Available online at www.sciencedirect.com
SCIENCE DIRECT®
http://www.cogsci.princeton.edu/~wn/.
ELSEVIER
Information Processing and Management 41 (2005) 891–908
INFORMATION PROCESSING & MANAGEMENT
www.elsevier.com/locate/infoproman

Enhancing semantic digital library query using a content and service inference model (CSIM)

Su-Hsien Huang [a,*], Hao-Ren Ke [b,1], Wei-Pang Yang [a,1]

[a] *Department of Computer and Information Science, National Chiao Tung University, 1001, Ta-Hsueh Road, Hsinchu, Taiwan, ROC*
[b] *Library of National Chiao Tung University, 1001, Ta-Hsueh Road, Hsinchu, Taiwan, ROC*

Received 2 August 2002; accepted 13 April 2004
Available online 15 June 2004

Abstract

Although digital library (DL) information is becoming increasingly annotated using metadata, semantic query with respect to the structure of metadata has seldom been addressed. The correlation of the two important aspects of DL—content and services—can generate additional semantic relationships. This study proposes a content and service inference model (CSIM) to derive 15 relationships between content and services, and defines functions to manipulate these relationships. Adding the manipulation functions to query predicates facilitates the description of structural semantics of DL content. Moreover, in search for DL services, inferences concerning CSIM relationships can be made to reuse DL service components. Highly promising with experimental results demonstrates that CSIM outperforms the conventional keyword-based method in both content and service queries. Applying CSIM in DL significantly improves semantic queries and alleviates the administrative load when developing novel DL services such as DL query interface, library resource-planning and virtual union catalog system.

Keywords: Semantic query; Content and service inference model; Digital library; Information retrieval; Metadata

1. Introduction

Rapid developments in information technology have accelerated worldwide access to information. A well-designed architecture is required to coordinate effectively and efficiently the dissemination of a large amount of information over the Internet (Grossman, Qin, & Xu, 1995; Monch & Drobnik, 1998; Nikolaou & Marazakis, 1998). Digital libraries (DL) have received considerable attention in recent years. A DL represents an Internet-based architecture that can access various kinds of information from anywhere. One

* Corresponding author. Tel.: +886-3-5131554; fax: +886-3-5718925.
E-mail addresses: sshuang@cis.nctu.edu.tw (S.-H. Huang), claven@lib.nctu.edu.tw (H.-R. Ke), wpyang@cis.nctu.edu.tw (W.-P. Yang).
[1] Tel.: +886-3-5131554; fax: +886-3-5718925.

0306-4573/$ - see front matter © 2004 Elsevier Ltd. All rights reserved.
doi:10.1016/j.ipm.2004.04.012

major DL activity is to find information; however, many DL use keyword-based searches, which rely on keyword matching and constitute non-semantic means of retrieving information. Keyword-based searches do not consider the multiple senses of a query term; for instance, the sense of the query term "JAVA" is ambiguous because the query system cannot distinguish whether the user's interest is in coffee, a programming language or an island in Indonesia. Therefore, digital libraries that employ typical keyword-based searches have begun to be adapted to allow information to be retrieved more semantically.

A response to a semantic query attempts to determine the precise meaning of a query term according to context. Substantial work has been conducted on automatically determining the correct sense of a polysemous word, including on referencing machine-readable dictionaries such as WordNet (Allan & Raghavan, 2002; Miller et al., 2002). Another way of semantically retrieving information is to add related concepts (Lee, Kim, Kageura, & Choi, 2002) by referring to external auxiliaries, such as computing co-occurrences of terms (Chen, Chung, Marshall, & Yang, 1998); discovering implicit semantics using latent semantic analysis (Kolda & O'Leary, 1998), or analyzing corpuses (Gauch & Wang, 1999). A semantic query can also be performed by collaborative information-filtering methods, including data mining or clustering usage profiles of users with the same interest (Dai, 2001; Mostafa, Mukhopadhyay, Lan, & Palakal, 1997; Wu, 2001).

The senses of query terms obtained from external information (like metadata) support semantic queries. In digital libraries, abundant metadata extend semantic queries from a structural perspective. For instance, VUCS determines which fields should be retrieved and integrated among heterogeneous data formats (such as for example, Dublin Core, MARC, or other canonical formats) to retrieve information from a virtual union catalog system (VUCS). A human being is normally required to judge which schemas contain the same structure and semantics. Data in different formats may include synonyms or homonyms in their attributes, so human intervention makes the process difficult and slow. Therefore, automatically exploiting comprehensive semantics by the structural relationship is essential to identify the heterogeneity of the schema and expand the semantics of metadata. Although some popular markup languages like XML have a namespace facility to elucidate structural information, a mechanism for identifying the semantics of the structure among different attributes is unavailable. MARIAN (Goncalves, France, & Fox, 2001) DL searches in an object-oriented fashion. Data are modeled as classes and relationships as weighted links to represent structural relationships. In the MARIAN approach, the structural relationships among the content are derived from the class hierarchy. Heterogeneity is evaluated by computing the weights of corresponding links.

Sharing metadata architecture in distributed digital libraries drives the use of metadata to expand semantic relationships (Blanchi & Petrone, 2001). Open digital library (Fox, Suleman, & Luo, 2002), INRIA (Abiteboul, Benjelloun, & Milo, 2002) and 5SL (Goncalves & Fox, 2002) apply metadata to describe, derive and federate services in distributed DL. The first step in conducting a semantic query in such distributed DL is to determine which services must execute the query. An individual service may not along suffice to response completely to the entire query, so required information must be collected from various services, or a single task must be executed separately by independent services. A resource-planning facility is required to generate a reasonable plan of queries in series to solve this problem. Considering the metadata of services, the induction process is related to the compatibility between the interfaces and the capabilities. Accordingly, comprehensive relationships among services must be derived to execute semantic queries.

Considerable attention has been paid to DL architecture's enabling: DL queries across distributed DL services using metadata (Paepcke et al., 1996), and managing interaction between the two most important elements of digital libraries—content and services (Lynch & Garcia-Molina, 1995; Monch & Drobnik, 1998). Content represents all the materials stored in a digital library, including texts, images and videos. A service is an application that interacts with users via an interface and can convert content into specific formats. Conventional keyword-based semantic queries cannot clarify two critical elements of DL—content schemas and service capabilities. Content schemas determine whether two pieces of content have the same format. Service capabilities are distinguished by comparing the functions of services. Metadata

support semantic queries on content schemas and service capabilities in many ways. First, metadata that describe content schema and service capabilities have comprehensive semantics regarding content and services. These semantics benefit the provision of accurate information in response to DL queries. Second, relationships between DL content and services can be derived from semantic information embedded in metadata; manipulating these relationships yields further semantics associated to the metadata. Third, metadata can be easily stored and indexed to support retrieval, since they have a formal structure. Metadata that describe semantic information about DL content and services motivates the derivation of semantic relationships among metadata. Given content and services, several questions may be raised. Does one type of content have the same format as another? Can two services perform the same task? Is one type of content produced (or manipulated) by a service? A model that formalizes the relationships between content and services is required to answer these questions.

The work seeks to formulate the structural relationships among metadata concerning DL content and services, to assist the extension of semantic DL queries. This work proposes a content and service inference model (CSIM) to elucidate the interaction of metadata, in terms of the structural relationship. CSIM defines 15 relationships between DL content and services. Using CSIM, the result of a DL query is extended by embedding relationships into the query predicates. For example, for a user who wants to retrieve content with the format "Dublin Core", CSIM returns content with the format "Dublin Core" and derives the content with other relationships, such as that which can be translated into "Dublin Core" (the "Translatable" relationship), and that which contains the same semantics but with different formats (the "Homonymous" relationship). Section 4.1 presents an example of a DL developer who desires to develop a virtual union catalog system and requests the functionality of the intended service using CSIM. The response recommends a list that connects existing services required to perform the VUCS service; these are an extracting service that extracts data from structured documents, a translation service that translates a native data format into the canonical one and an integration service that combines distributed extracting services. The recommendation can be implemented using DL componentization technology (Suleman & Fox, 2002). Restated DL designers can develop a new service that involves existing service components and translation rules, to yield the derived service list. In this manner, CSIM reuses DL components and reduces the administrative load to maintain a DL. Applying CSIM to a digital library makes DL queries semantic and effective. Moreover, CSIM can be applied to DL query interfaces, virtual union catalog systems and library resource-planning systems. A series of experiments were conducted to demonstrate that CSIM outperformed in both content and service queries because of the extended CSIM relationships.

The rest of this paper is organized as follows. Section 2 formally defines content and services in digital libraries, and analyzes the relationships between them. The algorithms that manipulate these relationships are also defined. Section 3 elaborates semantic queries that leverage CSIM to retrieve DL content and services. Section 4 presents the experiments that elucidate the feasibility of CSIM, and demonstrates a prototype system. Section 5 draws conclusions and highlights areas for future work on CSIM.

2. Content and service inference model (CSIM)

Content and services are two integral aspects of digital libraries. Content is the materials stored in digital libraries, which can be produced and processed by services. The types of content include Web pages, library holding records, and multimedia data (like texts, images, and videos). A service is a software application that transforms one type of content item into another. A service possesses specific functions and interacts with users via input and output interfaces. In this paper, both services and content are represented via metadata to facilitate the interaction between them. This paper proposes a novel framework called the content and service inference model (CSIM). CSIM infers relationships between content and services from

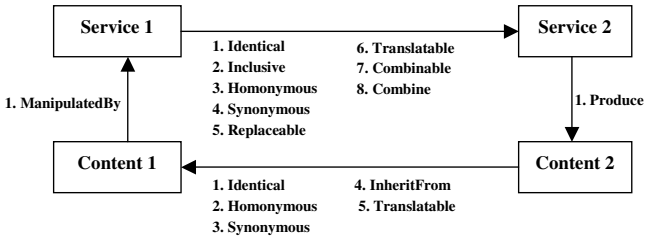

Fig. 1. Relationships between content and services.

their metadata. CSIM can raise DL queries to a semantic level by using content semantics, service capabilities, and the relationships between content and services.

Typically, the metadata of content includes an identifier, data schema, presentation format, and a set of semantic description items. The metadata of a service includes the identifier and a statement about its capabilities. Section 2.1 formally defines the metadata of content and services. A total of 15 relationships between content and services are defined, as illustrated in Fig. 1. The relationships are proposed according to the structure of metadata defined in Section 2.1 with considering reasonable semantic auxiliaries (five semantic tables defined in Section 2.1). These relationships are directional and categorized into four types:

Service to Service. Two services can relate to each other according to their capabilities and input/output interfaces. Eight relationships of this type are defined—*Identical*, *Inclusive*, *Homonymous*, *Synonymous*, *Replaceable*, *Translatable*, *Combinable*, and *Combine*. For example, two services are "*Synonymous*" if they possess the same capabilities but have different input/output interfaces; one service is "*Inclusive*" of another if it contains more capabilities than the latter.

Content to Content. Two pieces of content relate to each other based on their semantics and schema elements. Five relationships of this type are defined—*Identical*, *Homonymous*, *Synonymous*, *InheritFrom*, and *Translatable*. For example, two pieces of content are "*Identical*" if they have identical semantics and format.

Service to Content. A service can produce content. One relationship of this type is defined—*Produce*. For example, a WebPAC system may produce a data set in the Dublin Core format.

Content to Service. Content can be produced by a service. One relationship of this type is defined—*ManipulatedBy*. For example, various data sets can be manipulated by a virtual union catalog system to produce an integrated data set.

The total 15 relationships in CSIM are proposed after thoroughly examining all possible relationships between content and services under the CSIM data model and hypothesis. Section 2.1 formally defines content, services and their ingredients, and Section 2.2 formally defines these relationships.

2.1. Basic definition

In the following, the metadata of content and services [2] is formally defined.

Definition 1 (*Content*). A piece of Content, *C*, is a quadruple {*Id*, *Schema*, *Presentations*, *Semantics*} (McCray, Gallagher, & Flannick, 1999) where,

[2] Hereafter, "content" and "service" are used as shorthand for "the metadata of content" and "the metadata of service" respectively.

1. *Id* is the identifier of *C*.
2. *Schema* is the schema of *C*. *Schema* is a quadruple {*Ent*, *Incs*, *Attributes*, *Associations*}, where
   2.1. *Ent* ⊆ Names is the name of a content schema.
   2.2. *Incs* ⊆ (Names, Names). Each pair $(e_1, e_2) \in Incs$ indicates that e_1 is a subtype of e_2. *Incs* is stored in *CIT* (defined below) and assumed to be acyclic.
   2.3. *Attributes* ⊆ Names is the set of attribute names.
   2.4. *Associations* ⊆ (Association_name, Entity_name1, Entity_name2, Cardinality1, Cardinality2) is the association set, which indicates the cardinality between two entities.
3. *Presentations* ⊆ Names is the set of the presentation interfaces of content *C*.
4. *Semantics* ⊆ CST is the set of semantics of content *C*. CST is defined below.

Definition 2 (*Service*). A Service, *S*, is a quadruple {*Id*, *Capabilities*, *Outputs*, *Inputs*}, where

1. *Id* is the identifier of *S*.
2. *Capabilities* ⊆ SCT is the set of service capabilities. SCT is defined below.
3. *Outputs* ⊆ Names are the output schemas of service *S*.
4. *Inputs* ⊆ Names are the input schemas accepted by *S*.

Additional data structures are required for the formal definition of CSIM.

1. *Content Semantics Table (CST)*. CST contains ontological terms to identify the semantics of content. Dublin Core and MARC are two examples of ontological terms in CST. The ontology in CST is hierarchical; an ascendant term covers the semantics of a descendent term.
2. *Content Inheritance Table (CIT)*. CIT maintains the schema hierarchy of content (*Incs* attribute of *Schema* in Definition 1). For example, the fact that schema *A* is a subtype of schema *B* can be represented as (A, B) in *CIT*.
3. *Service Capability Table (SCT)*. SCT contains ontological terms to define possible service functionalities. The ontology in SCT is hierarchical; an ascendant term has more general capability than a descendent term. For example, a service that uses CORBA to implement a virtual union catalog system will have two capabilities—*CORBA_Distributed_System* and *Virtual_Union_Catalog_System*, where *CORBA_Distributed_System* and *Virtual_Union_Catalog_System* are the descendent terms of *Distributed_System* and *Catalog_System* respectively.
4. *Translation Rule Table (TRT)*. TRT stores the rules for translating between two pieces of content. A rule *R* for translating between two pieces of content is expressed as a quadruple: {*Id*, *FromSchema*, *ToSchema*, *Rules*}, where
   1. *Id* is the identifier of *R*.
   2. *FromSchema* ⊆ Names is the name of the source schema.
   3. *ToSchema* ⊆ Names is the name of the target schema.
   4. *Rules* ⊆ (FromAttributeName, ToAttributeName, TranslationRule) are the rules for translating between specific attributes.
5. *Content and Service Repository (CSR)*: CSR is a repository that stores the metadata of content and services, including the access methods for the CSIM metadata framework to retrieve content and services.

2.2. Relationships between content and services

As mentioned above, four types of 15 directional relationships between content and services exist—content to content, service to service, content to service, and service to content. This section formally explicates each relationship.

Definition 3 (*Content to Content*). Given two pieces of content $C_1 = \{Id_1, Schema_1, Presentations_1, Semantics_1\}$ and $C_2 = \{Id_2, Schema_2, Presentations_2, Semantics_2\}$, the possible relationships between C_1 and C_2 are as follows.

1. $Identical(C_1, C_2)$ iff $Schema_1 = Schema_2$ and $Semantics_1 = Semantics_2$.
2. $Homonymous(C_1, C_2)$ iff $Schema_1 = Schema_2$ but $Semantics_1 \neq Semantics_2$.
3. $Synonymous(C_1, C_2)$ iff $Schema_1 \neq Schema_2$ but $Semantics_1 = Semantics_2$.
4. $InheritFrom(C_1, C_2)$ iff $(Schema_1, Schema_2) \in Incs_2$.
   C_1 and C_2 have the InheritFrom relationship if and only if the schema pair $(Schema_1, Schema_2)$ exists in CIT. Namely, the schema of C_1 inherits from C_2. Given a sequence of content C_1 to C_n, such that any two consecutive pieces of content have the InheritFrom relationship, these pieces of content exhibit the transitive property.
5. $Translatable(C_1, C_2)$. C_1 and C_2 have the *Translatable* relationship if C_1 can be translated into C_2 by means of specific translation rules. In other words, C_1 and C_2 are translatable if and only if there exists a translation rule $T \in TRT$ such that $T.FromSchema = Sch_1$ and $T.ToSchema = Sch2_2$. Moreover, given a sequence of content C_1 to C_n, such that any two consecutive pieces of content have the Translatable relationship, these pieces of content exhibit the transitive property.

Definition 4 (*Service to Service*). Given two services $S_1 = \{Id_1, Capabilities_1, Outputs_1, Inputs_1\}$ and $S_2 = \{Id_2, Capabilities_2, Outputs_2, Inputs_2\}$, the possible relationships between S_1 and S_2 include the following.

1. $Identical(S_1, S_2)$ iff $Capabilities_1 = Capabilities_2$.
2. $Inclusive(S_1, S_2)$ iff $Capabilities_1 \subseteq Capabilities_2$.
3. $Homonymous(S_1, S_2)$ iff $Outputs_1 = Outputs_2$, $Inputs_1 = Inputs_2$, but $Capabilities_1 \not\subset Capabilities_2$.
4. $Synonymous(S_1, S_2)$ iff $Outputs_1 \neq Outputs_2$ or $Inputs_1 \neq Inputs_2$, but $Identical(S_1, S_2)$.
5. $Replaceable(S_1, S_2)$ iff $Inclusive(S_1, S_2)$, $Outputs_1 = Outputs_2$ and $Inputs_1 = Inputs_2$.
6. $Translatable(S_1, S_2)$ iff $Inclusive(S_1, S_2)$, $\exists T_1, T_2, T_3, T_4$ in TRT such that $Translate(Outputs_1, \{T_1\}) = Translate(Outputs_2, \{T_2\})$ and $Translate(Inputs_1, \{T_3\}) = Translate(Inputs_2, \{T_4\})$.
7. $Combinable(S_1, S_2)$ iff $Inputs_1 = Outputs_2$.
8. $Combine(S_c, \{S_i\})$ combines a set of services $\{S_i\}$ $(1 \leqslant i \leqslant n)$, $\forall i$ $Combinable(S_i, S_{i-1})$ (that is, $Outputs_1 = Inputs_2$, $Outputs_2 = Inputs_3, \ldots, Outputs_{n-1} = Inputs_n$), into a new service $S_c = \{Id_c, Capabilities_c, Outputs_c, Inputs_c\}$, where
   1. Id_c is the identifier of S_c.
   2. $Capabilities_c = Capabilities_1 \cup Capabilities_2 \cup \cdots \cup Capabilities_n$.
   3. $Outputs_c = Outputs_n$.
   4. $Inputs_c = Inputs_1$.

   Given a sequence of services S_1 to S_n, a new service S_c can be generated when any two successive services S_i and S_{i-1} have the Combinable relationship. The new service has a new Id, and the capabilities include all the capabilities of S_1 to S_n. The new service has the same input interface as the first service (S_1), and the same output interface as the final service (S_n).

Definition 5 (*Service to Content*). Given a service $S = \{Id_s, Capabilities_s, Outputs_s, Inputs_s\}$ and content $C = \{Id_c, Schema_c, Presentations_c, Semantics_c\}$, S and C have the $Produce(C, S)$ relationship iff $Schema_c = Outputs_s$ or $Translatable(Outputs_s, Schema_c)$. In other words, this definition determines if S can produce C.

Definition 6 (*Content to Service*). Given a content $C = \{Id_c, Schema_c, Presentations_c, Semantics_c\}$ and a service $S = \{Id_s, Capabilities_s, Outputs_s, Inputs_s\}$, C and S have the $ManipulatedBy(C, S)$ relationship iff $Schema_c = Inputs_s$ or $Translatable(Schema_c, Input_s)$. In other words, this relationship determines whether C can be manipulated by S.

2.3. Manipulating operations

CSIM applies manipulating operations to the above four types of 15 relationships. Manipulating operations are of two types—π *operations* and Π *operations*. π *operations* assess the relationship between the given content and service. If a specific relationship exists, the operation returns TRUE, otherwise it returns FALSE. Π *operations* return the corresponding content or services satisfying the specified relationship.

π *operations* check if the given content and service have the relationship specified in the π *operations*. A total of five π *operations* are defined:

- π_{Schemas}: Given two pieces of content, A and B, $A\ \pi_{\text{Schemas}}\ B$ refers to CIT and returns TRUE if A and B have the $InheritFrom(A, B)$ relationship.
- $\pi^{\sigma}_{\text{Schemas}}$: Given two pieces of content, A and B, $A\ \pi^{\sigma}_{\text{Schemas}}\ B$ refers to TRT and returns TRUE if A and B have the $Translatable(A, B)$ relationship.
- $\pi_{\text{Semantics}}$: Given two pieces of content, A and B, $A\ \pi_{\text{Semantics}}\ B$ refers to CST and returns TRUE if A and B have the $Identical(A, B)$ relationship.
- $\pi_{\text{Capabilities}}$: Given two services A and B, $A\ \pi_{\text{Capabilities}}\ B$ refers to SCT and returns TRUE if A and B have the $Inclusive(A, B)$ relationship.
- $\pi^{\sigma}_{\text{Capabilities}}$: Given a service A and a set of services Bs, $A\ \pi^{\sigma}_{\text{Capabilities}}\ Bs$ refers to SCT and returns TRUE if the $Inclusive(A, Bs)$ relationship holds, or one of the relationships holds: the $Combinable(A, Bs)$, $Replaceable(A, Bs)$ or $Translatable(A, Bs)$. In other words, $A\ \pi^{\sigma}_{\text{Capabilities}}\ Bs$ determines whether service A can be replaced by a series of services Bs according to one of the following four conditions.
  - Bs contain all the capabilities of A;
  - A can be combined by a set of services into Bs;
  - A can be replaced by a set of services into Bs;
  - Bs contain all the capabilities of A, but Bs also can be translated into the same input and output schemas as A.

The algorithms corresponding to the five π operations can be referenced in http://www.database.cis.nctu.edu.tw/.

Π *operations* return the content or services conforming to the specified relationship. Four categories of Π *operations* exist: Π^{c}, Π^{s}, Π^{sc} and Π^{cs}, with respect to the four types of relationships defined in Section 2.2.

Π^{c} operations return the content that conforms to the specified relationship.

- $\Pi^{c}_{\text{Translatable}}$: Given content A, $\Pi^{c}_{\text{Translatable}}$ determines the content that can be translated into A by direct or transitive translations.
- $\Pi^{c}_{\text{InheritFrom}}$: Given content A, $\Pi^{c}_{\text{InheritFrom}}$ determines the content that is inherited from A by direct or transitive inheritance.

- $\Pi^{c}_{Identical}$: Given content A, $\Pi^{c}_{Identical}$ determines the content that satisfies the *Identical* relationship with A.
- $\Pi^{c}_{Homonymous}$: Given content A, $\Pi^{c}_{Homonymous}$ determines the content that satisfies the *Homonymous* relationship with A.
- $\Pi^{c}_{Synonymous}$: Given content A, $\Pi^{c}_{Synonymous}$ determines the content that satisfies the *Synonymous* relationship with A.

Π^{s} operations return the services that exhibit the specified relationship.

- $\Pi^{s}_{Identical}$: Given a service A, $\Pi^{s}_{Identical}$ determines the services that exhibit the *Identical* relationship with A.
- $\Pi^{s}_{Inclusive}$: Given a service A, $\Pi^{s}_{Inclusive}$ determines the services that exhibit the *Inclusive* relationship with A.
- $\Pi^{s}_{Homonymous}$: Given a service A, $\Pi^{s}_{Homonymous}$ determines the services that exhibit the *Homonymous* relationship with A.
- $\Pi^{s}_{Synonymous}$: Given a service A, $\Pi^{s}_{Synonymous}$ determines the services that exhibit the *Synonymous* relationship with A.
- $\Pi^{s}_{Replaceable}$: Given a service A, $\Pi^{s}_{Replaceable}$ determines the services that exhibit the *Replaceable* relationship with A.
- $\Pi^{s}_{Translatable}$: Given a service A, $\Pi^{s}_{Translatable}$ determines the services that exhibit the *Translatable* relationship with A.
- $\Pi^{s}_{Combinable}$: Given a service A, $\Pi^{s}_{Combinable}$ determines the services that exhibit the *Combinable* relationship with A.

One Π^{sc} operation, $\Pi^{sc}_{Produce}$, is defined. Given a service S, $\Pi^{sc}_{Produce}$ returns the content that satisfies the *Produce* relationship with S.

One Π^{cs} operation, $\Pi^{cs}_{ManipulatedBy}$, is defined. Given content C, $\Pi^{cs}_{ManipulatedBy}$ returns all the services that satisfy the *ManipulatedBy* relationship with C.

The corresponding algorithms of Π operations can be referenced in http://www.database.cis.nctu.edu.tw/.

Fig. 2 depicts the architecture used to apply CSIM in DL queries. A *query interface* receives semantic queries and dispatches the queries to the *CSIM Engine*. The CSIM Engine parses the query predicates and applies the algorithms described above to solve semantic DL queries by referring to the Registry Authority. The Registry Authority contains access methods for all content and service metadata to expedite the access

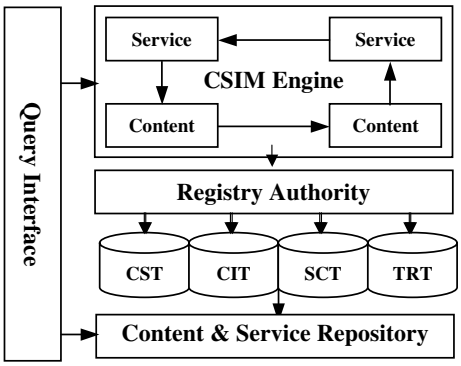

Fig. 2. CSIM in digital library.

to the metadata. Furthermore, the Registry Authority comprises a set of ontological tables. These ontological tables present a common vocabulary used for the ontological hierarchies that define content semantics and service capabilities. Index structures for *CST*, *CIT*, *SCT* and *TRT* are stored in the *Content and Service Repository* (*CSR*), to accelerate metadata retrieval. The ontology proposed herein consists of vocabulary with hierarchical structure, and an ascendant ontological concept implies all descendant concepts.

3. Semantic digital library query

Applying CSIM to digital libraries supports powerful semantic queries in content and service retrieval. In CSIM, content and service semantics can be elaborated more finely than conventional keyword-based approaches by adding relationships defined in the previous section. Using this abundant semantic information, CSIM accurately retrieves results and derives alternative answers that conventional approaches cannot do. For example, in response to a query for content with particular semantics, CSIM can retrieve the content not only in the same schema hierarchy and with identical semantics, but also in a different format, such as synonymous content. Content with various schemas, which are translatable into a single schema, can also be retrieved. Furthermore, in a semantic service query, CSIM can infer a list of recommendations to suggest that a user concatenates available services to create the desired service, using the combinable and translatable relationships.

3.1. Query language

The query language for CSIM is SQL-like. It consists of three main clauses.

1. *Select Clause*: The select clause contains the attributes of the content or service to be retrieved. The mode of attributes in a query can be set to *EXACT* or *AMBIGUOUS*. An *EXACT* query returns the attributes that exactly satisfy the given predicate without being translated or combined. For example, if one user wants to exactly retrieve the content with the data format, "Dublin Core", the query statement can be set to "*Select EXACT C.Id From Content C where C.Schema* = "*Dublin Core*"". An *AMBIGUOUS* query recommends answers that have the same semantics as those specified attributes. As in the preceding example, the query can be set to "*Select AMBIGUOUS C.Id From Content C where C.Schema* = "*Dublin Core*"". The query will return three types of answer: (1) The content with the data format "Dublin Core". (2) The content which is not in data format "Dublin Core" but can be translated into "Dublin Core" format. (3) The content which is not in data format "Dublin Core" but in the same hierarchy of "Dublin Core" in CIT. The absence of the attribute mode indicates the default query mode "EXACT".
2. *From Clause*: This clause specifies the content or service from which a user seeks. For example, a user wants to retrieve information from one piece of content C and two services S_1 and S_2.
3. *Where Clause*: This clause states conditional expressions that consist of the content or services given in the *From Clause*. In this clause, a set of Boolean operators (NOT, AND, OR), and a set of relationships defined in Section 2 are applied. For example, if a user wants to retrieve a service S_1 whose input is produced by another service S_2 and whose output data schema is Dublin Core, then the *Where Clause* is "$S_1.Input = ManipulatedBy(S_2)$ *AND* $S_1.Output =$ "*Dublin Core*"". Basically, the syntax of a conditional expression in this clause is like that of traditional SQL-like language.

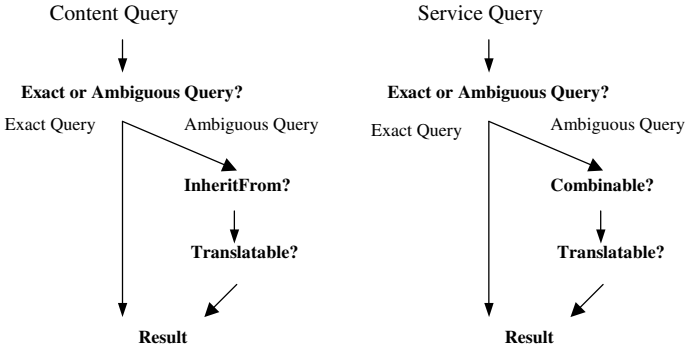

Fig. 3. CSIM semantic query.

3.2. Semantic query

Semantic queries are based on the relationships between content and services in CSIM. Semantic queries encompass two types—*EXACT query* and *AMBIGUOUS query* (Fig. 3). An *EXACT* query inquires the services or content that precisely satisfies the given predicates, without inferring other relationships. An *AMBIGUOUS* query returns the service or content that can be inferred from the translatable or combinable relationships, as well as the service or content with the same semantics as those specified attributes. Notably, an *AMBIGUOUS* query yields more results but takes more time to respond. Both content and service queries are illustrated by "Basic" and "Advanced" query. "Basic" query are in standard SQL statement and can be used in conventional query interface. "Advanced" query contain CSIM manipulation functions in their query predicate, which are able to derive more sophisticated semantic relationships.

3.2.1. Content query

A content query inquires about the content meeting the requirement specified in the query. A user can specify an *EXACT* or *AMBIGIOUS* query. An *EXACT* query returns the content that entirely satisfies the query, which means no inference is employed to obtain the result. An *AMBIGIOUS* query returns all the content that can have the same semantics specified in the query. Here, the term "can" means that the content may be translated into, or inherited from the target content.

Basic content query

Example. Determine the content with the schema of "Dublin Core".

Query Statement: *Select C.Id From Content C where C.Schema = "Dublin Core"*.

Algorithm: ContentQuery(Attributes A, Content C){

1. Locate content c. Let $c \in CSR$, and $c.Id = C.Id$, $c.Schema\ \pi_{\text{Schema}}\ C.Schema$, $c.Presentations = C.Presentations$, and $c.Semantics\ \pi_{\text{Semantics}}\ C.Semantics$;
2. If $A \in$ AMBIGUOUS, for each $r \in \Pi_{\text{Translatable}}(r, C)$, $c \leftarrow c \cup r$; (The symbol "←" indicates "assign the value".)
3. For each $k \in c$, return $k.A$.

Advanced content query

Example. Determine the content inherited from the "Dublin Core" schema.

Query Statement: *Select* $C_1.Id$ *From Content* C_1, C_2 *where* $C_2.Schema =$ "*Dublin Core*" *and* $InheritFrom(C_1, C_2)$.

Algorithm: AdvancedContentQuery(Attributes A, Contents C, Relationship R){

1. Locate content c. Let $c \in CSR$, and $c.Id = C.Id$, $c.Schema$ π_{Schema} $C.Schema$, $c.Presentations = C.Presentations$ and $c.Semantics$ $\pi_{\text{Semantics}}$ $C.Semantics$;
2. If $A \in AMBIGIOUS$, for each $r \in \Pi^{c}_{\text{Translatable}}$ $(r, C), c \leftarrow c \cup r$;
3. For each $i \in \Pi(R)$
   $c \leftarrow c \cup ContentQuery(Id, i)$;
4. For each $k \in c$, return $k.A$.

3.2.2. Service query

A service query inquires about the services meeting the requirement specified in the query. A user can specify the query to be *EXACT* or *AMBIGIOUS*. An *EXACT* query returns the services that entirely satisfy the query. An *AMBIGIOUS* query determines all the services that can have the same capabilities specified in the query. Here the term "can have" implies that the services can be concatenated or translated into the target service.

Basic service query

Example. Determine the services with the service capability "Catalog_System".

Query Statement: *Select S.Id From Service S where S.Capabilities =* "*Catalog_System*".

Algorithm: ServiceQuery(Attributes A, Services S){

1. Locate service s. Let $s \in CSR$, $s.Id = S.Id$, and $s.Capabilities$ $\pi_{\text{Capabilities}}$ $S.Capabilities$;
2. If $A \in AMBIGIOUS$, for each $r \in \Pi^{s}_{\text{Translatable}}(r, C.Schema)$, $c \leftarrow c \cup r$;
3. For each $k \in c$, return $k.A$.

Advanced service query

Example. Determine the services that have the same capabilities with "Catalog_System".

Query Statement: *Select* $S_1.Id$ *From Service* S_1, S_2 *where* $S_2.Capabilities =$ "*Catalog_System*" *and* $Inclusive(S_1, S_2)$.

Algorithm: AdvancedServiceQuery(Attributes A, Services S, Relationship R){

1. Locate service s. Let $s \in CSR$, $s.Id = S.Id$, and $s.Capabilities$ $\pi_{\text{Capabilities}}$ $S.Capabilities$;
2. If $A \in AMBIGIOUS$,
   Locate service $rs \in CSR$ where $S.Capabilities$ $\pi^{\sigma}_{\text{Capabilities}}$ $rs.Capabilities$
   $c \leftarrow c \cup rs$;
3. For each $i \in R$
   $c \leftarrow c \cup \text{ServiceQuery}(id, i)$;
4. For each $k \in c$, return $k.A$.

3.3. Ranking function

A result of a CSIM semantic query can be classified into one of the following types:

1. *Exact match*. The result conforms to the query predicate without additional translation, inheritance, or combination.
2. *Ambiguous match*. The result is a recommendation that may not completely satisfy all query predicates, but can satisfy the predicates by translating, inheriting, or combining available services or content.

Because the results of a query may not totally fulfill the user's requirements, a ranking function is proposed to evaluate the fitness of the results of a query. The ranking function W is separated into W_{Content} and W_{Service}, and defines as follows;

$$\begin{aligned} &\text{Ranking function } W_{\text{Content}}(\text{Content } A, \text{Content Results}) \\ &\quad = 1 \quad \text{if } A\ \pi_{\text{Schemas}} \text{ Results and } Num(\text{Results}) = 1 \\ &\quad = \Pi(1 - Ti) \quad \text{if } A\ \pi^{\sigma}_{\text{Schemas}} \text{ Results and } Num(\text{Results}) > 1 \end{aligned} \tag{1}$$

$$\begin{aligned} &\text{Ranking function } W_{\text{Service}}(\text{Service } A, \text{Service Results}) \\ &\quad = 1 \quad \text{if } A\ \pi_{\text{Capabilities}} \text{ Results and } Num(\text{Results}) = 1 \\ &\quad = \left(\sum W_{\text{Results}^i} * \Pi(1 - T_i)\right) \Big/ (Num(\text{Results}) - Num(T_i)) \\ &\qquad \text{if } A\ \pi^{\sigma}_{\text{Capabilities}} \text{ Results and } Num(\text{Results}) > 1 \end{aligned} \tag{2}$$

where

W_{Results^i}: $Num(\text{Results}^i.\text{Capabilities})/Num(A.\text{Capabilities}) * W^{\text{S}}_{\text{Result}^i}$;
$W^{\text{S}}_{\text{Result}^i}$: Service weight of Result^i. The larger weight represents the service is easier to compose in the result;
T_i: Overhead of the translation rules to produce Results;
Num(Results): The number of services and translation rules. The less number of Results is, the larger rank of the Result is.

Ranking functions W_{Content} and W_{Service} are proposed to rank content and services, respectively. The ranking follows the number of semantic concepts or capabilities that meet the query predicates, and the amount of content and services that are combined to yield the result.

Example. Assume a service A with five capabilities; we apply CSIM in the semantic query and obtain that A can be concatenated by three services S_1, S_2, and S_3 with two translations T_1 and T_2. Each of the three services owns three service capabilities of A, and the union of their capabilities includes all the service capabilities of A. If all of these services have $W^{\text{S}} = 1$ and the two translation rules have a 10% and 20% overhead respectively, what is the rank of this concatenation?

Result. In this example, W_{Results^1}, W_{Results^2} and W_{Results^3} are $3/5 = 0.6$. T_1 and T_2 are 0.1 and 0.2, respectively. Applying Eq. (2) yields the rank of this concatenation as $(0.6 * 1 + 0.6 * 1 + 0.6 * 1) * ((1 - 0.1) * (1 - 0.2))/(5 - 2) = 0.432$.

4. Experiments

4.1. Experimental set-up and approach

A series of experiments were performed in the digital library of National Chiao Tung University (NCTUDL, http://www.lib.nctu.edu.tw/) to demonstrate the feasibility of CSIM. These experiments involved service and content queries, which explored the structural relationship of the metadata of services and content.

The subjects of service queries were the 81 services in NCTUDL. The services were divided into six categories—Tutorial (TU), Query (QU), Service (SE), Database (DB), Journal (JO), and Holding (HO). The metadata of each service was given by experts and contained keywords that delineated their service capabilities. Experiments on service queries were conducted to compare the conventional keyword-based approach, the CSIM approach, and the CSIM approach with service inference. The keyword-based approach returned the services the description of whose capabilities exactly matched the specified keywords. The CSIM approach involved the *AMBIGUOUS* semantic query without enabling the $\Pi^{s}_{\text{Combinable}}$ manipulation function; this approach conducted a query to refer to related ontological terms for service capabilities and returned answers that would be inherited or translated into the desired services. CSIM with service inference involved the *AMBIGUOUS* semantic query with enabling the $\Pi^{s}_{\text{Combinable}}$ manipulation function. This approach returned the answer of the CSIM approach; in addition, a recommend list that could compose the required service from existing services was returned as well. For a digital library that decomposed services into reusable components, CSIM with inference advised the system developer to construct new services by combining the existing components. In this manner, a lot of development effort could be saved.

To evaluate content queries, a virtual union catalog system called VUCS@NCTUDL was developed, which harvested over 20 WebPAC systems of Taiwanese Libraries and integrated the results (Huang et al., 2000). The issue of various data schemas in the WebPAC systems complicated the mapping of schema attributes in response to a content query. To handle this issue, conventional VUCS systems involved user intervention in the design phase, or stored mapping information using metadata. This design strategy led to a much less flexible system and required user intervention in the system design. CSIM referred to the ontology stored in the Content Inheritance Table (CIT) to extend the hierarchical relationships between attributes of schemas. Furthermore, CSIM used the ontology of semantics stored in the Content Semantics Table (CST) to solve the problems of attribute semantics (such as those involving Synonymous and Homonymous relationships). The experiments retrieved four attributes (tile, subject, author, and publisher) of the content satisfying the query. The keyword-based approach returned the content contained the specified keywords within these fields (not all WebPAC systems contains all of these fields). The CSIM approach employed the ontological tables (both CST and CIT) to map various attributes of schemas into the same attribute if they were at the same level of the ontological hierarchy (in Fig. 8, ontology area). Additionally, the CSIM approach also exploited the ontological tables and TRT to convert heterogeneous attributes into the content suitable for the requested attributes. The keywords used in content queries were randomly generated by VUCS@NCTUDL. The top-k answers were calculated to average the performance in the four attributes.

4.2. Experimental metrics

Two metrics, *Accuracy* and *Coverage*, were used to evaluate both the service and content query and thereby elucidated the effectiveness of CSIM. *Accuracy* represents the effectiveness of the returned answers to be correct. *Coverage* represents the effectiveness of the returned correct answers to be included in the entire correct answers. For the service query, the *Accuracy* ($A_m(s)$) and *Coverage* ($C_m(s)$) are defined as

$$A_m(s) = (|\text{total services in } F_s| \cap |\text{total services in } F^s_{\text{ideal}}|)/|\text{total services in } F_s|$$

$$C_m(s) = (|\text{total services in } F_s| \cap |\text{total services in } F^s_{\text{ideal}}|)/|\text{total services in } F^s_{\text{ideal}}|$$

where m indicates the method to be examined, which can be keyword-based, CSIM and CSIM with inference. s represents the examined service category. In these formulas, F_s represents the services in each service category returned by the query; F^s_{ideal} represents all the services in category s, the services in concept s, and the services returned by CSIM with inference whose ranks exceed the threshold T_{service}:

$$F^s_{\text{ideal}} = \text{the services of category } s \cup \text{the services of concept } s \cup \text{the services returned by CSIM with inference whose ranks exceed } T_{\text{service}}$$

The threshold T_{service} discards the results that are cascadedly translated by too more translation rules between the input and output interfaces. The aim of T_{service} is to control the efficiency of the service query. For the content query, the *Accuracy* ($A_m(c)$) and *Coverage* ($C_m(c)$) are defined as

$$A_k(c) = (|\text{total content in } F_c| \cap |\text{total content in } F^c_{\text{ideal}}|)/|\text{total content in } F_c|$$

$$C_k(c) = (|\text{total content in } F_c| \cap |\text{total content in } F^c_{\text{ideal}}|)/|\text{total content in } F^c_{\text{ideal}}|$$

where k indicates the method to be examined, which can be keyword-based, CSIM or CSIM with inference. c represents the top-k content returned by VUCS@NCTUDL. In these formulas, F_c represents the response to a content query. F^c_{ideal} represents all the content returned by the keyword-based approach, and the content returned by CSIM whose ranks exceed the threshold T_{content}:

$$F^c_{\text{ideal}} = \text{the content returned by keyword-based approach} \cup \text{the content returned by CSIM whose ranks exceed } T_{\text{content}}$$

The threshold T_{content} discards the results that are cascadedly translated by too more translation rule between two content. The aim of T_{content} is to control the efficiency of the content query.

The percentage of the improvement of CSIM over the keyword-based approach was used to demonstrate the performance of CSIM in content query. The *Improvement* is defined as:

$$Improvement_{\text{Accuracy}} = (A_m^{\text{CSIM}}(c) - A_m^{\text{Keyword-based}}(c))/A_m^{\text{Keyword-based}}(c)$$

$$Improvement_{\text{Coverage}} = (C_m^{\text{CSIM}}(c) - C_m^{\text{Keyword-based}}(c))/C_m^{\text{Keyword-based}}(c)$$

4.3. Experiment results

Figs. 4 and 5 present the *Accuracy* and *Coverage* of the service query. Only one translation is allowed in their input/output schemas between two services. Both figures indicate that the CSIM approach outper-

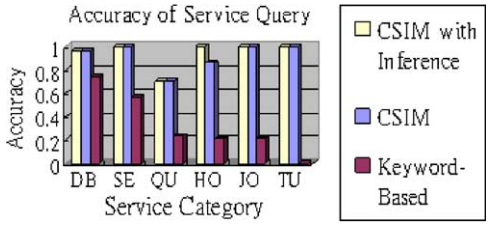

Fig. 4. *Accuracy* of service query.

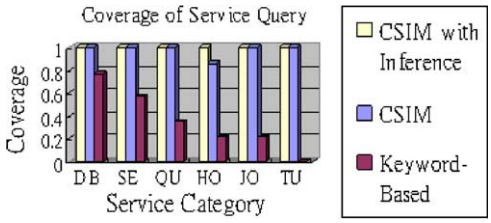

Fig. 5. *Coverage* of service query.

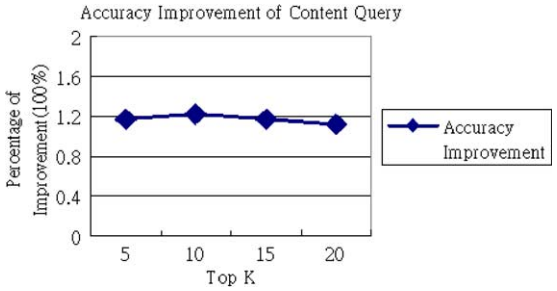

Fig. 6. *Accuracy* improvement of content query.

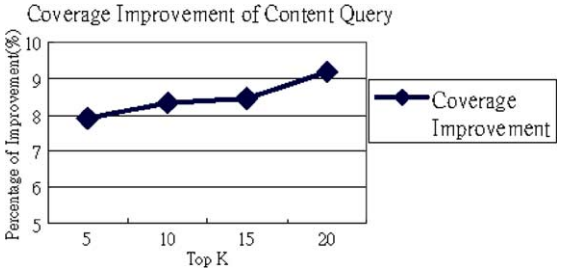

Fig. 7. *Coverage* improvement of content query.

forms the keyword-based approach. In all service categories, the keyword-based approach has poor *Accuracy* and *Coverage* because these services do not explicitly contain the searched keywords in their capabilities. The CSIM approach dramatically improves the performance with regard to both *Accuracy* and *Coverage*. This finding shows that exploiting concept approaches (like CSIM) is useful for semantic DL queries. Notably, the CSIM approach with inference outperforms pure CSIM in the "HO" category because the former approach can recommend users to combine existing services to generate additional ones such as the union of WebPAC system and the electronic Journal databases. This result may encourage libraries to spend less effort by integrating available ones on developing add-on services.

For content query, Figs. 6 and 7 plot the improvement of the average *Accuracy* and *Coverage* with respect to the queries in the four attributes (title, subject, author and publisher). The Top-K in X-axis means that the top-k books returned by VUCS are selected as the result. To avoid too many translations between two pieces of content, only one translation is allowed. In Fig. 6, CSIM increase 1.2 times performance in average than the keyword-based approach in *Accuracy* because CSIM refers to CIT and TRT to obtain more conceptually related attributes of schemas. For example, the "author" field in NCTUDL

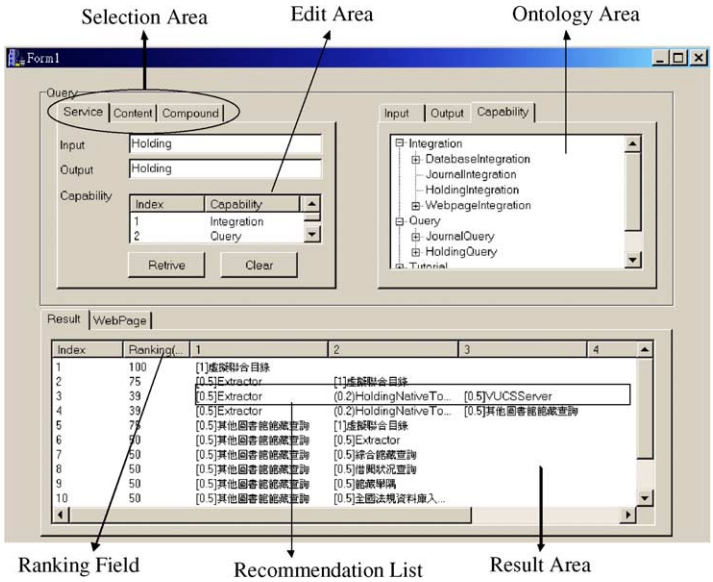

Fig. 8. CSIM@NCTU.

may appear in other WebPAC systems with different name such as creator; in this case, keyword-based approach cannot retrieve the correct results. In Fig. 7, the improvement of Coverage indicates that CSIM outperforms 8–10% to the keyword-based approach. The curve of *Coverage* improvement increases when the returned answers increase because CSIM refers to CST to return those attributes with the same semantics but different format (like Synonyms).

In summary, the performance of semantic DL queries with CSIM is highly promising. The CSIM model represents a significant improvement in both service and content queries. The CSIM model not only enhances the *Accuracy* and *Coverage* of a DL query but also suggests how librarians can integrate available components into a desired service.

4.4. Prototype system

A prototype system called CSIM@NCTUDL has been implemented. Fig. 8 illustrates the user interface of this system. This system supports advanced semantic DL queries for users to retrieve content and services in NCTUDL. Moreover, librarians can consult this system to determine reusable components in NCTUDL before they start establishing new services. CSIM@NCTUDL includes four main areas.

1. *Selection area*: This area allows the user to specify which one of the three query types is to be issued—service query, content query, and compound query. A compound query allows the user to retrieve content or services with both content and service manipulation functions.
2. *Edit area*: This area allows the user to specify the query predicates, including input/output schemas and semantics of content, and service capabilities. They are selected from the ontology area.
3. *Ontology area*: This area contains all ontological hierarchies defined in Section 2. The ontology is developed by domain experts.
4. *Result area*: This area displays the results that satisfy the predicates in the edit area. Each result is expressed as a recommendation list with one or more items. A list with one item indicates that the item

exactly matches the specified predicates; on the other hand, a list with more than one item indicates that the users can combine these items together to obtain a new content/service combination that satisfies the specified predicates. The ranking of each result is also given. A numeral in front of each service of the recommendation list denotes the fitness for the specified service capabilities. The result area presents a set of recommendation lists to advise the user; nevertheless, the system leaves the task of confirming the feasibility of the recommendation list to the user.

The example illustrates in Fig. 8 conducts a query to retrieve VUCS services. The input and output interfaces are specified as "holding" format. The "*holding*" format, which is the top item in content schema ontology, means that the service to be returned should contain any kind of library holdings. The capabilities are specified as "*Integration*" and "*Query*", each of which is the top item in one capability ontology. Consequently, the answer area shows not only the VUCS@NCTU service (Index 1 of the result area in Fig. 8), but also a recommended list to suggest another virtual union catalog system (Index 3 of the result area in Fig. 8) by combing three services: an extractor for structured documents (Huang et al., 2000), a translation service that translates native data schemas into a canonical schema (Ke, Huang, & Yang, 2001), and an integration service to combine distributed extractors.

5. Conclusions and future research

This work presents a novel content and service inference model (CSIM), which defines 15 relationships between content and services to handle semantic DL queries. It enumerates these relationships and presents manipulating functions, π and Π *operations*, to realize them. π *operations* return TRUE or FALSE for a specific relationship and Π *operations* determine all the content or services that exhibit a specific relationship. The proposed semantic DL query applies CSIM and comprises two queries. An exact query returns the answers that exactly match the predicate, and an ambiguous query returns recommendations that can be inherited, translated or combined from available content or services, as well as those that match the exact query.

CSIM was applied to the digital library of National Chiao Tung University (NCTUDL). A virtual union catalog system (VUCS@NCTUDL) and CSIM@NCTUDL was constructed (Huang et al., 2000). Experimental findings indicate that CSIM outperforms the conventional keyword-based approach in handling DL queries. An ambiguous DL query with CSIM recommends additional results beyond those of the conventional keyword-based approach, improving the *Accuracy* and *Coverage* of both content and service retrieval. Applying CSIM to digital library queries reveals that the administrative load can be reduced when new services and content are to be developed. Digital library designers can generate new content and services from those available, by considering the recommendations in response to an ambiguous query. Metadata and translation rules are applied to translate and reuse content and services to support the design of an object-oriented or component-based digital library.

CSIM can be used in DL applications. With respect to a DL resource-planning system, the inference capability of CSIM indicates that librarians should reuse available components to construct new DL services easily. Furthermore, various data fields can be categorized into semantic hierarchies to simplify the transformation between them, facilitating the combination of various data fields. Such a combination frequently arises when union systems are created. The experiments presented herein have demonstrated that CSIM outperform the conventional keyword-based approach in a virtual union catalog system.

CSIM support semantic DL queries. In most digital libraries, services are distributed on the Web, making the mapping between systems sloppy and large. Users may want to find for the services that meet their needs. Metadata can be used to describe services and CSIM applied in semantic searches to facilitate such a search. Users can specify the input type of a service (such as library holdings), output type (such as

Web pages) and capabilities (such as search systems). This semantic assignment and search bridges the gap between the aims of the user and the service capabilities. However, CSIM has an efficient problem when it generates all possible responses to an ambiguous query, especially when cascaded translations occur between different content. A properly chosen threshold T_{content} and T_{service} govern the performance of CSIM.

Future research will focus on accommodating broader semantics and developing new schemes to yield more knowledge from metadata with abundant semantics. In this manner, digital library queries can be further improved. As well as examining the optimization of operations of CSIM, our future work will develop more efficient indexing mechanisms for accessing related data structures.

Acknowledgements

The authors would like to thank the National Science Council of the Republic of China, Taiwan, for financially supporting this research under contract no. NSC90-2213-E-009-082.

References

Abiteboul, S., Benjelloun, O., & Milo, T. (2002). Web services and data integration. In *Proceedings of the third international conference on web information and system engineering* (pp. 3–6). Singapore.

Allan, J., & Raghavan, H. (2002). Using part-of-speech patterns to reduce query ambiguity. In *Proceedings of the 25th annual international ACM SIGIR conference on research and development in information retrieval* (pp. 303–314). Tampere, Finland.

Blanchi, C., & Petrone, J. (2001). Distributed interoperable metadata registry. *D-Lib Magazine, December, 7*(12). Available: http://www.dlib.org/dlib/december01/blanchi/12blanchi.html.

Chen, H. C., Chung, Y. M., Marshall, R., & Yang, C. C. (1998). An intelligent personal spider (agent) for dynamic Internet/Intranet searching. *Decision Support Systems, 23*(1), 41–58.

Dai, Y. M. (2001). *A data mining system for mining library borrowing history records.* Master Thesis of National Chiao-Tung University.

Fox, E. A., Suleman, H., & Luo, M. (2002). Building digital libraries made easy: toward open digital libraries. In *Proceedings of the fifth international conference on Asian digital libraries (ICADL2002)* (pp. 14–24). Singapore.

Gauch, S., & Wang, J. (1999). A corpus analysis approach for automatic query expansion and its extension to multiple databases. *ACM Transactions on Information System, 17*(3), 250–269.

Goncalves, M. A., & Fox, E. A. (2002). A language for declarative specification and generation of digital libraries. In *Proceedings of second joint ACM/IEEE-CS joint conference on digital libraries (JCDL'2002)* (pp. 263–272). Portland.

Goncalves, M. A., France, R. K., & Fox, E. A. (2001). MARIAN: flexible interoperability for federated digital libraries. In *Proceedings of 5th European conference of research and advanced technology for digital libraries (ECDL-01)* (pp. 173–186). Darmstadt, Germany.

Grossman, R., Qin, X., & Xu, W. (1995). An architecture for a scalable, high-performance digital library, mass storage systems. In *Proceedings of the fourteenth IEEE symposium* (pp. 11–14). Monterey, California.

Huang, S. S., Ke, H. R., & Yang, W. P. (2000). Information extraction for documents with common structure. In *The third international conference of Asian digital library (ICADL2000)* (pp. 105–112). Seoul, Korea.

Ke, H. R., Huang, S. S., & Yang, W. P. (2001). The study of interoperability of digital libraries with metadata. *University Library Journal, 5*(1), 49–78.

Kolda, T. G., & O'Leary, D. P. (1998). A semidiscrete matrix decomposition for latent semantic indexing in information retrieval. *ACM Transactions on Information Systems, 16*(4), 322–346.

Lee, K. S., Kim, D. W., Kageura, K., & Choi, K. S. (2002). A workbench for acquiring semantic information and constructing dictionary for compound noun analysis. *ICADL, Lecture Notes in Computer Science, 2555*, 315–327.

Lynch, C., & Garcia-Molina, H. (1995). *Interoperability, scaling and the digital libraries research agenda.* Information Infrastructure Technology and Applications (IITA) a Digital Libraries workshop. Available: http://www-diglib.stanford.edu/diglib/pub/reports/iita-dlw/main.html.

McCray, A. T., Gallagher, M. E., & Flannick, M. A. (1999). Extending the role of metadata in a digital library system. In *Proceedings of the IEEE research and technology advances in digital libraries, 1999 (ADL99)* (pp. 190–199). Baltimore.

Miller, G. A., Fellbaum, C., Tengi, R., Wolff, S., Wakefield, P., & Langone, H. (2002). *Wordnet: a lexical databases for the English language*. Available: >http://www.cogsci.princeton.edu/~wn/.