Interleaving Reasoning and Selection with Semantic Data

Zhisheng Huang

Department of Artificial Intelligence, Vrije University Amsterdam
De Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands
huang@cs.vu.nl

Abstract. The scalability is a crucial topic for practical applications of the Semantic Web, because of the extremely large scale data on the Web. In this paper we propose a general framework of interleaving reasoning and selection with semantic data. The main idea of the interleaving framework is to use some selectors to select only limited and relevant part of data for reasoning, so that the efficiency and the scalability of reasoning can be improved. In this paper, we examine how the interleaving framework can be achieved within the LarKC platform, a platform for massive distributed incomplete reasoning that will remove the scalability barriers of currently existing reasoning systems for the Semantic Web. We discuss the implementation of variant interleaving approaches within the LarKC platform.
Keywords. Web scale reasoning, Reasoning with Inconsistencies, Dynamic semantic data

1 Introduction

The scalability has become a crucial issue for practical applications of the Semantic Web, because of the extremely large scale data on the Web. Web scale semantic data have the following main features:

- **Infiniteness**. There are extremely large amount of semantic data on the Web. Linked Data have reached the scale of many billion triples. More linked data are expected to grow rapidly in coming few years. Therefore, Web scale data can be considered to be infinitely scalable.
- **Dynamics**. Web scale data are in flux. They are growing extremely rapidly, so that it is hard to know what the clear border of data is.
- **Inconsistency**. Re-using and combining multiple ontologies on the Web is bound to lead to inconsistencies between the combined vocabularies. Even many of the ontologies that are in use today turn out to be inconsistent once some of their implicit knowledge is made explicit. Therefore, infinitely scaled Web data tend to be semantically inconsistent. Moreover, consistency checking of Web scale data is impossible because of the infinite scale.

Because of those features of Web scale data, many traditional notions of reasoning are not valid any more. A solution for Web scale reasoning is to go beyond

traditional notions of absolute correctness and completeness in reasoning. We are looking for retrieval methods that provide useful responses at a feasible cost of information acquisition and processing. Therefore, generic inference methods need to be extended to non-standard approaches. That is the main goal of the LarKC project which develops the Large Knowledge Collider (LarKC), a platform for massive distributed incomplete reasoning that will remove the scalability barriers of currently existing reasoning systems for the Semantic Web.

In this paper, we will propose a framework of interleaving reasoning and selection for semantic data. The main idea of the interleaving framework is to use some selectors to select only limited and relevant part of data for reasoning, so that the efficiency and the scalability of reasoning can be improved. In this paper we will develop various strategies of interleaving reasoning and selection The interleaving framework is inspired by our previous work on reasoning with inconsistent ontologies[1]. PION[1] is a system of reasoning with inconsistent ontologies. However, in this paper we will develop a general framework of interleaving reasoning and selection, so that it can deal with not only reasoning with inconsistent ontologies, but also generic Web scale data. Furthermore, we will explore how the interleaving framework can be achieved within the LarKC platform and discuss various implementation of the interleaving approaches.

The rest of the paper is organized as follows: Section 2 discusses the main problems of web scale reasoning and proposes a general framework of interleaving reasoning and selection with semantic data. Section 3 examines the interleaving framework with inconsistent ontologies and discusses various approaches of interleaving reasoning and selection with the PION framework. Section 4 explores the interleaving approaches within the LarKC platform and discusses the implemented interleaving plug-ins and workflows. Section 6 concludes the paper.

2 Interleaving Reasoning and Selection

2.1 Main Problems

Web scale reasoning is reasoning with Web scale semantic data. As we discussed above, Web scale semantic data are infinite, dynamic, and inconsistent. Those features of Web scale data force us to re-examine the traditional notion of reasoning. The classical notion of reasoning is to consider the consequence relation between a knowledge base (i.e. a formula set Σ) and a conclusion (i.e., a formula ϕ), which is defined as follows:

$$\Sigma \models \phi \text{ iff for any model } M \text{ of } \Sigma, M \text{ is a model of } \phi.$$

For Web scale reasoning, Knowledge base Σ can be considered as an infinite formula set. However, when the cardinality $|\Sigma|$ of the knowledge base Σ becomes infinite and Σ is inconsistent, many notions of logic and reasoning in classical logics, including many existing description logics, which are considered to be

[1] http://wasp.cs.vu.nl/sekt/pion

standard logics for ontology reasoning and the Semantic web, are not valid any more.

It is worthy to mention that classical logics do not limit the cardinality of their knowledge bases to be finite, because the compactness theorem in classical logics[2] would help them to deal with the infiniteness.

The Compactness theorem states that:

(CT) a (possibly infinite) set of first-order formulas has a model iff every finite subset of it has a model,

Or conversely:

(CT') a (possibly infinite) set of formulas doesn't have a model iff there exists its finite subset that doesn't have a model.

That means that given an infinite set of formulas Σ and a formula ϕ, if we can find a finite subset $\Sigma' \subseteq \Sigma$ such that $\Sigma' \cup \{\neg\phi\}$ is unsatisfiable (namely, there exists no model to make the formula set holds), it is sufficiently to conclude that ϕ is a conclusion of the infinite Σ. In other words, the compactness theorem means that in the formalisms based on FOL we can positively answer the problems of the form $\Sigma \models \phi$, by showing that $\Sigma \cup \{\neg\phi\} \models$ contradiction. Thus, we have chances to show (even if Σ is infinite) if we are able to identify a finite subset of Σ (call it Σ') such that $\Sigma' \cup \{\neg\phi\} \models$ contradiction.

However, we would like to point out that the compactness theorem would not help for Web scale reasoning because of the following reason.

For Web scale data, Knowledge base Σ may be inconsistent. Now, consider the problem to answer the form $\Sigma \models \phi$ where Σ is inconsistent. When Σ is inconsistent, a finite subset of Σ (call it Σ') such that $\Sigma' \cup \{\neg\phi\} \models$ contradiction would not be sufficient to lead to a conclusion that $\Sigma \models \phi$, because there might exist another subset of Σ (call it Σ'') such that $\Sigma'' \cup \{\phi\} \models$ contradiction.

2.2 Framework of Interleaving Reasoning and Selection

A way out to solve the infiniteness and inconsistency problems of Web scale reasoning is to introduce a selection procedure so that our reasoning processing can focus on a limited (but meaningful) part of the infinite data. That is the motivation for developing the framework of Web scale reasoning by interleaving reasoning and selection.

Therefore, the procedure of Web scale reasoning by interleaving reasoning and selection consists of the following *selection-reasoning-decicison-loop*:

Namely, the framework depends on the following crucial processes: i) How can we select a subset of a knowledge base and check the consistency of selected data, ii) How can we reason with selected data, iii) how can we make the decision whether or not the processing should be stop. That usually depends on the problem how we can evaluate the answer obtained from the process ii). Our framework is inspired by our previous work in reasoning with inconsistent

Algorithm 1.1. Selection-Reasoning-Loop

repeat
  Selection: Select a (consistent) subset $\Sigma' \subseteq \Sigma$
  Reasoning: Reasoning with $\Sigma' \models \phi$ to get answers
  Decision: Deciding whether or not to stop the processing
until Answers are returned.

ontologies[1]. Since Web scale data may be inconsistent, we can apply the same framework to deal with the problem of Web scale reasoning.

3 Interleaving Reasoning and Selection with Inconsistent Ontologies

In this section, first we make a brief overview of the PION framework, a general framework of reasoning with inconsistent ontologies which is developed in [1,3]. Furthermore, we examine various strategies of interleaving reasoning and selection with inconsistent ontologies within the PION framework.

3.1 The PION framework

Selection functions are central in the PION framework of reasoning with inconsistent ontologies. Such a selection function is used to determine which consistent subsets of an inconsistent ontology should be considered during the reasoning process. The selection function can either be syntactic, e.g., using a syntactic relevance measure, or can be based on semantic relevance, such as using the co-occurrence of terms in search engines like Google[4].

The general strategy for reasoning with inconsistent ontologies is: given a selection function (which is based on a relevance measure), we select a consistent subset from an inconsistent ontology. Then we apply standard reasoning on the selected subset to find meaningful answers. If a satisfying answer cannot be found, we use the selection function to extend the selected set for further reasoning[2]. If an inconsistent subset is selected, we apply "over-determined processing" (ODP)[1]. One of the ODP strategies is to find a maximal consistent subset of the selected set. If the (firstly selected) maximal consistent subset entails the query, the algorithm will return 'yes', otherwise it will return 'no'.

3.2 Syntactic Relevance based Selection Functions

There are various ways to define syntactic relevance between two formulas. Given a formula ϕ, we use $I(\phi), C(\phi), R(\phi)$ to denote the sets of individual names, concept names, and relation names that appear in the formula ϕ respectively.

[2] Namely the relevance degree of the selection function is made less restrictive thereby extending the consistent subset.

In [1], we propose a direct relevance which considers the syntactic existence of a common concept/role/individual name in two formulas.

Two formula ϕ, ψ are directly syntactic relevant, written $\mathcal{R}_{SynRel}(\phi, \psi)$, iff there is a common name which appears both in formula ϕ and formula ψ, i.e.,

$$\langle \phi, \psi \rangle \in \mathcal{R}_{SynRel} \text{ iff } I(\phi) \cap I(\psi) \neq \emptyset \vee C(\phi) \cap C(\psi) \neq \emptyset \vee$$
$$R(\phi) \cap R(\psi) \neq \emptyset.$$

In [5,1], we provided a detailed evaluation of the prototype by applying it to several inconsistent ontologies. The tests show that the syntactic relevance approach can obtain intuitive results in most cases for reasoning with inconsistent ontologies. The syntactic relevance approach works with real-world inconsistent ontologies because it mimics our intuition that real- world truth is (generally) preserved best by the argument with the shortest number of steps; and whatever process our intuitive reasoning uses, it is very likely that it would somehow privilege just these shortest path arguments.

3.3 Semantic Relevance based Selection Functions

The syntactic relevance-based selection functions prefer shorter paths to longer paths in the reasoning. It requires knowledge engineers should carefully design ontologies to avoid unbalanced reasoning path. Naturally we will consider semantic relevance based selection functions as alternatives of syntactic relevance based selection functions. In the following, we will discuss a semantic relevance based section function that is developed based on Google distances. Namely, we want to take advantage of the vast knowledge on the web by using Google based relevance measure, by which we can obtain light-weight semantics for selection functions. The basic assumption here is that: more frequently two concepts appear in the same web page, more semantically relevant they are, because most of web pages are meaningful texts. Therefore, information provided by a search engine can be used for the measurement of semantic relevance among concepts. For PION, we select Google as the targeted search engine, because it is the most popular search engine nowadays. The second reason why we select Google is that Google distances are well studied in [6,7].

In [6,7], Google Distances are used to measure the co-occurrence of two keywords over the Web. Normalized Google Distance (NGD) is introduced to measure semantic distance between two concepts by the following definition:

$$NGD(x, y) = \frac{max\{logf(x), logf(y)\} - logf(x, y)}{logM - min\{logf(x), logf(y)\}}$$

where

$f(x)$ is the number of Google hits for the search term x,

$f(y)$ is the number of Google hits for the search term y,

$f(x, y)$ is the number of Google hits for the tuple of search terms x and y, and,

M is the number of web pages indexed by Google[3].

$NGD(x, y)$ can be understood intuitively as a measure for the symmetric conditional probability of co-occurrence of the search terms x and y.

$NGD(x, y)$ takes a real number between 0 and 1. $NGD(x, x) = 0$ means that any search item is always the closest to itself. $NGD(x, y)$ is defined for two search items x and y, which measures the semantic dissimilarity, alternatively called *semantic distance*, between them.

The semantic relevance is considered as a reverse relation of the semantic dissimilarity. Namely, more semantically relevant two concepts are, smaller distance between them. Mathematically this relation can be formalized by the following equation if the similarity measurement and the distance measurement take a real number between 0 and 1.

$$Similarity(x, y) = 1 - Distance(x, y).$$

In the following we use the terminologies *semantic dissimilarity* and *semantic distance* interchangeably. To use NGD for reasoning with inconsistent ontologies, we extend this dissimilarity measure on two formulas in terms of the dissimilarity measure on the distances between two concepts/roles/individuals from the two formulas. Moreover, in the following we consider only concept names $C(\phi)$ as the symbol set of a formula ϕ to simplify the formal definitions. However, note that the definitions can be easily generalized into ones in which the symbol sets contain roles and individuals. We use $SD(\phi, \psi)$ to denote the semantic distance between two formulas.

In [4], we propose a semantic distance which is measured by the ratio of the distance sum of the difference between two formulas to the total distance sum of the symbols between two formulas.

Definition 1 (Semantic Distance between two formulas).

$$SD(\phi, \psi) = sum\{NGD(C_i, C_j) | C_i, C_j \in (C(\phi)/C(\psi)) \cup (C(\psi)/C(\phi))\}/(|C(\phi)| * |C(\psi)|)$$

Using the semantic distance defined above, we can define a relevance relation for selection functions in reasoning with inconsistent ontologies. Naturally, an easy way to define a direct relevance relation between two formulas in an ontology Σ is to define them as the semantically closest formulas, i.e., there exist no other formulas in the ontology is semantically more close, like this,

$$\langle \phi, \psi \rangle \in \mathcal{R}_{sd} \text{ iff } \neg \exists \psi' \in \Sigma (SD(\phi, \psi') < SD(\phi, \psi)).$$

[3] Currently, the Google search engine indexes approximately ten billion pages.

4 Interleaving Reasoning and Selection within the LarKC Platform

We have implemented various approaches of interleaving reasoning and selection within the LarKC platform[8][4], a platform for massive distributed incomplete reasoning that will remove the scalability barriers of currently existing reasoning systems for the Semantic Web.

4.1 LarKC Platform

In [9], the LarKC architecture has been proposed. This design is based on a thorough analysis of the requirements and considering the lessons learned during the first year of the project. Figure 1 shows a detailed view of the LarKC Platform architecture.

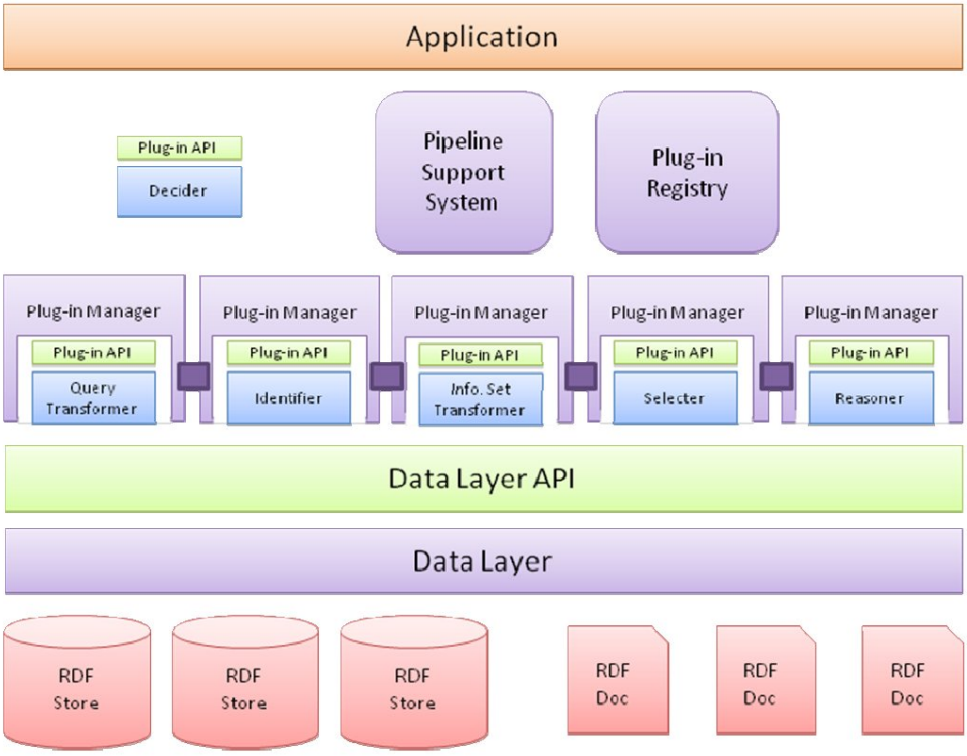

Fig. 1. The LarKC Platform Architecture

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

The LarKC platform has been designed in a way so that it is as lightweight as possible, but provides all necessary features to support both users and plug-ins. For this purpose, the following components are distinguished as part of the LarKC platform:

- **Plug-in API:** defines interfaces for plug-ins and therefore provides support for interoperability between platform and plug-ins and between plug-ins.
- **Data Layer API:** provides support for data access and management.
- **Plug-in Registry:** contains all necessary features for plug-in registration and discovery
- **Workflow Support System:** provides support for plug-in instantiation, through the deployment of plug-in managers, and for monitoring and controlling plug-in execution at workflow level.
- **Plug-in Managers:** provides support for monitoring and controlling plug-ins execution, at plugin level. An independent instance of a Plug-in Manager is deployed for each plug-in to be executed. This component includes the support for both local and remote execution and management of plug-ins.
- **Queues:** provides support for deployment and management of the communication between platform and plug-ins and between plug-ins.

4.2 Variant Interleaving Approaches within the LarKC Platform

We have implemented the following variants of interleaving reasoning and selection within the LarKC platform:

- **DIGPION**. DIGPION is the one in which an external PION reasoner is called via the DIG interface plug-in within the LarKC platform. The main advantage of DIGPION is that we can rely on an externally implemented PION system for interleaving reasoning and selection within the LarKC Platform.
- **SimplePION**. SimplePION is the one in which PION is implemented as a plug-in with some simplified functions, which include the support of standard boolean answers (i.e., either "true" or "false") without using the three-valued answers such as "accepted", "rejected", and "undertermined", which have been proposed in [1].
- **PIONwithStopRules**. PIONwithStopRules is the one in which PION uses some stop rules to decide when it would stop the selection and jump to provide its reasoning result. The idea of using stop rules is inspired by the investigation of human and animal search strategies in ecology and cognitive science. Using stop rules for LarKC has been investigated in LarKC deliverable D4.2.2 [10].
- **PIONWorkflow**. PIONWorkFlow is the one in which PION is designed to be a workflow which uses selection plug-ins and reasoner plug-ins. The main advantage of the scenario of PIONWorkflow is that this approach provides the possibility to use various selectors and reasoners which have been implemented independently from the interleaving framework.

More variants of interleaving reasoning and selection are under development. Furthermore we are now working on the deployment of the implemented plug-ins and workflows to the use studies of the LarKC project and conduct the experiments with large scale semantic data from those use cases for the evaluation of the interleaving framework.

Because of the page limitation, we will not discuss the details of all the plug-ins/workflows above in this paper. In the following we discuss just one workflow approach, i.e., the PIONWorkflow.

5 PIONWorkflow

PIONWorkFlow is the one in which PION is designed as a workflow which uses selection plug-ins, reasoner plug-ins, and a decider for the interleaving processing, as shown in Figure 2. One of the advantages of the PIONWorkflow is that it allows us to use various selectors in the processing.

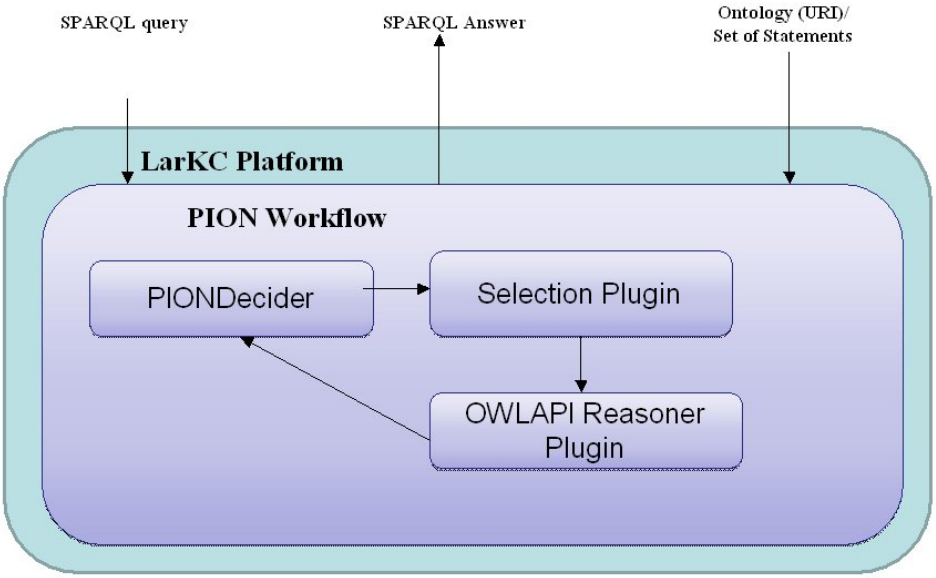

Fig. 2. PIONWorkflow

In the existing implementation of PIONWorkFlow, one can define a workflow which would launch a PION decider. At the beginning of the PION workflow processing, the decider first checks if the the ontology is consistent. If the ontology is consistent, then the decider will start the standard reasoning processing, namely, use the standard OWLAPI reasoner to obtain the result. If the ontology is inconsistent, then the decider will start a non-standard reasoning pro-

cessing, namely an interleaving processing. In the beginning of the interleaving processing, the decider calls the selector $SelectOntologybasedOnQuery$ to select a sub-ontology which is relevant to the query and checks if the selected ontology is consistent. If the selected ontology is consistent, then the decider will call $BasicOWLAPIReasoner$ to reason with the selected ontology and check the result. If the selected ontology is inconsistent, then that means that we cannot find a proper and consistent sub-ontology, then the decider will stop and return an answer. For the SPARQL Ask processing, the system would return "false". If the selected ontology is consistent, then the decider will continue the interleaving processing until the inconsistent sub-ontology is selected or a positive answer is obtained (say, the answer is "true" in the SPARQL Ask processing).

6 Discussion and Conclusions

In this paper, we have proposed a general framework of interleaving reasoning and selection with semantic data. We have investigated various approaches of interleaving reasoning and selection, which include PION, a framework of interleaving reasoning and selection with inconsistent ontologies. We have explored the implementation of variants of PION for interleaving reasoning and selection within the LarKC platform, which includes i) the DIGPION which uses the DIG interface reasoner to call an external PION system, ii) the SimplePION which provides basic implementation of the interleaving of reasoning by an OWLAPI reasoner and selection by syntactic-relevance -based selection functions, iii) the PIONwtihStopRule which uses a set of stop rules in the procedure of interleaving reasoning and selection, and iv) the PIONWorkflow which is designed to be an interleaving workflow of reasoning and selection within the LarKC Platform. We are going to report the experiment of those variant interleaving reasoning and selection for the evaluation of the proposed approaches in the future work.
Acknowledgements: The work reported in this paper was partially supported by the EU-funded LarKC project. We thank Szymon Klarman for his stimulating discussions on the Compactness Theorem and Web scale reasoning.

References

1. Huang, Z., van Harmelen, F., ten Teije, A.: Reasoning with inconsistent ontologies. In: Proceedings of the International Joint Conference on Artificial Intelligence - IJCAI'05. (2005)
2. Dawson, J.: The compactness of first-order logic: From gdel to lindstrom. History and Philosophy of Logic (14) (1993) 15–37
3. Huang, Z., van Harmelen, F., ten Teije, A.: Reasoning with inconsistent ontologies: Framework, prototype, and experiment. In Davies, J., Studer, R., Warren, P., eds.: Semantic Web Technologies: Trends and Research in Ontology-based Systems, John Wiley and Sons, Ltd. (2006) 71–93
4. Huang, Z., van Harmelen, F.: Using semantic distances for reasoning with inconsistent ontolgies. In: Proceedings of the 7th International Semantic Web Conference (ISWC2008). (2008)

5. Huang, Z., van Harmelen, F.: Reasoning with inconsistent ontologies: evaluation. Project Report D3.4.2, SEKT (2006)
6. Cilibrasi, R., Vitanyi, P.: Automatic meaning discovery using Google. Technical report, Centre for Mathematics and Computer Science, CWI (2004)
7. Cilibrasi, R., Vitany, P.: The Google similarity distance. IEEE/ACM Transactions on Knowledge and Data Engineering **19:3** (2007) 370–383
8. 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 International Conference on Semantic Computing. (2008) 524–529
9. Witbrock, M., Fortuna, B., Bradesko, L., Kerrigan, M., Bishop, B., van Harmelen, F., ten Teije, A., Oren, E., Momtchev, V., Tenschert, A., Cheptsov, A., Roller, S., Gallizo, G.: D5.3.1 - requirements analysis and report on lessons learned during prototyping (June 2009) Available from: http://www.larkc.eu/deliverables/.
10. Neth, H., Schooler, L.J., Rieskamp, J., Quesada, J., Xiang, J., Wang, R., Wang, L., Zhou, H., Qin, Y., Zhong, N., Zeng, Y.: D4.2.2 - analysis of human search strategies (September 2009) Available from: http://www.larkc.eu/deliverables/.