

Emergent Semantics in Knowledge Sifter: An Evolutionary Search Agent based on Semantic Web Services

Larry Kerschberg, Hanjo Jeong and Wooju Kim*

E-Center for E-Business, Department of Information and Software Engineering,
George Mason University, MSN 4A4, Fairfax, Virginia, 22030-4444
{kersch, hjeong} @gmu.edu, <http://eceb.gmu.edu/>

*Yonsei University, Sinchon-dong, Seodaemun-gu, Seoul Korea 120-749, Korea
{wkim@yonsei.ac.kr}

Abstract. This paper addresses the various facets of emergent semantics in content retrieval systems such as Knowledge Sifter, an architecture and system based on the use of specialized agents to coordinate the search for knowledge in heterogeneous sources, including the Web, semi-structured data, relational data and the Semantic Web. The goal is to provide just-in-time knowledge to users based on their decision-making needs. There are three important factors that can assist in focusing the search: 1) the user's *profile*, consisting user preferences, biases, and query history, 2) the user's *context* to focus on the current activity, and 3) the user's *information space*, in which he may receive the information on specialized hardware with limited bandwidth, implying so that the knowledge must be filtered and tailored to the presentation medium.

Emergent semantics in the context of Knowledge Sifter allow for evolutionary adaptive behavior. We present a meta-model that captures the agent operation and interactions, as well as the artifacts that are created and consumed during system operation. These are stored in a repository, and a collection of emergence agents are presented that perform emergence functions such as data mining for patterns, concept discovery, user preferences tracking, collaborative filtering of user profiles, results ranking, and data source reputation.

1 Introduction

The emergence of the Internet and World Wide Web have made it possible to represent people, places and events via the Web. The evolving Semantic Web allows the Web to be treated as a distributed data-, information- and knowledge space.

The advent of Internet Protocol Version 6 will allow virtually every object to have a fixed IP-address, thereby making it available on the Internet. New technologies such as RFID allow objects to be tracked through complex supply chains. Hand-held devices now

incorporate digital cameras, phones, e-mail, Web browsers, PDAs, GPS, smart cards, and other capabilities to allow users to conduct business transactions using these devices. We are literally immersed in a ubiquitous information space, and the key to managing the *infoglut* is to have effective tools to find, filter, aggregate and present information in a timely fashion for humans, and their proxies, to make informed decisions. This paper addresses the problem of intelligent search services to provide timely, focused and precise knowledge to users *just in time*. We also show how *emergent semantics and emergent behavior* play an important role in supporting the evolution of knowledge to improve the intelligent search agent, which we call Knowledge Sifter.

The concept of just-in-time knowledge management (JIT-KM) [17, 26] is appealing in that the goal is to provide the *right information*, to the *right people*, at the *right time* – just in time – so they can take action based on that information. While the just-in-time concept originated with Toyota in its drive to improve its manufacturing processes, the concept can also be applied to the timely delivery of information. There are a number of inter-related current trends that impact our study of JIT-KM. They are: On Demand Computing, On Demand Business, On Demand Retail, and On Demand Organizations.

On Demand Computing allows users to treat the computing infrastructure as an *information utility*, which can marshal the required resources (computers, storage, etc.) and charge based on usage. Users do not have to be aware of where the computers and storage facilities reside – they are virtual. On Demand Organizations, or Virtual Organizations, can be configured on the fly from existing Web services offered by vendors. The goal is dynamically configure a collection of Web services by searching for candidates, negotiating with service providers for quality-of-service agreements, vetting the selected services, composing them, orchestrating their workflow and managing the virtual organization life-cycle [10, 11].

Both On Demand Computing and On Demand Organizations are based on the *virtualization* of resources and services that are then managed on behalf of users to deliver the desired functions. They are both related to the notions of GRID computing [7] and Semantic Web Services [23, 32]. The amount of meta-data [15, 34] required to manage these virtual environments is considerable.

On Demand Business integrates the enterprise with its suppliers by optimizing business processes and the supply chain to reduce inventories. On Demand Retail treats store shelves as space to be managed by suppliers who are paid when customers actually purchase the merchandise. The main concept is the *integration and interoperation of information* among business partners and suppliers to achieve a high degree of transparency and efficiency among their business processes.

In a recent *New York Times* article [8], Wal-Mart, the world's largest retailer, was using its 460-terabyte data warehouse to monitor worldwide operations in near real-time. They have created an extranet called Retail Link that allows suppliers to see how well their products are selling. Eventually, Wal-Mart will have the capability to conduct *scan-based trading* in which the supplier will own the product until it is scanned for purchase. This will reduce inventory costs for Wal-Mart. Wal-Mart will be requiring its

major suppliers to use RFID tags on its shipments, in order to keep track of inventory as it enters the warehouses.

This example indicates that Wal-Mart is providing virtual shelf space for its suppliers; they are responsible for stocking those shelves and maintaining their inventories in a just-in-time fashion. Wal-Mart collects and uses massive amounts of data to obtain an up-to-the-minute picture of world-wide operations and can make command decisions based on their analysis of the data.

The above trends inform the concept of Just-in-Time Knowledge Management, which might also be called On Demand Knowledge Management. The sheer volume of data and information available to us make it imperative that we be able to sift and winnow through the mountains of data to find those *knowledge nuggets* so crucial to effective decision-making. The Wal-Mart case study motivates another aspect of the *data/knowledge space*, namely that large collections of data can be mined for patterns, rules and constraints. Clearly, these mined patterns are examples of emergent semantics that can be used to improve the knowledge delivered to users. In this paper we wish to explore the notions of just-in-time knowledge management and emergent semantics in the context of a meta-search agent called Knowledge Sifter.

Our goal is to combine the notions of just-in-time knowledge management with emergent semantics [1-3] so as to: 1) improve the semantic formulation of queries posed to KS; 2) retrieve more precise information from heterogeneous sources; 3) store KS artifacts in a knowledge repository that can be mined for emergent concepts and patterns; 4) use collaborative filtering and machine learning techniques to look for community-wide emergence for recommendation during query formulation and results rating; and 5) compose patterns into larger fragments that can be used to tailor KS performance for certain types of queries.

This paper is organized as follows. Section 2 deals with issues associated with requirements and technologies for JIT-KM. Section 3 presents Knowledge Sifter, which is an agent-based search tool based on Semantic Web Services. Section 4 presents a meta-model for Knowledge Sifter that can be used to populate a repository with artifacts such as user preferences, user queries, associated results and user feedback. We also introduce the concept of data-DNA as a metaphor for emergent-, learned- or compiled fragments. Emergence Agents mine emergent concepts and patterns from the repository of Knowledge Sifter artifacts. Also presented are some results related to collaborative filtering and user preference tracking. Section 5 presents our conclusions.

2 Just-in-Time Knowledge Management

The notion of Just-in-Time Knowledge Management (JIT-KM) is that the right information should be available to decision-makers at the right time and in the right place, just in time. This simple concept has very widespread implications for the systems needed to support it. First, how do we determine what is the right information? How do

we know who should receive that information? What is the right format of the information based on the decision-maker's location, context, and type of presentation device? How can we capture and represent the user's preferences, bias, context, and most importantly, his or her information requirements?

We explore these issues within the context of a research project called Knowledge Sifter, which is being conducted at George Mason University's E-Center for E-Business. We show how the Knowledge Sifter architecture can be used for JIT-KM.

2.1 Requirements for JIT-Knowledge Management

In order to deliver JIT-information, we must ensure that the information is timely, authentic, trusted and tailored to the decision-maker's needs. These *knowledge nuggets* are pieces of information that make a quantifiable difference in the decision-making process. Timeliness is important in that out-of-date (stale) information can be irrelevant to the current context and task. Authenticity and trust are related in that the decision-maker should have confidence in the information and the sources that provided it. Finally, there are the issues of *data lineage* and *data provenance*, that is, how was the data processed to derive the information? What is the quality of the original and derived data, and how reliable is the source of the data?

These issues are all crucial for the decision-makers to have confidence in the information products, how they were derived, and the assessment of the quality and reliability of the data provider.

2.2 Technologies for JIT-Knowledge Management

The JIT-information requirements suggest that active technologies are needed to support the timely delivery of JIT services. These include *pull scenarios* in which ad-hoc and standing queries are posed to heterogeneous sources - both internal to the enterprise and external - such as the Internet and World Wide Web. These queries represent items of interest to the decision-maker and evidence substantiating an existing decision scenario would help him to take action. Alternatively, *push scenarios* deliver content to users via syndication such as RSS feeds or by means of specialized subscriptions services. These messages can be searched for content related to the decision-maker's profile and alerts generated and delivered to a preferred device.

An important component of JIT-KM is the use of *meta-data* — data about data — to model and manage the JIT services. Metadata is important in capturing data lineage as information objects are processed throughout the various phases of the activity life cycle. This may include the evolution of user preferences; historical information regarding the results of standing- and ad-hoc queries; the ranking of search results; the authoritativeness of data sources; and the user's perceptions regarding the quality and timeliness of

information provided. We now address these issues in the context of the Knowledge Sifter research.

3 The Knowledge Sifter Architecture

The Knowledge Sifter project, underway at George Mason University, has as its primary goals: 1) to allow users to perform ontology-guided semantic searches for relevant information, both in-house and open-source, 2) to refine searches based on user feedback, and 3) to access heterogeneous data sources via agent-based knowledge services. Increasingly, users seek information outside of their own communities to open sources such as the Web, XML-databases, and the emerging Semantic Web.

The Knowledge Sifter project also wishes to use open standards for both ontology construction and for searching heterogeneous data sources. For this reason we have chosen to implement our specifications and data interchange using the Web Ontology Language (OWL) [5, 36], and Web Services [4] for communication among agents and information sources.

3.1 Knowledge Sifter Agent-Based Web Services Framework

The rationale for using agents to implement intelligent search and retrieval systems is that agents can be viewed as autonomous and proactive. Each agent is endowed with certain responsibilities and communicates using an Agent Communication Language [6]. Recently, Huhns [12] has noted that agents can be thought of a Web services, and this is the approach we have taken to implement the agent community comprising Knowledge Sifter. The family of agents presented here is a subset of those incorporated into the large vision for Knowledge Sifter. This work is motivated by earlier research into Knowledge Rovers [13, 14] performed at GMU. This research is also informed by our previous work on WebSifter, [18-20] a meta-search engine that gathers information from traditional search engines, and ranked the results based on user-specified preferences and a multi-faceted ranking criterion involving static, semantic, categorical and popularity measures.

The Knowledge Sifter architecture [15, 16] may be considered a service-oriented architecture consisting of a collection of cooperating agents. The application domain we are considering is that of Image Analysis. The Knowledge Sifter conceptual architecture is depicted in Figure 1. The architecture has three layers: User Layer, Knowledge Management Layer and Data Layer. Specialized agents reside at the various layers and perform well-defined functions. This collection of cooperating agents supports interactive query specification and refinement, query decomposition, query processing, ranking, as well as result ranking and presentation. The Knowledge Sifter architecture is general and modular so that new ontologies and new information resources can be easily incorporated [27]. The various agents and services are described below.

3.1.1 User and Preferences Agents

The User Agent interacts with the user to elicit user preferences that are managed by the Preferences Agent. These preferences include the relative importance attributed to terms used to pose queries, the perceived authoritativeness of Web search engine results, and other preferences to be used by the Ranking Agent. The Preferences Agent can also learn the user's preference based on experience and feedback related to previous queries.

3.1.2 Ontology Agent

The Ontology Agent accesses an imagery domain model, which is specified in the Web Ontology Language (OWL). In addition, there are two authoritative name services: Princeton University's WordNet [25] and the US Geological Survey's GNIS [35]. They allow the Ontology Agent to use terms provided by the name services to suggest query enhancements such as generalization or specialization.

For example, WordNet can provide a collection of synonyms for a term, while GNIS translates a physical place in the US into latitude and longitude coordinates that are required by a data source such as TerraServer. Other appropriate name and translation services can be added in a modular fashion, and the domain model would be updated to accommodate new concepts and relationships. We now discuss the various sources used by the Ontology Agent.

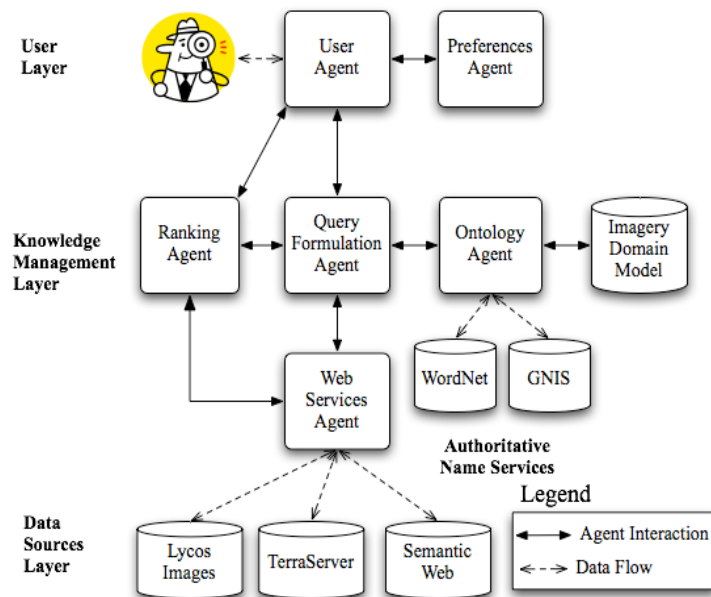


Fig. 1. The Knowledge Sifter Agent-Based Web Services Architecture

3.1.3 Imagery Domain Model and Schema

The principal ontology used by Knowledge Sifter is the Imagery Domain Model, specified using the Web Ontology Language, OWL. A UML diagram of the ontology is provided in Figure 2.

The class Image is defined as having *source*, *content*, and file descriptive *features*. Subcategories of content are *person*, *thing*, and *place*. Since we are primarily interested in satellite and geographic images, the class *place* has two general attributes, *name* and *theme*, together with the subclasses *region* and *address*. The Region is meant to uniquely identify the portion of the Earth's surface where the place is located, either by a *rectangle* or a *circle*. In the case of a rectangle we need two latitude values (*north* and *south*) and two longitude values (*east* and *west*), while to specify a circle we need the *latitude* and *longitude* of its center point, and a *radius*. The *address* of our location is identified by *country*, *state*, *city*, *zip code* and *street*. Each image belongs to a specific online source, the *server*, and has *URI-1* as a unique identifier, together with a secondary *URI-2* for a thumbnail (if any). Some qualitative and quantitative attributes are also modeled as subclasses of the general class *features*, namely *resolution* (in square meters per pixel), *projection* and *datum* (for future GIS utilizations), a *date* range, and image *size* (with *height* and *width* expressed in pixels).

3.1.4 Authoritative Name Services

The two name services are WordNet, developed at Princeton University, which is a lexical database for the English language. When the initial query instance, specifying whether a person, place, or thing, is sent to the Ontology Agent, it then consults WordNet to retrieve synonyms. The synonyms are provided to the Query Formulation Agent to request that the user select one or more synonyms. The decision is communicated to the Ontology Agent which updates the appropriate attribute in the instantiated version of the OWL schema. If the attribute value is the name of a class of type *place* then the Ontology Agent passes the instance to the USGS GNIS.

The second name service is the USGS Geographic Names Information System (GNIS) which is a database of geographic names within the United States and its territories [35]. GNIS was developed by the USGS and the U.S. Board on Geographic Names to meet major national needs regarding geographic names and their standardization and dissemination. It is an integration of three separate databases, the National Geographic Names Data Base, the USGS Topographic Map Names Data Base, and the Reference Data Base. Records within the database contain feature name, state, county, geographic coordinates, USGS Geographic Map name, and others. Other specialized name and translation services can be integrated into Knowledge Sifter and linked to the Domain Model.

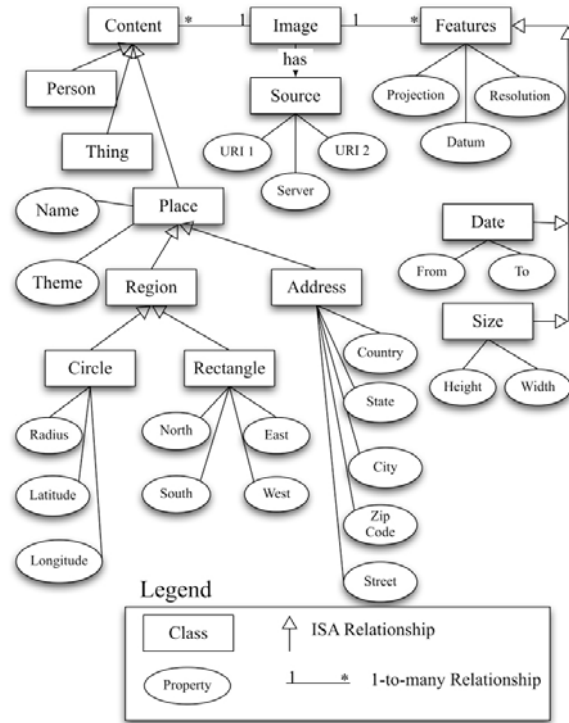


Fig. 2. Ontology Schema in the Unified Modeling Language Notation

3.1.5 Query Formulation Agent

The user indicates an initial query to the Query Formulation Agent. This agent, in turn, consults the Ontology Agent to refine or generalize the query based on the semantic mediation provided by the available ontology services. Once a query has been specified by means of interactions among the User Agent and the Ontology Agent, the Query Formulation Agent decomposes the query into subqueries targeted for the appropriate data sources. This involves semantic mediation of terminology used in the domain model ontology and name services with those used by the local sources. Also, query translation is needed to retrieve data from the intended heterogeneous sources.

For example, if the user specifies the domain of his search as *place*, Lycos and TerraServer will be chosen. In cases of *person* and *thing*, only Lycos will be chosen. In the case of *person* and *thing*, the user is asked to choose a specific meaning from the list retrieved from WordNet, and then the synonym set and hypernym set regarding that particular meaning are retrieved. Synonyms can be chosen as alternate names. Hypernyms

can be used to generalize the user's concept. The terms chosen by the user are used to query Lycos. For example, if the user specifies the concept 'Rushmore' the following synonym set is returned by WordNet:

Rushmore, Mount Rushmore, Mt. Rushmore – (a mountain in the Black Hills of South Dakota; the likenesses of Washington and Jefferson and Lincoln and Roosevelt are carved on it)

In this case, the synonym set {Rushmore, Mount Rushmore, Mt. Rushmore} and the hypernym set {Mountain Peak} are retrieved from WordNet. If user chooses "Mount Rushmore" and "Mountain Peak", two different queries, "Mount AND Rushmore" and "Mountain AND Peak" are posed to Lycos, because the Lycos image search doesn't support the logical connector "OR" in search terms.

In the case of place, the user-selected synonym set and hypernym set are requested from GNIS server using a similar approach, that is, the queries ("Mount AND Rushmore" and "Mountain AND Peak") are posed to the GNIS server in order to collect a list of locations from which the user can choose. The user can specify a state to restrict the GNIS results. After the user chooses one specific location, the name of the location is also used to submit queries to the Lycos server. Concurrently, a query is sent to TerraServer Web service with the appropriate latitude and longitude for the selected place.

In our future research, we will endow the Query Formulation Agent with more rules and policies to help it to make more intelligent decisions about query specification and query optimization. For example, in the case of image databases, a strategy might be to query the image metadata, retrieve and view thumbnails, and then request the collection of selected images. In addition, Knowledge Sifter will have a repository of processed queries, instantiated and annotated according to the OWL schema. This information will be used by the Query Formulation Agent as a Case Base that can be searched and the results reused. For example, a user query might be specified in stages, and the Case Base could be used to retrieve a relevant query processing strategy, send a request to the Web Services Agent and the results returned for user consideration. If needed, the Ontology Agent could assist in query enhancement as described above.

3.1.6 Web Services Agent

The main role of the Web Services Agent is to accept a user query that has been refined by consulting the Ontology Agent, and decomposed by the Query Formulation Agent. The Web Service Agent is responsible for the choreography and dispatch of subqueries to appropriate data sources, taking into consideration such facets as: user preference of sites; site authoritativeness and reputation; service-level agreements; size estimates of subquery responses; and quality-of-service measures of network traffic and dynamic site workload [24].

The Web Services Agent transforms the subqueries to XML Protocol (SOAP) requests to the respective local databases and open Web sources (TerraServer or Lycos) that have

Web Service published interfaces; this is the case for the TerraServer, while Lycos provides an HTTP interface.

3.1.7 Ranking Agent

The Ranking Agent is responsible for compiling the sub-query results from the various sources, ranking them according to user preferences, as supplied by the Preferences Agent, for such attributes as: 1) the authoritativeness of a source which is indicated by a weight – a number between 0 and 10 – assigned to that source, or 2) the weight associated with a term comprising a query.

3.1.8 Data Sources and Web Services

At present, Knowledge Sifter consults two data sources: Lycos Images and the TerraServer. The Lycos server supports keyword-based image search via the web page <http://multimedia.lycos.com>. It makes use of both an image server and external data sources such as web pages for the image search. For a Lycos image search, no advanced search is supported and only conjunctions of terms are used. Therefore, the user cannot specify the image metadata such as *size* or *resolution*, so the results of search are limited. To address these problems the Query Formulation Agent generates a collection of conjunctive and disjunctive queries, while the evaluation and ranking process is left to the Ranking Agent.

The TerraServer is a technology demonstration for Microsoft. There is a Web Service API for TerraServer. TerraServer is an online database of digital aerial photographs (DOQs – Digital Orthophoto Quadrangles) and topographic maps (DRGs – Digital Raster Graphics). Both data products are supplied by the U.S. Geological Survey (USGS). The images are supplied as small tiles and these can be made into a larger image by creating a mosaic of tiles. The demonstrator at terraserver-usa.com uses a mosaic of 2x3 tiles.

Our purpose is to take the ontology-enhanced query and generate specific sub-queries for the TerraServer metadata. The resulting image identifiers and their metadata are wrapped into an instance of our image ontology. And an array of these is returned to the Web Service Agent to compile with other results.

3.2 Knowledge Sifter End-to-End Scenario

Consider the following scenario in which a user wishes to search for the term ‘Rushmore’. This scenario shows how the various agents, name services, and data sources interact in handling a user query.

1. The user provides the User Agent with a keyword query: ‘Rushmore’.
2. The user identifies the term as being a person, place or thing via radio buttons in the query form. The user has chosen ‘Place’.
3. The User Agent passes the query to Query Formulation Agent.

4. The Query Formulation Agent invokes the Ontology Agent to instantiate an OWL schema for the 'Place' with Name = 'Rushmore'.
5. The Ontology Agent chooses a service agent based on the initial query. In this case, it requests from WordNet a list of concepts for 'Rushmore'. WordNet then passes the results back to the Ontology Agent which then passes the results to the User Agent via the Query Formulation Agent for the user decision.
6. The user chooses the 'Mount Rushmore' concept, which has three synonyms ('Rushmore', 'Mt. Rushmore', and 'Mount Rushmore').
7. The Ontology Agent then submits the synonym set to the USGS Geographic Name Information Server and receives a list of candidate geographic coordinates.
8. The list of candidate coordinates is sent to the Query Formulation Agent and the user chooses the desired location.
9. The Ontology Agent then updates the OWL schema instance with the chosen latitude and longitude.
10. The Query Formulation Agent then passes the fully-specified query to the Web Service Agent.
11. The Web Services Agent forwards appropriate sub-queries to both Lycos and TerraServer. The TerraServer and Lycos data sources are queried, and the results are sent back to the Web Services Agent. The results are compiled into new OWL instances that describe image metadata.
12. All results are combined and sent to the Query Formulation Agent.
13. The Query Formulation Agent sends the result sets and the original query to the Ranking Agent for ranking.
14. Within the Ranking Agent the image metadata for each returned item is ranked using the weights and preferences provided by the Preferences Agent. The Preferences Agent maintains the user preferences.
15. The Ranking Agent generates a score for each image result, and returns the scored list to the User Agent.
16. The User Agent then sorts the results by ranking and presents them to the user.
17. The user can then select an item from the list to download and view the image.

We have implemented a proof-of-concept prototype of Knowledge Sifter and this was reported at recent conferences [15, 16].

4 Emergent Semantics in Knowledge Sifter

Our approach to Emergent Semantics (ES) in Knowledge Sifter (KS) is to collect, organize and store significant artifacts created during the end-to-end scenarios for KS such as in Section 3.2. In addition, it is important to collect lineage and provenance information regarding agent interactions among themselves and with external sources. We specify a meta-model for Knowledge Sifter, KSMM, represented in UML and as a

Protégé ontology that can be exported in the Web Ontology Language (OWL). Sections 4.1 through 4.2 present this material in detail.

Emergent semantics processing is performed by agents specializing in different types of semantics associated with Knowledge Sifter. They access the KS Repository that contains *data-DNA fragments* and create *emergent artifacts* that are also stored in the KS Repository. This is discussed in Sections 4.3 and 4.4.

4.1 Knowledge Sifter Meta-Model

The previous sections have described how the cooperative agents and web services support the search for relevant knowledge from both local and open-source data sources. The end-to-end scenario shows how the various agents and sources interact. The OWL schema is instantiated with information regarding query and its various transformations into the final ranked results. In this section we elaborate on this concept by presenting a meta-model of the Knowledge Sifter framework so that relevant information can be captured regarding the *lineage* and *provenance* of all aspects of the search process, from query specification, to query reformulation, web service decomposition, results ranking and recommendation presentation. This includes information on the various Knowledge Sifter activities (managed by agents), the outcomes of those activities, the quality of the ranked results, measures of Web service performance, and the authoritativeness and reliability of data sources.

This meta-model is then used to capture and store both KS data and metadata for future analysis, filtering and mining for emergent properties related to the use of Knowledge Sifter resources. By stepping back and abstracting the agents, classes, their relationships and properties, we can construct the Knowledge Sifter Meta-Model (KSMM) [30]. Figure 3 depicts the UML Static Model for the KSMM. At the top is the Class Agent, which is specialized to those agents in the KS framework, specifically the UserAgent, PreferencesAgent, OntologyAgent, QueryFormulationAgent, RankingAgent and WebServicesAgent. These agents manage their respective object classes, process specifications, and WebServices. For example, the UserAgent manages the User Class, the UserInterfaceScenario, the User PatternMiningAlgorithm, and the WebServices. The User specifies User Preferences that can be specialized to Search Preferences and Source Preferences. The User poses UserQuery that has several QueryConcept, which in turn relates to an OntologyConcept. The Ontology Agent manages both the UserQuery and the OntologyConcept that is provided by an OntologySource. Both OntologySource and DataSource are specializations of Source. Source is managed by the WebServicesAgent and has attributes such as provenance, coverage, access protocol and history. DataSource has attributes such as Quality-of-Service Service-Level-Agreements (QoS-SLAS) and Certificate.

A UserQuery consists of several RefinedQuery, each of which is posed to several DataSource. DataSource provides one-or-more DataItem in response to a RefinedQuery as the QueryResult. Based on the returned QueryResult, the User may provide Feedback

as to the result relevance and other comments. These may impact the evolution of metadata associated with UserPreference, query formulation, data source usage and result ranking. A KSMML has been specified by a Protégé ontology [28] and a screen shot of the main panel is shown in Figure 4.

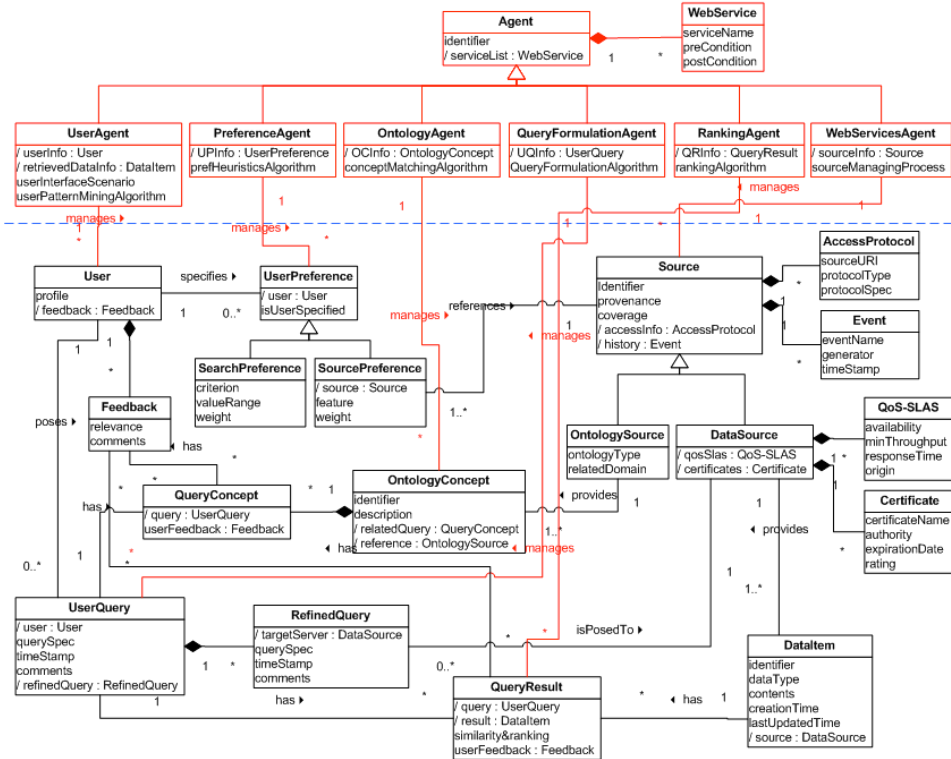


Fig. 3. The Knowledge Sifter Meta-Schema

The KS meta-classes correspond to those of the UML diagram in Figure 3. The Protégé KSMML can also be exported to a Web Ontology Language (OWL) specification. This specification can be consulted via a namespace hyper-link, thus making the agents, which are implemented as Web Services, portable and able to reside on different computers.

4.2 Data/Knowledge Lineage for Emergence Adaptation

The notion of data/knowledge lineage is crucial to the emergence adaptation. The Knowledge Sifter Meta-Model provides a specification of the object classes, their properties, relationships and constraints. It can also specify workflow processes among the agents that handle user requests and the processing of those requests. This concept can be extended to dynamically configure semantic web services for virtual organizations [10, 11]. In this discussion we focus on how the KSMM can address the just-in-time and emergent semantic issues for Knowledge Sifter.

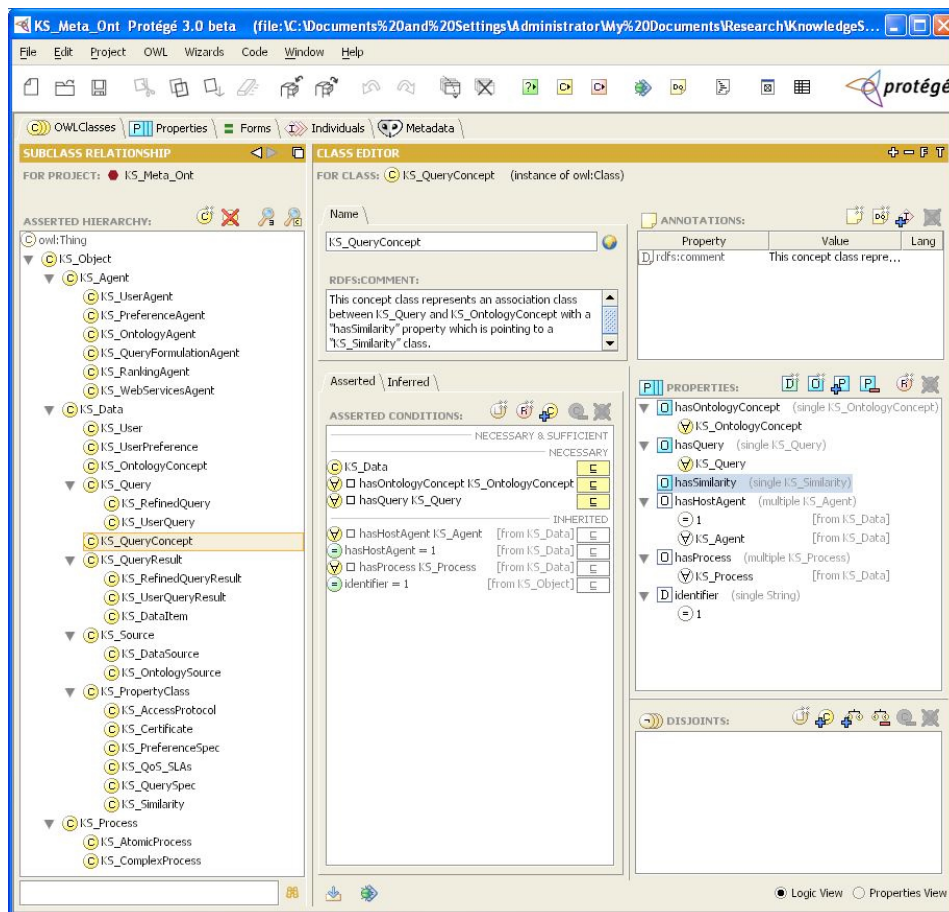


Fig. 4. The Knowledge Sifter Meta-Model (KSMM) in Protégé.

By creating the KSMM we can now capture metadata for the overall search process, from the user's initial query specification, to its refinement with semantic ontological concepts, to its processing and ranking. In addition, we can capture agent attributes, measures of agent interaction to determine overall KS performance.

User feedback and KS performance measures and metrics can be used to evolve the system in several ways that affect emergence. For example, User feedback allows the User Preference Agent to adjust the preferences profile to reflect evolving preferences and biases, to adjust the sources that he prefers and deems to be both of high quality and authoritative. Moreover, as user profiles and preferences are aggregated, we can use data mining and collaborative filtering techniques to discover patterns among groups of users. These learning approaches can be used to make Knowledge Sifter more active in taking advantage of emergent semantics.

Each Knowledge Sifter Agent can adapt to changing query and web services behavior patterns. For example the User Agent can inform the Web Services and Ranking Agents that a particular user's search and ranking preferences have changed, and that sites such as Google and Yahoo! have emerged as favorites and that their results should receive extra support in the rankings. In addition, the Web Services Agent can monitor network traffic and the response times of data sources to determine whether certain site will not be able to deliver their results in a timely fashion, in which case partial results would be provided in JIT-fashion to the user, until the full results could be assembled. In order to refine these concepts, the following section introduces the notion of data-DNA; these are fragments or snippets of data/knowledge that can be composed into larger fragments.

4.3 Data-DNA and Emergent Behavior

The concept of *data-DNA* provides for self-describing meta-data for objects in the context of emergent semantics in Knowledge Sifter. The idea is for every artifact created or used in the course of Knowledge Sifter's operation to be captured, stored and annotated so as to recreate end-to-end processing of user-defined queries. The KSMM provides the domain model by which tagging and indexing can be accomplished. Thus, individual artifacts such as a user query, a semantically enhanced query, results obtained by Web services, and the ranking of results can all be stored, indexed, and annotated. The artifacts, or *data-DNA fragments*, can be composed into larger fragments representing scenarios. The scenarios could be developed off-line and stored in the KS repository, or alternatively, they could be obtained by mining the collection artifacts stored during Knowledge Sifter's normal operation for a collection of users and their queries.

For example, a user might be driving in South Dakota on a family outing (vacation) and would like to visit Mount Rushmore, as per our previous example in section 3.2. He uses his advanced PDA to start the Knowledge Sifter search service. The first request is to find images of Mount Rushmore, and then view a topographical map of the area. Then KS retrieves data-DNA fragments corresponding to the user's context which is family outing (vacation) and proposes one of the following actions according to the workflow

contained in the fragments: 1) Driving instructions with autoloading into the car's GPS, 2) hotel reservations within a 15 mile radius at a Hilton Hotel chain (15 mile radius and Hilton Hotels in user's preference), reservations for dinner for four (from a list of restaurants in preferences), and tickets for tonight's Rushmore Sound-and-Light Show.

The collection and assembly of relevant data-DNA fragments, and their associated scenarios, would be managed by the Ontology Agent using KSMM Ontology. The Ontology Agent would handle both the Imagery Domain Model and the data-DNA fragments stored in the KS repository. The user model, consisting of intent, preferences and context information, would be used to retrieve, assemble and manage the fragments. The Ontology Agent cooperates with the Query Formulation Agent to instantiate the data-DNA fragments with user-specific data for presentation to the Web Services Agent for processing. The Web Services Agent then dispatches the instantiated fragments to appropriate Web services. The Web Services Agent also coordinates the workflow involved in processing the requests. Given the decentralized and distributed nature of these services, we envision that agents will search for external data-DNA fragments via a negotiation process. For example, a task might involve a business process whose provider subscribes to an ebXML protocol [29]. In that case the agent would negotiate the fragment, instantiate it with user data and add it to that user's instantiated data-DNA.

In our family outing example, there are opportunities for parallel execution of transaction fragments. For example, the GPS coordinates of the family can be determined from the cell phone GPS and then loaded into the car's GPS. Concurrently, the restaurant and Sound-and-Light Show reservations can be made, provided that the hotel reservation has first been obtained and guaranteed.

Figure 5 depicts the Knowledge Sifter Emergence Framework with the KS agent architecture on the left, the KS repository in the middle, and the Emergence Agents on the right. During the normal operation of KS, artifacts will be created in response to user requests. These artifacts are stored in the KS repository according to the KS Meta-Model shown in Figures 3 and 4. The Emergence Agents access the KS Repository and perform emergence functions suggested by their names; these new emergent concepts, called data-DNA are stored in the KS Repository and can be used to evolve KS at various levels. The data-DNA can be used to: 1) improve the operation of the KS agents by adapting the algorithms that govern their behavior, 2) enhance the ontological concepts known to the Ontology Agent, 3) provide collaboratively-filtered recommendations to users in query formulation, results selection, and user preference evolution.

In the following sections, we focus on the agents involved in collaborative filtering for emergent ontological concept learning, and emergent user preferences.

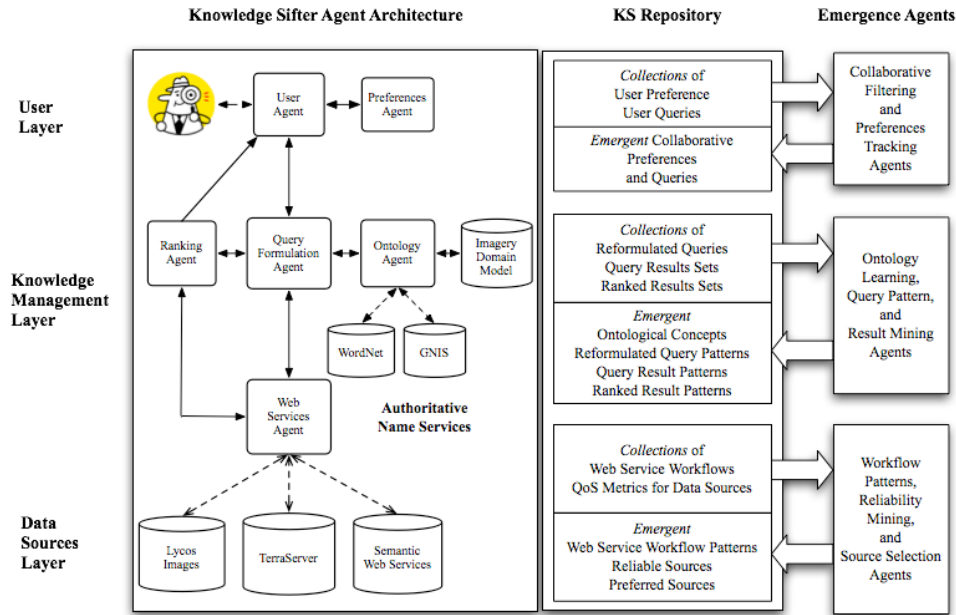


Fig. 5. Knowledge Sifter Emergence Framework

4.4 Collaborative Filtering for KS Emergent Concepts

Recommendation systems provide the right information to the right user by using both content-based filtering and collaborative filtering. One of the best-known information recommendation systems is search engine. Most search engines such as Yahoo and AltaVista use only the content-based filtering to provide web data related to user queries, therefore they have limitations in providing information that incorporates both the user's preferences and an assessment of the quality of the data. One reason for Google's success and popularity is that it combines content-based filtering and collaborative filtering in the PageRank algorithm [31].

In this section we present some results pertaining to the use of collaborative filtering to find emergent concepts, user preferences and recommendations. New concepts from an object retrieved by a particular user query could emerge by mining the explicit and/or implicit user feedback for that object. The user query would be analyzed for the emergent concept of the object. This would increase both the recall and precision of KS search by discovering the new concept and providing user feedback for that concept. Thus, one could combine content-based filtering and collaborative filtering in the KS search process, and then show ordered results by the overall similarity of the ranking.

Assume that we have following statistical values of similarities between user queries and query results from the KS Repository in Figure 5. Actually, KS evaluates result similarity based not only on user queries but also on user preferences. However, we ignore the user preferences for the following data set, in order to obtain new emergent concepts over the entire user community.

(Query, Result, Similarity)
 (q₁, r₁, 0.7)
 (q₁, r₂, 0.2)
 (q₂, r₁, 0.9)
 (q₃, r₁, 0.1)
 (q₃, r₂, 0.8)

The following formula calculates collaborative similarity between a new query and result.

$$sim^{col}(q_i, r_k) = \frac{\sum_{j \neq i} sim(q_i, q_j) \times (sim(q_j, r_k) - \theta_s)}{n(j)} \quad (1)$$

For $j \in \{q_j \mid sim(q_i, q_j) > \theta_q\}$ where j is in q_j s that are only related with r_k , and sim^{col} and sim stand for collaborative similarity and overall similarity respectively, i.e. We could use the above collaborative similarity (query, result) to compensate for determining the overall similarity (query, result) with measured similarity by the KS Rating Agent. The following formula represents the overall similarity. The θ_q represent the threshold for the query similarity to determine how similar queries would be used for calculating collaborative similarity with reducing noises, e.g. one history that has high similarity between a result and a query having low similarity with the new query. The θ_s represents the threshold for determining whether the instance is positive or negative one. The following formula represents the overall similarity.

$$sim(q_i, r_k) = (1 - \alpha) \cdot sim^{con}(q_i, r_k) + \alpha \cdot sim^{col}(q_i, r_k) \quad (2)$$

where α represents a weight for collaborative similarity to determine overall similarity and the sim^{con} represents content-based similarity, i.e., KS evaluates similarity in terms of the subject of the target object by using various ontologies such as WordNet and GNIS. Therefore, the content-based similarity between the query and result could be measured by ontology mapping between the query concept ontology and data resource ontology. The similarity between two queries could be measured in same way as the content-based similarity between query and result is measured.

One of the novel approaches in the above algorithm framework, the initial overall similarity ($\text{sim}(q, r)$) is a combination of both content-based similarity and collaborative similarity, which is based on user feedback, therefore, the final overall similarity could be considered as a *mediated similarity* accounting for the difference between the KS-measured similarity and user-rated similarity. This would compensate for implicit errors in KS measurement and user ratings for the similarities between queries and results. If the user feedback doesn't exist in a certain association between query and result, we could be regarded the KS measured similarity as the initial overall similarity.

4.5 Collaborative Filtering for Emergent User Preferences

Collaborative filtering enables us to suggest recommendations to users using statistical analysis of user patterns. Applying this concept to user profile mining would allow KS to use emergent knowledge provide the right information to right user, i.e., to customize knowledge for a particular user. There are two approaches to the problem: 1) determine user preferences by analyzing user profiles and 2) determine user preferences by analyzing user ratings of query data results.

In general, collaborative filtering recommendation systems filter out items by using only the statistical analysis of human assessments (user ratings) [9, 22], i.e., the systems use implicit correlation between the user ratings and user taste for the recommendations in a given domain of discourse. On the other hand, users could pose content-based queries to find objects they want in a content-based search engine such as KS. Furthermore, KS allows users to define their preferences in terms of some features for a domain object based on the domain ontology. Therefore, we also need to be concerned about influence between the user queries and their preferences and to use this additional information for the recommendation process. If we only consider user relationships between preferences and results in KS, it might cause the system to determine biased, or distorted, preferences because the preferences could be different over the queries, i.e., the preferences could be dependent on the objects that the user wants. To overcome this problem, we use combinations of user preferences and queries to recommend relevant results based on user queries at given the user preferences.

This approach is using the methods in mining a new concept related to a certain object by using user queries as explained in item 1 above. Assume that we have the following statistical values of similarities between three major factors in KS such as user preferences, user queries, and query results. Unlike the examples in item 1, we need to also incorporate the preference because our goal is to recommend related results based on user queries for given user preferences. Therefore, we use a combination of user preferences and queries to evaluate the similarity between the data results.

Consider the table of preference, query result and similarity measures:

(Preference, Query, Result, Similarity)
 (p₁, q₁, r₁, 0.7)
 (p₁, q₁, r₂, 0.2)
 (p₂, q₂, r₁, 0.3)
 (p₂, q₃, r₂, 0.8)
 (p₃, q₄, r₁, 0.8)

The collaborative similarity between a result and a query that is posed by a user having preference would be calculated from following formula:

$$sim^{col}(p_i, q_a, r_x) = \frac{\sum_{j \neq i, b \neq a} (sim(p_i, p_j) \times sim(q_a, q_b) \times (sim(p_j, q_b, r_x) - \theta_s))}{n(j) \times n(b)} \quad (3)$$

For $j \in \{p_j \mid sim(p_i, p_j) > \theta_p\}$, $b \in \{q_b \mid sim(q_a, q_b) > \theta_q\}$

where j is in q_s that are only posed by a user having preference p_i and contains result r_1 in their result set. The $sim(p_i, q_j, r_k)$ represents the overall similarity between the query q_j at given p_i and the result r_k . The θ_p and θ_q represent the threshold for the preference similarity and the query similarity respectively to determine the similar preferences and queries for calculating collaborative similarity. This prevents KS to do biased evaluation by eliminating possibility of including pointless instances, e.g. one that has high similarity between a result and query having low preference and query similarity. The θ_s represents the threshold that determines the instance would be positive or negative one.

The following formula represents the overall similarity.

$$sim(p_i, q_j, r_k) = (1 - \beta) \cdot sim^{con}(p_i, q_j, r_k) + \beta \cdot sim^{col}(p_i, q_j, r_k) \quad (4)$$

where β represents a weight for collaborative similarity to determine overall similarity in this situation.

In order to calculate the similarity between two preferences, we need to consider two kinds of preferences, which are weight preferences and weighted value preferences for each preference criterion because a KS user could define weight preferences for name and location queries evaluation and weighted value preferences for other criteria associated with image data. The weight preferences enable KS to determine which criteria the user considers more important between rating results data and the value preferences represent what value of certain feature user prefers.

The following formula represents the overall similarity between the two preferences.

$$sim(p_i, p_j) = (1 - \gamma)sim^v(p_i, p_j) + \gamma sim^w(p_i, p_j) \quad (5)$$

where γ represents the weight for weight preferences, and sim^v and sim^w represent the similarity between two preferences for the value preferences and the weight preferences, respectively.

The similarity between two weight preferences set could be calculated from following formula, which uses the well-known Euclidean distance function.

$$sim^w(p_i, p_j) = 1 - \frac{\sum_c \sqrt{((p_i(w_c) - p_j(w_c)))^2}}{n(c)} \quad (6)$$

where c represents the criterion of preference, and w_c represents the weight for the criterion c . Then, $P_i(w_c)$ denotes the weight value for a criterion c in a preference i and $n(c)$ represents the number of criterion in preferences. The similarity between two preferences has 1, if the two preferences are identical. But the distance should be 0 in the case of the two preferences being identical, therefore we inverse them. Note that the weight value is normalized one having 0 to 1.

The similarity between two value preferences could be calculated from following formula, which uses weighted Euclidean distance function for each preference criterion. This formula is derived from multipoint queries shown in Mars [33] and Falcon [37], i.e., the value preferences in KS could be treated as the multidimensional queries.

$$sim^v(p_i, p_j) = 1 - \frac{\sum_c w_c \sqrt{\left(\frac{(p_i(v_c) - p_j(v_c))}{\text{Max}(p(v_c)) - \text{Min}(p(v_c))} \right)^2}}{\sum_c w_c} \quad (7)$$

where v_c represents the value for the criterion c . The $p_i(v_c)$ represents the value of the criterion c at a preference p_i . Using Max and Min value for each criterion makes the normalization over the criterion value differences, e.g. size and time criteria in KS preferences for image domain use different measure and range, therefore, we couldn't numerate and normalize the distances not referencing the Max and Min values in collaborative user preferences. In our ongoing research we are extending the concept of

the user preference and results ranking to include multiple attributes and their corresponding weights, similar to our work on WebSifter II [19-21].

5 Conclusions

Knowledge Sifter is an agent-based ontology-driven search engine based on Semantic Web services. We have motivated the need for delivery of timely, focused and accurate information to users in a just-in-time fashion. We have discussed the KS architecture and the important role the semantics plays in all aspects of KS's operation. The KS agents accept a user's initial query, consult the User Preferences Agent, reformulate the query based on user preferences and ontological concepts, decompose it into subqueries handled by the Web Services Agent, and rank the query results according to user preferences, biases and context.

In order to incorporate emergence and evolution into the KS architecture, we develop the KS Meta-Model (KSMM) that allows us to capture, store and mine the artifacts produced and consumed during normal Knowledge Sifter operation. The KSMM is a specification of the agents, activities, and communications of the system's operation. This has been specified in both in UML and Protégé, which automatically generates an OWL specification that can be shared with other services via a namespace. The KSMM schema can be instantiated with actual user queries, their reformulations, query decomposition and processing strategies, Web Services invocations, query result sets, results rankings, and user feedback regarding the relevance of the results to the task at hand.

The Knowledge Sifter Emergence Framework has been presented for discovering, mining and compiling emergent concepts, emergent user preferences, emergent collaborative recommendations, etc. We term these emergent objects as data-DNA and discuss how these can be combined into larger data-DNA fragments.

We focus on Emergence Agents involved in collaborative filtering for emergent ontological concept learning, and emergent user preferences and discuss a novel approach to calculating emergence of content that incorporates both KS-computed similarity and user-rated similarity.

In our work on the precursor to Knowledge Sifter, called WebSifter II, we presented [20] a neural network model for learning user preferences and automatically updating those preferences. We plan to incorporate underlying concepts of the neural network model into the Knowledge Sifter Emergence Framework.

Acknowledgements. This work was sponsored by a NURI from the National Geospatial-Intelligence Agency (NGA). The authors wish to acknowledge the work on the Knowledge Sifter prototype by M. Chowdhury, A. Damiano, S. Mitchell, J. Si, and S. Smith.

References

1. Aberer, K., Catarci, T., Cudré-Mauroux, P., Dillon, T., Grimm, S., Hacid, M.-S., Illarranmendi, A., Jarrar, M., Kashyap, V., Mecella, M., Mena, E., Neuhold, E.J., Ouksel, A.M., Risse, T., Scannapieco, M., Saltor, F., de Santis, L., Spaccapietra, S., Staab, S., Studer, R. and De Troyer, O. Emergent Semantic Systems. in Bouzeghoub, M., Goble, C., Kashyap, V. and Spaccapietra, S. eds. *Semantics for a Networked World, Semantics for the Grid Databases, LNCS 3226*, Springer, Paris, France, 2004, 14-43.
2. Aberer, K., Cudré-Mauroux, P. and Hauswirth, M., The Chatty Web: Emergent Semantics Through Gossiping. in *The Twelfth International World Wide Web Conference, WWW2003*, (Budapest, Hungary, 2003).
3. Aberer, K., Cudré-Mauroux, P. and Hauswirth, M. Start making sense: The Chatty Web approach for global semantic agreements. *Journal of Web Semantics*, 1 (1). 89-114.
4. Chinnici, R., Gudgin, M., Moreau, J.-J. and Weerawarana, S. Web Services Description Language (WSDL) Version 1.2 (<http://www.w3.org/TR/wsdl12/>), W3C, 2002.
5. Fensel, D. Ontology-Based Knowledge Management *IEEE Computer*, 2002, 56-59.
6. Finin, T., Fritzson, R., McKay, D. and McEntire, R., KQML as an Agent Communication Language. in *International Conference on Information and Knowledge Management (CIKM-94)*, (1994), ACM Press.
7. Foster, I., Kesselman, C. and Tuecke, S. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. *International J. Supercomputer Applications*, 15 (3).
8. Hayes, C.L. What Wal-Mart Knows About Customers' Habits, The New York Times, New York, 2004.
9. Herlocker, J.L., Konstan, J.A., Borchers, A. and Riedl, J., An algorithmic framework for performing collaborative filtering. in *Proceedings of the 1999 Conference on Research and Development in Information Retrieval*, (1999).
10. Howard, R. and Kerschberg, L. A Framework for Dynamic Semantic Web Services Management. *International Journal of Cooperative Information Systems, Special Issue on Service Oriented Modeling*, 13 (4).
11. Howard, R. and Kerschberg, L., A Knowledge-based Framework for Dynamic Semantic Web Services Brokering and Management. in *International Workshop on Web Semantics - WebS 2004*, (Zaragoza, Spain, 2004).
12. Huhns, M. Agents as Web Services *IEEE Internet Computing*, July/August 2002.
13. Kerschberg, L. (ed.), *Knowledge Management in Heterogeneous Data Warehouse Environments*. Springer, Munich, Germany, 2001.
14. Kerschberg, L. The Role of Intelligent Agents in Advanced Information Systems. in Small, C., Douglas, P., Johnson, R., King, P. and Martin, N. eds. *Advanced in Databases*, Springer-Verlag, London, 1997, 1-22.
15. Kerschberg, L., Chowdhury, M., Damiano, A., Jeong, H., Mitchell, S., Si, J. and Smith, S. Knowledge Sifter: Agent-Based Ontology-Driven Search over Heterogeneous Databases using Semantic Web Services. in Bouzeghoub, M., Goble, C., Kashyap, V. and Spaccapietra, S. eds. *Semantics for a Networked World, Semantics for the Grid Databases, LNCS 3226*, Springer, Paris, France, 2004, 278-295.
16. Kerschberg, L., Chowdhury, M., Damiano, A., Jeong, H., Mitchell, S., Si, J. and Smith, S., Knowledge Sifter: Ontology-Driven Search over Heterogeneous Databases. in *SSDBM 2004, International Conference on Scientific and Statistical Database Management*, (Santorini Island, Greece, 2004), IEEE.

17. Kerschberg, L. and Jeong, H., Just-in-Time Knowledge Management. in *Third Conference on Professional Knowledge Management*, (Kaiserslautern, Germany, 2005), Springer.
18. Kerschberg, L., Kim, W. and Scime, A., Intelligent Web Search via Personalizable Meta-Search Agents. in *International Conference on Ontologies, Databases and Applications of Semantics (ODBASE 2002)*, (Irvine, CA, 2002).
19. Kerschberg, L., Kim, W. and Scime, A. A Semantic Taxonomy-Based Personalizable Meta-Search Agent. in Truszkowski, W. ed. *Innovative Concepts for Agent-Based Systems*, Springer-Verlag, Heidelberg, 2003, 3-31.
20. Kim, W., Kerschberg, L. and Scime, A. Learning for Automatic Personalization in a Semantic Taxonomy-Based Meta-Search Agent. *Electronic Commerce Research and Applications (ECRA)*, 1 (2).
21. Kim, W., Kerschberg, L. and Scime, A., Personalization in a Semantic Taxonomy-Based Meta-Search Agent. in *International Conference on Electronic Commerce 2001 (ICEC 2001)*, (Vienna, Austria, 2001), Elsevier Science.
22. Konstan, J.A., Miller, B.N., Maltz, D., Herlocker, J.L., Gordon, L.R. and Riedl, J. GroupLens: Applying collaborative filtering to Usenet news. *Communications of the ACM*, 40 (3). 77-87.
23. McIlraith, S.A., Son, T.C. and Zeng, H. Semantic Web Services *IEEE Intelligent Systems*, 2001, 46-53.
24. Menascé, D.A. QoS Issues in Web Services *IEEE Internet Computing*, Nov/Dec 2002, 72-75.
25. Miller, G.A. WordNet a Lexical Database for English. *Communications of the ACM*, 38 (11). 39-41.
26. Morikawa, R. and Kerschberg, L., MAKO-PM: Just-in-Time Process Model. in *Professional Knowledge Management: Workshop on Information Just-in-Time*, (Kaiserslautern, Germany, 2005), Springer.
27. Morikawa, R. and Kerschberg, L., MAKO: Multi-Ontology Analytical Knowledge Organization based on Topic Maps. in *Fifth International Workshop on Theory and Applications of Knowledge Management*, (Zaragoza, Spain, 2004).
28. Noy, N.F., Sintek, M., Decker, S., Crubezy, M., Fergerson, R.W. and Musen, M.A. Creating Semantic Web contents with Protege-2000 *IEEE Intelligent Systems*, 2001, 60-71.
29. OASIS. ebXML, <http://www.ebxml.org/>, 2004.
30. ObjectManagementGroup. Meta Object Facility (MOF) Specification Version 1.3, 2000.
31. Page, L., Brin, S., Motwani, R. and Winograd, T. The PageRank Citation Ranking: Bringing Order to the Web *Tech Report Series*, Stanford University; Digital Library Technologies Project, 1998.
32. Paolucci, M. and Sycara, K. Autonomous Semantic Web Services *IEEE Internet Computing*, Sept - Oct 2003, 34-41.
33. Porkaew, K., Chakrabarti, K. and Mehrotra, S., Query refinement for content-based multimedia retrieval in MARS. in *Proceedings of ACM Multimedia Conference*, (1999).
34. Pouchard, L., Cinquini, L., Drach, B., Middleton, D., Bernholdt, D.E., Chanchio, K., Foster, I.T., Nefedova, V., Brown, D., Fox, P., Garcia, J., Strand, G., Williams, D., Chervanek, A.L., Kesselman, C., Shoshani, A. and Sim, A., An Ontology for Scientific Information in a Grid Environment: the Earth System Grid. in *CCGRID 2003*, (2003), 626-632.
35. USGS. USGS Geographic Names Information System (GNIS), <http://geonames.usgs.gov/>.
36. W3C. OWL Web Ontology Language Overview, <http://www.w3.org/TR/owl-features/>. McGuinness, D.L. and van Harmelen, F. eds., W3C, 2003.
37. Wu, L., Faloutsos, C., Sycara, K. and Payne, T., Falcon: Feedback adaptive loop for content-based retrieval. in *Proceedings of VLDB Conference*, (2000).