Discovery Hub: a discovery engine on the top of DBpedia using real-time spreading activation

Nicolas Marie[1 2], Fabien Gandon[1], Myriam Ribière[2], Florentin Rodio[2]

[1] INRIA Sophia-Antipolis, Wimmics team,
2004 Route des Lucioles, Sophia-Antipolis, 06410 BIOT
{nicolas.marie,fabien.gandon }@inria.fr
[2] Alcatel-Lucent Bell Labs,
7 route de Villejust, 91260 NOZAY
{nicolas.marie,myriam.ribiere,florentin.rodio}@alcatel-lucent.com

Abstract. Web growth, both in size and diversity, and users' growing expectations increase the need for innovative search approaches and technologies. Exploratory search systems are built specifically to help user in cognitive consuming search tasks like learning or investigation. Some of these systems are built on the top of linked data and use its semantic richness to provide cognitively-optimized search experiences. This paper addresses the question of real-time linked data processing for exploratory search purposes. This real-time processing offers advantages in terms of data dynamicity-handling and flexible query expression. To achieve this goal we propose a semantic spreading activation algorithm which does not require a priori manual settings or periodical results pre-processing. The paper details algorithm behavior studies over DBpedia dataset and describes a real deployment introducing the Discovery Hub prototype. Finally evaluations of algorithm and interface relevance's through user's experiments are presented.

Keywords: Semantic web, linked data, DBpedia, spreading activation, semantic spreading activation, exploratory search system, discovery engine

1 Introduction

Exploratory search [12] refers to expensive cognitive search tasks like learning or investigating. In this case search objective is fuzzy and the user manipulates iteratively an evolving set of keywords. He has to synthesize an important amount of information coming from an unstable result space. According to [12] actual search engines are very efficient for lookup tasks (when user precisely know the results he wants i.e. known item search, query answering) but less for exploratory search ones due to their keyword oriented search paradigm. Author proposed to complete existing search solutions with systems cognitively optimized for exploratory tasks. To optimize the search experience some exploratory search systems are built on top of knowledge bases. Some of them make use of semantic knowledge sources including

linked open data[2]. Linked data-based approaches involve the use of (1) semantic web technologies where *semantic web* is the web augmentation by formal metadata giving to software the access to some semantic facets of information. Semantic web data are expressed according to ontologies. The main semantic web standards include RDF, a graph model with XML syntax to describe resources, SPARQL language for querying RDF bases, RDFS and OWL for modeling ontologies schemas; and (2) *Linked open data cloud* (LOD) dataset(s) where LOD is a web of interconnected public datasets published in RDF. Among all the LOD datasets DBpedia [1] is the most popular and used one. DBpedia results from the extraction of data from Wikipedia[3] then published in RDF following the LOD principles.

Beyond actual related works, presented in section 2, we propose to address the dynamic nature of Linked data. Indeed, even DBpedia, an encyclopaedic based knowledge source, is evolving quickly in term of ontology models[4] and instances[5]. Thus there is a need to value these data in real-time. Our objective is to perform on-the-fly linked data processing for exploratory search. Section 3 of the paper details our formal proposition, and section 4 describes the corresponding algorithm implementation in the Discovery Hub prototype including approximation strategies. For validation purpose, section 5 presents the hypotheses, protocols and results of the RTSA algorithm and the interface evaluations.

2 Related work

In this part we present related works supporting discovery and exploratory search with the web of data. They belong to the broader category of semantic search systems. Semantic search can be defined as "*search approaches that broadly speaking, use semantics to improve the search experience*" [18]. Semantic search is always based on explicit semantic processing but approaches vary a lot in tackled information needs, information resources representation, query representation and results computation. For an extensive survey about semantic search, readers may refer to [18]; in this article we focus on works sharing the following properties:

- *Fuzzy information need for knowledge discovery in learning or leisure situations*. We focus on two categories of systems that are recommenders and exploratory search systems.
- *Resource input query paradigm*: such systems take one or several resource(s) as input(s) and retrieve related and meaningful other resources. After computation, result-resources are rendered and constitute the output or serve to identify the final retrieved content (e.g. Youtube[6] videos thanks to API call).
- *Linked data processing*: such systems use LOD graphs, mainly DBpedia, as the primary material for processing.

[2] http://linkeddata.org
[3] http://wikipedia.org
[4] http://blog.dbpedia.org/2012/08/06
[5] http://live.dbpedia.org/LiveStats/
[6] http://www.youtube.com

Table 1. Selection of closest state-of-the art research systems summary

	Yovisto	Seevl	MORE	Aemoo	Kaminskas & al.
Type	Exploratory search	Recommender	Recommender	Exploratory search	Recommender
Imple-mention	Web application	Chrome extension	Facebook[7] application	Web application	No public application
Purpose	Exploratory feature for a video platform	Musical discovery	Film discovery	Exploratory search	Cross-domain recommendation
Data	DBpedia EN+DE subset	DBpedia EN subset	DBpedia EN subset	DBpedia EN, Twitter[8] and Google news[9]	DBpedia EN subset
Multi-domain	Yes	No, music domain	No, cinema domain	Yes	Cross-domain, 2 domains at least
Query expre-ssion	Keywords entity recognition	Youtube page entity recognition	Entity search	Entity search	Entity selection in a pre-processed list
Multi-inputs	No	No	Yes (results union)	No	No (uncertain)
Proce-ssing	Set of heuristics	DBrec algorithm	sSVM algorithm	EKP filtered view	Weighted sprea-ding activation
Ranking	Yes	Yes	Yes	No	Yes
Explana-tions	No	Shared properties	Shared properties	Wikipedia-based	Path-based
Real-time	No	No	No	No, EKP	No

In the past 3 years several research initiatives showed the interest to use the LOD for discovery and exploratory search objectives. For comparison purpose these works are presented in table 1. Seevl[10], MORE[11] and [8] are *Linked Data-based recommenders*, Seevl is a DBpedia-based band recommender for Youtube. It proposes a "*related*" function offering bands recommendations based on the DBrec algorithm [14]. The DBrec algorithm ranks related resources according to direct or indirect shared properties with the seed resource. The ranking is processed offline and stored in RDF. MORE [6] is a DBpedia-based film recommender. It uses a semantic adaptation of the vector space model called sSVM. The more features movies share the more similar they are. The ranking is processed offline. According to [6], sSVM was evaluated as more accurate than DBrec on film. [8] proposed a method to perform cross-recommendations on at least two chosen domains. Authors tested it on DBpedia data using the scenario of musical recommendation starting from tourists' attractions. The recommendation process is operated offline using weighted spreading activation. The positive evaluation results confirmed the potential of LOD for cross-domain, cross-type recommendations.

The second category of systems presented in table 1 is *exploratory search systems* one. Those applications differ from recommenders by the user active role in the

[7] http://www.facebook.com
[8] http://twitter.com
[9] http://news.google.com
[10] http://seevl.net/
[11] http://apps.facebook.com/new-more/

discovery process. *Yovisto*[12] [19] is an academic video platform offering an exploratory search feature. It proposes a ranked list of related resources besides search results. The selection and ranking of resources is computed offline thanks to a set of eleven heuristics. *Yovisto* user evaluation showed that the exploratory search feature significantly improved the video search activity efficiency. *Aemoo* [13] is a DBpedia-based exploratory search system which uses Encyclopedic Knowledge Pattern (EKP). EKP are knowledge patterns which define typical classes used to describe entities of a certain type. They are built thanks to an offline graph analysis. Starting from a resource of interest, *Aemoo* presents its direct neighborhood filtered with its corresponding EKP or inverted EKP when using the *curiosity* function. It illustrates that DBpedia can serve as a basis to retrieve both common and peculiar knowledge. Last but not least it is noticeable that a major industrial search player, Google[13], launched a feature in the spirit of exploratory search ("*explore your search*", "*things not strings*"[14]). It performs entity recognition on search keywords. Then it provides structured information using the Google Knowledge Graph[15] and recommendations based on collaborative filtering ("*people also search for*").

Having a general view on table 1 we can try to identify trends. The research initiatives seem to be more and more focused on multi-domain processing. DBpedia is from far the most used dataset but its processing varies a lot. The initiative of an industrial player offered the first large deployment of an exploratory search feature. It is noticeable that all these initiatives process the results offline or require periodical expensive offline processing. In this paper, we stress the advantages of a real-time linked data processing for exploratory search and explore its possibility.

3 Real Time Spreading Activation for exploratory search

The novelty in this paper is to focus on the challenge of a real-time linked-data processing for exploratory search purpose. Starting from user interests, our objective is to retrieve and render meaningful results in a few seconds without manual pre-settings and/or results pre-processing (partial or total). In other words we want to make sense from the graph on-the-fly. The real-time aspect brings two major advantages.

First, the live processing is a way to handle dynamic data by serving up-to-date results taking into account schema and data variations. It ensures that the freshest data available were taken into account to compute results. It prevents from any delay or rupture between the explored linked data source and the user e.g. schema change, instances addition.

Second, a rich graph like DBpedia supports a very large amount of inputs combination and query tuning possibilities. The settings variations based on graph information are quasi-"*infinite*" e.g. reinforcing or minimizing the importance of a set of relations, discarding certain information facets for instance. The resulting fine-grained querying potential has a high value for user as it can support advanced search

[12] http://www.yovisto.com, feature presented in [19] seems not to be accessible online anymore
[13] www.google.com
[14] http://www.google.com/insidesearch/features/search/knowledge.html

functions, personalization and contextualization. This fine and personalized graph exploitation is not easily reachable through results pre-processing as it can take many forms.

The difficulty of processing on-the-fly linked data is due to the graph complexity. It requires smart strategies to make sense from a large set of heterogeneous triples in only few seconds. We notably want the processing not to be domain-limited. For that purpose we choose spreading activation algorithm as a basis to ground our solution.

3.1 Spreading activation

Spreading activation is designed for semantic networks and proved many times its value for information retrieval purposes. It can easily be tuned and can integrate various constraints including, for example, a semantic sensitiveness measure. It also showed efficient response time on large graphs.

Spreading activation is an algorithm family having its roots in cognitive psychology. It originally aimed to model human remembering process [3]. It inspired a lot of algorithms in various fields and was successfully employed in information and knowledge retrieval. Early and important works include [2] and [5]. A lot of variants exist but the core functioning is always the same: first a stimulation value is assigned to one or several node(s) of interest (in our case the user interest). Then this value is propagated to neighbor's node(s). The value assigned to neighbors depends on the algorithm purpose, settings and heuristics. During the next iteration the propagation continues from newly activated nodes. This process is repeated till a stop condition is reached e.g. time limit, maximum number of nodes activated or iterations.

Many researchers used spreading activation to perform information retrieval on RDF graphs. Notable works include [15] and [16]. [7] describes high performance approximate spreading activation over a very large LOD dataset. They achieved the activation of millions nodes in few seconds. Nevertheless approximations they use are not accurate enough to be exploited in a knowledge retrieval context.

3.2 Formal proposition

To achieve our goal of real-time spreading activation on linked data graph we propose below a spreading activation adaptation designed to be performed on-the-fly i.e. which do not require periodical settings or pre-processing and exploits the graph semantics on real-time. Prior to the algorithm description and we introduce necessary definitions on RDF triples and classic graph functions we use:

Definition 1. (RDF triple, RDF graph). Given U a set of URI, L a set of plain and typed Literal and B a set of blank nodes. A RDF triple is a 3-tuple *(s, p, o)* $\in \{U \cup B\} \times U \times \{U \cup B \cup L\}$. s is the node subject of the RDF triple, p the predicate of the triple and *o* the node object of the triple. A RDF graph is a set of RDF triple.

Definition 2. (Node degree) $degree_j$ is the edges number of the node j.

$$degree_j = |\{ (j,p,x) \in KB \} \cup \{(x,p,j) \in KB\}|$$

Definition 3. (Node depth) $depth(t)$ uses the subsumption schema hierarchy in order to compute the depth of a type t. It is used to identify the most precise type(s) available for a node.

$$depth(t) = \begin{cases} depth(t) = 0 \; if \; t = \text{T the root of the hierarchy,} \\ depth(t) = 1 + Min_{\; s_t;(t,rdf:subClassOf,s_t)\in KB} depth(s_t) \text{ otherwise} \end{cases}$$

Definition 4. (Node neighborhood) $Neighbor(i)$ is the set of direct instances neighbors of node o in the linked data graph:

$$Neighbor(i) = \{x \; ; ((i,p,x) \in KB \; \vee \; (x,p,i) \in KB \;) \wedge p \neq rdf:type\}$$

Based on those definitions, we present the formal description of the Real Time Spreading activation algorithm (RTSA):

$$a(i,n+1,o) = s(i,n,o) + \sum_j w(i,o) * \frac{a(j,n,o)}{degree_j}$$

Where:

- o is the origin i.e. the node of interest initially stimulated, i is an arbitrary node of the graph, *n* is the current number of iterations.
- $a(i,n+1,o)$ is the activation of node i at iteration $n+1$ for an initial stimulation at o.
- $s(i,n,o)$ is the stimulation of node i at iteration n. Nodes with a positive stimulation are origin nodes. $s(i,n,o) = 1$ if $i = o$ and $n = 0$ and 0 otherwise.
- $a(j,n,o)$ *i*s the activation from the neighbor node *j* for a origin o at iteration n.
- $degree_j$ returns the degree of the node *j* (def. 2).
- $w(i,o)$ is a semantic weighting function which takes into account the semantics of nodes i and o. First, $w(i,o)$ aims to identify the propagation domain: nodes are activated depending on their types. Second, $w(i,o)$ encourages the activation of nodes similar to origin o using others semantics attributes. It is detailed below.

Let *KB* be the set of all typed triples in the triple store (def. 1). The classes' propagation domain identification noted $CPD(o)$ is the set of types through which the propagation spreads. To be precise, the propagation spreads through instances which have at least one type in $CPD(o)$. It aims to increase the results quality by focusing the activation distribution on relevant nodes only and at the same time to improve performance by narrowing the amount of considered nodes. The propagation domain is identified in real-time before the propagation starts thanks to seed node neighborhoods types. $Tmax(x)$ is the set of the deepest types *t* of a given node *x* according to their $depth(t)$ (def. 3):

$$Types(x) = \{t; (x,rdf:type,t) \in KB\}, \quad Tmax(x) = \begin{cases} t \in Types(x) \; ; \\ \forall t_i \; \in \; Types(x) \; ; \\ depth(t) \geq depth(t_i) \end{cases}$$

$NT(o)$ is a multiset counting occurrences of the deepest types in the seed node neighborhood (def 4). $NT(o)$ allows filtering $Neighbor(o)$ to obtain the final CPD.

$$NT(o) = \begin{Bmatrix} (t,c); t \in Tmax(x); \\ n \in Neighbor(o); c = |\{n \in Neighbor(o); t \in Tmax(n)\}| \end{Bmatrix}$$

A threshold function can be applied to exclude anecdotic types or to limit propagation domain size for performance purpose. After this last operation we obtain the classes' propagation domain $CPD(o)$ i.e. only nodes with a type included in $CPD(o)$ will be activated during propagation:

$$CPD(o) = \left\{ (t,c) \in NT(o); \frac{c}{\sum_{(n_i,c_i) \in NT(o)} c_i} \geq threshold \right\}$$

In addition to types information we use a triple-based measure to improve the algorithm relevance. The more a node is a subject of triples that share a property and an object with triples involving the origin node as a subject, the more it will receive activation:

$$w(i,o) = \begin{Bmatrix} 0 \; if \nexists t \in Types(i); t \in CPD(o) \\ 1 + |commontriple(i,o)| \; otherwise \end{Bmatrix}$$

Where: $commontriple(i,o) = \{triple; (i,y,z) \in KB \wedge (o,y,z) \in KB\}$

4 RTSA implementation

This part is dedicated to the implementation of the RTSA algorithm in the Discovery Hub application. It describes general algorithm architecture, settings, approximation strategies used and finally presents the application.

4.1 Architecture

The algorithm is coded in JAVA. Each time a query is processed a Kgram [4] inference engine instance is created. This *local* instance manipulates a limited sub-graph replicated from the DBpedia SPARQL endpoint. Indeed, propagate the activation in the whole graph to retrieve the results would be time consuming and is hardly compatible with our real-time objective. We transform the processing problem in a local one by performing spreading activation on a limited sub-graph per query. Kgram instance imports iteratively a subpart of DBpedia using `INSERT` queries and `SPARQL <service>`. Following the spreading activation logic, the graph is loaded iteratively regarding the nodes activation values. At the beginning the activation source node neighborhood (filtered by CPD) is loaded and a first round of propagation is performed. In next activations the top activated nodes neighborhoods are loaded into the Kgram instance till a loading limit is reached (discussed further). This *local* import frees the SPARQL endpoint for other queries and offers an advantage for scalability. Moreover, Kgram offers various caching systems which increase performances and balance the RDF `INSERT` cost. This architecture also offers flexibility e.g. easily query *external* public SPARQL endpoints.

4.2 Dataset

Like the majority of linked-data based exploratory search systems we decided to make a first implementation on top of DBpedia. Our motivations are somewhat similar to others. This dataset allows performing cross-domain and cross-type processing. It also offers an interesting ground for user experimentation as it contains common-knowledge items such as films or music artists. For comparative purpose we used the DBpedia 3.7 version (see part 5). As we needed to query the endpoint millions times during analysis we set up a local version. Our version contains the *wikiPageWikilink*[15] triples. This relation indicates that a hypertext link exists in Wikipedia between the 2 resources, often in the core of articles, but that the semantics of the relation was not captured. It provides valuable extra-links for connectionist methods like spreading activation. As said before the main difficulty for spreading activation over LOD source is due to the graph complexity. DBpedia 3.7 graph with *wikiPageWikiLink* triples is large (3.64 million nodes, 270 million triples) and heterogeneous since there is 319 classes in the DBpedia 3.7 ontology schema.

4.3 Settings

We set up some variables in order to implement our formula:

- The propagation spreads in *both directions* i.e. in and out links. As reverse properties are used in RDF, it is preferable to take into account incoming and outcoming neighbors to avoid knowledge loss.
- The $threshold$ filtering the propagation domain is set to a low value of 0.01. We consider that such value for a discovery objective minimizes knowledge loss. Nonetheless it filters anecdotic types present in CPD of highly connected nodes.
- *The maximum number of iterations* has still to be fixed and is discussed further.

We make use of *dcterms:subject*[16] property to compute $commontriple(i, o)$. In DBpedia instances are linked to their category thanks to the $dcterms: subject$ property. The categories constitute a topic taxonomy which is highly informative on resources nature. Thus it constitutes an interesting basis for $commontriple(i, o)$ as it aims to increase activation value of nodes which are similar to activation origin o. Moreover it is adapted to our cross-item and cross-domain processing objectives as lots of categories gather instances of several types.

4.4 Using approximation strategies to control and limit response time

At this point two parameters still need to be discussed: the *maximum number of iterations* and *the limit number of triples processed by query*. We set them thanks to algorithm behavior analyses on large sets of queries.

[15] http://dbpedia.org/ontology/wikiPageWikiLink
[16] http://purl.org/dc/terms/subject

4.4.1 Analysis method

To reduce the computational cost of the algorithm behavior study we ran it on a representative DBpedia sample. According to [10] the best sampling method to preserve large graph properties is a random walker sample of minimum 15% of the original graph. We followed these recommendations and computed a 546.000 nodes sample: 15% of 3.64 millions. The code and the sample[17] are publicly accessible.
To compare the results lists we obtained with various configurations we used the Kendall's tau-b coefficient τ_B. τ_B is a rank correlation measure reflecting the concordance of two ranked listed in terms of composition and order where:

$$\tau_B = \frac{\sum_{i<j}(sgn(x_i - x_j)sgn(y_i - y_j}{\sqrt{(T_0 - T_1)(T_0 - T_2)}},$$

$$\text{where } T_0 = \frac{n(n-1)}{2}, T_1 = \sum_k \frac{t_k(t_k-1)}{2}, \text{and } T_2 = \sum_l \frac{u_l(u_l-l)}{2}$$

and the t_k is the number of tied x value in the kth group of tied x values, u_l is the number of tied y values in the lth group of tied y values, n is the number of observation and $sgn(z)$ equals to 1 if z is positive, null if z is equal to 0 and equals to -1 if z is negative. τ_B is comprise between -1 and 1: -1 means a total discordance and 1 a total concordance.

Our configuration for tests was:
- Application server: 8 proc Intel Xeon CPU E5540 @2.53GHz 48 Go RAM
- SPARQL endpoint: 2 cores Intel Xeon CPU X7550 @2.00GHz 16Go RAM

4.4.2 Setting the maximum number of iterations

As spreading activation is an iterative algorithm we have to set a maximum number of iterations. To determine the best settings in our context we observed the RTSA convergence on DBpedia data. We performed two analyses on the 546.000 nodes of the sample. First, we measured the τ_B correlation coefficient between top 100 results at iteration *n* (*n* comprised between 1 and 100) and top 100 results at iteration 100. Second, we performed the same comparison between *n* and *n+1* top results. The triples loaded limit has not been studied yet and is experimentally fixed to 10000.

We observe on figure 1 that the top 100 results list is slowly converging. Figure 2 and table 2 shows that the results change very slowly after few iterations. In others words it becomes very expensive to continue the process after 6 iterations regarding the very slow evolution of results. Thus we decided to fix the maximum pulse at 6. Following studies were performed with 6 as maximum iterations number. A propagation visualization video using Web Import plugin for Gephi[18] has been published[19]. The fast convergence is observable on this video.

[17] http://semreco.inria.fr/hub/tools
[18] http://wiki.gephi.org/index.php/SemanticWebImport
[19] http://semreco.inria.fr/hub/videos/

Figure 2: τ_B of iteration n vs iteration 100 and iteration n vs iteration n+1 for top 100 results

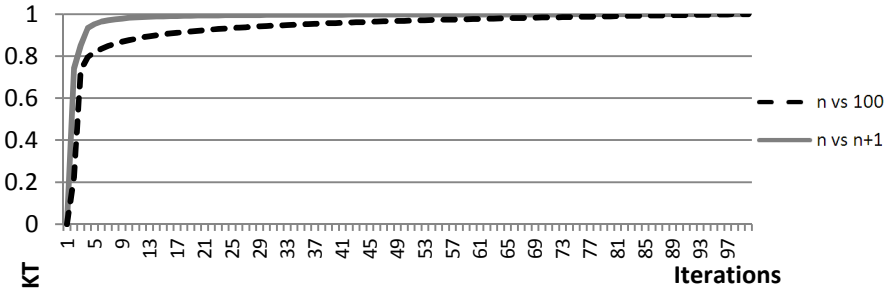


Table 2. Variation of τ_B between iteration n and n+1

Iteration	1	2	3	4	5	6	7	8
τ_B	0.74	0.11	0.08	0.02	0.01	0.005	0.005	0.002

4.4.3 Setting the triples loading limit

To control the size of the sub-graph processed we introduce a loading limit. When the imported graph overtakes the triples limit no more nodes neighborhoods are loaded anymore. We also introduce an experimental loading threshold of 0.1 i.e. nodes having an activation value under 0.1 are not eligible for loading process. This threshold allows distributing the loading process among iterations and thus *drives* the propagation more efficiently. It allows reaching more efficiently distant node. In order to define the importation loading limit we studied the query cost and the variation of results regarding several loading limits. We used the sample again and performed queries from 2000 to 20000 loading limit with a step of 2000 triples.

The figure 3 shows that the algorithm response time is linear regarding the triples loading limit. It also shows the top 100 results Kendall-Tau variation from a loading limit to another, 2000 by 2000. It is clearly observable that after 5000 the convergence starts to be very slow. The figure 4 shows response times for the 6000 loading limit and confirms that the loading limit well ensures its role as response time limiter. A large proportion of queries are processed in less than 5 seconds.

Figure 3: response time and KT regarding loading limit

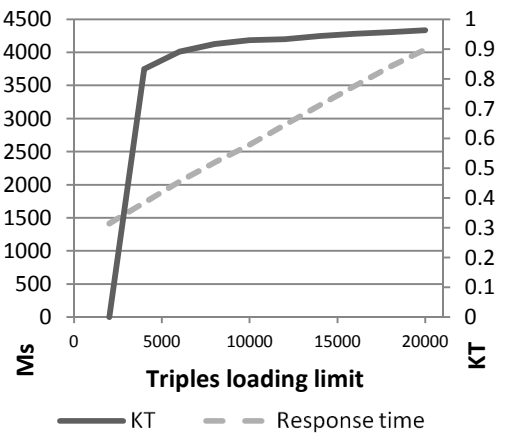
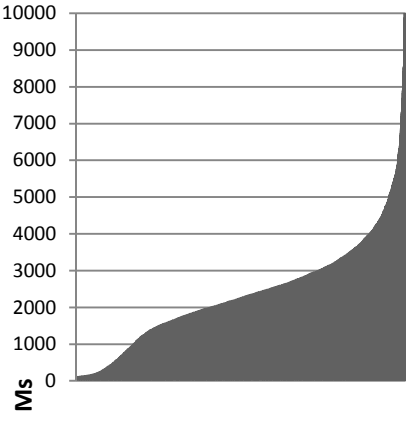


Figure 4: response time histogram

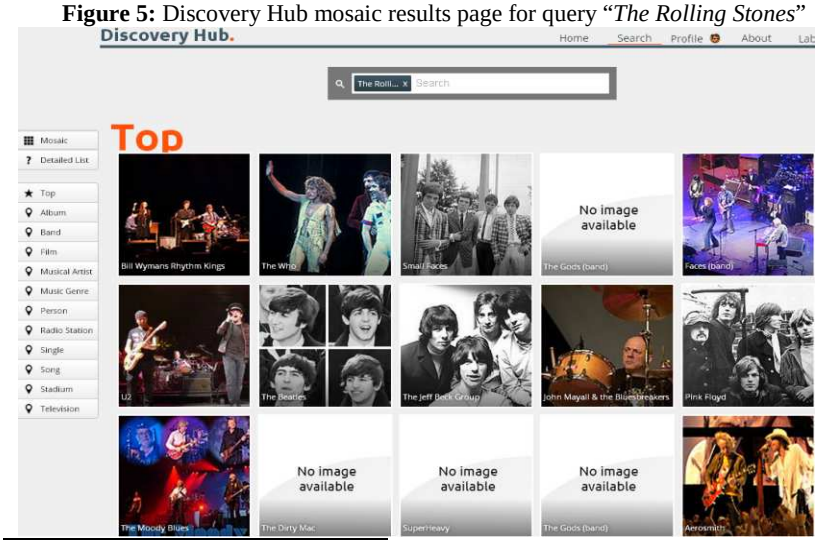


4.4 Discovery Hub: an operational prototype

Discovery hub[20] is based on the RTSA algorithm. It is an exploratory search engine which helps user to discover things he might like or might be interested in. It aims to widen his knowledge and cultural horizons. As a hub, it proposes redirections to tierce platforms to makes user benefits from his discoveries. Third-party services are proposed accordingly to the type of the considered resource e.g. music service for a *Band* or tourism platform for a *Museum*. Several demo videos are available online[21].

Users can start with an entity search thanks to a DBpedia lookup or use an import of third-party applications declared interests e.g. Facebook likes. In this last case an entity recognition is performed using *rdfs:label* properties and the DBpedia lookup. Then results exploration is facilitated thanks to various facets and filters. Classes encountered during the propagation are used to build facets (e.g. *Band*, *Film*, visible on the left of figure 5). Discovery Hub also proposes a "*top*" un-faceted results list. Classes are selected in decreasing $CDP(o)$ count order. Sets of filters are proposed using DBpedia categories e.g. *The Rolling Stones* results *Film* facet results proposes "*american rock music film*" and "*films directed by Martin Scorsese*" filters. Filters have a cumulative effect. To foster discovery we put in evidence categories (i.e. filters) with a low degree by presenting them with clearer colors. It aims to drive user in unexpected browsing paths and thus augments the discovery potential of the application. When a user is interested or intrigued by an item he can ask for 3 different explanations thanks to three features (see video[18]). First, a feature showing the seed and results common properties. Second a feature identifying crossed references in Wikipedia pages between the two resources if they exist. Third, a feature showing the relations between the result and the query-resource in a graph format. When the user goes over a node its abstract appears on the left. It is possible to get more information from the graph with the "*see links in Wikipedia*" functionality highlighting graph neighbors in its Wikipedia page.

Figure 5: Discovery Hub mosaic results page for query "*The Rolling Stones*"



[20] http://semreco.inria.fr

[21] http://semreco.inria.fr/hub/videos/

5 Evaluation

As mentioned in [19] research initiatives in exploratory search domain suffer from the lack of evaluation standardization. First we evaluated the information retrieval quality of our results. Second we judged separately the interface benefits and more specifically the explanatory features.

5.1 Information retrieval evaluation

We evaluated our results against the sVSM algorithm. The main reason is that it is the only linked-data based system of our list that has been compared to another: Seevl [6]. As MORE is a film recommender only the RTSA *Film* facet was taken in account. We compared the 2 systems on the same dataset: DBpedia 3.7. We evaluated both the relevance and the discovery potential of the RTSA algorithm regarding the sVSM baseline. We wanted to verify following hypothesizes:

- **Hypothesis 1**: the RTSA algorithm gives results at least as relevant as sVSM even if it is a real-time, multi-type and domain unspecific algorithm.
- **Hypothesis 2**: RTSA algorithm has a less strong degradation than sVSM algorithm. In others words, end-list results are better for RTSA.
- **Hypothesis 3**: Both algorithms results at the end of lists are less evident. In others words there is a greater chance that results are less relevant but newer to users and thus suppose a better discovery potential.

This evaluation was executed with an open-source survey tool[22]. Users had to judge 5 films-recommendations lists. These lists were composed with the top 20 results from the 2 algorithms and were fully randomized. The seed films used to generate the recommendation lists were randomly chosen in the « *50 films to see before you die* » list. It was chosen because it is very diverse in term of cinematographic experience: "*each film was chosen as a paragon of a particular genre or style*" (Wikipedia page). Selected films were: *2001: a space odyssey*, *Erin Brockovich*, *Terminator 2: judgment day*, *Princess Mononoke* and *Fight club*. Two Likert scale [14] questions were asked:

- *With the film [recommendation] I think I will live a similar cinematic experience as with [seed film]? Strongly agree, agree, disagree, strongly disagree.*
- *You and [recommendation]? Seen, Known but not seen, Not known*

To analyze relevance and discovery potential significance a 2(RTSA vs sSVM) * 5(Film 1 vs Film 2 vs Film 3 vs Film 4 vs Film 5) * 2(1-10 ranks vs 11-20 ranks) analysis of variance (ANOVA) test was realized. In statistics, an ANOVA [17] is a method used to compare more than two means simultaneously and determine if their differences are substantial or due to natural sampling fluctuation. As we study several factors at the same time and users participate in all conditions we performed a factorial ANOVA with repeated measure. About the sample characteristics the survey was filled by 15 persons (i.e. 3750 votes): 13 males, 2 females, average age of 31.7 years, mainly computer scientists. The average number of movies seen on any support

[22] www.limesurvey.org

monthly was 10.4 (min 4, max 30). In following results 0 corresponds to *strongly disagree*, 1 to *disagree*, 2 to *agree*, 3 to *strongly agree* for relevance score. 0 corresponds to *seen*, 1 to *known but not seen*, 2 to *not known* for discovery score.

Hypothesis 1. To verify the hypothesis 1, we observed the difference between RTSA and sSVM recommendation relevance scores. The figure 6 shows that overall RTSA (mean m = 1.42, standard deviation sd = 0.27) outperforms sVSM (m = 1.18, sd = 0.24). The ANOVA test being statistically significant ($F(1,14) = 113.85$, $p < .001$) the hypothesis 1 is verified. **Hypothesis 2.** To verify the hypothesis 2, we observed the difference between the RTSA and sSVM results relevance at the end of the list (rank 11-20). The table 3 presents the average scores about relevance and discovery for the beginning and the end of results lists. RTSA has a better relevance score (m = 1.28, sd = 0.243) than sSVM (m = 0.93, sd = 0.228) for results at the end of the list. The ANOVA test being statistically significant ($F(1,14) = 20.23$, $p = .001$) the hypothesis 2 is validated. **Hypothesis 3.** To validate the hypothesis 3 we compared both the relevance and discovery scores of the 2 two algorithms for beginning and end of results. Results are less relevant in the second half of the list (beginning m = 1.48, sd = 0.299, end m = 1.10, sd = 0.235) but have a higher discovery score (beginning m = 1.12, sd = 0.249 vs end m = 1.355, sd = 0.216). The ANOVA test being statistically significant for relevancy ($F(1,14) = 134.02$, $p < .001$) and discovery ($F(1,14) = 64.30$, $p < .001$), thus hypothesis 3 is validated. The discovery score between the beginning and the end of results lists difference is slight for RTSA. One explanation is that the algorithm computes a relatedness which is not reflecting any kind of popularity. It is noticeable that sSVM offers more discoveries than RTSA at the end of the list but its relevance decreases considerably. RTSA can be considered as more balanced.

Figure 6: relevance scores

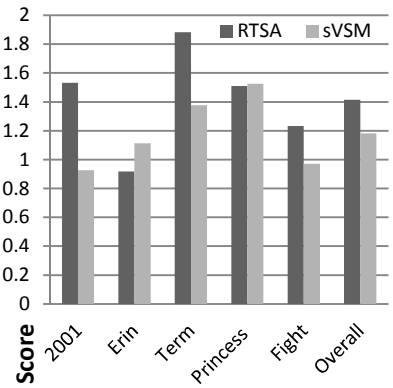


Table 3: scores on partial lists

Measure	Algorithm	Rank	Mean	St. Dev.
Relevance	RTSA	1-10	1.54	0.305
		11-20	1.28	0.243
	sVSM	1-10	1.42	0.294
		11-20	0.93	0.228
Discovery	RTSA	1-10	1.10	0.247
		11-20	1.21	0.228
	sVSM	1-10	1.14	0.251
		11-20	1.50	0.205

5.2 Interface relevance evaluation

We evaluated the benefit of our interface's explanatory features which we assume to be crucial for exploratory search tasks. **Hypothesis 4**: exploratory features increase user's overall judgments positivity.

To analyze the impact of the Discovery Hub interface on user relevance scores, the participants of the first experimentation were asked to evaluate again 20 random

RTSA results with the Discovery Hub interface. We excluded of this random selection films declared as "*seen*" during the first experimentation as we wanted to judge the attractiveness of potential discoveries only. We wanted to estimate if the explanatory features are efficient to increase user interest and discovery potential.

Hypothesis 4. To verify the hypothesis 4, we observed the difference between the relevance scores obtained with and without the explanatory features. For the randomly chosen films, the score significantly rose with the features (previously m = 1.26, sd = 0.40, with the features: m = 1.50, sd = 0.26). Average number of positive judgments reached 9.4 against 7.34 previously. A Student test was realized. It is used instead of ANOVA in the case of only two mean are compared [17]. The Student test being statistically significant ($t(14) = 3.872$), $p = 0.002$) hypothesis 4 is verified.

Then we asked participants to give their opinion about the three explanatory features. For the 3 features we asked "*the feature X helped me understand the relation between the films and to make a choice?"* and one more general question *"overall, I feel that these three features can help me to make new discoveries*". 0 corresponds to *strongly disagree*, 1 to *disagree*, 2 to *agree* and 3 to *strongly agree*. Common properties and graph-based features helped significantly the participants (average scores: 2.13 for both) whereas the benefit of the Wikipedia-based feature was less evident (average score: 1.86). The more general question received the very high average score of 2.53. Finally we asked participants to rank the 3 functionalities regarding their efficiency in terms of results explanation efficiency. The common property feature was perceived as the most efficient (ranked first: 60%, second: 13%, third: 27%). The Wikipedia-based (13%, 20%, 67%) feature was less appreciated; it is not a surprise as it does not retrieve results if there are no crossed references in resources Wikipedia pages. Thus it is not able to propose an explanation for all results. The graph-based (27%, 67%, 6%) received a large approbation by the quasi-totality of participants. Nevertheless, the results are not uniforms and confirm the necessity to propose various explanatory strategies.

6 Conclusion

In this paper, we have presented the RTSA algorithm and its implementation in the Discovery Hub application. This application allows processing linked data in real-time for exploratory search purpose. Our approach utilizes a semantic sensitive spreading activation algorithm combined with fine approximation strategies. We have shown that many requests were processed in only few seconds. Our evaluations showed the algorithm efficiency in terms of information retrieval against the sSVM baseline and also confirmed the efficiency of explanatory features deployed. Discovery Hub will soon benefit from the *live.dbpedia.org* updates. The application will take account of the latest Wikipedia changes, valuing its real-time capacities.

We are planning to extend this work in several directions. First, we want to study the behavior of the algorithm on graphs having diverse properties in order to determine to what extend the approximation strategies we proposed are generic and can be reused for others data sources. Then, we want to study the behavior and relevance of poly-centric queries which take several resources as inputs. These polycentric queries have a high potential of discovery as it can unveil unattended and non-trivial relations

notably between seeds resources of different types and/or belonging to different domains. Another research direction offered by our architecture is to query several SPARQL endpoints simultaneously to retrieve richer results. We will also evaluate the full Discovery Hub system with a qualitative evaluation on a large set of users.

Acknowledgements

We thank Julien Cojan (endpoint), Olivier Corby (SPARQL, Kgram), Alain Giboin (evaluations) and Damien Legrand (front-end, web-design) for their precious help.

References

[1] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, Z.Ives, 2007. DBpedia: A nucleus for a web of open data, *in: Proceedings of the 6th International Semantic Web*

[2] Cohen, P and Kjeldsen, R. Information Retrieval by Constrained Spreading Activation on Semantic Networks. *Information Processing and Management, 23(4):255-268, 1987.*

[3] A. Collins and E. Loftus. A spreading activation theory of semantic processing. *Psychological Review,82:407-428, 1975.*

[4] O. Corby and C. Faron-Zucker. The KGRAM abstract machine for knowledge graph querying. *In Web Intelligence, pages 338–341, 2010.*

[5] Crestani, F. Application of Spreading Activation Techniques in Information Retrieval. *Artificial Intelligence Review, 11(6): 453-482, 1997.*

[6] T. Di Noia, R. Mirizzi, V. C. Ostuni, D. Romito, and M. Zanker. Linked open data to support content-based recommender systems. *In 8th I-SEMANTICS 2012. ACM Press*

[7] HS. Haltakov, A. Kiryakov, D. Ognyanoff, R .Velkov - 2010 - D2. 4.2 Approximate Activation Spreading - *larkc.eu*

[8] Kaminskas M. and al, Knowledge-based Music Retrieval for Places of Interest, in Proceedings of *MIRUM'12, November 2, 2012.*

[9] Kendall, Maurice, Rank Correlation Methods. London: *Charles Griffin and Co*., 1948.

[10] J. Leskovec and C. Faloutsos. Sampling from large graphs. In KDD '06: *Proceedings of the 12th ACM SIGKDD conference on Knowledge discovery and data mining, ACM Press.*

[11] Likert, R. (1931). A technique for the measurement of attitudes. *Archives of Psychology.*

[12] G. Marchionini. 2006. Exploratory search: From finding to understanding. *Comm. ACM.*

[13] A. Musetti, A. G. Nuzzolese, F. Draicchio, V. Presutti, E. Blomqvist, A. Gangemi, and P. Ciancarini. 2012. Aemoo: Exploratory search based on knowledge patterns over the semantic web. *Semantic Web Challenge.*

[14] A. Passant,. 2010. dbrec – music recommendations using dbpedia. *ISWC 2010*

[15] Rocha, C., Schwabe, D., and de Aragão, M. P.: A Hybrid Approach for Searching in the Semantic Web. In *Proc. of the 13th International World Wide Web Conference.*

[16] Scheir, P., Ghidini, C., Lindstaedt, S.N.: Improving search on the semantic desktop using associative retrieval techniques. *In: Proc. of the I-Semantics 2007, pp. 415– 422 (2007)*

[17] Sprinthall, R.1997. Basic Statistical Analysis. *Prentice-Hall, New Jersey.*

[18] Tran, T., & Mika, P. 2012. Semantic Search-Systems, Concepts, Methods and the Communities behind It.

[19] Waitelonis, J., Sack, H.: Augmenting Video Search with Linked Open Data. In: *Proc. of Int. Conf. on Semantic Systems 2009 (i-semantics2009). Graz, Austria (Sep 2009)*