Ordonnancement d'entités appliqué à la construction de snippets sémantiques

Mazen Alsarem* — **Pierre-Edouard Portier*** — **Sylvie Calabretto*** — **Harald Kosch****

* *Université de Lyon, CNRS*
INSA de Lyon, LIRIS, UMR5205, F-69621, France
** *Universität Passau*
Innstr. 43, 94032 Passau, Germany

RÉSUMÉ. *Les avancées de l'initiative Linked Open Data (LOD) ont permis de mieux structurer le Web des données. En effet, quelques jeux de données servent de centralisateurs (par exemple, DBpedia) et permettent ainsi de maintenir les différentes sources de données du LOD liées entre elles. Ces jeux de données ont également permis le développement de services de détection des entités du Web des données dans une page du Web des documents (par exemple, DBpedia Spotlight). Ainsi, pour permettre l'émergence de nouveaux usages qui combineront les deux Webs, nous proposons un algorithme qui ordonne les entités détectées dans une page Web en fonction d'une requête exprimant un besoin d'information. Nous montrons que cet algorithme est significativement meilleur que les approches de l'état de l'art. Enfin, nous utilisons cet algorithme pour construire un système de génération de snippets sémantiques dont nous montrons expérimentalement l'utilité et l'utilisabilité.*

ABSTRACT. *The advances of the Linked Open Data (LOD) initiative are giving rise to a more structured Web of data. Indeed, a few datasets act as hubs (e.g., DBpedia) connecting many other datasets. They also made possible new Web services for entity detection inside plain text (e.g., DBpedia Spotlight), thus allowing for new applications that will benefit from a combination of the Web of documents and the Web of data. To ease the emergence of these new use-cases, we propose an algorithm for the ranking of entities, detected inside a Web page, by taking into account an information need expressed as a query. We show that this algorithm is significantly more efficient than the state of the art. Finally, we use this algorithm for the construction of semantic snippets for which we evaluate the usability and the usefulness on a panel of users.*

MOTS-CLÉS : *Web des données, Ordonnancement d'entités, Snippets sémantiques.*

KEYWORDS: *Web of data, Entity ranking, Semantic snippets.*

1. Introduction

Dans ce travail, nous présentons l'algorithme LDSVD (Linked Data Singular Value Decomposition) qui permet d'ordonner les entités d'un graphe issu du LOD en fonction d'un besoin d'information exprimé sous la forme d'une requête formée d'un ensemble de mots clés. L'algorithme LDSVD s'applique à des graphes pour lesquels des données textuelles descriptives peuvent être associées aux nœuds (c.à.d. aux entités). Nous appliquons cet algorithme à des graphes issus de la détection automatique au sein d'une page Web d'entités du LOD. Cette détection d'entités peut être réalisée grâce à des services tels que DBpedia Spotlight (Mendes *et al.*, 2011) (configurable par le biais de listes blanches et noires qui permettent de filtrer sur les types d'entités tels que définis par la hiérarchie des classes de l'ontologie DBpedia, incluant une phase de désambiguïsation contextuelle, et associant un score de confiance à chaque résultat), AlchemyAPI[1] (similaire à DBpedia Spotlight, mais référençant différentes sources de données du LOD et comportant donc une étape de résolution des coréférences), OpenCalais[1], SemanticAPI de Ontos[1], etc. Les données textuelles associées aux nœuds du graphe proviennent alors du résumé DBpedia de l'entité, ainsi que de passages de la page Web centrés sur les occurrences de l'entité. Afin de motiver la nécessité du LDSVD, il est nécessaire de rappeler brièvement le rapport entre les approches classiques pour l'ordonnancement de pages Web et les approches existantes pour l'ordonnancement d'entités du Web des données.

Dans le contexte du *Web des documents*, un hyperlien indique une relation entre des informations portées par deux pages Web. Bien que de telles relations soient généralement d'une granularité assez grossière, elles forment cependant une composante essentielle des algorithmes d'ordonnancement les plus reconnus (PageRank (Page *et al.*, 1999), HITS (Kleinberg, 1999), SALSA (Lempel et Moran, 2001))

Dans le contexte du *Web des données*, les liens (matérialisés par des triplets sujet / prédicat / objet) représentent des relations nommées dont les entités sources et cibles sont généralement d'une granularité plus fine que pour le Web des documents. La grande majorité des stratégies existantes pour l'ordonnancement d'entités du Web des données (voir (Roa-Valverde et Sicilia, 2014) et (Jindal *et al.*, 2014) pour des états de l'art récents) sont fondées sur des adaptations du PageRank. Il existe également des approches du type apprendre-à-ordonner (*learning-to-rank*) appliquées au Web des données (par exemple (Dali *et al.*, 2012)). Ces techniques dépendent de la disponibilité des jugements de pertinence pour l'apprentissage.

Avec l'algorithme LDSVD, nous adoptons une stratégie différente basée sur une application itérative de la décomposition en valeurs singulières. Dans cet article, nous montrons que cette approche produit des ordonnancements d'une qualité significativement meilleure à celle des ordonnancements produits par les stratégies de l'état de l'art basées sur des modifications du PageRank.

1. `www.alchemyapi.com` ; `www.opencalais.com` ; `www.ontos.com`

Finalement, nous montrons également le potentiel du LDSVD en l'appliquant à un contexte de construction de snippets sémantiques. Un snippet est un extrait d'une page Web calculé au moment de l'analyse de la requête et devant aider l'utilisateur à décider de la pertinence de la page Web par rapport à son besoin d'information. Un snippet sémantique vise à améliorer ce processus de décision et d'exploration en rendant explicites les relations entre un besoin d'information et les entités les plus pertinentes présentes dans une page Web.

Dans la section 2, nous introduisons les travaux relatifs à l'amélioration des snippets pour le Web des documents et pour le Web des données. Dans la section 3, nous présentons l'algorithme LDSVD et son évaluation comparative avec les approches de l'état de l'art. Dans la section 4, nous introduisons ENsEN (Enhanced Search Engine), le système logiciel que nous avons développé pour générer des snippets sémantiques. Dans la section 5, nous présentons les résultats d'une évaluation de l'utilisabilité et de l'utilité du système ENsEN, avant de conclure en section 6.

2. Travaux connexes

Nous mentionnons tout d'abord des travaux qui génèrent des snippets à partir de documents RDF natifs. Ge *et al.* (Ge *et al.*, 2012), et Penin *et al.* (Penin *et al.*, 2008) s'intéressent à la génération de snippets pour la recherche d'ontologies. Bai *et al.* (Bai *et al.*, 2008) génèrent des snippets pour un moteur de recherche sémantique.

Dans (Penin *et al.*, 2008), les auteurs commencent par identifier un sujet thématique grâce à un algorithme de clustering hiérarchique hors-ligne. Ensuite, ils calculent une liste de triplets RDF (c.à.d. des ensembles de triplets RDF connectés) sémantiquement proches du thème. Enfin, grâce à une mesure de similarité fondée sur Wordnet, ils classent les triplets RDF sélectionnés en considérant les propriétés structurelles du graphe RDF et les caractéristiques lexicales des termes présents dans l'ontologie.

Dans (Ge *et al.*, 2012), les auteurs commencent par transformer le graphe RDF en un graphe qui met en relation des paires de termes et pour lequel chaque arête est associée à un ensemble de triplets RDF. Leur objectif est de construire une représentation compacte des relations qui existent entre les termes de la requête. Ces relations sont à trouver dans le graphe RDF. Pour ce faire, les auteurs décomposent le graphe d'association des termes en composantes dont le rayon ne doit pas dépasser une valeur fixée (c'est un paramètre de l'algorithme) afin d'éviter la découverte de relations trop lointaines entre termes de la requête. Il s'agit ensuite de chercher au sein de ces composantes des sous-graphes connexes qui relient des termes de la requête. Le snippet est alors une somme de ces sous-graphes connexes.

Dans (Bai *et al.*, 2008), les auteurs commencent par attribuer un thème au document RDF. Pour cela, ils utilisent un prédicat tel que `p:primaryTopic` s'il existe, sinon ils s'appuient sur une heuristique fondée sur la comparaison des URIs des entités candidates pour représenter le thème avec l'URL du document RDF. Ensuite, ils ont proposé un algorithme pour l'ordonnancement des triplets RDF. A ce propos, il semble

intéressant de noter comment ils utilisent les propriétés : pour chaque propriété, ils définissent son importance par rapport aux autres propriétés d'un schéma donné, ils introduisent également les notions de propriétés corrélatives (par exemple `foaf:name` et `foaf:family`) et exclusives (par exemple `foaf:name` et `foaf:surname`). Enfin, ils utilisent cet algorithme d'ordonnancement pour présenter à l'utilisateur un ensemble de relations entre les triplets proches de la requête et les triplets proches du thème du document RDF.

Pour résumer, comme Ge *et al.* (Ge *et al.*, 2012) nous pensons que posséder des données structurées issues d'un graphe RDF offre la possibilité de trouver des relations non triviales entre les termes de la requête eux-mêmes, mais aussi entre les termes de la requête et les concepts les plus importants du document. En outre, nous suivons également Penin *et al.* (Penin *et al.*, 2008) à propos de la nécessité de concevoir un algorithme d'ordonnancement de triplets RDF qui tienne compte à la fois de la structure du graphe et des propriétés lexicales des données textuelles qui peuvent être associées aux nœuds ou aux arêtes du graphe.

Les travaux précédents exploitent des documents RDF natifs, mais en général les entités du LOD peuvent provenir (i) soit d'un jeu de données du LOD (elles peuvent alors être rassemblées via par exemple des requêtes SPARQL), (ii) soit des annotations sémantiques intégrées à une page Web (par exemple, en utilisant RDFa, des microdonnées, ou bien des microformats), ou bien (iii) de la détection automatique des entités dans le texte d'une page Web (au moyen par exemple de DBpedia Spotlight). Or, parmi les approches qui permettent d'améliorer les snippets pour le Web des documents en utilisant le Web des données ((Haas *et al.*, 2011) (Steiner *et al.*, 2010)), aucune ne repose sur la détection automatique d'entités : seules les annotations encapsulées explicitement sont utilisées. Haas *et al.* (Haas *et al.*, 2011) utilisent des métadonnées structurées (c.à.d. des données encodées grâce au formalisme RDFa, ou par le biais de microformats) ainsi que plusieurs techniques d'extraction ad-hoc d'information pour améliorer les snippets avec des éléments multimédias, des paires clé / valeur et des fonctionnalités interactives. Ainsi, en combinant les métadonnées créées par les éditeurs des documents avec des données structurées obtenues par des extracteurs ad-hoc conçus spécialement pour quelques sites populaires, les auteurs de ce travail sont capables de construire des snippets enrichis pour de nombreux résultats d'une requête. Ils ont choisi explicitement de ne pas utiliser directement le LOD afin d'éviter le problème du transfert de confiance entre le Web des documents et le Web des données. En effet, ils soutiennent que la qualité des processus éditoriaux qui génèrent des parties du Web des données à partir du Web des documents (par exemple la transformation de Wikipedia en DBpedia) ne peut pas être contrôlée. Par conséquent, de leur point de vue, utiliser le LOD à travers un processus de détection automatique d'entités pour enrichir des snippets s'accompagnerait du risque jugé trop important d'introduire du bruit incontrôlé dans les résultats. De plus, Google Rich Snippet (Steiner *et al.*, 2010) est une initiative similaire qui s'appuie exclusivement sur les métadonnées structurées rédigées par les éditeurs des pages Web.

Enfin, une étude faite en 2012 (Bizer *et al.*, 2013) sur plus de 40 millions de sites Web du Corpus Common Crawl montre que 5,64% des sites intègrent des données structurées. Cependant, près de 50% des 10 000 premiers sites de la liste Alexa des sites Web les plus populaires possédaient des données structurées. En outre, les auteurs de l'étude affirment que (nous traduisons) : "Les sujets des données [structurées] [. . .] semblent être en majeure partie déterminés par les principaux consommateurs qui constituent la cible de ces données : Google, Yahoo !, et Bing". Ainsi, il nous semble qu'il y a aujourd'hui un besoin évident pour un algorithme qui permette d'exploiter les données structurées issues d'un processus de détection automatique d'entités dans une page Web avec une confiance suffisante dans leur qualité, et ce de manière à permettre l'émergence d'applications qui utilisent conjointement le Web des données et le Web des documents. Dans ce travail, nous proposons une telle solution à travers un algorithme d'ordonnancement des entités d'un graphe du Web des données. De plus, nous montrons l'utilité de cet algorithme en l'appliquant au contexte de la construction de snippets sémantiques.

3. LDSVD, un algorithme pour l'ordonnancement d'entités du Linked Open Data Cloud

3.1. *Introduction au LDSVD*

LDSVD (Linked Data Singular Value Decomposition) est un algorithme pour ordonner les entités d'un graphe RDF étant donnée l'expression d'un besoin d'information sous la forme d'une liste de mots clés. LDSVD est spécialement adapté au cas de graphes creux pour lesquels des données textuelles peuvent être associées aux nœuds. En effet, LDSVD utilise à la fois la structure explicite du graphe et les similarités qui peuvent être calculées à partir de l'analyse du texte associé aux nœuds. Pour fixer les idées, notons que le texte associé à chaque nœud / entité du graphe peut par exemple provenir du résumé DBpedia de l'entité. D'un point de vue algorithmique, le LDSVD correspond à une application itérative de la décomposition en valeurs singulières (SVD) d'une matrice dont les lignes représentent les entités du graphe et dont les colonnes représentent les termes dont sont constituées les valeurs textuelles associées aux entités. Ainsi, la valeur à l'intersection d'une ligne et d'une colonne de la matrice correspond à la fréquence d'un terme dans le texte associé à une entité. Avant de pouvoir introduire le LDSVD, nous devons d'abord rappeler les étapes essentielles de la décomposition en valeurs singulières (SVD) :

Algorithme S (*SVD*). Etant donnée M, une matrice $m \times n$ d'entités (lignes) et de termes (colonnes), cet algorithme calcule la décomposition en valeurs singulières de M au rang k (sur notre cas d'utilisation nous trouvons expérimentalement qu'une valeur de k égale à 3 est adaptée).

S1. [*SVD pour matrice creuse.*] Puisque la matrice M est creuse, nous utilisons l'algorithme *las2* développé par Michael W. Berry (Berry, 1992) afin de calculer la décomposition :

$M_k = U\Sigma V^T$ avec U et V des matrice orthogonales, Σ une matrice diagonale, telle que la norme de Frobenius $\|M - M_k\|_F$ soit minimisée.

S2. [*Coordonnées des entités.*] Dans le nouvel espace k-dimensionnel, appliquer à chaque entité le facteur d'échelle correspondant de Σ. Ceci correspond au produit matriciel : ΣU^T.

S3. [*Normes des entités.*] Dans ce même espace, calculer la norme euclidienne de chaque vecteur correspondant à une entité.

S4. [*Coordonnées des termes.*] Similaire à S2 (ΣV^T).

S5. [*Normes des termes.*] Similaire à S3.

S6. [*Fin.*] ∎

Après exécution de l'Algorithme S, nous obtenons une représentation des entités et des termes dans un même espace k-dimensionnel. Les nouvelles coordonnées des entités (resp. des termes) sont données par le produit ΣU^T (resp. ΣV^T). Σ est une matrice diagonale, non-décroissante et positive dont les éléments non nuls sont appelés "valeurs singulières". U (resp. V) est une matrice orthogonale (c.à.d. $U^T U = I$, avec I symbolisant la matrice identité). L'opérateur U (resp. V) étant orthogonal, il représente une rotation car il conserve les normes ($\|Ux\|_2^2 = (Ux)^T(Ux) = x^T U^T U x = x^T x = \|x\|_2^2$). L'opérateur Σ étant diagonal, il représente un étirement ou une contraction des axes de la base. On prouve que la composition de ces deux transformations est telle qu'en ne gardant que les k premières valeurs singulières de Σ, on obtient la meilleure approximation au rang k des entités et des termes. Ici, "meilleure" s'entend au sens de la norme de Frobenius (c.à.d. si nous notons Σ_k la version réduite de Σ et $A_k = U\Sigma_k V^T$, alors $\|A - A_k\|_F$ est minimisée).

Nous sommes maintenant en mesure d'introduire l'Algorithme LDSVD proprement dit. Nous commençons par exposer brièvement la propriété de la décomposition en valeurs singulières sur laquelle repose l'algorithme LDSVD. Pour une forte réduction dimensionnelle (c.à.d. pour une faible valeur de k), la transformation $\Sigma_k U^T$ tend à placer les entités qui étaient orthogonales à de nombreuses entités dans l'espace des lignes de M proches de l'origine de l'espace k-dimensionnel résultant. En effet, comme vu plus haut, le SVD peut être vu comme un algorithme d'optimisation. Or pour minimiser l'erreur due à l'impossibilité pour une entité d'être orthogonale à plus de k entités non colinéaires, cette entité doit être placée aussi proche que possible de l'origine de l'espace réduit. Un argument similaire peut être utilisé pour montrer qu'une entité colinéaire à de nombreuses entités dans l'espace des lignes de M aura aussi tendance à se trouver proche de l'origine de l'espace réduit k-dimensionnel. De plus, les mêmes arguments sont valables pour les termes. Ainsi, et pour résumé, les entités éloignées de l'origine dans l'espace réduit ont tendance à posséder des relations "intéressantes" avec d'autres entités éloignées de l'origine. Par ailleurs, les entités qui sont dans la direction de plus grande variation des données dans l'espace initial sont les mieux alignées avec l'axe correspondant à la plus grande valeur singulière dans l'espace réduit (c.à.d. l'axe qui a subi l'extension la plus forte), elles ont donc tendance à être les plus éloignées de l'origine de l'espace réduit. Ainsi, le principe

à l'œuvre pendant une itération de l'algorithme LDSVD consiste à augmenter artificiellement l'importance des entités sémantiquement proches de la requête afin de les forcer dans la direction de plus grande variation des données, et à observer parmi les autres entités celles qui suite à cette opération s'éloignent le plus de l'origine de l'espace réduit. Grâce à l'argument énoncé ci-dessus, ces dernières entités peuvent ainsi être qualifiées de potentiellement proches du besoin d'information.

Algorithme L (*LDSVD*). Etant données M_0 une matrice $m \times n$ d'entités (lignes) et de termes (colonnes), A une matrice d'adjacence $m \times m$ représentant le graphe des entités, $entitesPhares$ une liste des entités sémantiquement proches de la requête, et $termesPhares$ une liste des termes proches de la requête, cet algorithme calcule un ordonnancement des entités.

L1. [*Initialiser*] Set $i \leftarrow 0$. Set $nbEntitesStimulees \leftarrow size(entitesPhares)$. Set $nbTermesStimules \leftarrow size(termesPhares)$. A chaque entité (resp. terme), associer une valeur entière (appelée son facteur de stimulation) initialisée à 1000 pour les entités phares et à 1 pour les autres entités (c'est la différence de plusieurs ordres de grandeur entre ces deux valeurs qui compte et non les valeurs en elles-mêmes). Calculer la décomposition en valeurs singulières de M_0 à un petit rang k (e.g. 3)(Algorithme S).

L2. [*Stimuler.*] Set $i \leftarrow i + 1$. Construire M_i à partir de M_{i-1} en multipliant chaque entrée $M_{i-1}[k][l]$ par le maximum des deux facteurs de stimulation qui correspondent l'un à la ligne k et l'autre à la colonne l.

L3. [*Décomposer.*] Calculer la décomposition en valeurs singulières de M_i à un petit rang k (e.g. 3)(Algorithme S).

L4. [*Ordonner.*] Set $ordreEntites \leftarrow$ entités classées par ordre croissant de la distance dont elles se sont éloignées de l'origine de l'espace k-dimensionnel entre la décomposition de M_{i-1} et la décomposition de M_i.
Set $ordreTermes \leftarrow$ termes classés par ordre croissant etc.

L5. [*Stabiliser.*] If $i = 1$ go to L6. Si les entités stimulées continuent à impacter les positions des autres entités d'une manière propre à modifier l'ordonnancement global des entités, faire :

- Pour chaque $entitesAStimuler$, augmenter son facteur de stimulation par sa position dans l'ordonnancement des entités (e.g. la seconde entité de $ordreEntites$ verra son facteur de stimulation augmenter de 2).
- Augmenter de 1 le facteur de stimulation de chaque $termesAStimuler$.
- Go to L2.

Considérer que les entités stimulées n'influencent plus l'ordonnancement global quand l'accélération moyenne du mouvement des entités relativement à l'origine de l'espace k-dimensionnel approche 0.

L6. [*Mettre à jour.*]

- Set $q \leftarrow$ construire un vecteur binaire n-dimensionnel à partir des top-i termes de $ordreTermes$.
- Set $q_k \leftarrow \Sigma^{-1} V^T q$ (projeter q sur l'espace k-dimensionnel).
- Set $q_k \leftarrow q_k +$ la colonne de U^T qui correspond à la meilleure entité de $ordreEntites$ (c.à.d. la dernière).

- Set $entitesAStimuler \leftarrow$ les top-i entités de $ordreEntites$.
- Grâce à A ajouter à $entitesAStimuler$ les entités liées aux entités déjà présentes dans $entitesAStimuler$.
- Ordonner les $entitesAStimuler$ par ordre croissant de leur cosinus avec q_k.
- Set $termesAStimuler \leftarrow$ les top-i termes de $ordreTermes$.
- Set $nbEntitesStimulees \leftarrow nbEntitesStimulees + entiteStimulationQuantum$ ($entiteStimulationQuantum$ est fixé expérimentalement à 1.)
- Set $nbTermesStimules \leftarrow nbTermesStimules + termeStimulationQuantum$ ($termeStimulationQuantum$ est fixé expérimentalement à 10.)
- If ($nbEntitesStimulees <= m$ and $nbTermesStimules <= n$) go to L2.

L7. [*Fin.*] Retourner une liste des entités ordonnées selon la distance dont elles se sont éloignées de l'origine de l'espace réduit au cours de l'exécution de l'algorithme. ▮

Ajoutons aux explications précédentes que l'algorithme LDSVD prend en compte la structure du graphe en stimulant également les entités directement liées aux entités qui se sont le plus éloignées de l'origine de l'espace réduit (voir l'étape L6 de l'algorithme L).

Enfin, les entités considérées comme importantes car elles se sont éloignées de l'origine de l'espace réduit peuvent pourtant être orthogonales au besoin d'information. Ainsi, étant donné un vecteur q_k qui combine les entités et les termes les mieux classés pour l'itération courante, l'algorithme LDSVD donne plus d'importance aux entités dont la valeur du cosinus avec q_k est forte (voir l'étape L6 de l'algorithme L).

3.2. *Evaluation de l'algorithme LDSVD*

Protocole expérimental

Etant donnée la connaissance d'un besoin d'information exprimé sous la forme d'une requête faite d'un ensemble de mots clés, l'algorithme LDSVD fournit un ordonnancement des entités d'un graphe RDF pour lequel des données textuelles sont associées aux nœuds. L'algorithme utilise à la fois la structure explicite du graphe et les relations implicites découvertes par analyse du texte associé aux nœuds afin de calculer un ordonnancement des entités. Grâce à cette stratégie double, il est particulièrement adapté aux graphes creux et bruités (c.à.d. avec une proportion importante de nœuds non pertinents étant donné le besoin d'information exprimé par la requête). De tels graphes apparaissent en particulier suite à l'annotation automatique d'une page Web (par exemple, via DBpedia, AlchemyAPI,...). Or, il a été récemment montré (Roa-Valverde et Sicilia, 2014) qu'il n'existe pas aujourd'hui de jeux de données adaptés pour l'évaluation d'algorithmes d'ordonnancement sur le Web des données. Cette remarque est d'autant plus valable dans notre contexte plus spécifique de graphes bruités et creux. Ainsi, nous avons construit notre propre jeu de données pour l'évaluation de l'algorithme LDSVD (disponible librement en ligne[2]). Nous avons d'abord

2. `http://134.214.104.90:8080/ensenTensorielWeb/`

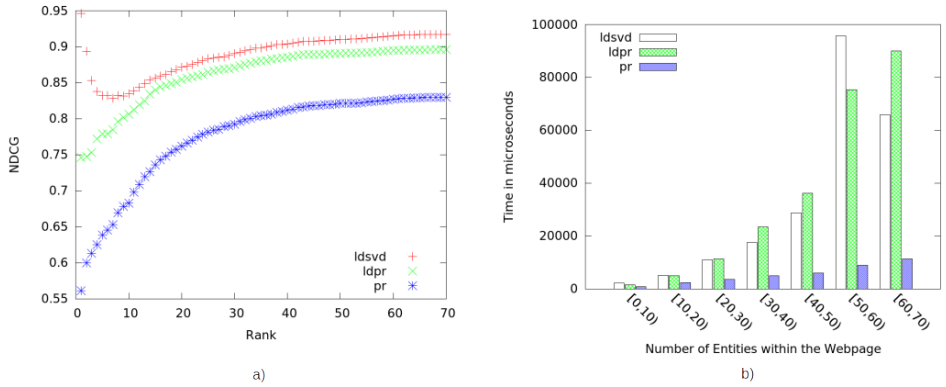

Figure 1. *a) Comparaison des valeurs de NDCG pour LDSVD, LDPR et PR, b) Comparaison des performances de LDSVD, LDPR et PR pour 100 exécutions de chaque algorithme sur chaque page Web, avec une configuration Intel Core i5-4570 CPU 3.20GHz x 4*

sélectionné un ensemble de 20 requêtes (sélectionnées au hasard parmi les éléments de l'historique des auteurs de cet article). Pour chaque requête, nous avons téléchargé les 5 meilleurs pages Web de la page de résultats du moteur de recherche Google. Les pages sont en langue anglaise. Pour chaque page Web, nous avons extrait le texte non balisé à partir de l'encodage HTML en utilisant l'algorithme introduit par Kohlschütter et al. (Kohlschütter *et al.*, 2010). Ensuite, nous avons annoté ce texte avec des entités de DBpedia en utilisant DBpedia Spotlight (Mendes *et al.*, 2011) (chaque page Web est annotée avec en moyenne 27 entités). Nous avons ensuite offert à l'évaluateur humain une interface pour visualiser la requête et le texte annoté, il devait alors donner à chaque entité annotée une note sur l'échelle suivante : "hors sujet", "pertinent", "très pertinent". Pour l'aider dans sa prise de décision, l'évaluateur pouvait également accéder au résumé DBpedia et aux classes de chaque entité. Ce jeu de données a été généré manuellement par l'ensemble des auteurs de l'article avec une procédure de validation croisée et une résolution manuelle des conflits jusqu'à obtention d'un consensus total.

Résultats

Nous avons comparé LDSVD à deux autres algorithmes : l'algorithme PageRank (Page *et al.*, 1999) (PR), et une version modifiée du PageRank (LDPR, pour Linked Data Page Rank) qui prend en compte les relations implicites découvertes à partir de l'analyse du texte associé aux entités.

Rappelons que l'algorithme PageRank transforme la matrice d'adjacence du graphe en une matrice stochastique et primitive afin de pouvoir calculer l'état stable de la chaîne de Markov correspondante en faisant par exemple appel à l'algorithme de la puissance itérée (Page *et al.*, 1999). Pour rendre stochastique la matrice d'adja-

cence, il faut (i) diviser chaque entrée non nulle d'une ligne non vide par la somme des valeurs de la ligne, et (ii) remplacer chaque ligne vide par un vecteur de transition équiprobable. Pour rendre primitive la matrice stochastique, le PageRank propose de combiner linéairement la matrice stochastique avec une matrice dite de "téléportation" pour laquelle chaque transition est équiprobable.

La grande majorité des algorithmes existants pour l'ordonnancement des entités d'un graphe provenant du Web des données sont basés sur une modification du PageRank (voir par exemple les états de l'art récents : (Roa-Valverde et Sicilia, 2014) et (Jindal *et al.*, 2014)). Ainsi, OntologyRank (Ding *et al.*, 2004) (utilisé par le moteur de recherche sémantique Swoogle[3]) modifie la matrice de téléportation pour prendre en compte les différents types de liens entre ontologies. Similairement, PopRank (Nie *et al.*, 2005) propose une version modifiée du PageRank qui considère les différents types de prédicats entre entités. Finalement, RareRank (Wei *et al.*, 2011) modifie la matrice de téléportation pour prendre en compte des relations thématiques entre entités qui peuvent être inférées à partir d'ontologies.

Ces approches ne correspondent pas exactement à notre contexte (c.à.d. un graphe d'entités creux et bruité avec des données textuelles associées à chaque entité et en présence d'un besoin d'information exprimé sous la forme d'un ensemble de mots clés). Cependant, elles s'y adaptent facilement. Ainsi, nous proposons une version modifiée du PageRank (LDPR) que l'on utilisera en comparaison du LDSVD. L'algorithme LDPR exploite les mêmes données en entrée que celles accessibles à l'algorithme LDSVD c.à.d., la requête, la matrice d'adjacence A et une matrice entités-termes M obtenue à partir des résumés DBpedia des entités et également à partir de passages de la page Web centrés sur les occurrences des entités. Premièrement, LDPR utilise M pour informer la transformation de A en une matrice stochastique. Pour chaque ligne non vide i de A, les valeurs non nulles $A[i][j]$ ne représentent pas des transitions équiprobables (comme c'est le cas pour le PageRank) : la probabilité de la transition est proportionnelle à la similarité cosinus entre l'entité i et l'entité j, plus elles sont similaires plus la transition est probable. Deuxièmement, LDPR utilise également M lors de la construction de la matrice de téléportation : les transitions ne sont pas équiprobables mais la probabilité d'une transition vers une entité e est proportionnelle au maximum de la similarité cosinus entre e et l'ensemble des entités qui ont été identifiées a priori comme proche de la requête. Cette détermination des entités proches de la requête est faite de la même manière que pour LDSVD : nous procédons à une détection d'entités dans les termes de la requête, en utilisant dans notre cas DBpedia Spotlight.

Afin de comparer les trois algorithmes (LDSVD, LDPR et PR), nous utilisons la métrique NDCG (Normalized Discounted Cumulative Gain). Le DCG (Discounted Cumulative Gain) au rang r est défini ainsi : $DCG_r = rel_1 + \sum_{i=1}^{r} \frac{rel_i}{log_2 i}$ (avec rel_i représentant la pertinence estimée du résultat au rang i, la pertinence est représentée sur une échelle discrète qui contient généralement bien moins de valeurs que le

3. `http://swoogle.umbc.edu/`

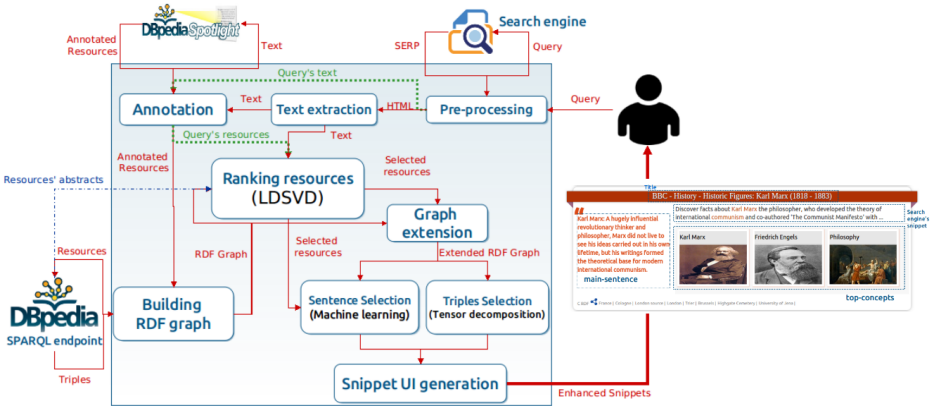

Figure 2. *Architecture logique du système ENsEN*

nombre total de résultats à ordonner, nous utilisons une échelle à trois gradations : “hors sujet”, “pertinent”, “très pertinent”). Le NDCG au rang r est égal au DCG au rang r normalisé par le DCG au rang r d’un ordonnancement idéal. La Figure 1 présente les résultats de cette évaluation. Pour des rangs inférieurs à 10, l’algorithme LDSVD est significativement meilleur que l’algorithme LDPR.

De plus, nous avons évalué les performances des trois algorithmes (voir Figure 1). LDSVD et LDPR ont des performances comparables. Comme attendu, PR montre de bien meilleures performances car il ne calcule pas les similarités cosinus requises par LDPR.

4. ENsEN, Enhanced Search Engine

Afin de convaincre de l’utilité et de l’efficacité de l’algorithme LDSVD, nous l’utilisons dans le contexte de ENsEN (Enhanced Search Engine), un système que nous avons conçu pour construire des snippets sémantiques (voir Figure 2, une démonstration *live* du système est disponible en ligne, voir une précédente note de bas de page pour l’URL). Nous présentons maintenant le workflow qui produit un snippet sémantique à partir d’une requête, et ce afin de mettre en avant le rôle joué par l’algorithme LDSVD dans ce processus.

Etant donnée une requête, nous obtenons la page de résultats d’un moteur de recherche du Web (nous avons utilisé Google pour nos expériences). Pour chaque page Web résultat, nous utilisons le service de détection automatique d’entités DBpedia Spotlight afin d’obtenir un ensemble d’entités. De la même manière, nous trouvons les entités associées aux termes de la requête. A partir de cet ensemble d’entités et par l’intermédiaire de requêtes émises auprès d’un point d’accès SPARQL de DBpedia,

nous obtenons un graphe en trouvant toutes les relations qui existent entre ces entités dans le jeu de données DBpedia.

A chaque entité nous associons un texte obtenu par concaténation de son résumé DBpedia et de fenêtres textuelles centrées sur les occurrences de l'entité dans la page Web (nous utilisons des fenêtres de 300 caractères). De plus, nous supprimons les mots vides et appliquons un enracineur[4]. Nous exécutons l'algorithme LDSVD avec en entrée ce graphe aux nœuds décorés de texte ainsi que les entités découvertes dans la requête, et nous récupérons un ordonnancement des entités. Pour chacune des meilleures entités, nous construisons des vignettes affichées sur le snippet.

A partir d'un point d'accès SPARQL de DBpedia, nous procédons à une extension 1-hop des meilleures entités afin d'augmenter le nombre de triplets parmi lesquels nous cherchons ensuite les plus importants en terme d'une analyse des liens du graphe. Pour ce faire, nous construisons un tenseur d'ordre 3 à partir du graphe étendu : chaque prédicat correspond à une couche horizontale d'ordre 2 qui représente la matrice d'adjacence pour la restriction du graphe à ce prédicat. Nous calculons ensuite la décomposition PARAFAC du tenseur en une somme de facteurs (qui sont des tenseurs de rang 1 et d'ordre 3), et nous l'interprétons d'une façon similaire à Franz *et al.* (Franz *et al.*, 2009) : pour chacune des meilleures entités (au sens du LDSVD), nous sélectionnons les facteurs auxquels elle contribue le plus (en tant que sujet ou en tant qu'objet), et pour chacun de ces facteurs, nous sélectionnons les triplets qui ont les prédicats avec les meilleurs scores. Ainsi, nous sommes capables d'associer à chacune des meilleures entités un ensemble de triplets que nous faisons apparaître sur le snippet dans la description de l'entité.

Enfin, nous avons employé une approche par apprentissage automatique afin de sélectionner de courts passages de la page Web qui viennent accompagner la description de chaque entité. Nous avons choisi des variables basées sur la requête, le texte de la page Web, et l'ordonnancement des entités. Nous avons mis en place un processus de sélection de variables : nous avons utilisé une métrique de gain d'information pour sélectionner un petit ensemble de variables que nous avons ensuite utilisé comme ensemble de départ pour une approche de type *wrapper* (en mode *forward*). Nous avons observé que les variables dérivées de l'ordonnancement produit par le LDSVD appartiennent systématiquement aux variables conservées. Ceci nous semble fournir une preuve supplémentaire, même si indirecte, de l'utilité du LDSVD. Dans le contexte de cette présentation, nous ne pouvons pas décrire en détails le processus d'apprentissage mais plus de détails sont accessibles en ligne (voir une précédente note de bas de page pour l'URL).

4. `http://snowball.tartarus.org/`

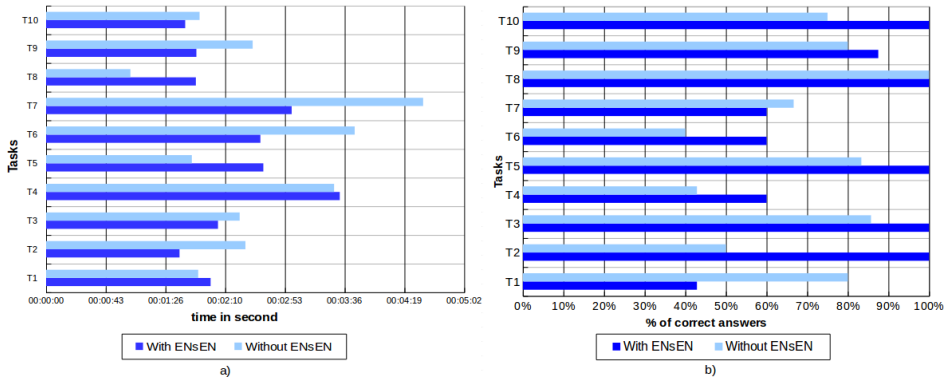

Figure 3. *a) Temps passé pour répondre aux tâches, b) Nombre moyen de réponses correctes*

5. Evaluation utilisateurs

5.1. *Evaluation de l'utilité*

Protocole expérimental

Nous avons procédé à une expérimentation en environnement contrôlé avec 14 participants âgés de 24 à 40 ans. 11 d'entre eux travaillent dans le domaine des technologies de l'information. A chaque participant nous avons confié 10 tâches propres à susciter un processus non trivial de recherche d'information (par exemple, "Nommer les deux états des Etats-Unis d'Amérique qui ont connu le plus grand nombre d'habitants infectés par la fièvre de la vallée."). Le formulaire utilisé pour introduire les tâches aux participants est disponible en ligne (voir une précédente note de bas de page pour l'URL). Chaque participant a dû résoudre la moitié de ses tâches en utilisant ENsEN, et l'autre moitié par d'autres moyens de son choix (par exemple, en utilisant un moteur de recherche traditionnel). Nous nous sommes assurés que chaque tâche soit réalisée par au moins deux participants, une fois avec et une fois sans l'aide de ENsEN. Nous avons mesuré le temps passé pour compléter la tâche (avec une borne supérieure fixée à 5 minutes). Seules les réponses complètes ont été considérées comme correctes.

Résultats

Les résultats de cette expérience apparaissent sur la Figure 3. En moyenne, les participants ont passé 1,44 fois plus de temps et ont obtenu un taux de succès plus faible (70% contre 81%) lorsqu'ils n'utilisaient pas ENsEN. Cette différence est d'autant plus marquée pour les tâches qui nous semblent pouvoir être considérées comme les plus difficiles (par exemple, "Quels rapports trouve-t-on entre les concepts introduits par Freud et ceux introduits par Nietzsche ?"). Pour certaines tâches (par exemple,

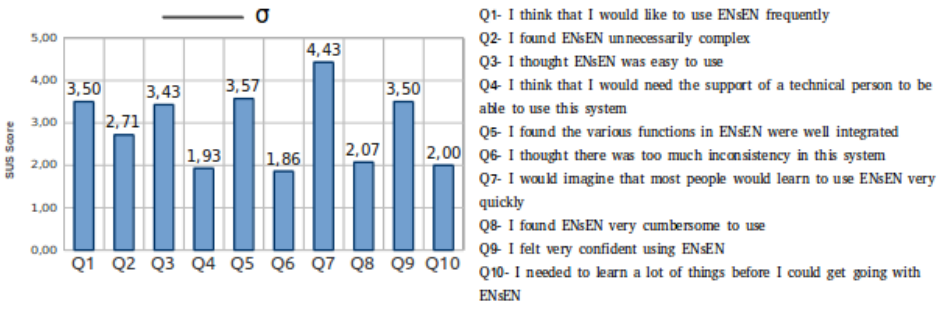

Figure 4. *Evaluation de l'utilisabilité sur l'échelle SUS*

"Qui fut le successeur de James Blaine ?"), ENsEN fournissait directement la réponse dans la description associée à la meilleure entité, et dans ces cas les temps de réponses étaient courts ($< 10s$) et les réponses toutes correctes.

5.2. *Evaluation de l'utilisabilité*

Protocole expérimental

Nous avons utilisé l'échelle psychométrique intitulée "System Usability Scale" (SUS) (Lewis et Sauro, 2009) afin d'obtenir une vision globale de l'utilisabilité du système ENsEN. Chaque participant a dû noter ses réactions subjectives au sujet de l'utilisabilité de ENsEN en répondant à un ensemble de 10 questions. Le questionnaire est accessible en ligne (voir une précédente note de bas de page pour l'URL).

Résultats

Un score SUS moyen supérieur à 68 indique une bonne utilisabilité. Le score moyen pour ENsEN est de 69,6. La Figure 4 offre plus de détails. Les participants ont trouvé ENsEN intéressant et bénéficiant d'une bonne intégration dans leur processus de recherche. Même étant donnée notre IHM non conventionnelle, ils ressentent qu'ils pourront apprendre rapidement à utiliser le système. Le score un peu plus faible en réponse à la question Q2 peut s'expliquer par cela que les participants sont habitués à des IHM minimalistes (comme par exemple celle du moteur de recherche Google).

6. Conclusion

Nous avons proposé un nouvel algorithme, LDSVD (Linked Data Singular Value Decomposition), pour l'ordonnancement des entités d'un graphe du Web des données qui peut être creux et bruité, mais pour lequel des données textuelles descriptives sont associées aux nœuds, et avec connaissance d'un besoin d'information exprimé sous la

forme d'un ensemble de mots clés. De tels graphes apparaissent en particulier suite à un processus de détection automatique d'entités du Web des données au sein d'une page Web (par exemple, à travers l'utilisation de DBpedia Spotlight). Notre approche se caractérise par la prise en compte à la fois de la structure explicite offerte par le Web des données, et des relations implicites qui peuvent être trouvées par analyse du texte de la page Web. LDSVD construit des ordonnancements d'une qualité significativement meilleure à celle des ordonnancements produits par les approches de l'état de l'art qui sont fondées sur des modifications de l'algorithme PageRank. De plus, nous avons appliqué cet algorithme au cadre applicatif de la construction de snippets sémantiques. Dans ce contexte, la bonne précision de l'algorithme LDSVD a permis d'obtenir des snippets sémantiques utiles et utilisables, ouvrant la voie à de nouvelles applications qui pourront bénéficier des apports mutuels du Web des données et du Web des documents. Par exemple, des travaux futurs évalueront le potentiel de cette approche pour la recherche d'information exploratoire.

7. Bibliographie

Bai X., Delbru R., Tummarello G., « RDF snippets for Semantic Web search engines », *On the Move to Meaningful Internet Systems : OTM 2008*, Springer, p. 1304-1318, 2008.

Berry M. W., « Large-scale sparse singular value computations », *International Journal of Supercomputer Applications*, vol. 6, n° 1, p. 13-49, 1992.

Bizer C., Eckert K., Meusel R., Mühleisen H., Schuhmacher M., Völker J., « Deployment of rDFa, microdata, and microformats on the web–A quantitative analysis », *The Semantic Web–ISWC 2013*, Springer, p. 17-32, 2013.

Dali L., Fortuna B., Duc T. T., Mladenić D., « Query-independent learning to rank for rdf entity search », *The Semantic Web : Research and Applications*, Springer, p. 484-498, 2012.

Ding L., Finin T., Joshi A., Pan R., Cost R. S., Peng Y., Reddivari P., Doshi V., Sachs J., « Swoogle : a search and metadata engine for the semantic web », *Proceedings of the thirteenth ACM international conference on Information and knowledge management*, ACM, p. 652-659, 2004.

Franz T., Schultz A., Sizov S., Staab S., « Triplerank : Ranking semantic web data by tensor decomposition », *The Semantic Web-ISWC 2009*, Springer, p. 213-228, 2009.

Ge W., Cheng G., Li H., Qu Y., « Incorporating compactness to generate term-association view snippets for ontology search », *Information Processing & Management*, vol. 49, n° 2, p. 513-528, 2012.

Haas K., Mika P., Tarjan P., Blanco R., « Enhanced results for web search », *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, ACM, p. 725-734, 2011.

Jindal V., Bawa S., Batra S., « A review of ranking approaches for semantic search on Web », *Information Processing & Management*, vol. 50, n° 2, p. 416-425, 2014.

Kleinberg J. M., « Authoritative sources in a hyperlinked environment », *Journal of the ACM (JACM)*, vol. 46, n° 5, p. 604-632, 1999.

Kohlschütter C., Fankhauser P., Nejdl W., « Boilerplate detection using shallow text features », *Proceedings of the third ACM international conference on Web search and data mining*, ACM, p. 441-450, 2010.

Lempel R., Moran S., « SALSA : the stochastic approach for link-structure analysis », *ACM Transactions on Information Systems (TOIS)*, vol. 19, nº 2, p. 131-160, 2001.

Lewis J. R., Sauro J., « The factor structure of the system usability scale », *Human Centered Design*, Springer, p. 94-103, 2009.

Mendes P. N., Jakob M., García-Silva A., Bizer C., « DBpedia spotlight : shedding light on the web of documents », *Proceedings of the 7th International Conference on Semantic Systems*, I-Semantics '11, ACM, p. 1-8, 2011.

Nie Z., Zhang Y., Wen J.-R., Ma W.-Y., « Object-level ranking : bringing order to web objects », *Proceedings of the 14th international conference on World Wide Web*, ACM, p. 567-574, 2005.

Page L., Brin S., Motwani R., Winograd T., « The PageRank citation ranking : Bringing order to the web. », 1999.

Penin T., Wang H., Tran T., Yu Y., « Snippet generation for Semantic Web search engines », *The Semantic Web*, Springer, p. 493-507, 2008.

Roa-Valverde A. J., Sicilia M.-A., « A survey of approaches for ranking on the web of data », *Information Retrieval*, vol. 17, nº 4, p. 1-31, 2014.

Steiner T., Troncy R., Hausenblas M., « How Google is using linked data today and vision for tomorrow », *Proceedings of Linked Data in the Future Internet*, 2010.

Wei W., Barnaghi P., Bargiela A., « Rational research model for ranking semantic entities », *Information Sciences*, vol. 181, nº 13, p. 2823-2840, 2011.