



LarKC

The Large Knowledge Collider:

a platform for large scale integrated reasoning and Web-search

FP7 – 215535

D2.4.1 Spreading Activation Components (v1)

Coordinator: Proletina Todorova, Onto

**With contributions from: Atanas Kiryakov, Damyan
Ognyanoff, Ivan Peikov, Ruslan Velkov, Zdravko Tashev**



Document Identifier:	LarKC/2008/D2.4.1 /vx.x
Class Deliverable:	LarKC EU-IST-2008-215535
Version:	1.0
Date:	06.05.2009
State:	Final
Distribution:	Confidential

EXECUTIVE SUMMARY

This deliverable describes DualRDF and PageRankRDF components related to selection and ranking in RDF graphs, as well as one selection plug-in (SASelector). The spreading activation component (DualRDF) is a scalable and customizable implementation of the popular connectionist method on top of RDF graphs. It allows “priming” of large datasets with respect to concepts relevant to the context and the query. It is meant to serve a basis for experimentation with cognitive-inspired methods for selection and reasoning.

The PageRankRDF provides a basis for experimentation with various weighting and selections. It allows for computation of Page-ranks of the nodes in large RDF datasets. The method determines the importance of a node on the basis of the importance of the nodes linked to it. The PageRank is relatively inexpensive metric and thus a good starting point in weighting a graph.

Spreading activation and PageRank selection components are implemented as extension of TRREE engine – the core of OWLIM and of the Data Layer in LarkC. The components are initiated and managed using SPARQL ASK queries. The deliverable lists and explains the necessary SPARQL queries used to manage and set up the spreading activation and the PageRank features of TRREE engine.

Linked Data Semantic Repository (LDSR) was set up as a testbed for the components delivered here and basis for future experiments on selection and ranking. LDSR represents a selection of few of the most central datasets in the Linking Open Data project, which represents a “reasonable view” to the emerging web of linked data. LDSR has total size of about 370 million explicit statements, which were loaded in LarkC’s Data Layer. Light-weight reasoning was applied to materialize additional 512 million statements. To the best of our knowledge, LDSR is the largest body of common sense knowledge that anyone took the challenge to reason against.

DualRDF and PageRankRDF were evaluated on top of LDSR and to collect first impressions about their performance: it takes just 10 seconds to perform one iteration of PageRank or 3 minutes to compute the ranks of the 100 million nodes in LDSR. The performance of the spreading activation tasks varies considerably in dependence of the parameters of the process. One can use as a reference point the observation that it takes 7 seconds to activate about 7 thousand nodes after spreading of activation from resource <http://dbpedia.org/resource/Berlin> with decay factor 0.25.

There are multiple directions in which the components delivered here can be developed, tuned and optimized. Still, the main driver for the future development of these components will be the feedback from the use cases of the project and the requirements of the reasoning plug-ins, which are using them. Further, there is still more groundwork to be performed in WP2 with respect to tuning cognitively inspired selection plug-ins.



DOCUMENT INFORMATION

IST Project Number	FP7 - 215535	Acronym	LarkC
Full Title	The Large Knowledge Collider: a platform for large scale integrated reasoning and Web-search		
Project URL	http://www.larkc.eu/		
Document URL			
EU Project Officer	Stefano Bertolo		

Deliverable	Number	D2.4.1	Title	Spreading Activation Components
Work Package	Number	WP2	Title	












Date of Delivery	Contractual		Actual	
Status	version 1.0		final x	
Nature	prototype <input type="checkbox"/> report <input type="checkbox"/> dissemination <input type="checkbox"/> other x			
Dissemination level	public x consortium <input type="checkbox"/>			

Authors (Partner)				
Responsible Author	Name	Atanas Kiryakov	E-mail	naso@sirma.bg
	Partner	Onto	Phone	

Abstract (for dissemination)	This deliverable describes the Spreading Activation and PageRank Components and their use as Selection Plug-in in LARKC project. Experimental data over LDSR are provided.
Keywords	Spreading activation, PageRank, selection plug-in

Version Log			

PROJECT CONSORTIUM INFORMATION

Participant's name	Partner	Contact
Semantic Technology Institute Innsbruck, Universitaet Innsbruck	 	Prof. Dr. Dieter Fensel, Semantic Technology Institute (STI), universitaet Innsbruck, Innsbruck, Austria, E-mail: dieter.fensel@sti-innsbruck.at
AstraZeneca AB		Bosse Andersson AstraZeneca Lund, Sweden Email: bo.h.andersson@astrazeneca.com
CEFRIEL - SOCIETA CONSORTILE A RESPONSABILITA LIMITATA		Emanuele Della Valle, CEFRIEL - SOCIETA CONSORTILE A RESPONSABILITA LIMITATA, Milano, Italy, Email: emanuele.dellavalle@cefriel.it
CYCORP, RAZISKOVANJE IN EKSPERIMENTALNI RAZVOJ D.O.O.		Michael Witbrock, CYCORP, RAZISKOVANJE IN EKSPERIMENTALNI RAZVOJ D.O.O., Ljubljana, Slovenia, Email: witbrock@cyc.com
Höchstleistungsrechenzentrum, Universitaet Stuttgart		Georgina Gallizo, Höchstleistungsrechenzentrum, Universitaet Stuttgart, Stuttgart, Germany, Email: gallizo@hlrs.de
MAX-PLANCK GESELLSCHAFT ZUR FOERDERUNG DER WISSENSCHAFTEN E.V.		Dr. Lael Schooler Max-Planck-Institut für Bildungsforschung Berlin, Germany Email: schooler@mpib-berlin.mpg.de
Ontotext Lab, Sirma Group Corp		Atanas Kiryakov, Ontotext Lab, Sofia, Bulgaria Email: atanas.kiryakov@sirma.bg
SALTLUX INC.		Kono Kim, SALTLUX INC, Seoul, Korea, Email: kono@saltlux.com
SIEMENS AKTIENGESELLSCHAFT		Dr. Volker Tresp, SIEMENS AKTIENGESELLSCHAFT, Muenchen, Germany, E-mail: volker.tresp@siemens.com
THE UNIVERSITY OF SHEFFIELD		Prof. Dr. Hamish Cunningham, THE UNIVERSITY OF SHEFFIELD Sheffield, UK, Email: h.cunningham@dcs.shef.ac.uk

<p>VRIJE UNIVERSITEIT AMSTERDAM</p>		<p>Prof. Dr. Frank van Harmelen, VRIJE UNIVERSITEIT AMSTERDAM, Amsterdam, Netherlands, Email: Frank.van.Harmelen@cs.vu.nl</p>
<p>THE INTERNATIONAL WIC INSTITUTE, BEIJING UNIVERSITY OF TECHNOLOGY</p>		<p>Prof. Dr. Ning Zhong, THE INTERNATIONAL WIC INSTITUTE, Mabeshi, Japan, Email: zhong@maebashi-it.ac.jp</p>
<p>INTERNATIONAL AGENCY FOR RESEARCH ON CANCER</p>	 <p>International Agency for Research on Ca Centre International de Recherche sur le Ca</p>	<p>Dr. Paul Brennan, INTERNATIONAL AGENCY FOR RESEARCH ON CANCER, Lyon, France, Email: brennan@iarc.fr</p>



TABLE OF CONTENTS

LIST OF ACRONYMS	8
LIST OF TABLES	8
LIST OF FIGURES	8
1. INTRODUCTION.....	9
1.1. RELATIONS TO OTHER DOCUMENTS	10
1.2. SPREADING ACTIVATION.....	10
1.3. PAGERANK ALGORITHM	11
1.4. RDF(S), OWL AND SPARQL.....	12
2. ARCHITECTURAL OUTLINE.....	14
3. DUALRDF: SPREADING ACTIVATION OVER RDF GRAPHS	16
3.1. PRIMING OF RDF GRAPHS	16
3.2. SEMANTIC PRIMING.....	16
3.3. MANAGEMENT OF THE SPREADING ACTIVATION COMPONENT.....	17
3.4. CONTEXT-AWARE INFERENCE AND QUERY EVALUATION.....	18
4. SPREADING ACTIVATION SELECTION PLUG-IN.....	19
4.1. IMPLEMENTATION APPROACH	19
4.2. USING THE SASSELECTOR PLUG-IN	20
5. PAGERANKRDF COMPONENT.....	21
5.1. IMPLEMENTATION APPROACH	21
5.2. SIMILARITIES AND DIFFERENCES BETWEEN SA AND PAGERANK	22
5.3. USING PAGERANK COMPONENT.....	22
6. EXPERIMENTS WITH LINKED DATA.....	23
6.1. LINKED DATA.....	24
6.2. LINKED DATA SEMANTIC REPOSITORY.....	25
6.3. LOADING AND INFERENCE STATISTICS	27
6.4. LOADING AND INITIAL INFERENCE AGAINST LDSR	28
6.5. OWL:SAMEAS OPTIMISATIONS	29
6.6. RANKING AND PRIMING EXPERIMENTS	30
7. CONCLUSION	33
8. REFERENCES.....	34



List of Acronyms

SA	Spreading Activation
RDF	Resource Description Framework
LDSR	Linked Data Semantic Repository
LOD	Linked Open Data

List of Tables

Table 1 Statistics from the loading and inference of the LDSR dataset	28
Table 2 Single-Node Activation Results	31
Table 3 Results in Dependence of the Decay Factor	31
Table 4 Results in Dependence to the Firing Threshold	32
Table 5 Multiple-Node Activation Results	32

List of Figures

<i>Figure 1 LarkC pipeline behaviour</i>	<i>14</i>
<i>Figure 2 LarkC Data Layer API and DualRDF and RDFPageRank components</i>	<i>15</i>
<i>Figure 3 Map of the datasets in Linking Open Data. W3C SWEO Community Project</i>	<i>25</i>

1. Introduction

The spreading activation (SA) components described in this deliverable are designed to serve as a basis for experimentation with cognitively-inspired methods for selection¹ and more generally for reasoning. The rationale was to provide a scalable and customizable implementation of the most popular connectionist method, namely spreading of activation, on top of RDF datasets. Such component allows for evaluation of different activation schemata (e.g. weighting approaches and activation functions) and of their appropriateness for specific reasoning tasks, scenarios, datasets and ontologies.

To provide some basis for experimentation with different weighting and selection configurations, we have developed an implementation of the PageRank algorithm, which allows for ranking of nodes in large RDF datasets. It is an implementation of the original algorithm, which operates on the graph of web pages (as nodes) and hyperlinks (as edges) and which determines the importance of a node based on the importance of the nodes that refer back (have outbound links) to it.

Technically, SA and PageRank are implemented as extensions of the TRREE engine – the core of OWLIM and the Data Layer of LarKC. Specific configuration parameters are required in order to enable the SA and PageRank features. The corresponding Java libraries should be available.

The usage of SA and PageRank features is initiated and managed through SPARQL ASK queries. The engine can be setup so that after activation spreading, it will work only with the “active” part of the graph (e.g. for the sake of query evaluation and reasoning). This provides an easy and efficient approach for setting a reasoning pipeline, where after selection, the plug-ins can receive a reference to a repository, where they “see” only the selected data, i.e. only the output of the selection plug-in. This approach allows one to avoid copying big amounts of data and transferring them between the plug-ins, shall the specific reasoning setup allows usage of shared storage across (some of) the plug-ins.

At a higher level, there is a shrink-wrap selection plug-in which performs the SA starting from the URIs referred to in a SPARQL query and returning as a result a triplesets, which contains the selected part of the graph.

The deliverable is structured as follows:

- Section 1 continues with comments on the role of this deliverable and the links it has to other documents, followed by brief introduction to the notion of spreading activation and to the basic PageRank algorithm;
- Section 2 elaborates the positioning of the SA and PR components and the SA plug-in with respect to the architecture of the Data Layer of LarKC;
- Section 3 presents the SA component and its usage through the Data Layer and ORD API;

¹ In LarKC project “selection” is a phase in a pipelined reasoning process, during which part of the data (an RDF dataset), relevant to the reasoning task, is selected for further processing.



- Section 4 documents the SA selection plug-in;
- Section 5 discusses the implementation and the usage of the PageRank component;
- Section 6 presents the evaluation of the SA and the PageRank components with respect to datasets from the Linking Open Data project.

1.1. Relations to Other Documents

Work package 2 of LarKC project applies various retrieval and selection methods to large datasets. This deliverable (D2.4.1) describes two components used for selection – spreading activation and PageRank – and provides hints as to their possible uses. It also presents the spreading activation plug-in, which makes available a specific setup of the SA components as LarKC selection plug-in.

Deliverable D2.1.1, [6], defines the basic notions of RDF dataset and tripliset (the data model adapted in LarKC) and discusses the selection and retrieval methods on top of them. Deliverable D2.2.1, 2.5.1, [14], describes other selection components, developed in WP 2 of LarKC.

Deliverable D2.3.1, [19], contains some general considerations on spreading activation and on PageRank and their applicability to dealing with massive, incomplete, and inconsistent sources of information. The deliverable describes the overall goal of WP 2 of LARKC project – sub-setting and the applicability of some theories on memory and representation from cognitive science to solve the problem addressed in WP2.

Deliverable D5.2.1,

[13], introduces the overall concept of LarKC platform and describes the architecture of the prototype. The deliverable is useful to understand the proper place of the selection components, described in this deliverable, mapped to the entire LarKC architecture.

Deliverable D3b from RASCALLI project [20] describes in brief the usage of an earlier version of the spreading activation component presented here for incomplete open-domain inference and as a basis for the cognitive architecture of a virtual personal assistant.

1.2. Spreading Activation

Spreading Activation (SA) was introduced as an approach for modelling of the human memory; more generally, it is mechanism used in the connectionist AI methods (e.g. artificial neural networks) that aim at modelling of the cognitive processes of the human brain by following its low-level structure. SA was originally coined as a term and applied to problems like categorization, [5]. Later on it was applied in various other tasks such as information retrieval, search engine ranking, etc. The essential SA ideas are applicable in weighted graph models, which are similar to neural networks.

The SA process starts with the activation of certain nodes in a graph by assigning initial activations to them and then propagating (or spreading) them to the connected nodes,



according to some activation function, which involves the weights² of the links. The propagation is an iterative process which uses a decay factor to model the intuition of activity which spreads and dies out, as some sort of electrical energy. After a predetermined number of iterations, the process stops and the nodes, the activations of which exceed a predefined threshold, are considered active. The inactive nodes are filtered out and the selection process ends up with a sub-graph, considered the most related to the initially chosen nodes.

Additional parameters of the method are:

- the firing threshold (to be propagated an activity should have a greater value than the one specified as a threshold);
- the graph edge weights, which indicate the strength of the relationship between the nodes in the network.

SA can be used as a method for selecting a small part of a network (graph) which is most related to a set of initially chosen concepts (nodes) in the network. Such a selection method is applicable whenever the processing of the entire network is computationally unfeasible or otherwise unreasonable.

1.3. PageRank Algorithm

The PageRank is a link analysis algorithm used by Google search engine. It assigns numerical ranks to each element in a hyperlinked set of documents. The PageRank, as defined by Brin and Page, [3], is the probability that a random surfer will visit a page, assuming that the surfer clicks on links randomly and never goes back to a page already visited. The intuition behind the analysis is that the PageRank of a page would be higher if there are many pages pointing to it. In essence, the PageRank is a model of the visitor behaviour and assumes that a visitor randomly walks a graph jumping from node to node. It is equally probable that a visitor will follow one of the connected outgoing edges or will start over from an unrelated node. In this model the PageRank rank of a certain node is defined as the likelihood that the visitor will finally arrive at it.

The PageRank rank can be interpreted as the result of an iterative process in which each node distributes equal portion of its rank across the outgoing links. The process repeats and stops when the changes fall below a certain threshold. In this way it is guaranteed that the “important” nodes will transfer their importance to “trusted neighbours”, thus increasing their importance. In a strongly interconnected sub-graph, the nodes with isolated (local) importance will gain smaller weight because the whole sub-graph will be poorly ranked.

It is relatively inexpensive to compute the PageRank metric even when working with large graphs because it requires time and space linear to the size of the graph. Although it is a static and ignores the local specifics of the data modelled by the graph, the PageRank metric is a good starting point in weighting a graph, especially when no other metric is available.

² Here we use “weight” to refer the strength of the links in the graph and “activation” to indicate the “energy” or the level of relevance of a particular node. Some authors use “weight” as “activation” as well.

1.4. RDF(S), OWL and SPARQL

Resource Description Framework (RDF) is a language for representing information about resources in the World Wide Web

[15]. Although it was designed to represent metadata about Web resources, RDF has much broader use as a generic data model for structured data management and reasoning. The atomic data element in RDF represents a statement about resource or a blank node. Each statement is a triple of the format `<Subject, Predicate, Object>`, e.g. `<John, loves, Mary>` Or `<Mary, hasBirthday, "14.11.1972">`. RDF description can be seen as directed labelled graph, where each triple defines an edge directed from the subject to the object and labelled with the predicate. The nodes of the graph could be URI (unified resource identifiers, e.g. an URL), blank node (auxiliary nodes), or XML literal. The predicates are always URIs. Literals are not allowed in subject position, i.e. they cannot be the start of an edge in the graph - intuitively, literals are used to describe resources identified by URIs, but there is no point in describing literals, because they represent primitive data value.

RDFS, [2], is the schema language for RDF, which allows definitions of classes of resources and properties. OWL, [7], is an ontology language, which allows more comprehensive logical descriptions of the schema elements.

SPARQL [18], is a SQL-like query language for RDF data, specified by the RDF Data Access Working Group of W3C. It differs from SQL in the following aspects:

- SPARQL does not contain specific Data Definition Language (DDL) provisions, because the schemata are represented in both RDFS and OWL as standard RDF graphs, thus requiring no specific language to deal with them;
- SPARQL is not a Data Modification Language (DML), i.e. one cannot insert, delete, and update RDF graphs using SPARQL. The major reason for this is that there is still no consensus on the optimal DML design for RDF.

SPARQL supports four types of queries:

- SELECT queries –return n-tuples of results just like the SELECT queries in SQL
- DESCRIBE queries – return RDF graph. The resulting graph describes the resources, which match the query constraints. Usually, a description of a resource is considered an RDF-molecule³, forming the immediate neighbourhood of an URI
- ASK queries – provide positive or negative answer indicating whether or not the query pattern can be satisfied

³ The molecules of an RDF graph are the finest components into which the graph can be decomposed without loss of information. If an RDF graph is free of blank nodes, each triple constitutes a molecule; otherwise, triples are grouped into molecules according to the background ontology and the decomposition algorithm. [8]



- CONSTRUCT queries - return an RDF graph constructed by substituting for the variables in the graph template and combining the triples into a single RDF graph by set union.

Named graph,

[4], is an RDF graph with assigned name in the form of an URI reference. A consequence of this proposal for extended RDF model is that the names of the graphs can be described using metadata. However, the definition of a named graph leaves unanswered multiple modelling questions. A better definition of a named graph is provided in the specification of SPARQL, where a named graph is the building block of a dataset. Intuitively, a *dataset* integrates several RDF graphs in such a way that each graph can be distinguished.

Formally, a dataset can be represented as an RDF multi-graph, The RDF multi-graph, in its turn, can be represented as a set of quadruples of the following type: $\langle S, P, O, G \rangle$. The first three elements of the quadruple, $\langle S, P, O \rangle$, represent an RDF statement; the fourth element, G , represents the name of the named graph.

A *triple set*, as discussed in [6] and [13], is a multi-graph, a subset of the set of all quadruples in the dataset. The triple set is also a part of the dataset. Triple sets are named, i.e. each triple set is associated with a unique name. The rationale behind the enhancement of the RDF data model with triple sets is that the named graphs allow for tracking provenance, for example, when multiple graphs from different sources are merged or referenced. Moreover, since the triple sets allow for designation or tagging of parts of a dataset, they are especially useful when selecting parts of the dataset.

2. Architectural Outline

LarkC platform aims at supporting massive distributed incomplete reasoning at Web-scale. In order to simplify the development process and to maximise the reuse of software components a plug-in framework approach is implemented. The platform serves functionality to construct, configure and invoke different plug-ins upon request from the Decider, which is responsible for orchestrating the complete process (see **Figure 1**). The selection plug-in plays role in narrowing the used dataset by using various heuristics, statistics or other strategy.

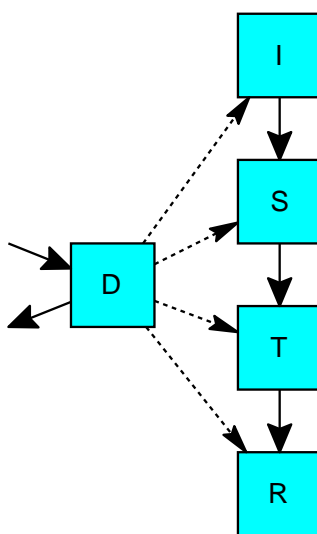


Figure 1 LarkC pipeline behaviour

(D is for Decider, S for Selector, T for Transformer, and R for Reasoner)

From plug-in perspective the LarkC platform is composed by:

- Pug-in API - responsible for defining the contract between the different components;
- Data Layer API - specifies several mechanisms to exchange RDF data between the plug-in instances.

Data Layer API makes the RDF data transparent for the plug-ins and automates the passing of RDF data by reference and value. Currently, there are four different ways to pass RDF statements:

- By value – if necessary, all the statements are serialized and transferred between the plug-ins;
- By dataset reference – only a pointer to SPARQL endpoint and dataset is used to identify the set of data;
- By tripliset (labelled group) reference – ORDI framework introduces a rich RDF data model, allowing it group arbitrary statements (part of those belonging to a single dataset) in a named collection, called tripliset as discussed in section 1.4.
- By URI reference – the URI is resolved by HTTP 303 redirect and then downloaded

To increase the efficiency of the RDF priming (spreading activation) and ranking algorithms two new components have been introduced in LarKC Data Layer, as displayed on Figure 2:

- DualRDF: implements RDF priming as discussed in section 4;
- RDFPageRank: implements RDF rank calculation as presented in section 5.

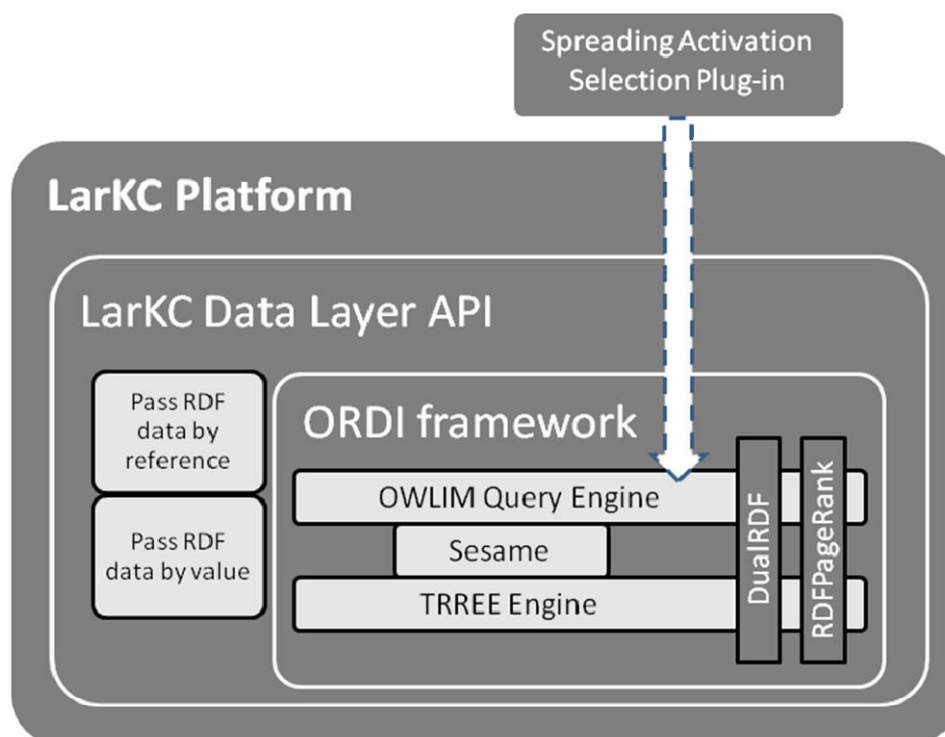


Figure 2 LarKC Data Layer API and DualRDF and RDFPageRank components

Spreading activation components are exposed in the LarKC platform as selection plug-in that receives as input RDF data from the local repository and a query and returns a reduced RDF set of statements, relevant to the query. In order to select a reduced set of data the plug-in has to process huge amounts of information, which, in cases of distributed deployed environments, may take significant input/output time.

The selection plug-in implements just one possible configuration for usage of the spreading activation component. Both DualRDF and RDFPageRank can be managed at a lower level via SPARQL ask queries, as specified in section 4 and 5 respectively.

3. DualRDF: Spreading Activation over RDF Graphs

The first version of DualRDF was developed in the scope of the project RASCALLI, [20]. The component which provides the basic SA functionality within LarKC is a further development of DualRDF.

3.1. Priming of RDF Graphs

The specific approach we followed is named “RDF-priming”; it consists of “activating” a portion of the dataset through spreading activation. Spreading activation is initiated by an *activation seed*: set of concepts, relevant to the inference task (e.g. query terms or contextual clues). The objective of the activation process is to end up with a “primed” dataset, a sub-set of the entire dataset loaded in the repository. The “primed” dataset should fulfil the following requirements:

- to be small-enough so that a more efficient application of complex symbolic processing (inference or query evaluation) is enabled;
- to contain knowledge, relevant to the current context, e.g. to exclude the irrelevant knowledge. For instance, in the LDSR dataset, described in section 6.2, most of the objects belong indirectly to many classes due to the large class hierarchy of OpenCyc. Although each class is relevant to some context, at any specific moment most of the classes are irrelevant.

DualRDF deals with weighted-RDF graphs and activation levels. The activation is spread by means of matrix calculus operations. In each moment only the active part of the graph along with its neighbourhood is handled. The rationale behind using the neighbourhood of the active part of the graph is to allow for a change in the „active“ sub-graph over time. As some concepts from the neighbourhood pass above the activation threshold, they become part of the „active“ sub-graph and their neighbourhood is retrieved from DualRDF. At the same time, some areas of the active sub-graph are „deactivated“ and as their activation goes below the threshold they are removed.

Currently, the approach followed does not support advanced link-weighting schema. Although, DualRDF can be used to set weights to specific statements, the implemented configuration mechanism allows setting weights at predicate level, e.g. one can determine specific weight for all statements with specific URI in predicate position.

3.2. Semantic Priming

If spreading activation and materialization is performed on top of the RDF graph before the priming starts, the newly inferred statements are involved in the activation process. This way, specific nodes or portions of the graph can be activated, although there may be no explicit link between them and the activation seed or the path in the graph between them and the seed is too long, to allow for activation.

Effectively, this allows hybrid approach where connectionist and symbolic methods work in conjunction, as symbolic inference is used to infer links, which are used for priming, which affects the results of heavier symbolic inference techniques used at a later phase. As long as



the priming process considers the semantics of the graph, such setup can be considered to deliver a sort of “semantic priming” functionality.

The weights of the inferred statements are intentionally set to lower values, as compared to those of the explicit ones to match the intuition that inferred links are in a sense “weaker” (or less trustworthy). The approach for weighting of the inferred statements is subject to further experiments. One obvious approach would be to follow the mechanisms from the so-called probabilistic logics, where the “credibility” of the inferred facts is function (e.g. multiplication) of the “credibility” of the facts it was inferred from.

3.3. Management of the Spreading Activation Component

We have already pointed out that SPARQL specifies four types of queries. The SA component can be managed using ASK queries. Unless one of the below listed predefined predicates is used, ASK queries return “Yes” or “No”, just like in the case of “regular” evaluation. However, the behaviour of ASK queries is modified so that, when they return positive answer, there is a side-effect that all variable bindings are set to a pre-determined level of activation. Note that in a regular evaluation of ASK query the engine will return “Yes” when the first binding is found and will discontinue the evaluation while in our case the engine will enumerate all possible bindings of the variable to set higher level of activation for all of them.

Several pre-defined predicates can be used to control the spreading activation process. Since they all use the namespace <http://www.ontotext.com>, it is advisable to add the following line in front of each of them.

```
PREFIX onto: <http://www.ontotext.com#>
```

To enable/disable the spreading activation feature, use the following query:

```
PREFIX onto: <http://www.ontotext.com#>
ASK {_:b1 onto:enableSpreading "true" .}
```

To determine the activation decay factor during the activation spreading, use the following query (“0.85” can be replaced with any valid float number between 0 and 1):

```
ASK {onto:SADecayFactor onto:setParam "0.85" .}
```

To set the firing threshold, use the following query (“0.15” can be replaced with any valid float number between 0 and 1):

```
ASK {onto:SAFiringThreshold onto:setParam "0.15" .}
```



The filter threshold is the value used to filter out the “inactive” statements. A statement is filtered out if it does not contain at least one node with activation level greater than the value specified for the threshold. To set the value of the threshold, use the following query:

```
ASK {onto:SAFilterThreshold onto:setParam "0.80" .}
```

To set the maximum number of nodes that could fire during a single activation spreading session, use the following query (“100000” can be replaced with any valid natural number):

```
ASK {onto:SAMaxNodesFiredPerCycle onto:setParam "100000" .}
```

To determine the weights of statement at predicate level, use the following query (`rdfs:subClassOf` can be replaced with any predicate URI and “0.5” with valid float number between 0 and 1):

```
ASK {rdfs:subClassOf onto:assignWeight "0.5" .}
```

3.4. Context-aware Inference and Query Evaluation

The context-aware query evaluation is a straightforward process once the RDF-priming is performed and the results reside in DualRDF. DualRDF filters out the inactive parts of the graph, so that any higher-level task such as the query evaluation and the inference can work only in the scope of the “primed” graph. This later speeds up both query evaluation and inference as long as many of the intermediary bindings of the variables of the queries or the rules are filtered out at a very early stage.

The approach taken also allows easy changing of the size of the active part of the graph – this is just a matter of changing the activation threshold. There is no need for one to perform the priming operation again with different parameters.

To make such “transparent” management and filtering possible, DualRDF is implemented as an extension of both OWLIM and TRREE engine, so that:

- It can intercept specific SPARQL queries, used for management purposes;
- Alternate the results returned through OpenRDF’s SAIL⁴ interfaces.

⁴ In Sesame, SAIL stands for Storage And Inference Layer. The SAIL APIs allow for isolation of the backend persistency and reasoning components from the rest of the infrastructure. For more details, refer to the documentation of Sesame 2.0, <http://www.openrdf.org/doc/sesame2/system/>.

4. Spreading Activation Selection Plug-In

The Spreading Activation plug-in implements a method of selection on top of DualRDF. Below we describe the usage of the plug-in.

4.1. Implementation Approach

The Spreading Activation selection plug-in executes the following steps:

1. The Spreading Activation for the DualRDF repository is turned on by executing the following SPARQL ASK query (use the PREFIX `onto: <http://www.ontotext.com#>` as specified in section 3.2):

```
ASK {_:b1 onto:enableSpreading "true" }
```

The parameters of the Spreading Activation process are configured by executing the following ASK queries. Note that special system URIs, recognized by the DualRDF repository, are used.

```
ASK {onto:SAInitialActivation onto:setParam "0.66" }
ASK {onto:SADecayFactor      onto:setParam "0.85" }
ASK {onto:SAFiringThreshold  onto:setParam "0.15" }
ASK {onto:SAFilterThreshold  onto:setParam "0.80" }
```

2. The input query of the Spreading Activation selection plug-in is processed and transformed into a SPARQL ASK query. This transformation involves a simple translation of the main verb of the query e.g. SELECT into ASK. The resulting ASK query is then executed over the repository. The ASK query automatically activates the resulting URI bindings of the result set and assigns the configured initial activation weight (e.g. 0.66 as above) to them.
3. The spreading activation process is triggered by executing the following ASK query:

```
ASK {_:b1 onto:spreadActivations "true" }
```

This query internally starts the spreading activation. When it finishes, the RDF graph in the repository is “primed”, e.g. only the “active” URIs are externally visible and selectable.

4. All visible statements (i.e. all statements, which have visible URIs as subject or objects), are associated by the selection plug-in with a uniquely generated new tripleset. The selection plug-in will return this tripleset as a result. Due to the priming of the RDF graph, the resulting tripleset will be associated only with the active part of the graph.
5. The spreading activation over the repository is turned off by executing the following SPARQL query:

```
ASK {_:b1 onto:enableSpreading "false" }
```



4.2. Using the SASelector Plug-in

As any other selection plug-in running over the LarkC platform, the SA Selector plug-in accepts an input query through its `setInputQuery()` method and then creates a tripliset containing the selected statements from the DualRDF graph:

```
// instantiate the plug-in
SASelector sa = new SASelector();

// set the SPARQL query to be used for the SA selection
sa.setInputQuery(
    new SPARQLQueryImpl(
        "SELECT ?s ?p ?o WHERE { ?s ?p ?o FILTER(?s = 'hello world') }"
    ));

// select a tripliset associated with active part of the graph
LabelledGroupOfStatements ts = s.select(null, null, null);
```

5. PageRankRDF Component

Here we present the PageRankRDF component, computing PageRanks for nodes of RDF graph. The PageRank algorithm is briefly introduced in section 1.3.

5.1. *Implementation approach*

Like the Spreading Activation component, the PageRankRDF does not introduce new APIs, it is managed and used through SPARQL queries. It is implemented on top of the DualRDF extension of the TRREE engine, which implements RDF priming, as discussed in section 3.

The SA component can be managed using ASK queries. Several pre-defined predicates (with namespace `PREFIX onto: <http://www.ontotext.com#>`) can be used to manage the ranking process.

The PageRank of the RDF graph in the repository is recomputed by executing:

```
ASK {_:b1 onto:computePageRank "true" }
```

This query drops the previously computed PageRank values and re-computes new ones, using the current RDF graph. To access the ranks of the URIs in the repository, use the system predicate `onto:hasPageRank`. The predicate maps all nodes in the repository to their respective rank value (floating point number in the range [0,1]). For example:

```
SELECT ?rank {<http://example.com/#> onto:hasPageRank ?rank }
```

Note that the PageRank metric is not updated automatically with the changes in the repository. After considerable repository updates, the repository owner should take care to re-compute the ranks manually by executing the `computePageRank` query as shown above.

The PageRank algorithm takes time linear to the size of the underlying RDF graph. Experiments and performance statistics are provided in section 6.3.

To limit the number of iterations, one can set the `PRMaxIterations` parameter:

```
ASK {onto:PRMaxIterations onto:setParam "10" }
```

where 10 could be replaced with any positive integer.

The PageRank algorithm iteratively computes approximations of the metric. Any iteration tends to converge to the actual node rank. It makes sense to limit the maximal allowed error of the approximation in order to cut off iterations which make insignificant difference. To set the limit, use the `PREpsilon` parameter:



```
ASK {onto:PREpsilon onto:setParam "0.01"}
```

where 0.01 might be replaced with any floating point value of the interval [0,1].

5.2. Similarities and Differences between SA and PageRank

One might approximate the PageRank algorithm using Spreading Activation by initially activating the whole graph with activity weights of $1/N$ and then spreading activation with decay factor of 1 and firing threshold of 0. For this to work as PageRank does, a weight of $1/\#(A)$ (where $\#(A)$ is the number of outgoing links of A) should be assigned to each edge $\langle A, B \rangle$ in the graph.

Although the two algorithms are similar (SA is the more general one), the SA method cannot simulate the damping factor of the PageRank algorithm (1 minus the probability that the random visitor will stop jumping from node to node and will start anew from a random node). It is a minor difference and it gets insignificant when considering that usually the damping factor is set close to 1 (empirically tested value measured for the purposes of a search engine is said to be around 0.85, [3]).

Despite their computational similarities, the typical usage of the two algorithms is different. In the process of RDF priming presented in section 3.1, spreading activation is used to implement context- or query-sensitive ranking of the nodes in the graph. On the other hand, PageRank can be seen as generic ranking mechanism, which unveils some static qualities of the RDF graph, independent of the context.

The PageRank for RDF graphs is an interesting starting point for experimentation. For example, it can be used for spreading activation and weighting in the following manner:

- The weight of an RDF statement can be determined as a function of the ranks of its subject and object. It is possible to consider a function which takes the predicate and its rank
- The ranks can be used during selection as a static relevance score, which complements, filters, or tunes the dynamically calculated relevance with respect to the query and the context.

5.3. Using PageRank Component

The PageRank of a graph is useful when opting for displaying and sorting of the result set of a search operation by significance. Although the PageRank metric is context-insensitive, it comes at a comparatively low price and can be used when it is too expensive to compute the local relevance.

The following sample SPARQL query selects the 100 most “important” nodes in the RDF graph using the PageRank metric:

```
SELECT ?n {?n onto:hasPageRank ?r} ORDER BY DESC(?r) LIMIT 100
```

6. Experiments with Linked Data

To test the components for selection and ranking in RDF graphs presented in this document, we needed a dataset that meets the following criteria:

1. To be large enough, so to allow acquiring of meaningful feedback on the scalability of the implementations. The larger the dataset, the better, but we needed a dataset that is larger than 100 million statements, because most of the semantic repositories can manage smaller datasets in the main memory of a contemporary server;
2. To present real-world data, which expose natural connectivity patterns and graph structures;
3. To have a predictable dataset, for which humans can make assumptions regarding the results of the ranking and selection components based on common knowledge and common sense;
4. To have some semantics attached to the data, or at least, to be easy to attach semantics to the data, so that further experiments with reasoning can be performed.
5. If possible, to integrate data from different data-sources, so that the graph structure is diverse.

The use cases of the project already provide comprehensive datasets, which allow for validation of the technology and evaluation of its exploitation potential. Still, their data do not match several of the requirements set above, so they were not appropriate for intermediate evaluation of early versions of selection and ranking components.

Natural selection for data matching the above requirements is the collection of the Linking Open Data (LOD) datasets, which represents a set of interconnected datasets, published in accordance with the “linked data principles” (section 6.1). Still, getting the full collection of the LOD datasets was not appropriate, because some of the datasets were inappropriate for our purposes. For instance, the YAGO fragment of DBpedia contains plenty of faulty classifications of Wikipedia articles, due to the natural limitations of the accuracy of the extraction techniques used for its generation. These inaccuracies are relatively small in number and probably do not represent a serious problem for humans who explore DBpedia, however, they can cause severe inconsistency during any type of reasoning. While, reasoning with inconsistency is an interesting subject in principle, it brings complexity of a nature unrelated to the selection and ranking techniques we need to benchmark here. Thus, we defined Linked Data Semantic Repository (LDSR, section 6.2), as “reasonable view” to the LOD data, namely, a selection of the LOD dataset which meets our requirements. Another such “reasonable view”, named Pathway and Interaction Knowledge base (PIKB) is defined and used in WP7A. While it presents a very interesting experiment on its own, it is too specialized and does not meet requirement 3 above, regarding predictability.

This section presents an introduction to linked data and LDSR and continues with several experiments based on LDSR.

6.1. *Linked Data*

The objective of linked data is to find other, related data. [1] The “linked data” is defined by Tim Berners-Lee as RDF graphs, published so that they can be navigated across servers by following the links in the graph (in a manner similar to the way the HTML web is navigated). The publishers of linked data should comply with four simple design principles:

1. Use URIs as names for things
2. Use HTTP URIs so that people can look up those names.
3. When someone looks up a URI, provide useful information.
4. Include links to other URIs, so that they can discover more things.

In fact, most of the RDF datasets fulfil principles 1, 2, and 4. Thus, the novelty of the design principles relates to the requirement to enable Semantic Web browsers to load HTTP descriptions of RDF resources based on their URIs. To this end, data publishers should make sure that:

- the “physical” addresses of the published data are the same as the “logical” addresses, used as RDF identifiers (URIs);
- upon HTTP request, the server should return the so-called RDF-molecule, i.e. the set of triples which describe the resource.

Linked Open Data (LOD) initiative aims at making data available to everyone⁵. The goal of the project is to extend the Web by publishing open data sets as RDF and by creating RDF links between data items from different data sources. The central dataset of the LOD is DBpedia - an RDF extract of the Wikipedia. Because of the many mappings between other LOD datasets and DBpedia, DBpedia serves as a sort of a hub in the LOD graph providing a certain level of connectivity. Among the linked LOD datasets are the following: the RDF representation of Wordnet (the most popular lexical knowledge base), Geonames (a database of all geographic features on Earth), World Factbook⁶ (an RDF version of CIA’s resource), UniProt⁷ (the largest integrated database with protein and gene-relates information), and OpenCyc⁸ (the most popular upper-level ontology).

UMBEL project deserves a special attention because it provides a consistent taxonomy interlinking DBpedia entities to OpenCyc concepts. In this way the data from DBpedia (or from any other datasets connected to it) can be interpreted with respect to the semantics of OpenCyc. In next section we discuss in more details DBpedia, Geonames, UMBEL, and Wordnet datasets.

⁵ <http://esw.w3.org/topic/SweoIG/TaskForces/CommunityProjects/LinkingOpenData>

⁶ <http://www4.wiwiwiss.fu-berlin.de/factbook/>

⁷ <http://www.uniprot.org/>

⁸ <http://www.opencyc.org/>

Currently LOD contains more than 40 datasets (See Figure 3) with total volume above 4.5 billion statements, interlinked with additional 5 million statements.

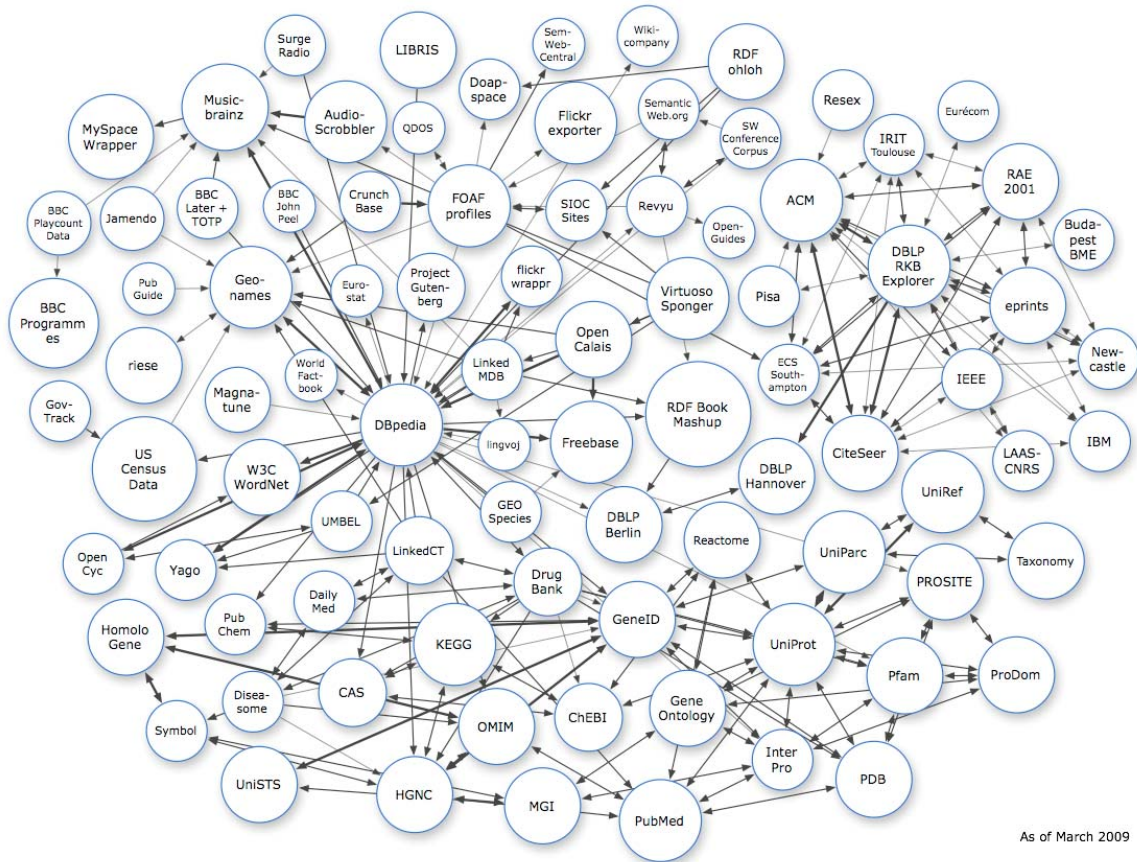


Figure 3 Map of the datasets in Linking Open Data. W3C SWEO Community Project⁹

6.2. Linked Data Semantic Repository

Linked Data Semantic Repository (LDSR) represents a "reasonable" view to the web data. In LDSR we selected several of the central datasets of the LOD project and loaded them in OWLIM. Reasoning was performed to "materialize" the facts that could be inferred from this data.

The overall setup is called Linked Data Semantic Repository (LDSR) to stress out that: (1) Linked Data datasets are loaded together and (2) their semantics is used for inference. LDSR is a unique experiment with LOD data. Although many experiments were performed using LOD data:

- there are no published results of inference over such dataset of a size above 100M statements;
- there are no published results of loading and inference over more than two datasets in a single repository of a size above 10M statements.

⁹ <http://esw.w3.org/topic/SweoIG/TaskForces/CommunityProjects/LinkingOpenData>

Below we discuss the LOD datasets loaded in LDSR.

- **DBPedia**¹⁰ is an RDF dataset derived from Wikipedia through information extraction. It is designed to serve two purposes (i) to provide as full as possible coverage of the factual knowledge that can be extracted with high precision from Wikipedia and (ii) to serve as a hub for the LOD project. To meet the latest requirement, DBPedia provides links to many other datasets in LOD, including GeoNames and Wordnet. Currently, DBPedia consists of almost 270 million explicit statements.
- **GeoNames**¹¹ is a geographic database, which covers most of the significant geographical data categorized into feature classes such as: state, country, provinces, populated place of any size, mountain, river, bridge, facility (e.g. oil fields), etc. Each feature has a designator (type of the feature), geographic coordinates, and relations to other features (e.g. “parent” feature in which the feature is nested). At present, GeoNames database provides information on 6.5 million unique geographic features.
- **UMBEL** (*Upper Mapping and Binding Exchange Layer*) is a lightweight ontology structure which aims at organizing the Web content and data¹². The subject concepts and relationships in UMBEL are derived from OpenSyc. The extracted well-formed class hierarchy consists of about 20 000 classes, ranging from general philosophical notions like `TangibleThing` to very specific classes like `AbaCloth`.
- **WordNet**¹³ is a lexical knowledge base which covers about 150 000 English words. WordNet defines the meanings of English words by grouping nouns, verbs, adjectives, and adverbs into sets of synonyms – the so called synsets. Each synset expresses a distinct concept. Obviously, the words linked to a given synset are synonyms with respect to the meaning expressed by the synset. A word can have multiple meanings, i.e. it can be associated with multiple synsets. The synsets are interlinked by means of semantic and lexical relations (links). The more general terms are associated with less general terms through hyponym-hypernym relations. In LDSR we use the Wordnet RDF/OWL representation¹⁴, provided by the Semantic Web Best Practices and Deployment Working Group of W3C.
- **CIA World Factbook**¹⁵ represents a collection of structured data about most of the countries in the world, including statistical, geographic, political, and other information;
- **Lingvoj**¹⁶ is a dataset presenting multi-lingual descriptions of the most popular human languages; it currently contains information about more than 500 languages.

¹⁰ <http://dbpedia.org/>

¹¹ <http://www.geonames.org/>

¹² <http://www.umbel.org/>

¹³ <http://wordnet.princeton.edu/>

¹⁴ <http://www.w3.org/2006/03/wn/wn20/>

¹⁵ <http://www4.wiwiw.fu-berlin.de/factbook/>

¹⁶ <http://www4.wiwiw.fu-berlin.de/factbook/>



We use ontologies to define the structure and the semantics of the datasets, loaded in LDSR. The following third-party ontologies and schemata are referred to or imported by these ontologies:

- **Dublin Core**¹⁷ is a relatively small but very popular metadata schema. It defines 15 attributes (e.g. author/contributor, data of publication, language, etc.) which can be used in resource description
- **SKOS**¹⁸ (Simple Knowledge Organization System) represents a relatively simple RDFS schema, which allows description of taxonomies of concepts, linked to each other by any sort of subsumption hierarchy. The most important properties defined by SKOS are `skos:broader` and `skos:narrower`, defined as inverse of each other. The subsumption semantics of these relationships is more appropriate for encoding of “topic ontologies” and subjects classifiers as compared to the semantics of `rdfs:subClassOf`. The set-theoretic semantics of sub-class is not applicable for topic taxonomies. For example, while `AfricanLions` is a sub-topic of `Africa`, it is not a valid sub-class relationship.
- **RSS**¹⁹ is an RDF schema designed to enable syndication of machine-readable information about updates from web sites.

DC, SKOS, and RSS ontologies are also loaded in LDSR, because they are the basis for the semantic descriptions of the datasets.

Although, we initially set up LDSR as a test dataset for scalable inference, we also made it publicly available at <http://www.ontotext.com/ldsr/>. The data can be explored through OpenRDF’s Workbench, a web UI which allows query evaluation and graph navigation. SPARQL end-point is also made publicly available.

6.3. Loading and Inference Statistics

The tests were performed using the owl-max ruleset of OWLIM reasoning on top of the LDSR dataset through forward-chaining. This ruleset delivers a combination of RDFS with incomplete OWL Lite, following the approach of Herman ter Horst, [12], to support the semantics of the OWL primitives in a tractable logical fragment. More information about this fragment can be found in section 4.2.1 of [11]. The rules and axioms in the owl-max ruleset are superset of those of language profile OWL-Lepton-I, defined in [11]. Based on our observation of the datasets and the definitions of the related ontologies, we believe that this way we have performed complete inference against LDSR.

¹⁷ <http://purl.org/dc>

¹⁸ www.w3.org/2004/02/skos/

¹⁹ <http://web.resource.org/rss/1.0/spec>

Dataset	Named Graph	Explicit Triples	Inferred after import	RDF nodes after import
Umbel	http://umbel.org/umbel#	3,167,205	56,833	1,230,550
DBpedia (sameAs)	http://dbpedia.org	145,120	278,139	1,414,157
Geonames	http://www.geonames.org/	72,747,880	428,696,785	34,813,153
DBpedia 3.2 core	http://dbpedia.org	280,697,077	38,922,702	100,131,770
lingvoj	http://lingvoj.org	19,692	848,978	100,141,681
Wordnet	http://wordnet.princeton.edu/	1,946,838	8,575,920	100,769,150
CIA Factbook	http://www4.wiwiss.fu-berlin.de/factbook	35,956	291,877	101,005,679
Total		357,844,134	511,522,747	101,005,679

Table 1 Statistics from the loading and inference of the LDSR dataset

6.4. Loading and initial inference against LDSR

The statistics from loading and materialization of the implicit facts is presented in

Dataset	Named Graph	Explicit Triples	Inferred after import	RDF nodes after import
Umbel	http://umbel.org/umbel#	3,167,205	56,833	1,230,550
DBpedia (sameAs)	http://dbpedia.org	145,120	278,139	1,414,157
Geonames	http://www.geonames.org/	72,747,880	428,696,785	34,813,153
DBpedia 3.2 core	http://dbpedia.org	280,697,077	38,922,702	100,131,770
lingvoj	http://lingvoj.org	19,692	848,978	100,141,681
Wordnet	http://wordnet.princeton.edu/	1,946,838	8,575,920	100,769,150
CIA Factbook	http://www4.wiwiss.fu-berlin.de/factbook	35,956	291,877	101,005,679
Total		357,844,134	511,522,747	101,005,679

Table 1. The first column lists the datasets in the order in which they were loaded into the repository. The number of triples listed in column “Explicit triples” refers to the increased number of the statements in BigTREE indices after the dataset has been loaded. Note that some data providers claim that their datasets contain an amount of statements, slightly different from the one presented in the table.

The total sum of the number of explicit and implicit statements after the initial incomplete inference step is 869 million statements. The total time, which the loading and the inference require, is less than 7 hours. The test is performed using a server with the following specifications: 2 x Xeon 5420 CPU (2.5 GHz), 64GB of RAM, 64-bit Linux, 64-bit JDK 1.6, RAID array of 8 SAS drives in RAID 5. We have allotted only 20GB of the RAM to the Java process.

It is not a trivial task to load and perform inference over a large dataset like LDSR. We performed many experiments until we found the weird parts of the datasets and resolved



all the bottlenecks in the architecture. Below we list some problems discovered in the datasets.

- YAGO module of DBpedia includes a questionable class hierarchy and classifications. For instance, it is declared that Pleven (a large Bulgarian city, <http://dbpedia.org/page/Pleven>) is related through `rdf:type` to <http://dbpedia.org/class/yago/Village108672738>. The latter has the label “hamlet” and irrelevant relations to other nodes. After the forward-chaining such “strange” statements generate faulty statements. We decided to remove YAGO from LDSR.
- FOAF ontology delivers implicit knowledge of limited utility while at the same time it increases considerably the reasoning complexity and. We decided not to load FOAF in LDSR
- DBpedia is loaded on several “tranches” for the sake of speed optimisation. This approach allows for deriving a smaller version of LDSR which does not contain the Wikilinks module. While it is a large portion of the DBpedia, Wikilinks serves no other purpose but to define in RDF the hyperlinks which exists in the original Wikipedia; it represents the associations between Wikipedia articles. Such associations have no formal semantics. For instance, there is a Wikilink between Madonna and Bjork, because the article about Madona mentions Bjork, but the nature of the link is entirely unspecified (are they colleagues, friends, enemies, relatives or something else). Such relationships might be useful for context modelling through spreading activation, but they add little value for most of the applications. We decided to exclude Wikilinks module from LDSR.

6.5. owl:sameAs Optimisations

The loading of LDSR benefited greatly from a specific feature of the BigTREE engine, which allows the engine to handle efficiently `owl:sameAs` statements. `owl:sameAs` is a predicate which declares that two different URIs denote one and the same object. Most often, it is used to align the different identifiers of the same real-world entity used in different data-sources. For instance, the URI of Vienna in DBpedia is <http://dbpedia.org/page/Vienna>, while the URI in Geonames is <http://sws.geonames.org/2761369/>. DBpedia contains the following statement

```
(S1) dbpedia:Vienna owl:sameAs geonames:2761369
```

It declares that the two URIs are equivalent.

`owl:sameAs` is probably the most important OWL predicate when it comes to merging data from different data-sources.

Following the formal definition of OWL, whenever two URIs are declared equivalent, all statements which involve one of the URI, should be “replicated” with the other URI. For instance, in Geonames the city of Vienna is defined as part of <http://www.geonames.org/2761367/> (the first-order administrative division in Austria with the same name), which in its turn is part of Austria (<http://www.geonames.org/2782113>):

```
(S2) geonames:2761369 gno:parentFeature geonames:2761367
```



(S3) `geonames:2761367 gno:parentFeature geonames:2782113`

As long as, `gno:parentFeature` is a transitive relationship, in the course of the initial inference, it will be derived that the city of Vienna is also part of Austria:

(S4) `geonames:2761369 gno:parentFeature geonames:2782113`

Due to the semantics of `owl:sameAs`, from (S1) it should be inferred that statements (S2) and (S4) also hold for Vienna, when it is referred with its DBpedia URI:

(S5) `dbpedia:Vienna gno:parentFeature geonames:2761367`

(S6) `dbpedia:Vienna gno:parentFeature geonames:2782113`

These are true statements and when querying RDF data, no matter which one of the equivalent URIs is used in the explicit statements, the same results will be obtained. When we consider that Austria also has equivalent URI in DBpedia, we should also infer that:

(S7) `dbpedia:Vienna gno:parentFeature dbpedia:Austria`

(S8) `dbpedia:Austria gno:parentFeature dbpedia:Austria`

As a result, for a pair of equivalent URIs, a number of new statements will be derived. Probably, there are many concepts with more than two URIs. For instance, Vienna has URI also in UMBEL which is also declared equivalent to the one in DBpedia.

`owl:sameAs` is transitive, reflexive, and symmetric, thus, a set of N equivalent URIs N^2 `owl:sameAs` statements will be generated for each pair.

Obviously, although `owl:sameAs` is useful for interlinking RDF datasets, its semantics causes considerable inflation of the number of the implicit facts which should be considered during inference and query evaluation (either through forward- or through backward-chaining).

To overcome this problem, BigTRREE handles `owl:sameAs` in a specific manner. In TRREE indices, each set of equivalent URIs (equivalence class with respect to `owl:sameAs`) is represented by a single super-node. In this way, BigTRREE does not inflate the indices while it retains the possibility to enumerate all statements which should be inferred using the equivalence. Special care is taken to ensure that this “trick” does not hinder the ability to distinguish explicit from implicit statements.

This optimisation allows BigTRREE to handle efficiently large datasets where `owl:sameAs` is extensively used. In the case of LDSR, this technique allows dealing with more than 6 billion statements at the computational costs required for 870 million statements. Technically, this means that the previous “world record” of BigOWLIM for scalable reasoning: 5.5 billion statements, inferred from LUBM(25 000)²² is improved.

²² <http://esw.w3.org/topic/LargeTripleStores> - a web page of W3C dedicated to publication of scalability records in handling RDF data.

6.6. Ranking and Priming Experiments

Little public information is available on scalable experiments with hybrid reasoning systems. The most relevant figures come from an experiment with SNIF-ACT²⁴ (based on the ACT-R system). The experiment is related to modelling of word co-occurrence derived from the TIPSTER corpus in order to support users in WWW exploration. The scalability figures, reported there are as follows: *“This database contained statistics relevant to setting the base-level activations of 200 million word tokens and the inter-word association strengths of 55 million word pairs”*. Although it is hard to interpret these results this is the only available data on scalability.

Our goal at this stage of evaluation was to acquire first observations on the performance of the new components and to make sure that the implementations allow efficient experiments with large datasets.

We measured the performance of SA and PR components of DualRDF over the LDSR dataset (see section 6.3 for statistics). The experiments were executed on the hardware setup described in section 6.4.

PageRank

The computation of the PageRank values for the DualRDF repository took **458s** of which **260s** were spent reading the RDF graph from disk-based structures into memory and **185s** were spent in PageRank iterations. The configuration parameters required that the error is less than 0.001 (PREpsilon = 0.001) which in that particular case made the algorithm iterate 17 times. Thus the speed of PageRank can be estimated as **10.88sec/iteration**.

Spreading Activation

For the following experiments we configured an initial activity of 0.66 and activity threshold of 0.80. We chose the following 3 URIs from the RDF graph as “activation seed”:

```
http://dbpedia.org/resource/London  
http://dbpedia.org/resource/Paris  
http://dbpedia.org/resource/Berlin
```

For brevity we refer to them as London, Paris, and Berlin respectively.

Sample Activation

For each of the chosen RDF graph nodes we ran activation spreading and got the following results:

Table 2 Single-Node Activation Results

²⁴ <http://www2.parc.com/ist/groups/uir/projects/snif-act/Model.html>

Initially active nodes	Decay	Firing Threshold	Fired Entities	Time (sec)
London	0.85	0.25	54,032	26
Paris	0.85	0.25	96,275	27
Berlin	0.85	0.25	25,721	19

The results give an impression of the amount of concepts related to the chosen ones.

Tuning the Decay

We supposed that by increasing the decay factor larger activity will be propagated which will result in larger number of fired graph nodes. In this experiment we choose one of the 3 sample nodes (Berlin) and spread activation starting from it. The following table proves that our guess was correct:

Table 3 Results in Dependence of the Decay Factor

Initially active nodes	Decay	Firing Threshold	Fired Entities	Time
Berlin	0.75	0.25	25,720	10
Berlin	0.50	0.25	7212	9
Berlin	0.25	0.25	7193	7

Tuning the Firing Threshold

When the other parameters of the SA component are constant, the firing threshold will be responsible for reducing/increasing the activity amount. This is evident from the results in:

Table 4 Results in Dependence to the Firing Threshold

Initially active nodes	Decay	Firing Threshold	Fired Entities	Time
London	0.50	0.20	54,030	8
London	0.50	0.40	10,959	7
London	0.50	0.60	9,700	6

Initial Activation of Several Nodes

Finally, we experimented with activation seed containing all the tree entities and observe how the results change as the decay factor is being reduced.

Table 5 Multiple-Node Activation Results



Initially active nodes	Decay	Firing Threshold	Fired Entities	Time
London,Paris,Berlin	0.75	0.25	594,938	149
London,Paris,Berlin	0.50	0.25	475,043	32
London,Paris,Berlin	0.25	0.25	368,483	25

We observe that a seed containing multiple, not directly interrelated entities as a seed, given the same parameter (as in table 3), generates much more intensive activation processes. Having that in real-world scenarios, most often the activation seeds will contain multiple nodes, this means, that unconstrained spreading activation is not feasible.

Overall Performance

In order to measure the overall performance of the activity spreading we activated a single node and set decay to 1 and all thresholds to 0 in order to propagate activity unconditionally. The SA component managed to spread activity over 75059 entities in 34 seconds. This amounts to an approximate speed of 2207 fired entities per second.

7. Conclusion

We presented the implementation approach behind couple of components related to selection in WP2. DualRDF component provides basic infrastructure for priming of RDF graphs, while PageRankRDF component allows for ranking of nodes in the graph. On the implementation side, we also deliver SSelector plug-in that provides a straight-forward way for usage of DualRDF for selection in LarKC pipelines.

The essential objective behind these components is to enable further experiments with different cognitively inspired selection approaches within LarKC. While DualRDF provides scalable implementation of RDF graph priming, the interesting research questions related to finding appropriate weighting schema for RDF statements and parameters for spreading activation, which meet the requirements of specific datasets and usage scenarios, are still to be investigated during the second year of the project. PageRank implementation is implemented again as a basis for experiments in weighting and ranking.

To prepare proper ground for selection and ranking experiments, we set up the Linked Data Semantic Repository (LDSR) – a selection of few of the most central datasets in the Linking Open Data project, which represents a “reasonable view” to the emerging web of linked data. LDSR has total size of about 370 million explicit statements, which were loaded in LarKC’s Data Layer. Light-weight reasoning was applied to materialize additional 512 million statements, which can be inferred from them after “practically complete” forward chaining. Entailments resulting from `owl:sameAs` mappings are subject of specific optimization –those are not materialized and added to the indices, however, they are still available for all sorts of retrieval, inference and query evaluation. Inference was performed with respect to OWLIM’s owl-max ruleset, which represents an extension of OWL-Lepton-I, [11]. Being a side “side effect” of the work on the selection components, LDSR appears to be also the largest body of common sense knowledge that anyway took the challenge to reason against.

We conducted initial experiments to make sure that DualRDF and PageRankRDF are operational on top of LDSR and to collect first impressions about their performance. We find the results from these experiments encouraging: it takes just 10 seconds to perform one iteration of PageRank or 3 minutes to compute the ranks of the 100 million nodes in LDSR in the course of 17 iterations. The performance of the spreading activation tasks varies enormously in dependence of the various parameters of the process. For instance, it takes as little as 7 seconds to activate about 7 thousand nodes after spreading of activation from resource <http://dbpedia.org/resource/Berlin> with decay factor 0.25. And it can take as much as 3 minutes to activate 0.6 million nodes starting with London, Paris, and Berlin (resources in DBPedia) as activation seed, and given decay factor of 0.85.

There are multiple directions in which the components delivered here can be developed, tuned and optimized. For instance, one can use special purpose hardware (e.g. FPGA or CUDA) to boost their performance; the underlying data structures can be optimized; there is also space for optimization for better exploitation of standard multi-core architectures. Still, the main driver for the future development of these components will be the feedback from the use cases of the project and the requirements of the reasoning plug-ins, which are using them. Further, there is still more groundwork to be performed in WP2 with respect to tuning cognitively inspired selection plug-ins.

8. References

- [1] Berners-Lee, T. (2006). *Design Issues: Linked Data*.
- [2] Brickley, D., Guha, R.V, eds.: *Resource Description Framework (RDF) Schemas*. W3C Recommendation. <http://www.w3.org/TR/2004/REC-rdf-schema-20040210/>, (2004)
- [3] Brin, S., Page, L.: *The Anatomy of a Large-Scale Hypertextual Web Search Engine*. <http://infolab.stanford.edu/~backrub/google.html>, (1998)
- [4] Carroll, J., Bizer, Ch., Hayes, P., Stickler, P.: *Named graphs, Provenance and Trust*, <http://www2005.org/cdrom/docs/p613.pdf>, (2005)
- [5] Collins, M., Loftus, E.: *A spreading-activation theory of semantic processing*. *Psychological Review*. 1975 Nov. Vol. 82(6), pp. 407-428. (1975)
- [6] Cunningham, H., Roberts, A., Li, Y., Kiryakov, A., Schooler, L., Quesada, J., Neth, H., Della Valle, E., Braga, D., Zhong, N.: *Deliverable D2.1.1 Selection and Retrieval Methods*. LarkC project deliverable. (2008)
- [7] Dean, M; Schreiber, G. – editors; Bechhofer, S; van Harmelen, F; Hendler, J; Horrocks, I.; McGuinness, D. L; Patel-Schneider, P. F.; Stein, L. A. (2004). *OWL Web Ontology Language Reference*. W3C Recommendation, 10 Feb. 2004. <http://www.w3.org/TR/owl-ref/>, (2004)
- [8] Ding, Li., Finin, T., Peng, Y., da Silva, P. , McGuinness, D.: *Tracking RDF Graph Provenance using RDF Molecules*. http://ebiquity.umbc.edu/file_directory/papers/178.pdf, (2005)
- [9] Fensel D., van Harmelen, F., Andersson, B., Brennan, P., Cunningham, H., Valle, E. D., Fischer, F., Huang, Z., Kiryakov, A., Lee, T. K., School, L., Tresp, V., Wesner, S., Witbrock, M., Zhong, N.: *Towards larkc: a platform for web-scale reasoning*. In *Proceedings of the IEEE International Conference on Semantic Computing (ICSC 2008)*, Santa Clara, USA, (2008)
- [10] Fensel, D., van Harmelen, F.: *Unifying reasoning and search to web scale*. *IEEE Internet Computing* 11(2) (2007) 9695
- [11] Fischer, F; Keller, U; Kiryakov, A; Huang, Z; Momtchev, V; Simperl, E.: *Initial Knowledge Representation Formalism*. LarkC project deliverable D1.1.3, (2008).
- [12] ter Horst, H. J.: *Combining RDF and Part of OWL with Rules: Semantics, Decidability, Complexity*. In: *Proc. of ISWC 2005*, Galway, Ireland, November 6-10, 2005. LNCS 3729, pp. 668-684.
- [13] Kerrigan, M., Bradesko, L., Fortuna B.: *Deliverable D5.2.1 Rapid Prototype of the LarkC*. (January 30, 2009)
- [14] Li, Y., Cunningham, H., Roberts, A., Kiryakov, A., Momtchev, V., Greenwood, M., Aswani, N., Damjanovic, D.: *Selection Components (report accompanying two software deliverables)*. LarkC project deliverable D2.2.1, 2.5.1, (2009)
- [15] Manola F., Miller, E. (eds), *RDF Primer*. W3C Recommendation 10 Feb 2004, <http://www.w3.org/TR/REC-rdf-syntax/>



- [16] Ontotext Lab. *OWLIM – Pragmatic OWL Semantic Repository. Presentation.* <http://www.ontotext.com/owlim/OWLIMPres.pdf>, (as of 18 Jun, 2008).
- [17] Ontotext Lab. *SwiftOWLIM. System Documentation.* <http://www.ontotext.com/owlim/OWLIMSysDoc.pdf>, (September 10, 2007)
- [18] Orud'hommeaux, E., Seaborne, A.: *SPARQL Query Language for RDF. W3C Recommendation*, 15 Jan, 2008 <http://www.w3.org/TR/rdf-sparql-query/>
- [19] Schooler, L., Zhou, H., Zhong, N., Qin, Y., Shengfu, L., Yao, Y., Gao, Y.: *Cognitive Memories Components (v1)*. LarkC project deliverable D2.3.1, (2009)
- [20] Velkov, R., Ognyanoff, D., Kiryakov, A.: *Open-Domain Incomplete Reasoner*. RASCALLI project deliverable D3b, (2009)