Vercoustre *et al.*, 2007). Au niveau de chaque instance de la classe c, il existe des valeurs d'attributs, elles-mêmes de type instance, qui pointent vers d'autres instances et inversement

(des instances qui pointent vers l'instance de la classe c à travers des valeurs d'attributs). Plus concrètement, nous sélectionnons les instances en fonction du nombre de liens entrant ($interLink$) vers la page associée de Wikipedia et du nombre de liens sortant ($extLink$) de cette page. Le calcul de cette sélection est similaire aux autres en remplaçant seulement $\arg\min_{i_j \in C^*} (|i_j|)$ par $\arg\max_{i_j \in C} (interLink + extLink)$.

Algorithm 1 Sélection des instances Shorty

soit $C = \{i_1, i_2, \ldots, i_z\}$ l'ensemble des instances qui appartiennent à la classe
$nbSelectedInstances \leftarrow n$
$Shorty \leftarrow \emptyset$
while $|Shorty| < nbSelectedInstances$ **do**
  $S_{select} = \arg\min_{i_j \in C^*} (|i_j|)$ avec $C^* = \{i_j, |i_j| > 1\}$
  $Shorty \leftarrow S_{select}$
  $C^* \leftarrow C^* - \{S_{select}\}$
end while

3.2. Tri des attributs

Nous devons, à ce niveau, trier les attributs en calculant un score $score\,(a)$ pour chaque attribut a de la classe c servant de requête afin d'identifier les attributs les plus pertinents pour l'agrégation. Sans cette étape, si nous agrégeons les attributs retournés sans les trier, les attributs les plus importants risquent d'être perdus et cachés dans l'ensemble. D'autre part, trier les attributs nous permet de minimiser les risques de valeurs non renseignées dans le tableau (les attributs associés au plus grand nombre d'instances seront privilégiés). Nous proposons de calculer le $score\,(a)$ de trois façons différentes.

– **Score basé sur la fréquence (1ère formule)**

En premier lieu, nous proposons d'utiliser la fréquence d'apparition des attributs dans la classe traitée ($tf\,(a, c)$) sur le nombre d'attributs dans la classe en question pour normaliser ($|c|$).

$$score\,(a) = \frac{tf\,(a, c)}{|c|} \tag{1}$$

Pour cette formule nous ne prenons pas en compte le classement des instances et considérons que toutes les instances ont le même impact. Nous favorisons ainsi les attributs qui se répètent dans la majorité des instances et les affichons devant ceux qui se répètent rarement. De cette façon nous espérons avoir moins de cases vides dans le tableau à afficher.

– **Score basé sur la probabilité de pertinence (2ème et 3ème formules)**

Une autre façon de faire est de considérer le score $score\,(a)$ comme une probabilité d'appartenance $P\,(a|c)$ à la classe recherchée c pour chaque attribut traité a.

En traitant les attributs comme des termes, les instances comme des documents et la classe comme la collection traitée, nous pouvons faire une analogie avec le principe de classement probabiliste (*Probability Ranking Principle*), énoncé par (Robertson,

1997). Dans notre cas, le mieux est de retourner les attributs a et les instances i en ordre décroissant de leur probabilité sachant la classe c.
Dans la suite, nous allons utiliser la formule de probabilité totale de cette façon :

$$score\,(a) = P\,(a|c)\ = \sum_{i \in c} P\,(a|i)\ \cdot P\,(i|c) \tag{2}$$

avec:

- $P\,(a|i)$: probabilité de pertinence de l'attribut a sachant l'instance i.
- $P\,(i|c)$: probabilité de pertinence de l'instance i sachant la classe c.

La probabilité $P\,(a|i)$ peut être calculée selon les deux formules suivantes :

- **Deuxième formule**

$P\,(a|i)$ est égale a l'inverse du nombre d'attributs $|i|$ dans l'instance i.

$$\begin{array}{ll} score\,(a) = P\,(a|c) & =\sum_{i \in c} \left(P\,(a|i)\ \cdot P\,(i|c)\right) \\ & = \sum_{i \in c} \left(\frac{1}{|i|} \cdot P\,(i|c)\right) \end{array} \tag{3}$$

Tous les attributs ont le même poids au sein de l'instance à laquelle ils appartiennent mais les attributs des instances ayant le moins d'attributs deviennent plus représentatifs que ceux des autres instances.

- **Troisième formule**

Pour cette troisième formule, nous utilisons le ratio de la fréquence $tf\,(a, c)$ de l'attribut a dans la classe c sur la somme des fréquences $tf\,(a_j, c)$ des attributs a_j de la classe c à laquelle appartient l'instance i en question. Ce qui donne comme formule finale:

$$\begin{array}{ll} score\,(a) = P\,(a|c) & =\sum_{i \in c} \left(P\,(a|i, c)\ \cdot P\,(i|c)\right) \\ & = \sum_{i \in c} \left(\frac{tf(a,c)}{\sum_{a_j \in i} tf(a_j,c)} \cdot P\,(i|c)\right) \end{array} \tag{4}$$

Cette formule combine l'intérêt des deux premières formules. En effet, elle donne plus d'importance aux attributs fréquents dans la classe et moins à ceux qui appartiennent aux instances possédant beaucoup d'attributs.

Selon les trois sélections déterminées dans la section 3.1, la probabilité $P\,(i|c)$ est calculée selon l'appartenance de l'instance i à la sélection traitée I (*Shorty, Maxy* ou *Fantasy*). Dans notre étude, nous souhaitons que $P\,(i|c)$ pour $i \in I$ soit le double de $P\,(i|c)$ pour $i \notin I$ (ce choix est fait à des fins expérimentales, dans le seul but de favoriser les attributs des instances de la sélection traitée).

Après calcul, ces valeurs sont estimées comme suit:

$$\begin{cases} P\,(i|c) = 2/(|Inst(c)| + |Inst(I)|) & si\ i \in \{I\} \\ P\,(i|c) = 1/(|Inst(c)| + |Inst(I)|) & sinon \end{cases} \tag{5}$$

avec

– I: l'ensemble des instances à valoriser qui peut être *Shorty, Maxy* ou *Fantasy*.

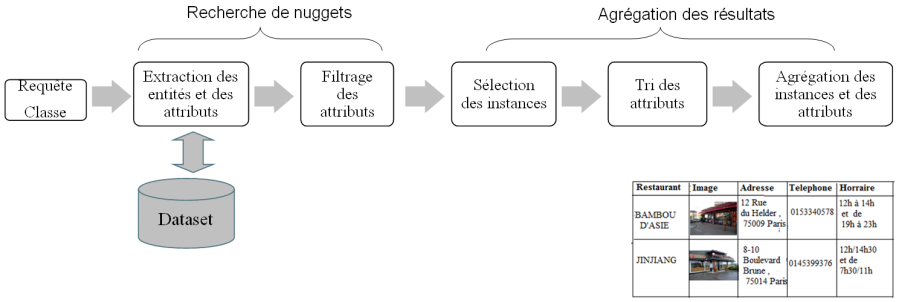

FIGURE 7. *Les différentes étapes pour l'agrégation des résultats*

– $|Inst(c)|$ et $|Inst(I)|$: le nombre d'instances dans la classe c et la sélection traitée I

Les probabilités vérifient que $\sum_{i \in c} P(i|c) = 1$.

Une fois que les données sont collectées et triées pour une requête classe, nous les agrégeons dans un tableau.

4. Evaluation

4.1. Mise en place du système pour l'évaluation

Afin d'évaluer notre approche pour l'agrégation des résultats, nous avons mis en place une chaîne de traitement basée sur Wikipedia et DBPedia, dont le fonctionnement est décrit sur la figure 7.

La requête est tout d'abord soumise au *dataset* DBPedia. Afin de rechercher les informations pertinentes, une première étape consiste à extraire les nuggets (instances et attributs) correspondant à la requête. La quantité importante d'informations obtenues nécessite une seconde étape de filtrage des attributs. L'agrégation des nuggets est ensuite effectuée en 3 étapes: une étape de sélection des instances importantes qui participeront ensuite au tri des attributs, le tri des attributs en lui-même et l'agrégation des instances et des attributs obtenus pour construire notre résultat final, sous forme de tableau. Les étapes 1,2 et 5 sont détaillées dans ce qui suit. Les étapes 3 et 4 quant à elles sont basées sur ce que nous avons présenté en section 3.

4.1.1. Extraction des instances et des attributs

Cette première étape consiste à collecter les instances et les attributs potentiellement pertinents pour une requête de type classe. Ces informations sont, dans notre

étude, collectées à partir de Wikipedia et plus particulièrement de DBpedia [5]. Une seule source est ainsi utilisée pour extraire les instances, mais une combinaison de plusieurs pourrait être bien sûr envisageable. Notre but ici n'est pas d'évaluer la qualité de l'extraction des instances, mais plutôt l'évaluation des résultats agrégés (bonne présentation des instances et des attributs).
Wikipédia, encyclopédie libre, collaborative et multilingue [6], contient de l'information compréhensible, de haute qualité et bien structurée. Parmi ces informations structurées, on peut citer les catégories et les listes, l'accès à l'historique et les infobox [7] (Medelyan *et al.*, 2009).
Nous avons choisi d'aborder les données de Wikipedia au travers de DBpedia. DBpedia convertit le contenu de Wikipedia en des données structurées en utilisant les techniques du Web sémantique (Auer *et al.*, 2007). A partir de DBpedia, nous récupérons toutes les instances qui appartiennent à la requête c'est-à-dire à la classe c en question. Pour ce faire, nous avons utilisé un des moyens d'accès à DBpedia, SPARQL Endpoint, qui consiste à envoyer une requête SPARQL au service offert par DBpedia (http://dbpedia.org/sparql), qui renvoie ensuite un document comportant des liens vers les instances de la classe c sous forme de triplets RDF. On trouvera un exemple de requête sur la figure 8.

```
SELECT distinct ?entities ?label
            WHERE {
   {?entities dcterms:subject  <http://dbpedia.org/resource/Category:AdventureNovels>}
      UNION
         {?entities rdf:type <http://dbpedia.org/ontology/AdventureNovels>}
UNION
   {?entities rdf:type  <http://dbpedia.org/class/yago/AdventureNovels>}
      ?entities rdfs:label ?label.
FILTER ( lang(?label) = 'en' )
                        }
```

FIGURE *8. Exemple de requête SPARQL récupérant les instances correspondant à la classe 'AdventureNovels'*

Pour augmenter le nombre d'instances renvoyées pour chaque requête c, nous interrogeons les 3 types de catégorisation offerts sous DBpedia qui sont :

– YAGO (Suchanek *et al.*, 2007), qui contient 286000 classes qui forment une arborescence hiérarchique. C'est une ontologie généraliste de grande taille et extensible qui s'appuie sur les instances et attributs extraits à partir de Wikipedia.

5. http://wiki.dbpedia.org
6. http://en.wikipedia.org/wiki/History_of_Wikipedia
7. Modèle qui permet d'insérer du texte ou du code wiki dans une page, de manière automatisée, dynamique et configurable

– Skos [8], qui contient 415000 classes qui sont les catégories offertes par le système de Wikipedia. Le problème avec cette catégorisation est qu'elle représente une faible relation de parenté entre les articles (les instances).

– OWL-DBpedia [9], qui contient 205 classes, sous forme d'une hiérarchie peu profonde, et 1210 propriétés construites manuellement à partir de 350 infobox les plus utilisés au niveau de la version anglaise de Wikipedia.

Les instances résultats (voir figure 9 pour un exemple), ainsi renvoyées pour la classe, sont ensuite utilisées pour extraire les attributs de la classe et leurs valeurs. Pour cela, nous utilisons un autre moyen d'accès à DBpedia, "Data Link", qui consiste à accéder directement au lien de l'instance concernée. Ce lien constitue un identifiant de l'instance au sein de la source DBpedia. Par exemple "http://dbpedia.org/page/Treasure _Island" est un identifiant unique pour l'instance *"Treasure_Island"*. Les résultats affichés proviennent de triplets RDF décrivant l'instance.

SPARQL results:

entities	label
:Adventures_of_Huckleberry_Finn	"Adventures of Huckleberry Finn"@en
:Moonfleet	"Moonfleet"@en
:The_Farther_Adventures_of_Robinson_Crusoe	"The Farther Adventures of Robinson Crusoe"@en
:Treasure_Island	"Treasure Island"@en
:The_Toll-Gate	"The Toll-Gate"@en
:Rupert_of_Hentzau	"Rupert of Hentzau"@en
:Peter_Duck	"Peter Duck"@en
:Coot_Club	"Coot Club"@en
:Missee_Lee	"Missee Lee"@en
:Swallowdale	"Swallowdale"@en
:White_Fang	"White Fang"@en

FIGURE *9. Exemple d'instances résultats pour la classe 'AdventureNovels'*

Chaque instance de DBpedia, reliée à article de Wikipedia, est décrite par un ensemble de propriétés générales et un ensemble de propriétés infobox-spécifiques (Bizer *et al.*, 2009). Ces propriétés sont utilisées pour extraire les attributs des instances et leur valeur. Les propriétés spécifiques sont spécifiques à l'instance en question, trouvées généralement dans l'infobox. Les propriétés générales sont celles qui se répètent dans toutes les instances quelque soit leur nature. Elles incluent une étiquette, un court et un long résumé en anglais, un lien à l'article correspondant de Wikipedia, des geo-coordonnées (si disponibles), un lien à une image de l'instance, des liens aux pages Web externes, des liens internes aux instances de DBpedia, etc.... Si une instance est décrite en plusieurs langues, alors il aura des résumés courts et longs dans ces langues.

8. http://www.w3.org/2004/02/skos/
9. http://wiki.dbpedia.org/Ontology

4.1.2. Filtrage des attributs

Les propriétés génériques et spécifiques, décrites dans la section précédente, contiennent un nombre important d'informations utiles et informatives sur les instances en question. Elles comportent cependant également de l'information répétitive, inutile, de mise en forme et de catégorisation que l'utilisateur n'a pas besoin de consulter. En effet, nous pouvons trouver la même valeur reliée à plusieurs attributs soit de noms différents (*birthdate* et *birthDate*), soit de niveaux différents (http://dbpedia.org/ontology/Place/location et http://dbpedia.org/ontology/location), soit carrément d'origines différentes (http://dbpedia.org/ontology/location et http://dbpedia.org/property/location). Une étape de filtrage est donc nécessaire. Elle consiste tout d'abord à ne garder que les attributs en anglais (car nous ne produisons que des documents en anglais). Ensuite, nous éliminons les attributs et les valeurs redondants, les attributs de mise en forme (*wikiPageUsesTemplate*) et de catégorisation (*rdf:type, skos:subject*) ainsi que les attributs vides [10] comme *redirect, display, disambiguates, same as*. Nous éliminons aussi les propriétés venues de ressources externes comme le dataset geo-name [11] et GeoRSS [12]. L'étape de filtrage permet de minimiser le nombre d'informations non pertinentes et d'augmenter les chances des propriétés (attributs) spécifiques, importantes et avec un caractère dominant à être classées devant les propriétés générales. Le classement des attributs est détaillé dans la section 3.2.
A l'issue de cette étape de filtrage, nous obtenons pour chaque instance de la requête classe c un ensemble d'attributs avec leurs valeurs. Les instances représentatives sont ensuite sélectionnées (voir section 3.1), et un score de pertinence est attribué aux attributs (voir section 3.2). La dernière étape est ensuite d'agréger ces données.

4.1.3. Agrégation des données

Pour présenter les attributs extraits et triés selon les formules 1 à 4, nous adoptons la forme la plus adéquate et la plus lisible à notre sens pour l'utilisateur, celle des tableaux. Ces tableaux contiennent, pour une requête de type classe, dans chaque ligne une instance décrite par des attributs situés dans les colonnes.
Nous plaçons les cinq premières instances de la sélection Maxy dans les lignes et les dix attributs les plus importants selon les équations 1 à 4 dans les colonnes [13] (voir figure 10).

Nous avons choisi d'afficher systématiquement les instances de la sélection Maxy car elle est censée être la sélection qui contient le plus d'attributs; cela limitera donc le risque de présenter des cellules vides à l'utilisateur. Dans chaque cellule du tableau nous pouvons trouver soit une valeur atomique comme le nom, la langue, la date de naissance, soit une liste de valeurs (comme par exemple les références). Une cellule

10. Une liste d'attributs définie manuellement que nous jugeons inutiles.
11. Formulation RDF d'un système géodésique associé au GPS.
12. Moyen communautaire, léger, pour à étendre les informations géographiques.
13. Les valeurs 5 et 10 sont prises à titre expérimental.

	comment	Image	family	order	class	hasPhotoCollection	kingdom	genus	division	reference
Avocado46	The avocado (Persea americana), aguacate,...		Entité:Lauraceae	Entité:Laurales	Magnoliids Entité:Magnoliid	http://www4.wiwiss.fu-berlin.de/flickrwr...	Entité:Plant	Persea Entité:Persea		http://www.cefg.org/pubs/ff/avocado.html http://www.avocado.org.au/ http://avocadosource.com/ http://www.fao.org/es/ess/top/commodity.... http://www.avocados.es/ ...
Pineapple43	Pineapple (Ananas comosus) is the common....		Entité:Bromalioideae Entité:Bromeliaceae	Entité:Poales Entité:Commelinids	Entité:Monocots	http://www4.wiwiss.fu-berlin.de/flickrwr...	Entité:Plantae	Ananas Entité:Ananas	Entité:Angiosperms	https://scholarspace.manoa.hawaii.edu/bi... http://www.cefg.org/pubs/ff/pineapple.ht...
Eggplant44	The eggplant or aubergine (Solanum melon....		Entité:Solanaceae	Entité:Asterids Entité:Solanales	Entité:Eudicots	http://www4.wiwiss.fu-berlin.de/flickrwr...	Entité:Plantae	Solanum Entité:Solanum	Entité:Angiosperms	http://www.nhm.ac.uk/research-curation/p... http://www.nhm.ac.uk/research-curation/r... http://faostat.fao.org/site/567/DesktopD... http://www.learn2grow.com/plantdatabase/... http://www.pfaf.org/database/plants.php?...
Pomegranate44	A pomegranate is a fruit-bearing deciduo....		Entité:Lythraceae	Entité:Myrtales	Entité:Rosidae Entité:Magnoliopsida	http://www4.wiwiss.fu-berlin.de/flickrwr...	Entité:Plant	Entité:Punica Punica	Entité:Flowering_plant	http://www.pomegranate.me.uk/ http://www.cefg.org/pubs/ff/pomegranate.... http://www.jewishrefplace.com/Symbolism... http://altmedicine.about.com/od/druenate... http://www.pomegranates.org/home.shtml
Banana42	Banana is the common name for herbaceous....		Entité:Musaceae	Entité:Zingiberales Entité:Commelinids	Entité:Monocots	http://www4.wiwiss.fu-berlin.de/flickrwr...	Entité:Plantae	Entité:Musa_%28genus%29	Entité:Angiosperms	http://www.sita.org/ems/details/banana_p... http://books.google.com/books?id=eSoVd-o... http://www.nytimes.com/2008/06/18/opinio... http://permaculture.org.au/2008/06/23/ba...

FIGURE *10. Tableau résultat de la requête "Tropical_fruit"*

peut également être aussi vide (sans valeur) si DBpedia n'a pas la valeur correspondante à cet attribut de l'instance en question.

4.2. Protocole expérimental

4.2.1. Choix des requêtes et construction des tableaux

Afin d'évaluer notre approche, nous avons tout d'abord défini 30 requêtes (classes) qui sont soumises à DBpedia comme indiqué dans la section 4.1.1. Ces requêtes sont listées ci-dessous (voir figure 11). Nous les avons sélectionnées parmi les trois catégorisations Yago, Skos et OWL-DBpedia. Ces classes ont été choisies en s'appuyant sur des articles de Wikipedia. Les classes devaient également comporter entre 20 et 100 instances.

Pour chaque classe soumise à DBpedia, nous avons extrait les instances, attributs et valeurs selon les étapes définies précédemment, puis les résultats sont présentés sous forme d'un tableau. Les 3 sélections d'instances (*Shorty, Maxy, Fantasy*) et les 3 formules de tri d'attributs proposées nous permettent de construire 9 tableaux, dont la description est donnée ci-après.

Chacun des tableaux contient 5 instances de la sélection Maxy (cf. section 4.1.3). Seuls les attributs et leur tri varient d'un tableau à l'autre. Pour la première formule (équation 1), nous construisons un seul tableau qui ne prend pas en compte les sélections d'instances que nous avons évoquées dans la section précédente pour le tri des attributs. Pour la deuxième et la troisième formule (équations 3 et 4), le tri des attributs se fait en calculant la probabilité $P\,(a|c)$ selon les trois sélections d'instances *Shorty, Maxy, Fantasy*, avec un quatrième cas qui ne prend pas en considération le classement des instances ($P(i|c)$ est toujours égal à 1). De cette façon, nous aurons quatre tableaux pour chaque formule. Nous obtenons donc à la fin neuf tableaux pour chaque requête c'est-à-dire pour chaque classe.

Seas	VIA_Rail_stations_in_ontario
Chancelor	Parts_and_harbours_of_New_Zealand
ArchipelagoesofIndonesia	Albums_produced_by_Mick_Ronson
Colour	Actors_from_Los_Angeles%2C_California
Eureopean_Union_member_states	Manga_of_1999
EnglishJournalists	Defunct_agencies_of_the_United_States_government
Programmable_calculators	Towns_in_Utah
Stoner_rock_musical_group	Burgess_Shale_fossils
Castles_in_Poland	Academics_of_the_University_of_Kent
Number-one_debut_singles	Markup_languages
Turkish_Riveria	Computing_acronyms
Radiocontrast_agents	Tropical_fruit
Nissan_vehicules	Countries_bordering_the_Atlantic_Ocean
Anstralian_World_War_I_battalions	AdventureNovels
Poker_companies	G20_nations

FIGURE *11. Liste des classes utilisées pour l'évaluation*

4.2.2. Procédure d'évaluation

Notre but est d'évaluer la qualité des tableaux construits. Il n'existe cependant pas de mesure spécifique pour estimer cette qualité. Nous avons donc décidé d'avoir recours aux jugements humains et aux mesures d'évaluation adoptées en RI classique. Pour ce faire, nous avons mis en place une interface appropriée [14]. Elle affiche pour chaque requête (classe) posée,

– d'abord, tous les attributs de la classe afin de juger de leur pertinence pour la requête (voir figure12) ,

– et ensuite les neuf tableaux correspondant aux différentes formules et sélections.

A l'aide de cette interface, nous avons demandé à quatre volontaires de tester notre approche en donnant à chacun dix requêtes parmi les trente choisies antérieurement [15]. Chaque volontaire devait, pour chaque requête, cocher parmi les attributs affichés ceux qui lui semblaient pertinents pour cette dernière. Ensuite il devait sélectionner le tableau qui semblait le plus représentatif de son besoin.

Nous avons adopté dans notre étude deux types d'évaluation.
Nous nous sommes basés, tout d'abord, sur les observations des volontaires et leur jugement personnel (c'est-à-dire leur choix du meilleur tableau présenté). Nous avons aussi utilisé deux mesures d'évaluation qui sont la précision à dix (P@10) pour évaluer

14. Cette interface, développée en Java, accède en ligne à DBPedia et construits les résultats à la volée.
15. Certaines requêtes ont été jugées en double.

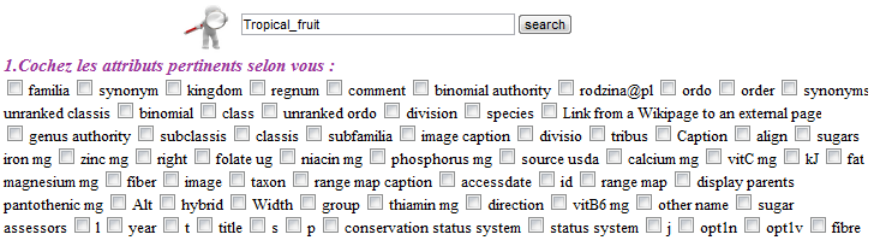

FIGURE *12. Ensemble d'attributs pour une requête donnée*

la capacité du système à afficher des attributs pertinents [16] et la précision moyenne ou MAP(*Mean Average Precision*) pour évaluer l'efficacité du système en terme de tri des attributs [17].

4.3. Résultats

Notre première évaluation tente de montrer les préférences (choix) des utilisateurs sur les tableaux renvoyés indépendamment de l'algorithme de sélection. La table 1 montre que 60% des utilisateurs choisissent les tableaux formés à partir de la troisième formule. La prise en compte de la fréquence de l'attribut dans la classe et l'instance est donc plus bénéfique.

Tableau 1. Choix des utilisateurs pour les formules sélectionnées

	Formule 1 (équation 1)	Formule 2 (équation 3)	Formule 3 (équation 4)
Pourcentage de sélection	7.5%	32.5%	60%

Afin d'approfondir notre étude, nous nous intéressons maintenant aux sélections d'instances adoptées dans la troisième formule. Le tableau 2 nous montre que la sélection *Fantasy* avec ses 41,8% est la plus choisie par les utilisateurs suivie par la sélection *Maxy* avec 33,3% alors que *Shorty* n'a récolté que 8.3%.
Ceci signifie que les instances les plus importantes en terme de liens et celles qui ont le plus d'attributs permettent de favoriser l'apparition des attributs représentatifs. Par contre, les instances qui n'ont pas beaucoup d'attributs n'ont pas un véritable impact sur le classement des attributs.

16. Dans notre cas, la précision à 10 est le rapport entre le nombre d'attributs pertinents parmi les 10 premiers renvoyés.

17. La précision moyenne est la moyenne des valeurs de précision à chaque attribut pertinent de la liste ordonnée.

Tableau 2. Choix des utilisateurs pour les différentes sélections d'instances au niveau de la troisième formule

	Sans sélection	Shorty	Maxy	Fantasy
Pourcentage de sélection	16.6%	8.3%	33.3%	41.8%

Nous avons ensuite évalué la précision à 10 et la MAP pour la liste d'attributs renvoyés pour chacune des sélections et formules de tri. Le tableau 3 liste ces résultats. Nous constatons que l'approche est capable de renvoyer au moins cinq attributs pertinents parmi les dix (P@10>0,5). Cependant, la troisième formule offre le meilleur résultat avec 0,595 de P@10 en moyenne pour les sélections *Maxy* et *Fantasy* alors que la première formule ne permet d'obtenir que 0,53. Ces résultats confirment les observations faites par les usagers et nous montrent que la troisième formule permet d'obtenir de meilleures performances que la première.
Considérons maintenant les sélections. La sélection *Fantasy* associée aux deux dernières formules offre le meilleur tri des attributs avec 0,489 de MAP pour la troisième formule et 0,477 pour la deuxième formule. Cela soutient, à nouveau, les choix des usagers et montre que les instances importantes en termes de liens classent les attributs pertinents mieux que les autres. La MAP la plus faible est obtenue sans sélection des instances dans la majorité des formules, ce qui pourrait signifier que *les instances ont un impact sur le classement des attributs*.
Les résultats concernant la sélection *Shorty* pourraient surprendre: utiliser cette sélection permet d'obtenir plus d'attributs pertinents que sans aucune sélection des entités, mais les tables générées à partir de la sélection sont moins plébiscitées par les utilisateurs (voir tableau 2). Cela peut être expliqué par le fait que les tables générées à partir de la sélection *Shorty* contiennent beaucoup de valeurs manquantes d'attributs. Ceci confirme que *la qualité des tables dépend du nombre d'attributs pertinents (nom et valeur) qu'elles contiennent*.
La majorité des formules ont une MAP au-dessous de 0,5 même si P@10 permet d'obtenir relativement de bonnes performances. Une telle constatation nous conduit à conclure que les attributs pertinents sont bien parmi les dix premiers mais pas forcément bien placés dans les tableaux ce qui nous pousse à améliorer davantage notre approche.

En conclusion de ces évaluations, nous avons montré que les résultats d'agrégations récupérés par notre approche sont globalement satisfaisants. D'autres évaluations sont cependant nécessaires pour confirmer les résultats et l'impact des formules utilisées pour le calcul de la pertinence des attributs. Nous souhaiterions également estimer l'impact du nombre d'instances d'une classe sur le résultat.

5. Conclusion et perspectives

Le travail présenté dans cet article s'inscrit dans le cadre de la recherche agrégée relationnelle. Nous nous sommes intéressés à l'agrégation de résultats pour des requêtes de type classe. Etant donnés des instances et des attributs de la classe requête,

Tableau 3. P@10 moyenne et MAP moyenne pour chaque formule et chaque sélection

		p@10	MAP
Formule 1 (équation 1)	Sans sélection	0,53	0,425
Formule 2 (équation 3)	Sans selection	0,58	0,473
	Shorty	0,57	0,47
	Maxy	0,54	0,456
	Fantasy	0,541	0,477
Formule 3 (équation 4)	Sans selection	0,58	0,37
	Shorty	0,583	0,45
	Maxy	0,595	0,475
	Fantasy	0,595	0,489

nous avons cherché à trier les attributs en fonction des instances les plus représentatives, afin de présenter les résultats dans un tableau.
Notre protocole expérimental, basé sur 30 requêtes et les instances et attributs associés extraits de DBPedia, nous permet de conclure que les instances utilisées pour le calcul du poids de pertinence des attributs ont une importance et que le choix des fonctions utilisées pour pondérer les attributs impacte significativement les résultats. La sélection d'instances *Fantasy* (basée sur les liens entrant et sortant de l'instance dans Wikipedia) semble en outre la plus intéressante dans notre cas. Enfin, notre approche nous permet de sélectionner jusqu'à 6 attributs pertinents parmi 10 dans les tableaux. Ces résultats, globalement satisfaisants, peuvent bien sûr toujours être améliorés.

Le travail réalisé dans cet article ouvre diverses perspectives parmi lesquelles la mise en place d'un module de filtrage et de vérification de valeurs des cellules du tableau. Nous souhaiterions également utiliser les ontologies reliées a DBpedia comme Yago et OWL-DBpedia pour améliorer la qualité et la lisibilité des données représentées dans les tableaux. Enfin, à plus long terme, notre approche sera intégrée dans un système plus généraliste, qui combinera l'extraction des instances et attributs via DPBedia avec d'autres sources d'extraction plus générales (comme les pages Web (Kopliku *et al.*, 2011)) , permettant ainsi de répondre à des requêtes dont la réponse n'est pas présente dans Wikipedia.

Bibliographie

Auer S., Bizer C., Kobilarov G., Lehmann J., Cyganiak R., Ives Z. G. (2007). Dbpedia: A nucleus for a web of open data. In *In proceedings of iswc/aswc*, p. 722-735.

Balog K., Vries A. P. de, Serdyukov P., Thomas P., Westerveld T. (2009, November). Overview of the trec 2009 entity track. In *Trec 2009 working notes.* NIST.

Bizer C., Lehmann J., Kobilarov G., Auer S., Becker C., Cyganiak R. *et al. (2009). Dbpedia - a crystallization point for the web of data.* Journal of Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 7, n° 3, p. 154-165.*

Boughanem M., Savoy J. (2008). Recherche d 'information. etat des lieux et perspectives. *Lavoisier.*

Cafarella M. J., Banko M., Etzioni O. (2006). Relational web search. *Rapport technique. University of Washington.*

Cafarella M. J., Halevy A. Y., Wang D. Z., Wu E., Zhang Y. (2008). Webtables: exploring the power of tables on the web. In Proceedings of VLDB*, vol. 1, n° 1, p. 538-549.*

Cafarella M. J., Halevy A. Y., Zhang Y., Wang D. Z., 0002 E. W. (2008). Uncovering the relational web. In In proceedings of webdb.

Elmeleegy H., Madhavan J., Halevy A. Y. (2009). Harvesting relational tables from lists on the web. In Proceedings of VLDB*, vol. 2, n° 1, p. 1078-1089.*

Etzioni O., Cafarella M. J., Downey D., Popescu A.-M., Shaked T., Soderland S. et al. (2005). Unsupervised named-entity extraction from the web: An experimental study. *Artif. Intell.*, vol. 165, n° 1, p. 91-134.

Haas K., Mika P., Tarjan P., Blanco R. (2011). Enhanced results for web search. In *In proceedings of sigir*, p. 725-734.

Ji L., Yan J., Liu N., Zhang W., Fan W., Chen Z. (2009). Exsearch: a novel vertical search engine for online barter business. In *In proceedings of cikm*, p. 1357-1366.

Kato M. P., Ohshima H., Oyama S., Tanaka K. (2009). Query by analogical example: relational search using web search engine indices. In *In proceedings of cikm*, p. 27-36.

Kopliku A. (2009). Aggregated search: From information nuggets to aggregated documents. In *Coria*, p. 495-502.

Kopliku A. (2011). *Approaches to implement and evaluate aggregated search*. Thèse de doctorat non publiée.

Kopliku A., Boughanem M., Pinel-Sauvagnat K. (2011). Towards a framework for attribute retrieval. In *In proceedings of cikm*, p. 515-524.

Medelyan O., Milne D. N., Legg C., Witten I. H. (2009). Mining meaning from wikipedia. *International Journal of Human-Computer Interactions*, vol. 67, n° 9, p. 716-754.

Murdock V., Lalmas M. (2008). Workshop on aggregated search. *SIGIR Forum*, vol. 42, n° 2, p. 80-83.

Paris C., Wan S., Thomas P. (2010). Focused and aggregated search: a perspective from natural language generation. *Information Retrieval*, vol. 13, n° 5, p. 434-459.

Robertson S. E. (1997). The probability ranking principle in ir. In, p. 281–286. Readings in information retrieval, Morgan Kaufmann Publishers Inc. Consulté sur http://portal.acm.org/citation.cfm?id=275537.275701

Sauper C., Barzilay R. (2009). Automatically generating wikipedia articles: A structure-aware approach. In *In proceedings of acl/afnlp*, p. 208-216.

Suchanek F. M., Kasneci G., Weikum G. (2007). Yago: a core of semantic knowledge. In *In proceedings of www conference*, p. 697-706.

Vercoustre A.-M., Pehcevski J., Thom J. A. (2007). Using wikipedia categories and links in entity ranking. In *In proceedings of inex*, p. 321-335.