

The Interspace Prototype Overview: Semantic Indexing and Retrieval in a Persistent Environment

Conrad T.K. Chang, Kevin R. Powell,
Yi-Ming Chung, Qin He,
Daniel X. Pape, Les Tyrrell,
Nuala A. Bennett, Bruce R. Schatz

Community Architectures for Network Information System (CANIS)
Graduate School of Library and Information Science,
University of Illinois at Urbana-Champaign
Urbana, IL 61801, USA

Report No. UIUCDCS-R-99-2096 UIIU-ENG-99-1717

January 1999

Abstract

A research prototype is presented for semantic indexing and retrieval in Information Retrieval. The prototype is motivated by a desire to provide a more efficient and effective information retrieval system compared to the current state of the art. An overview of the Interspace architecture layers is discussed. The service layer which implements the semantic indexing and retrieval with the prototype is presented. An object model supporting semantic operations is developed. The model contains a rich set of classes and relationships and forms the basis for the conceptspace. The conceptspace is the repository for the semantic knowledge generated. Each of the four major servers in the service layer is then discussed with emphasis on the conceptspace generation server. The other servers are multimedia concept extractor, key concept assigner, and category map server. Multimedia concept extractor is used to extract concepts from various media such as text and image. Concept assigner assigns key concepts to each source document for retrieval and indexing use. Category map server performs automatic categorization and indexing on a collection of source documents. Conceptspace generation server creates the underlying knowledge representation of the source documents.

1 Introduction

One of the basic problems in information retrieval is to get a good and concise set of results for a given query on a large set of information. The effectiveness of a large-scale search in information space that uses only precise term matching is low. There are several causes. First, the query may not represent what the user really want accurately.[3, 4] Since the results depends on the set of term that a user uses, the quality of the result is therefore depending on the quality of the question. Second, the result obtained from the search is usually too large to be useful. This is because the result set contains various semantic meanings that a term might have, even though most of these meanings may not be relevant in the current context. Third, the inclusion of this large set of information results in inefficient operations in retrieval. This is especially the case when the documents are in raw texts and the search is linear. To achieve greater effectiveness from the query, semantic indexing and retrieval is needed. This is a major issue in achieving good results in information retrieval.

In this paper, we describe the design and implementation of semantic indexing and retrieval in a persistent environment, which is part of the Interspace Prototype. We present the design, algorithms, and implementation of the system. At the end, we also present the results of our experimental study using prototype of our semantic indexing and retrieval module.

The Interspace prototype is a large system that try to accomplish many goals. We will discuss the background and motivating the prototype briefly in the following section. Then, we will focus our discussion on the semantic indexing and retrieval module, which resides in one of the layer in the Interspace prototype.

2 Background and Motivation

The coming global infrastructure for asynchronous collaboration will be an abstract space of concepts derived from the underlying concrete space of objects. The few research systems that have tried to support federation across multiple information sources discovered that sharing of objects across subject domains is not sufficient to enable sharing of information across community boundaries. These research systems were able to support syntactic interoperability that enables objects to be shared (linked, grouped) across physical boundaries (hardware, software, networks). But they discovered the real problem was semantic interoperability that enables objects to be compared (correlated, matched) across logical boundaries (repositories, communities, subjects), where the representation (terminology, media, modality) is different.

The Worm Community System (WCS)[1], developed by Schatz et al demonstrated this difficulty by supporting browsing and sharing across multiple sources of data and literature for a national community of molecular biologists. It was quite possible to support syntactic interoperability to search objects and make links across multiple sources distributed across wide-area networks and other

physical boundaries. This is why WCS was frequently cited as the national model of future science collaboration systems. But semantic relationships had to be hand-coded or use special-purpose heuristics to be correct since the base representation was the raw objects. It was difficult to match, for example, the same gene across a literature abstract, gene description, physical map, sequence coding, lineage tree, developmental micrograph, and so on. The same semantic concept occurs in text, data, graphics, and images with different representations.

A global infrastructure for user-centered systems also must be built on an abstract space of concepts rather than on a concrete space of objects. Multi-mode and multi-medial information visualization requires a semantic infrastructure common across subject domains above the syntactic infrastructure that provides distributed computing. Thus an environment to manipulate the common semantics is needed – information middleware for generation, retrieval, and categorization. The ARPA Tipster projects demonstrated that semantic interoperability could be achieved for large collections but only by using domain-specific heuristics. Tipster achieved scalable semantics in number of objects but not in number of subjects.

WCS, while experiencing much the same fragility as Tipster, hinted at a general technique for domain-independent semantic representation called *Concept Spaces*. This was aimed at a lower semantic level than traditional artificial intelligence techniques, being closer to the statistical computations found in information retrieval for text analysis. Initially, the concept space research concentrated on the context within documents by generating word co-occurrence matrices. This proved successful in the small biology domain for augmenting search performance – it was not an automatic intelligent agent but an interactive advisor suggesting alternative search terms.

3 Overview of the Interspace Research Project

The Interspace is a system that seeks to unify disparate distributed information resources in one coherent model. The Interspace, as an entity, is a collection of interlinked information spaces where each component space encodes the knowledge of a community or a subject domain. An information space is a collection of interlinked objects. The basic abstraction of this model is a special object called an Information Unit (IU); this serves as the fundamental unit of organization in the Interspace system. Using IUs as a basis, the system also provides a rich set of support for complex inter-operable applications. Standard services include inter-object linking, remote execution, object caching, and support for compound objects (usually referred to as compound documents.) Additionally the Interspace system acknowledges the importance of legacy applications and provides support for integrating them into the system in a relatively seamless manner. Ultimately, the Interspace system is an attempt to represent all types data/objects in one flexible, cohesive, and scalable system.

The Interspace system uses a layered system design. See figure 1. The kernel contains the generic services needed for all information applications and

Interspace Architecture Diagram

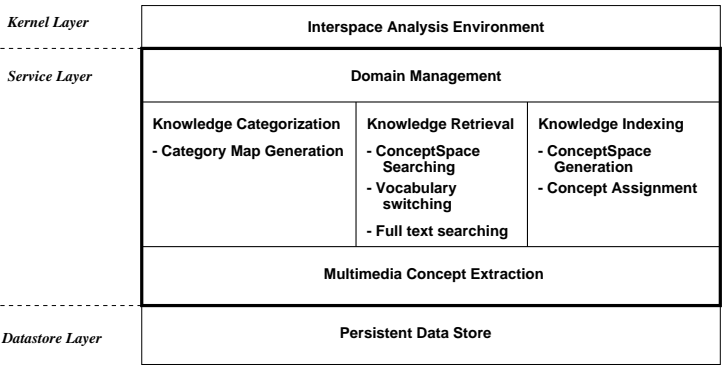


Figure 1: Interspace system architecture

the basic functionality of the information space and IUs. Under the kernel is the service layer. This layer contains the core functionalities required by the kernel. These functions are provided by various servers residing in this layer. Some of the functionalities include semantic indexing and retrieval, full text searching and vocabulary switching. In the future, additional functionalities may be added to this layer as needs arise. For example, we are planning to add path matching in the future. Finally, the bottom layer consists of a distributed persistent data layer implemented by using a persistent object database system.

4 Building Support for Semantic Indexing and Retrieval

Semantic indexing and retrieval is provided by several servers in the service layer of the Interspace prototype. There are currently four major servers: concept assignment servers, conceptspace generation server, multimedia concept extraction server, and category map server. Other supporting servers also exist. In general, the servers can be classified into four broad categories: concept extraction, knowledge retrieval, knowledge indexing, knowledge categorization. Concept extraction creates concepts that represent the underlying information. Extracted concepts form conceptspaces. These conceptspaces are the central knowledge repositories for the rest of the servers. Their design and implementation are an important part in the whole process and will be discussed in the following sections. The knowledge retrieval deals with getting information out

listed below:

- **SourceUnit:** This is the abstract class of a basic unit of information source in the model. In general, an object instance would be created from the subclasses of this class. Currently, there are two subclasses: TextSU and MapSu. Each sourceunit has a specific SUPolicy and a SUSetting. SUPolicy specifies the type of SourceUnit. SUSetting specifies the quality of service that should be used in the multimedia concept extraction phase. A SourceUnit also contains a list of concepts associated with it.
- **TextSU:** This is a subclass of SourceUnit. It specifies the information source is of textual type.
- **MapSU:** This is a subclass of SourceUnit. It specifies the information source is of image type.
- **AudioSU:** This is a subclass of SourceUnit. It specifies the information source is of audio type.
- **VideoSU:** This is a subclass of SourceUnit. It specifies the information source is of video type.
- **SUPolicy:** SUPolicy specify the media parser for a source unit. Depending on the types of the SourceUnit, a different could be used. A HTML SourceUnit would need a HTML media parser for instance. In addition, the SUPolicy also specify the parts of SourceUnit what are useful for concept extraction and computation.
- **HTMLPolicy:** This is a subclass of SUPolicy . It specifies settings for HTML type of SourceUnit.
- **RawTextPolicy:** This is a subclass of SUPolicy . It specifies settings for raw text type of SourceUnit. Raw text is basic any SourceUnit without a type.
- **SUSetting:** SUSetting specify the quality of service for concept extraction. Concept extraction requires intensive computation. This setting provides a means to choose the appropriate tradeoff between computation time and quality of results.
- **Domain:** Domain is the connectionpoint between various servers in the service layer and the kernel. Its function is to group sourceunits into similar domain area. For example, we might have a domain for Cancer. This domain would then contain a collection of sourceunits which are about cancer. In addition, the domain contains domain specific information about concepts extracted from each sourceunit. This information is stored in ConceptInDomain. An example would be statistics such as cooccurrence of concepts in sourceunits, which are needed in the conceptspace generation. Each domain is also associated with a single conceptspace.

In addition, a domain can contain multiple subdomain. For this reason, domain objects can be used to form multiple hierarchies that can be used for categorization purpose.

- **Concept:** Concept represents a piece of semantic meanings. Each concept is unique and can appear in many domains and conceptspaces. Also, it does not contain any computational information. This information is contained in `ConceptInCS` and `ConceptInDomain` classes.
- **Representation:** This is an abstract class which represent the appearance of a concept. It has many subclasses, each of which is a different types of representation. It is possible that a concept can have multiple representations. For example, the concept “dog” could be represented by the string ‘dog’, or a picture of dog. Both of these are representations of the same idea.
- **StringRep:** This is a subclass of `Representation`. It specifies textual representation.
- **VideoRep:** This is a subclass of `Representation`. It specifies video sequence representation.
- **AudioRep:** This is a subclass of `Representation`. It specifies audio sequence representation.
- **MapRep:** This is a subclass of `Representation`. It specifies Image map representation.
- **ConceptSpace:** This class contains a collection of concepts extracted from a list of sourceunits. This is the basic unit of analysis and computation when the conceptspace algorithm is applied.
- **ConceptInCS:** This is the computation part of a concept in a `ConceptSpace`. Since each concept can appear in multiple conceptspaces, information related to a concept with respect to a particular conceptspace is stored in this class. This information includes relationships with other concepts in this conceptspace.
- **ConceptInDomain:** This is the computation part of a concept in a `Domain`. Since each concept can appear in multiple domains, information related to a concept with respect to a particular domain is stored in this class. This information includes relationships with other concepts in this domain.
- **Cooccurrence:** This class contains information about one specific type of relationships that concepts might have with each others, namely the cooccurrence matrix. The cooccurrence matrix store information about whether concepts co-appear with each other in a particular sourceunit.

4.2 Semantic Indexing and Retrieval Servers Design

As stated in previous sections, there are four main servers that provide the semantic indexing and retrieval functionality in the service layer. In this paper, we will briefly discuss these servers: conceptspace generation, multimedia concept extraction server, concept assignment server, and category map server.

4.2.1 ConceptSpace Generation Server

The automatic generation of thesauri represents an area of growing importance in the field of computational science. Developed under the auspices of the federal NII Digital Library Initiative (DLI) at the Universities of Illinois and Arizona [6], *Concept Space* thesauri are based on a hybrid symbolic/numeric computation which determines relationships between concepts in a collection of sourceunits. The resulting map between concepts is designated a *Concept Space* and is useful in the refinement of queries presented to the collection. Concept Spaces are used, for example, in interactive query sessions as part of the DLI testbed at the University of Illinois, Urbana-Champaign [5]. Algorithms to perform iterative search refinement which incorporate the computation of Concept Spaces are also under development at the CANIS¹ Lab in Illinois.

4.2.2 Multimedia Concept Extraction Server

The main service provided by this server is to extract concepts from various types of source units. Type of information source could be text, image, audio or video. Concepts are extracted from sourceunits by a series of complex processing steps. In general, these steps might include media feature extraction, image analysis, pruning, parsing, noun phrase extraction and similarity matching.

4.2.3 Concept Assignment Server

The primary functionality of Concepts Assigner server is to produce a set of concepts (or subject descriptors) which are the “most” descriptive of the sourceunit. These concepts are generated from a particular domain and a pregenerated concept space from the same domain or related domain(s) and represent key concepts in the sourceunit. The assigner uses the Hopfield network to compute the assignment.

The Hopfield network [2, 7] was introduced as a neural net that can be viewed as nonlinear associative memory or content-addressable memory. Knowledge and information can be stored in single-layered interconnected neurons (nodes) as fundamental memories, representing the patterns to be memorized by the network. This information can then be retrieved based on the network’s parallel relaxation function—nodes, activated in parallel, and traversed until the network reaches a stable state. The concept spaces serve as a trained network in this application. The concepts represent nodes (concepts) of the network and weights

¹Community Architectures for Network Information Systems research lab in University of Illinois at Urbana-Champaign

represent synaptic weights between any two nodes. Asymmetric link weights represented similarity between two concepts. The synaptic weights remained fixed after they had been assigned. Key concepts are then calculated from the network and are assigned to each sourceunit.

4.2.4 Category Map Server

The category map server attempts to categorize the concepts for a group of source units received from the MCE into a hierarchical set of "category maps." It does this by feeding the above concepts into another special kind of neural network called a self-organizing map (SOM). A SOM is a neural network algorithm introduced by Teuvo Kohonen in 1982. It is used to visualize and interpret large high-dimensional data sets. It does this by mapping n-dimensional data points that are similar to each other onto nearby regions of a q-dimensional space; q is usually much smaller than n, and is also usually 2.

The category map server takes these two dimensional mappings of the n dimensional space of concepts, and transforms it into a three dimensional terrain which can be "flown" through by the user in order to investigate the mappings of concepts in relation to each other and the rest of the terrain.

5 Future Works

There are many issues that are still under researched in the current Interspace prototype. Some of our planned future works include parallel implementation of the servers on multiprocessor systems. This would reduce the computation time significantly enough that real-time interactive usage of these servers will be feasible. Moreover, we are working on an automatic validation method that would provide quantitative measurement of the goodness of our conceptspace generation algorithms. This type of measurement is important in our effort to optimize the algorithms. In addition, we also plan to make use of distributed services like CORBA to convert the current system to a distributed system. Finally, we are planning to add significant improvement to the persistent data store layer. We foresee that this layer would eventually become a bottleneck in the future distributed and parallel version. One major improvement planned is to add a multilevel locking mechanism depending on access patterns. This would increase concurrency access and reduce contention of underlying data.

References

- [1] Hsinchun Chen, et. al., and Bruce Schatz. A Concept Space Approach to Addressing the Vocabulary Problem in Scientific Information Retrieval: An Experiment on the Worm Community System. *Journal American Society for Information Science (JASIS)*, 48(1):17-31, Jan 1997.

- [2] J. J. Hopfield. Neural network and physical systems with collective computational abilities. *Proceedings of the National Academy of Science, USA*, 79(4):2554–2558, 1982.
- [3] B. Schatz, E. Johnson, and P. Cochrane. Interactive term suggestion for users of digital libraries: Using subject thesauri and co-occurrence lists for information retrieval. In E. Fox and G. Marchionini, editors, *Proceedings of the 1st ACM International Conference on Digital Libraries*, pages 126–133. ACM and SIGIR and SIGLINK, ACM, Mar 1996.
- [4] B. R. Schatz. Information retrieval in digital libraries: Bringing search to the net. *Science*, 275:327–334, January 1997.
- [5] B. R. Schatz, E. Johnson, P. Cochrane, and H. Chen. Interactive Term Suggestion for Users of Digital Libraries: Using Subject Thesauri and Co-occurrence Lists for Information Retrieval. In *1st International ACM Conference on Digital Libraries*, pages 126–133, Bethesda, MD, 1996. ACM.
- [6] Bruce R. Schatz and Hsinchun Chen. Building Large-Scale Digital Libraries. *IEEE Computer*, 29(5):22–26, May 1996.
- [7] D. W. Tank and J. J. Hopfield. Collective computation in neuronlike circuits. *Scientific American*, 257(6):104–114, December 1987.