

Information Retrieval in the SemanticLIFE Personal Digital Memory Framework

Hanh Huu Hoang, Tho Manh Nguyen

(Invited Paper)

Abstract—Ever increasing capacities of contemporary storage devices inspire the vision to accumulate (personal) information without the need of deleting old data over a long time-span. Hence the target of SemanticLIFE project is to create a Personal Information Management system for a human lifetime data. One of the most important characteristics of the system is its dedication to retrieve information in a very efficient way. By adopting user demands regarding the reduction of ambiguities, our approach aims at a user-oriented and yet powerful enough system with a satisfactory query performance. We introduce the query system of SemanticLIFE, the Virtual Query System, which uses emerging Semantic Web technologies to fulfill users' requirements.

Keywords—Ontology-based Information Retrieval, Digital Memories, SemanticLIFE.

I. INTRODUCTION

TOWARDS the goals of personal information storage and retrieval of all one's data throughout a lifetime, researchers consider continuous archival and retrieval of all media relating to personal experiences including emails, contacts, appointments, web browsing, documents, phone calls, etc. The challenging issues are how to extract useful knowledge from this rich library of information; and how to use this knowledge effectively, how to effectively retrieve memories and knowledge to different kinds of users. The grand challenge is to manage this data, the digital memories, for the benefit of human life and for a life time [8].

The SemanticLIFE project [2] is an effort to come a step closer to a solution for mentioned issues and Vanevar Bush's vision of Memex [5] by providing a general semantic Personal Information Management (PIM) system. The SemanticLIFE user is supported in issuing imprecise queries to retrieve the rich semantic information from his/her historical personal data.

The similar systems focus on back-end issues, i.e. capturing all data sources, integrate and then store them in huge repositories. For this purpose it is necessary to map the ontologies of the various data sources into a common ontology of the system. However, users are confronted with the lack of knowledge concerning the stored information inside the system, and they would formulate ambiguous requests, so that many barriers have to be overcome before the system could deliver the demanded results.

H. H. Hoang is a Doctoral researcher at the Institute of Software Technology and Interactive Systems, Vienna University of Technology, Favoritenstrasse 9-11/188, A-1040 Vienna, Austria (phone: +43.1.58801-18861, fax: +43.1.58801-18861, email: hanh@ifs.tuwien.ac.at).

T. M. Nguyen is a Post-Doctoral researcher at the Institute of Software Technology and Interactive Systems, Vienna University of Technology, Favoritenstrasse 9-11/188, A-1040 Vienna, Austria (email: tho@ifs.tuwien.ac.at)

Beside of presenting the motivation and overview of the SemanticLIFE framework, this paper is aiming at its query system, the *Virtual Query System* with innovative features focused on user query formulation. This query system is based on a front-end approach allowing the user to retrieve information from huge ontology-based repositories in an efficient way. The conception of this query system which is primarily based on the reduction of semantic ambiguities of user query specifications at the very early stage of the retrieving process; and continually guide the user in query process with the *user context-based mechanism*.

The remainder of this paper is organized as follows: a range of projects is currently addressing similar issues are briefly presented in section II. Section III introduces the SemanticLIFE framework and the relevant issues. Details of its 'virtual query system' are pointed out in sections IV. Section V presents the query language for the VQS and an innovative query formulation feature is discussed in section VI.

II. THE STATE OF THE ART

In the area of digital memories research, a lot of work has already been carried out in some major projects. In this section we highlight some of their significant features.

A. MyLifeBits (Microsoft Research)

It is a system for storing all of one's lifetime data on a PC. The guiding principles are: (a) collections and search must replace hierarchy for organization, (b) multiple visualizations, (c) easy annotations (d) the authoring tool should support reuse of external references [11]. As an experiment, G. Bell has captured all his articles, books, cards, etc, and stored them digitally. He is now paperless, and is beginning to capture phone calls, instant message (IM) logs, television, and radio [10]. They have successfully incorporated multiple annotation types, and creation of stories which are helpful for the short term memory.

They are still trying to explore features such as versioning, document similarity ranking and faceted classification. Until now, they were more concerned with functionality, but now the future work is related with user interface (UI), advanced visualization techniques, data mining for search, new capture mode and devices, shared usage, security, privacy and social issues.

B. Haystack (MIT)

Haystack uses ontologies for information management [13]. Its ultimate goal is to provide high-quality retrieval. Primarily

it is designed as a single machine single user tool so as to give a psychological illusion of privacy and security. The guiding design principles are: (a) generic handling of all types of information, (b) flexibility to define additional information types by the user, (c) ability to define the interaction objects and associated operations directly by the user and (d) ability to delegate certain information processing tasks to the agents.

Haystack has a typical three tier architecture [1], i.e., a database layer, service layer and client layer. They have also done some implementation at the Trust layer of Semantic Web by using reification mechanism for RDF¹ storage, and identifier strings as digital signatures during storage of RDF statements. The future developments include ontology conversion, enhanced query mechanisms using machine learning tools to improve retrieval, provide better interface for hybrid search, recommender system based upon user's interests.

C. e-Person (HP)

Developed by HP, an ePerson is a personal representative on the net that is trusted by a user to store personal information, and make it available under appropriate controls for shared working environments. HP's approach is focused on three principals, i.e., social filtering of information by the users themselves, structured knowledge in terms of ontologies mutually agreed upon by the communities, and person-centric instead of being corporate-centric in terms of ownership, vocabularies and scaling [3].

The ePerson infrastructure is designed as a series of layers, transport layer (TCP/UDP and Jabber transports), knowledge-base access layer (remote access to RDF stores and services), Structure layer (modeling of RDF vocabularies using DAML²), Knowledge sources layer (provides specific knowledge services such as classification servers, importing profiles from the history server and a discovery server), and Applications layer (reusable UI components, viewing tools for knowledge based access during development and the SnippetManager application itself).

D. Lifestreams (Yale University)

Lifestreams [9], is an academic project from Yale University. It is a personal store that uses a simple organizational metaphor, a time-ordered stream of documents combined with several powerful operators that replaces many conventional computer constructs (such as named files, directories, and explicit storage). Their work on the client side includes an X-Window client, command line interface and a PDA client.

The motivating ideas were, (a) storage should be transparent, (b) directories are inadequate as an organizing device, (c) archiving should be automatic, (d) the system should provide sophisticated logic for summarizing or compressing a large group of related documents on one screen, (e) reminding should be made more convenient, and (f) personal data should be accessible anywhere and compatibility should be automatic.

¹Resource Description Framework, <http://www.w3.org/RDF/>

²DARPA Agent Markup Language, <http://www.daml.org/>

III. SEMANTICLIFE: DIGITAL MEMORY FRAMEWORK

A. 'SemanticLIFE'

Living systems have different characteristics like self-regulation of processes, reproduction and growth [15]. Nevertheless, the relevant characteristics could be envisioned in a semantic way in personal knowledge management. Ontologies of personal life items grow and reproduce new ones with processes and services. These ontologies include information about our life objects such as documents, persons, places, organizations, events and tasks.

In the physical world, entities are usually interconnected, either by physical or by semantic means; in the latter case, the semantic meaning is added by human interaction (in an abstract sense) with the physical world. Life items in the system proposed in this paper can be understood as information entities (in some cases they are representations of such physical entities) stored according to ontologies in a semantic database, which are connected to other information entities according to their semantic meaning. Also ontologies 'live' in a way, as they develop and modify permanently during the system- and user- lifetime.

Current (Web-) technologies are highly efficient in processing data for human reception; that is, the transformation from data to information, the 'generation of meaning' is up to the human. A great deal of effort has already been made, and work is still going on to represent semantics explicitly on the Web. This is required to give computer systems the capability to enhance preprocessing of huge amounts of data for the user. It becomes more important as the 'awareness radius' of the contemporary knowledge worker and consumer is continuously increasing. This results from the observation, that users do not limit their information search to specific data repositories, like searching for an address or an event in a calendar any longer. The availability of databases under common or similar interfaces (like web-pages) creates the demand to express more complex queries demanding information aggregated from many different systems using different semantic concepts.

The proposed PIM Systems can contribute significantly in overcoming the common inherent human problems such as limited short term memory, memory loss, forgetfulness, high complexity of data etc. Therefore, it is useful for the system to be able to define and capture the user's life-related events and takes or triggers appropriate action(s) for it. This process involves the following sub-processes:

- 1) Capture events and associated information
- 2) Process action associated with events (e.g., in the sense of an active database system)
- 3) Extract Metadata from the event, or allow the user to enrich the data manually with semantic meaning
- 4) Store the data including semantic context as ontology in an efficient manner
- 5) Allow the user to query the data or support the user directly or via associated applications and tools with context-sensitive information or action

The typical usage of such a PIM can be illustrated with two examples: A student searches a book written by a specific

person knowing only a part of the title and the fact, that the author is a graduate of a specific university. The desired result is, e.g., a link to the book provided by an online bookstore, where the student is customer.

A second example: Consider scientists, who work in a specific domain. They might be interested to get into contact with other researchers in the scientific community that (1) share the same interests or have similar problems (2) are publishing in similar conferences and (3) were recently active in the specific field of research (4) and speak a common language. The result of such a query could be the web pages and email addresses of the researchers coming into question.

It is clear that such problems can only be solved by querying a multitude of information resources like web pages, conference journals, scientific databases, email repositories, newsgroups and the like. Moreover, the system needs to ‘understand’ that entities differently labeled are identical in a semantic sense and also need to be able to ‘understand’ and solve specific issues like the fact, that some results are only valid in a specific interval of time or in a specific language and so on.

Additionally as described in [7], the system must be able to adjust to new user features derived from user interactions with the system or from the information being fed. Thus each user may have individual views and navigational possibilities for working with the system. From the technology perspective, new technologies emerge and older ones fade out. If a system has a too tight coupling with some technology, it may become obsolete with the change in technology. A layered approach that provides some extent of separation from the technology is more suitable, making the overall structure still working if there is a change in the technology or even in case of replacement by the newer ones.

B. SemanticLIFE's Architecture

The SemanticLIFE framework is developed on a highly modular architecture provides the basic components for the VQS modules that will be discussed in later sections. SemanticLIFE stores, manages and retrieves the lifetime's information entities of individuals. It enables the acquisition and storage of data while giving annotations to emails, browsed webpages, phone calls, images, contacts, life events and other resources. It also provides intuitive and effective search mechanism based upon the stored semantics.

An overview of the system architecture is depicted in Fig. 1. The whole SemanticLIFE system has been designed as a set of interactive plug-ins that fit into the main application and this guarantees flexibility and extensibility of the SemanticLIFE platform. Communication within the system is based on a service-oriented design with the advantage of its loosely coupled characteristics. To compose complex solutions and scenarios from atomic services from SemanticLIFE plug-ins, the Service Oriented Pipeline Architecture (SOPA)³ has been introduced. SOPA provides a paradigm to describe the system-wide service compositions and also external web services as

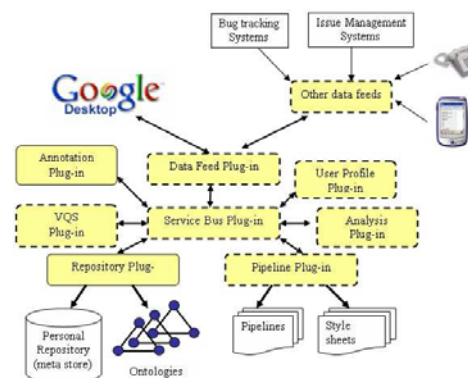


Fig. 1. The SemanticLIFE Framework Architecture

pipelines. SOPA provides some mechanisms for orchestration of services and transformation of results.

Data with user annotation is fed into the system using a number of dedicated plug-ins from variety of data sources like Google Desktop⁴ captured data, communication logs, and other application's metadata. The data objects are passed on by the message handler to the analysis plug-in. This plug-in contains a number of specific analysis plug-ins providing semantic mark-up by applying a bunch of feature extraction methods and indexing techniques in a cascaded manner. The semi-structured and semantically enriched information objects are forwarded to the repository plug-in for an ontologically structured storage, so-called the metastore. A set of query processing and information visualization tools provides the means for information exploration and report generation. The analysis module and metadata extraction capabilities make associations among the lifetime items and lifetime events based on user annotation, user profile and the system ontologies.

C. Information Retrieval in SemanticLIFE

In SemanticLIFE, it is uneasy to define highly structured queries when a multitude of information systems are addressed or the information is only semi-structured. Hence the system must be capable to support user-queries which are formulated with an “imprecise search” terminology by automatically transforming them into more specific queries.

In the SemanticLIFE metastore, the data is already stored in a semantically enriched manner, it provides more powerful imprecise searches. Here, the term “imprecise” has two meanings: firstly, the query processing system has to satisfy imprecisely defined user information needs. Secondly, the target of the user-query is well specified but there are ambiguities in processing the query because of the heterogeneity of the different data sources. Therefore, the system solves these problems during query generation by exploring the system database and ontology. A part of the query module uses system metadata and ontology to provide the user a better awareness of stored data.

The querying process in SemanticLIFE is supported by our Virtual Query System. This mediator system is not only

³JAX Innovation Award 2006 Proposal, <http://www.jax-award.com/>.

⁴Google Desktop, <http://desktop.google.com/>.

capable to deal with the discussed issues above but also reduces the imprecision of the user's requests by offering the user an overview of the relevant stored information. As a result, the user will significantly specify more precise queries on information and data stored in the system.

IV. THE VIRTUAL QUERY SYSTEM

A. Goals

Formulating unambiguous queries is always a demanding task to users as they do not have the overview on the semantics of stored data. The principle of the Virtual Query System (VQS) is to provide an ontology-based virtual information view of the system data. Hereby, the *virtual information* are the metadata extracted from the metastore and delivered to the user after a well-organizing process. The user can clearly specify queries on the "real" data in the repositories when he/she can "be aware of" what is inside of the system.

The VQS also provides predefined query patterns which will be matched with users' query space and the matched ones then will be recommended to the users. In addition, based on a common ontology and the internal analysis (inference, detecting ambiguity and fuzziness of user queries), the VQS refines the user's queries and generates "real" queries against data sources in the metastore.

B. Components

The Virtual Query System consists of six modules, presented in Fig. 2, to deal with the challenging task of a complete Semantic Web query system.

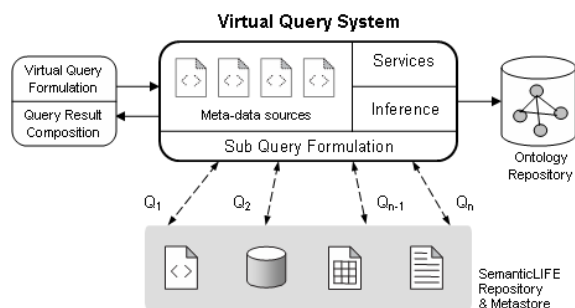


Fig. 2. The Components of the Virtual Query System

1) *Virtual Data Component*: The *virtual data component* (VDC) is a VQS's crucial part. It contains the metadata of storage sources. This module acts as a *virtual data source* to be offered to the user. It enables the user to be aware of the semantics of the data sources stored and to specify more precise queries.

The VQS harvests metadata from data sources in the metadata repository of the SemanticLIFE system. An analysis process and statistical computation are carried out on these meta-data sources to get the semantic information. Then the processed information is stored in this module as a *context ontology* and will be delivered. The features of VQS are very similar to those of a recommender system. Furthermore, this part is also referred as an "image" of the system database in further query processing, so-called the context-based querying.

A query languages is developed for querying the virtual data from this component [12].

The VDC is the main different point of our system in compared with mentioned systems. The behind idea is that when the user is aware of his/her data then he/she could generate more unambiguous requests. This ultimately leads to the reduction of the query refinement process complexity. Additionally, this virtual data component plays as a *context ontology*. This makes the SemanticLIFE system very flexible because the system can adapt a new scenario by simply changing the context ontology.

2) *The Ontology Repository*: The second part of the system is the ontology repository which builds up the core of the VQS. The repository contains the ontologies used in the VQS system such as the global ontology or inference ontology. Following [6] an ontology-driven approach to data integration relies on the alignment of the concepts of a global ontology that describe the domain, with the concepts of the ontologies that describe the data in the local databases. Once the alignment between the global ontology and each of the local ontologies is established, users can potentially query hundreds of databases using a single query that hides the underlying heterogeneities.

3) *Sub-Query Formulation*: Sub-queries formulation is another essential part of the VQS. From the user's initial virtual query, this part parses it into the sub queries (Q_i in the Fig. 2) based on the global ontology for specific data sources. This module does not only transform the virtual query to sub-queries for specific data sources but additionally perform inference on the user's request in order to create more possible sub-queries afterward. After this process, the 'real' queries (RDF queries) for each of the data sources will be generated.

4) *The VQS Services*:

a) *Ontology Mapping*: Mapping service is a mechanism to map local ontologies into a global one. This service deals with new data sources added with their respective ontologies, so that these ontologies are mapped or integrated to the global ontology. In our approach, we do not reinvest to develop a new ontology mapping framework. We use the MAFRA framework [14] for our mapping tasks.

b) *Query Caching*: This service improves the performance of the VQS by caching the queries in a period of time. We distinguish two kinds of caching mechanisms: query caching and result caching. The first addresses the process of generating sub-queries; and the second covers the caching of query results. Both caching types will use the semantic query caching methodology proposed in [18].

c) *Ontology-based Inference*: The inference service provides a basis for the deduction process on the relationships (rules) of concepts of the ontologies specified. Inference tasks are performed on the foundation of the ontologies and the data described by them. This service helps the system to analyze and evaluate the user's virtual queries in the process of generating sub-queries based-on the inference ontology.

5) *Query Refinement*: Query refinement is another important service for our query processing. This is the interactive way for the VQS dealing with user's ambiguous queries based on incrementally and interactively (step-by-step) tailoring a query to the current information needs of a user [17]. VQS's

query refinement service is a semi-automated process: in the refinement process, the user is provided with a ranked list of refinements, which leads to a decrease of some of these ambiguities. In another hand, by exploiting the user's profile, the ontology background, and as well as user's annotation on data, this VQS service supports finding related results.

6) *The Virtual Query User Interface (VQUI)*: The VQUI delivers the virtual data to the user and helps the user to define virtual queries. A set of query patterns is offered to the user. If these patterns do not match the demands, the user can use a query-by-example tool alternatively to write the virtual queries. The VQUI also acts as query results composition which performs the integration and aggregation of the sub-query results and show to the user.

V. THE VIRTUAL QUERY LANGUAGE

A. VQL Aims

Virtual Query Language (VQL) [12] is designed for the VQS. The VQL is used to model the query patterns and generate the virtual queries. The VQL is intended to be a simple query language which supports "semantic" manner from the user's queries. In context of the VQS and SemanticLIFE system:

- VQL helps clients making queries without knowledge of RDF query languages.

- VQL assists users in navigating the system via semantic links or associations, categorized context queries provided in the powerful query operators based-on ontologies.

- VQL simplifies the communication between Query module and other parts as the components asking for information do not need to issues the RDF query statement. This keeps the SemanticLIFE's components more independent.

- VQL enables the portability of the system. The SemanticLIFE and VQS choose a specific RDF query language for the its back-end database. In the future, they probably could be shifted to use another query language, and this change does not affect to the system's parts.

Fig. 3 presents a screenshot of a VQL querying session with the results are displayed in a XML-based form according to [19] which is similar to SPARQL⁵ XML results format [4].

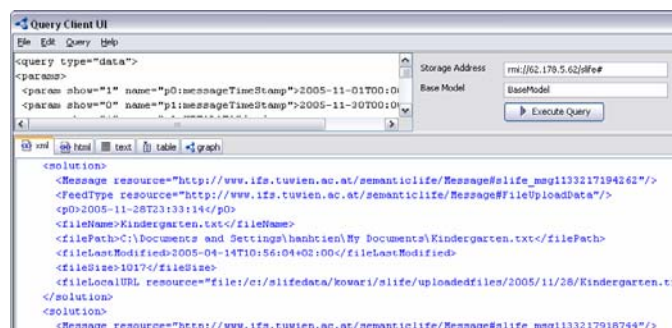


Fig. 3. An example of a VQL query

⁵Simple Protocol And RDF Query Language [16]

B. VQL Operators

The VQL aims at supporting the user in generating queries according to their nature: *minimum of words, maximum of information*. The VQL defines the virtual query operators and they allow the user to simplify the complex queries:

- 1) *GetInstances* retrieves the appropriate information according to the criteria of the specific query.
- 2) *GetInstanceMetadata* assists the user easily retrieve all metadata properties with result instances.
- 3) *GetRelatedData* provides the accessible related information to the current found information, i.e. finding relevant or associated information.
- 4) *GetLinks* operates using the system's ontology and RDF graph pattern to find out the associations/links between the instances and the objects.

By providing these operators⁶, VQL offers a powerful feature of navigating the system by browsing data source by data source, instances by instances based on found semantic associations. More details of the VQL are discussed in [12].

VI. AN INNOVATIVE QUERY FORMULATION

Users are confronting with the query formulation not only in the initial phase but during the querying process when the new information and knowledge come to them. Guiding them formulate the new queries in a correct manner is our aim with the *innovative query formulation* which bases on the user's *querying context*, predefined *query patterns*, and the analysis of resources used in the user's knowledge discovery.

A. Query Templates and Query Maps

The *query templates* are query patterns defined to assist the user in formulating unambiguous queries. The query templates are associated with the appropriate queries, and they have parameters and the related data sources. The templates are classified on the concepts of the VDC's context ontology and the data sources which they are involved with. When a template is in use, the associated virtual query will be loaded and its variables are then replaced by the values. The virtual query is continually edited by the user afterward dependent on the interest of the user.

With the associated data sources and the VDC's context ontology, the query templates create a *query map* to make the connection of them and underlined resources. A query map contains the query templates network which the nodes are the templates and the edges are the relevant concepts and their properties. According to the connections between the templates, when a query template is chosen for making the new queries, the system also recommend the linked templates. Besides, when the user selects properties to formulate his/her query, the system would recommend the relevant templates based on the query map. The connections in the query map are used to determine which templates could be used.

⁶VQL Operators, <http://www.ifs.tuwien.ac.at/~hhanh/VQL/samples/>

B. Context-based Querying

With VDC, the user is supported during the query process by the “context-based” querying feature. According to the context where the user is in, the appropriate query patterns will be proposed by the system. A user’s query context not only contains the queried objects and the querying concepts but they are also associated to each other based on the context ontology.

For example, the context query being applied is about *project management* which contains the concepts of Project, Person, Document, Publication, Partner, Time, Location and so on. The user’s query context could be a graph of *Person, Location, Web search for Project*. In this case a query template such as “*finding a person I have contacted in Vienna in a related project found by Google search engine*” will be proposed.

This feature is applied in the VQS’s interactive interface, in which the user can right clicks on the results objects, instances or virtual data objects and the system will show dedicated templates based on his/her context.

VII. CONCLUSION

In this paper we have presented the SemanticLIFE framework briefly and its query system, the Virtual Query System. SemanticLIFE project aims at building a personal digital diary closely to the Memex’s vision.

The Virtual Query System is an approach of building a complete Semantic Web query system based on a front-end approach. Besides applying current Semantic Web technologies known in the area such as ontology mapping, user annotation and semantic query caching for RDF data, we have designed a query system which aims at a significant complexity reduction in formulating semantic meaningful queries and at the same time aims at a considerable reduction of the number of ambiguous user queries.

ACKNOWLEDGMENT

We are dedicated to all members of the SemanticLIFE team for their excellent contribution to the success of the project.

REFERENCES

- [1] E. Adar, D. Karger, and L. A. Stein, “Haystack: Per-user information environments,” in *Proceedings of the 8th International Conference on Information and Knowledge Management.*, 1999.
- [2] M. Ahmed, H. H. Hoang, S. Karim, S. Khusro, M. Lanzenberger, K. Latif, E. Michlmayr, K. Mustofa, T. H. Nguyen, A. Rauber, A. Schatten, T. M. Nguyen, and A. M. Tjoa, “Semanticle - a framework for managing information of a human lifetime,” in *Proceedings of the 6th International Conference on Information Integration and Web-based Applications and Services*, September 2004.
- [3] D. Banks, S. Cayzer, I. Dickinson, and R. Dave, “The ePerson Snippet Manager: A Semantic Web Application,” HP Laboratories Bristol, Tech. Rep., 2002.
- [4] D. Beckett, “Sparql query results xml format,” W3C Working Draft, Tech. Rep., August 2005. [Online]. Available: <http://www.w3.org/TR/rdf-sparql-XMLres/>
- [5] V. Bush, “As we may think,” *The Atlantic*, vol. 176, no. 1, pp. 101–108, July 1945.
- [6] I. F. Cruz, W. Sunna, and A. Chaudhry, “Ontology alignment for real-world applications,” in *Proceedings of The National Conference on Digital Government - DG.O 2004*, 2004.
- [7] P. Dolog, N. Henze, W. Nejdl, and M. Sintek, “Towards the adaptive semantic web,” in *Proceedings of the International Workshop on Principles and Practice of Semantic Web Reasoning*. LNCS-2901, Springer-Verlag, 2003, pp. 51–68.
- [8] A. Fitzgibbon and E. Reiter, “Memories for life - managing information over a human lifetime,” Grand Challenges in Computing Workshop, UK Computing Research Committee, Tech. Rep., May 2003.
- [9] E. Freeman and D. Gelernter, “Lifestreams: A storage model for personal data,” in *ACM SIGMOD Record, Bulletin 25.1*, March 1996, pp. 80–86.
- [10] J. Gemmel, G. Bell, and R. Lueder, “Mylifebits: Living with a lifetime store,” in *ATR Workshop on Ubiquitous Experience Media*, September 2003.
- [11] J. Gemmel, G. Bell, R. Lueder, S. Drucker, and C. Wong, “Mylifebits: Fulfilling the memex vision,” in *ACM Multimedia '02*, December 2002, pp. 235–238.
- [12] H. H. Hoang and A. M. Tjoa, “The virtual query language for information retrieval in the semanticlife framework,” in *Proceedings of the International Workshop on Web Information Systems Modeling - CAiSE 06*, June 2006, pp. 1062–1076.
- [13] D. Huynh, D. Karger, and D. Quan, “Haystack: A platform for creating, organizing and visualizing information using rdf,” in *Proceedings of International Workshop on the Semantic Web*, 7 2002.
- [14] A. Maedche, B. Motik, N. Silva, and R. Volz, “Mafra - an ontology mapping framework in the semantic web,” in *Proceedings of the 12th International Workshop on Knowledge Transformation*, July 2002.
- [15] J. M. Nicolau, “On thoughts about the brain,” *Brain Processes, Theories and Models*, pp. 71–77, 1996.
- [16] E. Prud’hommeaux and A. Seaborne., “Sparql query language for rdf,” W3C Working Draft, November 2005. [Online]. Available: <http://www.w3.org/TR/rdf-sparql-query/>
- [17] N. Stojanovic, R. Studer, and L. Stojanovic, “An approach for step-by-step query refinement in the ontology-based information retrieval,” in *Proceedings of the IEEE International Conference on Web Intelligence (WI'04)*, 2004, pp. 36–43.
- [18] H. Stuckenschmidt, “Similarity-based query caching,” in *Proceedings of the 6th International Conference on Flexible Query Answering Systems*, 2004.
- [19] D. Wood, P. Gearon, and T. Adams, “Kowari: A platform for semantic web storage and analysis,” in *Proceedings of the 14th International WWW Conference*, 2005.