# Composite interests' exploration thanks to on-the-fly linked data spreading activation

Nicolas Marie, Olivier Corby, Fabien Gandon, Ribière Myriam

# Composite interests' exploration thanks to on-the-fly linked data spreading activation

Nicolas Marie
INRIA Sophia-Antipolis
Alcatel-Lucent Bell Labs
France
nicolas.marie@inria.fr

Olivier Corby, Fabien Gandon
INRIA Sophia-Antipolis
2004 route des Lucioles
06902 Sophia Antipolis, France
firstname.surname@inria.fr

Myriam Ribière
Alcatel-Lucent Bell Labs
Route de Villejust
91620, Nozay, France
myriam.ribiere@alcatel-lucent.com

## ABSTRACT

Exploratory search systems are built specifically to help the user in his cognitive consuming search tasks like learning or topic investigation. Some of these systems are built on the top of linked data and use semantics to provide cognitively-optimized search experiences. Thanks to their richness and to their connected nature linked data datasets can serve as a ground for advanced exploratory search. We propose to address the case of mixed interests' exploration in the form of composite queries (several unitary interests combined) e.g. exploring results and make discoveries related to both *The Beatles* and *Ken Loach*. . The main contribution of this paper is the proposition of a novel method that processes linked-data for exploratory search purpose. It makes use of a semantic spreading activation algorithm coupled with a sampling technique. Its particularity is to not require any results preprocessing. Consequently this method offers a high level of flexibility for querying and allows, among others, the expression of composite interests' queries on remote linked data sources. This paper also details the analysis of the algorithm behavior over DBpedia and describes an implementation: the Discovery Hub application. It is an exploratory search engine that notably supports composite queries. Finally the results of a user evaluation are presented.

## Categories and Subject Descriptors

G.2.2 [**Mathematics of Computing**]: Graph Theory – *Graph algorithms*; E.1 [**Data**]: Data Structures – *Graphs and networks*

## General Terms

Algorithms, experimentation

## Keywords

Semantic web, linked data, DBpedia, spreading activation, semantic spreading activation, exploratory search system, discovery engine, composite interest query.

## 1. INTRODUCTION

In 2006, Gary Marchionini [18] stressed the distinction between lookup and exploratory search tasks. Lookup tasks refer to search tasks when the user looks for something in particular (e.g. known item search, question answering, fact checking). During lookup tasks the search keywords are well-defined and consequently the number of results is often limited and they are easy to understand.

**Exploratory search** [18] refers to expensive cognitive search tasks when the search objective is fuzzy (e.g. learning or topic investigation). In this case the users manipulate iteratively an evolving set of keywords and have to synthesize an important amount of information coming from a changing results space. According to Gary Marchionini the actual search engines are not very efficient for the exploratory search tasks due to their keyword-based search paradigm [18]. He proposed to complete the existing solutions with new approaches that are cognitively optimized for exploratory tasks.

To optimize the search experience some exploratory search systems are built on the top of knowledge bases. Some of them make use of semantic knowledge sources including the linked open data[1] datasets. Linked data-based approaches involve the use of (1) the semantic web technologies where the **semantic web** is the web augmentation by formal metadata giving to software the access to some semantic facets of information. The semantic web data are expressed according to ontologies. An ontology is a partial representation of a world's conceptualization [9] or in others words the conceptual vocabulary of a domain. The main semantic web standards include RDF [14], a graph model with XML syntax to describe resources, SPARQL language [22] allowing querying an RDF base, RDFS [3] and OWL [19] for modeling ontologies. They also use (2) **linked open data cloud** (LOD) dataset(s) where the 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[2] then published in RDF following the LOD principles[3]. As it is extracted from an encyclopedia DBpedia contains in a single graph a vast amount of highly heterogeneous resources of various types (e.g. persons, places) belonging to diverse domains (e.g. art, fashion, science, sport).

The graph structure of the LOD datasets like DBpedia offers a great potential to enable composite interests explorations on the following form "*knowing my interest for X, Y and Z what can I discover/learn which is related to all these resources?*". Using the linked data knowledge to solve such queries gives the possibility to identify complex, indirect, non-trivial paths between the combined interests. It can give a new perspective compared to the keyword-based search engines results which retrieve the web pages containing both strings. The composite queries can be used to get a fast summary of the connections between several resources (e.g. a journalist writing on the relations between


24th ACM Conference on Hypertext and Social Media
1–3 May 2013, Paris, France.


[1] http://linkeddata.org
[2] http://wikipedia.org
[3] http://www.w3.org/DesignIssues/LinkedData.html

*Winston Churchill* and *Charles De Gaulle*) or to unveil hidden connections (e.g. a fan of *Ken Loach* and *The Beatles* searching for cultural content related to both). It can also be used in some applications to be run without user intervention in order to provide content exploratory search features. In others' words, not all the inputs need to be query; they can also be context, preferences, etc.

The combined inputs are potentially heterogeneous, having diverse types and/or belonging to different domains. An illustration of the power of DBpedia to identify paths between two very different resources is the *"everything is connected"* application [28].

In this paper we propose a method allowing composite interests explorations exploiting the knowledge of the LOD. This paper is organized as follows. The section 2 presents the related works. The section 3 details our formal proposition. The section 4 describes the implementation we built on top of DBpedia. It also presents the algorithm behavior analysis' results and motivates the choices of the parameters. The Section 5 presents the hypotheses, the protocols and the results of a user's evaluations. Finally the section 6 concludes. We illustrate our proposition with a query combining *The Beatles* and *Kean Loach*. It was the first composite query entered by a user on the Discovery Hub prototype.

## 2. RELATED WORK

We present in this part the related works concerning linked data based knowledge exploration and discovery. The applications presented below belong to the broad category of semantic search systems. Semantic search can be defined as "*search approaches that broadly speaking, use semantics to improve the search experience*" [27]. These approaches are always based on an explicit semantic processing but they vary a lot in the tackled information needs, the information resources' representation, the query's representation and the results' computation. For an extensive survey about semantic search, the readers may refer to [27]. We focus on the works sharing the following properties:

- *Fuzzy information need for knowledge exploration and discovery in learning or leisure situations*. We focus on two categories of systems that are the recommenders and the exploratory search systems.
- *Graph-based inputs*: such systems take one or several resource(s) as input(s) and retrieve others related and meaningful resources. After the computation, the result-resources are rendered and constitute the output or serve to identify the final content that is retrieved.
- *Linked data based processing*: such systems use LOD graphs, mainly DBpedia, as the primary material for the processing.

Seevl[4] [21] is a band recommender helping the discovery of musical content and artists on Youtube[5] thanks to a recommendation algorithm and a faceted browsing functionality. The ranking is processed offline. MORE [8] is a DBpedia-based film recommender accessible in the form of a Facebook[6] application[7]. It uses a semantic adaptation of the vector space model called sSVM. The more features two movies share the more similar they are. They can be linked through direct properties (e.g. "*subsequentWork*") or be the subject of triples having the same property and object ("*starring*" "*Robert de Niro*"). The system accepts several films as inputs and retrieves the union of the unitary results. The ranking is also processed offline. These two firsts systems are limited to a defined domains and to defined resource types.

Several cross-domain recommendation systems are based on linked data. In [26] the authors propose to use DBpedia in order to identify new potential collaborators. The objective is to solve industrial problems in an innovative way. A set of entities is first extracted from the description of the targeted problem. Then the *hyProximity* algorithm tries to find experts coming from others domains that might suggest innovative and unexpected solutions to the initial problem. [26] takes several resources as inputs and performs a cross-domain processing. [11] proposes a recommendation method starting from a defined domain, with a defined type, which retrieves recommendations in another domain with another defined type. The authors tested it over DBpedia using the scenario of musical recommendations starting from tourists' attractions (e.g. "*Vienna State Opera*"). The recommendation process is operated offline using a weighted spreading activation algorithm. The positive evaluation of their experimentation confirmed the high potential of Linked data for cross-domain and cross-type recommendations.

Others systems take advantage of linked data semantics to support exploratory search tasks. Aemoo[8] [20] is an exploratory search system offering a filtered view on the DBpedia graph and giving Wikipedia-based explanations on the resources shown to the user. Yovisto[9] [29] is a video platform offering an exploratory search feature that proposes a ranked list of topics related to the search results. It is also noticeable that a major search player, Google[10], launched recently an exploratory search feature ("*explore your search*", "*things not strings*"[11]). This functionality takes advantage of the Google Knowledge Graph[11] semantic network. At the beginning of 2013 it is composed of 570 million objects and more than 18 billion facts. This functionality includes collaborative filtering recommendations ("*people also search for*"). The exploration is centered on one interest at a time. There is no API available at the moment; the Google Knowledge Graph is not part of the LOD.

The state-of-the-art showed the value of using linked data for knowledge discovery and exploration purpose. As it is a young research area there are many important improvements possible:

- (1) No work is focused on composite queries solving. The systems that accept several inputs retrieve a combination of the unitary results (e.g. sum).
- (2) No work addresses the data freshness issue. Indeed, the linked data datasets are evolving over the time. The continuous update of the data and its impact on the preprocessing is not addressed in the state-of-the-art.
- (3) No work is applied on public SPARQL endpoints. Indeed the preprocessing they use is specific to the knowledge base addressed and often requires a local copy to be performed.

These limitations are mainly due to the preprocessing step used by the aforementioned systems. It is strongly conditioning the type and the range of results that the applications are able to retrieve.

[4] http://seevl.net/

[5] http://youtube.com

[6] http://www.facebook.com

[7] http://apps.facebook.com/new-more/

[8] http://wit.istc.cnr.it/aemoo

[9] http://www.yovisto.com/

[10] http://www.google.com

[11] www.google.com/insidesearch/features/search/knowledge.html

# 3. PROPOSITION

## 3.1 On-the-fly linked data processing for composite interest exploration

In this paper we propose an "*on-the-fly*" processing for linked-data based exploratory search and we explore more particularly its potential in the context of composite queries. We use the term "*on-the-fly*" to stress that the method does not need any preprocessing to retrieve the results. It opens new perspectives for exploratory search:

- (1) It is a way to handle the quasi-*infinite* number of potential composite queries (resources combinations) brought by some LOD dataset(s). It is time-consuming to preprocess this ensemble. Moreover it is even more difficult if we consider the point (2) and limiting if we consider the point (3).
- (2) It ensures that the freshest data in the knowledge base were used to compute the results. Even DBpedia, a knowledge source based on an encyclopaedia, is evolving quickly in terms of ontology models[12] and instances[13]. Some topics are interesting for exploratory search and strongly evolving (e.g. *Northern Mali conflict 2012 - present*). This is a realistic use-case, for data-journalism for instance, that justifies the need for an approach that provides fresh views of the topic. It prevents from any delay or rupture between the explored linked data source and the user e.g. schema change, instances addition, new alignments available.
- (3) As it does not need any preprocessing the method we propose can be applied on public SPARQL endpoints.

The absence of preprocessing is a challenge itself due to the complexity of the LOD graphs. Indeed, it requires smart strategies to compute the results on-the-fly starting from a very large and heterogeneous graph. To sum up our requirements we need a method that retrieves relevant results for exploratory search, that is fast but that does not depends on a preprocessing step.

## 3.2 Spreading activation basis

We chose to ground our solution on a spreading activation algorithm for the following reasons. First spreading activation was designed to process semantic networks and proved its value for information and knowledge retrieval purposes. Second it can easily be tuned and integrate various constraints including, for example, semantic sensitive weights. Third it showed efficient response times on large graphs.

Spreading activation is an algorithm family having its roots in cognitive psychology. In 1968, Quillian [23] proposed to model the human memory in the form of a semantic network. Then, Collins and Loftus [5] proposed the spreading activation mechanism to simulate the human remembering process. Later it inspired a lot of algorithms in various fields, often uncorrelated with the initial purpose. It was very successful in information and knowledge retrieval. Early and important works include [4] and [7]. 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) representing the user's interest. Then this value is propagated to neighbor's node(s). The value assigned to the neighbors depends on the settings and heuristics used to reach the algorithm goal. During the next iterations the propagation continues from the newly activated nodes. This process is repeated till a stop condition is reached e.g. maximum number of nodes activated, maximum number of iterations, time limit.

Many researchers applied the spreading activation algorithm to perform information retrieval on RDF graphs. The notable works include [24] in which the authors present a hybrid search approach combining a classical search method and an ontology-based weighted spreading activation. [25] uses a spreading activation algorithm to perform information retrieval over a RDF knowledge base. The authors make use of a schema-based similarity measure. In [15] the authors propose a semantic association search system using two pre-computed weight: a specificity and a generality one.

The LOD also motivated researches on highly fast, robust and scalable algorithms processing RDF data. This was the purpose of the LarKC[14] project: an open-source and distributed semantic computing platform using, among others, spreading activation techniques. In [10] the authors achieved the activation of millions nodes in only few seconds over locally stored LOD graphs. Nevertheless, the approximation strategies proposed are not accurate enough to be used in a knowledge retrieval context. Indeed the method massively activates the nodes, does not rank them and does not exploit finely their semantics.

We propose below a spreading activation adaptation designed to explore the graph by exploiting its semantics on-the-fly.

## 3.3 Algorithm proposition

The algorithm identifies and ranks the results starting from a user interest represented by one or several nodes of the graph (e.g. *The Beatles + Ken Loach*). At the end of its execution the activation values of the nodes determine their ranks. They are presented to the user in decreasing activation value order. Prior to the algorithm description, we introduce several necessary definitions on RDF triples, (extended from [2]), and the classic graph functions we use:

**Definition 1.(**RDF triple, RDF graph). Given $U$ a set of URI, $L$ a set of plain and typing Literal and $B$ a set of blank nodes. An 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. An RDF graph is a set of RDF triples.

**Definition 2**.(RDF typing triple, RDF non-typing triple.) An RDF typing triple is a 3-tuple$(s,p,o) \in \{U \cup B\} \times \{rdf:type\} \times \{U \cup B \cup L\}$. An RDF non-typing triple is a 3-tuple$(s,p,o) \in \{U \cup B\} \times \{U \setminus rdf:type\} \times \{U \cup B \cup L\}$.

**Definition 3. (**Infered RDF triples, IRDF triples) Infered RDF triples of an RDF non-typing triple *(s, p, o)* is the set of RDF triples$\{(s,p,o)\} \cup \{(s,rdf:type,t_i), 1 < i < n\} \cup \{(o,rdf:type,c_j), 1 < j < m\}$ obtained after RDFS closure. To ensure that each node has at least one type we give by default the type rdf:resource to each node.

Let KB be the set of all the typing triples asserted and inferred in the triple store (def. 1,2,3).

**Definition 4.** (Node degree) $degree_j$ is the number of edges involving node $j$:

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

[12] http://blog.dbpedia.org/2012/08/06/

[13] http://live.dbpedia.org

[14] http://www.larkc.eu/

**Definition 5.** (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.

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

Where type $t$ is a class in the hierarchy of the RDFS schema and $S_t$ is a direct super class of $t$ in this hierarchy before any transitive closure is computed.

**Definition 6.** (Node neighborhood) $Neighbor(i)$ is the set of neighbors of node *i* in the linked data graph:

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

Here is the formula for a *monocentric* query *i.e.* for an interest captured in the form of a unique stimulated resource (e.g. *The Beatles*). The monocentric formula serves as a basis for the polycentric one used for the composite interest queries:

**Definition 7**. (Semantic Spreading Activation algorithm, monocentric query)

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

Where:

- $o$ is the origin node *i.e.* the instance of interest initially stimulated;
- $i$ is an arbitrary instance node of the graph;
- $j$ iterates over the neighbors of $i$ ;
- $n$ 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 the node $i$ at $n$. The nodes with a positive stimulation are the origin/seeds nodes *i.e.* here $s(i,n,o)$ =1 if $i = o$ and $n = 0$ and 0 otherwise;
- $a(j,n,o)$ is the activation from a neighbor node $j$ of $i$ for a propagation origin $o$ at iteration $n$;
- $\mathrm{degree}_j$ returns the degree of the node $j$ (def. 4);
- $w(i,o)$ is a semantic weighting function which takes into account the semantics of the nodes $i$ and $o$. First, it aims to identify the propagation domain: the nodes are activated or not depending on their types. Second, it encourages the activation of the nodes similar to the origin $o$ using others semantics attributes. $w(\mathrm{i,o})$ is explained in detail below.

**Definition 8**. (Semantic Spreading Activation algorithm, polycentric query)

The query is *polycentric* when several nodes of interest are stimulated at a same time. The stimulations correspond to the unitary inputs constituting the composite interest in our case (e.g. *The Beatles* and *Ken Loach*). The results of a polycentric query are the product-intersection of several monocentric propagations results (def. 7):

$$a(i,n) = \prod_{o\in O} [a(i,n,o)] / \log(degree_i)$$

Where:

- $O$ is the set of seeds nodes *i.e.* the origins of the activations;
- $a(i,n)$ is the aggregated value of the node $i$, *i.e.* the product of the activation values of $i$ for the various propagation spreading at the iteration $n$ (differentiated by their origin $o$). The product was chosen instead of the sum in order to avoid a potential disequilibrium introduced by the difference in the monocentric activations distributions. This difference is due to the graph topologies around their respective origin node. The division by $\log(degree_i)$ aims to minimize the importance of the highly connected nodes that can be present in the monocentric propagations intersections but not very informative;
- $a(i,n,o)$ is the activation value of node $i$ at iteration $n$ for a spreading activation taking its origin at $o$ as in definition 7.

The class-based propagation domain for a polycentric query noted $CPD(O)$ is the set of types through which the propagation spreads with $O$ the set of all the seeds. To be precise, the propagation spreads through instances which have at least one type present in $CPD(O)$. It aims to increase the results quality by focusing the activation distribution on a consistent subset of nodes only. At the same time it improves the performances by narrowing the amount of processed nodes. The propagation domain is identified on-the-fly before the propagation starts thanks to the seed nodes neighborhoods' types. In case of polycentric queries it takes into account the neighborhood of all the seeds in order to identify a *shared* propagation domain.

**Definition 9**. $Tmax(x)$ is the set of the deepest types *t* of a given node *x* according to their $depth(t)$ (def. 5):

$$Types(x) = \{t; (x, rdf{:}type, t) \in KB\}$$

$$Tmax(x) = \begin{cases} t \in Types(x)\ ; \\ \forall t_i\ \in\ Types(x)\ ; \\ depth(t) \geq depth(t_i) \end{cases}$$

**Definition 10.** $NT(o)$ is a multi-set counting the occurrences of the deepest types in the seed node's neighborhood (def 6.). $NT(O)$ is the union of the $NT(o)$ with $o \in O$ and is used for polycentric-queries.

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

$$NT(O) = \bigcup_{o\in O} NT(o)$$

**Definition 11.** $CPD(O)$ is the classes propagation domain, it constitutes the class-based "*semantic pattern*" used all along the propagation. A threshold function can be applied to limit the propagation domain size for performance purpose. After this last operation we obtain the classes' propagation domain $CPD(O)$ *i.e.* nodes with a type included in $CPD(O)$ will be activated during the propagation:

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

In addition to this class-based filtering we use another 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 $o$ as a subject, the more it will receive activation:

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

Where:

$$commontriple(i,o)=\{(i,p,v)\in KB;\exists(o,p,v)\in KB\}$$

# 4. IMPLEMENTATION

This part is dedicated to the implementation of the algorithm in the Discovery Hub application. It describes the general architecture plus the settings and the approximation strategies chosen after analyses. It finally presents the application.

## 4.1 Dataset

We decided to make a first implementation on top of DBpedia. First, DBpedia is cross-domain due to its encyclopedic nature and captures a very heterogeneous knowledge in a single graph. It enables cross-domain and cross-type processing and is consequently adapted to our objective of solving composite, potentially heterogeneous, interest queries. Second it can support users' experiments as it contains common-knowledge items such as films or music artists.

As we needed to query the SPARQL endpoint millions times during the benchmark we set up a local version of DBpedia. Our version contains the *wikiPageWikilink*[15] triples. The *wikiPageWikiLink* relations indicate 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 a vast amount of extra-links which can increase the relevance of the connectionist algorithms like spreading activation ones.

As previously mentioned the main difficulty to perform spreading activation over a LOD source is due to the graph complexity. Here are some characteristics of DBpedia 3.7 dataset including *wikiPageWikiLink* triples:

- Graph size: 3.64 million nodes, 270 million triples.
- Graph heterogeneity: 319 classes in the DBpedia ontology.

As the stimulation propagates it can potentially reach a very high amount of nodes. The semantic pattern $CPD$ identified by the algorithm helps to manage the graph heterogeneity. We introduce in the next section a sampling process used to apply the algorithm on a limited and selected amount of data only.

## 4.2 Architecture

The algorithm is coded in JAVA. Each time a query is processed a Kgram [6] inference engine instance is created. This *local* instance manipulates a limited sub-graph replicated from the SPARQL endpoint. Indeed, propagating the activation in the whole DBpedia graph to retrieve the results would very be time consuming and is clearly not compatible with our performance requirements. Thus we transform this processing problem in a *local* one by performing the spreading activation on a limited sub-graph per query. The Kgram instance imports a subpart of DBpedia using `INSERT` queries. The method used to identify this sub-graph is detailed below. To control and limit the response time we also introduced a triples loading limit (discussed in 4.4.3).

In the case of polycentric queries a two-arc non-oriented SPARQL path query is performed on the endpoint[16] to identify the sub-graph that will be addressed by the spreading activation algorithm. If this query fails we augment the path length. If the queries do not produce any results and the SPARQL endpoint starts to refuse them because it is too complex we search oriented paths between the seeds in both directions. This approximation is useful for the queries combining distant nodes. In our actual implementation only the *wikiPageWikiLink* properties, which are the most current, are taken into account for this step. The *wikiPageWikiLink* considerably increase the number of connections between the nodes and help to identify more paths. Moreover when two nodes are linked by a well-defined property (e.g. *http://dbpedia.org/ontology/*) the relation is often mentioned in the Wikipedia plain text. Consequently a corresponding *wikiPageWikiLink* triple exists. Thus, restraining the path queries to these properties leads to a minor knowledge loss. The path identification can also be replaced by a random walkers-based approach if the SPARQL queries give insufficient results on the endpoint.

The nodes' neighborhoods that have been found by the path query are loaded in increasing degree order till the loading limit is reached. We assume that nodes having a lower degree are more informative about the connections between the seed nodes. To maximize the chance of retrieving results the *pivot* nodes identified by the SPARQL path query are eligible for activation even if they do not have a type present in $CPD(O)$.

```
select distinct ?x ?y where {
 service <sparqlEndpoint>
  {
   select * where {
    ?a(<…wikiPageWikiLink>|
      ^<…wikiPageWikiLink>){0,X} :: $path ?b
    filter (?a=<resource1> &&?b=<resource2>)
   }
  }
 graph $path {?x ?p ?y}
 filter(?x!=<resource1> && ?x!=<resource2>)
}
```

## 4.3 Settings

In order to implement our formula and run it over DBpedia, we have to set up some variables:

- The $threshold$ filtering the propagation domain is set to a low value of 0.01. Such value minimizes loss of knowledge.
- 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 the incoming and outcoming neighbors in order to avoid any loss of knowledge. From a spreading point of view the orientation is arbitrary and depends on a modeling choice.
- We still need to set the maximum number of iterations. It is discussed in the next part.

We make use of the *dcterms:subject* properties to compute the $commontriple(i,o)$ value. In DBpedia, the instances are linked to their categories thanks to the $dcterms:subject$ property. The categories constitute a topic taxonomy which is highly informative on the resources' nature. Thus it constitutes a valuable knowledge for $commontriple(i,o)$ which aim to increase the activation values of nodes having similarities with the activation origin.

## 4.4 Using approximation strategies to control and limit the response time

Two parameters still need to be discussed: the *maximum number of iterations* and *the limit of triples processed by query*. As the polycentric queries' results are the product intersection of several monocentric queries we use the analysis we did for monocentric

[15] http://dbpedia.org/ontology/wikiPageWikiLink

[16] Kgram is able to translate the path queries in their expanded form. This is necessary for some SPARQL endpoints.

queries' behavior to set these two remaining parameters. We use here an approximation that might require further studies. Indeed for the queries with a single origin node the sampling process we use is different. In the case of monocentric query the graph is loaded iteratively along the iterations regarding the nodes' activation values, following the spreading activation logic. At the beginning the seed node's neighborhood (filtered by classes' propagation domain) is loaded and a first round of propagation is performed. During the next activations the top activated nodes' neighborhoods are loaded into the Kgram instance till the loading limit is reached. Considering the amount of nodes processed during the analysis we reasonably think that the same study with the polycentric loading process would lead to similar results.

### 4.4.1 Analysis method

To reduce the computational cost of the algorithm behavior study we ran it on a DBpedia sample. According to [16] the best sampling method to preserve a large graph's properties is a random walker. We followed this recommendation and computed a sample[17] using this method.

To compare the results lists we obtained with various configurations we notably used the Kendall's tau-b coefficient $\tau_B$ [12]. $\tau_B$ is a rank correlation measure reflecting the concordance of two ranked lists where:

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

$$\text{where } T_0 = \frac{n(n-1)}{2},$$

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

and the $t_k$ is the number of tied $x$ value in the $k$th group of tied $x$ values, $u_l$ is the number of tied $y$ values in the $l$th group of tied $y$ values, $n$ is the number of observation and $sgn(z)$:

$$sgn(z) = \begin{cases} 1 \text{ if } z > 0 \\ 0 \text{ if } z = 0 \\ -1 \text{ if } z < 0 \end{cases}$$

$\tau_B$ is comprised between -1 and 1: -1 means a total discordance and 1 a total concordance. Thanks to it we observe the similarity of the rankings from iteration to another. It allows observing the algorithm convergence. 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 algorithm convergence over DBpedia graph. We performed an analysis on 100.000 queries using the sample nodes as inputs. We counted the number of shared results and measured the Kendall-Tau correlation coefficient between the top 100 results list at iteration *n* and the top 100 results list at iteration *n+1* for the first hundred iterations. The Kendall-Tau is calculated considering the shared results in the two lists. The triple loading limit is not studied yet and is experimentally set to 10.000 for this first analysis.

For clarity purpose the figure 1 shows only the twenty first iterations, after 16 iterations the percentage of average shared results exceeds 99% and the average $\tau_B$ is superior to 0.99. We observe that the top results are quickly converging. The table 1 focuses on the $\tau_B$ variation along the iterations. It is clear that the results change very slowly after few iterations. In others words it becomes very expensive to continue the process after few iterations regarding the very slow evolution of results. We decided to fix the maximum pulse at 6. A propagation visualization video using the Semantic Web Import plug-in for Gephi[18] has been published[19]. The fast convergence is observable on the video.

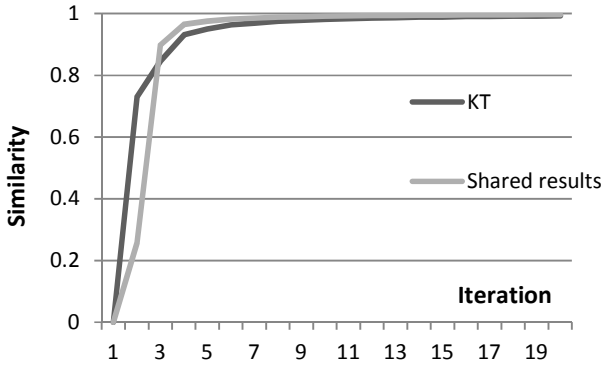


**Figure 1: Percentage of shared results and $\tau_B$ from one iteration to another, top 100 results**

**Table 1. Variation of $\tau_B$ for iterations n and n+1**

| **It. n** | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| $\tau_B$ | .73 | .11 | .08 | .018 | .013 | .006 | .006 | .002 |

### 4.4.3 Setting the triples loading limit

To control the size of the sub-graph processed and consequently the response time we introduce a maximum limit of triples imported per query. When the imported graph overtakes the triples limit no more neighborhoods are loaded anymore. We processed again 100.000 queries using the sample nodes as inputs. Each query was executed ten times with a limit ranging from 2000 to 20000 triples (step of 2000). The figure 2 shows that the algorithm response time is linear regarding the triples loading limit.

[17] The code and a 546.000 nodes sample are publicly accessible for reuse: http://semreco.inria.fr/hub/tools

[18] http://wiki.gephi.org/index.php/SemanticWebImport

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

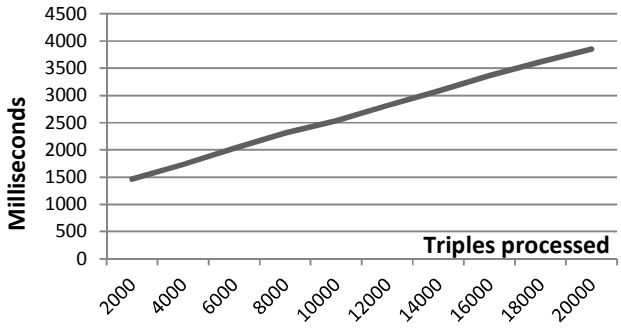


**Figure 2: response time regarding the loading limits for monocentric queries**

The figure 3 shows the top 100 results Kendall-Tau variation from a loading limit to another, 2000 by 2000. It is clearly observable that after 6000 the convergence starts to be very slow. Thus we chose 6000 as loading limit.

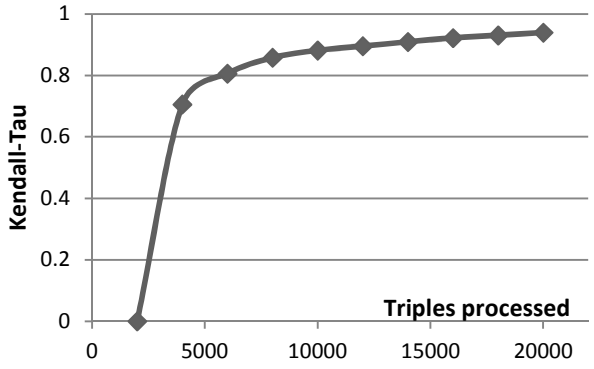


**Figure 3: $\tau_B$ from one loading limit to another (step: 2000), top 100 results**

## 4.5 Polycentric behavior analysis

Then we wanted to observe if the polycentric results were specific to the inputs combination. We processed again 100.000 nodes of our sample. For each of them we selected randomly 2 nodes in the 2 arc-max neighbourhood thanks to a random walker. We processed 3 queries: the sample's node only (monocentric) and two polycentric queries by adding each time one of the randomly selected neighbours. The figure 4 shows a histogram of the top 100 shared results percentage between the monocentric and the polycentric queries results. The figure 5 shows it for the 2 polycentric queries amongst themselves. These 2 histograms both point out that the results list similarities are very low. In others words the results are highly specific to the input(s).

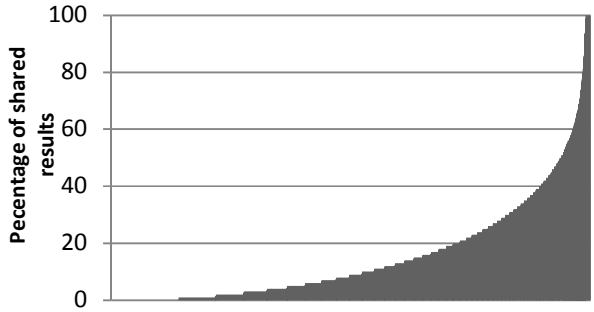


**Figure 4: Shared top 100 results histogram between the monocentric and polycentric queries results**

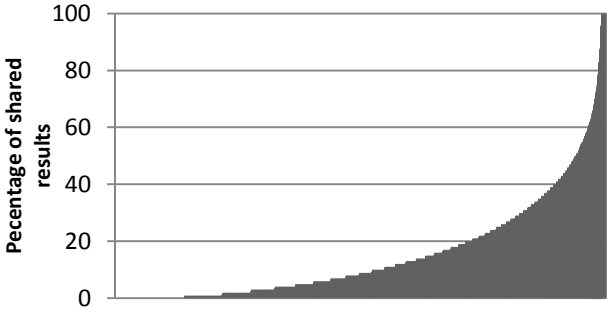


**Figure 5: Shared top 100 results histograms between the first and second polycentric queries results**

The response time histogram of polycentric queries (figure 6) shows that a majority of queries are processed in less than 10 seconds. Overall the response time of the polycentric queries is superior to the monocentric ones (figure 2) due to the path queries costs.

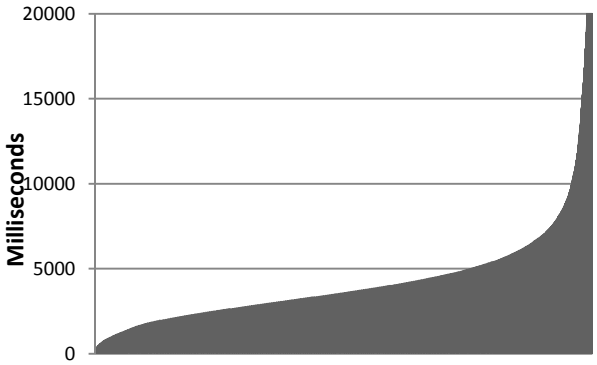


**Figure 6: polycentric queries response time histogram in ms**

## 4.6 Discovery Hub: an operational prototype

Discovery Hub[20] implements the algorithm and the sampling process previously described. It uses DBpedia as knowledge base. It is an exploratory search engine which helps the user to discover things he might like or might be interested in. It aims to widen his knowledge and cultural horizons. It proposes many redirections to tierce platforms to extend the search process. The third-party services are proposed according to the type of the considered result e.g. music service for a *Band* or tourism platform for a *Museum*. Several demonstration videos are available online[21].

To enter the queries the user can start with an entity search thanks to a DBpedia lookup[22] or use an import of third-party profile information e.g. Facebook likes. In this last case an entity recognition is performed using the *rdfs:label* properties and the DBpedia lookup. The composite queries are encouraged thanks to the "*search box*" in which the user can drag and drop items all along his navigation (figure 7). He can pick resources of interest on the homepage, the results pages or in his profile for instance. The composition is limited today to 4 resources.

The navigation in the results space is facilitated thanks to various facets and filters allowing a deep exploration. The classes in the

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

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

[22] http://wiki.dbpedia.org/lookup/

propagation domain are used as navigational items to build the facets e.g. *Band*, *Film*. 40 results at maximum are presented by facet on the actual prototype. Discovery Hub also proposes a "*top*" un-faceted results list showing the 40 most activated resources without any condition on their types.

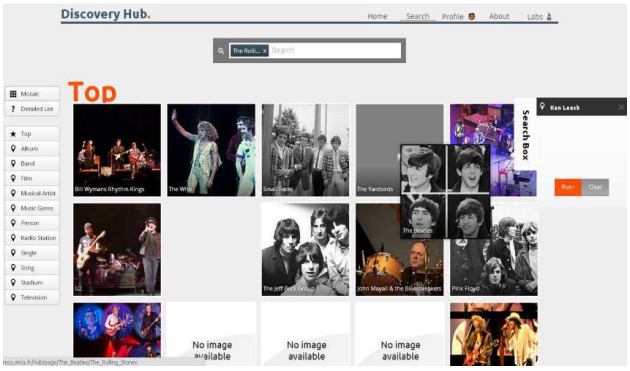


**Figure 7: The user is currently dragging *The Beatles* in the *"search box"* to complete *Ken Loach* and launch the query**

A set of filters per facet is proposed using the DBpedia categories. These filters are retrieved thanks to the query below. For instance on the figure 9 the user filtered the *Film* facet's results with "*2000s comedy-drama films*". We put in evidence the categories (*i.e.* the filters) having a low degree by presenting them with clearer colors. It aims to drive the user in unexpected browsing paths and thus augments the discovery potential of the application. The filters have a cumulative effect.

```
select ?p where {
 service <sparqlEndpoint>
 {
  select ?p (count(?x) as ?count) where {
   ?x <http://purl.org/dc/terms/subject> ?p
   filter( ?x = result1Facet1  || ?x=
    result2Facet1 || ?x = result3Facet1 … )
  } order by desc (?count)
 }
 filter(?count>1)
}
```

To give a real example of results, the composite query *The Beatles + Ken Loach* provides the following facets (or *CPD*): *Album, Band, Film, Musical Artist, Music Genre, Person, Radio Station, Single, Television show*. The *Film* facet proposes, among others, these filters: *British drama films, films associated with the Beatles, films directed by Ken Loach, films set in Liverpool*.

When a user is interested or intrigued by an item, he can ask for various explanations thanks to three dedicated features. These features are mandatory for composite heterogeneous queries when some non-trivial and unattended results are retrieved. They need to be explained to receive the user's approval. The following explanatory features are presented in a video[23]:

- A feature showing the common properties that share the results with the query-resource(s).
- A feature identifying and highlighting the crossed references between the result and the query-resources in the Wikipedia pages (figure 8).
- A feature showing the connections between the results and the query-resources in a graph format (figure 9). When the user goes over a node its abstract appears on the left. It is possible to get even more information with the "*see links in Wikipedia*" functionality that highlights the graph neighbors in the Wikipedia pages. This graph is built on demand thanks to a SPARQL path query. It is often instantaneous, require few seconds when the graph is dense.

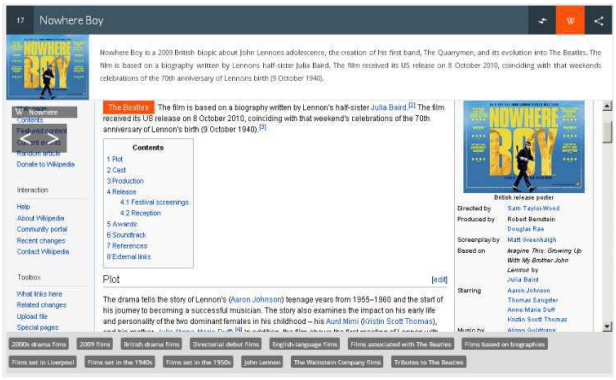


**Figure 8: Wikipedia-based explanation for "*Nowhere Boy*"**

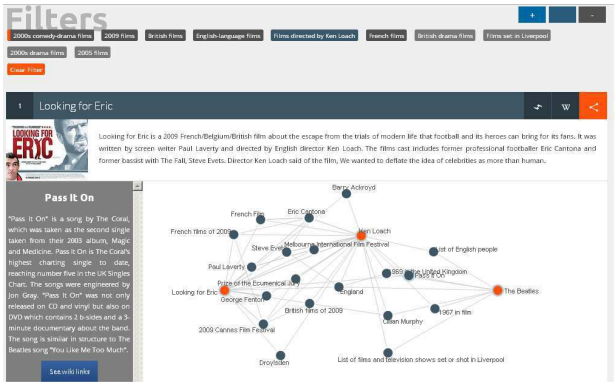


**Figure 9: Graph-based explanation for "*Looking for Eric*"**

## 5. EVALUATION

As mentioned in [13] the research initiatives in the exploratory search domain suffer from the lack of evaluation standardization. It is even more the case for our composite interest objective as there is no clear baseline available[24]. We wanted to verify the following hypothesizes:

- **Hypothesis 1:** the composite queries results interest the user.
- **Hypothesis 2:** some results are unattended; they offer a high discovery potential.
- **Hypothesis 3:** the explanatory features help the user to understand the link between the query-resources and the results; thus they support efficiently the results space understanding.

This evaluation was executed using an adaptation of the Discovery Hub interface. The users had to judge 2 results lists of the top 10 algorithm results. Each list was generated starting from 2 of their individual Facebook likes randomly combined. In this way we wanted to simulate a composite interest query that the user was susceptible to enter in the system.

The following scenario was introduced: "*you heard about a new discovery engine that can help you to learn and discover new*

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

[24] We propose an API for comparison purpose.

*things easily starting from items you like. This tool notably allows generating results starting from several interests. You decide to test it on yours. We propose you to judge 2 results lists generated from your Facebook likes*".

Two Likert scales [17] were used:

- Question 1: *The result interests me: strongly agree, agree, disagree, strongly disagree*.
- Question 2: *The result is unexpected: strongly agree, agree, disagree, strongly disagree*.

Concerning the sample characteristics, the survey was filled by 12 persons: 3 females, 9 males from various backgrounds, mainly people who asked an access to the Discovery Hub beta. In the following results 0 corresponds to *strongly disagree*, 1 to *disagree*, 2 to *agree*, 3 to *strongly agree* for relevance score and discovery scores (respectively question 1 and 2).

To verify the hypothesis 1, we observed the relevance score (question 1). The average relevance score was 1.65, with a standard deviation of 0.94. The figure 10 shows the average relevance scores per query histogram. 71% of queries received a relevance score over the mean (1.5). Thus the hypothesis 1 is verified. It is noticeable that one query received the worst score possible; all its results were rated 0. Its seeds were very distant: *Samuel L. Jackson* and *Grooveshark*.

To verify the hypothesis 2, we observed the unexpectedness score (question 2). The average unexpectedness score was 1.90 with a standard deviation of 1. The figure 11 shows the average unexpectedness scores histogram. 58.33% of queries received an average score over the mean (1.5). Thus, the hypothesis 2 is verified.

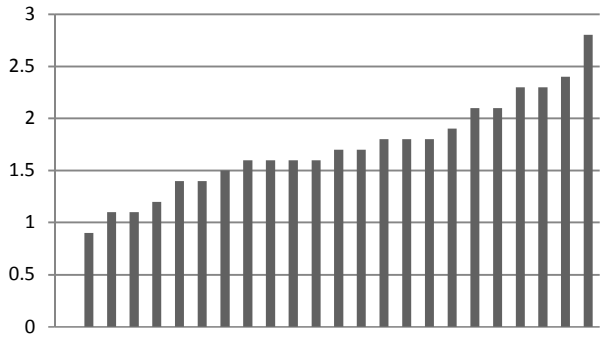


**Figure 10: average relevance score per query histogram**

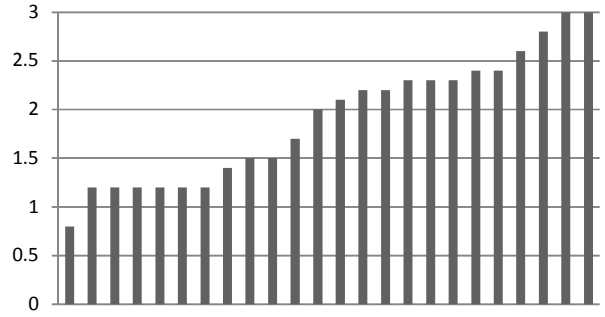


**Figure 11: unexpectedness score per query histogram**

Concerning the discovery potential it is also interesting to observe the recovery between relevance and unexpectedness:

- 61.6% of the results were rated as *strongly relevant* or *relevant* by the participants.
- 65% of the results were rated as *strongly unexpected* or *unexpected*.
- 35.42% of the results were rated both as *strongly relevant* or *relevant* and *strongly unexpected* or *unexpected*.

Then we asked the 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 result and my interests 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* (figure 12). The graph-based explanatory feature which was designed specifically to understand the non-trivial connections between several resources received a very high average score (mean m = 2.92). It is particularly adapted to results explanations for the polycentric queries results as they often unveil indirect links. The Wikipedia-based explanatory feature received an average score over the mean (m = 1.83). Participants liked the possibility to use it from the graph explanation. Finally the common property feature received an average score close the mean (m = 1.58). It is often impossible to find common properties between the results and all the different seed nodes constituting the composite interest. This feature is more efficient in the case of monocentric queries. The more general question received the high average score of 2.67 and confirms the interest of these explanatory features to increase the discovery potential of the application. The hypothesis 3 is verified.

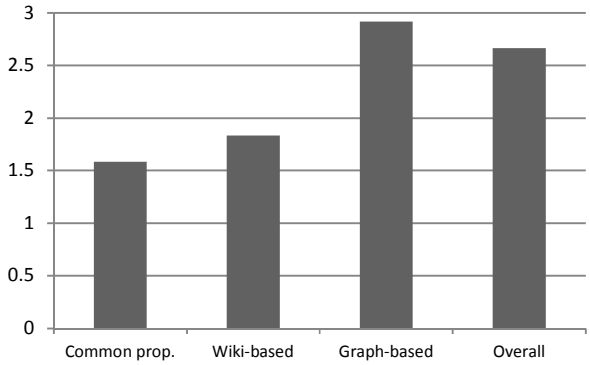


**Figure 12: users' opinions about the explanatory features**

Finally we asked the participants to rank the 3 functionalities regarding their perceived efficiency in terms of results explanations. The rankings confirmed the previous results. The common property feature was perceived as the less efficient (ranked first: 0%, second: 72.7%, third: 27.3%). The Wikipedia-based feature was more appreciated (54.5%, 27.3%, and 18.2%). Finally the graph-based received a very large approval (45.45%, 45.45%, and 9%). Nevertheless, the results are not totally uniforms and confirm the interest to propose various strategies.

# 6. CONCLUSION

In this paper, we have presented a method for processing linked data graph for exploratory search purposes. It is composed of a semantic spreading activation algorithm associated to a sampling process. Its main particularity regarding the state-of-the-art is to not need any preprocessing step to compute the results. We also detailed its implementation in the Discovery Hub application. We

presented extensive analysis of its behavior that helped us to set the main parameters of the implementation. This article was especially focused on composite interests (interests captured in the form of several unitary resources) explorations thanks to polycentric spreading activation. During our analysis a majority of the polycentric queries results were specific to the nodes combination used as inputs. Overall the queries required few seconds to be processed. The participants of the evaluation mostly rated the results as relevant or unexpected. More than one third of the results was rated as both relevant and unexpected. It is encouraging as it reflects a high discovery potential for the application. The evaluation notably showed the high efficiency of the graph-based and Wikipedia-based explanatory features for explaining the polycentric queries results.

We plan to extend this work in several directions. An interesting possibility offered by the architecture is to modify the semantics embedded in the propagation in order to support personalization and contextualization functions, using for example the property semantics. Another possible research direction is to query several SPARQL endpoints at the same times (e.g. French, Spanish, and Italian DBpedias), build one meta-lingual graph and retrieve richer results. Finally we will evaluate the full Discovery Hub system thanks to a qualitative evaluation on a large set of users. We still need to validate the usefulness of applying the algorithm in the context of exploratory search with the interactive dimension. This evaluation will also allows us to determine the quality of the navigational items we propose like the facets, filters and to identify some browsing patterns.

## 7. ACKNOWLEDGMENTS

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

## 8. 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 Conference, 2007*.

[2] Basse A., Gandon F., Mirbel I., Lo M., Incremental characterization of RDF Triple Stores, *INRIA Research Report RR-7941*, April 2012

[3] D. Brickley and R. Guha. 2004. Rdf vocabulary description language 1.0: Rdf schema, Feb. 2004.

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

[5] A. Collins and E. Loftus. A spreading activation theory of semantic processing. *Psychological Review,8,2 1975.*

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

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

[8] 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*

[9] N. Guarino and P. Giaretta. 1995. Ontologies and Knowledge Bases - Towards a Terminological Clarification, *pages 25-32. IOS Press, Amsterdam, The Netherlands, 199*

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

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

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

[13] Kules, B., & Capra, R. (2008) Creating exploratory tasks for a faceted search interface. *Paper presented at the Second Workshop on Human-Computer Interaction* (HCIR 2008).

[14] Lassila O. and Swick R. (1999). RESOURCE DESCRIPTION FRAMEWORK (RDF). W3C proposed Recommendation, *January 1999*

[15] M. Lee, W. Kim, and T. G. Wang, "An explorative association-based search for the semantic web," *in Proceedings of the 2010 IEEE Fourth International Conference on Semantic Computing, ser. ICSC '10.*

[16] J. Leskovec and C. Faloutsos. Sampling from large graphs. In KDD '06: *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining, pages 631–636, New York, NY, USA, 2006. ACM Press.*

[17] Likert, R. (1931). A technique for the measurement of attitudes. *Archives of Psychology. New York: Columbia University Press*.

[18] G. Marchionini. 2006. Exploratory search: From finding to understanding. *Comm. Of the ACM, 49(4), 2006.*

[19] D. L. McGuinness and F. van Harmelen. Owl web ontology language overview. *Technical Report REC-owl-features-20040210, W3C, 2004.*

[20] 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.*

[21] A. Passant,. 2010. dbrec – music recommendations using dbpedia. *The Semantic Web–ISWC 2010 209–224.*

[22] E. Prud'hommeaux and A. Seaborne. 2005. *SPARQL query language for RDF, 2005.*

[23] Quillian, M. (1968). Semantic memory. In M. Minsky (Ed.), *Semantic Information Processing, pp. 227–270. MIT Press, Cambridge, MA.*

[24] 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 (WWW 2004), NY (2004) 374-383*

[25] 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)*

[26] Stankovic, M., Rowe, M. and Laublet, P. Finding Co-solvers on Twitter , with a Little Help from Linked Data. *In Proceedings of Extended Semantic Web Conference ESWC 2012*, May 27-31, Heraklion, Crete, Grece Tran, T., & Mika, P. 2012. Semantic Search-Systems, Concepts, Methods and the Communities behind It.

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

[28] Miel Vander Sande and al.s, 2012. Everything is Connected: Using Linked Data for Multimedia Narration of Connections between Concepts,. In: *Proceedings of the 11th International Semantic Web Conference, 2012.*

[29] 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)*