



## **LarKC**

*The Large Knowledge Collider*

*a platform for large scale integrated reasoning and Web-search*

**FP7 – 215535**

---

# **D2.2.1, 2.5.1 Month 12 Selection Components (report accompanying two software deliverables)**

---

**Coordinator: H. Cunningham**

**With contributions from: Y. Li, M. Greenwood, A. Kiryakov,  
V. Momtchev, I. Peikov, H. Cunningham, A. Roberts, N.  
Aswani, D. Damljanovic**

**Quality Assessor: J. McKay**

**Quality Controller: A. Roberts**

Document Identifier:	LarKC/2008/D2.2.1, 2.5.1/V1.5
Class Deliverable:	LarKC EU-IST-2008-215535
Version:	1.5
Date:	September 4, 2009
State:	final
Distribution:	public



## EXECUTIVE SUMMARY

This report accompanies the Month 12 software deliveries D2.2.1 (baseline components v1) and D2.5.1 (geometrical semantics components v1). Since Month 6 (with reference to D2.1.1 Chapters 2 and 5) we have changed our model of selection for LarKC in four main ways: dropped our initial simplifying assumption regarding the availability of annotated text corpora linked to the triple space; developed a more sophisticated approach to indexing terms based on RDF molecules; responded positively to the requirements set out by the LarKC Carcinogenesis Research use case; and refined our understanding of geometrical approaches to RDF selection. We describe five new LarKC plugins, delivered as D2.2.1 and D2.5.1.



## DOCUMENT INFORMATION

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

<b>Deliverable</b>	<b>Number</b>	2.2.1, 2.5.1	<b>Title</b>	Month 12 Selection Components (report accompanying two software deliverables)
<b>Work Package</b>	<b>Number</b>	2	<b>Title</b>	Selection and Retrieval

<b>Date of Delivery</b>	<b>Contractual</b>	M12	<b>Actual</b>	31-Mar-09
<b>Status</b>	1.5		final <input checked="" type="checkbox"/>	
<b>Nature</b>	prototype <input checked="" type="checkbox"/> report <input type="checkbox"/> dissemination <input type="checkbox"/>			
<b>Dissemination Level</b>	public <input checked="" type="checkbox"/> consortium <input type="checkbox"/>			

<b>Authors (Partner)</b>	University of Sheffield, Ontotext			
<b>Resp. Author</b>	H. Cunningham		<b>E-mail</b>	hamish@dcs.shef.ac.uk
	<b>Partner</b>	University of Sheffield	<b>Phone</b>	+44 114 222 1891












<b>Abstract (for dissemination)</b>	This report accompanies the Month 12 software deliveries D2.2.1 (baseline components v1) and D2.5.1 (geometrical semantics components v1). Since Month 6 (with reference to D2.1.1 Chapters 2 and 5) we have changed our model of selection for LarKC in four main ways: dropped our initial simplifying assumption regarding the availability of annotated text corpora linked to the triple space; developed a more sophisticated approach to indexing terms based on RDF molecules; responded positively to the requirements set out by the LarKC Carcinogenesis Research use case; and refined our understanding of geometrical approaches to RDF selection. We describe five new LarKC plugins, delivered as D2.2.1 and D2.5.1.
<b>Keywords</b>	Selection, baseline, geometrical methods, Vector space model, WP2



<b>Version Log</b>			
<b>Issue Date</b>	<b>Rev No.</b>	<b>Author</b>	<b>Change</b>
10/Feb/2009	1	Yaoyong Li	0.1 – the first draft
16/Feb/2009	2	Yaoyong Li	0.2 – added an updated version of the analysis on geometric methods
20/Feb/2009	3	Hamish Cunningham	0.3 – restructured to cover all the M12 plugins
21/Feb/2009	4	Atanas Kiryakov	0.4 – discussion of molecular approach
22/Feb/2009	5	Vassil Momtchev	0.5 – LarKC data layer
22/Feb/2009	6	Hamish Cunningham	0.6 – editorial
09/March/2009	7	Angus Roberts	0.7 – new plugins
09/Feb/2009	8	Angus Roberts	0.8 – new template version
10/Mar/2009	9	Angus Roberts	0.9 – rearrange text, tidy, link
10/Mar/2009	10	Angus Roberts	1.0 – editorial
17/Mar/2009	11	Angus Roberts	1.1 – minor changes to plugin descriptions
17/Mar/2009	12	Angus Roberts	1.2 – add a conclusion
17/Mar/2009	13	Angus Roberts	1.3 – Address QC and other comments
25/Aug/2009	14	Yaoyong Li	1.4 – Address comments from review report
04/Sep/2009	15	Danica Damljanovic	1.5 – editorial



## 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 Email: 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 RE- SPONSABILITA LIMITATA Milano, Italy Email: emanuele.dellavalle@cefriel.it
CYCROP, RAZISKOVANJE IN EKSPERI- MENTALNI RAZVOJ D.O.O.		Michael Witbrock CYCROP, RAZISKOVANJE IN EKSPERIMEN- TALNI 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 AD		Atanas Kiryakov, Ontotext Lab, Sofia, Bulgaria Email: naso@ontotext.com
SALTLUX INC.		Kono Kim SALTLUX INC Seoul, Korea Email: kono@saltlux.com
SIEMENS AKTIENGESELLSCHAFT		Dr. Volker Tresp SIEMENS AKTIENGESELLSCHAFT Muenchen, Germany Email: 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
VRIJE UNIVERSITEIT AMSTERDAM		Prof. Dr. Frank van Harmelen, VRIJE UNIVERSITEIT AMSTERDAM Amsterdam, Netherlands Email: Frank.van.Harmelen@cs.vu.nl
THE INTERNATIONAL WIC INSTI- TUTE, BEIJING UNIVERSITY OF TECHNOLOGY		Prof. Dr. Ning Zhong, THE INTERNATIONAL WIC INSTITUTE Mabeshi, Japan Email: zhong@maebashi-it.ac.jp
INTERNATIONAL AGENCY FOR RE- SEARCH ON CANCER		Dr. Paul Brennan, INTERNATIONAL AGENCY FOR RE- SEARCH ON CANCER Lyon, France Email: brennan@iarc.fr
INFORMATION RETRIEVAL FACILITY		Dr. John Tait, Dr. Paul Brennan, INFORMATION RETRIEVAL FACILITY Vienna, Austria Email: john.tait@ir-facility.org



# TABLE OF CONTENTS

LIST OF ACRONYMS	7
1 INTRODUCTION: AN EVOLVING MODEL OF SELECTION	8
1.1 RDF Molecules, Full Text Search and Keyword Selection . . . . .	9
2 MONTH 12 PLUGINS	11
2.1 Introduction . . . . .	11
2.2 Baseline and Key-Phrase Selectors . . . . .	11
2.2.1 Implementation Dependencies and Configuration . . . . .	12
2.2.2 Baseline Selection Plugin Usage . . . . .	12
2.2.3 Key-Phrase Selection Plugin . . . . .	13
2.3 Prior Knowledge Selector . . . . .	13
2.4 Ranked Selector . . . . .	14
2.5 Semantic Annotation Transformer . . . . .	15
3 FUTURE PLUGINS	16
3.1 Introduction . . . . .	16
3.2 Full Text Search and Keyword Selection in RDF Graphs . . . . .	16
3.3 IR-style Selector . . . . .	17
3.3.1 Base-line Selection Through SPARQL evaluation . . . . .	18
3.3.2 Design considerations . . . . .	18
3.4 IR queries over RDF graphs . . . . .	18
4 GEOMETRIC METHODS FOR SELECTION	20
4.1 Keywords, Vector Spaces and Selection . . . . .	20
4.2 Semantic Space Model for RDF Triple Selection . . . . .	21
4.3 Related work . . . . .	23
4.4 Efficiency Issue of the Methods . . . . .	24
5 CONCLUSIONS	25
REFERENCES	25



## LIST OF ACRONYMS

<b>B FDP</b>	Bayesian False Discovery Probability
<b>BLS</b>	Baseline Selector
<b>CORE</b>	Ontotext CO-REference engine
<b>FTS</b>	Full Text Search
<b>GATE</b>	General Architecture for Text Engineering
<b>GWAS</b>	Genome Wide Association Study
<b>IR</b>	Information Retrieval
<b>KB</b>	Knowledge Base
<b>KIM</b>	Knowledge and Information Platform
<b>KPS</b>	Keyphrase Selector
<b>OWLIM</b>	OWL In Memory
<b>RDF</b>	Resource Description Framework
<b>SMC</b>	Set of Molecule Centers
<b>SNP</b>	Single Nucleotide Polymorphism
<b>SPARQL</b>	SPARQL Protocol And RDF Query Language
<b>TFIDF</b>	Term Frequency, Inverse Document Frequency
<b>URI</b>	Uniform Resource Identifier
<b>URL</b>	Uniform Resource Locator
<b>VSM</b>	Vector Space Model



## 1. Introduction: an Evolving Model of Selection

This report accompanies the Month 12 software deliveries D2.2.1 (baseline components v1) and D2.5.1 (geometrical semantics components v1). At Month 12 we are delivering the following LarKC plugins:

**Baseline Selector** Uses simple full-text search methods to extract parts of the dataset based on text tokens found in the query. It acts as a baseline naive implementation of selection.

**Key Phrase Selector** Similar to the Baseline selector, it adapts selection criteria based only on characteristic key-phrases, instead of any text tokens appearing in the dataset.

**Prior Knowledge Selector** Given a set of ontology instance identifiers, a triple store repository, and some knowledge selection criteria, returns a subset of the repository that further describes the identifiers. This plugin is derived from a use case to prior knowledge about gene probes (SNPs) in a LarKC use case, but is generalised beyond this use case.

**Ranked Selector** Given a set of ontology instance identifiers and prior knowledge about the instances, rank these instances according to some model. In this case, we use Bayesian False Discovery Probability (BFDP), selecting and returning the top  $n$  ranked objects. Again, this is a general solution influenced by the specific LarKC use case of ranking gene probes (SNPs) based on their BFDP.

**Semantic Annotation Transformer** This is a retrieval plugin that wraps an arbitrary semantic annotation pipeline from GATE. It has been written in response to the need for retrieving semantically annotations from text in several LarKC use case scenarios.

Since Month 6 (with reference to D2.1.1 Chapters 2 and 5) we have changed our model of selection for LarKC in four main ways.

First, we have dropped our initial simplifying assumption regarding the availability of annotated text corpora linked to the triple space. This worked nicely for our initial selection plugin experiments (see deliverable D2.1.1 for descriptive text and D2.1.2 for example code), but did not generalise to most of the use cases for selection that the consortium are now working with.

Second, whereas we began with a straightforward derivation of indexing terms from node identifiers and URIs we have developed a more sophisticated approach based on RDF molecules (introduced in section 1.1). Those are already used in the baseline and key-phrase plugins (section 2.2), as a simple approach for dealing with text and granularity in RDF graphs. We have also started developing more sophisticated models, for text-based relevance and selection in RDF graphs, that will form the basis of plugins in delivered in the future, as discussed in section 3.

Third, we have responded positively to the requirements set out by the LarKC Carcinogenesis Research use case, as set out in D7b.1.1a. The selection scenarios outlined in the D7b.1.1a requirements document include models not considered in our previous deliverables. These include selection of triples for multiple data points, calculating metrics on triples according to statistical models, and selecting based on



ranking according to these metrics. We therefore introduce, in Chapter 2, several new selection and retrieval plugins. While these are generic in their design, they are influenced by the needs of the Carcinogenesis Research use case.

Fourth, we have refined our understanding of geometrical approaches to RDF selection and performed some initial experiments. Our current feeling is that the quantum geometries (based on Hilbert spaces) will be very difficult to scale to the data sizes that we are confronted with in LarKC, and we are also not convinced that their accuracy will be radically different from the other approaches that we are working on. We include in this report a discussion of geometrical methods (including quantum geometries – chapter 4) but for the time being we have de-emphasised further development of the quantum methods. This is explained further in chapter 4.

This report provides background for these four points, describes the plugins delivered as a result of this approach, and points to future developments in the workpackage. The report is structured as follows. The next chapter describes the four plugins delivered as D2.2.1 and D2.5.1. This is followed by a chapter describing planned future work in the area of selection based on RDF molecules. Last, we provide a chapter discussing our current thinking on geometrical approaches to selection.

## 1.1 RDF Molecules, Full Text Search and Keyword Selection

In an RDF graph, we can consider the collection of triples describing a specific, resource node (i.e. a resource identified with a URI, not a blank node or literal). This collection is referred to as a molecule. For a node  $S$ , this collection represents:

- all statements where  $S$  is the subject (i.e. outgoing arcs) - this is called the immediate neighbourhood
- in the cases where a statement from those above has a blank node as an object (e.g.  $\langle S, \dots, \textit{blank - node} \rangle$ ), its immediate neighbourhood is also added to the molecule, so, the molecule expands recursively over blank nodes.

In other words, the molecule of node  $S$  is the part of the graph that you can reach following all paths in the graph, starting from  $S$ , until you reach non-blank nodes. The notion is sensible because blank nodes are auxiliary nodes for things that are hard to express in triples. E.g.

```
Hamish hasName "..."  
Hamish hasBirthDate "..."  
Hamish leavesIn http://.../Sheffield  
Hamish hasCar _:C  
_:C hasPlateNumber "..."  
_:C model http://.../Octavia  
_:C hasInsurance _:I  
_:I validFrom "..."  
_:I validUntil "..."
```

In this case  $_:C$  is a blank node, used for the description of the car of Hamish. It is only important as part of the description of Hamish: no one cares to think of separate unique name of this particular vehicle, because it is not likely to be referred often,



without mentioning Hamish itself. The same holds for the insurance policy of the car. In short, following the above algorithm, all the above statements will be part of the RDF molecule about Hamish. Still, the statements describing Sheffield and Octavia will not be part of it.

The concept of an RDF molecule was proposed in [8] as a minimal component in a loss-less decomposition of an RDF graph. RDF molecules provide an intermediate level of granularity between RDF graphs and triples. The paper also provides an automatic algorithm to automatically produce all of the RDF molecules from an RDF graph. [20] converted the RDF graph into RDF molecules for scale-out distributed querying and reasoning over large scale RDF triple stores. [20] also extended the original RDF molecule definition to include hierarchy and ordering of the RDF molecules.



## 2. Month 12 Plugins

### 2.1 Introduction

This Chapter describes the four plugins delivered as D2.2.1 and D2.5.1. Each section in this chapter gives a brief description of each plugin. The plugins themselves are published in the LarKC source code repository, together with full documentation (as javadoc). They are published as follow:

Plugin	Availability
Baseline Selector	<a href="https://svn.gforge.hlr.de/svn/larkc/trunk/platform_v04/src/plugins/eu/larkc/plugin/select/sparqlkeyword">https://svn.gforge.hlr.de/svn/larkc/trunk/platform_v04/src/plugins/eu/larkc/plugin/select/sparqlkeyword</a>
Key Phrase Selector	<a href="https://svn.gforge.hlr.de/svn/larkc/trunk/platform_v04/src/plugins/eu/larkc/plugin/select/sparqlkeyword">https://svn.gforge.hlr.de/svn/larkc/trunk/platform_v04/src/plugins/eu/larkc/plugin/select/sparqlkeyword</a>
Prior Knowledge Selector	<a href="https://svn.gforge.hlr.de/svn/larkc/trunk/modules/WP2Plugins/usfd/prior-knowledge-selector">https://svn.gforge.hlr.de/svn/larkc/trunk/modules/WP2Plugins/usfd/prior-knowledge-selector</a>
Ranked Selector	<a href="https://svn.gforge.hlr.de/svn/larkc/trunk/modules/WP2Plugins/usfd/ranked-selector">https://svn.gforge.hlr.de/svn/larkc/trunk/modules/WP2Plugins/usfd/ranked-selector</a>
Semantic Annotation Transformer	<a href="https://svn.gforge.hlr.de/svn/larkc/trunk/modules/WP2Plugins/usfd/semantic-annotation-transformer">https://svn.gforge.hlr.de/svn/larkc/trunk/modules/WP2Plugins/usfd/semantic-annotation-transformer</a>

### 2.2 Baseline and Key-Phrase Selectors

Since Month 6 we have completely revised and reimplemented the base-line selection (BLS) plugin, which now returns part of a Dataset, based on full-text search (FTS) in the graph. It works as follows:

- keywords are extracted from the SPARQL queries;
- FTS is performed in the dataset to find relevant literals;
- the result is a tripliset composed of all RDF molecules (see section 1.1) which contain these keywords.

At the second step the BLS plugin performs a query conceptually equivalent to the following one:

```
select ?s
where {
  ?s ?p ?l.
  ?l contains "kw1 kw2 ... kWn"}
```

The list of results is processed as follows:

- if the result is an URI, it is added to the set of molecule centers SMC;



- if the result is a blank node, it is the inbound arcs (statements where it is an object) are followed backwards until one or more URIs are reached; the latter are added to SMC.

The result of the selection are the molecules of the URIs from set SMC.

### 2.2.1 Implementation Dependencies and Configuration

As the BLS plugin (BaseLineFTSelector) performs a FTS query, its performance is dependent on the efficiency of the evaluation of such queries by the underlying engine. The implementation of the BLS plugin depends on the FTS of BigTRREE/BigOWLIM version 3.1.x (there is no specific support for full-text indexing and search in the previous versions). In order to turn that feature on for a specific repository, the following parameters should be supplied upon initialization:

```
ftsIndexPolicy = onStartup
ftsLiteralsOnly = false
```

The second parameter is not mandatory but it would guarantee that FTS will be executed against all RDF nodes in the repository (not only literals) which might possibly extend the selected tripliset e.g. over URIs that have relevant local names.

A possible optimization that affects that baseline selector plugin (especially on large datasets using large variety of predicates) is the so-called "predicate lists" optimization available in BigOWLIM versions 3.1.a5 and above. In order to turn it on the following parameter should be added to the configuration:

```
enablePredicateList = true
```

This optimization, switches on the creation of usage of an additional index, which requires extra loading time (or initialization time, if the feature is switched on for a full repository). It also requires extra memory, which is usually in the range of 30% on top of BigOWLIM's regular requirements, but it pays off during baseline selector plugin normal operation as well as in any queries where the predicate is not bound.

The current implementation of the baseline selector was extensively tested on RDF/OWL representation of Wordnet, where selection is performed in less than 1 second.

### 2.2.2 Baseline Selection Plugin Usage

The baseline selector plugin shares the common Selector and Plugin interfaces and its most common usage scenario contains the following three steps:

- plugin is initialized;
- input query is supplied via the setInputQuery() method;
- a tripliset is retrieved by calling the select() method.

The following code demonstrates these steps:



```
BaseLineFTSelector s = new BaseLineFTSelector();
s.setInputQuery(
    new SPARQLQueryImpl(
        "SELECT ?s ?p ?o WHERE { ?s ?p ?o FILTER(?s = 'hello world') }"
    ));
TripleSet ts = s.select(null, null);
```

### 2.2.3 Key-Phrase Selection Plugin

The Key-Phrase Selector (KPS) plug-in is built on top of the baseline selector (BLS) plug-in. The essential difference is in the way in which the relevancy of nodes and literals is determined. In the BLS, a literal is considered relevant if it matches (via FTS) a keyword extracted from the input SPARQL query. The KPS plugin on the other hand matches the keywords from the input query against the key-phrases extracted from the nodes' molecules. Assuming that all nodes were annotated during a preprocessing phase with their key-phrases (i.e. by adding an additional  $\langle S, \text{hasKeyPhrases}, "phrase1, phrase2, phrase3..." \rangle$  triple statement to the repository) the nodes relevant to some keyword might be extracted using the following SPARQL select query:

```
select ?s
where { ?s hasKeyPhrases ?l.
       ?l contains "keyword"}
```

The key-phrase annotation preprocessing phrase is implemented as a separate Java utility program (`eu.larkc.plugin.select.sparqlkeyword.annotator.KeyPhraseAnnotator`) which uses TF.IDF-based GATE plug-in in order to retrieve the key-phrases from the text extracted from node's molecule.

The TF.IDF algorithm is first trained over a fixed amount of molecules in the repository in order to get an adequate model of the inverse document frequencies. These frequencies are then used to find for each node's molecule the key-phrases having highest TF.IDF score. Those key-phrases are concatenated together in a string literal which is then attached to the molecule's central node, as discussed above. This preprocessing runs in multiple threads to achieve better performance on multiple CPUs (the number of threads is determined dynamically based on the number of CPUs or CPU-cores available).

Discussion on the motivation and possible future developments of the KPS plugin can be found in chapter 3.

## 2.3 Prior Knowledge Selector

This plugin takes a list of ontology instance identifiers, a repository, and some knowledge selection criteria, and returns a subset of the repository that further describes the identifiers. The Ontology instance identifiers are provided as a `KeywordQuery` to the `setInputQuery` method. The repository and the knowledge selection criteria are currently provided in a base class wrapped by the plugin, although these could easily be exposed in later versions, were this to be supported by the LarKC API.



The plugin has been directly influenced by the requirements of a scenario from the LarKC WP7b use case, which supports carcinogenesis research. In this scenario, prior knowledge for the relationship between gene probes (SNPs) and a given disease are used to provide greater discrimination in the analysis of wet lab experiments. The scenario is fully described in deliverable D7b.1.1a.

Although the plugin has been designed to support this specific use case, we believe that the problem tackled is more widely relevant, and have therefore generalised the base design of the plugin. We have created a hierarchy of three base classes for wrapping as plugins. The first two of these are abstractions, and the third is a concrete implementation specific to the gene-disease association study problem. It is this latter that we have wrapped as a LarKC selection plugin. However, by extending the abstractions and implementing the LarKC selection plugin interface over this extension, a plugin implementer may create new background knowledge plugins. The three base classes are:

1. Abstract BackgroundKnowledge, represents background knowledge of any arbitrary form.
2. Abstract SPARQLBackgroundKnowledge represents background knowledge obtained from SPARQL queries. The knowledge is obtained in multiple steps using separate queries. Assuming that there are  $n$  queries then the first  $n-1$  queries retrieve meta-knowledge that may be needed either by downstream plugins, or for binding in the following queries for obtaining the main background knowledge. The final query then retrieves the full set of background knowledge.
3. Concrete SNPGeneRIFKnowledge represents background knowledge required for finding prior knowledge of a given gene probe in the context of a gene-disease association study. The background knowledge is retrieved from the LarKC data layer using SPARQL queries.

## 2.4 Ranked Selector

This plugin takes a triple set of ontology instance identifiers, and prior knowledge about the identified objects, and ranks those objects within a context, according to some metric, selecting and returning the top  $n$  ranked objects. Instance identifiers and context are provided as a TriplePatternSetQuery. This is a Set of the existing LarKC TriplePatternQuery. Each TriplePatternQuery in the TriplePatternSetQuery defines a triple with a subject of the instance identifier, and an object of the context against which the instance will be scored and ranked.

Again, this plugin has been directly influenced by the requirements of the carcinogenesis research scenario briefly described above and in detail in deliverable D7b.1.1a.

Although the plugin has been designed to support this specific use case, we believe that the problem tackled is more widely relevant, and have therefore generalised the design of the plugin. As before, we have created a hierarchy of two base classes, the first being an abstraction. The second base class implements a specific metric used in gene-disease association studies, Bayesian False Discovery Probability, BFDP. We believe that the BFDP base class and its plugin implementation are not specific to the gene-disease scenario, and could be reused elsewhere. In addition, the abstract base



class could be extended to provide further score style classes, implementing the LarKC plugin interface over these extensions, as we have done here. The two base classes are:

1. Abstract base class `Score`. Scores and ranks a list of instances given some background knowledge on those instances, and a context. Inputs are instances with background knowledge from a `BackgroundKnowledge` plugin (see above), and a context.
2. Concrete base class `BFDPSScore` specialises the above to give a class that takes output from a `SPARQLBackgroundKnowledge` and a set of keywords. The `SPARQLBackgroundKnowledge` output is searched for the keywords, and instances scored and ranked using a Bayesian model.

## 2.5 Semantic Annotation Transformer

The previous plugin introduced the idea of ranking ontology instances according to the context, for instance `BFDPSScore` ranks are based on the occurrences of keywords in text associated with instances. This next plugin allows us to take this idea further, by providing a layer of semantic annotation on the text associated with an instance, and allowing scores to be assigned based on semantic retrieval from those texts.

To this future end we have therefore provided a **Transformer** plugin that wraps an arbitrary semantic annotation pipeline from GATE, a leading toolkit for text mining.

(**Note** that paths to the wrapped pipeline and documents to be transformed are currently hard coded in this version, while we await changes scheduled for the next version of the LarKC platform, that will enable this information either to be passed as a `Context` object, or wrapped behind a LarKC `NaturalLanguageDocument` interface).

Semantic annotation can be carried out in manual, automatic, and semi-automatic fashion and each approach has its advantages and disadvantages [5]. The GATE software that we have developed implements all the three approaches:

- GATE provides GUI and other facilities such as schema and searching function for manual annotation.
- One can write JAPE rules in GATE to automatically annotate text.
- One can use machine learning facilities in GATE to automatically annotate text.
- The combination of manual and automatic annotation processes leads to semi-automatic annotations.

Accuracy of semantic annotation clearly affects the accuracy of the applications using those annotations. Accuracy of semantic annotation is dependent upon the annotation type. In general, if one type of annotation is well-defined and has very distinguishing features, it can be annotated using automatic or semi-automatic approach, with accuracy as high as 90–100% [1, 17]. For more complex annotations, such as those appearing in a complicated context, an acceptable accuracy is reached through several iterations which require careful authoring of JAPE rules. Also, with sufficient training data it is possible to use machine learning algorithms in an effective way, in which case the annotation accuracy can reach 75–90% [17, 18].



### 3. Future Plugins

#### 3.1 Introduction

The previous Chapter described the existing plugins delivered for this iteration of LarKC plugin development. We next describe a novel selection method, based on RDF molecules.

#### 3.2 Full Text Search and Keyword Selection in RDF Graphs

We put forward the hypothesis that RDF molecules (see section 1.1), are the lowest granularity chunks of RDF graph which can be considered separately in a search process. We have to consider how we might select nodes, based on the text selection of their molecules. We do this so that we may use typical IR metrics such as TF\*IDF for the selection. The intuition is that TF\*IDF only makes sense in bodies of text/data that are not too small, i.e. large enough for meaningful frequencies to be calculated. TF\*IDF would not make sense on top of a single triple. It makes some sense when the object of the triple is a long text literal, but again, aggregating all such values from the molecule is likely to produce key-phrases which are more representative for the corresponding node. In other words, the intuition is that an RDF molecule better matches the notion of a document in TF\*IDF and in IR in principle, than a single triple.

The implementation idea can be summarized as follows:

1. pre-processing phase
  - (a) for each node in the graph,  $S$ , collect the text from its molecule into a single string  $SS$
  - (b) extract key phrases from  $SS$  (based on TF\*IDF)
  - (c) add an extra triple to each node, which denotes it's key-phrases, in other words,  $\langle S, hasKeyPhrases, "kp1, kp2, kp3" \rangle$
  - (d) get this extra triple (along with all others) indexed through FTS
2. selection and retrieval phase
  - (a) extract key-words from the SPARQL query
  - (b) use FTS to retrieve the nodes which have key-phrases in common with the query
  - (c) compose a triple-set of all the molecules for these nodes and return it as selected part of the graph

The above presents how one could make selection plugin, similar to the current LarKC BaseLineFTSelector <sup>1</sup>, with the difference that on Step 2b the BaseLineFTSelector is searching for all possible mentions of the key-words from the query. One can describe it as a query pattern:

<sup>1</sup>See the SVN repository [https://svn.gforge.hlr.de/svn/larkc/trunk/platform\\_v04](https://svn.gforge.hlr.de/svn/larkc/trunk/platform_v04)



```
select ?s
where {?s ?p ?l.
      ?l contains "..."}

```

In the current LarKC KeyPhraseSelector <sup>2</sup>, the criteria is

```
select ?s
where {?s hasKeyPhrases ?l.
      ?l contains "..."}

```

The result is that the LarKC KeyPhraseSelector is more selective, because it will not return nodes, for which the query keywords are not characteristic enough. The above algorithms could be modified in various ways. For example, it may make sense in some contexts to index and search not only against tokens found in plain string literals, but also to extract tokens from the local names of URIs, e.g. getting “Hamish” and “Cunningham” as tokens/features describing URI `http://www.shef.ac.uk/dcs/people/HamishCunnigham``http://www.shef.ac.uk/dcs/people/HamishCunnigham`.

While many such modifications are possible, they can be tackled at a latter stage, if the overall approach delivers some meaningful results.

The above description of a selection plugin can also be cast into a geometrical model:

- The geometrical plug-in can take the keywords attached to the nodes of the graph after pre-processing as features of the molecule. I.e. in the vector-space model, the documents are the RDF molecules and the coordinates / features are the key-words.
- At query time, the geometrical selector uses the method for extraction of keywords from SPARQL queries (as described above), then retrieves the most relevant URIs from its VSM model.
- Given a list of URIs, produces a tripliset of their RDF molecules (as in 2.3 above) - this is the result tripliset of the geometrical plugin

One can think of various modifications. The most obvious one is to have some weights assigned to each key-phrase for each node. This makes the use of key-phrases as input for the VSM and relevance ranking much more interesting.

### 3.3 IR-style Selector

In the previous section, we described an approach to index term generation based on RDF molecules. This section looks at implementation considerations for a new family of plugins, based on this approach, that we plan for future LarKC iterations.

Any classical IR engine could be used on top of the model described in the last section. Imagine that we store the results of Step 1a above:

1.a for each node in the graph, S, collect the text from its molecule into a single string SS

This is treated as document content and passed to some IR engine. This way, one can use an IR library instead of a semantic repository for selection. Importantly, we do not pass to the IR engine only the keywords, we pass it all of the content.

<sup>2</sup>See the SVN repository [https://svn.gforge.hlr.de/svn/larkc/trunk/platform\\_v04](https://svn.gforge.hlr.de/svn/larkc/trunk/platform_v04)



### 3.3.1 Base-line Selection Through SPARQL evaluation

An interesting base-line implementation could be implemented by selecting parts of the graph relevant to a SPARQL query, exactly by evaluating the query. A more sophisticated baseline could be created as follows:

- modify the query so that selection criteria is \*, i.e. all of the variable (if feasible);
- collect all of the URIs received in the result set and consider the aggregation of their molecules as selection.

### 3.3.2 Design considerations

The above introduces several different designs. We need to evaluate each of these, in order that:

- we can see which of them seem most interesting (in terms of estimates for development costs and value);
- we can consider what is reusable across the different modalities, so, that we can design an architecture which delivers a set of different selection modalities with minimum redundancy.

## 3.4 IR queries over RDF graphs

The ideas introduced above can be placed in a more general context. FTS and the key-phrase selectors can be seen as a way of performing IR queries on top of RDF graphs. The different parts of the picture are:

- basic FTS can be used to return meaningful results;
- keywords can be associated with nodes;
- PageRank can be derived for nodes (adapting existing implementations), allowing us to have context-unaware (stateless) rank for nodes;
- spreading activation can be layered on top of RDF graph that can be used for the sake context-based ranking of graphs.

This all sums up in a set of features which will allow us develop an RDF search engine. This set can be further extended with RDF graph enrichment, based on:

- co-occurrence analysis based on semantic annotation from text (if available) using CORE
- inference

Is such an RDF search engine necessary? In linked-data style environments an RDF search engine would make at least as much sense as structured (e.g. SPARQL) queries. An RDF search engine would also make sense in any environment which integrates data from multiple sources. Data from multiple structured sources usually contains more



meaning than can be retrieved through SPARQL. This is because such sources are not perfectly integrated. Whenever there is a properly designed and maintained single database, it by definition allows good structured queries: this is what a good databases are all about. On the other hand, when there are multiple databases integrated and accessible from a single point, people make mappings/links which make the databases “sufficiently” connected, meaning that “there are enough links to allow you select data and define some constraints on all the databases at once”. This “sufficiently” and “enough” usually delivers much lower level of semantic and structural homogeneity than what you can find in a single database or KB. It is complex and labour intensive exercise to align all the data modelling assumptions and practises inherent in the source databases. Thus assuring “basic” interlinking is already quite expensive.

From the LarKC life science use cases, it seems that many of the public servers providing access to life-science information have very limited usage, with very few queries each day. Users try out of curiosity, but may not come back. So, there should be something wrong in the assumption that exposing a somehow-interlinked set of datasets is extremely useful. In essence, the value that a person performing a structured query against a set of interlinked dataset is determined by (i) the level of knowledge he has about the schema and the data modelling for each of the resources and (ii) the level of knowledge he has about the interlinking schema and patterns. In a perfect world, a user may be happy to be able to query 10 databases all together, which he knows well one by one and he has also taken care to read the perfectly written documentation about the mapping. This perfect situation is probably representative for 0.1% of the potential users of data integrated from multiple sources. The selection-style search discussed above, could therefore provide a valuable information access method that is likely to become a good extension of other search and selection methods in LarKC.



## 4. Geometric Methods for Selection

The previous two Chapters discussed the current, delivered, plugins, and described an idea for future plugins based on RDF triples. In this chapter we will discuss information retrieval (IR) and linguistics based methods for RDF triple selection, in particular the vector space model (VSM).

First we can obtain the keywords from one triple or the bigger representation like RDF molecule. Then we can assign a virtual document containing those keywords to the triple or molecule. Once we have virtual documents, we can use the document based IR methods for selection. We will discuss the applications of two major IR methods to RDF selection, the keyword matching and VSM. We also discuss the recent improvements on the VSM and their applications to the selection problem.

### 4.1 Keywords, Vector Spaces and Selection

**Boolean retrieval:** The simplest method for selecting a set of RDF triples relative to a SPARQL query is boolean retrieval based on keywords: first collect keywords from the triples and create an inverted index from keyword to all triples associated with the keyword; then, given a query, extract keywords from the query and retrieve from the inverted index those triples which associate with some of the query's keywords. Of course in this procedure we can use the RDF molecules instead of triples as the unit for extracting keywords, as discussed in Chapter 1.

Boolean retrieval technique is a basic method in information retrieval. Complicated ranking methods, such as those based on distance or inner-product of vectors, have been developed using the same set of keywords and data structure like inverted index. It was also used in the Semantic Web search engines like Sindice [25]. Note that the Sindice indexed the keywords against RDF documents, while the method proposed here is to index the keywords against the RDF triples (or molecules). Therefore we implement the keyword based Boolean retrieval as the baseline.

One concern with the keyword matching for selecting RDF triples is where to find the keywords to associate with a triple. We can assume that the local names of the three components of a triple are important keywords. We can also extract the content words from some fields of a triple's URI as keywords for a triple using the URI reference, though those content words may be less important than the local names of the URI reference for characterising the triple. If a triple has a link back to the source document from which it was generated, then those content words in the source document can also be regarded as the keywords for the triple (especially those proximate to the point of reference).

Boolean retrieval method is quick and efficient in storage, but it is not flexible in the sense that it just produces a fixed subset for one query and cannot take into account the constraints represented by context and quality of service criteria. One method for adding flexibility to the keyword matching approach is to rank those triples selected by matching.

**Weighted keyword matching** assigns a weight (or an importance indicator) to each keyword in a triple and stores the weights in the inverted index of keywords and triples. Then we can rank the triples selected by the word matching method for a given query, based on the weights of the keywords of the query in those triples. Another method for ranking is to create vectors (e.g. based on TFIDF scores and the weights



of keywords) for each triple (or molecule) and query and then rank the triples based on the inner product or cosine between the triples' and query's vectors. For convenience, we may call the first extension of the keyword matching approach as weighted keyword matching, and the second extension as vector ranking.

**Vector space model:** Once we create a vector space model by associating a vector with a triple or query, we can explore vector space retrieval methods which are potentially richer and more sensitive than the keyword methods. For example, we may cluster the triples based on the vector representation for more efficient selection (with a first step that selects clusters, i.e. that starts to move away from the requirement to consider all triples for each selection task). We may also map the vector space to a higher dimensional one or even infinite-dimensional Hilbert space e.g. via kernelisation, for more effective comparison between query and triples. We will give more considerations on the Hilbert space based methods in Section 4.4.

One particular implementation of the method is for the case that all triples are generated from documents and each triple has a link back to the source document. If we only want to use the content words from the source documents to characterise the triples, then we have a selecting method based on the conventional document based IR methods. It first selects those documents which are relevant to a query and then retrieves all the triples generated from those documents as a subset for the query. Note that this method cannot distinguish the keywords from different components of a triple, though this kind of distinction is available from both the triples and the SPARQL query.

Actually vector space model is one major method in information retrieval. Once the query and retrieval data are formulated as vectors, a variety of mathematical tools such as those in Linear Algebra can be explored to deal with various problems. Moreover, since the vector space has rich geometric structure like the Euclidean space, we can exploit the geometry of vector space as well.

**Random indexing:** The standard way of forming a vector in information retrieval (i.e. the vector space model (VSM)) is to assign one coordinate of vector to one keyword. The dimension of vector is equal to the number of keywords. Hence the vector's dimension increase as more and more keywords are added. In the random indexing [24], each keyword is assigned a sparse random vector with a pre-specified dimension. If an object contains multiple keywords, the vector of the object is the sum of the random vectors for the keywords that the object has. In comparison with the conventional method, the random indexing method results in the vector with the pre-defined dimension, which is an advantage for software implementation. In addition, the random indexing provides a natural way to encode the order in the context by the means of permutations [24].

The word space model with random indexing can catch more information than the VSM model, and the random indexing based implementation of word space is quite efficient for large data (see the Deliverable 2.1.1 for more information about word space and random indexing). In the next section we propose a method for using the semantic space model for RDF triples or molecules selection and retrieval.

## 4.2 Semantic Space Model for RDF Triple Selection

Vector space model in IR generates vector representations for documents (or web pages) and query solely based on the content words that those objects contains (i.e.



the internal contents of objects). More advanced methods of IR (e.g., LSA, HAL, BEAGLE, random indexing) propose mathematical models that attempt to discover the meaning of words through their context (through an implicit statistical analysis).

Interestingly, an RDF triple (or molecule) can be seen at an intermediate level between document and word, because it contains meaningful terms or words and relations among them; it could also have meaningful relations with some other triples or molecules. Both the content of an RDF triple and its relations with other triples are useful for selection and retrieval. Therefore, when we use the geometric methods for RDF triple selection, we need to combine the vector space models for IR with more advanced methods, in order to encode the internal and external information for one RDF triple or molecule.

[24] proposed to use permutations for encoding order information. From our discussions on semantic space models presented in Deliverable 2.1.1 we can see that permutations can be used for encoding other relations. Since an RDF triple (or molecule) can have different relations with other triples or molecules, we can use permutation to encode these relations. We will also use sparse random indexing rather than the full random indexing, because the former is more efficient computationally.

Therefore we propose the following procedure for applying semantic space model to RDF triple selection. It has two parts. The first part encodes RDF triples or molecules using random indexing and permutation operations so that each triple or molecule is assigned one vector with pre-defined dimension. The second part retrieves the relevant RDF triples for a given query by comparing the vector of query with those of RDF triples in database.

**Encoding:**

1. assign a sparse random vector  $e_a$  to a word or term  $a$  encountered.
2. assign a sparse random vector  $e_r$  to a relation  $r$ .
3. assign a permutation  $\Pi_r$  to a type of relation  $r$  encountered. The inverse of the relation  $r$  is the inverse of the permutation,  $\Pi_r^{-1}$ .
4. assign to an RDF triple  $t = \langle a, r, b \rangle$  a vector  $v_t = e_a + e_b + e_r + \Pi_r e_b + \Pi_r^{-1} e_a$ , meaning that the triple has three terms  $a$ ,  $b$ , and  $r$ , and has the two partial relations  $\langle \cdot, r, b \rangle$  and  $\langle a, r, \cdot \rangle$ .
5. given an RDF molecule  $m$  containing two triples  $t_1$  and  $t_2$  which have relation  $R$ , assign to the molecule a vector  $v_m = v_{t_1} + v_{t_2} + e_R + \Pi_R v_{t_2} + \Pi_R^{-1} v_{t_1}$ .
6. If two RDF molecules  $m_1$  and  $m_2$  have relation  $R$ , the vectors for the two molecules are updated,  $v_{m_1} = v_{m_1} + \Pi_R v_{m_2}$  and  $v_{m_2} = v_{m_2} + \Pi_R^{-1} v_{m_1}$ , respectively.

**Selection:** given a query  $q$ , e.g. finding out a term  $x$  which satisfies the two constraints  $\langle x, r_1, b \rangle$  and  $\langle a, r_2, x \rangle$ , one first obtains a vector from the query,  $v_q = e_a + e_b + e_{r_1} + e_{r_2} + \Pi_{r_1} e_b + \Pi_{r_2}^{-1} e_a$ ; then computes the inner product between  $v_q$  and each of the RDF triple or molecule vectors stored in the RDF database; finally, as the inner product indicates the degree of relevancy of the triple to the query, one can rank the RDF triples according to their relevance.

The computational complexity is a concern for any methods applied to large amount of data. Encoding part can be done at initialisation time, before issuing any query and can be updated locally and in an incremental fashion. Selection or retrieval



part could be computationally extensive, because of computing an inner product between the query vector and every RDF triple or molecule vector and the number of RDF triples or molecules could be potentially very large. However, since computations of those inner products are not dependent on each other, they can be parallelised. One can even design special hardware to execute the inner product computation in a massively parallel way. Sparse random indexing is also helpful for reducing computation time and memory consumption. In addition, one can first use more efficient method like keyword based Boolean retrieval to select a subset of RDF triples and then apply the geometric method to the subset for more accurate selection. This way, the computation time could be reduced to a reasonable level. However, in our future work we will carefully evaluate the ways to improve computational time, before proceeding further.

### 4.3 Related work

There are quite a few works carried out recently on using linguistic features and information retrieval methods for the selection and retrieval of the RDF triples. In the following we review some of the works, and also some other works which could be used to improve the geometric methods.

[23] extracted a virtual document for each URI reference in an RDF triple store (or equivalently each node in a RDF graph). This virtual document contains the local name and labels of the URI reference, other associated literals such as those in `rdfs:comments`, and the names of neighbour nodes in the RDF graph.

[3] used these virtual documents to create a system for searching and browsing RDF triples using a vector space retrieval model. Basically the system uses a keyword matching method to compare a query against the virtual documents associated with a triple in order to determine the relevancy of the triple to the query.

[11] presented an A-Box summarization technique for efficiently making inference over the A-box. The technique used one summarized node to represent all individuals belonging to one concept, unless two individuals were explicitly stated to be different. In the latter case two more summarized nodes were used to represent the two individuals. The relation between two individuals were assigned to the two corresponding summarized nodes.

[25, 21] described the *Sindice*, a system to locate the RDF documents being relevant to the given keywords and/or URIs. It collected the Semantic Web resources from internet. It then indexed the keywords and URIs against the RDF documents containing those keywords and URIs, using the inverted index scheme. Hence its indexing is similar to the baseline plugin. On the other hand, we index the keywords against the RDF triples, while the *Sindice* indexed the keywords and URIs against the RDF documents.

[12] presented the *ReConRank* algorithm which adapted the well-known PageRank algorithm to Semantic Web data. It basically ranked the nodes in a topical subgraph that was selected based on keyword matching from the RDF files. In another word, it ranked the results of a query based on the RDF links in the results. The subgraph that the algorithm applied to includes not only the subject nodes related to the query, but also the contexts of the subject nodes (i.e. the provenances or sources of the subjects), in order to improve the quality of ranking.



## 4.4 Efficiency Issue of the Methods

Efficiency is an important issue for the methods to be applied to the large data sets that the LarKC project deals with. In the following we discuss the efficiency of the methods.

In this chapter we have discussed several IR and linguistics based methods for RDF triple selection. The two major IR methods, the keyword based Boolean retrieval and the vector comparison in the VSM, are applicable to triple selection. Keyword matching is very efficient, because basically it just retrieves data from a structured list. In comparison, the VSM is less efficient because it involves many more computations such as forming a vector for each document and computing inner product of two vectors for comparing the query and document. Therefore the Boolean matching has been used in internet search engines such as Google and the RDF search engines such as Sindice. The WP2 baseline plugin also uses Boolean retrieval. On the other hand, the VSM method may provide more accurate results than the keyword matching. So we will investigate the VSM method in the later stage.

One of the most recent advances in IR is to develop new geometric methods based on the interesting analogies between components of information retrieval and quantum mechanics [26, 19]. Basically it generalises the VSM method based on the Euclidean space to the geometric methods based on the Hilbert space. It also tries to apply the principles and methods in quantum mechanics, such as uncertainty principle and quantum entanglement, to information retrieval. The WP2 deliverables D2.1.1 included a review of the applications of the quantum methods to IR and natural language processing.

Though the study of quantum methods for IR is a promising direction for dealing with more complicated aspects of IR such as context-aware and dynamical IR, so far the IR community has not developed any significantly effective IR methods based on quantum methods. One major proposal in this area is to replace the Euclidean space with the Hilbert space. However, the previous experiments indicated that the Hilbert space based method would not provide radically different results from the VSM method. For example, in the so-called kernel learning algorithms such as support vector machines and Perceptron, the linear kernel and polynomial kernels correspond to the Euclidean space as feature space, while the Gaussian kernel corresponds to the infinite-dimensional Hilbert space. In the document classification experiments using support vector machines, there were no significant difference in accuracy between using linear kernel and Gaussian kernel (see e.g. [13]). On the other hand, since Gaussian kernel involves exponential computation, the computational complexity of Gaussian kernel is much higher than that of linear kernel and polynomial kernel. Therefore, as mentioned in Chapter 1, we will not spend much time on further development of the quantum methods, until there are significant developments in the applications of the quantum modelling to IR and natural language processing. Although we may support geometrical models in the future (see for example, the discussion of VSMs in the context of RDF molecule based selection presented in Chapter 3), these will probably not include quantum geometrical models.



## 5. Conclusions

This report has described the Month 12 software deliveries D2.2.1 (baseline components v1) and D2.5.1 (geometrical semantics components v1). To recap, at Month 12 we have delivered the following LarKC plugins:

**Baseline Selector** a simple baseline naive implementation of selection, using full-text search applied to RDF graphs;

**Key Phrase Selector** Similar to the Baseline selector, it adapts selection criteria based only on characteristic key-phrases, instead of any text tokens appearing in the dataset.

**Prior Knowledge Selector** given a set of ontology instance identifiers, a triple store repository, and some knowledge selection criteria, returns a subset of the repository that further describes the identifiers;

**Ranked Selector** given a set of ontology instance identifiers and prior knowledge about the instances, rank these instances according to some model;

**Semantic Annotation Transformer** a transformer plugin wrapping an arbitrary semantic annotation pipeline from GATE.

The latter three plugins have been written in response to the requirements set out by the LarKC Carcinogenesis Research use case from WP7b. The selection scenarios outlined in the D7b.1.1a requirements document include models not considered in our previous deliverables, such as selection of triples for multiple data points, calculating metrics on triples according to statistical models, and selecting based on ranking according to these metrics.

In addition, we have:

reported further on the geometrical methods discussed in our previous deliverable D2.1.1, outlining problems with their computational complexity, and our intentions to de-emphasise this work;

started to develop a more sophisticated approach to the derivation of indexing terms from node identifiers and URIs than that provided by our current baseline, based on RDF molecules.

In future releases of plugins from this workpackage, we intend to further develop plugins that directly support the use cases; look more closely at selection and retrieval plugins that provide textual analysis of both RDF graph regions and of text documents; consider other ways in which geometrical models can be used to support these.



## REFERENCES

- [1] M. Agatonovic, N. Aswani, K. Bontcheva, H. Cunningham, T. Heitz, Y. Li, I. Roberts, and V. Tablan. Large-scale, parallel automatic patent annotation. In *Proc. of 1st International CIKM Workshop on Patent Information Retrieval - PaIR'08*, Napa Valley, California, USA, October 30 2008.
- [2] K. Bontcheva, V. Tablan, D. Maynard, and H. Cunningham. Evolving GATE to Meet New Challenges in Language Engineering. *Natural Language Engineering*, 10(3/4):349–373, 2004.
- [3] G. Cheng, W. Ge, and Y. Qu. Falcons: Searching and Browsing Entities on the Semantic Web. In *Proceedings of WWW2008*, pages 1101–1102, 2008.
- [4] H. Cunningham. GATE, a General Architecture for Text Engineering. *Computers and the Humanities*, 36:223–254, 2002.
- [5] H. Cunningham. Information Extraction, Automatic. *Encyclopedia of Language and Linguistics, 2nd Edition*, pages 665–677, 2005.
- [6] H. Cunningham and K. Bontcheva. Computational Language Systems, Architectures. *Encyclopedia of Language and Linguistics, 2nd Edition*, pages 733–752, 2005.
- [7] H. Cunningham, D. Maynard, K. Bontcheva, and V. Tablan. GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications. In *Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics (ACL'02)*, 2002.
- [8] L. Ding, T. Finin, Y. Peng, P. P. da Silva, and D. L. McGuinness. Tracking RDF Graph Provenance using RDF Molecules. Technical Report TR-CS-05-06, University of Maryland, Baltimore, MD, US, 2005.
- [9] M. Dowman, V. Tablan, H. Cunningham, and B. Popov. Web-assisted annotation, semantic indexing and search of television and radio news. In *Proceedings of the 14th International World Wide Web Conference*, Chiba, Japan, 2005. <http://gate.ac.uk/sale/www05/web-assisted-annotation.pdf>.
- [10] D. Fensel, F. van Harmelen, B. Andersson, P. Brennan, H. Cunningham, E. Della Valle, F. Fischer, Z. Huang, A. Kiryakov, T. K. Lee, L. School, V. Tresp, S. Wesner, M. Witbrock, and N. Zhong. Towards larkc: a platform for web-scale reasoning. In *Proceedings of the IEEE International Conference on Semantic Computing (ICSC 2008)*, Santa Clara, CA, USA, 2008. IEEE Computer Society Press.
- [11] A. Fokoue, A. Kershenbaum, L. Ma, E. Schonberg, and K. Srinivas. The summary abox: Cutting ontologies down to size. In *Proc. of the Int. Semantic Web Conf. (ISWC2006)*, pages 136–145, 2006.
- [12] A. Hogan, A. Harth, and S. Decker. Reconrank: A scalable ranking method for semantic web data with context. In *Second International Workshop on Scalable Semantic Web Knowledge Base Systems (SSWS 2006)*, Athens, GA, USA, 2006.



- [13] T. Joachims. Text categorization with support vector machines: Learning with many relevant features. In Claire Nédellec and Céline Rouveirol, editors, *Proceedings of ECML-98, 10th European Conference on Machine Learning*, number 1398 in Lecture Notes in Computer Science, pages 137–142, Chemnitz, Germany, 1998. Springer Verlag, Heidelberg.
- [14] A. Kiryakov, B. Popov, D. Ognyanoff, D. Manov, A. Kirilov, and M. Goranov. Semantic annotation, indexing and retrieval. *Journal of Web Semantics, ISWC 2003 Special Issue*, 1(2):671–680, 2004.
- [15] Atanas Kiryakov. OWLIM: balancing between scalable repository and light-weight reasoner. In *Proc. of WWW2006*, Edinburgh, Scotland, 2006.
- [16] Y. Li, K. Bontcheva, and H. Cunningham. SVM Based Learning System For Information Extraction. In M. Niranjan J. Winkler and N. Lawrence, editors, *Deterministic and Statistical Methods in Machine Learning*, LNAI 3635, pages 319–339. Springer Verlag, 2005.
- [17] Y. Li, K. Bontcheva, and H. Cunningham. Using Uneven Margins SVM and Perceptron for Information Extraction. In *Proceedings of Ninth Conference on Computational Natural Language Learning (CoNLL-2005)*, 2005.
- [18] Y. Li, K. Bontcheva, and H. Cunningham. Hierarchical, Perceptron-like Learning for Ontology Based Information Extraction. In *16th International World Wide Web Conference (WWW2007)*, pages 777–786, May 2007.
- [19] Y. Li and H. Cunningham. Geometric and Quantum Methods for Information Retrieval. *SIGIR Forum*, 42(2):22–32, 2008.
- [20] A. Newman, Y.-F. Li, and J. Hunter. A Scale-Out RDF Molecule Store for Improved Co-Identification, Querying and Inferencing. In *The 4th International Workshop on Scalable Semantic Web knowledge Base Systems (SSWS) 2008*, Karlsruhe, Germany, 2008.
- [21] E. Oren, R. Delbru, M. Catasta, R. Cyganiak, H. Stenzhorn, and G. Tummarello. Sindice.com: A document-oriented lookup index for open linked data. *International Journal of Metadata, Semantics and Ontologies*, 3(1), 2008.
- [22] B. Popov, A. Kiryakov, A. Kirilov, D. Manov, D. Ognyanoff, and M. Goranov. KIM – A semantic platform for information extraction and retrieval. *Natural Language Engineering*, 10:375–392, 2004.
- [23] Y. Qu, W. Hu, and G. Cheng. Constructing virtual documents for ontology matching. In *Proceedings of WWW2006*, pages 23–31, 2006.
- [24] M. Sahlgren, A. Holst, and P. Kanerva. Permutations as a means to encode order in word space. In *Proceedings of the 30th Annual Meeting of the Cognitive Science Society (CogSci'08)*, Washington D.C., USA, 2008.
- [25] G. Tummarello, R. Delbru, and E. Oren. Sindice.com: Weaving the open linked data. In *Proceedings of the 6th International Semantic Web Conference*, Busan, Korea, 2007.



- [26] K. van Rijsbergen. *The Geometry of Information Retrieval*. Cambridge, 2004.