Regrouper des résultats SPARQL par comparaison de leurs contenus tels qu'ils sont agencés dans la base RDF interrogée.

Sonia Djebali — Thomas Raimbault

Pôle Universitaire Léonard De Vinci
De Vinci Research Center (DVRC) – Digital Group – ESILV
Paris La Défense, France
sonia.djebali@devinci.fr, thomas.raimbault@devinci.fr

RÉSUMÉ. *Cet article présente une nouvelle approche permettant de regrouper les résultats d'une requête SPARQL selon leurs similitudes. Afin de comparer les résultats, l'originalité de notre approche est de considérer pour chaque résultat les données constituant ce résultat telles qu'elles sont présentes et agencées dans la base RDF interrogée. Nous ne nous limitons donc pas à comparer les résultats entre eux, mais nous les re-contextualisons dans la base où ils ont été sélectionnés afin de tenir compte non seulement des ressources des résultats mais aussi de leurs liaisons avec les autres ressources dans la base, c'est à dire leurs voisinages. À partir des résultats et de leurs voisinages, nous déterminons des motifs descriptifs communs permettant de grouper de manière hiérarchique les résultats par degré de similitude, offrant ainsi à l'utilisateur une navigation et une compréhension plus aisées des résultats.*

ABSTRACT. *This paper proposes a novel approach to group SPARQL query results according to their similarities. In order to compare results, the originality of our approach is to consider for each result the data constituting this result such as they are present and arranged in the queryed RDF dataset. We therefore do not limit ourselves to comparing the results between them, but we recontextualize them in the dataset where they were selected in order to take into account not only the resources of results but also their links with others resources in the dataset, i.e. their neighborhoods. From the results and their neighborhoods, we define common descriptive patterns to hierarchically group results in order of similarity, thus providing the user with easier navigation and understanding of the results.*

MOTS-CLÉS : *SPARQL, clustering query results, motif de description, hierarchical clustering.*

KEYWORDS: *SPARQL, clustering query results, descriptive pattern, hierarchical clustering.*

1. Introduction

Le Web – au sens général du World Wide Web – est aujourd'hui un espace incontournable en matière de partage de l'information, où il y a peu ou pas de barrières entre la publication et l'accès aux documents. Afin d'offrir une bonne (ré-)utilisation de l'information, le Linked Data est une initiative du W3C visant à favoriser des méthodes communes, calculables par une machine, pour stocker, partager, rechercher et combiner des données (Bizer *et al.*, 2009). Les données sont structurées au sein de documents RDF [1] – *Resource Description Framework* – où les ressources décrites sont interconnectées les unes avec les autres pour former un grand réseau global d'informations. Le *framework* RDF utilise un formatage des données sous forme de triplets pour décrire les ressources, et des IRIs[2] comme identifiants globaux pour les ressources. SPARQL [3] est le standard du W3C permettant d'interroger et de mettre à jour des données RDF disponibles à travers internet.

Aujourd'hui, le Linked Open Data (LOD) – désignant la partie « ouverte » d'un point de vue licence du Linked Data – constitue une grande masse d'informations, gérées principalement de façon communautaire, et regroupe des milliards de triplets RDF répartis sur des centaines de bases RDF publiquement accessibles, dont la fameuse base DBPedia[4]. Face à cette grande quantité d'informations (toujours en augmentation) et à la grande variété des données et de leurs relations, l'interrogation du LOD par des requêtes SPARQL génère généralement un grand nombre de résultats difficiles à analyser par l'utilisateur. Ce problème de surcharge d'informations ("*information overload*" en anglais) pose de nouveaux[5] défis pour améliorer l'accès aux données du LOD. Un premier axe de travail pour répondre à cette problématique est d'*ordonner* les résultats selon leur « importance » (définie par un score). En effet, dans la mesure où l'utilisateur associe intuitivement les premiers résultats présentés comme les plus pertinents, plusieurs approches se sont intéressées au classement des résultats. Les algorithmes et méthodes de *ranking* utilisés pour le Web traditionnel (Brin et Page, 1998) ont été adaptés au LOD, tels que DBpediaRanker (Mirizzi *et al.*, 2010), Dataset rankING (Delbru *et al.*, 2010), SPRING (Mulay et Kumar, 2011), COLINA (Feyznia *et al.*, 2014), et PageRank pour Wikipedia (Thalhammer et Rettinger, 2016). D'autres travaux, tels que BANKS (Bhalotia *et al.*, 2002) and Objectrank (Balmin *et al.*, 2004), ont eux été adaptés des méthodes de recherche par mot-clés utilisées en base de données relationnelles.
Un deuxième axe de travail pour répondre à cette problématique de surinformation est de *grouper* les résultats par « similitude », de sorte que les éléments d'un même groupe, appelé *cluster*, soient plus similaires les uns aux autres qu'à ceux des autres

1. RDF 1.1 Concepts and Abstract Syntax, `http://www.w3.org/TR/rdf-concepts/`
2. Comme pour une URL qui permet un accès standard à un document sur le Web, un IRI – *Internationalized Resource Identifier* – fournit un accès unique à une ressource dans le LOD.
3. The W3C SPARQL Working Group, `http://www.w3.org/TR/sparql11-overview/`
4. `http://wiki.dbpedia.org/`
5. Les premiers défis du LOD étaient (et sont toujours) la création/adaptation de bases au format RDF et la mise à disposition de ces bases.

clusters (Berkhin, 2006 ; Kaufman et Rousseeuw, 2009). En effet, lorsque la requête contient des termes généraux ou ambigus – car possédant plusieurs sens comme "Avocat" ou "Jaguar" – les résultats sont très diverses (en plus d'être nombreux) ce qui les rend difficiles à interpréter. L'organisation des résultats par groupe selon certaines similitudes peut ainsi apporter une aide précieuse à l'utilisateur. Bien que la clause GROUP BY existe en SPARQL, permettant comme en base de données relationnelle de grouper les résultats en fonction d'opérations d'agrégation, elle s'avère souvent limitée à l'usage dans le cadre du LOD où la structuration des données est bien plus diversifiée qu'en base de données classique. En effet, pour pouvoir utiliser une telle clause, l'utilisateur (i) doit savoir *a priori* – lors de la rédaction de sa requête – ce sur quoi il veut grouper les résultats, (ii) ne souhaite pas nécessairement utiliser un opérateur d'agrégation ; de plus (iii) l'ordre des paramètres de la clause a une forte influence sur les groupes générés, et (iv) il est difficile pour l'utilisateur de connaître le schéma des données (*i.e.* l'ontologie). Afin de lever ces trois dernières limitations, certaines approches inspirées des moteurs de *clustering* pour le Web traditionnel (Carpineto *et al.*, 2009) – *e.g.* Carrot2[6] et Yippy[7] – ont été adaptées au LOD, telles que Asparagus (Lawrynowicz *et al.*, 2011) et (d'Amato *et al.*, 2010) qui introduisent la clause CATEGORIZE BY et groupent les résultats en se basant sur les types des ressources des résultats et sur les liens hiérarchiques entre ces types fournis par l'ontologie de la base RDF. L'approche (Alam et Napoli, 2014) quant à elle introduit la clause VIEW BY et se base sur une analyse formelle de concepts (Ganter et Wille, 1999) afin d'en générer un treillis de Galois à partir duquel les résultats sont groupés.

Cet article présente une nouvelle approche pour répondre à la problématique de comment grouper de manière pertinente les résultats d'une requête SPARQL selon leurs similitudes. Afin de comparer les résultats, l'originalité de notre approche est de considérer pour chaque résultat les données constituant ce résultat telles quelles sont présentes et agencées dans la base RDF interrogée. Ainsi, nous pouvons tenir compte non seulement des résultats eux-mêmes – en tant que sous-graphe de la base interrogée – mais aussi de leurs liaisons avec les autres données dans la base, c'est à dire de leurs *voisinages*. À partir des résultats et de leurs voisinages, nous déterminons des *motifs descriptifs* communs permettant d'organiser de manière hiérarchique les résultats par degré de similitude, offrant ainsi à l'utilisateur une visualisation et une compréhension plus aisées des résultats.
Notre apport est double, d'une part notre approche est relativement simple à mettre en place, ne nécessitant pas d'extension du langage SPARQL (contrairement à l'introduction des mot-clés CATEGORIZE BY et VIEW BY dans la littérature). D'autre part, notre approche ne contraint pas l'utilisateur à spécifier lors de sa requête comment grouper les résultats (*e.g.* via clauses GROUP|CATEGORIZE|VIEW BY), ce qui en permet une utilisation plus globale comme lors de recherche d'informations dans les bases du LOD parfois très généralistes.

6. `http://project.carrot2.org`
7. `http://yippy.com`, anciennement Vivisimo

Cet article est organisé de la manière suivante : la Section 2 définit la notion de *motif descriptif* qui est à la base de notre approche de classification hiérarchique de résultats SPARQL. La Section 3 traite du groupement des résultats à proprement parler selon une organisation hiérarchique des *voisinages de résultat*, où chaque branche de l'arbre est fonction d'un motif descriptif calculé, offrant une visualisation et une interprétation des groupes constitués. Dans cette section sont aussi présentées une modélisation pratique de notre approche en utilisant une Analyse des Correspondances Multiples combinée avec une Classification Hiérarchique sur Composantes Principales, ainsi que les résultats obtenus par une implémentation avec le logiciel R. La Section 4 conclut cet article et ouvre nos travaux sur certaines perspectives.

2. Motif descriptif

Une base RDF est composée d'un ensemble de *triplets*, où chaque triplet s'exprime de la forme (sujet, prédicat, objet). Le sujet représente la *ressource* à décrire, le prédicat représente une *propriété* (i.e. relation) applicable à cette ressource, enfin l'objet représente la valeur de cette propriété. Cette valeur peut correspondre soit à une autre ressource, soit à une "donnée brute" appelée un *littéral* (*e.g.* un nombre, une date, une chaîne de caractères). Une base RDF peut donc naturellement être vue comme un multi-graphe orienté et étiqueté – appelé *graphe RDF* – décrivant les ressources et leurs relations : les sujets et objets constituent les *nœuds*, les prédicats forment les *arcs*. Les étiquettes des nœuds sont les identifiants des ressources ou les valeurs des littéraux selon le cas, et les étiquettes des arcs sont les identifiants des prédicats. Notons ici, qu'un *nœud blanc* sert à indiquer l'existence[8] d'une ressource (dite anonyme) ou d'un littéral sans en préciser l'identifiant ou la valeur en particulier. D'un point de vue graphe, un nœud blanc correspond à un nœud non étiqueté[9].

Considérons le vocabulaire suivant : soit G un graphe RDF, on note $\mathcal{V}_G$ l'ensemble des nœuds de G, $\mathcal{R}_G$ le sous-ensemble de $\mathcal{V}_G$ composé des ressources (*i.e.* les nœuds qui ne sont pas des littéraux ; nœuds blancs possibles), $\mathcal{B}_G$ l'ensemble des nœuds blancs de G, et $\mathcal{E}_G$ l'ensemble des arcs. Un arc d'un nœud a vers un nœud b d'étiquette e sera noté (a, b, e) ; a est appelé le prédécesseur de l'arc et b le successeur.

Désignons enfin par le terme ***couple graphe–ressource***, noté de la forme (G, R), le couple formé d'un graphe non vide G et d'un ensemble de ses ressources $R = \{r_1, r_2, ..., r_k\} \subseteq \mathcal{R}_G$ avec $k \in \mathbb{N}^*$. Nous appelons *ressource notable* de G un élément r_i de R, et nous appelons *graphe d'appartenance* d'une/des ressources notables le graphe G.

8. *cf.* `www.w3.org/TR/2014/REC-rdf11-mt-20140225/#blank-nodes`

9. En pratique, un nœud blanc dispose d'un identifiant local et propre à la base où il est défini ; il est donc parfaitement identifiable parmi d'autres nœuds (blancs ou non).

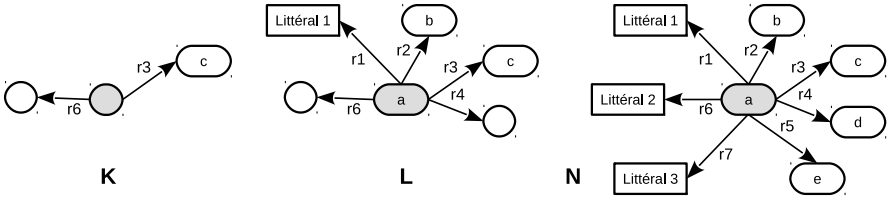

Figure 1. *Exemples de généralisation de couples graphe–ressources, où* $(K, \{\text{nœud blanc central}\}) \succ (L, \{a\}) \succ (N, \{a\})$

En pratique, les graphes qui vont être comparés dans cet article sont connexes et en étoile centrés[10] autour d'une (ou plusieurs) ressources, car l'on se place dans le cadre de graphes constituant le *voisinage* de ressources (*cf.* Section 3). Notons cependant que les définitions suivantes sont valables dans un cadre général pour des graphes RDF quelconques.

Définition 1 (Généralisation d'un couple graphe–ressource) *Soient* (G, R) *et* (H, S) *deux couples graphe–ressources, c'est à dire* G *et* H *deux graphes RDF non vides, et* $R \subseteq \mathcal{R}_G$ *et* $S \subseteq \mathcal{R}_H$*.*

Considérons l'ensemble de ressources R' *initialement défini tel que* $R' = R$*.* (H, S) *est une* généralisation *de* (G, R)*, notée* $(H, S) \succ (G, R)$*, si* H *s'obtient à partir de* G *en appliquant une combinaison non nulle et finie des règles suivantes :*

– supprimer un arc et son successeur, sauf si le successeur $\in R'$

- si après suppression du successeur un autre arc se retrouve sans une de ses extrémités, supprimer aussi cet autre arc.

– remplacer un nœud par un nœud blanc ; si ce nœud $\in R'$ *alors* R' *est mis à jour en conséquent ;*

et si en fin d'application des règles, $S = R'$*.*

La Figure 1 présente un exemple de généralisation entre plusieurs couples graphe–ressources. Le couple graphe–ressource $(L, \{a\})$ est une généralisation du couple $(N, \{a\})$ puisqu'on peut passer du second au premier par suppression des arcs r_5 et r_7 (avec leurs successeurs respectifs e et le Littéral 3) et par remplacement de la ressource d et du Littéral 2 en deux nœuds blancs ; la ressource notable a et les autres arcs (et leurs successeurs) restant inchangées. De même, le couple $(K, \{\text{nœud blanc central}\})$ est une généralisation de $(L, \{a\})$, car disposant en moins des arcs r_1, r_2 et r_4 (et leurs successeurs) et mutation de la ressource notable a en un nœud blanc ; ce nœud blanc constituant toujours la ressource notable.

10. L'aspect en étoile et centré autour d'une ressource est très visible pour un voisinage au rang 1 de cette ressource, mais moins évident aux rangs supérieurs (surtout si la base RDF où sont extraits les voisinages est fortement connexe).

Il est interressant de constater plusieurs propriétés attenantes aux couples graphe–ressources. La cardinalité de l'ensemble des ressources notables est identique pour un couple graphe–ressource et une généralisation de ce couple (*i.e.* $|S| = |R|$). Cette cardinalité est donc constante pour des couples graphe–ressources comparables par généralisations successives (*cf.* Figure 1 avec à chaque fois une ressource notable, sur fond gris). Les ressources notables non anonymes (*i.e.* étiquetées $\neq$ nœuds blancs) dans un couple graphe–ressource le plus général sont nécessairement présentes dans le couple le plus spécifique (*i.e.* $\forall r \in S \backslash \mathcal{B}_H, r \in R$; *cf.* Figure 1 et la ressource notable a dans L et donc dans N). Le graphe d'appartenance du couple graphe–ressource plus général est de taille – d'un point de vue graphe – plus petite ou égale que le graphe du couple le plus spécifique (*i.e.* $|\mathcal{V}_H| \leqslant |\mathcal{V}_G|$ et $|\mathcal{E}_H| \leqslant |\mathcal{E}_G|$). Le couple graphe–ressource plus général peut comporter plus de nœuds blancs en terme de ressources notables que dans le couple plus spécifique (*i.e.* $|\mathcal{B}_S| \geqslant |\mathcal{B}_H|$). Enfin, la notion de généralisation est transitive (*cf.* Figure 1 avec $(K, \{\text{nœud blanc central}\}) \succ (N, \{a\})$).

Nous définissons un *motif descriptif* comme une description des éléments communs à un ensemble de graphes RDF, en tenant compte de certains éléments en particulier. La désignation « motif descriptif » et sa définition sont empruntées pour partie à la notion de « motif de description » dans (Maillot *et al.*, 2016).

Définition 2 (Motif descriptif) *Soit* $\mathcal{G} = \{(G_1, R_1), (G_2, R_2), ...(G_k, R_k)\}$*, avec* $k \in \mathbb{N}^*$*, un ensemble de couples graphe–ressources.*

Le couple graphe–ressource (M, R) *est un* motif descriptif *de* $\mathcal{G}$ *si il est une généralisation de chaque élément de* $\mathcal{G}$*.*

Un *motif descriptif* représente donc ce qui apparaît de façon commune, tant d'un point de vue étiquette que de la structure, à un ensemble de graphes RDF en y considérant certaines ressources de façon particulières.
Pour que le couple graphe–ressource (M, R) soit un motif descriptif des couples (G_i, R_i), il est donc nécessaire que chaque R_i ainsi que R aient la même cardinalité. Remarquons aussi que dans le cas où les différents ensembles R_i n'ont aucune ressource notable étiquetée en commun, alors les ressources notables du motif descriptif sont uniquement des nœuds blancs. Et finalement, si les différents graphes d'appartenance G_i n'ont aucun élément en commun (arcs et nœuds), alors $M = R$, c'est à dire M est uniquement constitué de nœuds blancs non connexes.

Les Figures 2 et 3 sont des exemples de motifs descriptifs (en bas des figures). En Figure 2 quatre propriétés et leurs valeurs sont communes entre les graphes constitués des deux voisinages de Apple_Inc. et Apple_Records dans DBPedia, en considérant ces centres de voisinage comme ressources notables ; quant à la propriété rdfs:label, elle est commune aux deux voisinages par son existence mais pas par sa valeur, d'où la présence d'un nœud blanc dans le motif descriptif pour le successeur de cette propriété. Les ressources notables, Apple_Inc. et Apple_Records, des deux couples graphe–ressources étant différentes, la ressource notable du modif descriptif est un nœud blanc (grisé sur la Figure). En Figure 3, seuls deux arcs sont partagés par les

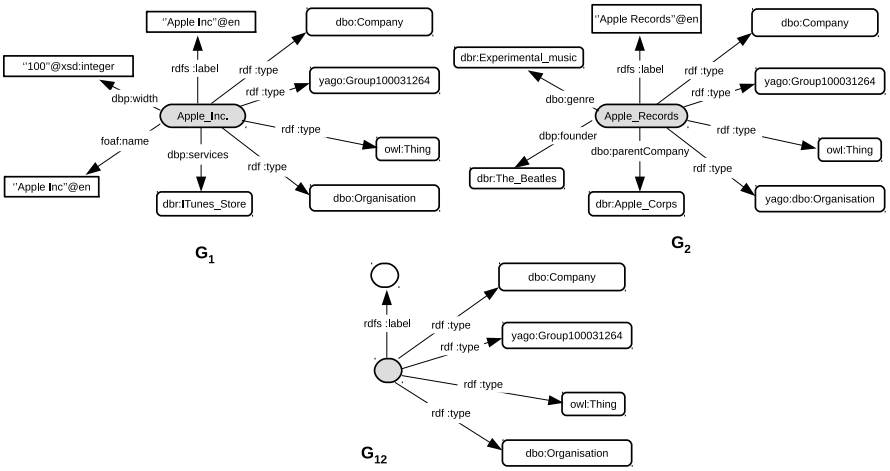

Figure 2. *Motif descriptif (G_{12},{nœud blanc central}) de l'ensemble {(G_1,{Apple_Inc.}), (G_2,{Apple_Records})}. Les graphes G_1 et G_2 sont respectivement des extraits du voisinage au rang 1 des ressources* Apple_Inc. *et* Apple_Records *dans DBPedia.*

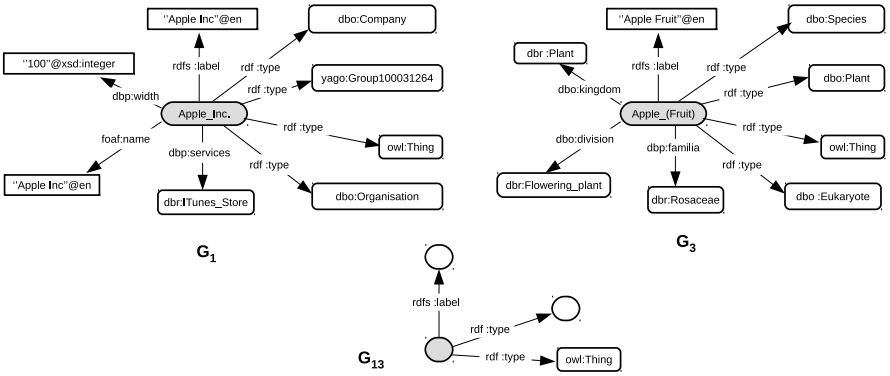

Figure 3. *Motif descriptif (G_{13},{nœud blanc central}) de l'ensemble {(G_1,{Apple_Inc.}), (G_3,{Apple_(Fruit)})}. G_1 et G_3 sont respectivement des extraits du voisinage au rang 1 des ressources* Apple_Inc. *et* Apple_(Fruit) *dans DBPedia.*

deux voisinages de Apple_Inc. et Apple_(Fruit) : rdf:type est présent d'une part avec le même successeur owl:Thing, et d'autre part avec d'autres successeurs non communs. La second arc partagé rdfs:label l'est uniquement sans tenir compte du successeur.

Définition 3 (Motif descriptif maximal) *Soit* (M, R) *un motif descriptif d'un ensemble* $\mathcal{G}$ *de couples graphe–ressources.*

(M, R) *est le motif descriptif* maximal *de* $\mathcal{G}$*, si il n'existe pas de motif descriptif* (M', R') *de* $\mathcal{G}$ *tel que* $(M, R) \succ (M', R')$ *avec* $M' \neq M$.

Les deux motifs descriptifs présentés en Figures 2 et 3 constituent chacun le motif descriptif maximal des deux couples graphe–ressources concernés. En d'autres termes, il n'existe pas de modtif descriptif de {(G_1,{Apple_Inc.}), (G_2,{Apple_Records})} plus spécifique que celui en Figure 2 ; de même en Figure 3.

3. Classification des résultats d'une requête SPARQL

SPARQL est le langage recommandé par le W3C pour interroger (et manipuler) une base RDF. L'interrogation par une requête SPARQL – de type SELECT – sur un graphe RDF correspond à un mécanisme de "*matching pattern*" cherchant une valuation de chaque variable de la requête dans le graphe interrogé selon les contraintes définies dans la requête. Un résultat de la requête est un sous-graphe de la base interrogée (donc un graphe sans variable).

Considérons à titre d'exemple la requête SPARQL en Figure 4, où un utilisateur souhaite identifier une ou des ressources concernant « Apple ». Notons qu'il s'agit ici d'une requête certes très simple et générale mais tout à fait typique pour un utilisateur cherchant à identifier une ressource en particulier ; une telle requête est généralement préliminaire à une seconde requête plus complexe où la ressource désirée serait alors parfaitement identifiée afin d'en connaître sa description plus complète. Rappelons qu'une ressource est identifiée par un IRI, qui certes peut être compréhensible par l'humain dans la base DBPedia, mais pas dans toutes les bases du LOD[11]. L'utilisation de la propriété rdfs:label est donc souvent nécessaire.

```
SELECT ?resource ?label
WHERE {
  ?resource rdfs:label ?label.
  FILTER( regex(?label, "Apple", "i") )
}
```

Figure 4. *Requête SPARQL permettant d'identifier les ressources dont le label contient la chaîne de caractères "Apple" (case* ***i****nsensitive).*

L'interrogation de la base DBPedia[12] par cette requête en Figure 4 fournit un grand nombre de résultats (extrait en Figure 5), contenant des informations aussi bien pour des personnes, des fruits, des ordinateurs ou encore des entités de la société Apple. Si pour aider l'utilisateur un mécanisme de ranking était utilisé, les premiers résultats

11. *e.g.* `https://www.wikidata.org/wiki/Q312` identifie la société Apple dans la base Wikidata.
12. Accès au SPARQL endpoint : `http://dbpedia.org/sparql`

?resource	?label
...	...
<http://dbpedia.org/resource/Apple_Inc.>	"Apple Inc."@en
<http://dbpedia.org/resource/Apple_Records>	"Apple Records"@en
<http://dbpedia.org/resource/Apple_(Fruit)>	"Apple (Fruit)"@en
<http://dbpedia.org/resource/Apple_vs._microsoft>	"Apple vs. microsoft"@en
<http://dbpedia.org/resource/Apple_Store>	"Apple Store"@en
<http://dbpedia.org/resource/Apple_AirPort>	"Apple AirPort"@en
<http://dbpedia.org/resource/Sugar-apple>	"Sugar-apple"@en
<http://www.wikidata.org/entity/Q7020003>	"Newton's Apple"@en
<http://dbpedia.org/resource/Fiona_Apple>	"Fiona Apple McAfee Maggart"@en
<http://dbpedia.org/resource/Mike_Apple>	"Michael Apple"@en
<http://www.wikidata.org/entity/Q18608095>	"Apple tree"@en
...	...

Figure 5. *Extrait des résultats de la requête en Figure 4 en interrogeant DBPedia.*

mis en avant seraient les plus populaires soit localement à la base DBPedia, soit sur l'ensemble des bases du LOD, donc certainement autour de la compagnie Apple. Cependant si l'utilisateur souhaite s'intéresser en particulier à un fruit ou une personne, la classification des résultats lui seraient de peu d'utilité : l'utilisateur devrait analyser une seule et longue liste de résultats. Dans ce scénario, un mécanisme de clustering serait plus adapté, permettant à l'utilisateur d'éliminer rapidement de son champ de vision les groupes de résultats qui manifestement ne traiteraient pas de la thématique voulue.

3.1. *Notre méthode de regroupement de résultats à une requête SPARQL*

Notre approche exploite les notions de *mofif descriptif* (Section 2) et de *voisinage*. Au sein d'une base RDF, le voisinage d'une ressource au rang 0 est la ressource elle-même, au rang 1 il est constitué de tous les arcs qui en partent et leurs successeurs, au rangs suivants il est en plus constitué des voisinages au rang 1 de chaque successeur et ainsi de suite récursivement. Notons qu'avec SPARQL, le voisinage au rang 1 d'une ressource pourrait être obtenu en utilisant une requête de type DESCRIBE (et non de type SELECT). Cependant, une telle requête n'est pas conseillée d'être utilisée dans le standard actuel SPARQL[13]. Pour plus de détails concernant le voisinage de ressources et leur extraction dans une base, le lecteur est invité à lire (Maillot *et al.*, 2014).

Définition 4 (Ressource résultat) *Soient g un résultat non vide d'une requête SPARQL Q de type* SELECT *dans un graphe RDF G, et* `?var` *une variable de Q.*

La ressource $r \in g$, image de `?var` *dans G en respectant les contraintes de Q, est appelée* ressource résultat.

13. "The DESCRIBE query is underspecified a bit by the standard, hence inconsistent results are possible.", *cf.* `https://www.w3.org/TR/sparql11-query/#describe`

Définition 5 (Voisinage de résultat) *Soient g un résultat non vide d'une requête SPARQL Q de type* SELECT *dans un graphe RDF G, et $n \in \mathbb{N}$.*

Le graphe formé de l'union des voisinages au rang n de chaque ressource résultat de g dans G est appelé le voisinage du résultat *g.*

Notre méthode de regroupement d'un ensemble non vide de résultats à une requête SPARQL en interrogeant une base RDF donnée se décompose selon les quatre étapes suivantes :

1) sélectionner dans la base les voisinages de chaque résultat (Définition 5).

2) générer les différents motifs descriptifs maximaux possibles à partir des couples graphe–ressources (voisinage_de_résultat$_i$, {ressources_résultats}$_i$) associées à chaque résultat (Définition 3).

3) générer de proche en proche les motifs descriptifs maximaux à partir des motifs descriptifs précédemment générés.

4) organiser hiérarchiquement les graphes d'appartenance des motifs descriptifs obtenus, de sorte qu'en bas de l'arbre – au niveau des feuilles – sont présents les voisinages de résultat en étape 1, et plus on monte dans l'arbre sont présents aux intersections les graphes d'appartenance des motifs descriptifs les plus généraux (*i.e.* une bifurcation étant le graphe du motif maximal de tous ces fils).

Le groupement des résultats peut finalement être obtenu simplement à partir de la hiérarchie obtenue. Le rang des voisinages à prendre en compte constitue un paramètre à fournir en entrée à notre méthode. Le nombre de groupes voulu constitue aussi un paramètre d'entrée, il peut être choisi dans l'absolu ou bien être fonction du nombre de résultats par sous-arbre (afin que les groupes soient équilibrés).

À titre pédagogique, considérons les seuls cinq résultats Apple_Inc., Apple_Store, Apple_Records, Sugar-apple et Apple_(Fruit) présentés en Figure 5. Selon notre méthode, nous obtenons la classification hiérarchique de ces résultats en Figure 6. Un nœud de l'arbre correspond à un motif descriptif indiquant les points communs aux éléments de son sous-arbre[14]. Chaque nœud de l'arbre est donc une généralisation de ses fils ; et de manière récursive de ses descendants, apportant ainsi la preuve de l'adéquation hiérarchique. Ici, pour une meilleure visualisation, au niveau des feuilles, les voisinages de résultat sont simplifiés à la seule ressource résultat. Si par exemple on coupe l'arbre au niveau du nœud (c) en Figure 6, on obtient deux[15] groupes : l'un constitué de compagnies (dans l'industrie du software ou non[16]) – Apple_Inc., Apple_Store et Apple_Records – et l'autre de plantes (qui fleurissent) – Sugar-apple et Apple_(Fruit).

14. On retrouve le motif descriptif G_{12} de la Figure 2 qui correspond au nœud (b) de l'arbre, et le motif descriptif G_{13} de la Figure 3 qui correspond au nœud (c) de l'arbre.

15. Ce qui semble suffisant pour seulement 5 résultats de ce cas pédagogique.

16. Si plus de groupes, alors Apple_Records aurait été séparé des deux autres résultats Apple_Inc. et Apple_Store.

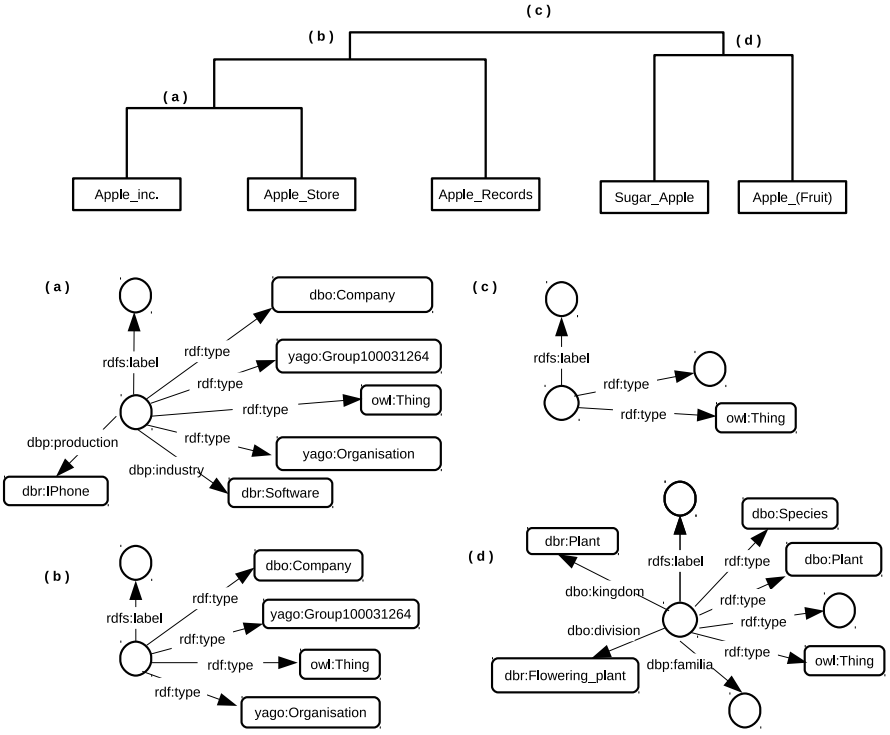

Figure 6. *Classification hiérarchique des résultats.*

3.2. *Une expérimentation de notre approche par ACM, implémentée sous R*

3.2.1. *Une classification par ACM & CHCP*

Le principe de la classification de données est d'organiser une collection de données en groupes, appelé *clusters*, de sorte que les éléments d'un cluster soient plus similaires les uns aux autres qu'à ceux des autres clusters (Berkhin, 2006 ; Kaufman et Rousseeuw, 2009). Dans la littérature, il existe différentes catégories de classification ; les deux catégories traditionnelles sont la *classification hiérarchique* et *classification de partition*. La classification hiérarchique (Murtagh, 1983 ; Olson, 1995) regroupe les données suivant une variété d'échelles en créant une arborescence de classification multi-niveaux, où le cluster d'un niveau et relié aux clusters du niveau suivant. La classification de partition répartit initialement de façon arbitraire les données en différents groupes (ou classes), cette répartition est ensuite améliorée itérativement jusqu'à la convergence des critères choisis.

Notre méthode de regroupement de résultats SPARQL s'apparente à la première catégorie. Afin d'expérimenter notre approche, il est intéressant d'exploiter ici une

technique de clustering de classification hiérarchique existante et adaptée. Dans la mesure où les données que nous avons à traiter sont uniquement qualitatives (puisque composées d'étiquettes), nous nous sommes tournés vers une Analyse des Correspondances Multiples (ACM) combinée à une Classification Hiérarchique sur Composantes Principales (CHCP). L'article (Husson *et al.*, 2010) décrit en détail la classification hiérarchique avec en entrée une analyse factorielle.
L'ACM est une méthode statistique permettant d'étudier l'association entre variables qualitatives[17] (Benzécri, 1973). Il s'agit d'une extension de l'Analyse des Correspondances [Simples] (Hirschfeld, 1935 ; Benzécri, 1973) pour l'analyse de données contenant plus de deux variables qualitatives. À partir d'un tableau en entrée, dit « tableau de codage condensé » (TDD), l'ACM construit le tableau disjonctif complet sur lequel elle opère, pour aboutir à des cartes de représentation sur lesquelles les données sont représentées sous forme de points dans un espace euclidien de faible dimension. On peut donc visuellement y observer les proximités entre les catégories des variables qualitatives. Ce caractère visuel très intéressant pour une bonne compréhension par l'utilisateur, peut être obtenu de manière hiérarchique grâce à une CHCP, qui classifie de manière non supervisée les individus sur les composantes principales issues d'une analyse factorielle (ici, l'ACM).

Préparation des données. Afin d'utiliser une ACM (puis une CHCP), nous préparons les résultats d'une requête SPARQL de la manière suivante pour l'obtention du TDD :

– chaque une ligne correspond à un résultat ;

– la première ligne est constituée en première cellule du numéro/identifiant du résultat, et les autres cellules sont construites à partir du voisinage résultat V_1 (du premier résultat) tel que : pour chaque variable `?x` de la requête, on considère la ressource résultat associé r dans V_1, puis pour chaque arc (r, b_j, e_j) de V_1 ayant comme prédécesseur r, on créer les cellules suivantes – et donc autant de colonnes – dont les valeurs qualitatives sont : `concat("?x",`e_1`)`, `concat("?x",`e_1`,`b_1`)`, `concat("?x",`e_2`)`, `concat("?x",`e_2`,`b_2`)`, ... , `concat("?x",`e_n`)`, `concat("?x",`e_n`,`b_n`)`, avec n le nombre d'arcs partant de r et `concat` une fonction de concaténation de chaînes de caractères.

– les autres lignes sont construites sur le même principe que la première (à partir des voisinages résultat V_i du i-ème résultat), où cette fois-ci, si une valeur de cellule existe déjà dans le tableau, en colonne c, alors elle est stockée dans cette même colonne c, sinon une nouvelle colonne est créée pour y stocker cette nouvelle valeur.

Le tableau ainsi obtenu renseigne par résultat (en ligne) chaque arc (`concat("?x",`e_i`)`) et chaque binôme arc–successeur (`concat("?x",`e_i`,`b_i`)`) partant des ressources résultats (en colonne). La classification va donc prendre en entrée ces éléments qui constituent une représentation possible des voisinages de résultat de notre approche théorique. Chaque cluster obtenu sera ainsi fait que les éléments corrélées concernées forment ensemble le motif descriptif du groupe.

17. L'ACM est aux variables qualitatives ce que l'Analyse en Composantes Principales (ACP) est aux variables quantitatives.

3.2.2. *Implémentation avec R*

Nous avons implémenté notre approche avec le logiciel R[18] d'analyses statistiques, en utilisant particulièrement le package SPARQL pour le requêtage de bases RDF et le package FactoMineR dédié à l'analyse exploratoire multidimensionnelle de données incluant les fonctions d'ACM et de CHCP (Lê *et al.*, 2008).

Conditions d'expérimentation. La base DBPedia nous a servi de terrain d'expérimentation. Notons que le serveur Virtuoso[19] qui héberge DBPedia dispose d'une gestion de la mémoire permettant de « mettre en cache » les dernières requêtes traitées et leurs résultats. Afin d'éviter un biais important lors nos tests, nous avons utilisé un serveur local – avec une copie[20] de DBPedia et 32 Go de RAM réservée – sur lequel nous avions la main pour désactiver cette mise en cache. Les requêtes de test ont été exécutées sur un ordinateur personnel tournant sous Windows 10 avec un processeur Intel Core i5 et 8 Go de RAM.

La Figure 7 présente la décomposition des différents temps d'exécution de la requête initiale (*cf.* Figure 4) jusqu'à la classification hiérarchique CHCP, en passant respectivement par les étapes de sélection des voisinages de résultat (pour tous les résultats), de leurs mise en forme en tableau de codage condensé pour l'ACM, et du calcul de l'ACM à proprement parler.
On constate que notre approche valide dans une certaine mesure le passage à l'échelle, du moins pour un nombre de résultats encore acceptable à présenter à l'utilisateur (même groupés en clusters), avec un temps de quelques secondes pour traiter 100 résultats et un temps raisonnable de l'ordre de la minute pour 500 résultats. Remarquons que le temps de création du tableau à fournir en entrée à l'ACM devient contre performant au delà de 700 résultats. Dans la mesure où nous avons développé cette partie de façon très naïve, une marge d'amélioration certainement conséquente est envisageable ; ce qui est encourageant sur la performance totale.

Résultat final obtenu. La Figure 8 constitue les résultats à la requête SPARQL en Figure 4 en interrogeant DBPedia, mais présentés cette fois-ci (*versus* Figure 5) au sein des différents clusters calculés par nos soins. Le motif descriptif d'un cluster est précisé avant de lister les résultats. On retrouve le motif descriptif (b) (Figure 6) au niveau du cluster `#1`, le motif descriptif (a) au niveau du cluster `#1.1` et le motif descriptif (d) au niveau du cluster `#2`.

4. Conclusion et perspectives

Cet article se positionne sur la problématique de classification hiérarchique des résultats d'une requête SPARQL, afin de présenter à l'utilisateur des groupes de résul-

18. `www.r-project.org/`
19. `https://virtuoso.openlinksw.com/`
20. `http://wiki.dbpedia.org/Downloads2015-10`

nb de résultats	REQUÊTE initiale	Sélection des VOISINAGES	Création du TDD	ACM	HCPC	Temps TOTAL
50	0,05	3,37	0,16	1,01	1,74	6,33
100	0,08	4,50	0,20	1,40	2,59	8,76
300	0,17	16,33	2,58	7,56	7,56	34,20
500	0,25	27,80	8,68	18,56	12,42	67,71
700	0,38	39,45	25,42	39,68	19,93	124,86
1000	0,59	60,30	129,53	142,42	44,64	377,48

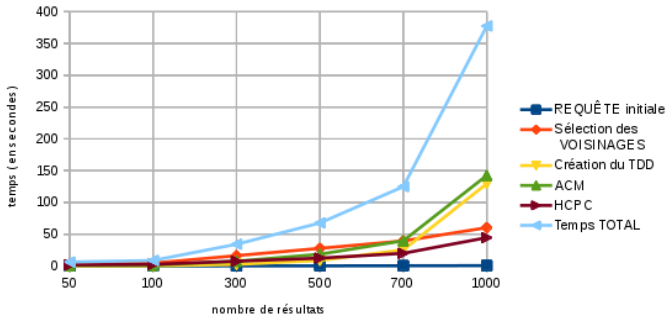

Figure 7. *Temps de traitement des résultats de la requête SPARQL en Figure 4 sur DBPedia, selon différentes valeurs de la clause* LIMIT *(50, 100, 300, ..., 1000).*

```
Cluster #1: { ([_:1],rdf:type,dbo:Company); ([_:1],rdf:type,yago:Group100031264); ([_:1],
rdf:type,yago:Organisation); ([_:1], rdf:type, owl:Thing); ([_:1],rdfs:label,[_:2]) }
-------------------------------------------------------------------------------------
|                    ?resource                        |              ?label              |
| ================================================================================|
| <http://dbpedia.org/resource/Apple_Inc.>            | "Apple Inc."@en                  |
| <http://dbpedia.org/resource/Apple_Store>           | "Apple Store"@en                 |
| <http://dbpedia.org/resource/Apple_Records>         | "Apple Records"@en               |
-------------------------------------------------------------------------------------

Cluster #1.1: { ([_:1],dbp:production,dbr:IPhone); ([_:1],dbp:industry,dbr:Software);
([_:1],rdf:type,dbo:Company); ([_:1],rdf:type,yago:Group100031264); ([_:1],rdf:type,
yago:Organisation); ([_:1], rdf:type, owl:Thing); ([_:1],rdfs:label,[_:2]) }
-------------------------------------------------------------------------------------
|                    ?resource                        |              ?label              |
|=====================================================================================
| <http://dbpedia.org/resource/Apple_Inc.>            | "Apple Inc."@en                  |
| <http://dbpedia.org/resource/Apple_Store>           | "Apple Store"@en                 |
|-------------------------------------------------------------------------------------

#Cluster 2: { ([_:1],dbo:division,dbr:Flowering_plant); ([_:1],dbp:familia,[_:2]);
([_:1],rdf:type,[_:3]); ([_:1],rdf:type,owl:Thing); ([_:1], rdf:type, dbo:Plant);
([_:1],rdf:type,dbo:Species); ([_:1],rdfs:label,[_:4]); ([_:1],dbo:kingdom,dbr:Plant) }
-------------------------------------------------------------------------------------
|                    ?resource                        |              ?label              |
=====================================================================================
| <http://dbpedia.org/resource/Apple_(Fruit)>         | "Apple (Fruit)"@en               |
| <http://dbpedia.org/resource/Sugar_Apple>           | "Sugar_Apple"@en                 |
-------------------------------------------------------------------------------------
```

Figure 8. *Extrait des résultats, organisés selon notre approche de clustering, à la requête SPARQL en Figure 4 en interrogeant la base DBPedia.*

tats (clusters) selon leurs similitudes. L'originalité de notre approche est de comparer non pas seulement les résultats entre eux, mais de considérer le graphe constituant un résultat ainsi que son voisinage au sein même de la base interrogée. À partir de ces voisinages de résultat nous en déterminons des motifs descriptifs caractérisant les éléments communs à un ensemble, voire un ensemble d'ensembles, de résultats. Ces motifs descriptifs sont l'essence même de la classification hiérarchique que nous proposons et offre à l'utilisateur une visualisation et une compréhension des clusters et donc des résultats.

Notre approche, finalement assez simple à mettre en place en exploitant des techniques existantes de clustering (comme ACM + CHCP en statistiques), se positionne d'une certaine manière comme une généralisation des approches de clustering adaptées au LOD telles que (Alam et Napoli, 2014) et (d'Amato *et al.*, 2010). En effet, notre approche (i) ne se focalise pas que sur les résultats eux-mêmes, comme c'est le cas avec la contribution VIEW BY dont l'apport principal est de lever la contrainte lier à l'ordre des paramètres de la clause GROUP BY, (ii) exploite tout le voisinage des résultats, et pas uniquement[21] les liens de typage associés aux ressources résultats, comme c'est le cas avec la contribution CATEGORIZE BY, et (iii) ne présuppose pas que l'utilisateur sache sur quel(s) critères grouper les résultats.

Les résultats obtenus lors d'expérimentations sont encourageants, ils valident tant qualitativement que quantitativement notre approche, et nous invitent à nous tourner vers d'autres techniques de clustering pour des données qualitatives – comme l'analyse formelle de concepts – pour en comparer les performances.

Pour finir, au sein même d'un groupe de résultats, des approches de *ranking* pourraient être appliquées à chaque groupe de manière complémentaire, et non comme des approches orthogonales à celles de *clustering*.

5. Bibliographie

Alam M., Napoli A., « Lattice-Based Views over SPARQL Query Results », *Proceedings of the 1st Workshop on Linked Data for Knowledge Discovery (LD4KD)*, CEUR-WS.org, 2014.

Balmin A., Hristidis V., Papakonstantinou Y., « Objectrank : Authority-based keyword search in databases », *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*, VLDB Endowment, p. 564-575, 2004.

Benzécri J.-P., *L'Analyse des Données. Vol II : l'Analyse des Correspondances*, Dunod, 1973.

Berkhin P., « A survey of clustering data mining techniques », *Grouping multidimensional data*, Springer, p. 25-71, 2006.

Bhalotia G., Hulgeri A., Nakhe C., Chakrabarti S., Sudarshan S., « Keyword searching and browsing in databases using BANKS », *Data Engineering, 2002. Proceedings. 18th International Conference on*, IEEE, p. 431-440, 2002.

21. Dans la majorité des bases du LOD, lorsqu'une ressource possède un type, pour des raisons de performances les types hérités lui sont aussi associés (*versus* inférés dynamiquement).

Bizer C., Heath T., Berners-Lee T., « Linked data - the story so far », *International Journal on Semantic Web and Information Systems*, vol. 5, nº 3, p. 1-22, 2009.

Brin S., Page L., « The Anatomy of a Large-Scale Hypertextual Web Search Engine », *Proceedings of the 7th International World-Wide Web Conference (WWW'98)*, 1998.

Carpineto C., Osiński S., Romano G., Weiss D., « A Survey of Web Clustering Engines », *ACM Comput. Surv.*, vol. 41, nº 3, p. 17 :1-17 :38, 2009.

d'Amato C., Fanizzi N., Lawrynowicz A., « Categorize by : Deductive Aggregation of Semantic Web Query Results », *Proc. of ESWC (1)*, vol. 6088 of *LNCS*, Springer, p. 91-105, 2010.

Delbru R., Toupikov N., Catasta M., Tummarello G., Decker S., « Hierarchical link analysis for ranking web data », *The Semantic Web : Research and Applications*, Springer, 2010.

Feyznia A., Kahani M., Zarrinkalam F., « COLINA : a method for ranking SPARQL query results through content and link analysis », *Proceedings of the 2014 International Conference on Posters & Demonstrations Track*, vol. 1272, CEUR-WS. org, p. 273-276, 2014.

Ganter B., Wille R., *Formal Concept Analysis*, Springer, Berlin Heidelberg, 1999.

Hirschfeld H., « A connection between correlation and contingency », *Cambridge Philosophical Society*, vol. 31, p. 520-524, 1935.

Husson F., Josse J., Pagès J., Principal component methods-hierarchical clustering-partitional clustering : why would we need to choose for visualizing data, Technical report, Agrocampus/France, 2010. Applied Mathematics Department, 2010.

Kaufman L., Rousseeuw P. J., *Finding groups in data : an introduction to cluster analysis*, vol. 344, John Wiley & Sons, 2009.

Lawrynowicz A., Potoniec J., Konieczny L., Madziar M., Nowak A., Pawlak K. T., « Asparagus - a system for automatic SPARQL query results aggregation using semantics. », *Proceedings of ICCCI (1)*, vol. 6922 of *LNCS*, Springer, p. 304-313, 2011.

Lê S., Josse J., Husson F., « FactoMineR : A Package for Multivariate Analysis », *Journal of Statistical Software*, vol. 25, nº 1, p. 1-18, 2008.

Maillot P., Genest D., Loiseau S., Raimbault T., « Targeted Linked-Data Extractor », *Proc. of ICAART*, vol. 1, p. 336-341, 2014.

Maillot P., Genest D., Loiseau S., Raimbault T., « Diagnostic pour le maintien de la qualité des bases du Web des données », *Congrès Reconnaissance des Formes et l'Intelligence Artificielle (RFIA), Conférence Nationale en Intelligence Artificielle (CNIA)*, 2016.

Mirizzi R., Ragone A., Di Noia T., Di Sciascio E., *Ranking the linked data : the case of dbpedia*, Springer, 2010.

Mulay K., Kumar P. S., « SPRING : Ranking the results of SPARQL queries on Linked Data », *Proceedings of the 17th International Conference on Management of Data*, Computer Society of India, p. 12, 2011.

Murtagh F., « A survey of recent advances in hierarchical clustering algorithms », *The Computer Journal*, vol. 26, nº 4, p. 354-359, 1983.

Olson C. F., « Parallel algorithms for hierarchical clustering », *Parallel computing*, vol. 21, nº 8, p. 1313-1325, 1995.

Thalhammer A., Rettinger A., « PageRank on Wikipedia : Towards General Importance Scores for Entities », *Proceedings of ESWC2016 Satellite Events*, vol. 9989 of *Lecture Notes in Computer Science*, Springer, p. 227-240, 2016.