



INSEMTIVES
FP7-ICT-2007-3
Contract no.: 231181
www.insemtives.eu

INSEMTIVES

Deliverable <2.3.2>

Specification of Information Retrieval methods for semantic content

Editor:	Borislav Popov, ONTO
Deliverable nature:	<Report (R)>
Dissemination level: (Confidentiality)	<Public (PU) >
Contractual delivery date:	30.11.2009
Actual delivery date:	30.11.2009
Suggested readers:	Researchers, engineers
Version:	1.0
Total number of pages:	15
Keywords:	

Abstract

This deliverable defines the specification of the semantic retrieval methods to be exposed by the INSEMTIVES platform for the usage of the toolkit and the end user applications. It is based on the requirements described in INSEMTIVES deliverable D2.3.1 and in accordance with the specification of the platform (D3.1) and the specification of the annotation models (D2.1.2). The specification covers a wide range of retrieval methods starting from annotation-centric retrieval, conceptual queries, and hybrid multi-modality methods.

Disclaimer

This document contains material, which is the copyright of certain INSEMTIVES consortium parties, and may not be reproduced or copied without permission.

In case of Public (PU):

All INSEMTIVES consortium parties have agreed to full publication of this document.

In case of Restricted to Programme (PP):

All INSEMTIVES consortium parties have agreed to make this document available on request to other framework programme participants.

In case of Restricted to Group (RE):

All INSEMTIVES consortium parties have agreed to full publication of this document. However this document is written for being used by ;organisation / other project / company etc.; as ;a contribution to standardisation / material for consideration in product development etc.;.

In case of Consortium confidential (CO):

The information contained in this document is the proprietary confidential information of the INSEMTIVES consortium and may not be disclosed except in accordance with the consortium agreement.

The commercial use of any information contained in this document may require a license from the proprietor of that information.

Neither the INSEMTIVES consortium as a whole, nor a certain party of the INSEMTIVES consortium warrant that the information contained in this document is capable of use, or that use of the information is free from risk, and accept no liability for loss or damage suffered by any person using this information.

Impressum

[Full project title] INSEMTIVES – Incentives for Semantics

[Short project title] INSEMTIVES

[Number and title of work-package] WP 2 INSEMTIVES platform

[Document title] Specification of Information Retrieval methods for semantic content

[Editors: Name, company] Borislav Popov, ONTO and Ilya Zaihrayeu, UNITN

[Work-package leader: Name, company] Ilya Zaihrayeu, UNITN

Copyright notice

©2009-2012 Participants in project INSEMTIVES

Executive summary

The current deliverable deals with the specification of the semantic retrieval methods in the INSEMTIVES Platform. The various methods are grouped according to the types of metadata they work on: annotations, concepts, entities or textual content. Each of the methods is specified through its input parameters, resulting output, possible exceptions and description of what the method does. The specification in this deliverable is based on the specification of the annotation models (D2.1.2), the requirements for the retrieval methods (D2.3.1) and the platform specification and architecture (D3.1). The collection of retrieval methods over the various metadata modalities will empower the INSEMTIVES tools to interact with the data, access and retrieve metadata on the appropriate level of abstraction.

List of authors

Company	Author
<University of Trento>	<Uladzimir Kharkevich>
<Ontotext>	<Mihail Konstantinov>
<Ontotext>	<Borislav Popov>

Contents

Executive summary	3
List of authors	4
Abbreviations	6
Definitions	6
1 Introduction	8
2 Retrieval of Resources	8
Annotation-centric Retrieval	8
Annotation Patterns	9
Conceptual Queries	10
Full-text search	11
Co-occurrence and Ranking of entities	12
3 Retrieval of Annotations	13
4 Retrieval of Entities	14
Entity-centric methods	14
5 Retrieval of User Data	14
6 Conclusion	15

List of Figures

1	Core Elements of the Generic Model	10
---	--	----

Abbreviations

JMS Java Message Service

REST Representational State Transfer

API Application Programming Interface

RDF Resource Description Framework

SA-WSDL Semantic Annotations for WSDL

WSDL Web Service Description Language

XML Extensible Markup Language

MTP My Tiny Planets

TP Tiny Planets

COPPA Children’s Online Privacy Protection Act

SPARQL SPARQL Protocol and RDF Query Language

FTS Full-Text Query

IR Information Retrieval

LAS Landscape Awareness Service

Definitions

Web API Crawler A Web API crawler is a computer program that browses the World Wide Web in a methodical, automated manner and collects data about exposed services.

Unit Test In computer programming, unit testing is a software verification and validation method in which a programmer tests if individual units of source code are fit for use. A unit is the smallest testable part of an application.

Regression Testing Regression testing is any type of software testing that seeks to uncover software regressions. Such regressions occur whenever previously working software functionality stops working as intended.

Acceptance Testing In engineering and its various subdisciplines, acceptance testing is black-box testing performed on a system (e.g. software, lots of manufactured mechanical parts, or batches of chemical products) prior to its delivery.

Java Annotation An annotation, in the Java computer programming language, is a special form of syntactic metadata that can be added to Java source code.

Attribute An *attribute annotation element* is a pair $\langle AN, AV \rangle$, where AN is the name of the attribute and AV is the value of the

Ontology As defined by Studer et al., “an ontology is an explicit specification of a (shared) conceptualization”.

Relation A relation annotation element is a pair $\langle Rel, Res \rangle$, where *Rel* is the name of the relation and *Res* is another resource (i.e., different from the one being annotated).

Resource A *resource* is described by an annotation within annotation models. A resource can be any identifiable content identified by a unique identifier such as URIs. Examples of resources in the context of the Web are electronic documents such as images or text documents or their parts (e.g., a reference to an entity from a text document or an area in an image).

Semantic Annotation *Semantic annotation* describes both the process and the resulting annotation or metadata consisting of aligning a resource or a part of it with a description of some of its properties and characteristics with respect to a formal conceptual model or ontology.

Tag A *tag* annotation element is a non-hierarchical keyword or free-form term assigned to a resource.

1 Introduction

The current deliverable deals with the specification of the semantic retrieval methods in the INSEMTIVES Platform. The various methods are grouped according to the types of metadata they work on: annotations, concepts, entities or textual content. Each of the methods is specified through its input parameters, resulting output, possible exceptions and description of what the method does. The specification in this deliverable is based on the specification of the annotation models (D2.1.2), the requirements for the retrieval methods (D2.3.1) and the platform specification and architecture (D3.1). The collection of retrieval methods over the various metadata modalities will empower the INSEMTIVES tools to interact with the data, access and retrieve metadata on the appropriate level of abstraction.

The remainder of this document includes sections that discuss retrieval methods for objects of different kinds: resources (Section 2), annotations (Section 3), entities (Section 4), and user data (Section 5). Section 6 concludes the document.

2 Retrieval of Resources

This section specifies operations for retrieving resources, namely textual documents, web-services and multimedia. Additional difference is made of the retrieval methods - semantic retrieval (i.e. in the RDF metadata), free-text search and free-text search using synsets and semantic common-sense knowledge base.

Annotation-centric Retrieval

Annotation-centric retrieval uses formal specifications in annotations to find resources. This is the most powerful method used, since annotations build to a semantic model describing the resources which allows for making complex queries using SPARQL capabilities and reasoning over this metadata.

- *getResources*
 INPUT: *AnnotationRestriction*
 OUTPUT: *ResourceSet*
 EXCEPTION: *InvalidRestrictionException* – thrown if the *AnnotationRestriction* is not valid
 DESCRIPTION: Retrieves all the resources containing annotations fulfilling the restriction.

The *AnnotationRestriction* describes: (i) the *type* of the annotation, e.g. *tag*, *attribute*, and *relation*, (ii) possibly incomplete *annotation* of the given *type*, e.g. attribute name can be specified for attribute annotation and the attribute value can be left empty; (iii) the relation between the given *annotation* and the annotations describing the resources. It can be a semantic relation, when entities, concepts, and relations from background knowledge are used in order to retrieve resources with semantically similar annotations. For instance, resources with tags *dogs* and *cats* can be retrieved in response to the query with tag *animals*. Resources with attributes <topic, genetics> and <topic, immunology> can be retrieved in response to the query with attribute <topic, biology>. Resources with relation annotation <placed in, Trento> can be retrieved in response to the query with relation annotation <located in, Trento>. Relation between annotations in query and resources can also be purely syntactic, i.e., when only the textual representation of the annotations is used.

ResourceSet is a set of the retrieved resources.

- *getResources*
 INPUT: *UserIdSet*, *Mode*
 OUTPUT: *ResourceSet*
 EXCEPTION: none
 DESCRIPTION: Retrieves all the resources containing annotations made by the users in *UserIdSet*. If the set is empty, no resources are retrieved.

UserIdSet – a set of annotator Ids

Mode – defines how the multiple values in *UserIdSet* are combined. Values can be {*and,or*} for conjunction and disjunction of user Ids respectively.

- *getResources*
 INPUT: *AnnotationStartTime*, *AnnotationEndTime*
 OUTPUT: *ResourceSet*
 EXCEPTION: *InvalidTimeInterval* – if time interval specified by *AnnotationStartTime* and *AnnotationEndTime* is invalid.
 DESCRIPTION: Retrieves all the resources annotated in the period specified by *AnnotationStartTime* and *AnnotationEndTime*. If the first parameter is *null*, all resources annotated by the *AnnotationEndTime* are retrieved. Similarly, if the second parameter is omitted, all resources since *AnnotationStartTime* are retrieved.
- *getResources*
 INPUT: *EntitySet*, *Mode*
 OUTPUT: *ResourceSet*
 EXCEPTION: none
 DESCRIPTION: Retrieves all the resources containing references to entities from *EntitySet*. If the set is empty, no resources are retrieved.

EntitySet – a set of entities
Mode – defines how the multiple values in *EntitySet* are combined. Values can be *{and,or}* for conjunction and disjunction of entities respectively.

Annotation Patterns

The purpose of this type of retrieval is to use queries that allow restrictions on annotations as well as a intermingling of free-text terms and annotation specifications. As defined in General Model of Annotation in INSEMTIVES deliverable 2.1.2 we distinguish 6 types of annotations:

- Uncontrolled Attribute Annotation
- Uncontrolled Relation Annotation
- Uncontrolled Tag Annotation
- Controlled Attribute Annotation
- Controlled Relation Annotation
- Controlled Tag Annotation

Using this types and their features, we create an annotation query specification that has the form:
`{AnnotationType feature op value feature op value ...}`

where *AnnotationType* is one of the above 6 types, *feature* is a property name of an annotation and *op* is a relational operator from the following set – `{==, !=, >=, <=, >, <}`

For example, Controlled Tag Annotation has the form $tag^c = \langle lc, r, [u,]ts[, \alpha] \rangle$, where *lc* is a linguistic concept whose concept *c* belongs to a taxonomy (i.e., $c \in tx$); *r* is the annotated resource ($r \in R$); when present, *u* is the user who created this annotation; and *ts* is the time stamp recorded when the annotation was created. When the annotation is added by an automatic or semi-automatic algorithm, α is provided to indicate the accuracy of the algorithm that created the annotation. To specify query that retrieves all resources with controlled tag annotation restriction created by user different from *self* and are created later than 2009-11-24 we construct the following string:

```
{ControlledTagAnnot user!="self" timestamp>20091124}
```

In order to construct complex queries or complex queries mixing free-text terms and annotation query specifications, the query syntax provides 3 logical operators: *and* using & symbol, *or* using | and *sequence* with no operator symbol. For example, to query for resources containing the value 144 Hz the following query is used:

```
144 & {ControlledRelAnnot targetResource="http://dbpedia.org/page/Hertz"}
```

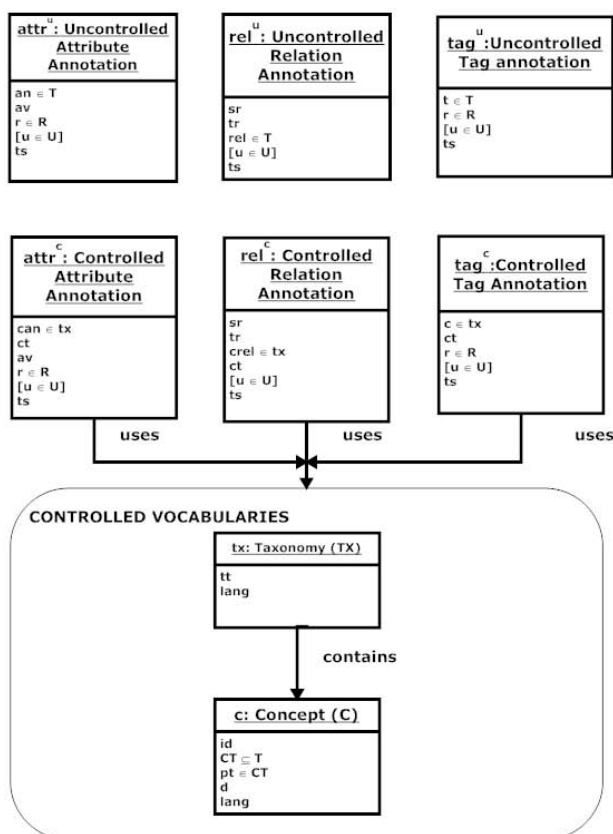


Figure 1: Core Elements of the Generic Model

- *getResources*
 INPUT: *PatternQuery*
 OUTPUT: *ResourceSet*
 EXCEPTION: *InvalidPatternQueryException* – if there is syntax error in the query
 DESCRIPTION: Retrieves all resources matching the supplied pattern query.

Conceptual Queries

These queries are capable of selecting graph patterns in the knowledge base. This is very similar to the expressive power exposed by the lower level RDF querying performed through SPARQL, but in this case it will also expose an object oriented model for query construction with definitions of restrictions over labels and attributes of resources, as well as properties linking them to other resources.

- *getResources*
 INPUT: *ConceptualQuery*
 OUTPUT: *ResourceSet*
 EXCEPTION:
InvalidQueryException – if assumptions about query are not satisfied.
IncompatibleResultException – if the result is not of type *Resource*
 DESCRIPTION: Retrieves all resources specified by query result variable in the matched sub-graph.
- *getAnnotations*
 INPUT: *ConceptualQuery*
 OUTPUT: *AnnotationSet*
 EXCEPTION:
InvalidQueryException – if assumptions about query are not satisfied.
IncompatibleResultException – if the result is not of type *Annotation*

DESCRIPTION: Retrieves all annotations from the matched sub-graph and specified by query's result variable.

ConceptualQuery is a sub-graph construct with one variable defining the node for the result. The query provides the following methods:

- `setInterestedInLiteral`(URI prop, String regex) – sets regular expression to be matched for value of type literal. Null values are treated as wildcards.
- `setInterestedInResource`(URI prop, URI resource, URI type) – sets the value of the property and its type. The type parameter is optional. Null values are wildcards.
- `removeInterestedInProperty`(URI prop) – removes property and value specification for the result node. Property cannot be *null*.
- `addStatement`(URI subj, URI pred, URI obj, URI type) – adds triple with optional type restriction. Subject cannot be *null*, all other are treated as wildcards.
- `addStatement`(URI subj, URI pred, String regex) – add triple with regular expression to be matched for literal object value. Subject cannot be *null*, all other are wildcards.
- `removeStatement`(URI subj, URI pred) – removes triple for the given subject and predicate. Both subject and predicate cannot be *null*.
- `removeStatements`(URI subj) – removes all statements for the given subject. Subject cannot be *null*.
- `clear`() – removes all statements.

Full-text search

The full-text search retrieval methods will be applicable to information resources with textual representation. Although exposing an API for construction of such queries, it seems enough to exploit a powerful query syntax that is already existent. Examples are the syntaxes of Lucene¹, MG4J², and others.

- *getResources*

INPUT: *FTSQuery*, *QDMatching*

OUTPUT: *ResourceSet*

EXCEPTION: *InvalidQueryException* - if the query syntax is messed up

DESCRIPTION: Retrieves all the resources matching the query *FTSQuery* by using the matching technique *QDMatching*, which can be either syntactic or semantic matching. If syntactic matching approach is used, then words or multi-words phrases that occur in documents and query will be used as atomic elements in document and query representations and the retrieval procedure will be based on syntactic matching of document and query representations, e.g., string comparison of words (or stems) in documents and query will be used. If semantic matching approach is used, then concepts (word senses), expressed in an unambiguous formal language, will be used instead of words in document and query representations. Semantic matching of (complex) concepts (e.g., checking subsumption between complex concepts by exploiting knowledge about concept relatedness) will be used instead of word matching. For instance, if semantic matching approach is used, then the document about *black dogs* can be retrieved as an answer to the query *black animal*.

FTSQuery will utilize simple but very powerful syntax using boolean operators. The simplest query possible is a single term: the answer that you will obtain by such a query is the set of all documents that contain the term or any other uppercase/lowercase variant thereof. For more complex queries the following operator will be available:

¹Lucene - an IR engine, <http://lucene.apache.org/>

²MG4J - an IR engine, <http://mg4j.dsi.unimi.it/>

- AND (operator: &) - writing a query that contain several terms separated by white-space will have the effect of retrieving resources that contain all of the terms. Optionally one could junction the terms with the & operator. For example, the query `monkey & parrot` will retrieve all resources containing `monkey` and `parrot` nevermind the order they appear in the text;
- OR (operator: |) - this operator implements disjunction of the searched terms, i.e. the system will retrieve resources that contain any of the given words;
- NOT (operator: !) - this is the negation operator. For example, the query `monkey & !gorilla` will get all documents containing `monkey`, but not `gorilla`;
- phrase (operator: "phrase") - a list of terms in quotation marks will make the system retrieve resources that contain these terms consecutively;
- proximity restriction (operator: (query) number) - to search for terms within limited portion of a document, a tilde operator is used. The number after the operator means in how many words the searched terms must appear (in any order). For example, the query `(jungle monkey)~7` will yield resources where `jungle` and `monkey` appear within 7 words in the document;
- ordered AND (operator: !) - writing more than one term separated by < will find documents containing the given terms in the specified order;
- wildcard search (operator: *) - appending * at the end of a term will search for documents containing the term with concatenated zero or more characters after it;
- grouping/priority (operator: (query)) - in order to group a part of the query a parentheses are put around it. For example, the query `(Crab-eating|Stump-tailed) macaque` will retrieve all document containing `crab-eating macaque` or `stump-tailed macaque`.

The precedence of operators is as in logic.

Co-occurrence and Ranking of entities

This retrieval method will allow the exploration of associative links and statistically-derivable soft relationships between annotations or referred entities. If ordered attribute spaces are available for the annotated resources, like time stamps or annotation actions, the method can also be used in accumulative manner to examine trends of either occurrence or co-occurrence. The contexts in which co-occurrence makes sense vary on the size of the resource and especially the density of annotations in it, as well as the level of diversity of information presented in the same resource. For this reason the context of co-occurrence will be configurable (like an entire document, a chapter, paragraph, sentence for textual documents). For certain needs multiple contexts can be meaningful - so in this case the platform will support a layered approach to context-based co-occurrence.

- *getResources*
 INPUT: *ResourceQuery*
 OUTPUT: *ResourceList*
 EXCEPTION: *InvalidQueryException* - in case of illegal query
 DESCRIPTION: retrieves all resources that match *ResourceQuery* that contains co-occurrence search criteria according to ranking mechanism specified in a the query.

ResourceQuery object sets the criteria for co-occurrence as well as a ranking mechanism. The methods that it provides are:

- *setCooccurringEntities(Set < Entity > entities)* - sets the entities that must co-occur in the search resource. Their order is irrelevant.
- *setKeywords(Set < String > keywords)* - sets the keywords that must co-occur in the search resource. Their order is irrelevant.

- *setRankingMechanism(RankingMechanismType)* - sets one of the predefined ranking mechanisms to be used with the query.
- *setContext(Contextcontext)* - sets spatial (in document) and temporal constraints for co-occurrence (e.g from date to date). This parameter is optional since it has default value in a configuration file.

ResourceList is a list of resources ordered according to the ranking mechanism defined in the query.

3 Retrieval of Annotations

This operations operate over the space of the annotations.

- *getAnnotations*
 INPUT: *ResourceID*
 OUTPUT: *AnnotationSet*
 EXCEPTION: *NoSuchResourceException* – if resource with the provided ID does not exist.
 DESCRIPTION: Retrieve all annotations of a resource as a set.
- *getAnnotations*
 INPUT: *UserIdSet, Mode*
 OUTPUT: *AnnotationSet*
 EXCEPTION: none
 DESCRIPTION: Retrieves all annotations made by the users in *UserIdSet*. If the set is empty, no annotations are retrieved.

UserIdSet – a set of annotator Ids
Mode – defines how the multiple values in *UserIdSet* are combined. Values can be {*and,or*} for conjunction and disjunction of user Ids respectively.
- *getAnnotationsAt*
 INPUT: *ResourceID, startOffset, endOffset*
 OUTPUT: *AnnotationSet*
 EXCEPTION:
NoSuchResourceException – if resource with the provided ID does not exist.
InvalidOffsetIntervalException – if offset specification is invalid.
 DESCRIPTION: Retrieve all annotations in a particular offset range of a resource. This method is applicable for textual documents.
- *getAnnotationsAt*
 INPUT: *ResourceID, startTime, endTime, Layer*
 OUTPUT: *AnnotationSet*
 EXCEPTION:
NoSuchResourceException – if resource with the provided ID does not exist.
InvalidTimeIntervalException – if time interval specification is invalid.
InvalidLayerException – if there is no layer with the specified number. DESCRIPTION: Retrieve all annotations in a particular time offset range of a resource. This method is applicable for continuous multimedia resources.
Layer specifies on what layer the annotations are located. This parameter is optional.
- *getAnnotationsAt*
 INPUT: *ResourceID, Shape, Layer*
 OUTPUT: *AnnotationSet*
 EXCEPTION:
NoSuchResourceException – if resource with the provided ID does not exist.
InvalidTimeIntervalException – if time interval specification is invalid.

InvalidLayerException – if there is no layer with the specified number. DESCRIPTION: retrieve all annotations covered by a particular geometrical shape covering a part of a static multimedia resource - like an image.

4 Retrieval of Entities

Entity-centric methods

These retrieval methods again exploit the link between annotations and information resources, but go one step further and in the case of semantic annotations expose the links to entities in the knowledge base.

- *getEntities*
 INPUT: *AnnotationSet*
 OUTPUT: *EntitySet*
 EXCEPTION: none
 DESCRIPTION: Retrieves all entities linked from the provided annotation set. If the annotation set is empty, the result is an empty *EntitySet*.
- *getEntities*
 INPUT: *ResourceID*
 OUTPUT: *EntitySet*
 EXCEPTION: *NoSuchResourceException* – if resource with the provided ID does not exist.
 DESCRIPTION: Retrieves all entities linked from the provided resource.

5 Retrieval of User Data

The purpose of these operations is to facilitate extraction of user data from multiple annotations and from resources.

- *getUsers*
 INPUT: *AnnotationSet*
 OUTPUT: *UserSet*
 EXCEPTION: none
 DESCRIPTION: Retrieves all users participating in the creation and update of the provided annotation set. If the annotation set is empty, the result is an empty *UserSet*.
- *getUsers*
 INPUT: *ResourceID*
 OUTPUT: *UserSet*
 EXCEPTION: *NoSuchResourceException* – if resource with the provided ID does not exist.
 DESCRIPTION: Retrieves all users that have annotated this resource.
- *getUsers*
 INPUT: *RatingLow*, *RatingHigh*
 OUTPUT: *UserSet*
 EXCEPTION: *InvalidRatingSpecificationsExceptions* – if the rating interval is not valid.
 DESCRIPTION: Retrieves all users with rating in the specified interval. If one of the range specifiers are omitted, the interval is considered open on that side.
- *getRatedUsers*
 INPUT: *UserID*
 OUTPUT: *UserSet*
 EXCEPTION: *NoSuchUserException* – if there is no user with the provided ID.
 DESCRIPTION: Retrieves all users that this have rated.

- *getRatingUsers*
INPUT: *UserID*
OUTPUT: *UserSet*
EXCEPTION: *NoSuchUserException* – if there is no user with the provided ID.
DESCRIPTION: Retrieves all users that have rated a user with ID *UserID*.

6 Conclusion

In the current deliverable we have defined the specification of the access, search and retrieval methods of semantic content in the INSEMTIVES platform. The collection of retrieval methods are on various levels and concern annotations, entities, co-occurrence and ranking of entity references and annotations, concepts, textual data and metadata. With this set of methods the platform will empower the INSEMTIVES tools to implement the use case scenarios and at the same time be abstracted from the low level RDF-based representation.