Discovery Hub: a discovery engine on the top of DBpedia

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

Fabien Gandon, Damien Legrand
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

This paper supports the Discovery Hub demonstration proposal. 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 presents the Discovery Hub operational prototype after detailing its Real Time Spreading Activation (RTSA) algorithm. This latter processes linked data in real-time and does not require partial or total results pre-processing. This real-time processing offers advantages in terms of data dynamicity-handling and querying flexibility.

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 (figure 1). 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 search keywords are well-defined and consequently lead to a results space which is narrowed and easy to manipulate.

Exploratory search [18] refers to expensive cognitive search tasks when the search objective is fuzzy (e.g. learning or investigating). In this case users manipulate iteratively an evolving set of keywords and have to synthesize an important amount of information coming from an unstable results space.

Search can be considered as a partially solved problem. Indeed, according to Gary Marchionini actual search engines are not very efficient for exploratory search tasks due to their keyword oriented paradigm. He proposed to complete existing search solutions with systems cognitively optimized for exploratory tasks.

An essential characteristic of exploratory search systems is that user involvement in the search process is high. Their interfaces offer smart browsing, filtering and explanation strategies to reveal desired or unattended knowledge. Thus the human computer interaction aspects are as important as information retrieval ones.

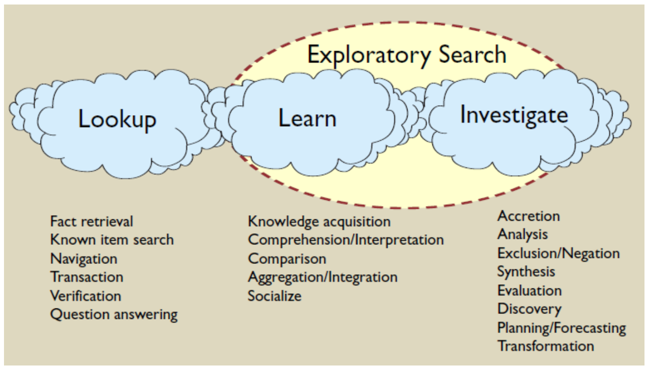


Figure 1: Distinction between lookup and exploratory search tasks, taken from [18]

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[1]. 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. An ontology is a partial representation of a world's conceptualization [10] or in others words the conceptual vocabulary of a domain. Main semantic web standards include RDF [16], a graph model with XML syntax to describe resources, SPARQL language [22] allowing querying an RDF base, RDFS [4] and OWL [19] for modeling ontologies; and (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 [2] is the most popular and used one.

[1] http://linkeddata.org

DBpedia results from the extraction of data from Wikipedia[2] then published in RDF following the LOD principles. Due to its encyclopedic provenance DBpedia contains a vast amount of highly heterogeneous resources of various types (e.g. persons, places) belonging to diverse domains (e.g. art, fashion, science, sport) in a unique graph.

In this paper we present the Discovery Hub application. This paper is organized as follows. Section 2 presents related works. Section 3 details our formal proposition. Section 4 describes the corresponding algorithm implementation on the top of DBpedia. Section 5 presents the interface of the Discovery Hub prototype.

2. LINKED-DATA BASED DISCOVERY

We present in this part related works concerning linked data based processing for knowledge exploration and discovery. 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*" [26]. 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 [26]; 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[3] videos thanks to API call).
- *Linked data processing*: such systems use LOD graphs, mainly DBpedia, as the primary material for processing.

In the past 3 years several research initiatives showed the interest to use the LOD for discovery and exploratory search objectives. Seevl[4], MORE[5] and [13] are Linked Data-based recommenders, Seevl is a DBpedia-based band recommender for Youtube. It proposes a "*related*" function offering band recommendation based on the DBrec algorithm [21]. The DBrec algorithm ranks results according to direct or indirect shared properties with the seed resource. The ranking is processed offline and stored in RDF. MORE [9] 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 [9], sSVM was evaluated as more accurate than DBrec on cinema domain. Kaminskas and al. [13] 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 the LOD for cross-domain, cross-type recommendations.

[2] http://wikipedia.org

[3] http://www.youtube.com

[4] http://seevl.net/

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

Exploratory search systems differ from recommenders by the user active role in the discovery process. Yovisto [27] is an academic video platform offering an exploratory search feature. It proposes a ranked list of related resources besides search results[6]. 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 [20] is a DBpedia-based exploratory search system which uses Encyclopedic Knowledge Patterns (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[7], launched a feature in the spirit of exploratory search ("*explore your search*", "*things not strings*"). It performs entity recognition on search keywords. Then it provides structured information using the Google Knowledge Graph[8] and recommendations based on collaborative filtering ("*people also search for*"). This knowledge graph is presented as a composition of Wikipedia, Freebase[9] and CIA World Factbook[10] composition.

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 ALGORITHM

3.1 Real-time linked data processing

We propose to process the linked data in real-time. The real-time processing brings several major advantages.

- First it allows handling the quasi-*infinite* number of possible queries brought by some LOD dataset(s). For instance, DBpedia 3.7 contains 3.64 millions instances[11]. Thus there are more than 13.249 billion composite queries possibilities with two resources as inputs. This huge amount of potential queries is hard to pre-process (partially or totally) especially if we want the processing to retrieve results that are specific to the composition of the query nodes and not just retrieve a sum of unitary results.
- Second, the settings variations based on graph information are also quasi-*infinite* e.g. reinforcing or minimizing the importance of a set of relations, discarding certain

[6] This functionality seems not to be accessible anymore.

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

[8] http://www.google.com/insidesearch/features/search/knowledge.html

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

[10] https://www.cia.gov/library/publications/the-world-factbook/

[11] http://blog.dbpedia.org/2011/09/11/

information facets for instance. The resulting fine-grained querying potential has a high value for user as it can support advanced search functions, personalization or contextualization for instance. This fine and adaptive graph exploitation is not easily reachable through results pre-processing as it can take many forms. It requires a flexible conceptual framework.

- Third, real-time processing allows handling data source dynamicity. Indeed, linked data are dynamic. Even DBpedia, an encyclopaedic based knowledge source, is evolving quickly in term of ontology models[12] and instances[13]. A real-time processing ensures that the freshest data available were taken in account to compute the results. It prevents from any delay or rupture between the explored linked data source and the user e.g. schema change, instances addition, new alignments.

The real-time processing offers a very flexible conceptual framework. Its difficulty increases with the graph complexity. It requires smart strategies to make sense from a large set of highly heterogeneous triples in only few seconds.

This challenge is even more difficult for some complex queries that can be solved thanks to LOD richness. We notably want to enable composite interests exploration on the following form "*knowing my interest for The Beatles and Kean Loach what can I discover/learn which is related to both these resources?*". We also want the results to be not limited to a specific domain (e.g. cinema) or one or several defined resource types (e.g. films and actors only).

3.2 Spreading activation basis

We chose to ground our solution on a spreading activation method for following reasons. First spreading activation is designed for semantic networks and proved many times its value for information retrieval purposes. Second it can easily be tuned and can integrate various constraints including, for example, a semantic sensitiveness measure. Third it showed efficient response times on large graphs. For all these reasons it is adapted to our objective to process linked data graph with performance constraints.

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 [6] 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 [5] and [8]. 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. 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. maximum number of nodes activated, maximum number of iterations, time limit. Following [1] we propose a generic spreading activation formula:

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

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

$$a(i,n+1) = w_s * s(i,n) + \mu f(\sum_j w_{ij} * a(j,n))$$

Where:

- i is an arbitrary node of the graph;
- n is the current number of iterations;
- $a(i,n+1)$ is the activation value of node i at $n+1$ time;
- w_s is a weight balancing the stimulation value;
- $s(i,n)$ is the stimulation value of the node i *at* $n+1$ time. Stimulation is often only applied at initial time 0;
- w_{ij} is the semantic attenuation between nodes i and neighbor j;
- $a(j,n)$ is the incoming activation from node j at time n;
- μ is a weight balancing the activation value;
- The function f, applied to the incoming activation, represents constraints which depend on algorithm goal (e.g. threshold).

Many researchers used spreading activation to perform information retrieval on RDF graphs which are typical semantic networks. Notable works using advanced semantic processing include [24] in which authors present a hybrid search approach combining a classical search method and an ontology-based weighted spreading activation. [25] uses spreading activation to perform information retrieval over a knowledge base. Authors notably make use of a schema-based similarity measure. In [17] 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 is the purpose of the LarKC[14] project which developed an open-source and distributed semantic computing platform using among others spreading activation techniques. In [11] authors propose very fast spreading activation over LOD. They achieved the activation of millions nodes in only few seconds. Nevertheless, used approximation strategies are not accurate enough to be used in a knowledge retrieval context as they massively select nodes, do not rank them and do not exploit their semantics finely.

To achieve the 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 a-priori settings or processing and exploits the graph semantics on real-time.

3.3 RTSA formalization

Prior to the algorithm description, we introduce necessary definitions on RDF triples, (extended from [3]), 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.(RDF typed triple, RDF untyped triple.) A RDF typed triple is a 3-tuple $(s,p,o) \in \{U \cup B\} \times \{rdf{:}type\} \times \{U \cup B \cup L\}$. A RDF untyped triple is a 3-tuple $(s, p, o) \in \{U \cup B\} \times \{U \ rdf{:}type\} \times \{U \cup B \cup L\}$.

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

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

Definition 4. (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 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.

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

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

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

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

Definition 7. (Real Time Spreading Activation algorithm, monocentric query)

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

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/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 the neighbor node j of i for a propagation origin o at iteration n;
- 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 nodes i and o. First, it aims to identify the propagation domain i.e. nodes are activated depending on their types. Second, it encourages the activation of nodes similar to origin o using others semantics attributes. $w(\mathrm{i},\mathrm{o})$ is explained in detail below.

Definition 8. (Real Time Spreading Activation algorithm, polycentric query)

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

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

Where:

- O is the set of seeds i.e. the origin nodes of the activations;
- $a(i,n)$ is the aggregated value of node i, i.e. the product of activation value of i for the various propagations 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 differences in monocentric activations distributions due to the local graph topology of their respective origin node. The division by $\log(degree_i)$ aims to minimize the importance of highly connected nodes that can be very present in polycentric query results 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.

Let KB be the set of all typed triples asserted and inferred in the triple store (def. 1,2,3). The class-based identification of the propagation domain is noted CPD. It is the set of types through which the propagation spreads with O the set of all the seeds (only one for monocentric queries). 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 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 nodes neighborhoods types. For polycentric queries it takes into account the neighborhood of all seeds to identify a *shared* propagation domain.

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

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

$$Tmax(x) = \left\{ \begin{array}{c} t \in Types(x)\,; \\ \forall t_i \in Types(x)\,; \\ depth(t) \geq depth(t_i) \end{array} \right\}$$

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

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

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

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 o as a subject, the more it will receive activation:

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

Where:

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

4. IMPLEMENTATION ON DBPEDIA

This part is dedicated to the implementation of the RTSA algorithm on the top of DBpedia.

4.1 Architecture

The figure 2 shows an overview of the Discovery Hub architecture. Its interface presented in section 5. The algorithm is coded in JAVA. Each time a query is processed a Kgram [7] inference engine instance is created. This *local* instance manipulates a limited sub-graph replicated from the SPARQL endpoint. Indeed, propagate the activation in the whole DBpedia 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. The Kgram instance imports a subpart of DBpedia using `INSERT` queries and `SPARQL <service>`. The method used to identify this sub-graph is detailed below. To control and limit the response time we introduced a triples loading limit. In case of polycentric queries the monocentric propagations are multi-threaded.

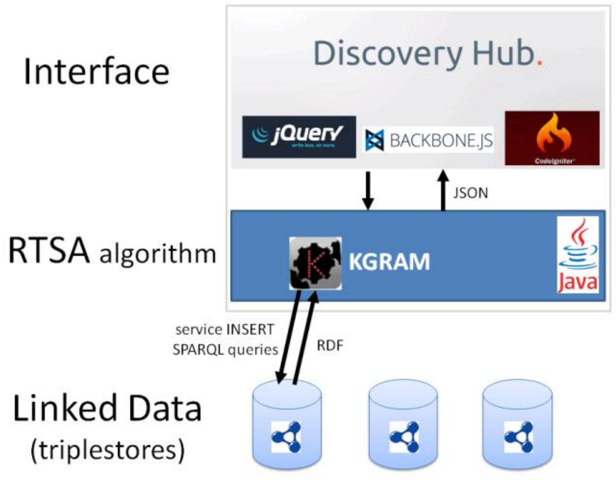


Figure 2: General architecture of the Discovery Hub application

In the case of mono-centric queries, 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 the loading limit is reached. 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 the case of polycentric queries a two-arcs unoriented SPARQL path query is performed on the endpoint to identify the sub-graph that will be addressed by spreading activation. For performance purpose only *wikiPageWikiLink* relations, which are the most current, are taken into account. If this query fails we perform a 3-arcs (or more) oriented path between seeds in both directions. Nodes neighborhood that have been found by the path query are loaded in decreasing degree order till the loading limit is reached. We assume that a lower degree brings more specific information about the seed nodes. To maximize the chance of retrieving results *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>)
}
```

This import in a *local* instance has several advantages. First, it frees the SPARQL endpoint for other queries and offers an advantage for scalability. Second, Kgram offers various caching systems which increase performances and balance the RDF `INSERT` cost. Third, this architecture offers flexibility. It is possible to easily query *external* SPARQL endpoints to add information for instance.

4.2 Dataset

We decided to make a first implementation on top of DBpedia. First, DBpedia is cross-domain due to its encyclopedic nature and captures very highly heterogeneous information in a same graph. It notably allows performing cross-domain and cross-type processing and is consequently adapted to our objective of solving composite, potentially heterogeneous, interest queries. Second it offers an interesting ground for user experimentation as it contains common-knowledge items such as films or music artists.

As we needed to query the endpoint millions times during analysis we set up a local version. Our version contains the *wikiPageWikilink*[15] triples. The *wikiPageWikiLink* 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 extra-links which are interesting for connectionist methods like spreading activation.

As said before the main difficulty for spreading activation over 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.

4.3 Settings

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

- Extensive analyses[16] we made demonstrated that we obtain a good approximation and satisfying response time with a

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

[16] Papers currently being reviewed.

maximum number of iterations of 6 and a triples loading limit of 5000. According to our analyses mono-centric and poly-centric queries are processed in only few seconds with these parameters[14].

- 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 allows filtering anecdotic types present in propagation domain of highly connected nodes.
- 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.

We make use of *dcterms:subject* property to compute $commontriple(i,o)$. In DBpedia, instances are linked to their categories 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 $commontriple(i,o)$ which aim to increase activation value of nodes which are similar to the activation origin o. Moreover it is adapted to our cross-item and cross-domain processing requirements as lots of categories gather instances of several types and domains.

5. DISCOVERY HUB AN OPERATIONAL PROTOTYPE

Discovery Hub[17] is based on the RTSA algorithm and uses DBpedia as knowledge source. 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 third party services to makes user benefits from his discoveries. Several demo videos are available online[18]. The figure 3 shows its homepage.

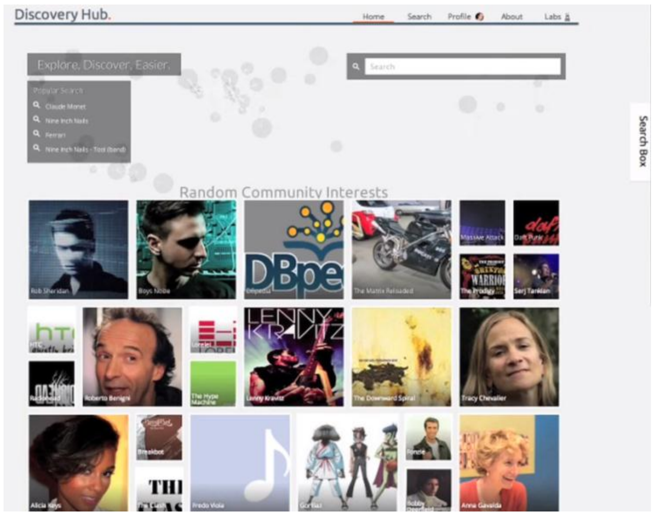


Figure 3: Discovery Hub homepage

5.1 Querying possibilities

In order to avoid a cold start problem during the first connections we put in evidence on the homepage a selection of randomly chosen users' interests (figure 3). The new users can quickly test the system using these resources as starting points for exploration.

It is possible to perform queries thanks to an entity search powered by a DBpedia lookup[19]. The users can also import their tierce applications interests e.g. Facebook likes (see figure 4) or Twitter[20] followings. In this last case an entity recognition is performed using the *rdfs:label* properties. The imported interests appear in the users profiles (see figure 5) with the ones internal to Discovery Hub which also propose a *like* system.

Composite queries are encouraged thanks to the *Search Box* in which users can drag and drop items all along their navigation (figure 6). The users can pick resources of interest on the homepage, the results pages or profile pages for instance. The composition is limited today to 4 resources.

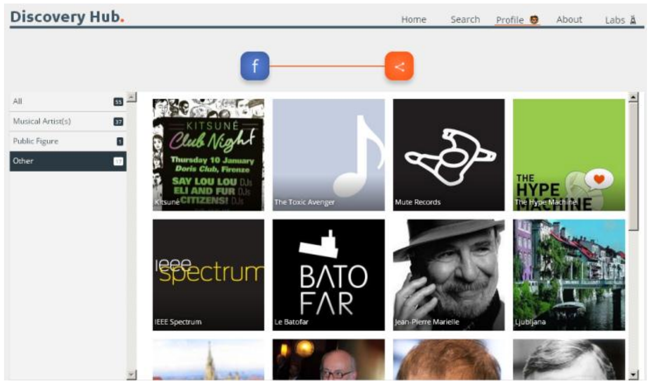


Figure 4: Facebook likes import function

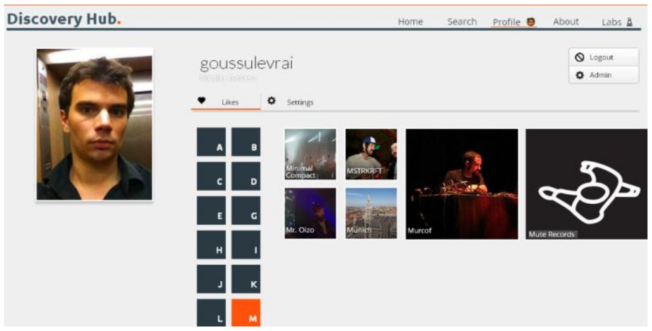


Figure 5: Discovery Hub profile page

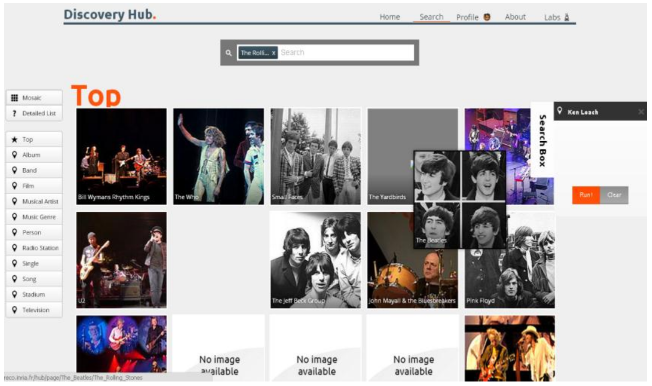


Figure 6: The user is currently dragging *The Beatles* in the "*search box*" to complete *Ken Loach* and launch the query

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

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

[19] http://lookup.dbpedia.org /

[20] http://www.twitter.com

5.2 Results browsing

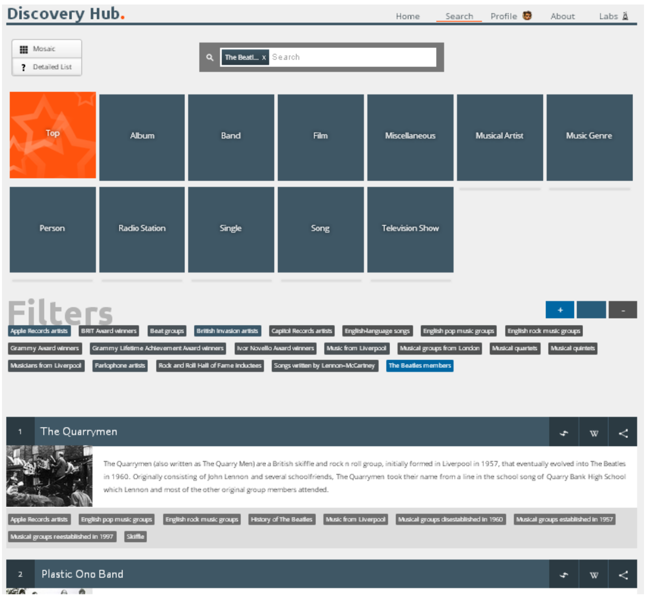
[14] and [15] proved the great value and major role of facets and categorization for exploratory search tasks. According to their experiments they help the users to organize, explore and assess the results with no extra-complexity. One consequence is that the users explore the results space more deeply.

We followed the design guidelines proposed in [14]. The Discovery Hub results space exploration is facilitated thanks to various facets and filters (figure 7). The classes in the propagation domain are used as navigational items to build 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. We limited the number of facets to 14 including the "*top*" one. Classes are chosen in decreasing CPD count order.

A set of filters per facet is proposed using DBpedia categories thanks to the query below. For example on figure 9, user filtered the *Film* facet results with "*2000s comedy-drama films*". These sets of filters are close to the "*categorized overviews*" studied in [14].

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. 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)}
```


Discovery Hub.
Home Search Profile About Labs
Mosaic
Detailed List
The Beatl... x Search

Top | Album | Band | Film | Miscellaneous | Musical Artist | Music Genre
Person | Radio Station | Single | Song | Television Show

Filters + -
Apple Records artists · BRIT Award winners · Beat groups · British Invasion artists · Capitol Records artists · English-language songs · English pop music groups · English rock music groups · Grammy Award winners · Grammy Lifetime Achievement Award winners · Ivor Novello Award winners · Music from Liverpool · Musical groups from London · Musical quartets · Musical quintets · Musicians from Liverpool · Parlophone artists · Rock and Roll Hall of Fame inductees · Songs written by Lennon–McCartney · The Beatles members

1 The Quarrymen

The Quarrymen (also written as The Quarry Men) are a British skiffle and rock n roll group, initially formed in Liverpool in 1957, that eventually evolved into The Beatles in 1960. Originally consisting of John Lennon and several schoolfriends, The Quarrymen took their name from a line in the school song of Quarry Bank High School which Lennon and most of the other original group members attended.

Apple Records artists · English pop music groups · English rock music groups · History of The Beatles · Music from Liverpool · Musical groups disestablished in 1960 · Musical groups established in 1957 · Musical groups reestablished in 1997 · Skiffle

2 Plastic Ono Band

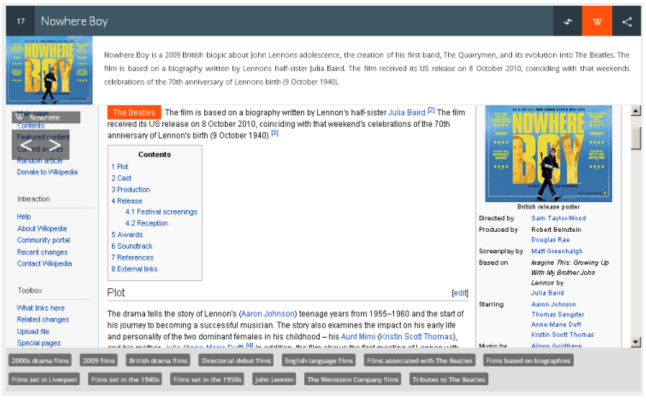
Figure 7: Detailed results mode designed for exploration

To give a real example of results the composite interest query *The Beatles + Ken Loach* offers 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: *2000s comedy-drama films*, *British drama films*, *films associated with the Beatles*, *films directed by Ken Loach*, *films set in Liverpool*.

5.3 Explanation features

When a user is interested or intrigued by an item, he can ask for 3 different explanations thanks to three features. These features are mandatory for composite heterogeneous queries when non-trivial and unattended results are retrieved and need to be explained to receive user's approval. Following explanatory features are presented in a video[21]:

- A feature showing the seeds and results common properties e.g. *dbpedia:hometown Liverpool*, *dcterms:subject English pop music groups*.
- A feature identifying and highlighting crossed references in Wikipedia pages between the resources if they exist (see figure 8).
- A feature showing the relations between the results and the query-resources in a graph format (figure 9). The interest of path-based explanations for spreading activation results was already presented in [12]. When the user goes over a node its abstract appear on the left. It is possible to get even more information with the "*see links in Wikipedia*" functionality highlighting graph neighbors in the Wikipedia page. This graph is built on demand thanks to a SPARQL path query. It is often instantaneous, require few seconds when the graph is dense.


17 Nowhere Boy

Nowhere Boy is a 2009 British biopic about John Lennons adolescence, the creation of his first band, The Quarrymen, and its evolution into The Beatles. The film is based on a biography written by Lennons half-sister Julia Baird. The film received its US release on 8 October 2010, coinciding with that weekends celebrations of the 70th anniversary of Lennons birth (9 October 1940).

The Beatles. The film is based on a biography written by Lennon's half-sister Julia Baird.[2] The film received its US release on 8 October 2010, coinciding with that weekend's celebrations of the 70th anniversary of Lennon's birth (9 October 1940).[3]

Contents
1 Plot
2 Cast
3 Production
4 Release
4.1 Festival screenings
4.2 Reception
5 Awards
6 Soundtrack
7 References
8 External links

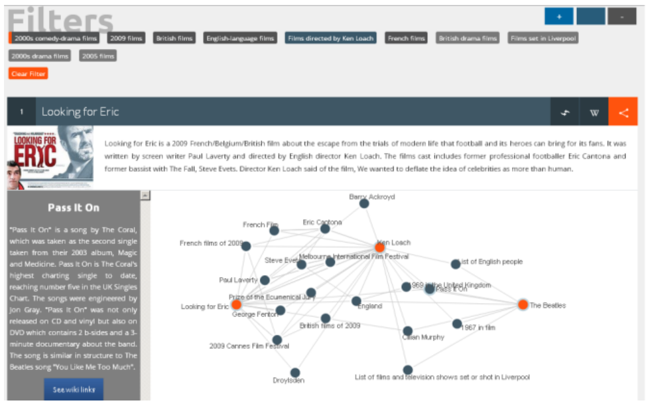
Plot [edit]

The drama tells the story of Lennon's (Aaron Johnson) teenage years from 1955–1960 and the start of his journey to becoming a successful musician. The story also examines the impact on his early life and personality of the two dominant females in his childhood – his Aunt Mimi (Kristin Scott Thomas),

Directed by	Sam Taylor-Wood
Produced by	Robert Bernstein Douglas Rae
Screenplay by	Matt Greenhalgh
Based on	*Imagine This: Growing Up With My Brother John Lennon* by Julia Baird
Starring	Aaron Johnson Thomas Sangster Anne-Marie Duff Kristin Scott Thomas

2000s drama films · 2009 films · British drama films · Directorial debut films · English-language films · Films associated with The Beatles · Films based on biographies · Films set in Liverpool · Films set in the 1940s · Films set in the 1950s · John Lennon · The Weinstein Company films · Tributes to The Beatles

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


Filters + -
2000s comedy-drama films · 2009 films · British films · English-language films · Films directed by Ken Loach · French films · British drama films · Films set in Liverpool · 2000s drama films · 2005 films
Clear Filter

1 Looking for Eric

Looking for Eric is a 2009 French/Belgium/British film about the escape from the trials of modern life that football and its heroes can bring for its fans. It was written by screen writer Paul Laverty and directed by English director Ken Loach. The films cast includes former professional footballer Eric Cantona and former bassist with The Fall, Steve Evets. Director Ken Loach said of the film, We wanted to deflate the idea of celebrities as more than human.

Pass It On

"Pass It On" is a song by The Coral, which was taken as the second single taken from their 2003 album, Magic and Medicine. Pass It On is The Coral's highest charting single to date, reaching number five in the UK Singles Chart. The songs were engineered by Jon Gray. "Pass It On" was not only released on CD and vinyl but also on DVD which contains 2 b-sides and a 3-minute documentary about the band. The song is similar in structure to The Beatles song "You Like Me Too Much".

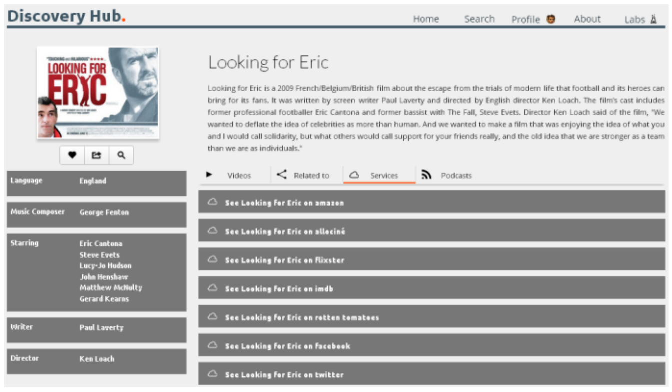
See wiki links

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

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

5.4 Services redirections

The application is named Discovery Hub because it redirects the user toward services which extend its search and discovery experience. These services are proposed regarding the nature of the item considered. For instance "*Looking for Eric*" is a film and we attached to the "*Film*" type services associated to cinema or movies in general like Flixster[22], Rotten Tomatoes[23] or Amazon[24] (complete list on figure 10). The application also proposes services without condition on the considered item type like Youtube or Twitter for instance. We use URL query strings with the result label to perform the redirection[25].



Looking for Eric

Looking for Eric is a 2009 French/Belgium/British film about the escape from the trials of modern life that football and its heroes can bring for its fans. It was written by screen writer Paul Laverty and directed by English director Ken Loach. The film's cast includes former professional footballer Eric Cantona and former bassist with The Fall, Steve Evets. Director Ken Loach said of the film, "We wanted to deflate the idea of celebrities as more than human. And we wanted to make a film that was enjoying the idea of what you and I would call solidarity, but what others would call support for your friends really, and the old idea that we are stronger as a team than we are as individuals."

Language	England
Music Composer	George Fenton
Starring	Eric Cantona Steve Evets Lucy-Jo Hudson John Henshaw Matthew McNulty Gerard Kearns
Writer	Paul Laverty
Director	Ken Loach

See Looking for Eric on amazon
See Looking for Eric on allociné
See Looking for Eric on flixster
See Looking for Eric on imdb
See Looking for Eric on rotten tomatoes
See Looking for Eric on facebook
See Looking for Eric on twitter

Figure 10: Third-party service redirection

5.5 Languages

More and more localized DBpedia versions emerge[26]. As we make use of SPARQL service to access remote endpoints it is very easy to switch from one SPARQL endpoint to another. Today Discovery Hub is operational in Czech, English, French, Italian and Spanish. These specific versions are supported because they retrieve the *wikiPageWikiLink* triples. It is not the case for the German one for instance.

It is interesting to notice that the results reflect some cultural differences. For instance the French-speaking results for Ken Loach are strongly related to the Cannes festival, it is not the case for the others versions results.

5.6 First usages statistics

The Discovery Hub beta was opened on November 2012. At the time of the redaction of this article it already answered to more than 270 queries from various users. The repartition by classes (table 1) shows that users performed queries related to very diverse interests. We can notice that the third first classes are related to culture (*Band*, *Musical Artist* and *Film*) and represent 49.36% of the queries. Nonetheless the high variety of queries confirms the interest of proposing an application which is not limited to a specific domain (e.g. cinema or music).

Class	Query volume
http://dbpedia.org/ontology/Band	18.99%
http://dbpedia.org/ontology/MusicalArtist	16.45%
http://dbpedia.org/ontology/Film	13.92%
http://dbpedia.org/ontology/Person	7.59%
http://dbpedia.org/ontology/Artist	6.33%
http://dbpedia.org/ontology/Settlement	3.80%
http://dbpedia.org/ontology/Country	3.80%
http://dbpedia.org/ontology/Album	2.53%
http://dbpedia.org/ontology/TelevisionShow	2.53%
http://dbpedia.org/ontology/Museum	2.53%
http://dbpedia.org/ontology/Company	2.53%
http://dbpedia.org/ontology/Website	1.90%
http://dbpedia.org/ontology/City	1.90%
http://dbpedia.org/ontology/Book	1.26%
http://dbpedia.org/ontology/Disease	1.26%
http://dbpedia.org/ontology/Boxer	1.26%
http://dbpedia.org/ontology/Software	1.26%
http://dbpedia.org/ontology/Architect	1.26%
http://dbpedia.org/ontology/OfficeHolder	1.26%
Other (12 different classes)	7.59%

Table 1: Discovery Hub users' queries volume by classes

6. CONCLUSION

In this paper, we have presented the Discovery Hub prototype. This application process DBpedia in real-time for exploratory search purpose. Our approach uses a semantic sensitive spreading activation algorithm. Discovery Hub will soon benefit from the *live.dbpedia.org*[27] updates. Thus the application will take into account the latest Wikipedia changes, valuing its real-time capacities.

We plan to extend this work in several directions. A possible research direction offered by our architecture 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. We also want to evaluate the Discovery Hub system thanks to a qualitative evaluation on a large set of users. It will notably allow us to determine the quality of the navigational items we use like facets, filters and to identify browsing patterns.

7. ACKNOWLEDGMENTS

We thank Julien Cojan (SPARQL endpoint) and Olivier Corby (Kgram) for their precious help.

[22] http://www.flixster.com/

[23] http://www.rottentomatoes.com/

[24] http://www.amazon.com/

[25] e.g. http://www.flixster.com/search/?search=Looking+for+Eric

[26] http://dbpedia.org/internationalization

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

8. REFERENCES

[1] Akim, Nazihah Md and Dix, Alan and Katifori, Akrivi and Lepouras, Giorgos and Shabir, Nadeem and Vassilakis, Costas (2011). *Spreading Activation for Web Scale Reasoning: Promise and Problems.* pp. 1-4. In: Proceedings of the ACM WebSci'11, June 14-17 2011, Koblenz, Germany.

[2] 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.*

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

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

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

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

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

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

[9] 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*

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

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

[12] Tim Hussein and Sebastian Neuhaus, "Explanation of spreading activation based recommendations," *SEMAIS 2010, 2010, pp.1-5.*

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

[14] Kules, B. and Shneiderman, B. (2007). Users can change their web search tactics: Design guidelines for categorized overviews. *Information Processing & Management. 44(2): 463-484.*

[15] B. Kules, R. Capra, M. Banta, and T. Sierra. What do exploratory searchers look at in a faceted search interface? *In Proceedings of the 9th ACM/IEEE-CS joint conference on Digital libraries, JCDL '09, pages 313–322, New York, NY, USA, 2009. ACM.*

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

[17] 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.*

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

[27] 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)*